Islam *et al. J Surveill Secur Saf* 2024;5:1-18
**DOI:** 10.20517/jsss.2023.28

**Journal of Surveillance, Security and Safety**

**Original Article**

**Open Access**

Check for updates

# Privacy preserving vertical distributed learning for health data

**Tanzir Ul Islam[1], Noman Mohammed[1], Dima Alhadidi[2]**

[1]Department of Computer Science, University of Manitoba, Winnipeg, MB, R3T 2H4, Canada.
[2]School of Computer Science, University of Windsor, Windsor, ON, N9B 3P4, Canada.

**Correspondence to:** Tanzir Ul Islam, Department of Computer Science, University of Manitoba, Winnipeg, MB, R3T 2H4, Canada. E-mail: islamtu@cs.umanitoba.ca;

**How to cite this article:** Islam TU, Mohammed N, Alhadidi D. Privacy preserving vertical distributed learning for health data. *J Surveill Secur Saf* 2024;5:1-18. http://dx.doi.org/10.20517/jsss.2023.28

**Received:** 8 Aug 2023 **First Decision:** 21 Sep 2023 **Revised:** 26 Oct 2023 **Accepted:** 6 Dec 2023 **Published:** 1 Jan2024

**Academic Editor:** Leandros Maglaras, Songze Li **Copy Editor:** Yanbin Bai **Production Editor:** Yanbin Bai

## Abstract

Federated learning has become a pivotal tool in healthcare, enabling valuable insights to be gleaned from disparate datasets held by cautious data owners concerned about data privacy. This method involves the analysis of data from diverse locations, which is subsequently aggregated and trained on a central server. Data distribution can occur vertically or horizontally in this decentralized setup. In our approach, we employ a unique vertical partition learning process, segmenting data by characteristics or columns for each record across all local sites, known as Vertical Distributed Learning or features distributed machine learning. Our collaborative learning approach utilizes Stochastic Gradient Descent to collectively learn from each local site and compute the final result on a central server. Notably, during the training phase, no raw data or model parameters are exchanged; only local prediction results are shared and aggregated. Yet, sharing local prediction results raises privacy concerns, which we mitigate by introducing noise into the local results using a Differential Privacy algorithm. This paper introduces a robust vertical distributed learning system that emphasizes user privacy for healthcare data. To assess our approach, we conducted experiments using the sensitive healthcare data in the Medical Information Mart for Intensive Care-III dataset and the publicly available Adult dataset. Our experimental results demonstrate that our approach achieves an accuracy level similar to that of a fully centralized model, significantly surpassing training based solely on local features. Consequently, our solution offers an effective federated learning approach for healthcare, preserving data locality and privacy while efficiently harnessing vertically partitioned data.

**Keywords:** Machine learning, security, privacy, vertical federated, electronic health records

## 1. INTRODUCTION

Machine learning (ML) research is gaining increasing prominence in the healthcare sector, offering the potential to save lives by predicting cases and providing valuable healthcare applications. Real-world healthcare data can be leveraged to develop medical diagnostic tools, detect disease risk factors, and assess gene sequence information for therapeutic purposes. For example, Kwekha-Rashid *et al.*[1] showcased how ML can be utilized in healthcare to investigate, predict, and differentiate COVID-19 instances for analysis and triaging. Additionally, Choudhury *et al.*[2] employed health data and a federated ML algorithm to assess the effectiveness of the application in anticipating adverse drug reactions (ADR) in patients. Numerous other studies have also demonstrated the significant contributions of ML in the healthcare field, including disease diagnosis and other vital healthcare tasks. The potential of ML to transform the healthcare industry is vast, opening doors to enhanced case prediction, more effective medical diagnostics, and improved patient outcomes.

The primary challenge in healthcare ML research lies in data distribution, as many clinics are hesitant to share their raw Electronic Health Records (EHR) due to healthcare privacy regulations. As a result, patient data is not centralized but rather dispersed across multiple sites or clinics. This data distribution can be categorized into two types: horizontal and vertical. In the case of horizontally partitioned data silos, each client possesses a unique collection of records, but their data shares common features. Typically, horizontal partitioning involves distributing rows of a table across several database clusters. However, in certain contexts, such as healthcare, there is a need to process data across multiple sites for the same set of records but with different sets of attributes, which does not fit the traditional horizontally partitioned arrangement. Vertical partitioning, on the other hand, involves distributing data in a way where the same set of records can have distinctly different attributes on each site. For example, one clinic may possess patient data with attributes such as name, age, and disease code, while another clinic might have data with attributes such as name, age, and timestamps of ICU stays for the same group of patients. While most current ML research focuses on training with horizontally partitioned data, vertically partitioned data has received less attention in this context. This article centers on ML training using vertically partitioned data distribution within the healthcare industry. Healthcare organizations may utilize this type of data segregation when they aim to assess a patient's health using clinical data from various locations.

The centralized ML architecture is not a feasible solution for handling distributed data silos. To address this challenge, Federated Learning (FL)[3,4] emerges as a distributed ML approach. FL enables multiple data owners to collaboratively develop and utilize a shared prediction model while preserving the privacy of their local training data. This method iteratively enhances model accuracy by allowing each site to update its local model and exchange locally computed gradients or model parameters with a central server. This entire process aims to minimize the need to send raw data outside of individual facilities. Instead of sending raw data to a centralized server, ML is trained using distributed data stored across multiple sites. Local data providers retain their raw data, sharing it solely with locally constructed ML models. However, the sharing of model data in distributed learning raises several privacy concerns. There is a risk that an attacker may reconstruct original data or extract sensitive personal information from the shared model[5,6]. Therefore, the FL method requires an additional layer of privacy protection to further safeguard the data and mitigate potential privacy risks.

In a horizontal setting, recent approaches typically employ data anonymization or differential privacy (DP) algorithms to introduce a privacy layer. Choudhury *et al.*[7,8] demonstrated how customized *k-anonymity* or DP can be utilized in FL to maintain privacy while ensuring acceptable utility. They applied this approach to healthcare data to predict mortality rates based on patients' ICU stays. Similarly, much of the current research in this field focuses on integrating suitable privacy algorithms into FL to comply with healthcare regulatory policies. In our earlier research[9], we proposed a framework that communicates model data only after sufficient data sanitization to protect patients' privacy while preserving data utility. For vertically partitioned data, the FL setting differs from the well-researched horizontal FL configuration as it presents its own unique

characteristics and challenges. Local models in vertical FL (VFL) require data from other sites to jointly train the ML model and learn about all features. Existing methods[10,11] mostly rely on encrypted multiparty communication to exchange information about each other's features. However, these cryptographic techniques reduce privacy concerns at the cost of increased computational overhead. In contrast, our approach reduces the necessity to learn about other sites' features for training. It only exchanges local predictions with the server for score aggregation, eliminating the need to share model or feature sets and thereby enhancing privacy.

In our vertical architecture, models are independently trained locally based on the available features at each site. The central server's role is solely to aggregate the local predictions; it does not partake in the training process. Training the full model is achieved through mini-batched Stochastic Gradient Descent (SGD) using distributed computation. The local training progresses asynchronously, allowing different parties to execute various iterations of parameter updates, which aligns with our design approach. Local feature sets are used for classification through Logistic Regression (LR) and long short-term memory (LSTM) neural networks (NNs), producing local predictions. These local predictions are then shared with the central server, where they are aggregated to obtain the final prediction.

