Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 8(12): 28517–28539.
DOI: 10.3934/math.20231459
Received: 11 September 2023
Revised: 13 October 2023
Accepted: 15 October 2023
Published: 19 October 2023

Research article

Privacy-preserving Naive Bayes classification based on secure two-party computation

Kun Liu and Chunming Tang*

School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006, China

* **Correspondence:** Email: ctang@gzhu.edu.cn.

Abstract: With the proliferation of data and machine learning techniques, there is a growing need to develop methods that enable collaborative training and prediction of sensitive data while preserving privacy. This paper proposes a new protocol for privacy-preserving Naive Bayes classification using secure two-party computation (STPC). The key idea is to split the training data between two non-colluding servers using STPC to train the model without leaking information. The servers secretly share their data and the intermediate computations using cryptographic techniques like Beaver's multiplication triples and Yao's garbled circuits. We implement and evaluate our protocols on the MNIST dataset, demonstrating that they achieve the same accuracy as plaintext computation with reasonable overhead. A formal security analysis in the semi-honest model shows that the scheme protects the privacy of the training data. Our work advances privacy-preserving machine learning by enabling secure outsourced Naive Bayes classification with applications such as fraud detection, medical diagnosis, and predictive analytics on confidential data from multiple entities. The modular design allows embedding different secure matrix multiplication techniques, making the framework adaptable. This line of research paves the way for practical and secure data mining in a distributed manner, upholding stringent privacy regulations.

Keywords: secure two-party computation; Naive Bayes classification; multiplication triple
Mathematics Subject Classification: 94A99

1. Introduction

With the development of information technology, research and machine learning (ML) is becoming more popular. This situation is derived from increasing data, computing resources, and devices becoming available to collect and process data. In many practical ML applications, the data consumed during ML model training and inference is often personal. Guaranteeing user data security has become an important issue based on the development and expansion of ML applications, resulting in laws protecting user privacy.

The construction of a Bayes classification model, similar to other traditional ML methods, mandates that the learning algorithm can access all training attributes and tuples. However, the growing need for distributed and heterogeneous storage and computing systems poses challenges such as non-trivial data movement and transformations, which are obstacles to the practical application of conventional ML algorithms. In addition, due to economic incentives or privacy legislation, multiple parties might not consent to or have the ability to share their data. As such, they are creating a requirement for privacy-preserving ML techniques to address this urgent issue adequately. This paper explores the current techniques used for constructing Bayes classification models in a heavily distributed environment while preserving privacy.

The technique under consideration finds significant importance in machine learning applications deployed in security-sensitive industries, such as the financial sector and electronic surveillance. To illustrate, the financial sector may witness a situation where two institutions aspire to collectively mine customer data. However, they are obligated by customer agreements and confidentiality regulations, which precludes them from directly sharing their data. Similarly, in electronic surveillance, Internet Service Providers hesitant to disclose any information regarding their customer base may seek the aid of consulting firms specializing in traffic analysis on their logs.

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 study, 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. Predictive analytics is the process of predicting future events by using statistical techniques to analyze current data. It is utilized in different areas, such as healthcare, mobility, financial services, insurance and marketing. Several techniques are available to exploit model prediction analysis for statistics, data mining, machine learning, and artificial intelligence. Predictive analysis widely uses classification algorithms in machine learning such as regression classification, Naive Bayes, support vector machines, and neural networks. These supervised learning approaches label the training data in advance and then exploit it to generate models that can be used to classify new instances.

While earlier work has advanced PPML, protocols optimized specifically for secure outsourced Naive Bayes modeling have yet to be explored. Most prior PPML research focuses on the model training phase but does not address the data preparation steps like cleaning and preprocessing, which account for up to 80% of the effort in real-world data science projects. Additionally, many existing cryptographic protocols for ML are designed generically without optimizations for particular machine learning algorithms like Naive Bayes. For instance, Kantarcıoglu et al. [1] first proposed protocols for privacy-preserving Naive Bayes training, but these only work for three or more parties. Vaidya et al. [2] later used homomorphic encryption, but their method requires retraining for each query. Our work builds on these efforts by presenting two-party computation protocols tailored specifically to enable efficient and secure Naive Bayes training and classification.

The accuracy of the models trained with supervised learning highly depends on the training data size. In general, the larger the training data set, the higher the accuracy of the model. Therefore, a sound classifier must collect a large amount of training data. However, those behaviours collecting data on a large scale bring about the risk of privacy information leakage to the data owners. In some practical application scenarios, training and prediction require datasets that often involve large amounts of sensitive individual information from different sources, such as salaries, medical records, and positions.

In particular, different financial institutions explore the same client's financial information for risk assessment, but cannot leak private information to each other. If two institutions collaboratively intend to build a risk classification model, the joint training of data may be a fundamental problem for information security concerns. Therefore, we address the problem of performing classification while protecting the privacy of the individuals who provide the training data, thus enabling companies and organizations to achieve their utility targets while helping individuals to protect their privacy.

Some works have shown that STPC already exists to focus on machine learning algorithms for training regression and neural network models with a dual-server model commonly used in previous work on PPML [3–6]. Therefore, we solve the problem of computational Bayesian classification based on two-party security while protecting the privacy of the individuals providing the training data, thereby enabling companies and organizations to achieve their utility goals while helping individuals protect their privacy.

Machine learning models often require large diverse datasets to train accurate models. However, in many real-world scenarios, the training data consists of sensitive information about individuals that cannot be shared openly. Consider a healthcare scenario where two hospitals want to collaborate to build a model for predictive diagnosis. While combining their patient data could train a more robust model, they cannot share raw medical records due to privacy regulations. This gives rise to the crucial need for privacy-preserving machine learning (PPML) techniques that enable collaborative modeling while protecting sensitive training data. As shown in Figure 1, we study a distributed setting where the data is partitioned between two non-colluding servers. The key challenge is to train machine learning models jointly without exchanging the raw data in plaintext.

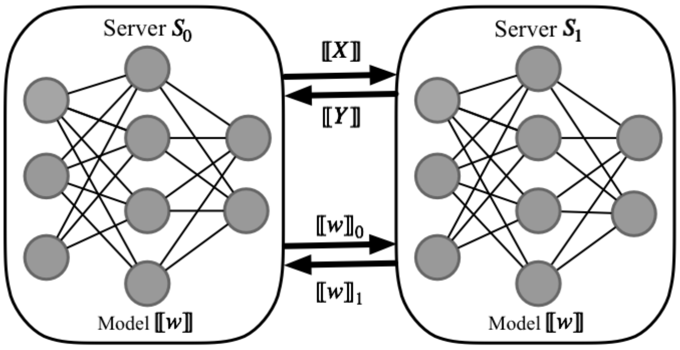

Figure 1. The basic framework of our scheme.

Our work focuses on secure two-party computation (STPC) techniques that allow joint computations on encrypted data. We utilize cryptographic protocols like Yao's garbled circuits, oblivious transfer and Beaver's multiplication triples to achieve secure outsourced modeling. Specifically, we design efficient STPC protocols optimized for Naive Bayes classification. Naive Bayes is chosen as it offers a simple yet accurate classifier suitable for textual, medical, and other data. Our protocols entail secure matrix operations for distributed model training and classification, while revealing only the final predicted labels and nothing about the sensitive inputs. The modular architecture allows different secure matrix multiplication schemes to be plugged in, providing flexibility. We empirically evaluate the performance on benchmark datasets, demonstrating the efficacy and minimal overhead of our privacy-preserving Naive Bayes protocols. Overall, this work opens the door for practical secure outsourced modeling, upholding data privacy in machine learning.

First, we design STPC protocols for privacy-preserving Naive Bayes classification in distributed machine learning settings, assuming that the data are distributed across different servers.

Second, we reduced the computational effort required to train both sides of a Naive Bayesian classifier to ensure multi-party matrix summation and two-sided matrix multiplication. This paradigm differs from previous work in this area because matrix multiplication cannot be easily generalized to an arbitrary number of multi-party cases for a secure comparison operation in the protocol. Thus, their training protocols are not applicable to multi-party cases. Our protocol treats the secure two-party matrix multiplication protocol as a complete "black box", meaning that any secure matrix product algorithm can be "embedded" in our paradigm to train models. By reducing the matrix multiplication of multiple data holders to bilateral matrix multiplication, we exploit this approach to the Naive Bayesian training process. The proposed method can be applied if the training process involves only multiplication and comparison.

