Privacy-Preserving Filter-based Feature Selection with Secure Multiparty Computation

Xiling Li

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

Dr. Rafael Dowsley (Monash University)

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

Privacy-Preserving Filter-based Feature Selection with
Secure Multiparty Computation

Xiling Li

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

The data pre-processing stage with steps of data cleaning (handling of missing/noisy data, dealing with outliers), data transformation (normalization, discretization, and rebalancing), and data reduction (feature extraction/selection) is crucial for the machine learning workflow. Existing work on privacy-preserving machine learning (PPML) with Secure Multiparty Computation (MPC) is almost exclusively focused on model training and on inference with trained models, thereby overlooking the important data pre-processing stage. In this work, we propose an MPC based protocol $\pi_{\mathsf{FILTER-FS}}$ for private feature selection based on the filter method. It is independent of model training, and can be used in combination with any MPC protocol to rank features. For ranking of features in a privacy-preserving manner, we propose a feature scoring protocol $\pi_{\mathsf{MS-GINI}}$ based on Gini impurity. The computation of a Gini score for continuous valued features traditionally requires sorting of the feature values to determine candidate split points in the feature value range. As sorting is an expensive operation to perform in a privacy-preserving way, we instead propose a "mean-split Gini score" (MS-GINI) that avoids the need for sorting by selecting the mean of the feature values as the split point. Feature selection with MS-GINI leads to accuracy improvements that are on par with those obtained with the traditional Gini score in the data sets used in our experiments.

To demonstrate the feasibility of our approach for practical data science, we propose a protocol $\pi_{\mathsf{GINI-FS}}$, which combines $\pi_{\mathsf{FILTER-FS}}$ with $\pi_{\mathsf{MS-GINI}}$, and perform experiments with the proposed MPC protocols for feature selection in a commonly used machine-learning-as-a-service configuration where computations are outsourced to three servers (3PC), with semi-honest and malicious adversaries. Regarding effectiveness, we show that secure feature selection with the proposed protocols improves the accuracy of classifiers on a variety of real-world data sets, without leaking information about the feature values or even which features were selected. Regarding efficiency, we document runtimes between ≈48 sec and ≈25 hours for our protocols to finish, depending on the size of the data set, the security settings, and the hardware configuration.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I want to appreciate my advisor, Dr. Martine De Cock. She taught me a lot and guided me into the world of research. I also want to appreciate Dr. Anderson Nascimento from UW and Dr. Rafael Dowsley from Monash University to be in my reading committee. They provided many useful and helpful suggestions.

DEDICATION

To my family and friends

Chapter 1
INTRODUCTION

Machine learning (ML) thrives because of the availability of an abundant amount of data, and the development of computational resources and devices to collect and process such data. In many successful ML applications, the data that is consumed during ML model training and inference is of a very personal nature. Privacy protection of user data has become an urgent and serious concern for ML development, prominently so in social network applications[1] and applications based on biomedical data[2], to name a few. One direction towards safeguarding the privacy of user data is through laws and regulations, such as the European General Data Protection Regulation (GDPR)[3] and the California Customer Privacy Act (CCPA)[4]. In this work, we follow another direction, namely the development of cryptographic protocols that allow computations on encrypted data. In this way, we contribute to the field of privacy-preserving machine learning (PPML), a burgeoning and interdisciplinary research of cryptography and ML, that has gained significant traction in tackling privacy issues.

In particular, we use techniques from Secure Multiparty Computation (MPC), an umbrella term for cryptographic approaches that allow two or more parties to jointly compute a specified output from their private information in a distributed fashion, without actually revealing the private information on parties' inputs beyond what can be computed from the output itself [13]. We consider the scenario where different data owners or enterprises are

[1]`https://www.facebook.com/notes/2420600258234172/`

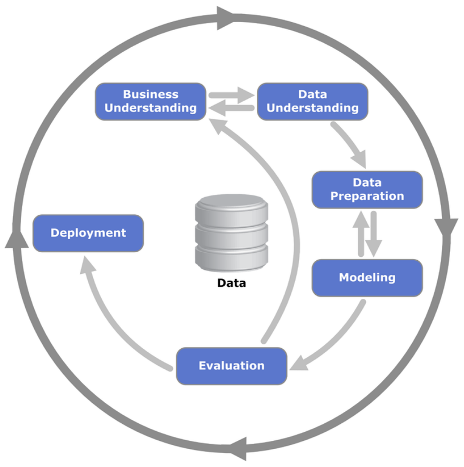
[2]`http://www.humangenomeprivacy.org/2019/`

[3]`https://en.wikipedia.org/wiki/General_Data_Protection_Regulation`

[4]`https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act`

interested in training a ML model over their combined data. There is a lot of potential in training ML models over the aggregated data from multiple enterprises. First of all, training on more data typically yields higher quality ML models. For instance, one could train a more accurate model to predict the length of hospital stay of COVID-19 patients when combining data from multiple clinics. This is an application where the data is *horizontally distributed*, meaning that each data owner or enterprise has records/rows of the data. Furthermore, being able to combine different data sets enables new applications that pool together data from multiple enterprises, or even from different entities within the same enterprise. An example of this would be a ML model that relies on lab test results as well as healthcare bill payment information about patients, which are usually managed by different departments within a hospital system. This is an example of an application where the data is *vertically distributed*, i.e. each data owner has their own columns. While there are clear advantages to training ML models over data that is distributed across multiple data owners, often these data owners do not want to disclose their data to each other, because the data in itself constitutes a competitive advantage, or because the data owners need to comply with data privacy regulations. These roadblocks can even affect different departments within the same enterprise, such as different clinics within a healthcare system.

Figure 1.1 shows the cross-industry platform for data mining (CRISP-DM) workflow [40], which breaks the ML pipeline down in six major stages. More details about three of the stages are shown on the right, namely data pre-processing (data preparation), modeling and evaluation. During the last decade, cryptographic protocols designed with MPC have been developed for training of ML models over aggregated data, without the need for the individual data owners or enterprises to reveal their data to anyone in an unencrypted manner. This existing work includes MPC protocols for training of decision tree models [32, 18, 12, 1, 16], linear models [36, 14, 2, 15], as well as neural network architectures [35, 3, 17, 43, 26]. Existing work on PPML with MPC is focused almost exclusively on the modeling stage, and assumes that the data sets are pre-processed and clean, with features that have been pre-selected and constructed. In practical data science projects, model building constitutes



data preparation	handling missing values outlier detection normalization discretization feature extraction **feature selection (wrapper, filter, and embedded methods)**
modeling	linear regression decision trees random forest boosted decision trees neural networks ...
evaluation	k-fold cross-validation root mean squared error accuracy

Figure 1.1: Left: Cross-industry standard platform for data mining (CRISP-DM), which breaks the process of data mining down into six major phases (picture from Wikipedia). Our focus is on feature selection in the data preparation stage. Previous work on PPML has focused almost exclusively on the *modeling* phase.

only a small part of the workflow: Real-world data sets must be cleaned and pre-processed, outliers must be removed, training features must be selected, and missing values need to be addressed before model training can begin. Data scientists are estimated to spend 50% to 80% of their time on data wrangling as opposed to model training itself [33]. PPML solutions will not be adopted in practice if they do not encompass these *data preparation* steps. Indeed, there is little point in preserving the privacy of clean data sets during model training – which is currently already possible – if the raw data has to be leaked first to arrive at those clean data sets!

In this work, we contribute to filling this gap in the open literature by proposing *a protocol* $\pi_{\mathsf{FILTER-FS}}$ *for privacy-preserving feature selection.* Feature selection is the process of selecting a subset of relevant features from the original data set to perform as good as – or even better than – using all features directly for model training [11]. Using a well-chosen subset of features can lead to more accurate models, as well as efficiency gains during model training. A commonly used technique for feature selection is the so-called *filter method* in which features are ranked according to a score indicative of their predictive ability, and subsequently the highest ranked features are retained. However, the popularity of the filter method stems from the fact that it is computationally very efficient, and that it is independent of the ML model architecture. This sets the filter method apart from wrapper and embedded feature selection methods in which the feature selection and modeling stages are intertwined, resulting in a higher computational cost.

The MPC based protocol $\pi_{\mathsf{FILTER-FS}}$ for private feature selection that we propose in this work can be used in combination with any MPC protocol to rank features in a privacy-preserving manner. Well-known techniques to score features in terms of their informativeness include mutual information (MI), Gini impurity (GI), and Pearson's correlation coefficient (PCC). We propose a new measure called *Mean-Split Gini Impurity (MS-GINI)* in Section 4.1 based on GI, and an efficient feature scoring protocol $\pi_{\mathsf{MS-GINI}}$ to compute the MS-GINI score of features in a privacy-preserving manner, leaving the development of secure protocols for other feature scoring techniques as future work. The computation of the traditional GI score for continuous valued features requires sorting of the feature values to determine candidate split points in the feature value range. As sorting is an expensive operation to perform in a privacy-preserving way, we instead propose a *MS-GINI score* that avoids the need for sorting by selecting the mean of the feature values as the split point. As we show in Chapter 5, feature selection with MS-GINI leads to accuracy improvements that are on par with those obtained with GI, PCC, and MI in the data sets used in our experiments. Depending on the application and the data set at hand, one may want to use a different feature scoring technique, in combination with our protocol $\pi_{\mathsf{FILTER-FS}}$ for private feature selection.