Table 1 and Table 2 illustrate how feature sets can be distributed for the same patients. The first table, $P_1(ID, InTime, OutTime)$, lists a group of patients' ICU stays, while the second table, $P_2(ID, Age, DiseaseCode)$, contains their disease codes. At each site, local predictions for mortality rates or the probability of a specific acute disease are generated and shared with the central server. The server aggregates these local predictions without exchanging any model parameters, as shown in Table 3. To address privacy concerns and prevent potential privacy attacks, we apply the well-established privacy method, DP[12], to perturb the local predictions before sharing them. By using this approach, the server only receives the perturbed predictions to determine the final prediction for each classification task. This ensures a robust vertical distributed learning (VDL) framework with data privacy, although there may be some loss in accuracy due to the addition of DP noise.

The primary challenge in VDL is to maintain data privacy while achieving an efficient level of accuracy without incurring significant communication overhead. Our approach addresses this challenge by ensuring data privacy with minimal communication overhead, as we only share prediction results with the central server. Once each local prediction is sent to the central server, we apply a weighted feature approach to build the final accuracy. In a similar manner to Hu *et al.*[13], who proposed a feature distributed collaborative learning technique using a continuously differentiable function to aggregate local intermediate predictions as a weight parameter, we shuffle and weight local predictions randomly. This approach effectively influences the ultimate score. For instance, if a local prediction has a feature set $x_1$ and $x_2$ that could contribute to higher accuracy when trained in a central architecture with all the feature sets, assigning more weight to these features will result in improved performance. A sample weighted matrix is provided in Table 4.

**Contributions:** The contributions of our work can be summarized as follows:

- We proposed a VFL method for the healthcare sector to improve performance and ensure robust data privacy. Our investigation spanned four specific applications in the healthcare domain: In-hospital Mortality, Forecasting Length of Patient Stay, Phenotype Classification, and Decompensation Detection.
- We introduced a randomized feature distribution technique, which yielded superior results, showcasing enhanced granularity in feature selection. The combination of LSTM-based modeling and a refined feature selection process is a key contribution to our work, aiming to optimize performance and relevance in healthcare applications.
- We conducted experiments using the *Adult* and *Medical Information Mart for Intensive Care (MIMIC)-III* datasets, employing LR and LSTM-based deep NNs. We compared our results with a baseline centralized model. Our approach achieves accuracy levels almost on par with the baseline model while maintaining

**Table 1. Raw data - site 1 (ICU stays)**

| Patient ID | In time | Out time |
|---|---|---|
| 831 | 1955-02-25 | 1955-02-27 |
| 832 | 1957-12-02 | 1957-12-31 |
| 833 | 1957-12-02 | 1957-12-31 |
| 834 | 1973-11-30 | 1973-12-31 |
| 835 | 1973-12-31 | 1974-01-14 |
| 836 | 1929-03-21 | 1929-04-06 |
| 837 | 1929-03-21 | 1929-05-06 |
| 838 | 1929-03-21 | 1929-06-06 |

**Table 2. Raw data - site 2**

| Patient ID | Age | Disease code |
|---|---|---|
| 831 | 34 | 710 |
| 832 | 70 | 480 |
| 833 | 33 | 34 |
| 834 | 55 | 48 |
| 835 | 47 | 683 |
| 836 | 67 | 809 |
| 837 | 77 | 807 |
| 898 | 58 | 586 |

**Table 3. Data available - central**

| Classification | Site 1 (prediction) | Site 2 (prediction) | Final prediction |
|---|---|---|---|
| Logistic regression | 0.67 | 0.75 | 0.84 |
| LSTM | 0.56 | 0.70 | 0.86 |

**Table 4. Weighted feature matrix**

| Prediction task | In time | Out time | Age | Dis.Code |
|---|---|---|---|---|
| Mortality rate | 0.85 | 0.85 | 0.25 | 0.40 |
| Acute disease | 0.15 | 0.15 | 0.86 | 0.85 |

privacy. Specifically, for the *MIMIC-III* dataset, our solution achieves up to 80% and 82% accuracy for LR and LSTM methods, respectively. For the *Adult* dataset, we achieve 90.3% and 90.4% accuracy for LR and LSTM methods, respectively.

- In summary, our model experiences a maximum accuracy loss of 5% due to the addition of noise in the distributed learning framework. Despite this minimal loss, we successfully preserve privacy in the process.

## 2. BACKGROUND

In this part, we will provide a quick explanation of some of the important principles that were used to develop this suggested framework.

### 2.1. Vertical federated learning

Vertical federated learning (VFL) is a ML technique that empowers organizations with vertically partitioned data to develop and train a decentralized ML model while safeguarding data privacy and security. In this approach, data samples are distributed among different parties, with each party possessing its unique set of features or attributes. The main objective of VFL is to create a global ML model using these vertically partitioned data samples while ensuring that the raw data remains confidential and secure. To achieve this, each party independently trains its local model based on its own data and later combines the results at a central server to construct the global model. This method proves effective in scenarios where organizations have data silos and desire to harness ML to improve models without sharing raw data with one another. In VFL, data is partitioned in a manner where two nodes may share the same user profiles but contain different feature information. These nodes could represent various health institutions or providers of healthcare data applica-

tions. The aim of VFL in such cases is to create a comprehensive model by aggregating patient features from multiple institutions without directly exchanging patient data. This way, VFL ensures collaboration and model improvement while maintaining strict data privacy and security. Each node sharing the same sample of data $I$ and contributing its own unique set of patient features $X$ and labels $Y$ information, VFL can be denoted as the following:

$$I_i = I_j = \begin{cases} X_i \neq X_j & i \neq j \\ Y_i = Y_j \end{cases} \tag{1}$$

In this scenario, we are considering a homogenous label setting where all parties are trying to predict the same target variable, but each party may have access to different features that are relevant to the prediction. For example, from each client, the same data sample $I$ can be denoted with different features $\{x_1^i, x_2^i, x_3^i\}$ and $\{x_3^j, x_4^j, x_5^j\}$ but with the same label $y$. Consider a scenario where a hospital and a nearby immunization center are two distinct healthcare institutions operating in the same area. As they serve local populations, the patients utilizing these facilities may exhibit significant similarities. However, the data they retain could differ, with vaccine centers storing users' immunization histories and hospitals maintaining medical treatment histories. Due to this disparity in data storage, the user features may not be directly related. In this context, VFL provides a safe and effective approach to integrating diverse feature sets to enhance model performance while still preserving the locality of the data. By utilizing VFL, valuable insights can be gleaned from both types of healthcare institutions without compromising data privacy, enabling improved healthcare decision-making for the benefit of the community.

### 2.2. LSTM neural networks

A NN is a network of interconnected processing "nodes" or units functioning analogously to biological neurons. These nodes, synthetic versions of biological neurons, receive inputs that are multiplied by weights and then delivered to the cell body, where they are combined through basic arithmetic to produce node activations. A threshold logic unit (TLU) carries out this calculation, resulting in an output of either zero or one. Constructing a NN involves defining its model structure, including the number of input features and outputs, and initializing its parameters before running them in a loop[14,15]. During this process, forward propagation calculates the current loss, backward propagation computes the current gradient, and gradient descent updates the parameters. Preparing the dataset and tuning the learning rate can significantly influence the algorithm's performance.

Recurrent NNs (RNNs) are a specialized type of NN that allows information to be propagated from one step of the network to the next. They are particularly suited for tasks involving sequential data, such as language translation and speech recognition. RNNs process input sequences element by element while maintaining an internal state that encodes the context of the sequence up to that point. This enables the network to utilize information from earlier elements in the sequence when processing later ones, facilitating the capture of dependencies between elements in the input sequence. RNNs come in various forms, such as LSTM networks and gated recurrent units (GRUs).

