Privacy-Preserving Feature Selection with Secure Multiparty Computation

Xiling Li, Rafael Dowsley and Martine De Cock

***Abstract*—Existing work on privacy-preserving machine learning 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 the first MPC based protocol for private feature selection based on the filter method, which is independent of model training, and can be used in combination with any MPC protocol to rank features. We propose an efficient feature scoring protocol based on Gini impurity to this end. To demonstrate the feasibility of our approach for practical data science, we 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 multiple servers, with semi-honest and with 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 ranging from several seconds to an hour for our protocols to finish, depending on the size of the data set and the security settings.**

I. Introduction

Machine learning (ML) thrives because of the availability of an abundant amount of data, and of computational resources and devices to collect and process such data. In many effective ML applications, the data that is consumed during ML model training and inference is often of a very personal nature. Protection of user data has become a significant concern in ML model development and deployment, giving rise to laws to safeguard the privacy of users, such as the European General Data Protection Regulation (GDPR) and the California Customer Privacy Act (CCPA). Cryptographic protocols that allow computations on encrypted data are an increasingly important mechanism to enable data science applications while complying with privacy regulations. In this paper, we contribute to the field of privacy-preserving machine learning (PPML), a burgeoning and interdisciplinary research area at the intersection 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 their private information to each other [12]. We consider the scenario where different data owners or enterprises are interested in training an 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 an 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.

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 [26], [17], [11], [1], linear regression models [29], [15], [2], and neural network architectures [28], [3], [34], [21], [16]. Existing approaches assume 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 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 [27]. 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!

Xiling Li is with the School of Engineering and Technology, University of Washington, Tacoma, WA, USA. Email: xl32@uw.edu

Rafael Dowsley is with the Faculty of Information Technology, Monash University, Clayton, Australia. Email: rafael.dowsley@monash.edu

Martine De Cock is with the School of Engineering and Technology, University of Washington, Tacoma, WA, USA and Ghent University, Ghent, Belgium. Email: mdecock@uw.edu

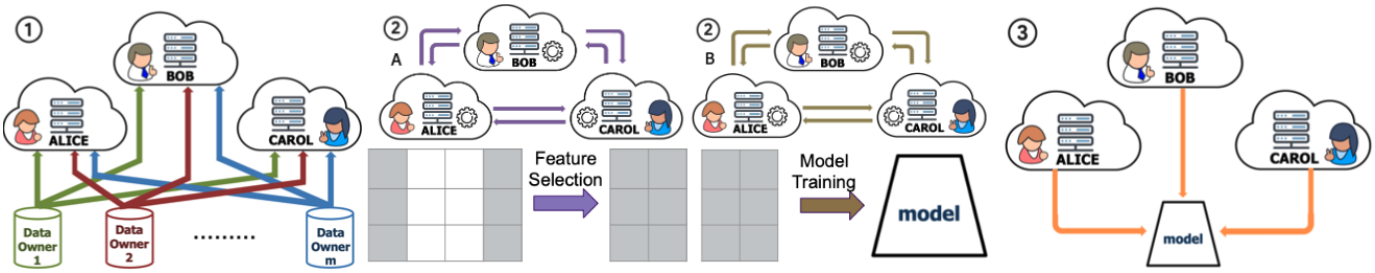

Fig. 1. Overview of private feature selection and model training in 3PC setting with computing servers (parties) *Alice*, *Bob*, and *Carol*.

In this paper, we contribute to filling this gap in the open literature by proposing *the first MPC based protocol for privacy-preserving feature selection*. Feature selection is the process of selecting a subset of relevant features for model training [10]. 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. Despite of its known shortcomings, including the fact that it considers each feature in isolation and ignores feature dependencies, the filter method is popular in practical data science because it is computationally very efficient, and independent of any specific ML model architecture.

The MPC based protocol $\pi_{\mathsf{FILTER-FS}}$ for private feature selection that we propose in this paper 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 an efficient feature scoring protocol $\pi_{\mathsf{MS-GINI}}$ based on Gini impurity, leaving the development of privacy-preserving protocols for other feature scoring techniques as future work. The computation of a GI 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. As we show in Sec. V, 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.