Figure 1.2 illustrates the flow of private feature selection and subsequent model training at a high level in an outsourced "machine-learning-as-a-service" setting with three computing

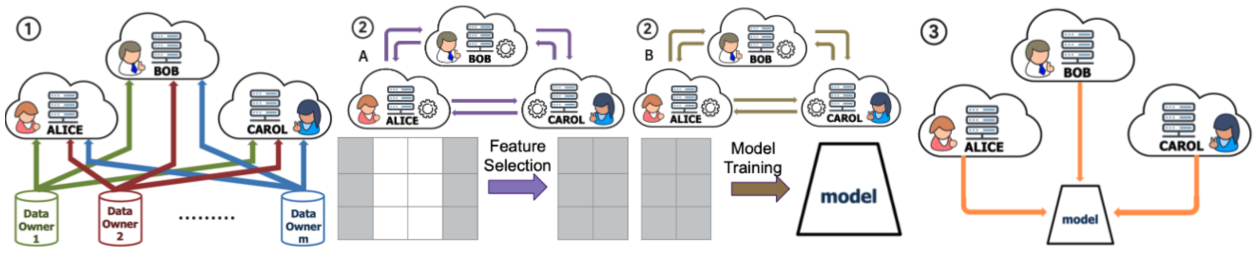


Figure 1.2: Overview of private feature selection and model training in 3PC configuration with computing servers (parties) *Alice*, *Bob*, and *Carol*.

servers, nicknamed *Alice*, *Bob*, and *Carol* (three-party computation, 3PC). 3PC with honest majority, i.e. with at most one server being corrupted, is a configuration that is often used in MPC because this setup allows for some of most efficient MPC schemes. In Step 1 of Figure 1.2, each of m data owners sends secret shares of their data to the three servers (parties). While the secret shared data can be trivially revealed by combining shares, no information about the data is revealed by the shares received by any single server, meaning that none of the servers by themselves learn anything about the actual values of the data. In Step 2A, the three servers execute protocols $\pi_{\mathsf{MS-GINI}}$ and $\pi_{\mathsf{FILTER-FS}}$ to create a reduced version of the data set that contains only the selected features. Throughout this process, none of the parties learns the values of the data or even which features are selected, as all computations are done over secret shares. Next, in Step 2B, the parties train a ML model over the pre-processed data using existing privacy-preserving training protocols, e.g., a privacy-preserving protocol for logistic regression training [17]. Finally, in Step 3, the servers can disclose the trained model to the intended model owner by revealing their shares. Steps 1 and 3 are trivial as they follow directly from the choice of the underlying MPC scheme (see Section 2.2). MPC protocols for Step 2B have previously been proposed. The focus of this thesis is on Step 2A. Our approach works in scenarios where the data is horizontally partitioned (each data owner has one or more of the rows or instances), scenarios where the data is vertically partitioned (each data owner has some of the columns or attributes), or any other partition.

This thesis is organized as follows. In Chapter 2, we recall preliminaries about feature selection and MPC. In Chapter 3, we discuss related work. In Chapter 4, we present details about the proposed MS-GINI feature scoring technique, as well as the MPC protocols $\pi_{\mathsf{FILTER-FS}}$, $\pi_{\mathsf{MS-GINI}}$, and $\pi_{\mathsf{GINI-FS}}$ for privacy-preserving feature selection, along with concrete examples to illustrate the methodology. We implemented our protocols in MP-SPDZ[5], an open source framework for MPC that includes many existing MPC primitives [29]. In Chapter 5, we present experimental results obtained with this implementation of our MPC protocols, in terms of accuracy and runtime, on a variety of data sets. In Chapter 6, we conclude our work and present some ideas for future research.

We first proposed the MS-GINI score and analyzed its utility for feature selection in an application on cognitive load detection from wrist-band sensor measurements, as described in the following publication:

> **Xiling Li** and Martine De Cock. *Cognitive load detection from wrist-band sensors.* In Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers (UbiComp-ISWC '20), p. 456–461. DOI:https://doi.org/10.1145/3410530.3414428

[5]https://github.com/data61/MP-SPDZ

Chapter 2
PRELIMINARIES

2.1 Feature Selection

2.1.1 Introduction

Feature selection is a step of the data pre-processing stage of the machine learning (ML) workflow. It is a technique of data reduction similar to feature extraction. The aim of feature selection is to form a subset with relevant features of a data set instead of using all features, while feature extraction is a way to create new features from raw data [30].

There are several reasons data scientists do feature selection before training a ML model:

- Decrease training time
- Increase accuracy
- Improve model interpretability [34]

To achieve those goals, the selected feature subset should also act as good as – or even better than – the original one to construct a well-performing ML model [27].

2.1.2 Approaches to Feature Selection

For methods of feature selection, there are mainly three approaches: wrapper, filter, and embedded approaches [11].

The *wrapper approach* works in a greedy fashion by picking several features or removing several features from all features to form a temporal subset for each round. Each round of the feature selection algorithm trains and tests ML models on the temporarily selected features.

Eventually, the subset with the best performance among the subsets of all rounds becomes the final selected one. Typically, the wrapper approach uses sequential forward selection, backward elimination and/or a genetic algorithm.

The *filter approach* uses statistical criteria to score or rank all features and selects the most discriminative features from raw data. Generally, filter methods perform feature selection before ML model training and usually fall into a two-step strategy. First, all features are ranked according to certain criteria. Then, the features with the best rankings are selected. Those selected features will form the subset after feature selection for the modeling stage. Particularly, criteria like Pearson correlation coefficient (PCC), mutual information (MI) and Gini impurity (GI) (Section 2.1.3) can be used in the filter approach.

The *embedded approach* takes a combination of both aforementioned approaches by applying a wrapper approach initially and using a filter approach to pick a final subset.

In this thesis, we use the filter approach. This approach has less computational cost than the wrapper and embedded approaches, and it can also be done before the model training stage, which makes it possible for us to design secure protocols decoupled from operations of ML model training and testing stages.

In the filter approach, we select features based on a variant of GI called *Mean-Split Gini Impurity (MS-GINI)* (Section 4.1). As the accuracy results in Table 5.1 of Chapter 5 show, MS-GINI can perform as well as other well-known methods (e.g. PCC, MI and GI) from the filter approach. Furthermore, it can be computed more efficiently in a privacy-preserving manner.

2.1.3 Gini Impurity (GI)

In the filter approach to feature selection, all features are first assigned a score that is indicative of their predictive ability. Subsequently only the best scoring features are retained. A well-known feature scoring criterion is Gini impurity (GI), made popular as part of the classification and regression tree algorithm (CART) [8].

Assume that we have a set S of m training examples, where each training example consists of an input feature vector $(x_1, \ldots, x_p)$ and a corresponding label y. Throughout this thesis, we assume that there are n possible class labels. The goal of supervised ML is to induce a ML model from this training data that can infer, for a previously unseen input feature vector, a label y as accurately as possible. Not all p features may be equally beneficial to this end.

If the j^{th} feature F_j is a discrete feature that can assume ℓ different values, then it induces a partition $S_1 \cup S_2 \cup \ldots \cup S_\ell$ of S in which S_i is the set of instances that have the i^{th} value for the j^{th} feature. The Gini impurity of S_i is defined as:

$$G(S_i) = \sum_{c=1}^{n} p_c \cdot (1 - p_c) = 1 - \sum_{c=1}^{n} p_c^2 \tag{2.1}$$

where p_c is the probability of a randomly selected instance from S_i belonging to the c^{th} class. The Gini score of feature F_j is a weighted average of the Gini impurities of the S_i's:

$$G(F_j) = \sum_{i=1}^{\ell} \frac{|S_i|}{|S|} \cdot G(S_i) \tag{2.2}$$

Conceptually, $G(F_j)$ estimates the likelihood of a randomly selected instance to be misclassified based on knowledge of the value of the j^{th} feature. During feature selection, the k features with the lowest Gini scores are retained.

If F_j is a feature with continuous values, then $G(F_j)$ is defined as the weighted average of the Gini impurities of a set $S_{\leq\theta}$ containing all instances for which the j^{th} feature value is smaller than or equal to θ, and a set $S_{>\theta}$ with all instances for which the j^{th} feature value is larger than θ. In the CART algorithm, an optimal threshold θ is determined based on sorting of all the instances on their feature values. Since privacy-preserving sorting is a time-consuming operation in MPC [7, 25], in Section 4.1, we propose a more straightforward approach for threshold selection which, as we show in Table 5.1 of Chapter 5, yields desirable improvements in accuracy.