LSTM is a specific type of RNN well-suited for modeling temporal data with long-range dependencies. RNNs process sequential data by iterating through the time steps of the input and maintaining a hidden state that carries information about the past. LSTMs, as a variant of RNNs, incorporate an additional "memory cell" capable of storing information for an extended duration. Additionally, they feature three "gate" mechanisms (input, output, and forget gates) that control access to and modification of the memory cell, enabling effective long-term information retention. The equations for the forward pass of an LSTM cell can be denoted below:

Forget gate's compact forms:

$$\hat{c}^{<t>} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{2}$$

Update gate's activation vector:

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{3}$$

Forget gate's activation vector:

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{4}$$

Output gate's activation vector:

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{5}$$

Finally, cell state vector $c^t$:

$$c^t = \Gamma_u * \hat{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{6}$$

These are the standard equations employed to update the hidden state and memory cell in an LSTM at each time step. $x^{<t>}$ represents the input at time step $t$, $a^{<t>}$ denotes the hidden state at time step $t$, and $a^{<t-1>}$ represents the hidden state at the previous time step. The weight matrices, $W_u$, $W_f$, and $W_o$, correspond to the update, forget, and output gates, respectively, while $b$ represents the bias term. The function $\sigma(x)$ denotes the sigmoid function, which maps a value to the range [0, 1]. Additionally, the function $tanh(x)$ denotes the hyperbolic tangent function, which maps a value to the range [-1, 1]. These mathematical operations play a crucial role in the LSTM's ability to update and control its hidden state and memory cell, allowing for effective memory retention and handling of sequential data. A detailed explanation of the LSTM algorithm can be found in [16–18].

### 2.3. Differential privacy

DP is a mathematical framework for protecting the privacy of individuals in a dataset. Its goal is to enable the release of useful statistical information about a dataset while ensuring that no individual's data can be inferred from the released information. The privacy level is controlled by the *epsilon* variable, which can be changed in accordance with varied circumstances and security guidelines. The system provides increased security as *epsilon* changes.

**Definition 1 ($\varepsilon$-Differential Privacy)** *An algorithm X is differentially private[12] if, for any datasets D and D′ differing by at most a single record, and for all sets S ∈ R, where R is the range of X, the following condition holds*

$$Pr[X(D) \in S] \leq e^{\varepsilon} \times Pr[X(D') \in S] \tag{7}$$

Two datasets, in this instance, are said to be neighbors if their sole difference is one record. Here, the privacy budget of the method can be represented by the non-negative parameter $\varepsilon$. The privacy budget determines the trade-off between the accuracy of the query and the privacy of the individuals in the dataset. A smaller privacy budget corresponds to a larger amount of noise added and, therefore, a lower accuracy but stronger privacy guarantees.

One common algorithm for achieving DP is the Laplace mechanism. This mechanism adds noise to the output of a statistical query in a way that preserves DP. The amount of noise added is determined by the "sensitivity" of the query, which measures how much the output of the query can change when data from a single individual is removed from the dataset.

## 3. RELATED WORK

Over the last few years, significant attention in distributed ML research has been directed towards data privacy and security within the healthcare industry. In this section, we provide a brief overview of several related works in this area.

### 3.1. Distributed learning and privacy attacks

FL is a distributed learning method that avoids sharing sensitive raw data to preserve data privacy. Recent studies have demonstrated the advantages of using FL over traditional centralized ML models, especially for sensitive data applications. However, in this distributed framework, there are potential privacy attacks, such as inference[6], reconstruction[19,20], or backdoor[21,22] attacks, which could lead to the retrieval of data through exposed information. To address these concerns, Rajkumar *et al.*[23] proposed the use of DP algorithms to effectively minimize privacy attacks while maintaining reasonable utility. Choudhury *et al.*[7] applied this DP approach in the FL architecture for healthcare data. However, they also found that the DP noise introduced could adversely affect data utility. In their subsequent work[8], Choudhury *et al.* aimed to strike a balance between privacy and utility by implementing tailored *k-anonymity* to reduce DP noise as much as possible. An empirical assessment of the health records of one million patients demonstrated robust model performance that outperformed standard DP approaches. Similarly, in our recent work[9], we were motivated by the idea of "data sensitization first, noise applied last" to minimize DP noise in model data and enhance the overall data utility of the framework. Our approach seeks to achieve a more optimal balance between privacy protection and maintaining meaningful utility in the context of distributed healthcare data analysis.

Privacy-preserving ML can be achieved through encryption[24–26] or secret sharing[27–29] via secure multiparty communication. In these frameworks, models are communicated using encrypted data or shared secrets. However, encryption requires more computational resources and may not be suitable for all scenarios. Additionally, secure multiparty communication necessitates frequent data transfers between clients, leading to communication overhead.

In recent years, distributed ML techniques focusing on horizontal data have received increasing attention. In contrast, our work enables feature-parallel ML among nodes with vertically partitioned data, which is equally significant but has not yet been extensively investigated. This approach addresses the challenges unique to vertically distributed data, opening up new avenues for privacy-preserving ML in a different context.

### 3.2. Privacy-preserving distributed learning on vertical data

Numerous research efforts have been dedicated to developing privacy-preserving techniques for VDL. The primary goal of these techniques is to safeguard the privacy of training data while ensuring effective model training. One common approach involves the application of DP[12,30], where noise is added to the data in a manner that preserves individual privacy while enabling the model to capture meaningful patterns. Other methods include leveraging secure multiparty computation (SMC)[31] to facilitate joint computation of the model by multiple parties without revealing their respective data to each other. Additionally, homomorphic encryption is utilized to encrypt the data, allowing the model to be trained on the encrypted data without exposing the underlying data itself. These privacy-preserving techniques contribute to a more secure and efficient VDL process.

Liu *et al.*[10] introduced the Federated Stochastic Block Coordinate Descent (FedBCD) algorithm, which utilizes an SMC approach. In their work, each party performs multiple local updates to minimize communication overhead. The focus is on vertically partitioned data, where only a single value is shared instead of the entire model or raw data, ensuring data privacy is maintained. Similarly, Hu *et al.*[32] explored a similar approach for VFL using the Alternating Direction Method of Multipliers (ADMM), commonly employed in distributed ML. In their method, they shared a single value for model training and incorporated the $\varepsilon, \delta$- DP algorithm