Fig. 1 illustrates the flow of private feature selection and subsequent model training at a high level in an outsourced "ML as a service setting" with three computing 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 the most efficient MPC schemes. In Step 1 of Fig. 1, 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 an ML model over the pre-processed data using existing privacy-preserving training protocols, e.g., a privacy-preserving protocol for logistic regression training [16]. 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 Sec. II-B). MPC protocols for Step 2B have previously been proposed. The focus of this paper 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.

After presenting preliminaries about Gini impurity and MPC in Sec. II, and discussing related work in Sec. III, we present our main protocol $\pi_{\mathsf{FILTER-FS}}$ for private feature selection and the supporting protocols $\pi_{\mathsf{GINI-FS}}$ and $\pi_{\mathsf{MS-GINI}}$ in Sec. IV. In Sec. V we demonstrate the feasibility of our approach for practical data science in terms of accuracy and runtime results through experiments executed on real-world data sets. In our experiments, we consider honest-majority 3PC settings with semi-honest as well as malicious adversaries. 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. Defending against the latter comes at a higher computational cost which, as we show, can be mitigated by using a recently proposed MPC scheme for 4PC.

II. Preliminaries

A. Feature Selection based on Gini Impurity

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 paper, we assume that there are n possible class labels. We wish to induce an 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. 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, made popular as part of the classification and regression tree algorithm (CART) [7].

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:

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

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:

$$G(F_j) = \sum_{i=1}^{\ell} \frac{|S_i|}{m} \cdot G(S_i) \tag{2}$$

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 [6], [20], in Sec. IV-B we propose a more straightforward approach for threshold selection which, as we show in Sec. V, yields desirable improvements in accuracy.

B. Secure Multiparty Computation

Protocols for MPC 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.

Adversarial Model: An adversary $\mathcal{A}$ can corrupt any number of parties. In a *dishonest-majority* setting, half or more of the parties may be corrupt, while in an *honest-majority* setting, more than half of the parties are honest (not corrupted). Furthermore, $\mathcal{A}$ can be a *semi-honest* or a *malicious* adversary. While a party corrupted by a semi-honest or "passive" adversary follows the protocol instructions correctly but tries to obtain additional information, parties corrupted by malicious or "active" adversaries can deviate from the protocol instructions. The protocols in Sec. IV are sufficiently generic to be used in dishonest-majority as well as honest-majority settings, with passive or active adversaries. This is achieved by changing the underlying MPC scheme to align with the desired security setting. Some of the most efficient MPC schemes have been developed for 3 parties, out of which at most one is corrupted. We evaluate the runtime of our protocols in this honest-majority 3PC setting, which is growing in popularity in the PPML literature, e.g. [14], [24], [31], [34], and we demonstrate how even better runtimes can be obtained with a recently proposed MPC scheme for 4PC with one corruption [13].

In the MPC schemes used in this paper, all computations by the parties (servers) are done over integers in a ring $\mathbb{Z}_q$. 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 [9]. After this conversion, the data owners secret share their values with the parties using a secret sharing scheme and proceed by performing operations over the secret shares.

For the **passive 3PC** setting, we follow a *replicated secret sharing* scheme from Araki et al. ([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 $z_{(i+1 \bmod 3)} = x_{(i+1 \bmod 3)} + y_{(i+1 \bmod 3)}$. 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 $x \cdot y = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3)$, 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 **active 3PC** setting, we use the MPC scheme *SYReplicated2k* recently proposed by Dalskov et al. ([13]). In this MPC scheme, the parties are prevented from deviating from the protocol and from gaining knowledge from other parties

through the use of information-theoretic message authentication codes (MACs). In addition to computations over secret shares of the data, the parties also perform computations required for MACs. See [13] for details. Finally, we use the MPC scheme recently proposed by Dalskov et al. ([13]) for the **active 4PC** setting, where the computations are outsourced to four servers out of which at most one has been corrupted by a malicious adversary.

