Privacy-Preserving Federated Learning Model for Healthcare Data

by

Tanzir Ul Islam

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
February 2023

Thesis advisor

Noman Mohammed

Author

Tanzir Ul Islam

Privacy-Preserving Federated Learning Model for Healthcare Data

Abstract

Federated Learning (FL) is a method for training machine learning algorithms on decentralized data where sharing the raw data is not feasible due to privacy regulations. An instance of such data is Electronic Health Records (EHRs), which contain confidential patient information. In FL, the sensitive data is not shared, rather local models are trained and the model parameters are then aggregated on a central server. However, this method presents privacy challenges, necessitating the implementation of privacy protection strategies, such as data anonymization, before sharing the model parameters. Balancing the trade-off between privacy and utility is a crucial aspect in FL research, as integrating privacy algorithms can have an impact on the utility. The objective of this thesis is to improve the performance of FL while maintaining privacy, through techniques like data generalization, feature selection for dimension reduction, and minimizing noise in the anonymization process. This research also investigates separating data based on features instead of records and evaluates the performance of the proposed model using real healthcare data, with the aim of developing a predictive model for healthcare applications.

Contents

List of Figures

List of Tables

Dedicated to my family for their unconditional support and encouragement

Chapter 1

Introduction

Machine learning models hold great potential for the analysis of sensitive information, particularly in the field of healthcare. These algorithms have the ability to analyze medical images and patient data, helping to diagnose diseases and support treatment decisions. ML models can also be utilized to predict the likelihood of patients developing certain conditions or complications, allowing for early prevention. Furthermore, the integration of machine learning into medical systems can assist healthcare providers in making informed decisions by providing real-time, evidence-based recommendations. For example, research conducted by Zheng et al. [1] showed the use of Electronic Health Records (EHR) in identifying type-2 diabetes through machine learning. Another study by Choudhury et al. [2] tested the effectiveness of using EHR and a distributed machine learning algorithm to predict Adverse Drug Reactions (ADR) in patients.

One of the major challenges in applying machine learning in healthcare research is the distribution of data. Many clinics are reluctant to release their raw EHR due to legal laws and data privacy regulations, making it difficult to establish centralized data repositories across geographical locations. As a result, the raw data remains within the boundaries of each institution. In reality, a patient's data is distributed across several sites or clinics, rather than being confined to a central

repository. This data can be divided in two ways: horizontally or vertically. Horizontally partitioned data silos occur when each site has a unique collection of records, yet their data share common features. In general, a horizontal partition involves distributing the rows of a table across multiple database clusters. However, in many current contexts, such as healthcare, it is necessary to process data from multiple sites for the same set of records, with different sets of attributes, as opposed to a horizontally partitioned arrangement. Vertical partitioning involves distributing data where the same set of records has distinct attributes on each site. For example, one clinic may have patient data with attributes like name, age, and disease code, while another may have attributes like name, address, and ICU stay timestamps. Therefore, it is important to design a framework that can perform computations using summary statistics from participating sites, rather than relying on centralized storage of raw data in both horizontal and vertical settings. This framework should also ensure data privacy and provide a usable utility. Fig. 1.1 illustrates horizontal and vertical data partitions. In the horizontal setting, for the different records 1 to 6, the feature set is the same $\{x_1, x_2, y\}$, whereas in the vertical partition, for the same records 1, 2, and 3, the feature set is unique for each data source.