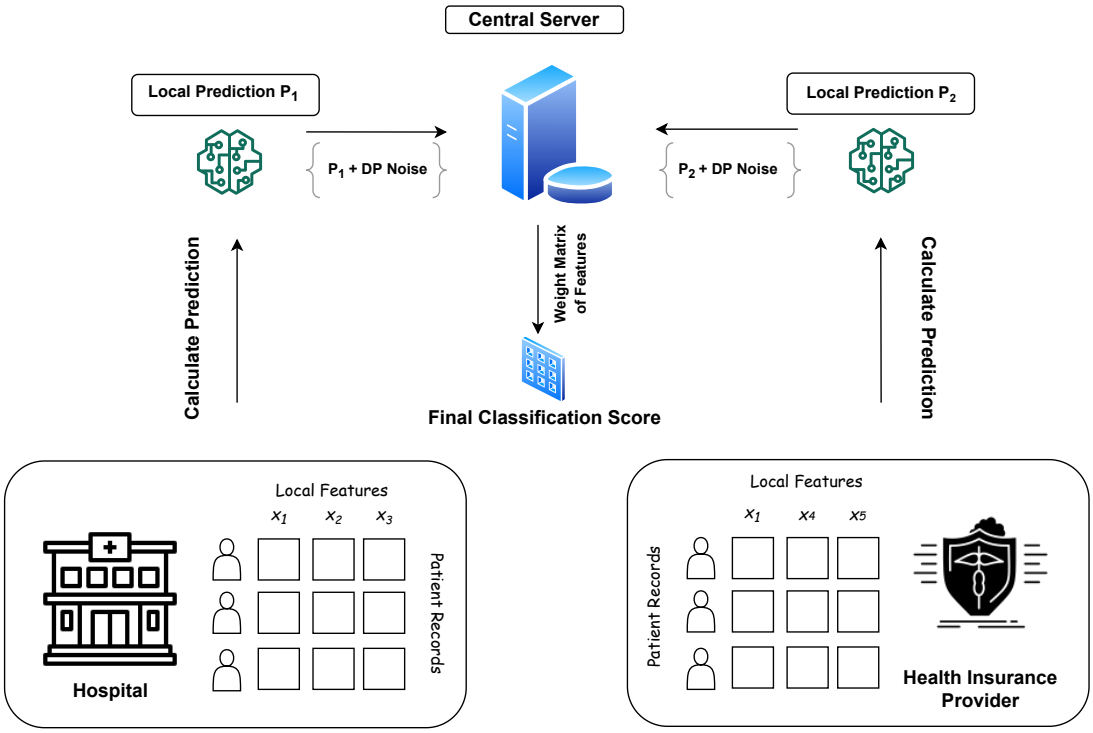



**Figure 1.** Multiple Data Owners are training a model collaboratively using distributed feature learning.

to perturb the shared value using Gaussian noise. The perturbed approach ensures that the probability distribution of communicated values remains largely unaffected by modifications to any individual feature in a party's local dataset. With theoretical privacy guarantees, their method converges with significantly fewer epochs compared to the state-of-the-art SGD approach. Chen *et al.*[11] also proposed a VFL approach in an asynchronous manner, utilizing SMC for communication between parties. They incorporated "perturbed local embedding" with Gaussian DP noise to enhance data privacy. Their experiments covered both LR and deep learning for healthcare data, demonstrating the efficacy of their approach in maintaining privacy while training ML models on distributed features.

In a VFL scenario, effective communication of each partner's unique feature information is crucial for joint learning and training of any classification algorithm. Currently, many technologies rely on SMC techniques to exchange feature information with clients. Some strategies employ homomorphic encryption or DP for data perturbation or sanitization to preserve privacy while sharing feature information. Hu *et al.*[13] proposed a feature-distributed collaborative learning strategy where each client undergoes independent training, and the final prediction output is shared with a central server for computing the final score. This approach maintains data locality in an asynchronous SGD method by making predictions solely based on local features in a parallel computing fashion. They also employed Laplacian DP noise to safeguard the shared prediction sent to the main server. However, their technique did not specifically target healthcare data and utilized the NN deep learning technique, with LSTM-based approaches showing better performance. Our approach is quite similar to theirs, except that we adopted the latest LSTM methods specifically designed for healthcare data. By leveraging these advanced techniques, our approach demonstrates improved performance and effectiveness in the context of healthcare data analysis within the VFL framework.

## 4. METHODS

In this section, we introduce our proposed approach for training a model using vertically partitioned data while simultaneously safeguarding the data privacy.

### 4.1. Problem overview

In the proposed architecture of VFL, depicted in Figure 1, models are trained locally based on the available features at each site. The central server acts as an aggregator for the overall predictions but does not engage in any training. The complete model is trained using mini-batched SGD [2,30,33] at each local party. Once local training is completed, the final results are shared with the central server. To classify the local feature set and produce a local prediction, LR and LSTM NNs are employed. The central server aggregates these local predictions to generate the final prediction. To determine the appropriate feature weight for each feature, random value assignment is used. This weight matrix, together with the local prediction results, is aggregated by the central server. This design eliminates the need for communication between parties, thereby reducing the risk of data exposure. In addition to the weight-based aggregation, the local predictions are perturbed with DP noise, creating a more robust privacy-preserving model, albeit resulting in a slight decrease in accuracy in the final outcome.

The model performance will be evaluated using an appropriate evaluation metric, aiming to minimize the error between the predicted outputs and the true labels. The objective is to build a model that generalizes well to unseen data and can be effectively deployed in a real-world healthcare setting to address the problem at hand.

### 4.2. Local prediction calculation

In our model, each data source or site contains a unique feature set. As a result, we train our model based solely on the available local features in this step. In a similar setting, Hu *et al.* [13] utilized LR and convolutional NN (CNN) approaches in mobile app datasets. In contrast, we leverage both LR and LSTM models for local feature-based training specifically tailored for healthcare data. LR is a straightforward yet effective linear model commonly employed for classification tasks. It operates by using a linear combination of input features to predict the probability of a given example belonging to a particular class. On the other hand, LSTM is a type of RNN well-suited for modeling sequential data, such as time series or natural language. LSTMs excel at capturing long-term dependencies within data by employing gating mechanisms to control information flow throughout the network.

For local feature-based training using LR, we first define a set of local features, compute the local feature representation for each feature vector, and subsequently use these local features as input to train an LR model.

Suppose we have a set of training data, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where each $x_i$ is a feature vector and $y_i$ is the corresponding label. To train a model based on local features, we first need to define a set of local features $f_1, f_2, \ldots, f_m$ that we will use to represent each feature vector $x_i$.

Then, for each feature vector $x_i$, we can compute its local feature representation $z_i$ as:

$$z_i = [f_1(x_i), f_2(x_i), \ldots, f_m(x_i)]$$

Once we have computed the local feature representation for each feature vector in the training data, we can train the LR model using these local features as input and the corresponding labels $y_i$ as output.

Before selecting local features, it is important to perform feature engineering and data preprocessing as necessary. Let $x_i$ represent the original feature vector, which consists of a set of local features $f_1, f_2, \ldots, f_m$ that

are selected based on domain knowledge and relevance to the problem. In healthcare data analysis, these local features can represent different aspects of patient data or medical measurements. Once the local features are chosen, the local feature representation $z_i$ for each feature vector $x_i$ is computed by applying each local feature function to the original feature vector $x_i$. Here, $f_j(x_i)$ represents the result of applying the $j$-th local feature function to the $i$-th feature vector to train a ML model using these local feature representations $z_i$. The choice of ML algorithms depends on the problem and data characteristics. In our case, we choose both LR and LSTM. To evaluate the model performance using appropriate metrics, we compute accuracy, precision, recall, F1-score, and other relevant metrics based on the model's predictions and the true labels. The relationship between the original feature vector $x_i$ and the local feature functions is one of transformation and extraction. The local feature functions transform the raw, high-dimensional data in $x_i$ into a set of lower-dimensional features represented by $f_1(x_i), f_2(x_i), \ldots, f_m(x_i)$. These local features are chosen based on domain knowledge and relevance to the problem, and they effectively "highlight" certain characteristics of the data while potentially discarding less relevant information. In summary, the local feature functions are used to derive informative features from the original data $x_i$, and these features are employed for training ML models in healthcare data analysis. The choice of local features and their calculation methods significantly mold the model's ability to capture relevant patterns in the data.

On the contrary, when employing LSTM for local feature-based training, we adopt a similar process. However, we need to reformat the local feature representation into a sequence suitable for LSTM processing. For the given set of local features $f_1, f_2, \ldots, f_m$ and each feature vector $x_i$, we create a sequence by concatenating these local features. Subsequently, we use this sequence as input to an LSTM model, following the equations described in the *Background* section.

To replicate our methodology within the healthcare industry, we train the model using the sensitive *MIMIC-III* dataset. Prior to commencing training, we partition the entire dataset into two distinct sets, each containing the same records but with different attributes, as illustrated in Figure 1. During a single training session, we adopt a homogenous label setting, wherein all parties aim to predict the same target variable. However, each party may have access to different features that are relevant to the prediction. To mimic the VFL scenario, we segregate the full dataset into two separate sets with identical records but varying features. The model training is then conducted independently on two different clients to compute the local prediction results for both LR and LSTM. In this simulated scenario, we can consider one party as a Hospital and the other party as a local health insurance provider. After suitable data preprocessing, for the same set of records, the two parties may possess different features. For example, as depicted in Figure 1, the Hospital may have features $x_1, x_2, x_3$, while the health insurance provider may have features $x_1, x_4, x_5$. Hence, $x_1$ is the common feature shared by both parties, while the other two features differ. For benchmarking, we use the tasks defined by Harutyunyan *et al.*[34] as below:

#### *4.2.1. In-hospital mortality (IHM)*

This task involves binary classification to predict in-hospital mortality based on data from the first 48 hours of an ICU stay. Our LSTM model takes a sequence of vital signs and other patient data collected over time as input and predicts the probability of death during the hospital stay.