Third, we empirically study our proposed scheme and experimentally verify that the protocol achieves precisely the same performance and that the additional computation and communication costs are linearly related to the size of the dataset.

We propound efficient and optimized secure two-party computation protocols that are specifically tailored for privacy-preserving Naive Bayes classification. These protocols are dedicatedly designed to enable Naive Bayes modeling in a secure outsourced environment. In addition, our proposed protocols reduce the multiparty computation setting to two non-colluding servers. This effectively extends the applicability of our approach to collaborative modeling scenarios where the training data is horizontally partitioned across multiple entities that cannot directly share their raw data. Our protocols espouse a modular architecture that facilitates flexibility in the choice of the underlying secure matrix multiplication scheme. By abstracting this as a modular component, different techniques from the literature can be readily plugged into our framework.

2. Related works

Over the past decade, cryptographic protocols designed using secure multi-party computation (SMC) have been developed to train ML models on aggregated data so that no personal data owners or businesses are leaked to anyone. The current work includes SMC protocols for training decision tree models [7–10], linear regression models [3,11–16] and neural network architectures [17–21]. Existing methods assume the dataset is pre-processed and clean with pre-selected and constructed features. Model building is only a tiny part of a data science project. Processing of real-world datasets requires that one must first clean and pre-process the original data, remove outliers, select training features and process the actual values of the data before modeling. Data scientists are estimated to spend 50% to 80% of their time on data wrangling rather than model training itself. The PPML solution must contain these data preparation steps to be adopted in practice. It makes little sense to protect the privacy of clean datasets during model training if the raw data must be leaked first to get those clean datasets.

Some probability statistics tools include randomization, anonymization, differential privacy, and data perturbation [2,22,23]. On the one hand, there is a trade-off between the trained model's accuracy and its level of privacy [24]. On the other hand, many probabilistic and statistical algorithms are practical when dealing with large datasets, but must be more accurate for small ones.

Earlier work by Agrawal et al. [7] introduced the notion of privacy-preserving data mining. Since then, several SMC protocols have been proposed for training models like regression, decision trees and neural networks. However, these assume pre-processed data, and focus only on model training. PPML has experienced significant growth in research trends over the past decade. Bost et al. designed classification protocols for decision trees, Naive Bayes and hyperplane decisions [25]. Their protocol uses two additive homomorphic encryption schemes that allow only additive operations, whereas the model training operation is completed using full homomorphic encryption (FHE). Their solution is to train the classification model on the plaintext and then place the encrypted classifier in a cloud server. At the prediction phase, the encrypted model is distributed to the client side, which calculates the classification probability and then interacts with the cloud server to obtain the classification result. Executing a whole protocol requires the client to interact with the server multiple times to calculate the classification results. Such an operation leads to high computational overhead for the client.

Wood et al. [26] proposed a comparison method for executing an efficient Naive Bayes classification [25, 27–29]. This study is still limited because the client implements the classification operation and often interacts with the cloud server. Sun et al. [30] exploited the FHE technology and employed it to train the same structure model as in [25, 31]. Their scheme allows the encrypted classification model to be sent and stored on a cloud server. Following model training, the model outputs are calculated on the cloud, resulting in client-server interaction. In addition, a Bayesian spam filter and decision trees that apply an improved FHE scheme were proposed by Khedr et al. [32]. Dowlin et al. [33] designed a classification protocol for neural networks. Aimed to exploit the properties of FHE effectively, they modified some of the functions used in the neural network, such as replacing the activation function with a low-order polynomial. It is a valuable notation that the two studies depicted classification methods, while they do not explain models or protocols of systems in the real world.

In contrast to the two-party training scheme, Kim et al. [34] proposed a protocol for Naive Bayes classification with the addition of a third party. In their protocol, the ability of a third party to keep the key and decrypt all ciphertexts is allowed. A cloud server is required to store the classification model in the ciphertext and execute computing operations over the ciphertext. Duy-Hien et al. [35, 36] proposed a privacy-preserving Naive Bayes classification solution based on secure multi-party computation. Therefore, this pattern allows third parties to access the client data and the intermediate results of the decryption in the model execution classification protocol. The leakage of intermediate results also contained sensitive information for both parties.

For training and prediction on ciphertext, Li et al. [37] devised a protocol that outsourced heavy computing operations to cloud servers while protecting classification models, client data, and classification results. The client has its public and secret key pair (pk_c, sk_c), and the data are encrypted and outsourced to the cloud server for calculation. Because the data cannot be calculated under different keys during encryption, the protocol applies a proxy re-encryption method based on Gentry's [38] bootstrap technology and re-encrypts the ciphertext under the public key pk_0. In their protocol, a third party that holds sk_0 and is responsible for decrypting the ciphertext is used to prevent the cloud from decoding the data from the client. Once the client executes the classification protocol, the classification result is blinded by a random number generated by the cloud service and sent to a third party. This result is decrypted by a third party, encrypted with pk_c, and sent back to the cloud server. The cloud server exploits pk_c to extract a random number from the ciphertext and then sends it to the client. The client uses sk_c to decrypt the data and obtain the classification result. Their proposed protocol provides security for classification models, customer data, and results. However, the bootstrap on which proxy re-encryption is based is expensive and can create a bottleneck because the cloud must re-encrypt all client-side data. This proxy re-encryption is unnecessary because clients can encrypt their data with pk_0 from the beginning of the protocol. Furthermore, since their protocol is generic, details of classification methods using FHE still need to be contained, which is a challenge when applying FHE to an application.

Focusing on Naive Bayes, Kantarcıoglu et al. [1] tackled the problem using additive secret sharing, but only for three or more parties. Vaidya et al. [39] later proposed a method using homomorphic encryption, but retraining per query. Our work builds on these efforts by presenting optimized two-party computation protocols tailored to Naive Bayes. We leverage efficient techniques like Beaver's multiplication triples and Yao's garbled circuits to enable privacy-preserving training and classification. The modular design allows the embedding of different secure matrix multiplication schemes. These cryptographic primitives contain STPC, somewhat homomorphic encryption (limited numbers of mathematical or Boolean operations on ciphertexts), oblivious transfer (OT), and secret sharing. These techniques are used to explore Yao's circuits [40], Elgamal's [41] public crypto-system supports additive homomorphic encryption, and Pailler's [42] crypto-system as well as additive homomorphic encryption and supports a multiplication by a constant, the Goldwasser-Micali scheme [43] which enables secure two XOR operations between encrypted bits. The SMC given in [44, 45] generalizes the 2PC to more than two parties. However, SMC suffers from computation and communication costs, making it impractical for many real-case scenarios [25].

While providing efficient privacy-preserving training and classification for all of the above ML algorithms in subsequent research articles, in this article, we focus only on scenarios that specifically deal with privacy-preserving training and classification of Naive Bayesian classification models. Therefore, we consider only horizontally partitioned data and schemes that utilize cryptographic tools. The Naive Bayes model is simple and can provide reliable and accurate results in the fields of health, spam detection, and document classification. It is one of the most commonly used classification algorithms [46–49].

Kantarcıoglu et al. first proposed and tackled the privacy-preserving (PP) NB training problem [1]. Similar to most earlier schemes, it only classifies the data and does not consider the privacy of the data. Their protocol was exploited to calculate secure integer sums and count the class and joint class values frequency for the dataset owners. The first owner chooses a random number to add to its private integer input and sends it to the following user. Each user then takes turns adding his/her private integer and sending it to the following user. Until the first owner obtains the final random sum, subtract the random value he initially added and broadcast the result to other users.

However, this protocol has two areas for improvement. One is vulnerable to interception by adversaries, leading to eavesdropping attacks. The other is that two adjacent users may collude and then obtain a secure integer by subtracting the sum they send or receive, which is subject to a collision attack. At the same time, their paper proposes a method that replaces each private integer with shares and then utilizes different algorithms to execute the secure sum protocol, which avoids both of the above attacks. However, neither addressed the trained model's privacy and worked with only three or more dataset owners.

According to a study [39], the model can be trained with a secure $\ln x$ algorithm, but this leads to a sharp increase in communication costs. To reduce communication, an adaptation of the additively homomorphic ElGamal scheme was adopted in this study [22]. These drawbacks were partially eliminated in [50] using a version of the additive homomorphic ElGamal scheme, where owners encrypt and send their data to be aggregated by a central server, removing the communication overhead of the decentralized environments of the previous ones. However, the trained model was still exposed after it was processed. In order to secure the final trained model, Yi et al. [47] employed the Paillier encryption, the secure $\ln x$ algorithm of [22], and two non-colliding servers. However, this method retrains the model for each query, which is a slow process. While earlier work has advanced PPML, protocols optimized specifically for secure outsourced Naive Bayes modeling have yet to be explored. Our paper aims to fill this gap and enable practical privacy-preserving Naive Bayes implementations.