2.2 Secure Multiparty Computation

2.2.1 Introduction

Secure Multiparty Computation (MPC) originated in the 1980s from Yao's millionaire problem [44], inspired by mental poker [39] and one-way functions [19]. In this problem, there are two millionaires – Alice and Bob. They want to know who is richer, but they do not want to reveal their wealth to each other. This is a prototypical secure two-party computation problem (2PC), and Alice/Bob are typical names for two parties who need to jointly execute a protocol to solve the problem.

MPC has become a state-of-the-art subfield of cryptography [13], and has been extended from computations involving two parties to computations that can involve any number of parties $P_1, \ldots, P_n$. Each party P_i holds a secret input x_i, and the parties agree on some function f that takes n inputs. Their goal is to compute $y = f(x_1, ..., x_n)$ while making sure that the following two conditions are satisfied:

- Correctness: the correct value of y is computed; and
- Privacy: y is the only new information that is released. (Each party does not know any information other than what can be inferred from y and its own input.)

Protocols for MPC in other words enable a set of parties to jointly compute the output of a function over each of the parties' private inputs, without requiring parties to disclose their input to anyone. MPC is concerned with the protocol execution coming under attack by an adversary which may corrupt parties to learn private information or cause the result of the computation to be incorrect. MPC protocols are designed to prevent such attacks being successful, and use proven cryptographic techniques to guarantee privacy.

2.2.2 Adversarial Model

In this work, we consider both semi-honest as well as malicious adversaries (definitions and related security are shown in Section 2.3.2 and 2.3.3 of [21]). While parties corrupted by semi-honest adversaries follow the protocol instructions correctly but try to obtain additional information, parties corrupted by malicious adversaries can deviate from the protocol instructions (static corruption). We consider the three-party secure computation (3PC) setting; the adversary can corrupt at most one server. Because this setting allows for efficient protocols, the honest-majority 3PC setting is growing in popularity in the PPML literature (e.g. [6, 38, 43]).

2.2.3 Replicated Secret Sharing

In our solution, all computations by the parties (servers) are done over integers in a ring $\mathbb{Z}_q$ with addition ($+$) and multiplication ($\cdot$) operations. Raw data in ML applications is often real-valued. As is common in the MPC literature, we convert real numbers to integers using a fixed-point representation [10]. After this conversion, in the *semi-honest* adversary setting, the data owners secret share their values with the parties using a replicated secret sharing scheme [4]. To share a secret value $x \in \mathbb{Z}_q$ among parties P_1, P_2, and P_3, the shares x_1, x_2, x_3 are chosen uniformly at random in $\mathbb{Z}_q$ with the constraint that $x_1 + x_2 + x_3 = x \mod q$. P_1 receives x_1 and x_2, P_2 receives x_2 and x_3, and P_3 receives x_3 and x_1. Note that it is necessary to combine the shares available to two parties in order to recover x, and no information about the secret shared value x is revealed to any single party. For short, we denote this secret sharing by $[\![x]\!]_q$. Let $[\![x]\!]_q$, $[\![y]\!]_q$ be secret shared values and c be a constant, the following computations can be done locally by parties without communication:

- Addition ($z = x + y$): Each party P_i gets shares of z by computing $z_i = x_i + y_i$ and $z_{(i \bmod 3)+1} = x_{(i \bmod 3)+1} + y_{(i \bmod 3)+1}$. This is denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q + [\![y]\!]_q$.
- Subtraction $[\![z]\!]_q \leftarrow [\![x]\!]_q - [\![y]\!]_q$ is performed analogously.
- Multiplication by a constant ($z = c \cdot x$): Each party multiplies its local shares of x by c to obtain shares of z. This is denoted by $[\![z]\!]_q \leftarrow c \cdot [\![x]\!]_q$
- Addition of a constant ($z = x + c$): P_1 and P_3 add c to their share x_1 of x to obtain z_1, while the parties set $z_2 = x_2$ and $z_3 = x_3$. This will be denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q + c$.

The main advantage of replicated secret sharing compared to other secret sharing schemes is that replicated shares enables a very efficient procedure for multiplying secret shared values. To compute $x \cdot y = (x_1 + x_2 + x_3) \cdot (y_1 + y_2 + y_3)$, the parties locally perform the following computations: P_1 computes $z_1 = x_1 \cdot y_1 + x_1 \cdot y_2 + x_2 \cdot y_1$, P_2 computes $z_2 = x_2 \cdot y_2 + x_2 \cdot y_3 + x_3 \cdot y_2$, and P_3 computes $z_3 = x_3 \cdot y_3 + x_3 \cdot y_1 + x_1 \cdot y_3$. By doing so, without any interaction, each P_i obtains z_i such that $z_1 + z_2 + z_3 = x \cdot y \mod q$. After that, the parties are required to convert from this additive secret sharing representation back to the original replicated secret sharing representation (which requires that the parties add a secret sharing of zero and that each party sends one share to one other party for a total communication of three shares). See [4] for more details.

In the *malicious* adversary setting, the parties are prevented from deviating from the protocol and gain knowledge from another party through the use of information-theoretic message authentication codes (MACs) in this work. In addition to computations over secret shares of the data, the parties also perform computations required for MACs. The operations are slightly more involved than in the semi-honest security setting, and the total communication is a bit bigger. We refer the reader to the relevant literature for details [20].

2.2.4 Building Blocks

Building on the cryptographic primitives listed above for addition and multiplication of secret shared values, MPC protocols for other operations have been developed in the literature. In this paper, we use:

- Secure matrix multiplication π_{DMM}: at the start of this protocol, the parties have secret sharings $[\![A]\!]$ and $[\![B]\!]$ of matrices A and B; at the end, the parties have a secret sharing $[\![C]\!]$ of the product of the matrices, $C = A \times B$. π_{DMM} can be constructed as a direct extension of the secure multiplication protocol for two integers, which we will denote as π_{DM} in the remainder of the paper. Similarly, we use π_{DP} to denote the protocol for the secure dot product of two vectors. In a replicated sharing scheme, dot products can be computed more efficiently than the direct extension from π_{DM}, and matrix multiplication can use this optimized version of dot products; we refer to Keller [29] for details.

- Secure comparison protocol π_{LT} [9]: at the start of this protocol, the parties have secret sharings $[\![x]\!]$ and $[\![y]\!]$ of two integers x and y; at the end, they have a secret sharing of 1 if $x < y$, and a secret sharing of 0 otherwise.
- Secure argmin protocol π_{ARGMIN}: this protocol accepts secret sharings of a vector of integers and returns a secret sharing of the index at which the vector has the minimum value. π_{ARGMIN} is straightforwardly constructed using the above mentioned secure comparison protocol.
- Secure equality test protocol π_{EQ} [10]: at the start of this protocol, the parties have secret sharings $[\![x]\!]$ and $[\![y]\!]$ of two integers x and y; at the end, they have a secret sharing of 1 if $x = y$, and a secret sharing of 0 otherwise.
- Secure division protocol π_{DIV} [10]: at the start of this protocol, the parties have secret sharings $[\![x]\!]_q$ and $[\![y]\!]_q$ of two integers x and y; at the end, they have a secret sharing $[\![z]\!]_q$ of the division of x and y, $z = x/y$.

Chapter 3

RELATED WORK

3.1 Private Feature Selection

Given the fact that feature selection is an import step in the data preparation pipeline, it has received remarkably little attention in the PPML literature to date. Feature selection techniques have been proposed that favor features that do not contain sensitive information [28]. Work like that is orthogonal to ours, as it assumes the existence of a data curator with full access to all the data.

Regarding approaches to private feature selection among multiple data owners, early attempts [5, 41] in the semi-honest setting use a "distributed secure sum protocol" reminiscent of the way in which sums are computed in MPC based on secret sharing (Section 2.2.3). The limitations of this work in terms of security include the fact that the parties find out which features are selected, and statistical information about the data is leaked to all parties during the computation of the feature scores, as only summations, and not other operations, are done in a secure manner. Rao et al. [37] proposed a more principled 2PC protocol with Paillier homomorphic encryption for private feature selection with χ^2 as filter criteria in the semi-honest setting, without an experimental evaluation of the proposed approach. To the best of our knowledge, private feature selection with malicious adversaries has not yet been proposed or evaluated. The recent approach by Ye et al. [45] is not based on cryptography, does not provide any formal privacy guarantees, and leaks information through disclosure of intermediate representations.