#### *4.2.2. Forecasting length of stay (LOS)*

The benchmark addresses a multiclass classification problem that predicts the remaining ICU stay duration, which is divided into ten classes/buckets: $< 1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-13, > 14$. This classification is performed only for individuals who did not die in the intensive care unit.

#### *4.2.3. Phenotype classification (PH)*

Phenotyping involves 25 discrete binary classification tasks aimed at determining the presence of 25 acute care conditions in a specific patient's ICU stay record.

#### *4.2.4. Decompensation (DC)*

The objective is to identify patients who are physically deteriorating. Decompensation is a sequential prediction task where the model forecasts after each hour in the ICU. The goal at each hour is to anticipate the patient's mortality within a 24-hour time frame.

To prepare the model with the local feature set and time series observations, we capture the patient's length of ICU stay over $T$ hours, with $x_t$ at each time step $t$ (one-hour interval). To model the time series observations, we define the feature as $[x_t]_{t=1}^{T}$, taking into account the previous hidden state $a^{<t-1>}$ and the current hidden state $a^{<t>}$ in the LSTM model. According to the LSTM equation mentioned in the *Background* section, the model equation for all the above four tasks will be as follows:

$$\hat{IHM} = \sigma(W_o[a^{<48>}, x^{<48>}] + b_o) \tag{8}$$

$$\hat{LOS} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{9}$$

$$\hat{DC} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{10}$$

$$\hat{PH} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{11}$$

For the in-hospital mortality task, the binary label $o$ is determined at $t = 48$ hours, whereas for the other three tasks (decompensation, phenotyping, and LOS), it is predicted at each time step $t = 5...T$. Notably, in-hospital death prediction is made at the end of 48 hours, while decompensation, phenotyping, and LOS tasks are anticipated at each time step after the first four hours of ICU stay, following the benchmark set by Harutyunyan *et al.*[34]

The LSTM model is trained using the Adam optimizer with a learning rate of $D$ and a batch size of $E$. A categorical cross-entropy loss function is employed, and the model performance is evaluated on the validation set at the end of each epoch. The training is conducted for a maximum of 20 epochs, and the best model is selected based on the validation loss.

Similarly, alongside the *MIMIC-III* dataset, we also perform training on the *Adult* dataset to benchmark and compare the model performance. In the *Adult* dataset, the dependent variable for the ML classification training is the *Income* attribute, which can take values either > 50k or <= 50k. We divide the full dataset into two separate sets with the same records but different features to simulate the VFL scenario. Subsequently, we apply the same model training as described above for the *MIMIC-III* dataset. Additionally, we compare the performance of our LSTM model to a baseline LR model and observe that the LSTM model outperforms the LR model for both datasets, as detailed in the *Experiment* section.

### 4.3. Feature weight mechanism

In ML, feature weights are values learned during the model training process to predict the target variable. These weights indicate the relative importance of each input feature in making predictions. For example, in a model predicting the mortality rate of ICU patients based on features such as length of stay (LOS), drug history, and age, the feature weights reveal which features have the most significant impact on the prediction. For instance, the LOS might be assigned the highest weight, followed by age and drug history. In VFL, a weighted feature algorithm is employed to determine the importance of each feature in the final prediction. Each party trains an LR model locally using its own set of features and labels. This process involves iterative optimization of the weights to minimize the difference between predicted probabilities and actual target values. The learned coefficients of the LR model represent the feature weights. Positive weights indicate a positive relationship with the target variable, while negative weights indicate a negative relationship. The magnitude of the weight reflects the strength of the impact. To protect privacy, only the locally calculated coefficients are shared with the central server, along with a local prediction. As the central server does not perform any training, it solely aggregates the coefficients received from each party and uses them for weighted multiplication to obtain the final score. This approach avoids the need to share raw feature data among parties, preserving privacy. The coefficients obtained from the locally trained LR models are used as the feature weights. By leveraging these feature weights, the central server can aggregate the local predictions effectively without compromising data privacy. This process ensures that the importance of each feature is considered in the final prediction without sharing the raw feature data itself.

Finally, the weight matrix is constructed, as exemplified in Table 4. The weight values in this matrix represent the relative importance of each feature in the classification tasks. For instance, when predicting the mortality of hospital patients, the duration of their stay (*In Time* and *Out Time*) has a significantly greater effect on the classification compared to *Age* or *Disease Code*. Therefore, *In Time* and *Out Time* are assigned higher weights of 85%, while *Age* and *Disease Code* are given weights of 25% and 40%, respectively. In contrast, when performing phenotyping for acute diseases, the duration of hospital stay (*In Time* and *Out Time*) has a minimal impact, whereas the *Disease Code* plays a more critical role. Hence, in this scenario, *In Time* and *Out Time* are assigned lower weights of 15%, while *Age* and *Disease Code* receive higher weights of more than 85%. These weight values reflect the varying degrees of influence each feature has on the different classification tasks.

### 4.4. Aggregator server execution

The local predictions using both LR and LSTM models are computed at each individual client and subsequently sent to the aggregator server for the final aggregation. Once all clients have shared their results, the aggregator server utilizes the weight matrix to calculate the ultimate output. The aggregated score is then optimized by minimizing the loss function through the application of the differential function, as illustrated in Table 3.

For a number of parties $P = \{1, 2, ...p\}$, we can simplify the calculation of the final score as below:

$$\mathcal{Y} = \delta\left(\sum_{p=1}^{P} w^p y(x^p)\right) \tag{12}$$

In this framework, we only consider the scenarios of two parties where only the local predictions $y(x^p)$ and the corresponding weight matrix $w^p$ are shared to calculate the final prediction. The aggregation of local predictions is carried out using a continuously differentiable function $\delta : \mathbb{R} \rightarrow \mathbb{R}$, which takes into account the provided weight matrix. In future work, we plan to extend the parties to $\geq 2$ to showcase the experiment.

It is important to note that the server's role is solely to perform the aggregation based on the local predictions and weight matrix without engaging in any ML classifications. As a result, this framework resembles a parallel

or distributed learning algorithm.

For instance, in the case of LR training, the local scores from site 1 and site 2 are 0.67 and 0.75, respectively, as shown in Table 3. Subsequently, after obtaining the average weight coefficients from Table 4 as 0.85 and 0.35, the aggregation is performed, resulting in a calculated score of 0.84 $(0.67 * 0.85 + 0.75 * 0.35)$. This value lies between the scores of each local party due to the inclusion of noise addition and weight multiplication during the calculation process.