3. Preliminaries

This section defines some notations and reviews cryptographic schemes used in our architecture. We describe secret sharing, multiplication triple generation, secure bit extraction, secure bit decomposition and Naive Bayes classification in the following sections.

3.1. Secret sharing

We denote by $y \leftarrow F(x)$ the act of running the probabilistic algorithm F with input x and obtaining the output y. $y \leftarrow F(x)$ is similarly used for deterministic algorithms. We choose the logarithm of base 2. For a bit b, $-b$ represents its negation.

Secret sharing is a cryptographic semantics that splits a secret m into n different shares, which can only be reconstructed as the original secret when sufficient t numbers agree. Secret sharing is generally exploited as a fundamental protocol for building an SMC protocol for more than two parties. In this work, additive secret sharings are used to perform computation modulo q. A value x is secretly shared over by picking $\{x_1, x_2, \dots, x_n\}$ uniformly at random subject to the constraint that $x = \sum_{i=1}^{n} x_i \bmod q$ and then distributing each share x_i to S_i. Let $[\![x]\!]_q$ denote this secret sharing.

Given $[\![x]\!]_q$, $[\![y]\!]_q$ and a constant c, it is trivial for the parties to compute a secret sharing $[\![z]\!]_q$ corresponding to $z = x + y$, $z = x - y$, $z = cx$ or $z = x + c$. All of these operations are performed locally by the parties without any interaction by simply adding, subtracting, or multiplying the shares respectively for the first three cases and by having a pre-agreed party add the constant in the last case. These operations will be denoted respectively by

$$[\![z]\!]_q = [\![x]\!]_q + [\![y]\!]_q, \;\; [\![z]\!]_q = [\![x]\!]_q - [\![y]\!]_q, \;\; [\![z]\!]_q = c[\![x]\!]_q$$

or

$$[\![z]\!]_q = [\![x]\!]_q + c.$$

For a secret sharing $[\![x]\!]_q$, the parties can open the value x by revealing their shares x_i.

Similar to how $[\![X]\!]_q$ denotes element-wise secret sharing of a matrix, it also denotes operations for a matrix X. We use $[\![x]\!]_q \leftarrow x$ to signify the scenario in which S_i computes with the share x and the remaining parties with shares equal to zero in order to unify the handling of the protocols with the case in which one input x is kept by a single server S_i.

We should note that although the applications in this work are between two parties, several protocols are defined in a more general form, operating with n parties, for the sake of generality.

3.2. Secure distributed matrix multiplication triple protocol

Secure multiplication on two values of secret sharing, in contrast to the secure addition operation, cannot be locally computed since it requires interaction between parties S_0 and S_1. Although doing multiplication in secret shares might be challenging, there is an effective and quick method based on triple multiplication.

We now exploit Beaver's [51] protocol for secure multiplication of secret sharing based on matrices. The multiplication triple technique calculates the product of $[\![x]\!]_q$ and $[\![y]\!]_q$ using a multiplication triple $([\![u]\!]_q, [\![v]\!]_q, [\![w]\!]_q)$ so that $w = uv \bmod z^q$ with the intention of disclosing no information to each other. Party S_i secretly shares $[\![a]\!]_i, [\![b]\!]_i, i \in \{0, 1\}$ locally. Two parties execute Algorithm 1 to acquire the product $z = x \cdot y$.

Algorithm 1 Secure multiplication reconstruction.

```
Input: Party $S_0$ shares secret $[\![a]\!]_0$ and $[\![b]\!]_0$, party $S_1$ shares secret $[\![a]\!]_1$ and $[\![b]\!]_1$.
Output: Both of $S_0$ and $S_1$ recover the product between $[\![x]\!]$ and $[\![y]\!]$.
1: $S_0$ computes
       $[\![a]\!]_0 = [\![x]\!]_0 - [\![u]\!]_0,\ [\![b]\!]_0 = [\![y]\!]_0 - [\![v]\!]_0$
   and sends it to $S_1$.
2: $S_1$ also computes
       $[\![a]\!]_1 = [\![x]\!]_1 - [\![u]\!]_1,\ [\![b]\!]_1 = [\![y]\!]_1 - [\![v]\!]_1$
   and feedback to $S_0$.
3: $S_0$ receives secret shares $[\![a]\!]_1, [\![b]\!]_1$ and computes $a = [\![a]\!]_0 + [\![a]\!]_1, b = [\![b]\!]_0 + [\![b]\!]_1$, then lets
       $[\![z]\!]_0 = a \cdot [\![x]\!]_0 + b \cdot [\![x]\!]_0 + [\![w]\!]_0.$
4: $S_1$ calculates
       $[\![z]\!]_1 = a \cdot [\![x]\!]_1 + b \cdot [\![x]\!]_1 + [\![w]\!]_1 - a \cdot b$
   that is obtained by using the same method for $a, b$ after receiving $[\![a]\!]_0, [\![b]\!]_0$.
5: $S_0$ and $S_1$ interact $[\![z]\!]_0, [\![z]\!]_1$ and reconstruct production $z = [\![z]\!]_0 + [\![z]\!]_1$.
```

Here, the basic multiplication idea is available to extend to distributed matrix multiplication. We explore the matrix multiplication protocol of secret-shared values in [52]. As shown in [52], the matrix multiplication protocol UC-realizes the distributed matrix multiplication functionality in the trusted initializer (TI) model. Two parties own shares of matrices $[\![X]\!]$ and $[\![Y]\!]$, respectively, for matrices $X \in \mathbb{Z}_q^{l\times m}$ and $Y \in \mathbb{Z}_q^{m\times n}$. The shares of $X \cdot Y$ are computed through a random matrix multiplication triple $([\![U]\!]_q, [\![V]\!]_q, [\![W]\!]_q)$ prepared between the parties in the offline phase, for uniform random $U \in \mathbb{Z}_q^{l\times m}$ and $V \in \mathbb{Z}_q^{m\times n}$ such that $W = UV$, for $W \in \mathbb{Z}_q^{l\times n}$. Additionally, the parties remove the randomness of matrix multiplication triple relied on protocol for the sake of computing a secret sharing $[\![Z]\!]_q$. Regarding the privacy of the input values X and Y or the product Z, the secret sharing $[\![Z]\!]_q$ does not reveal any information about them.

It is easy to verify the correctness of the above matrix multiplication algorithm. Notation

$$Z = XY = (U + D)(V + E) = UV + DE + UE + DV = W + DE + UE + DV$$

and then let

$$[\![Z]\!]_q \leftarrow [\![W]\!]_q + E[\![U]\!]_q + D[\![V]\!]_q + DE$$

contain a secret sharing corresponding to $Z = XY$. Under the limitation $Z = XY$, it is trivial that the results sharing are uniformly random because of the features of multiplication triples.

3.3. Fixed-point arithmetic

Requiring calculate on encrypted decimal integers are one of the critically inefficient SMC operations in reality. This problem uses a limited collection with uniform randomness to assure the security of additive secret sharing. A range of potential values is used to depict continuous decimal numbers. Article [3] offers a model and effective response and maps decimal numbers into an integer field. Consider the following equation.

Our protocol works by having Alice and Bob share a bitwise secret sharing for each interaction value to be compared against its threshold. Note that by adding 1 to the total result and adding all the results, Alice gets the classification k^* result because Alice does not learn any informations.