3.2 Secure Gini Score Computation

Besides as a technique to score features for feature selection, as we do in this thesis, Gini impurity is traditionally used in ML in the CART algorithm for training decision trees [8], and it has been adopted in MPC protocols for privacy-preserving training of decision tree models [18, 12, 1]. Gini score computation for continuous valued features, as we do in this thesis, is especially challenging from an MPC point of view, as it requires sorting of feature values to determine candidate split points in the feature range. Abspoel et al. [1] put ample effort in performing this sorting process as efficiently as possible in a secure manner. We take a drastically different approach by assuming that the mean of the feature values serves as a good approximation for an optimal split threshold. This has the double advantage that (1) there is no need for oblivious sorting of feature values, and (2) for each feature only one Gini score for one threshold θ has to be computed as opposed to computing the Gini score for multiple candidate thresholds and then selecting the best one through secure comparisons. This leads to significant efficiency gains, while preserving good accuracy, as we demonstrate in Chapter 5.

Chapter 4
METHODOLOGY

4.1 Feature Selection with Mean-Split GINI (MS-GINI)

Computing the Gini score of a feature with continuous values requires sorting the feature values to find an optimal split point. Since sorting feature values in an oblivious manner is computationally expensive, we propose a sort-free feature scoring algorithm called *Mean-Split GINI (MS-GINI)* shown in Algorithm 1. We first introduced MS-GINI and analyzed its utility in an application for cognitive load inference from wrist-band sensor measurements [31].

We have a set S of m training examples, where each training example consists of an input feature vector $(x_1, \ldots, x_p)$ and a corresponding label y. In MS-GINI, instead of sorting a feature, we propose to split the set of values of the j^{th} feature F_j based on its mean value as a threshold θ. We denote by $S_{\leq\theta}$ the set of instances that have $x_j \leq \theta$, and by $S_{>\theta}$ the set of instances that have $x_j > \theta$. Furthermore, for $c = 1, \ldots, n$, we denote by L_c the set of examples from S that have class label $y = c$. Based on the binary split, we define the MS-GINI ("Mean-Split" GINI) score for feature F_j as:

$$G(F_j) = \frac{1}{|S|} \cdot (|S_{\leq\theta}| \cdot G(S_{\leq\theta}) + |S_{>\theta}| \cdot G(S_{>\theta})) \tag{4.1}$$

with the Gini impurities of $S_{\leq\theta}$ and $S_{>\theta}$ defined as:

$$G(S_{\leq\theta}) = 1 - \sum_{c=1}^{n} (p_c^{\leq\theta})^2; \;\; G(S_{>\theta}) = 1 - \sum_{c=1}^{n} (p_c^{>\theta})^2 \tag{4.2}$$

and the probabilities defined as:

$$p_c^{\leq\theta} = \frac{|S_{\leq\theta} \cap L_c|}{|S_{\leq\theta}|}; \quad p_c^{>\theta} = \frac{|S_{>\theta} \cap L_c|}{|S_{>\theta}|} \tag{4.3}$$

Formulas (4.1), (4.2), and (4.3) are consistent with the definition of Gini score given in Section 2.1.3, and presented here in more detail to enhance the readability of our secure protocol $\pi_{\mathsf{MS-GINI}}$ for the computation of the Gini score $G(F_j)$ of feature F_j (described in Protocol 4 of Section 4.2).

MS-GINI is computationally cheaper than GI (Section 2.1.3) in two ways: MS-GINI does not require sorting of feature values, and with MS-GINI, for each feature, we have to compute the GINI score for only one threshold θ, instead of for potentially multiple candidate thresholds.

Unlike CART, we do not use MS-GINI as a criterion to select the best feature for each node while growing a tree-based model. Instead, as in [31], we use MS-GINI as a filtering approach to feature selection for other ML models. Algorithm 2 shows how we use MS-GINI to select features from raw features, and Section 4.1.1 provides a concrete example.

Algorithm 1: Mean-Split Gini Impurity Algorithm (MS-GINI)

```
Input  : A feature column $F = (f_1,f_2,...,f_m)$, a label vector $L = (l_1,l_2,...,l_m)$, a
         number $m$ and a number $n$, where $m$ is number of instances and $n$ is
         number of classes in the label vector
Output: MS-GINI score $G(F)$ of the feature $F$
1  $\theta \leftarrow \frac{f_1+f_2+...+f_m}{m}$
2  Initialize counters for two subsets (i.e. $a$ for $S_{\leq\theta}$ and $b$ for $S_{>\theta}$) and respective
     counters based on all $n$ classes from $L$ (i.e. vectors $A$ and $B$ of length $n$; $A[i]$ is the
     counter for values from $S_{\leq\theta}$ with corresponding label being equal to $i^{th}$ class; $B[i]$ is
     the counter for values from $S_{>\theta}$ with corresponding label being equal to $i^{th}$ class).
3  for $i \leftarrow 1$ to $m$ do
4      if $f_i > \theta$ then
5          $b \leftarrow b + 1$
6          $B[l_i] \leftarrow B[l_i] + 1$
7      else
8          $a \leftarrow a + 1$
9          $A[l_i] \leftarrow A[l_i] + 1$
10     end
11 end
12 $G(S_{\leq\theta}) \leftarrow a - \frac{1}{a} \cdot A \bullet A$
13 $G(S_{>\theta}) \leftarrow b - \frac{1}{b} \cdot B \bullet B$
14 $G(F) \leftarrow G(S_{\leq\theta}) + G(S_{>\theta})$
15 return $G(F)$
```

Algorithm 2: Feature Selection based on MS-GINI in the clear

```
Input  : A $m \times p$ data matrix $D = (F_1,F_2,...,F_p)$, a label vector $L = (l_1,l_2,...,l_m)$, a
         number $k$ and a number $n$, where $k$ numbers of features will be selected
         from $D$ and $n$ is number of classes in the label vector
Output: A $m \times k$ matrix $D'$
1 for $i \leftarrow 1$ to $p$ do
2     $G[i] \leftarrow$ MS-GINI($F_i$,$L$,$m$,$n$)
3 end
4 Select the $k$ indices from $\{1, 2, \ldots, p\}$ corresponding to the $k$ smallest GINI scores in
    $G$ (break any ties by selecting the lowest indices)
5 Build a $m \times k$ matrix $D' = (F'_1, F'_2, ..., F'_k)$ by directly accessing the $k$ selected
    columns of $D$
6 return $D'$
```

	F_1	F_2	F_3	F_4	F_5	F_6	Label
1	-0.6725	1.4488	0.6695	1.2530	-1.7579	-1.3341	1
2	-0.3324	-1.5118	-0.7126	-2.0453	1.5131	1.4599	0
3	0.0502	-0.9029	1.0801	-0.4622	0.3691	0.5204	1
4	0.1808	-0.6880	-0.5104	-1.0291	1.3735	-0.9454	1

Table 4.1: Sample data

4.1.1 Illustration of Plaintext Feature Selection

Suppose we have sample data (Table 4.1) with $m = 4$ instances and $p = 6$ features. We want to select $k = 2$ features from those 6 features and form a new data set with 4 instances and 2 features. Also, $n = 2$, i.e. we are dealing with a binary classification problem. We will apply Algorithm 2 on the sample data; the inputs D and L are as shown in (4.4):

$$D = \begin{pmatrix} -0.6725 & 1.4488 & 0.6695 & 1.2530 & -1.7579 & -1.3341 \\ -0.3324 & -1.5118 & -0.7126 & -2.0453 & 1.5131 & 1.4599 \\ 0.0502 & -0.9029 & 1.0801 & -0.4622 & 0.3691 & 0.5204 \\ 0.1808 & -0.6880 & -0.5104 & -1.0291 & 1.3735 & -0.9454 \end{pmatrix} \qquad L = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{4.4}$$

For each column (feature) in D (Line 1–3 of Algorithm 2), we apply Algorithm 1 to compute its MS-GINI score. For example, to compute the score for F_1 (Line 2 of Algorithm 2), the inputs for Algorithm 1 are as (4.5):

$$F_1 = \begin{pmatrix} -0.6725 \\ -0.3324 \\ 0.0502 \\ 0.1808 \end{pmatrix} \qquad L = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \qquad m = 4 \qquad n = 2 \tag{4.5}$$

First, we compute the mean value of F_1 as Line 1 of Algorithm 1:

$$\theta = \frac{-0.6725 + -0.3324 + 0.0502 + 0.1808}{4} = -0.1935$$

After splitting F_1 by threshold θ, $S_{\leq\theta} = \{-0.6725, -0.3324\}$ and $S_{>\theta} = \{0.0502, 0.1808\}$. Then, we compute counters a, b and counter vectors A, B as Line 3–11 for F_1 based on $S_{\leq\theta}$, $S_{>\theta}$ and L: $a = |S_{\leq\theta}| = 2$, $b = |S_{>\theta}| = 2$, $A = (1, 1)$ and $B = (0, 2)$. The entries in A denote that one instance is class 0 and one instance is class 1 for $S_{\leq\theta}$. B means no instance is class 0 and two instances are class 1 for $S_{>\theta}$. After computing the counters, we compute $G(S_{\leq\theta}) = a - \frac{1}{a} \cdot A \bullet A = 2 - \frac{1}{2} \cdot (1 \cdot 1 + 1 \cdot 1) = 1$ and $G(S_{>\theta}) = b - \frac{1}{b} \cdot B \bullet B = 2 - \frac{1}{2} \cdot (0 \cdot 0 + 2 \cdot 2) = 0$ as Line 12 and 13 of Algorithm 1. Finally, we compute the MS-GINI score of F_1: $G(F_1) = G(S_{\leq\theta}) + G(S_{>\theta}) = 1 + 0 = 1$ as Line 14 of Algorithm 1.