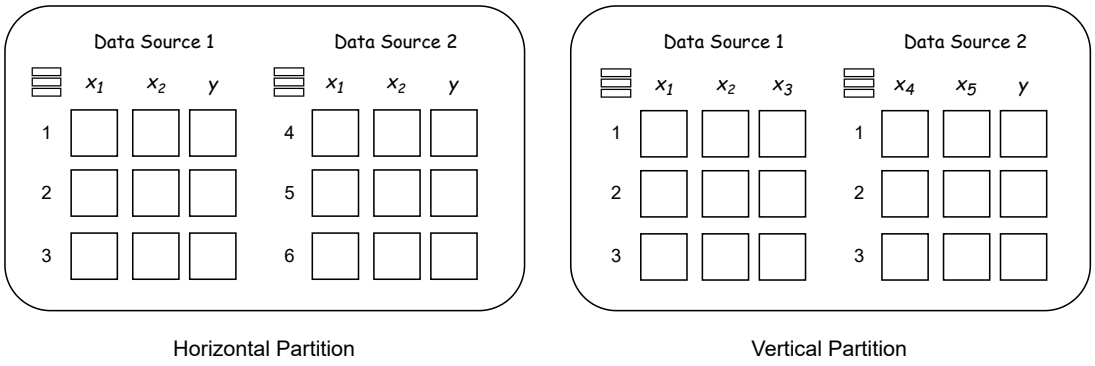


Figure 1.1: Exmaple of Data Partition

Federated learning is a new approach for training machine learning models iteratively on distributed data. The gradient descent method is typically used by the sites to train a global model using their local data in each iteration. The updated parameters of the local models are sent to an

aggregation server and integrated into the overall model. The sites are then given access to the updated global model for the next training cycle, and this process continues until a certain criterion is met. This approach reduces the need for transmitting raw data outside the facility and enables machine learning to be trained on data from multiple locations without sending raw data to a central server. Only the parameters from locally created machine learning models will be shared, while raw data will stay with the local storage providers. However, the federated learning model has privacy concerns, such as the risk of model inversion or reconstruction attacks [3, 4] on the data supplied to the model. These attacks aim to reconstruct the original training dataset using the model parameters. An intruder could also determine if a specific person's data was used to train the model. To protect data from such attacks, most current approaches use *Data Anonymization* or *Differential Privacy* (DP) algorithm [2, 5, 6] to add a layer of privacy to the federated learning framework. However, privacy comes at a cost, and the differential privacy method, which adds noise to the data, impacts the framework's utility. The more noise added, the more privacy is provided, but the less utility is available. Therefore, choosing the right privacy method that balances privacy and utility constraints and model performance is an important research objective in the field of federated learning and our research focuses on this.

1.1 Contributions

This thesis investigates various methods for constructing a federated machine learning (ML) system that balances privacy and utility in both horizontal and vertical data distributions. To enhance privacy with utility in horizontally divided data, the proposed frameworks use *Feature Selection* and *Data Sanitization* techniques. For vertically partitioned data, all feature information is required for final training, therefore our framework employs a distributed parallel learning method to gather local predictions in a central server.

Following we discuss the technical contributions of this thesis in detail.

Privacy-Preserving Federated Learning Model with Feature Selection

In this work, we introduce a comprehensive data-sharing approach for Federated Learning aimed at predicting heart failure diseases. Our framework incorporates the use of differential privacy to ensure privacy while making predictions. To improve the efficiency of our predictions, we employ a feature selection technique that reduces the dimensionality of the data. By doing so, the data dimension gets reduced as well as the used of noise through differential privacy, leading to improved utility while maintaining privacy. Our proposed method achieves 79% and 98% accuracy with a maximum privacy budget of 30. The results of this chapter appear in IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) [7].

Privacy-Preserving Federated Learning Model with Data Sanitization

In this work, we present a privacy-sensitive data-sharing framework for FL by utilizing a differential privacy approach. Our framework employs a generalization technique that utilizes a top-down taxonomy tree to summarize the raw data by records or rows, followed by adding count-based noise through differential privacy. The use of generalization shrinks data and minimises the noise applied, thereby improving the utility, while differential privacy ensures data privacy. Our approach achieves at most 77% accuracy against the baseline model with a maximum privacy budget of 5. The results appear in IEEE International Conference on Bioinformatics and Biomedicine (BIBM) [8].

Privacy Preserving Vertical Distributed Learning

In this study, we presented a privacy-preserving distributed machine learning framework that was specifically designed for the healthcare sector and applied to vertically partitioned data sets. Our approach involves the use of a weighted feature implemented at each local site, which results in

a high-performing score comparable to a centralized architecture. We tested the efficacy of our method by conducting experiments with real-life health data from the MIMIC-III dataset, using both linear regression and LSTM-based deep neural network models in four different applications. Our approach was able to achieve near-baseline accuracy while still preserving a level of privacy.