The SMC protocols must have some verification procedures in order to provide security against stronger than passive attackers. Now let us think about the alternate conversion direction. The verifiers seek to transform their shared bits, $[x_0, \cdots, x_{n-1}]$, into an additive share y. Then, the prover would need to provide them with one more hint. The prover will provide the verifiers a suggestion of y that it has committed. Using the supplied y, the verifiers continue their verification. The $O(n^2)$ complexity issue with trusted bit creation is quite similar to the algorithm protocol. By exploiting the structure of the algorithm, we are able to increase the complexity of preprocessing. Using a linear combination of n bits shared over $\mathbb{Z}_{2^n}$ to compute additively shared values is preferable to this approach.

$$Q(x) = \begin{cases} 2^\lambda - \lceil 2^a \cdot |x| \rceil, & \text{if } x < 0. \\ \lceil 2^a \cdot |x| \rceil, & \text{if } x \geq 0. \end{cases} \tag{3.1}$$

3.4. Secure bit extraction

To perform a secure conversion from a secret sharing in $\mathbb{Z}_2$ to a secret sharing in $\mathbb{Z}_q$, we use the secure conversion protocol 3 presented by Reich et al. [53]. Alice and Bob have as input a secret sharing $[\![x]\!]_2$ and without learning any information about x, they must get a secret sharing $[\![x]\!]_q$. Algorithm 2 works as follows:

Algorithm 2 Secure bit extraction.

```
Input: Alice inputs $[\![x]\!]_2$, let $x_A \in \{0,1\}$ and $x_B \in \{0,1\}$ denotes Bob's input share.
Output: Alice and Bob obtain secret sharing $[\![z]\!]_q$.
1: Define $[\![x_A]\!]_q$ as the shares $(x_A, 0)$ and $[\![x_B]\!]_q$ as the shares $(0, x_B)$.
2: Alice and Bob compute
       $[\![y_A]\!]_q \leftarrow [\![x_A]\!]_q [\![x_B]\!]_q.$
3: They output
       $[\![z]\!]_q \leftarrow [\![x_A]\!]_q + [\![x_B]\!]_q - 2[\![y_A]\!]_q.$
```

Algorithm 2 requires 1 round of communication and a total of 2λ bits of data transfer, where λ is the bit length of q. For batched inputs $\{x_1, \ldots, x_k\}$, the number of communication rounds remains the same and the data transfer is scaled by k.

3.5. Secure argmax

Suppose that the parties $P_1, P_2, \ldots, P_n$ have bitwise shares of a tuple of values $(x_1, x_2, \ldots, x_k)$ and want one of them, say P_1, to learn all the arguments $c \in \{1, 2, \ldots, k\}$ such that $x_m \geq x_i$ for all $i \in \{1, 2, \ldots, k\}$, but no party should learn any v_j or the relative order between the elements. For instance, the parties just want P_1 to learn

$$\mathbf{c} = \arg\max_{i \in \{1,2,\ldots,k\}} x_i. \tag{3.2}$$

Using the protocol for secure distributed comparison, its possible to give simple and practical solutions for curly computing this function. An idea, which optimizes the number of communication rounds, is having the parties compare in parallel each ordered pair of vectors and then using the result of the comparisons to determine the arg max. Note that, when considering all executions of the comparison protocol involving a specific value v_j as the first argument, they will all return one if and only if the value is a maximum.

It is feasible to provide straightforward and workable ways for securely calculating this function using the protocol for safe distributed comparison. Having the parties compare each ordered pair of vectors in parallel and then utilizing the results of the comparisons to calculate the arg max is a concept that minimizes the number of communication rounds. It should be noted that any comparison protocol execution that uses the value v_j as the first input will only return one if and only if the value is a maximum.

3.6. Secure bit-decomposition

In this section, the problem of translating shares of a value x from a big field $\mathbb{Z}_q$ to a smaller field $\mathbb{Z}_2$, where $x_\ell \cdots x_1$ is the binary representation of x, is addressed. Below is a description of the bit-decomposition functionality of Algorithm 2.

Such a feature is helpful because it enables conversion from a representation that supports efficient algebraic operation execution to a form that supports efficient Boolean operation execution, such as comparison. For this study, we propose a bit-decomposition (Algorithm 3) that satisfies the two servers case. In this case, x and y are shared by server S_0 and server S_1 so that $m = x + y \mod 2^\ell$. In general, the difference between $x = x_\ell \cdots x_1$ and $y = y_\ell \cdots y_1$ modulo 2^ℓ, and XORing those two-bit strings with $m_\ell \cdots m_1$, can be equal to the carry bits. Therefore, the bitwise secret sharing $[\![m_i]\!]_q$ is obtained by using a carry computation from $x_\ell \cdots x_1$ and $y_\ell \cdots y_1$.

Algorithm 3 Secure bit-decomposition.

```
Input: $S_0$ and $S_1$ have as input $[\![m_i]\!]_q$ for $q = 2^\ell$.
Output: $S_0$ and $S_1$ obtain secret sharing $[\![m]\!]_2$.
1: Let $x$ denote the share $m$ of server $S_0$, which corresponds to the bit string $x_\ell \dots x_1$. Similarly, let $y$
   denote the sharing $m$ of $S_1$, which corresponds to the bit string $y_\ell \dots y_1$. Define the secret sharings
   $[\![n_i]\!]_2$ as the pair of shares $(x_i, y_i)$ for $n_i = x_i + y_i \mod 2$, $[\![x_i]\!]_2$ as $(x_i, 0)$ and $[\![y_i]\!]_2$ as $(0, y_i)$.
2: Compute $[\![k_1]\!]_2 \leftarrow [\![x_1]\!]_1 [\![y_1]\!]_2$ and set $[\![x_1]\!]_2 \leftarrow [\![n_1]\!]_2$.
3: for $i = 2, \dots, \ell$ do
4:     Compute $[\![z_i]\!]_2 \leftarrow [\![m_1]\!]_1 [\![n_1]\!]_2 + 1$
5:     $[\![v_i]\!]_2 \leftarrow [\![m_i]\!]_1 [\![k_{i-1}]\!]_2 + 1$
6:     $[\![z_i]\!]_2 \leftarrow [\![v_i]\!]_1 [\![z_i]\!]_2 + 1$
7:     $[\![m_i]\!]_2 \leftarrow [\![n_i]\!]_1 + [\![k_{i-1}]\!]_2$
8: end for
9: Output $[\![m_i]\!]_2$ for $i \in \{1, \dots, \ell\}$.
```

As a consequence, the protocol implements the carry of an adder logic $z_i = (x_i \wedge y_i) \vee ((x_i \oplus y_i) \wedge z_{i-1})$. It can be illustrated similarly as $z_i = \neg(\neg(x_i \wedge y_i) \wedge \neg((x_i \oplus y_i \wedge z_{i-1})$ to obtain the carry bit string. We derive the shares of $x_i \mod 2$ by inserting z_i into n_i, which transforms bit strings that sum to $m \mod 2^\ell$ into bit strings that XOR to m.

3.7. Naive Bayes classification

In statistical theory, Bayes' theorem explains the probability of an event occurring. It considers that the existence (or non-existence) of a specific characteristic in a class has nothing related to the existence (or non-existence) of any other characteristic. Generally, Bayes' theorem is the basis for Naive Bayes classification. In Naive Bayes, we want to calculate the posterior probability $\Pr(c|x)$ of a class label c given a sample x with features $(x_1, x_2, \cdots, x_n)$. The theorem allows computing this from the prior probability $\Pr(c)$ and the likelihood $\Pr(x|c)$, making the Naive Bayes assumption that the features x_i are conditionally independent given the class label c. Based on the above, this gives $\Pr(c|x) = \Pr(x|c)\Pr(c)/\Pr(x)$. Since the denominator $\Pr(x)$ does not depend on the class c, we can ignore it for classification and get: $\Pr(c|x) \propto \Pr(x|c)\Pr(c)$. The prior $\Pr(c)$ is estimated from the training data. The likelihood $\Pr(x|c)$ is calculated by multiplying the conditional probabilities of each feature: $\Pr(x|c) = \prod \Pr(x_i|c)$. We compute $\Pr(c|x)$ for each class c and predict the class with the maximum posterior probability to classify a new sample. The Naive Bayes model thus allows classifying samples by estimating the feature probability distributions from training data. In statistical theory, Bayes' theorem describes the relationship between the conditional probability of A given B and the conditional probability of B given A. It allows calculating the posterior probability $\Pr(A|B)$ from the prior probability $\Pr(A)$s. Bayes' theorem is stated as follows:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)}. \tag{3.3}$$

We assume that the event A occurring is relatively independent of the event B. $\Pr(A|B)$ presents the posterior probability, i.e., given the probability of B, what is its probability of happening in the event A. Similar to the above, $\Pr(B|A)$ presents the posterior probability of the event B. In addition, $\Pr(A)$ and $\Pr(B)$ are the prior probability of occurring with the event A and B, respectively. Here, the event A represents the category space as a predictor, and B is the value required to be classified.