After computing the MS-GINI score for the other features in D in a similar way, we have the following MS-GINI scores vector as (4.6):

$$G = \begin{pmatrix} 1 \\ 1.33 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{4.6}$$

The lowest score, namely 1, appears 4 times in G, yet we only wish to select 2 features with lowest MS-GINI score. We break the ties by selecting the features with lowest indices among those least scores. That means we select the 1^{st} and 3^{rd} feature as Line 4 of Algorithm 2. Then, on Line 5 of Algorithm 2, we pick the 1^{st} and 3^{rd} columns of D and form D' as (4.7):

$$D' = \begin{pmatrix} -0.6725 & 0.6695 \\ -0.3324 & -0.7126 \\ 0.0502 & 1.0801 \\ 0.1808 & -0.5104 \end{pmatrix} \tag{4.7}$$

4.2 MPC Protocols $\pi_{\mathsf{FILTER-FS}}$, $\pi_{\mathsf{MS-GINI}}$, and $\pi_{\mathsf{GINI-FS}}$

In this section, we present a protocol for oblivious feature selection $\pi_{\mathsf{FILTER-FS}}$ based on precomputed scores for the features, followed by a protocol $\pi_{\mathsf{MS-GINI}}$ for computing the feature scores themselves in a private manner. We then combine $\pi_{\mathsf{FILTER-FS}}$ and $\pi_{\mathsf{MS-GINI}}$ into $\pi_{\mathsf{GINI-FS}}$ for experiments in Chapter 5, evaluated in a 3PC, honest majority setting with parties *Alice*, *Bob*, and *Carol* shown in Figure 1.2.

Although the protocols may look similar to the algorithms from the previous section at a high level, there are some important differences. $\pi_{\mathsf{FILTER-FS}}$ is based on Line 5–6 of Algorithm 2, $\pi_{\mathsf{MS-GINI}}$ is based on Algorithm 1 and $\pi_{\mathsf{GINI-FS}}$ is based on Algorithm 2. We designed protocols with care to retain computational efficiency while at the same time not leaking information about the data or the selected features.

Secure Filter-based Feature Selection $\pi_{\mathsf{FILTER-FS}}$

At the start of the protocol $\pi_{\mathsf{FILTER-FS}}$ for secure feature selection, the parties have secret shares of a data matrix D of size $m \times p$, in which the rows correspond to instances and the columns to features. The parties also have secret shares of a vector G of length p containing a score for each of the features. At the end of the protocol, the parties have a reduced matrix D' of size $m \times k$ in which only the columns from D corresponding to the lowest scores in G are retained (note that this protocol can be trivially modified to select the k features with the highest scores). The main ideas behind the protocol (which is described in Protocol $\pi_{\mathsf{FILTER-FS}}$) are to:

1. Determine the indices of the features that need to be selected (these are stored in a secret-shared way in I).
2. Create a matrix T in which the columns are one-hot-encoded representations of these indices.
3. Multiply D with this feature selection matrix T.

Before walking through the pseudocode of Protocol $\pi_{\mathsf{FILTER-FS}}$, we present a plaintext example to illustrate the notation.

Example 1. Consider the data matrix D at the left of Equation (4.8), containing values for $m = 5$ instances (rows) and $p = 4$ features (columns). Assume that the feature score vector is $G = [65, 26, 83, 14]$ and that we want to select the $k = 2$ features with the lowest scores in G.

$$\underbrace{\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \end{pmatrix}}_{D} \cdot \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}}_{T} = \underbrace{\begin{pmatrix} 4 & 2 \\ 8 & 6 \\ 12 & 10 \\ 16 & 14 \\ 20 & 18 \end{pmatrix}}_{D'} \tag{4.8}$$

The lowest scores in G are 14 and 26, hence the 4th and the 2nd column of D should be selected. The columns of T in Equation (4.8) are a one-hot-encoding of 4 and 2 respectively, and multiplying D with T will yield the desired reduced data matrix D'. This multiplication takes place on Line 9 in Protocol $\pi_{\mathsf{FILTER-FS}}$. The bulk of Protocol $\pi_{\mathsf{FILTER-FS}}$ is about how to construct T based on G. As explained below, this process involves an auxiliary vector, which, at the end of the protocol, contains the following values for our example: $I = [4, 2]$.

In the protocol, vector $[\![I]\!]_q$ of length k stores the indices of the k selected features out of the p features of $[\![D]\!]_q$ and matrix $[\![T]\!]_q$ is a $p \times k$ transformation matrix that eventually holds one-hot-encodings of the indices in I. Through executing Lines 1–8 of Protocol $\pi_{\mathsf{FILTER-FS}}$, the parties construct a feature selection matrix T based on the values in G. In Line 2 the index of the i^{th} smallest value in $[\![G]\!]_q$ is identified. To this end, the parties run a secure argmin protocol π_{ARGMIN}. The inner for-loop serves two purposes, namely constructing the i^{th} column of matrix T, and overwriting the score in G of the feature that was selected in Line 2 by the upper bound, so that it will not be selected anymore in further iterations of the outer for-loop (such an upper bound t is passed as input to Protocol $\pi_{\mathsf{FILTER-FS}}$ and is usually very easy to determine in practice, as most common feature scoring techniques range between 0 and 1):

- To construct the i^{th} column of T, the parties loop through row $j = 1 \ldots p$, and on Line 5, update $T[j][i]$ with either a 0 or a 1, depending on the outcome of the secure equality test on Line 4. The outcome of this test will be 1 exactly once, namely when j equals $I[i]$, hence Line 5 results in a one-hot-encoding of $I[i]$ stored in the ith column of T.
- The flag $flag_k$ computed on Line 4 is used again on Line 6 to overwrite $G[I[i]]$ with t in an oblivious manner, where t is a value that is larger than the highest possible score that occurs in $[\![G]\!]_q$. This theoretical upper bound t ensures that feature $I[i]$ will not be selected again in later iterations of the outer for-loop.

As is common in MPC protocols, we use multiplication instead of control flow logic for conditional assignments. To this end, a conditional based branch operation as "**if** c **then** $a \leftarrow b$" is rephrased as $a \leftarrow a + c \cdot (b - a)$. In this way, the number and the kind of operations executed by the parties does not depend on the actual values of the inputs, so it does not leak information that could be exploited by side-channel attacks. Such a conditional assignment occurs in Line 6 of Protocol $\pi_{\mathsf{FILTER-FS}}$, where the value of the condition c itself is computed on Line 4.

In the final step, on Line 9, the parties multiply matrix D with matrix T in a secure manner to obtain a matrix D' that contains only the feature columns corresponding to the k best features. Throughout this process, the parties are unaware of which features were actually selected. The secret shared matrix D' can subsequently be used as input for a privacy-preserving ML model training protocol, e.g. [17].

Protocol 3: Protocol $\pi_{\mathsf{FILTER-FS}}$ for Secure Filter based Feature Selection

```
Input : A secret shared $m \times p$ data matrix $[\![D]\!]_q$, a secret shared $p$-length score
        vector $[\![G]\!]_q$, the number $k < p$ of features to be selected, and a constant $t$
        that is bigger than the highest possible score in $[\![G]\!]_q$
Output: a secret shared $m \times k$ matrix $[\![D']\!]_q$
1  for $i \leftarrow 1$ to $k$ do
2      $[\![I[i]]\!]_q \leftarrow \pi_{\mathsf{ARGMIN}}([\![G]\!]_q)$
3      for $j \leftarrow 1$ to $p$ do
4          $[\![flag_k]\!]_q \leftarrow \pi_{\mathsf{EQ}}([\![I[i]]\!]_q, j)$
5          $[\![T[j][i]]\!]_q \leftarrow [\![flag_k]\!]_q$
6          $[\![G[j]]\!]_q \leftarrow [\![G[j]]\!]_q + \pi_{\mathsf{DM}}([\![flag_k]\!]_q, t - [\![G[j]]\!]_q)$
7      end
8  end
9  $[\![D']\!]_q \leftarrow \pi_{\mathsf{DMM}}([\![D]\!]_q, [\![T]\!]_q)$
10 return $[\![D']\!]_q$
```

Secure Feature Score Computation $\pi_{\mathsf{MS-GINI}}$

Protocol $\pi_{\mathsf{FILTER-FS}}$ assumes the availability of a feature score vector G and an upper bound t for the values in G. Below we explain how this can be obtained from the data in a secure manner. To this end, we present a protocol $\pi_{\mathsf{MS-GINI}}$ for computation of the score of a feature based on Gini impurity. This protocol is applicable to data sets with continuous features. It is computationally cheaper than previously proposed protocols for Gini impurity that rely on sorting of feature values. Furthermore, as showed in previous work [31] and in Table 5.1, the "Mean-Split" GINI score can yield similar accuracy improvements.