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 ([23]) for details.
- Secure comparison protocol π_{LT} [8]: 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} [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 division protocol π_{DIV} [9]: 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 $z = x/y$.

III. Related Work

Private Feature Selection: Given that feature selection is an important 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 [22]. 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], [32] 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 (see Sec. II-B). 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. [30] 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 [35] is not based on cryptography, does not provide any formal privacy guarantees, and leaks information through disclosure of intermediate representations.

Secure Gini Score Computation: Besides as a technique to score features for feature selection, as we do in this paper, Gini impurity is traditionally used in ML in the CART algorithm for training decision trees [7], and it has been adopted in MPC protocols for privacy-preserving training of decision tree models [17], [11], [1]. Gini score computation for continuous valued features, as we do in this paper, 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 Sec. V.

Protocol 1 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 $[\![D']\!]_q$

for $i = 1$ **to** k **do**
  $[\![I[i]]\!]_q \leftarrow \pi_{\mathsf{ARGMIN}}\ ([\![G]\!]_q)$
  for $j \leftarrow 1$ **to** p **do**
    $[\![flag_k]\!]_q \leftarrow \pi_{\mathsf{EQ}}\ ([\![I[i]]\!]_q, j)$
    $[\![T[j][i]]\!]_q \leftarrow [\![flag_k]\!]_q$
    $[\![G[j]]\!]_q \leftarrow [\![G[j]]\!]_q + \pi_{\mathsf{DM}}\ ([\![flag_k]\!]_q, t - [\![G[j]]\!]_q)$
  end for
end for
$[\![D']\!]_q \leftarrow \pi_{\mathsf{DMM}}\ ([\![D]\!]_q, [\![T]\!]_q)$
return $[\![D']\!]_q$

IV. Methodology

We present a protocol for oblivious feature selection based on precomputed scores for the features, followed by a protocol for computing the feature scores themselves in a private manner. In Sec. V we evaluate the protocols in 3PC and 4PC honest-majority settings.

A. Secure Filter based Feature Selection

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 1) 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 1, we present a plaintext example to illustrate the notation.

Example 1. Consider the data matrix D at the left of Equation (3), 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.

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

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 (3) 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 1. The bulk of Protocol 1 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 1, 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 1 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 1, 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. [16].

B. Secure Feature Score Computation

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 shown in previous work [25] and in Sec. V, the "Mean-Split" GINI score can yield similar accuracy improvements.

Recall 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. 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:

$$G(F_j) = \frac{1}{m} \cdot (|S_{\leq\theta}| \cdot G(S_{\leq\theta}) + |S_{>\theta}| \cdot G(S_{>\theta})) \tag{4}$$

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

$$G(S_{\leq\theta}) = 1 - \sum_{c=1}^{n} (p_c^{\leq\theta})^2; \;\; G(S_{>\theta}) = 1 - \sum_{c=1}^{n} (p_c^{>\theta})^2 \tag{5}$$

and the probabilities defined as:

$$p_c^{\leq\theta} = \frac{|S_{\leq\theta} \cap L_c|}{|S_{\leq\theta}|}; \quad p_c^{>\theta} = \frac{|S_{>\theta} \cap L_c|}{|S_{>\theta}|} \tag{6}$$

Formulas (4), (5) and (6) are consistent with the definition of Gini score given in Sec. II, 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 $G(F)$ of feature F (described in Protocol 2).

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

Protocol 2 Protocol $\pi_{\mathsf{MS-GINI}}$ for Secure MS-GINI Score of a Feature