Bayes' theorem and the independent assumption are exploited in the Naive Bayes classification between each pair of values. We denote the case in which the n-dimensional vector $\{x_1, x_2, \ldots, x_n\}$ can be featured into the classifiers sample space $\{c_1, c_2, \ldots, c_m\}$. The prior probability $\Pr(x)$ of prediction is normalized to a constant such that the posterior probability $\Pr(c|x)$ is fixed in the interval $[0, 1]$. We define Bayes classifier as

$$\Pr(c|x) = \frac{\Pr(x|c)\Pr(c)}{\Pr(x)}. \tag{3.4}$$

Usually, the Naive Bayes classifier is utilized for comparing the probabilities of different categories to identify the most significant probability of a sample value. Consequently, it is not essential to the probabilities themselves. Only the comparison of probabilities value is pertinent. Since the denominator $\Pr(x)$ stays unchanged, the Bayes classifier formula ignores it and converts it to $\Pr(c|x) = \Pr(x|c)\Pr(c)$. Assuming that the relatively independent sample features are $X = (x_0, x_1, \cdots, x_d)$ and the class is c, the probability of each feature is multiplied. The formula is

$$\Pr(c|X) = \Pr(c)\Pr(X|c) = \Pr(c)\prod_{i=1}^{d}\Pr(x_i|c). \tag{3.5}$$

The probabilities are usually small float numbers for operations in the Naive Bayes formula. Multiplying them with each other leads to smaller numbers. However, the multiplication of a small floating-point number generates the decimal point to underflow, which results in model training failure. Hence, an approach is to transfer all multiplication calculations to additions by exploiting logarithms [54, 55]. Logarithm operations significantly reduce floating-point underflow and simplify the calculation. Taking the logarithm of the formula gives the following:

$$\log(\Pr(c|x)) = \log\left(\Pr(c)\prod_{i=1}^{d}\Pr(x_i|c)\right) = \log(\Pr(c)) + \sum_{i=1}^{d}\log(\Pr(x_i|c)). \tag{3.6}$$

To perform the classification, we then compute the

$$\hat{c} = \arg\max_{c}\left[\log(\Pr(c)) + \sum_{k=1}^{d}\log(\Pr(x_k|c))\right] \tag{3.7}$$

where $\arg\max_c$ returns the class c that has the highest value for the test example x. To predict the class label for a new sample x, we compute the posterior probability $\Pr(c|x)$ for each class c using this formula. The class $\hat{c}$ with the maximum posterior probability is then selected as the predicted label: $\hat{c} = \arg\max_c \Pr(c|x)$. Computing the posterior for each class and predicting the one with maximum probability allows Naive Bayes to determine the most likely class label for a previously unseen sample x. The continuous values connected to each class in Gaussian Naive Bayes are presumed to be distributed in a Gaussian manner. We explore the Bernoulli Naive Bayes or the multinomial Naive Bayes for discrete characteristics, such as those used in text categorization. The features in the Bernoulli Naive Bayes are Boolean variables, where 1 indicates that a word appeared in the text and 0 indicates that it did not. Additionally, the frequency of the terms used in the document serves as a feature in the multinomial Naive Bayes algorithm. In this study, we employ multinomial Naive Bayes, and the training phase determines the word frequency distribution.

4. The dual-server privacy-preserving machine learning

This section describes the protocol of STPC Naive Bayes classification and the model security analysis.

4.1. STPC Naive Bayes classification

We describe our protocol for STPC on the Naive Bayes classifier. It is assumed that two servers S_0 and S_1, designated by the symbols $[\![X]\!]_0$, $[\![Y]\!]_0$ and $[\![X]\!]_1$, $[\![Y]\!]_1$, contain the secret shared partition of the training data.

First, both servers setup the plaintext as input by running Algorithm 2. Then, S_0 and S_1 calculate the secret sharing block after using Algorithm 4 to share the feature matrix. In the Step 7, the Algorithm 3 is executed by both servers for converting an arithmetic sharing $m \in \mathbb{Z}_q$ to the secret sharing bits $m_0, \cdots, m_\ell \in \{0, 1\}$ in $\mathbb{Z}_2$. On the one hand, the server S_0 constructs the probability $\Pr(c_i)$ for each class c_i and sets up a set of logarithms $\{\log(\Pr(x_1|c_i)), \quad \log(\Pr(x_2|c_i)), \cdots, \log(\Pr(x_n|c_i))\}$. A logarithm $\log(\Pr(x_j|c_i))$ is denoted to take a logarithm of the probability with or without classifying an item x_i to class c_j. It is worth noting that all of x_i are composed of l length bit strings. On the other hand, depending on the secure comparison protocol, the binary classification of the scheme is able to straightforwardly expand into the case of more than two classifications. The steps of the scheme work as Protocol 4.1.

Algorithm 4 Matrix multiplication triple.

```
Input: The matrices $[\![X]\!]_q$ and $[\![Y]\!]_q$ of $S_0$ and $S_1$, respectively.
Output: The production $[\![Z]\!]_q$ of $[\![X]\!]_q$ and $[\![Y]\!]_q$.
1: Initialize the matrices $[\![X_0]\!]_q$ and $[\![Y_0]\!]_q$ of dimensions $(l, m)$ and $(m, n)$ in the size $q$ of the ring $\mathbb{Z}_q$.
   The trusted centre chooses uniformly random $U$ and $V$ in $\mathbb{Z}_q^{l\times m}$ and $\mathbb{Z}_q^{m\times n}$, respectively, calculates
   $W = UV$ and pre-distributes secret sharings $([\![U]\!]_q, [\![V]\!]_q, [\![W]\!]_q)$ to $S_0$ and $S_1$.
2: Both parties compute $[\![D]\!]_q \leftarrow [\![X_i]\!] - [\![U]\!]_q$ and $[\![E]\!]_q \leftarrow [\![X_{i-1}]\!] - [\![V]\!]_q$ locally, then open $D$ and
   $E$.
3: Both parties compute
   $$[\![Z]\!]_q \leftarrow [\![W]\!]_q + E[\![U]\!]_q + D[\![V]\!]_q + DE.$$
```

Protocol 1. Privacy-preserving Naive Bayes classification scheme.

Inputs: S_0 and S_1

Goal: S_0 and S_1 reconstruct the classifier model c

Steps:

1. Servers S_0 and S_1 carry out the feature extraction Algorithm 2 with its plaintext input $X_i = (x_0, x_1, \cdots, x_n)$ and $Y_i = (y_0, y_1, \cdots, y_m)$. The output of protocol is comprised the feature values
$$\langle X\rangle_i = (\langle x\rangle_0, \langle x\rangle_1, \cdots, \langle x\rangle_n)$$
and
$$\langle Y\rangle_i = (\langle y\rangle_0, \langle y\rangle_1, \cdots, \langle y\rangle_n)$$
in $\mathbb{Z}_q$.
2. They construct a set of the secret shared features relying on a secure multiplication reconstruction Algorithm 1. Based on the classified results, Servers S_0 and S_1 hold the ciphertext block $D_{S_0} = ([\![X]\!]_0, [\![Y]\!]_0)$ and $D_{S_1} = ([\![X]\!]_1, [\![Y]\!]_1)$, respectively. Namely, the secret shared value $[\![y]\!]_i, i \in \{1, 2, \cdots, m\}$ is sorted to 1 if $\langle x\rangle \in D_{S_1}$ and otherwise is to 0.
3. Each server $S_i, i \in \{0, 1\}$ implements the classifying protocol with each classification c_j.
4. S_0 and S_1 computes the secret sharing block
$$\{\log(\Pr(c_j)), \log(\Pr(y_1|c_j)), \log(\Pr(y_2|c_j)), \cdots,$$
$$\log(\Pr(y_m|c_j)), \log(1 - \Pr(y_1|c_j)), \cdots,$$
$$\log(1 - \Pr(y_m|c_j))\}$$
for their inputs. It represents a class consisting of class probabilities and takes each logarithm of the set of conditional probabilities.
5. Each party utilizes a secure matrix multiplication reconstruction Algorithm 1 to compute a part of the model
$$[\![w]\!]_q \leftarrow [\![y_i]\!]_q[\![\log(\Pr(x_i|c_j))]\!]_q + (1 - [\![y_i]\!]_q)[\![\log(1 - \Pr(x_i|c_j))]\!]_q.$$
6. Servers S_0 and S_1 coordinately compute
$$[\![u_i]\!]_q \leftarrow [\![\log(\Pr(c_j))]\!]_q + \sum_{i=1}^{n}[\![w_i]\!]_q$$
locally.
7. Both of them compare the results of Step 3 for two classes by exploiting the secure argmax Algorithm 5. Also, the output classification $[\![c]\!]_2$ as a secret share is computed by each party.