### 4.5. Privacy mechanism

A central focus in our design is the preservation of local data privacy. To achieve this objective, we implement a privacy-preserving strategy that involves solely transferring local prediction results derived from the local data and features. The decision to refrain from exchanging raw data is driven by the potential risk of exposing sensitive information to unauthorized entities or servers. To mitigate this privacy concern, we employ the widely accepted DP algorithm [12,30,35], which establishes a standard for safeguarding the privacy of algorithms working with aggregated data. Ensuring the anonymity of feature characteristics is pivotal, and to achieve this, each party introduces Laplacian noise to their local prediction result, denoted as $y(x^p)$ at party $p$. This noise addition guarantees a heightened level of privacy protection against potential malicious servers or parties. Consequently, the shared prediction, with added DP noise, is represented as $y(x^p) + \varepsilon$, where $\varepsilon$ is a variable representing the Laplacian noise introduced to control the level of privacy. The calibration of noise is based on the function's sensitivity, detailed in [12]. The noise exhibits an inverse relationship with performance but a direct correlation with data privacy. Introducing more noise increases data obfuscation but enhances privacy. However, this comes at the cost of lower originality in the data, resulting in reduced ML scores. Therefore, our framework seeks to strike a balance, maintaining a middle ground to simultaneously preserve data privacy and optimize performance. We would like to emphasize that DP stands as a well-established method for ensuring privacy, backed by provable privacy boundaries [36]. Extensive theoretical and experimental explorations have already been conducted in this area (e.g., [37–40]). Consequently, much research has focused on examining how the adoption of DP affects model accuracy (e.g., [41–44]) without delving into its defense against specific privacy attacks. Our work aligns with this methodology.

## 5. RESULTS

In this section, we present the experimental results obtained from testing our approach with both the *MIMIC-III* and *Adult* datasets under various settings. To benchmark and compare the outcomes, we set a fixed privacy range for the DP algorithm, ranging from $\varepsilon = 1$ to $\varepsilon = 5$, for each experiment instance. For the study, we simulated a multi-client environment by utilizing multiple computers in our lab.

### 5.1. Experimental setup

In our experiments, we used the MIMIC-III dataset and followed the benchmark configuration proposed by Harutyunyan *et al.* [34] for processing time series signals from ICU devices. The test set used in the benchmark was also utilized, and we allocated 15% of the remaining data for the validation set. We conducted all four benchmarking tasks described in the methodology section. For the in-hospital mortality task, we considered only patients who were hospitalized in the ICU for at least 48 hours. We removed clinical notes without a linked chart time and any patients without any notes. In the LSTM model, we used 64 hidden units for decompensation and LOS prediction. The phenotyping classification task involved 25 distinct binary classification tasks based on ICU phenotypic data.

Additionally, we tested our approach on another public dataset called Adult, a traditional census dataset with 48,842 samples, each having 124 characteristics.

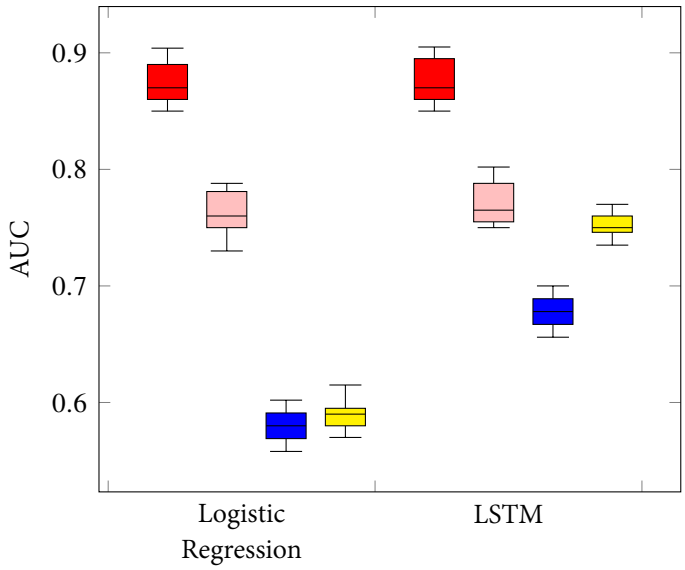



**Figure 2.** AUC for Adult Dataset with Privacy Budget $\varepsilon$ : 1 to 5 [RED: Central], [PINK: VDL], [BLUE: Local1], [YELLOW: Local2]

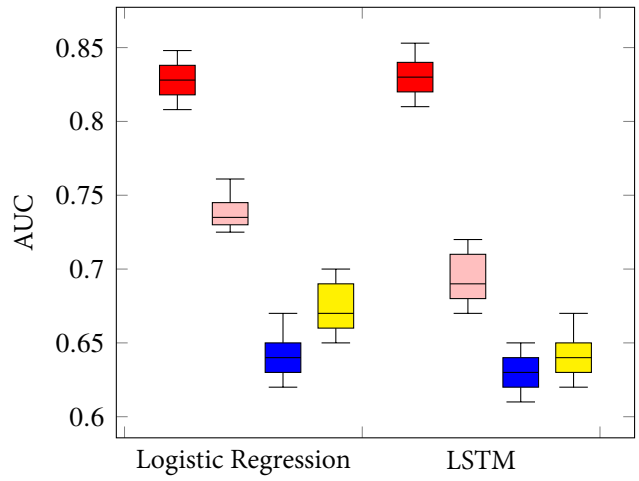
The experiments were conducted using both LR and LSTM models in four different scenarios for both datasets. Our test setup consisted of a central server and two clients. In the benchmarking phase, we first examined the prediction in a central architecture where all the datasets with all attributes were available in a single location. Next, we calculated the prediction independently for each local client. Finally, to simulate the VDL architecture, we performed the experiments with two clients and a central server. To create the VDL scenario, we used the Linux *Cut* command to split the data columns-wise for the identical set of records, effectively separating the entire dataset by feature.

### 5.2. Evaluation

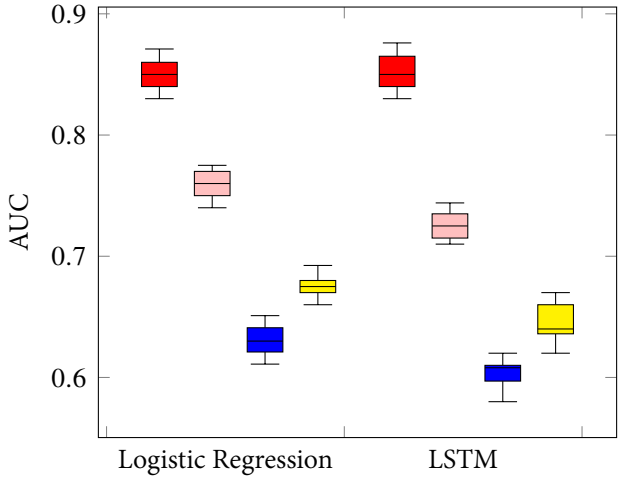
We used the Area Under Precision-Recall (AUC) measure to evaluate the performance of the LR and LSTM models for the in-hospital mortality, phenotyping, and decompensation tasks in the MIMIC dataset. For the LOS task, we used Cohen's linear weighted kappa, as proposed by Harutyunyan *et al.*[34], which assesses the correlation between predicted and actual multi-class buckets. We presented the results in box plots, with red, pink, blue, and yellow boxes representing non-private central, VDL, and local predictions for parties 1 and 2, respectively.

Figure 3(a), Figure 3(b), and Figure 3(c) show the AUC values for the three tasks in the MIMIC dataset. The central architecture consistently performs best since it serves as the benchmark reference with access to the full dataset and all features without privacy constraints. However, the local predictions for parties 1 and 2 have relatively lower scores compared to the central architecture, which is expected as they are trained only on the available local features. On the other hand, the VDL approach outperforms the local parties and comes close to the central architecture's performance. The slight drop in scores for VDL is due to the addition of noise and the weighted feature application during the final score computation on the central server. Figure 3(d) shows the kappa values for the *LOS* forecasting task. Similar relationships are observed, where the VDL approach performs better than the local predictions but lower than the central architecture. We employed the AUC metric in the same way for the Adult Dataset, and Figure 2 illustrates that the VDL approach achieves higher scores compared to the local parties, approaching the performance of the central architecture.