At the start of protocol $\pi_{\mathsf{MS-GINI}}$, the parties have secret shares of a feature column F

(think of this as a column from data matrix D in Example 1), as well as secret shares of an one-hot-encoded version of the label vector. The latter is represented as a label-class matrix $[\![L]\!]_q$, in which $[\![L[i][j]]\!]_q = [\![1]\!]_q$ means that the label of the i^{th} instance is equal to the j^{th} class. Otherwise, $[\![L[i][j]]\!]_q = [\![0]\!]_q$. We note that, while there are n classes, it is sufficient for L to contain only $n-1$ columns: as there is exactly one value 1 per row, the value of the n^{th} column is implicit from the values of the other columns. We indirectly take advantage of this fact by terminating the loop on Line 6–10 at $n-1$, and performing calculations for the n^{th} class separately and in a cheaper manner on Line 13–14, as we explain in more detail below.

On Line 1, the parties compute $[\![\theta]\!]_q$ as a threshold to split the input feature $[\![F]\!]_q$, as the mean of the feature values in the column. To this end, each party first sums up the secret shares of the feature values, and then multiplies the sum with a known constant $\frac{1}{m}$ locally. Line 2 is to initialize all counters related to $S_{\leq\theta}$ and $S_{>\theta}$ to zero. After Line 14, these counters will contain the following values:

$$\begin{aligned} a &= |S_{\leq\theta}| \\ b &= |S_{>\theta}| \\ A[j] &= |S_{\leq\theta} \cap L_j|, \text{ for } j = 1 \ldots n \\ B[j] &= |S_{>\theta} \cap L_j|, \text{ for } j = 1 \ldots n \end{aligned}$$

These counters are needed for the probabilities in Equation (4.3). For each instance, in Line 4 of Protocol $\pi_{\mathsf{MS-GINI}}$, the parties perform a secure comparison to determine whether the instance belongs to $S_{>\theta}$. The outcome of that test is added to b on Line 5. Since the total number of instances is m, a can be straightforwardly computed as $m - b$ after the outer for-loop, i.e. on Line 12. Lines 7–8 check whether the instance belongs to $S_{>\theta} \cap L_j$, in which case $B[j]$ is incremented by 1. The equivalent operation of Line 7–8 for $A[j]$ would be $[\![A[j]]\!]_q \leftarrow [\![A[j]]\!]_q + \pi_{\mathsf{DM}}((1 - [\![flag_s]\!]_q), [\![L[i][j]]\!]_q)$. We have simplified this instruction on Line 9, taking advantage of the fact that $\pi_{\mathsf{DM}}([\![flag_s]\!]_q, [\![L[i][j]]\!]_q)$ has been precomputed as $[\![flag_m]\!]_q$ on Line 7.

On Line 13–14 the parties compute $[\![A[n]]\!]_q$ and $[\![B[n]]\!]_q$, leveraging the fact that sum of all values in $[\![A]\!]_q$ is $[\![a]\!]_q$, and the sum of all values in $[\![B]\!]_q$ is $[\![b]\!]_q$. All operations on Line 13–14 can be performed locally by the parties, on their own shares. Moving the computation of $[\![A[n]]\!]_q$ and $[\![B[n]]\!]_q$ out of the for-loop, reduces the number of secure multiplications needed from $m \times n$ to $m \times (n-1)$. In the case of a binary classification problem, i.e. $n = 2$, this means that the number of secure multiplications required is cut down by half.

Using the notations for the counters from the pseudocode of Protocol $\pi_{\textsf{MS-GINI}}$, Equation (4.1) comes down to:

$$G(F) = \frac{1}{m} \cdot \left[a \cdot \left(1 - \sum_{j=1}^{n} \left(\frac{A[j]}{a} \right)^2 \right) + b \cdot \left(1 - \sum_{j=1}^{n} \left(\frac{B[j]}{b} \right)^2 \right) \right] = \frac{1}{m} \cdot \left[\left(a - \frac{1}{a} \cdot A \bullet A \right) + \left(b - \frac{1}{b} \cdot B \bullet B \right) \right]$$

in which $A \bullet A$ and $B \bullet B$ are the dot products of A and B with themselves, respectively. These computations are performed by the parties on Lines 15–17 using, among other things, the protocol $\pi_{\textsf{DP}}$ for secure dot product of vectors, and the protocol $\pi_{\textsf{DIV}}$ for secure division. We note that the final multiplication with the factor $1/m$ is omitted altogether from Protocol $\pi_{\textsf{MS-GINI}}$ as this will have no effect on the relative ordering of the scores of the individual features (Line 14 of Algorithm 1 also omits multiplication $1/m$ for the same reason).

If data are vertically partitioned and all data owners have the label vector, they can compute MS-GINI scores offline without $\pi_{\textsf{MS-GINI}}$, and the computing servers would only have to do feature selection based on pre-computed MS-GINI scores with protocol $\pi_{\textsf{FILTER-FS}}$. In reality, often, it is not reasonable to allow each data owner to have all labels, so we do not assume this scenario in our protocols.

Protocol 4: Protocol $\pi_{\mathsf{MS-GINI}}$ for Secure MS-GINI Score of a Feature

Input : A secret shared feature column $[\![F]\!]_q = ([\![f_1]\!]_q, [\![[\![f_2]\!]_q]\!]_q, ..., [\![f_m]\!]_q)$, a secret shared $m \times (n-1)$ label-class matrix $[\![L]\!]_q$, where m is the number of instances and n is the number of classes.

Output: MS-GINI score $[\![G(F)]\!]_q$ of the feature F

```
1  $[\![\theta]\!]_q \leftarrow ([\![f_1]\!]_q + [\![f_2]\!]_q + ... + [\![f_m]\!]_q) \cdot \frac{1}{m}$
2  Initialize $[\![a]\!]_q$, $[\![b]\!]_q$, $[\![A]\!]_q$ and $[\![B]\!]_q$ with zeros.
3  for $i \leftarrow 1$ to $m$ do
4      $[\![flag_s]\!]_q \leftarrow \pi_{\mathsf{LT}}([\![\theta]\!]_q, [\![f_i]\!]_q)$
5      $[\![b]\!]_q \leftarrow [\![b]\!]_q + [\![flag_s]\!]_q$
6      for $j \leftarrow 1$ to $n-1$ do
7          $[\![flag_m]\!]_q \leftarrow \pi_{\mathsf{DM}}([\![flag_s]\!]_q, [\![L[i][j]]\!]_q)$
8          $[\![B[j]]\!]_q \leftarrow [\![B[j]]\!]_q + [\![flag_m]\!]_q$
9          $[\![A[j]]\!]_q \leftarrow [\![A[j]]\!]_q + [\![L[i][j]]\!]_q - [\![flag_m]\!]_q$
10     end
11 end
12 $[\![a]\!]_q \leftarrow m - [\![b]\!]_q$
13 $[\![A[n]]\!]_q \leftarrow [\![a]\!]_q - ([\![A[1]]\!]_q + ... + [\![A[n-1]]\!]_q)$
14 $[\![B[n]]\!]_q \leftarrow [\![b]\!]_q - ([\![B[1]]\!]_q + ... + [\![B[n-1]]\!]_q)$
15 $[\![G(S_{\leq\theta})]\!]_q \leftarrow [\![a]\!]_q - \pi_{\mathsf{DM}}(\pi_{\mathsf{DP}}([\![A]\!]_q, [\![A]\!]_q), \pi_{\mathsf{DIV}}(1, [\![a]\!]_q))$
16 $[\![G(S_{>\theta})]\!]_q \leftarrow [\![b]\!]_q - \pi_{\mathsf{DM}}(\pi_{\mathsf{DP}}([\![B]\!]_q, [\![B]\!]_q), \pi_{\mathsf{DIV}}(1, [\![b]\!]_q))$
17 $[\![G(F)]\!]_q \leftarrow [\![G(S_{\leq\theta})]\!]_q + [\![G(S_{>\theta})]\!]_q$
18 return $[\![G(F)]\!]_q$
```

Secure Feature Selection with MS-GINI $\pi_{\mathsf{GINI-FS}}$

Protocol $\pi_{\mathsf{GINI-FS}}$ performs secure filter-based feature selection with MS-GINI, used for the experiments in this work. It combines the building blocks presented earlier in the section.