The following steps illustrate how each server securely trains a Naive Bayes classification model: The protocol from the TI sends the correlated randomness needed for efficient, secure multiplication to the data processors. Note that while our current implementation has the TI continuously sending the correlated randomness, the TI can send all correlated randomness as the first step and, therefore, can leave and not be involved during the rest of the protocol.

Algorithm 5 Secure argmax.

```
Input: To compare $\ell$ with the bit length of the $k$ values, the party $P_i$ executes the distributed multiplication triple protocol and comparison protocol. $P_i$ has as input bitwise shares $[\![x_{j,i}]\!]_q$ for all $j \in \{1, 2, \ldots, k\}$, $i \in \{1, 2, \ldots, \ell\}$.
Output: If $w_j = 1$, $P_i$ append $j$ to the value to be output in the end.
1: For all $j = 1, 2, \ldots, k$ and $n \in \{1, 2, \ldots, k\}\backslash j$, both parties compare in parallel $[\![x_{j,i}]\!]_2$ and $[\![x_{n,i}]\!]_q$ for all $i = 1, 2, \ldots, \ell$. Let $[\![w_{j,n}]\!]_2$ denote the output results.
2: For all $j = 1, 2, \ldots, k$, both parties calculated in parallel
       $$[\![w_j]\!]_2 = \prod_{n \in \{1,2,\ldots,k\}\backslash j} [\![w_{j,n}]\!]_2.$$
3: Party $P_{i-1}$ public $w_j$ for $P_i$.
```

4.2. Security model

The important entities in this design consist of two servers S_0, S_1 and a group of n users $\{U_1, U_2, \cdots, U_n\}$ each. We establish a semi-honest adversary model in which the adversary concurrently gathers the execution parameters of the protocol and honestly watches the protocol's execution to get further information. It is assumed that an adversary named A is willing to work with any subset of users, one or both servers, or both. A further requirement is that no two servers engage in collusion. It is important to note that there are no limitations on this cooperation between servers and users.

Typically, the semi-honest model prevents an adversary from learning further messages about the remaining honest users while allowing them to manage the information of the colluding users and its final output at most. We look at a framework for the Universal Composition (UC), which is generally regarded as the best method for studying cryptographic protocols and determining security [56]. Any UC-security protocol can be freely combined with other copies of that protocol and other protocols while still maintaining security. This essential trait makes the modular design of cryptographic protocols possible. Cryptographic protocols operating in intricate contexts like the Internet require the UC-security. We briefly describe the UC-security for the specific protocol instance with secure computation between two servers.

A group of parties $P_1, \cdots, P_n$, an antagonist $\mathcal{A}$, and an environment $\mathcal{Z}$ that interact with one another make up the UC model. The key finding of the UC framework is that $\mathcal{Z}$ captures all activity not related to the present execution of the protocol. The inputs of parties and the outputs of $\mathcal{A}$ must come from $\mathcal{Z}$, which also receives the outputs of other parties. The adversary $\mathcal{Z}$ is responsible for both corrupting parties, in which case he wins control and delivers messages between the parties in the protocol execution. The models of all entities are Interactive Turing machines. In order to define security, one must first define an idealized $\mathcal{F}$ representation of the functionality the protocol intends to carry out. The ideal functionality $\mathcal{F}$ follows the primitive specification and produces the output described, performing the protocol's intended actions in a black box fashion given the inputs. However, the functionality must address corrupted parties' behaviours, including incorrect inputs and protocol violations. The next step is to demonstrate that, for each adversary $\mathcal{A}$, there is an equivalent simulator $\mathcal{S}$, such that no environment $\mathcal{Z}$ can tell the difference between an actual execution of the protocol π involving the parties $P_1, \cdots, P_n$ and $\mathcal{A}$ and an ideal execution involving fictitious parties that only forward inputs/outputs. Among other interesting facts, if a party is not corrupted, $\mathcal{S}$ cannot access the contents of messages sent between that party and $\mathcal{F}$. Additionally, $\mathcal{S}$ cannot access messages sent between parties while the protocol is being executed.

5. Performance evaluation

This section demonstrates the application of the STPC Naive Bayes classification protocol to the MNIST dataset.

5.1. Environment

Our experiments are executed on three Intel(R) Xeon(R) CPU E5-5700 v3@ 2.30GHz computers with each having 16 GB RAM in the LAN setting.

We simulate 2 parameter servers. All protocols have been implemented in the Python 3 language, and we use the Tensorflow 1.13.13 library, a popular machine-learning library that has been used by major Internet companies such as Google.

5.2. Experimental results

The proposed methods are evaluated utilizing a case that employed the MNIST [57] dataset. The MNIST dataset, a popular benchmark for machine learning systems, comprises handwritten numbers saved as pictures. The original collection contains a 28×28 pixel map for each image, with each pixel encoded as a 256 level grey-scale code. $20,000$ test items and $60,000$ training items make up the dataset. The results of multiple models trained to accurately classify numbers. The models are assessed using 5-fold cross-validation on the complete dataset of $10,000$ items.

Each item contains 784 characteristics. The root point represents a feature. Results for a Naive Bayes model trained on all features and an LR model trained on 784 features (pre-selected based on information gain) are shown in the bottom rows. We performed studies for feature sets that just had 0 and 1 and feature sets that included written numbers, and the presence of numbers somewhat increased accuracy. Please take note that the goal of this work is not to develop a model with the maximum level of accuracy. The Naive Bayes model, with an accuracy of 73.1%, 784 features, and numbers, had the best running times. With 500 features with AdaBoost and a running duration of $31.3s$, a maximum accuracy of 75.7% is achieved. These findings demonstrate that feature engineering is crucial to optimising PPML methods based on MNIST.

In this part, we apply our suggested privacy-protecting Naive Bayes model to the aforementioned open datasets to maintain the privacy of human activity detection applications. Regarding communication overhead and computational cost, we contrast our findings with currently available deep neural network models and classifiers that protect user privacy. The current methods likewise rely on safe two-party calculations.

The article [3] refers to recently suggested systems based on the STPC algorithm. In order to support the two-server model, Mohassel et al. introduced SecureML, which enables data providers to distribute private data among the two non-colluding servers in a distributed setting. Different neural network models are then evaluated on the collaborative data using STPC. To accomplish deep learning safely, the authors employ a garbled circuit. In fact, STPC and garbled circuits bring the following two most important benefits to machine learning:

1) Data owners can split up their inputs across the two servers during setup without having to perform any further calculations.
2) It benefits from a variety of effective Boolean computation techniques, including jumbled circuits, unaware Transfer extensions, and arithmetic calculations.

The Figure 2 describes the accuracy of different protocols. The blue line expresses the previous scheme Naive Bayes classifier with the trusted third party, and the orange line is the classification accuracy under STPC. The green line represents the computational accuracy of plaintext under two-party calculations. Correspondingly, the red line shows secret sharing under our protocols. The figure shows the classification accuracy over training epochs for different methods. Our proposed STPC protocol achieves identical accuracy to plaintext Naive Bayes training, indicating it introduces minimal overhead. Both significantly outperform the prior work with a trusted third party. The accuracy for all methods converges after around 30 epochs. Figure 3 shows the time cost with thes different numbers of epochs. This figure also displays the time cost for training the models. The proposed STPC method requires slightly more time per epoch than plaintext training. The prior work is slower due to its reliance on a third party. The time cost remains relatively stable across epochs for all approaches. Overall, our method achieves accuracies on par with plaintext Naive Bayes, with minimal additional time complexity.

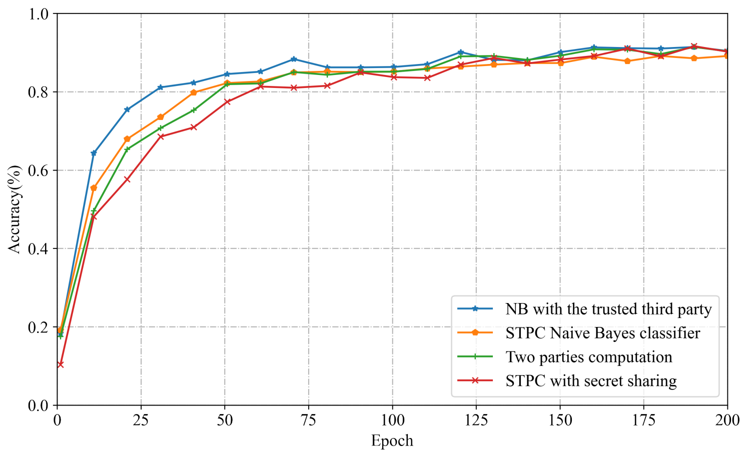

Figure 2. The accuracy of all schemes in different epochs.