Input: A secret shared feature column $[\![F]\!]_q$ = ($[\![f_1]\!]_q$,$[\![f_2]\!]_q$,...,$[\![f_m]\!]_q$), 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: $\quad [\![flag_s]\!]_q \leftarrow \pi_{\mathsf{LT}}([\![\theta]\!]_q, [\![f_i]\!]_q)$
5: $\quad [\![b]\!]_q \leftarrow [\![b]\!]_q + [\![flag_s]\!]_q$
6: $\quad$ **for** $j \leftarrow 1$ **to** $n-1$ **do**
7: $\quad\quad [\![flag_m]\!]_q \leftarrow \pi_{\mathsf{DM}}([\![flag_s]\!]_q, [\![L[i][j]]\!]_q)$
8: $\quad\quad [\![B[j]]\!]_q \leftarrow [\![B[j]]\!]_q + [\![flag_m]\!]_q$
9: $\quad\quad [\![A[j]]\!]_q \leftarrow [\![A[j]]\!]_q + [\![L[i][j]]\!]_q - [\![flag_m]\!]_q$
10: $\quad$ **end for**
11: **end for**
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** $[\![G(F)]\!]_q$

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 (6). For each instance, in Line 4 of Protocol 2, 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 2, Equation (4) comes down to:

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

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 π_{DP} for secure dot product of vectors, and the protocol π_{DIV} for secure division. We note that the final multiplication with the factor $1/m$ is omitted altogether from Protocol 2 as this will have no effect on the relative ordering of the scores of the individual features.

If data are vertically partitioned and all data owners have the label vector, they can compute MS-GINI scores offline without $\pi_{\mathsf{MS-GINI}}$, and the computing servers would only have to do feature selection based on pre-computed MS-GINI scores with Protocol $\pi_{\mathsf{FILTER-FS}}$. 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.

C. Secure Feature Selection with MS-GINI

Protocol $\pi_{\mathsf{GINI-FS}}$ (described in Protocol 3) 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 3 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: $\quad [\![G[i]]\!]_q \leftarrow \pi_{\mathsf{MS-GINI}}([\![F_i]\!]_q, [\![L]\!]_q, m, n)$
3: **end for**
4: $[\![D']\!]_q \leftarrow \pi_{\mathsf{FILTER-FS}}([\![D]\!]_q, [\![G]\!]_q, k, m)$
5: **return** $[\![D']\!]_q$

V. Experiments and Results

The first four columns of Table I contain details for three data sets corresponding to binary classification tasks with continuous

TABLE I
FEATURE SELECTION ACCURACY AND RUNTIME RESULTS

	data set details				logistic regression accuracy results					runtime		
Data set	m	p	k	#folds	RAW	MS-GINI	GI	PCC	MI	passive 3PC	active 3PC	active 4PC
CogLoad	632	120	12	6	50.90%	52.50%	52.70%	48.57%	51.59%	50 sec	163 sec	79 sec
LSVT	126	310	103	10	80.09%	86.15%	82.74%	78.89%	85.38%	60 sec	254 sec	89 sec
SPEED	8,378	122	67	10	95.24%	97.26%	95.56%	95.89%	95.83%	949 sec	3,634 sec	1,435 sec

TABLE II
RUNTIME DETAILS FOR ACTIVE 3PC

	data set details			runtime		
Data set	m	p	k	Prot 1	Prot 1, Ln 9	Prot 2
CogLoad	632	120	12	27 sec	23 sec	1.13 sec
LSVT	126	310	103	152 sec	53 sec	0.33 sec
SPEED	8,378	122	67	1,837 sec	1,812 sec	14.73 sec

valued input features: Cognitive Load Detection[1] (CogLoad) [19], Lee Silverman Voice Treatment[2] (LSVT) [33], and Speed Dating[3] (SPEED) [18], along with the number of instances m, raw features p, selected features k, and folds for cross-validation (CV). The middle five columns of Table I 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 on par with the other methods, and substantially improves the accuracy compared to model training on the RAW data sets.