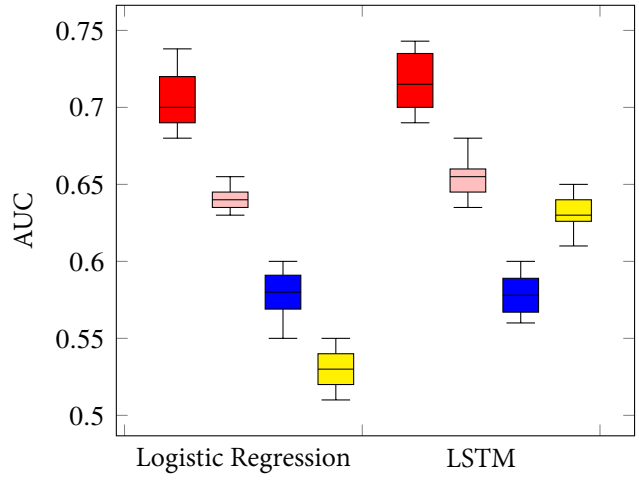
Furthermore, we noticed that the LSTM consistently outperforms the LR technique in all figures. This is



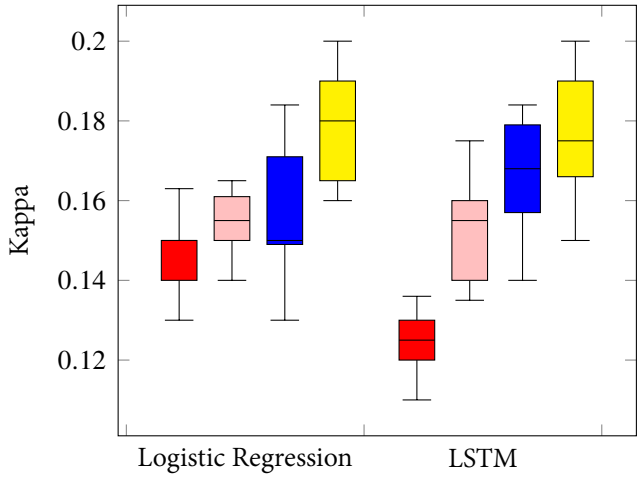

**(a)** AUC for MIMIC-III Dataset (In Hospital Mortality)




**(b)** AUC for MIMIC-III Dataset (Decompensa-tion)




**(c)** AUC for MIMIC-III Dataset (Phenotyping)




**(d)** Cohen's (Kappa) for the MIMIC-III Dataset (Length of Stay Forecasting)

**Figure 3.** AUC for the MIMIC-III Dataset [RED: Central], [PINK: VDL], [BLUE: Local1], [YELLOW: Local2]

expected as LSTM benefits from multiple epochs and continuous improvement in deep NN training. When LSTM approaches the LR score, we conclude the epoch rounds to demonstrate LSTM's dominance over LR.

To ensure the confidentiality of the shared score, we implement the epsilon ($\varepsilon$) DP algorithm[12]. Epsilon, a pivotal parameter in DP, significantly influences the ML score or model performance. It acts as a decisive factor in determining the degree of privacy protection applied to the data, thereby introducing a trade-off between privacy and utility. As epsilon decreases, the level of privacy protection intensifies, albeit at the expense of utility or the accuracy of the ML model. Lower epsilon values permit the introduction of more noise into the data, enhancing privacy but potentially diminishing the accuracy of the model's predictions—a phenomenon evident in our experiment. Meanwhile, the increase of epsilon values just performs the opposite. In our study, we specifically utilized the epsilon range of 1 to 5. Each box in Figure 3 represents the lower, upper, and median scores of an ML model. The upper value signifies $\varepsilon = 5$, while the lower part of the box denotes $\varepsilon =$

1. Consequently, each box encapsulates a privacy-utility spectrum, with the upper portion exhibiting higher utility at the cost of lower privacy and the lower portion featuring lower utility with increased privacy noise. Our experimental results consistently demonstrate that the median converges to a point where an optimal balance between privacy and utility is achieved, effectively avoiding the inherent trade-off. In the realm of DP, Laplacian noise is commonly introduced to the data to fulfill privacy guarantees. The degree of noise injected is modulated by epsilon. Lower epsilon values correspond to increased noise, serving to safeguard privacy but potentially distorting the original data and compromising the model's accuracy. Striking the right balance is pivotal, with the choice of epsilon contingent on specific privacy requirements and the acceptable level of impact on the model's accuracy.

In conclusion, our proposed technique maintains an acceptable level of accuracy for both datasets, even with a minimal privacy budget of 1 to 5. The score disparities observed in comparison to the central server can be attributed to noise addition during local result sharing and the weighted feature computation with loss minimization on the central server. This highlights the trade-off between privacy and utility in this context. However, given that our approach maintains data locality and sensitive patient information remains local, it can be a viable solution in the healthcare sector where privacy preservation is crucial. Our architecture provides a robust privacy-preserving distributed ML framework in this regard.

## 6. DISCUSSION

We have presented a privacy-preserving distributed ML framework specifically designed for vertically partitioned data. In our approach, each client employs both LR and LSTM NNs to generate local predictions based solely on their respective local feature sets. To ensure an additional layer of privacy, we introduce noise into the prediction results using the DP algorithm. Furthermore, we apply a weighted feature function computed from the local feature sets to the final prediction process. Throughout the entire training procedure, no raw data, features, or model parameters are shared among the parties. This effectively mitigates the risk of sensitive data exposure, ensuring strong privacy protection while maintaining a high level of utility through data localization. To showcase the practical application of our system, we conduct experiments using the MIMIC-III and Adult datasets in the healthcare domain. The results demonstrate that our solution, which combines local features with a fully centralized architecture, achieves accuracy levels that are almost comparable to the centralized model. This validates the effectiveness of our FL approach, which preserves data locality and privacy while effectively utilizing vertically partitioned data.

## DECLARATIONS

### Acknowledgments

We sincerely thank the reviewers for their insightful comments. This manuscript is based on the dissertation "Privacy-Preserving Federated Learning Model for healthcare data"[45].

### Authors' contributions

Performed data acquisition and experiments, conducted initial full writing, and provided material support: Islam TU

Made substantial contributions to the conception and revisions of the writing: Mohammed N, Alhadidi D

### Availability of data and materials

Not applicable.

### Financial support and sponsorship

This research was supported in part by the NSERC Discovery Grants (RGPIN-04127-2022).

### Conflicts of interest

All authors declared that there are no conflicts of interest.

### Ethical approval and consent to participate

Not applicable.

### Consent for publication

Not applicable.