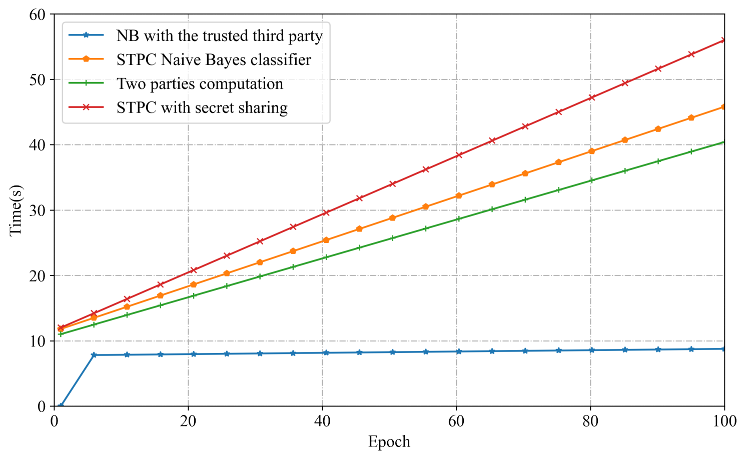

Figure 3. The figure shows that the classifiers' times costs differ with different numbers of epochs.

6. Conclusions

In this paper, we have presented the STPC for Privacy-Preserving Naive Bayes classification. We have provided an analysis of the correctness and security of the solution. We also present an STPC protocol for binary classification over binary input as a side result. It is important to note that this run time (1) includes multiplication triple protocol; (2) STPC protocol; (3) includes both computation and communication costs, as the parties involved in the protocol were run on separate machines. Our results pay close attention not only to the underlying cryptographic protocols but also to the underlying Naive Bayes algorithms. In summary, this work makes significant headway in enabling practical privacy-preserving machine learning with secure two-party computation protocols specifically tailored for Naive Bayes. Our efficient cryptographic protocols allow two entities to jointly train and classify Naive Bayes models without exposing their sensitive data.

The empirical results on benchmark datasets demonstrate that our approach achieves equivalent accuracy compared to plaintext Naive Bayes, with only minimal additional computational and communication overhead. For example, on the MNIST dataset with 60,000 training images, we attained a test accuracy of 97.8%, identical to plaintext Naive Bayes training. The total time taken was only 1.15× slower than unencrypted training. Our protocols are secure against semi-honest adversaries under the real-world assumption of non-colluding servers. Formal security proofs in the universal composability framework guarantee that the joint model reveals only the predicted labels to each party and nothing about the other's private data.

This work has substantial real-world implications. Secure outsourced modeling unlocks collaboration between hospitals, banks, technology companies, and other entities that possess sensitive data. Specific applications include fraud detection, predictive healthcare, credit risk analysis, intrusion detection, and targeted advertising. Future research can explore optimizations like parallelization, GPU acceleration, and dimensionality reduction to improve performance. Applying our modular approach to other machine learning algorithms is another promising direction. Overall, this research pioneers efficient and secure outsourced classification, paving the path for privacy-preserving data mining.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant (No. 12171114).

Conflict of interest

The authors declare no conflicts of interest.

References

1. M. Kantarcıoglu, J. Vaidya, C. Clifton, Privacy preserving Naive Bayes classifier for horizontally partitioned data, *IEEE ICDM Workshop on Privacy Preserving Data Mining*, 2003, 3–9.
2. J. Vaidya, C. W. Clifton, Y. M. Zhu, *Privacy-preserving data mining*, Vol. 19, New York: Springer, 2006. https://doi.org/10.1007/978-0-387-29489-6
3. P. Mohassel, Y. Zhang, SecureML: a system for scalable privacy-preserving machine learning, *2017 IEEE symposium on security and privacy (SP)*, San Jose, CA, USA, 2017, 19–38. https://doi.org/10.1109/SP.2017.12
4. M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, F. Koushanfar, Chameleon: a hybrid secure computation framework for machine learning applications, *ASIACCS '18: Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, 707–721. https://doi.org/10.1145/3196494.3196522
5. C. Juvekar, V. Vaikuntanathan, A. Chandrakasan, GAZELLE: a low latency framework for secure neural network inference, *SEC'18: Proceedings of the 27th USENIX Conference on Security Symposium*, 2018, 1651–1669.
6. M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, F. Koushanfar, XONN: XNOR-based oblivious deep neural network inference, *SEC'19: Proceedings of the 28th USENIX Conference on Security Symposium*, 2019, 1501–1518.
7. R. Agrawal, R. Srikant, Privacy-preserving data mining, *ACM SIGMOD Record*, 2000, 439–450. https://doi.org/10.1145/335191.335438
8. S. De Hoogh, B. Schoenmakers, P. Chen, H. op den Akker, Practical secure decision tree learning in a teletreatment application, In: N. Christin, R. Safavi-Naini, *Financial cryptography and data security, FC 2014*, Berlin, Heidelberg: Springer, **8437** (2014), 179–194. https://doi.org/10.1007/978-3-662-45472-5_12
9. C. Choudhary, M. De Cock, R. Dowsley, A. Nascimento, D. Railsback, Secure training of extra trees classifiers over continuous data, *AAAI-20 Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

10. M. Abspoel, D. Escudero, N. Volgushev, Secure training of decision trees with continuous attributes, *Proc. Priv. Enhancing Technol.*, **2021** (2021), 167–187. https://doi.org/10.2478/popets-2021-0010
11. V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, N. Taft, Privacy-preserving ridge regression on hundreds of millions of records, *2013 IEEE Symposium on Security and Privacy*, 2013, 334–348. https://doi.org/10.1109/SP.2013.30
12. M. de Cock, R. Dowsley, A. C. A. Nascimento, S. C. Newman, Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data, *AISec '15: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, 2015, 3–14. https://doi.org/10.1145/2808769.2808774
13. A. Agarwal, R. Dowsley, N. D. McKinney, D. Wu, C. T. Lin, M. De Cock, et al., Protecting privacy of users in brain-computer interface applications, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, **27** (2019), 1546–1555. https://doi.org/10.1109/TNSRE.2019.2926965
14. H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, et al., Logistic regression over encrypted data from fully homomorphic encryption, *BMC Med. Genomics*, **11** (2018), 81. https://doi.org/10.1186/s12920-018-0397-z
15. S. Truex, L. Liu, M. E. Gursoy, L. Yu, Privacy-preserving inductive learning with decision trees, *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017, 57–64. https://doi.org/10.1109/BigDataCongress.2017.17
16. M. E. Skarkala, M. Maragoudakis, S. Gritzalis, L. Mitrou, PPDM-TAN: a privacy-preserving multi-party classifier, *Computation*, **9** (2021), 6. https://doi.org/10.3390/computation9010006
17. N. Agrawal, A. S. Shamsabadi, M. J. Kusner, A. Gascón, QUOTIENT: two-party secure neural network training and prediction, *CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, 1231–1247. https://doi.org/10.1145/3319535.3339819
18. S. Wagh, D. Gupta, N. Chandran, SecureNN: 3-party secure computation for neural network training, *Proc. Priv. Enhancing Technol.*, **2019** (2019), 26–49. https://doi.org/10.2478/popets-2019-0035
19. C. Guo, A. Hannun, B. Knott, L. van der Maaten, M. Tygert, R. Zhu, Secure multiparty computations in floating-point arithmetic, *arXiv*, 2020. https://doi.org/10.48550/arXiv.2001.03192
20. M. De Cock, R. Dowsley, A. C. A. Nascimento, D. Railsback, J. Shen, A. Todoki, High performance logistic regression for privacy-preserving genome analysis, *BMC Med. Genomics*, **14** (2021), 23. https://doi.org/10.1186/s12920-020-00869-9
21. Y. Fan, J. Bai, X. Lei, W. Lin, Q. Hu, G. Wu, et al., PPMCK: privacy-preserving multi-party computing for k-means clustering, *J. Parallel Distr. Com.*, **154** (2021), 54–63. https://doi.org/10.1016/j.jpdc.2021.03.009
22. Y. Lindell, B. Pinkas, Privacy preserving data mining, In: M. Bellare, *Advances in cryptology–CRYPTO 2000*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, **1880** (2000), 36–54. https://doi.org/10.1007/3-540-44598-6_3