By executing the loop on Line 1–3, the parties compute the MS-GINI score of the i^{th} feature from the original data matrix $[\![D]\!]_q$ using protocol $\pi_{\mathsf{MS-GINI}}$, and store it into $[\![G[i]]\!]_q$. On Line 4, the parties perform filter-based feature selection using protocol $\pi_{\mathsf{FILTER-FS}}$ to obtain a $m \times k$ matrix $[\![D']\!]_q$ with k selected features from $[\![D]\!]_q$.

As the standard GINI score is upper bounded by 1, and $\pi_{\mathsf{MS-GINI}}$ ignores the multiplication by $1/m$ for efficiency reasons, it is safe to use m as the upper bound that is passed to protocol $\pi_{\mathsf{FILTER-FS}}$ on Line 4.

Protocol 5: Protocol $\pi_{\mathsf{GINI-FS}}$ for Secure Filter-based Feature Selection with MS-GINI

```
  Input  : A secret shared $m \times p$ data matrix $[\![D]\!]_q = ([\![F_1]\!]_q, [\![F_2]\!]_q, ..., [\![F_p]\!]_q)$, a secret
           shared $m \times (n-1)$ label-class matrix $[\![L]\!]_q$, where $m$ is the number of
           instances, $p$ the number of features, $n$ the number of classes, and $k$ the
           number of features to be selected.
  Output: a secret shared $m \times k$ matrix $[\![D']\!]_q$
1 for $i \leftarrow 1$ to $p$ do
2     $[\![G[i]]\!]_q \leftarrow \pi_{\mathsf{MS-GINI}}([\![F_i]\!]_q, [\![L]\!]_q, m, n)$
3 end
4 $[\![D']\!]_q \leftarrow \pi_{\mathsf{FILTER-FS}}([\![D]\!]_q, [\![G]\!]_q, k, m)$
5 return $[\![D']\!]_q$
```

4.2.1 Illustration of Secure Feature Selection

Section 4.1.1 shows an example of feature selection on plaintext data. There are some differences in the secure manner, which we illustrate below.

Suppose we use the same sample data (Table 4.1) with $m = 4$ instances and $p = 6$ features. Each value is converted to fixed-point representation and secret shared as described in Section 2.2.3. Our target is also to select $k = 2$ features from 6 raw features and form a selected data matrix with 4 instances and 2 features, but we do it in a privacy-preserving manner this time with protocol $\pi_{\mathsf{GINI-FS}}$. Also, $n = 2$, i.e. there are 2 classes. As input for protocol $\pi_{\mathsf{GINI-FS}}$, D stays the same as in Algorithm 2, but we use a label-class matrix instead of the label vector of Algorithm 2. As explained in Section $\pi_{\mathsf{MS-GINI}}$, L is a 4×1 matrix precomputed by data owners and L[i][j] means that the label of the i^{th} instance is equal to the j^{th} class. Hence, our secret shared $[\![D]\!]_q$ and $[\![L]\!]_q$ are shown in (4.9):

$$[\![D]\!]_q = \begin{pmatrix} [\![-0.6725]\!]_q & [\![1.4488]\!]_q & [\![0.6695]\!]_q & [\![1.2530]\!]_q & [\![-1.7579]\!]_q & [\![-1.3341]\!]_q \\ [\![-0.3324]\!]_q & [\![-1.5118]\!]_q & [\![-0.7126]\!]_q & [\![-2.0453]\!]_q & [\![1.5131]\!]_q & [\![1.4599]\!]_q \\ [\![0.0502]\!]_q & [\![-0.9029]\!]_q & [\![1.0801]\!]_q & [\![-0.4622]\!]_q & [\![0.3691]\!]_q & [\![0.5204]\!]_q \\ [\![0.1808]\!]_q & [\![-0.6880]\!]_q & [\![-0.5104]\!]_q & [\![-1.0291]\!]_q & [\![1.3735]\!]_q & [\![-0.9454]\!]_q \end{pmatrix}$$
$$[\![L]\!]_q = \begin{pmatrix} [\![0]\!]_q \\ [\![1]\!]_q \\ [\![0]\!]_q \\ [\![0]\!]_q \end{pmatrix} \tag{4.9}$$

For each feature in $[\![D]\!]_q$, the parties compute the MS-GINI score securely. For instance, for $[\![F_1]\!]_q$, the parties compute $[\![G(F_1)]\!]_q = [\![1]\!]_q$ with protocol $\pi_{\mathsf{MS-GINI}}$, which is similar to Algorithm 1. After computing all MS-GINI scores (Line 1–3 of Protocol $\pi_{\mathsf{GINI-FS}}$), the parties have a secret shared vector of MS-GINI scores $[\![G]\!]_q$ as (4.10):

$$[\![G]\!]_q = \begin{pmatrix} [\![1]\!]_q \\ [\![1.33]\!]_q \\ [\![1]\!]_q \\ [\![1]\!]_q \\ [\![1]\!]_q \\ [\![1]\!]_q \end{pmatrix} \tag{4.10}$$

As long as we have $[\![G]\!]_q$, we can do privacy preserving filter-based feature selection by protocol $\pi_{\mathsf{FILTER-FS}}$. In the plaintext case, it is easy to pick k least elements in a vector and build D' by directly accessing k selected columns of D. In PPML, we have to use protocol π_{ARGMIN} k times and overwrite the score of the selected feature with a highest possible score, so that the feature will not be selected again in the next iteration. To this end, we use $t = m$ as explained in Section $\pi_{\mathsf{GINI-FS}}$. Similar to the in-the-clear version, we select the 1^{st} and 3^{rd} features. The indices 1 and 3 are stored in $[\![I]\!]_q$ and used to build the transformation matrix $[\![T]\!]_q$ (line 1–8 in protocol $\pi_{\mathsf{FILTER-FS}}$) as (4.11):

$$[\![I]\!]_q = \begin{pmatrix} [\![1]\!]_q \\ [\![3]\!]_q \end{pmatrix} \quad [\![T]\!]_q = \begin{pmatrix} [\![1]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![1]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \end{pmatrix} \tag{4.11}$$

Finally, on Line 9 of protocol $\pi_{\mathsf{FILTER-FS}}$, $[\![D']\!]_q$ is computed by secure matrix multiplication protocol π_{DMM} as (4.12):

$$\begin{aligned} [\![D']\!]_q &= [\![D]\!]_q \cdot [\![T]\!]_q \\ &= \begin{pmatrix} [\![-0.6725]\!]_q & [\![1.4488]\!]_q & [\![0.6695]\!]_q & [\![1.2530]\!]_q & [\![-1.7579]\!]_q & [\![-1.3341]\!]_q \\ [\![-0.3324]\!]_q & [\![-1.5118]\!]_q & [\![-0.7126]\!]_q & [\![-2.0453]\!]_q & [\![1.5131]\!]_q & [\![1.4599]\!]_q \\ [\![0.0502]\!]_q & [\![-0.9029]\!]_q & [\![1.0801]\!]_q & [\![-0.4622]\!]_q & [\![0.3691]\!]_q & [\![0.5204]\!]_q \\ [\![0.1808]\!]_q & [\![-0.6880]\!]_q & [\![-0.5104]\!]_q & [\![-1.0291]\!]_q & [\![1.3735]\!]_q & [\![-0.9454]\!]_q \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} [\![1]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![1]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \\ [\![0]\!]_q & [\![0]\!]_q \end{pmatrix} \\ &= \begin{pmatrix} [\![-0.6725]\!]_q & [\![0.6695]\!]_q \\ [\![-0.3324]\!]_q & [\![-0.7126]\!]_q \\ [\![0.0502]\!]_q & [\![1.0801]\!]_q \\ [\![0.1808]\!]_q & [\![-0.5104]\!]_q \end{pmatrix} \end{aligned} \tag{4.12}$$

Chapter 5

EXPERIMENTS AND RESULTS

The first four columns of Table 5.1 and Table 5.2 contain details for three data sets corresponding to binary classification tasks with continuous valued input features: Cognitive Load Detection (CogLoad) [24], Lee Silverman Voice Treatment (LSVT) [42], and Speed Dating (SPEED) [22], along with the number of instances m, raw features p, selected features k, and folds for cross-validation (CV).

The last five columns of Table 5.1 contain accuracy results by averaging from CV for logistic regression (LR) models trained on the RAW data sets with all p features, and on reduced data sets with only the top k features selected with a variety of scoring techniques, namely MS-GINI (as proposed in this paper), traditional Gini impurity (GI), Pearson correlation coefficient (PCC), and mutual information (MI). Feature selection with all these techniques was performed according to the filter approach, i.e. independently of the fact that the selected features were subsequently used to train a LR model. As the results show, feature selection based on MS-GINI is at par with the other methods, and substantially improves the accuracy compared to model training on the RAW data sets.