The last three columns of Table I contain runtime results for protocol $\pi_{\mathsf{GINI-FS}}$ for secure filter-based feature selection with MS-GINI (see Protocol 3). To obtain these results, we implemented $\pi_{\mathsf{GINI-FS}}$ along with the supporting protocols $\pi_{\mathsf{MS-GINI}}$ and $\pi_{\mathsf{FILTER-FS}}$ in MP-SPDZ [23]. All benchmark tests were completed on 3 or 4 co-located F32s V2 Azure virtual machines. Each VM contains 32 cores, 64 GiB of memory, and up to a 14 Gbps network bandwidth between each virtual machine. The runtime results are for semi-honest ("passive") and malicious ("active") adversary models (see Sec. II-B) in a 3PC or 4PC honest-majority setting over a ring $\mathbb{Z}_q$ with $q = 2^{64}$. Each of the parties ran on separate machines, which means that the results in Table I cover communication time in addition to computation time. Similarly as for the accuracies, the reported runtimes in Table I are an average across the folds. The relative differences between the passive 3PC, active 3PC , and active 4PC settings are in line with known findings from the MPC literature, in particular the fact that completing private feature selection in the active setting takes substantially longer than in the passive setting; this increase in runtime is a price one has to pay for security and correctness in case the parties can not be trusted to follow the protocol instructions.

[1] https://www.ubittention.org/2020/data/Cognitive- load%20challenge%20description.pdf

[2] https://archive.ics.uci.edu/ml/datasets/LSVT+Voice+Rehabilitation

[3] https://www.openml.org/d/40536

For further insight in the dominating factors in the runtime cost, in Table II we present more fine-grained runtime results for the active 3PC setting. Protocol 2, which is executed once per feature, in itself grows in the number of instances m. While the nested for-loop on Line 1-8 in Protocol 1 depends on k and p only, the matrix multiplication on Line 9 in Protocol 1 depends on all of m, p, and k, and contributes substantially to the runtime. 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 1, and Line 3-11 in Protocol 2), and the increase in k (which affects Line 1-8 of Protocol 1).

VI. CONCLUSION AND FUTURE WORK

Data preprocessing, an important part of the ML model development pipeline, has been largely overlooked in the PPML literature to date. In this paper we have proposed an 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 paper, we have proposed an efficient MPC protocol based on Gini impurity to this end.

In addition to MPC protocols for other feature selection techniques, 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.

REFERENCES

[1] Mark Abspoel, Daniel Escudero, and Nikolaj Volgushev. Secure training of decision trees with continuous attributes. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, pages 167–187, 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 805817, 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] Dan Bogdanov, Sven Laur, and Riivo Talviste. Oblivious sorting of secret-shared data. *Technical Report*, 2013.

[7] Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and Regression Trees*. Taylor and Francis, 1st edition, 1984.

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

[9] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2010.

[10] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28, 2014.

[11] C.A. Choudhary, M. De Cock, R. Dowsley, A. Nascimento, and D. Railsback. Secure training of extra trees classifiers over continuous data. In *AAAI-20 Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[12] Ronald Cramer, Ivan Bjerre Damgard, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 1st edition, 2015.

[13] A. Dalskov, D. Escudero, and M. Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. Cryptology ePrint Archive, Report 2020/1330, 2020.

[14] A. Dalskov, D. Escudero, and M. Keller. Secure evaluation of quantized neural networks. *Proceedings on Privacy Enhancing Technologies*, 2020(4):355–375, 2020.

[15] 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)*, page 314, 2015.

[16] 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. *BMC Medical Genomics*, 14(1):23, 2021.

[17] 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.

[18] 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.

[19] Martin Gjoreski, Tine Kolenik, Timotej Knez, Mitja Lutrek, Matja Gams, Hristijan Gjoreski, and Veljko Pejovi. Datasets for cognitive load inference using wearable sensors and psychological traits. *Applied Sciences*, 10(11):38–43, 2020.

[20] M. Goodrich. Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in o(n log n) time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 684–693, 2014.

[21] 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.

[22] 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.

[23] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, page 15751590, 2020.

[24] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. CrypTFlow: Secure TensorFlow inference. In *41st IEEE Symposium on Security and Privacy*, 2020.

[25] 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*, page 456461, 2020.

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

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

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

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

[30] Vanishree Rao, Yunhui Long, Hoda Eldardiry, Shantanu Rane, Ryan A. Rossi, and Frank Torres. Secure two-party feature selection. *arXiv preprint arXiv:1901.00832*, 2019.

[31] M.S. Riazi, C. Weinert, O. Tkachenko, E.M. Songhori, T. Schneider, and F. Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

[32] 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.

[33] 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.

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

[35] 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.