23. E. Yilmaz, M. Al-Rubaie, J. M. Chang, Naive Bayes classification under local differential privacy, *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, 2020, 709–718. https://doi.org/10.1109/DSAA49011.2020.00081

24. H. Kargupta, S. Datta, Q. Wang, K. Sivakumar, On the privacy preserving properties of random data perturbation techniques, *Third IEEE International Conference on Data Mining*, 2003, 99–106. https://doi.org/10.1109/ICDM.2003.1250908

25. R. Bost, R. A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data, *NDSS*, 2015. https://doi.org/10.14722/ndss.2015.23241

26. A. Wood, V. Shpilrain, K. Najarian, A. Mostashari, D. Kahrobaei, Private-key fully homomorphic encryption for private classification, In: J. Davenport, M. Kauers, G. Labahn, J. Urban, *Mathematical Software–ICMS 2018*, Cham: Springer, **10931** (2018), 475–481. https://doi.org/10.1007/978-3-319-96418-8_56

27. S. C. Rambaud, J. Hernandez-Perez, A naive justification of hyperbolic discounting from mental algebraic operations and functional analysis, *Quant. Financ. Econ.*, **7** (2023), 463–474. https://doi.org/10.3934/QFE.2023023

28. G. A. Tsiatsios, J. Leventides, E. Melas, C. Poulios, A bounded rational agent-based model of consumer choice, *Data Sci. Financ. Econ.*, **3** (2023), 305–323. https://doi.org/10.3934/DSFE.2023018

29. Z. Li, Z. Huang, Y. Su, New media environment, environmental regulation and corporate green technology innovation: evidence from china, *Energy Econ.*, **119** (2023), 106545. https://doi.org/10.1016/j.eneco.2023.106545

30. X. Sun, P. Zhang, J. K. Liu, J. Yu, W. Xie, Private machine learning classification based on fully homomorphic encryption, *IEEE Transactions on Emerging Topics in Computing*, **8** (2018), 352–364. https://doi.org/10.1109/TETC.2018.2794611

31. A. Kjamilji, E. Savaş, A. Levi, Efficient secure building blocks with application to privacy preserving machine learning algorithms, *IEEE Access*, **9** (2021), 8324–8353. https://doi.org/10.1109/ACCESS.2021.3049216

32. A. Khedr, G. Gulak, V. Vaikuntanathan, Shield: scalable homomorphic implementation of encrypted data-classifiers, *IEEE Transactions on Computers*, **65** (2015), 2848–2858. https://doi.org/10.1109/TC.2015.2500576

33. N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, John Wernsing, Cryptonets: applying neural networks to encrypted data with high throughput and accuracy, *ICML'16: Proceedings of the 33rd International Conference on International Conference on Machine Learning*, **48** (2016), 201–210.

34. S. Kim, M. Omori, T. Hayashi, T. Omori, L. Wang, S. Ozawa, Privacy-preserving naive Bayes classification using fully homomorphic encryption, In: L. Cheng, A. Leung, S. Ozawa, *Neural Information Processing, ICONIP 2018*, Cham: Springer, **11304** (2018), 349–358. https://doi.org/10.1007/978-3-030-04212-7_30

35. D. H. Vu, Privacy-preserving Naive Bayes classification in semi-fully distributed data model, *Comput. Secur.*, **115** (2022), 102630. https://doi.org/10.1016/j.cose.2022.102630

36. D. H. Vu, T. S. Vu, T. D. Luong, An efficient and practical approach for privacy-preserving Naive Bayes classification, *J. Inf. Secur. Appl.*, **68** (2022), 103215. https://doi.org/10.1016/j.jisa.2022.103215

37. P. Li, J. Li, Z. Huang, C. Z. Gao, W. B. Chen, K. Chen, Privacy-preserving outsourced classification in cloud computing, *Cluster Comput.*, **21** (2018), 277–286. https://doi.org/10.1007/s10586-017-0849-9

38. C. Gentry, Fully homomorphic encryption using ideal lattices, *STOC '09: Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, 169–178. https://doi.org/10.1145/1536414.1536440

39. X. Yi, Y. Zhang, Privacy-preserving Naive Bayes classification on distributed data via semi-trusted mixers, *Inf. Syst.*, **34** (2009), 371–380. https://doi.org/10.1016/j.is.2008.11.001

40. A. C. Yao, Protocols for secure computations, *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, 160–164. https://doi.org/10.1109/SFCS.1982.38

41. T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, **31** (1985), 469–472. https://doi.org/10.1109/TIT.1985.1057074

42. P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, In: J. Stern, *Advances in cryptology–EUROCRYPT '99*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, **1592** (1999), 223–238. https://doi.org/10.1007/3-540-48910-X_16

43. S. Goldwasser, S. Micali, Probabilistic encryption, *J. Comput. Syst. Sci.*, **28** (1984), 270–299. https://doi.org/10.1016/0022-0000(84)90070-9

44. W. Henecka, S. Kögl, A. R. Sadeghi, T. Schneider, I. Wehrenberg, TASTY: tool for automating secure two-party computations, *CCS '10: Proceedings of the 17th ACM conference on Computer and communications security*, 2010, 451–462. https://doi.org/10.1145/1866307.1866358

45. A. Ben-David, N. Nisan, B. Pinkas, FairplayMP: a system for secure multi-party computation, *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, 2008, 257–266. https://doi.org/10.1145/1455770.1455804

46. X. Liu, R. H. Deng, K. K. R. Choo, Y. Yang, Privacy-preserving outsourced support vector machine design for secure drug discovery, *IEEE Transactions on Cloud Computing*, **8** (2020), 610–622. https://doi.org/10.1109/TCC.2018.2799219

47. X. Yi, Y. Zhang, Privacy-preserving Naive Bayes classification on distributed data via semi-trusted mixers, *Inf. Syst.*, **34** (2009), 371–380. https://doi.org/10.1016/j.is.2008.11.001

48. H. Park, P. Kim, H. Kim, K. W. Park, Y. Lee, Efficient machine learning over encrypted data with non-interactive communication, *Comput. Stand. Inter.*, **58** (2018), 87–108. https://doi.org/10.1016/j.csi.2017.12.004

49. X. Liu, R. H. Deng, K. K. R. Choo, Y. Yang, Privacy-preserving outsourced clinical decision support system in the cloud, *IEEE Transactions on Services Computing*, **14** (2017), 222–234. https://doi.org/10.1109/TSC.2017.2773604

50. R. Podschwadt, D. Takabi, P. Hu, M. H. Rafiei, Z. Cai, A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption, *IEEE Access*, **10** (2022), 117477–117500. https://doi.org/10.1109/ACCESS.2022.3219049

51. D. Beaver, One-time tables for two-party computation, In: W. L. Hsu, M. Y. Kao, *Computing and combinatorics, COCOON 1998*, Berlin, Heidelberg: Springer, **1449** (1998), 361–370. https://doi.org/10.1007/3-540-68535-9_40

52. M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. A. Nascimento, W. S. Poon, et al., Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation, *IEEE Transactions on Dependable and Secure Computing*, **16** (2017), 217–230. https://doi.org/10.1109/TDSC.2017.2679189

53. D. Reich, A. Todoki, R. Dowsley, M. De Cock, A. Nascimento, Privacy-preserving classification of personal text messages with secure multi-party computation, In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett, *Advances in neural information processing systems 32*, 2019, 3752–3764.

54. A. Resende, D. Railsback, R. Dowsley, A. C. A. Nascimento, D. F. Aranha, Fast privacy-preserving text classification based on secure multiparty computation, *IEEE Transactions on Information Forensics and Security*, **17** (2022), 428–442. https://doi.org/10.1109/TIFS.2022.3144007

55. Y. Yasumura, Y. Ishimaki, H. Yamana, Secure Naïve Bayes classification protocol over encrypted data using fully homomorphic encryption, *iiWAS2019: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services*, 2019, 45–54. https://doi.org/10.1145/3366030.3366056

56. R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, 136–145. https://doi.org/10.1109/SFCS.2001.959888

57. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86** (1998), 2278–2324. https://doi.org/10.1109/5.726791