Table 5.1: Feature selection accuracy

	data set details				logistic regression accuracy results				
Data set	m	p	k	#folds	RAW	**MS-GINI**	GI	PCC	MI
CogLoad	632	120	12	6	50.90%	**52.50%**	52.70%	48.57%	51.59%
LSVT	126	310	103	10	80.09%	**86.15%**	82.74%	78.89%	85.38%
SPEED	8,378	122	67	10	95.24%	**97.26%**	95.56%	95.89%	95.83%

Table 5.2: Feature selection runtime results

	data set details				runtime	
Data set	m	p	k	#folds	semi-honest	malicious
CogLoad	632	120	12	6	48 sec	1,238 sec
LSVT	126	310	103	10	103 sec	5,408 sec
SPEED	8,378	122	67	10	1,531 sec	90,540 sec

The last two columns of Table 5.2 contain runtime results for protocol $\pi_{\mathsf{GINI-FS}}$ for secure filter-based feature selection with MS-GINI. To obtain these results, we implement $\pi_{\mathsf{GINI-FS}}$ along with the supporting protocols $\pi_{\mathsf{MS-GINI}}$ and $\pi_{\mathsf{FILTER-FS}}$ in MP-SPDZ [29]. All benchmark tests are completed on AWS *c5.xlarge* virtual machines. Each VM contains 4 cores, 8 GiB of memory, and up to a 10 Gbps network bandwidth between each virtual machine. The runtime results are for semi-honest and malicious adversary models (see Section 2.2.2) in a 3PC, honest-majority setting over a ring $\mathbb{Z}_q$ with $q = 2^{64}$. Each of the parties runs on separate machines, which means that the results in Table 5.2 cover communication time in addition to computation time. The reported runtime results in Table 5.2 are per fold.

As expected, the runtime increases in terms of the number of instances m, the number of original features p, and the number of selected features k. The increase in runtime for the SPEED vs. the CogLoad data set e.g., which have almost the same number of original features p, is due both to the increase in m (which affects Line 9 in Protocol $\pi_{\mathsf{FILTER-FS}}$, and Line 3-11 in Protocol $\pi_{\mathsf{MS-GINI}}$), and the increase in k (which affects Line 1-8 of Protocol $\pi_{\mathsf{FILTER-FS}}$). Also, completing private feature selection in the malicious setting takes substantially longer than in the semi-honest setting; this increase in runtime is a price one may wish to pay for security (correctness) in case the parties can not be trusted to follow the protocol instructions. Since all operations in MP-SPDZ are based on arithmetic black-box model, security of our protocols implemented by MP-SPDZ in this work is proved.

Chapter 6
CONCLUSION AND FUTURE WORK

Data preprocessing, an important part of the machine learning (ML) model development pipeline, has been largely overlooked in the privacy-preserving machine learning (PPML) literature to date. In this thesis we have proposed a Secure Multiparty Computation (MPC) protocol for privacy-preserving selection of the top k features of a data set, and we have demonstrated its feasibility in practice through an experimental evaluation. Our protocol is based on the filter approach for feature selection, which means that it is independent of any specific ML model architecture. Furthermore, it can be used in combination with any feature scoring technique. In this thesis, we have proposed an efficient MPC protocol based on Gini impurity to this end.

MPC protocols for many more tasks related to the data preprocessing phase still need to be developed, including privacy-preserving hyperparameter search to determine the best value of k for the number of features to be selected, as well as protocols for dealing with outliers and missing values. While these may be perceived as less exciting tasks of the ML end-to-end pipeline, they are crucial to enable PPML applications in practical data science.

BIBLIOGRAPHY

[1] Mark Abspoel, Daniel Escudero, and Nicolaj Volgushev. Secure training of decision trees with continuous attributes. In *Proceedings on Privacy Enhancing Technologies (PoPETs), Issue 1*, 2021.

[2] Anisha Agarwal, Rafael Dowsley, Nicholas D McKinney, Dongrui Wu, Chin-Teng Lin, Martine De Cock, and Anderson Nascimento. Protecting privacy of users in brain-computer interface applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(8):1546–1555, 2019.

[3] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. QUOTIENT: two-party secure neural network training and prediction. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1231–1247, 2019.

[4] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, page 805–817, 2016.

[5] Madhushri Banerjee and Sumit Chakravarty. Privacy preserving feature selection for distributed data using virtual dimension. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2281–2284, 2011.

[6] Kyle Bittner, Martine De Cock, and Rafael Dowsley. Private speech characterization with secure multiparty computation. *arXiv preprint arXiv:2007.00253*, 2020.

[7] Dan Bogdanov, Sven Laur, and Riivo Talviste. Oblivious sorting of secret-shared data. 2013.

[8] Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984.

[9] Octavian Catrina and Sebastiaan De Hoogh. Improved primitives for secure multiparty integer computation. In *International Conference on Security and Cryptography for Networks*, pages 182–199. Springer, 2010.

[10] Octavian Catrina and Amitabh Saxena. Secure computation with fixed-point numbers. In Radu Sion, editor, *Financial Cryptography and Data Security*, pages 35–50, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[11] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers Electrical Engineering*, 40(1):16 – 28, 2014. 40th-year commemorative issue.

[12] Chaitali Choudhary, Martine De Cock, Rafael Dowsley, Anderson C.A. Nascimento, and Davis Railsback. Secure training of extra trees classifiers over continuous data. In *AAAI-20 Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[13] Ronald Cramer, Ivan Bjerre Damgard, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[14] 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 *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, AISec '15, page 3–14, New York, NY, USA, 2015. Association for Computing Machinery.

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

[16] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson C.A. 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*, PP(99):1–1, 2017.

[17] Martine De Cock, Rafael Dowsley, Anderson Nascimento, Davis Railsback, Jianwei Shen, and Ariel Todoki. Fast secure logistic regression for high dimensional gene data. In *Privacy in Machine Learning workshop (PriML2019) - NeurIPS Workshop*, 2019.

[18] 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.

[19] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[20] Hendrik Eerikson, Marcel Keller, Claudio Orlandi, Pille Pullonen, Joonas Puura, and Mark Simkin. Use your brain! Arithmetic 3PC for any modulus with active security. In

1st Conference on Information-Theoretic Cryptography (ITC 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

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

[22] Raymond Fisman, Sheena S. Iyengar, Emir Kamenica, and Itamar Simonson. Gender Differences in Mate Selection: Evidence From a Speed Dating Experiment. *The Quarterly Journal of Economics*, 121(2):673–697, 2006.

[23] Monica Franzese and Antonella Iuliano. Correlation analysis. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 706 – 721. Academic Press, Oxford, 2019.

[24] Martin Gjoreski, Tine Kolenik, Timotej Knez, Mitja Luštrek, Matjaž Gams, Hristijan Gjoreski, and Veljko Pejović. Datasets for cognitive load inference using wearable sensors and psychological traits. *Applied Sciences*, 10(11):3843, May 2020.

[25] Michael T. Goodrich. Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in $\mathcal{O}(n)$ time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 684–693, 2014.

[26] 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.

[27] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3(null):1157–1182, March 2003.

[28] Yasser Jafer, Stan Matwin, and Marina Sokolova. A framework for a privacy-aware feature selection evaluation measure. In *13th Annual Conference on Privacy, Security and Trust (PST)*, pages 62–69. IEEE, 2015.

[29] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. Cryptology ePrint Archive, Report 2020/521, 2020. `https://eprint.iacr.org/2020/521`.

[30] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. *Proceedings of 2014 Science and Information Conference, SAI 2014*, pages 372–378, 10 2014.

[31] Xiling Li and Martine De Cock. Cognitive load detection from wrist-band sensors. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, UbiComp-ISWC '20, page 456–461, New York, NY, USA, 2020. Association for Computing Machinery.

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

[33] Steven Lohr. For big-data scientists, 'janitor work' is key hurdle to insights. The New York Times, 2014.

[34] Jianyu Miao and Lingfeng Niu. A survey on feature selection. *Procedia Computer Science*, 91:919 – 926, 2016. Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016).

[35] 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, May 2017.

[36] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy (SP)*, pages 334–348, 2013.

[37] Vanishree Rao, Yunhui Long, Hoda Eldardiry, Shantanu Rane, Ryan A. Rossi, and Frank Torres. Secure two-party feature selection. *ArXiv*, abs/1901.00832, 2019.

[38] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

[39] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman. *Mental Poker*, pages 37–43. Springer US, Boston, MA, 1981.

[40] Colin Shearer. The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, 5(4):13–22, 2000.

[41] Mina Sheikhalishahi and Fabio Martinelli. Privacy-utility feature selection as a privacy mechanism in collaborative data classification. In *IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 244–249, 2017.

[42] Athanasios Tsanas, Max A. Little, Cynthia Fox, and Lorraine O. Ramig. Objective automatic assessment of rehabilitative speech treatment in parkinson's disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):181–190, 2014.

[43] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019.

[44] Andrew. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, Nov 1982.

[45] Xiucai Ye, Hongmin Li, Akira Imakura, and Tetsuya Sakurai. Distributed collaborative feature selection based on intermediate representation. In *International Joint Conference on Artificial Intelligence*, pages 4142–4149, 2019.