### Copyright



## REFERENCES


1. Kwekha-Rashid AS, Abduljabbar HN, Alhayani B. Coronavirus disease (COVID-19) cases analysis using machine-learning applications. *Appl Nanosci* 2021:1–13. DOI
2. Choudhury O, Park Y, Salonidis T, et al. Predicting adverse drug reactions on distributed health data using federated learning. *AMIA Annu Symp Proc* 2019;2019:313–22. Pubmed
3. Xu J, Glicksberg BS, Su C, et al. Federated learning for healthcare informatics. *J Healthc Inform Res* 2021;5:1–19. DOI
4. Vaid A, Jaladanki SK, Xu J, et al. Federated learning of electronic health records to improve mortality prediction in hospitalized patients with COVID-19: machine learning approach. *JMIR Med Inform* 2021;9:e24207. DOI
5. Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. CCS '15. New York, NY, USA: Association for Computing Machinery; 2015. pp. 1322–33. DOI
6. Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP); 2017. pp. 3–18. DOI
7. Choudhury O, Gkoulalas-Divanis A, Salonidis T, et al. Differential privacy-enabled federated learning for sensitive health data; 2020. Available from: https://arxiv.org/pdf/1910.02578. [Last accessed on 19 Dec 2023]
8. Choudhury O, Gkoulalas-Divanis A, Salonidis T, Sylla I. Anonymizing data for preserving privacy during use for federated machine learning. Google Patents; 2021. US Patent 11,188,791. Available from: https://patentimages.storage.googleapis.com/82/a7/42/f741ab230e217a/US11188791.pdf. [Last accessed on 19 Dec 2023]
9. Islam TU, Ghasemi R, Mohammed N. Privacy-preserving federated learning model for healthcare data. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). IEEE; 2022. pp. 0281–87. DOI
10. Liu Y, Kang Y, Zhang X, et al. A communication efficient collaborative learning framework for distributed features. arXiv; 2020. ArXiv:1912.11187 [cs, stat]. DOI
11. Chen T, Jin X, Sun Y, Yin W. VAFL: a method of vertical asynchronous federated learning. arXiv; 2020. ArXiv:2007.06081 [cs, math, stat]. DOI
12. Dwork C, Lei J. Differential privacy and robust statistics. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. STOC '09. New York, NY, USA: Association for Computing Machinery; 2009. pp. 371–80. DOI
13. Hu Y, Niu D, Yang J, Zhou S. FDML: a collaborative machine learning framework for distributed features. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19. New York, NY, USA: Association for Computing Machinery; 2019. pp. 2232–40. DOI
14. Bishop CM. Neural networks and their applications. *Rev Sci Instr* 1994;65:1803–32. DOI
15. Gurney K. An introduction to neural networks. CRC press; 2018. DOI
16. Yu Y, Si X, Hu C, Zhang J. A review of recurrent neural networks: LSTM cells and network architectures. *Neur Comput* 2019;31:1235–70. DOI
17. Understanding LSTM networks – colah's blog;. Available from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Last accessed on 19 Dec 2023]
18. Staudemeyer RC, Morris ER. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:190909586* 2019. DOI
19. Wang Z, Song M, Zhang Z, et al. Beyond inferring class representatives: user-level privacy leakage from federated learning. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE; 2019. pp. 2512–20. DOI
20. Melis L, Song C, De Cristofaro E, Shmatikov V. Exploiting Unintended Feature Leakage in Collaborative Learning. In: 2019 IEEE Symposium on Security and Privacy (SP); 2019. pp. 691–706. DOI
21. Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics. PMLR; 2020. pp. 2938–48. Available from: https://proceedings.mlr.press/v108/bagdasaryan20a.html. [Last accessed on 19 Dec 2023]
22. Song M, Wang Z, Zhang Z, et al. Analyzing user-level privacy attack against federated learning. *IEEE J Select Areas Commun*

2020;38:2430–44. DOI
23. Rajkumar A, Agarwal S. A differentially private stochastic gradient descent algorithm for multiparty classification. In: Artificial Intelligence and Statistics. PMLR; 2012. pp. 933–41. Available from: https://proceedings.mlr.press/v22/rajkumar12.html. [Last accessed on 19 Dec 2023]
24. Kikuchi H, Hamanaga C, Yasunaga H, et al. Privacy-preserving multiple linear regression of vertically partitioned real medical datasets. *J Inform Process* 2018;26:638–47. DOI
25. Fang H, Qian Q. Privacy preserving machine learning with homomorphic encryption and federated learning. *Fut Internet* 2021;13:94. DOI
26. Zhang C, Li S, Xia J, et al. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In: 2020 USENIX annual technical conference (USENIX ATC 20); 2020. pp. 493–506. Available from: https://www.usenix.org/conference/atc20/presentation/zhang-chengliang. [Last accessed on 19 Dec 2023]
27. Xu G, Li H, Liu S, Yang K, Lin X. Verifynet: Secure and verifiable federated learning. *IEEE Trans Inform Forensic Secur* 2019;15:911–26. DOI
28. Zhu H, Goh RSM, Ng WK. Privacy-preserving weighted federated learning within the secret sharing framework. *IEEE Access* 2020;8:198275–84. DOI
29. Zhang Y, Jia R, Pei H, et al. The secret revealer: generative model-inversion attacks against deep neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2020. pp. 253–61. Available from: https://openaccess.thecvf.com/content$_C$VPR$_2$020/html/Zhang$_T$he$_S$ecret$_R$evealer$_G$enerative$_M$odel-Inversion$_A$ttacks$_A$gainst$_D$eep$_N$eural$_N$etworks$_C$VPR$_2$020$_P$aper.html. [Last accessed on 19 Dec 2023]
30. Geyer RC, Klein T, Nabi M. Differentially private federated learning: a client level perspective. *arXiv preprint arXiv:171207557* 2017. DOI
31. Cho H, Wu DJ, Berger B. Secure genome-wide association analysis using multiparty computation. *Nat Biotechnol* 2018;36:547–51. DOI
32. Hu Y, Liu P, Kong L, Niu D. Learning Privately over Distributed Features: An ADMM Sharing Approach. arXiv; 2019. ArXiv:1907.07735 [cs, stat]. Available from: http://arxiv.org/abs/1907.07735. [Last accessed on 19 Dec 2023]
33. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR; 2017. pp. 1273–82. Available from: https://proceedings.mlr.press/v54/mcmahan17a?ref=https://githubhelp.com. [Last accessed on 19 Dec 2023]
34. Harutyunyan H, Khachatrian H, Kale DC, Ver Steeg G, Galstyan A. Multitask learning and benchmarking with clinical time series data. *Sci Data* 2019;6:96. DOI
35. Ji Z, Jiang X, Wang S, Xiong L, Ohno-Machado L. Differentially private distributed logistic regression using private and public data. *BMC Med Genomics* 2014;7:S14. DOI
36. Carlini N, Tramer F, Wallace E, et al. Extracting training data from large language models. In: 30th USENIX Security Symposium (USENIX Security 21); 2021. pp. 2633–50. Available from: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting. [Last accessed on 19 Dec 2023]
37. Chaudhuri K, Monteleoni C, Sarwate AD. Differentially private empirical risk minimization. *J Mach Learn Res* 2011;12:1069. Available from: https://www.jmlr.org/papers/volume12/chaudhuri11a/chaudhuri11a.pdf. [Last accessed on 19 Dec 2023]
38. Abadi M, Chu A, Goodfellow I, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna Austria: ACM; 2016. pp. 308–18. DOI
39. Yeom S, Giacomelli I, Fredrikson M, Jha S. Privacy risk in machine learning: analyzing the connection to overfitting. In: 2018 IEEE 31st computer security foundations symposium (CSF). IEEE; 2018. pp. 268–82. DOI
40. Watson L, Guo C, Cormode G, Sablayrolles A. On the importance of difficulty calibration in membership inference attacks. *arXiv preprint arXiv:211108440* 2021. DOI
41. Papernot N, Abadi M, Erlingsson U, Goodfellow I, Talwar K. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:161005755* 2016. Available from: https://doi.org/10.48550/arXiv.1610.05755. [Last accessed on 19 Dec 2023]
42. Papernot N, Song S, Mironov I, et al. Scalable private learning with pate. *arXiv preprint arXiv:180208908* 2018. Available from: https://doi.org/10.48550/arXiv.1802.08908.[Last accessed on 19 Dec 2023]
43. Bagdasaryan E, Poursaeed O, Shmatikov V. Differential privacy has disparate impact on model accuracy. *Advances in neural information processing systems* 2019;32. Available from: https://proceedings.neurips.cc/paper$_f$iles/paper/2019/hash/fc0de4e0396fff257ea362983c2dda5a-Abstract.html. [Last accessed on 19 Dec 2023]
44. Torfi A, Fox EA, Reddy CK. Differentially private synthetic medical data generation using convolutional GANs. *Inform Sci* 2022;586:485–500. DOI
45. Islam TU. *Privacy-preserving federated learning model for healthcare data* 2023 Feb. Available from: http://hdl.handle.net/1993/37192. [Last accessed on 19 Dec 2023]