1.2 Organization of this Thesis

This thesis is organized as follows:

- Chapter 2 discusses related works and necessary background materials, which are utilized by the different methods proposed in this thesis.
- Chapter 3 shows a privacy-preserving federated learning model with feature selection to reduce data dimension.
- Chapter 4 shows a privacy-preserving federated learning model with data sanitisation to lessen the use of noise and improve utility.
- Chapter 5 describes a vertical distributed learning technique with a weighted feature algorithm, to mitigate the necessity of sharing feature information between sites.
- Chapter 6 concludes the thesis.

Chapter 2

Background

In this chapter, we will explain some of the ideas that are necessary to comprehend the approaches provided in this thesis. Some broadly related works will also be discussed. Specific related works will be explored in subsequent chapters.

2.1 Preliminaries

2.1.1 Federated Learning

Federated learning [9] is a distributed machine learning technique that develops a model across a number of dispersed sites before aggregating it on a centralised server. It is also referred to as collaborative learning, in which a local model is taught at each site before all models are aggregated and trained at the central server. Every local model update is available to the aggregating server. However, this model assumes that there is no communication between the local models in terms of raw data transmission.

During the local model training, model data is shared with an aggregated trusted server based on predetermined criteria. The aggregated server's global model is then updated and shared with the local sites for additional training. Until a convergence requirement, such as loss function mini-

mization and reasonable accuracy, is fulfilled, the process is repeated. Numerous machine learning techniques, such as Naive Bayes Classifier, Random Forest Model, Logistic Regression (LR), or K-Nearest Neighbor, are frequently used to assess accuracy. In our approach, we will mainly use *Random Forest*, *Naive Bayes*, and *Logistic Regression (LR)* model to evaluate the performance in FL.

2.1.2 Differential Privacy

Differential privacy [6] is a technique that involves introducing some noise to already-existing data. It guarantees that adding or deleting a record won't substantially alter the results, making it impossible for a viewer to draw any important conclusions about any specific individual in the database. The ϵ variable controls the security level and can be altered in accordance with various conditions and security policies. The scheme offers more security the higher the ϵ.

Definition 1 (ε-Differential Privacy)**.** *An algorithm X is differentially private [6] if for any dataset D_1 and D_2 are differed by at most a single record and for all sets $S \in R$, where R is the range of X,*

$$Pr[X(D_1) \in S] \leq e^{\varepsilon} \times Pr[X(D_2) \in S]. \tag{2.1}$$

In this case, two datasets that only differ by one record are said to be neighbours.

Here, the non-negative parameter ε can be used to represent the algorithm's privacy budget.

In order to ensure differential privacy, it is common practise to introduce random noise to a function's actual output. The noise is calibrated based on the function's sensitivity. The greatest difference between a function's outputs from two datasets that only differ by one record is the function's sensitivity.

2.1.3 Generalization

Assuming D as "dataset" with multiple records and domain of D can be represented as $\Omega(D) = \Omega(A_1), ..., \Omega(A_d)$. Here, $A = \{A_1, ..., A_d\}$ is a list of attributes for a particular record d from D and we assume that each attribute A_i has a finite domain, denoted by $\Omega(A_i)$. A value of an attribute is replaced with a more general value by generalization to sanitise a data collection D. The attribute partition determines the precise general value.

Definition 2 (Generalization)**.** *Generalization is defined by a function $\phi = \{\phi_1, \phi_2, ..., \phi_d\}$, where $\phi_i : v \rightarrow p$ maps each value $v \in \Omega(A_i)$ to a $p \in P(A_i)$.*

Here, $P(A_i)$ is the partition of numerical attribute with intervals $\{I_1, I_2, ..., I_k\}$ in $\Omega(A_i)$ which includes the union of all the intervals.

As an example, according to the taxonomy tree in Fig. 4.1 and raw data from Table 4.1a, Post-Secondary is the general value of *Undergrad* or *Postgrad* of Education. In addition, the interval (17-33] can be used to represent age 20. These intervals are determined adaptively from the dataset for numerical attributes.

2.1.4 Vertical Federated Learning (VFL)

Vertical Federated Learning is a machine learning technique that enables organizations with vertically partitioned data to build and train a machine learning model in a decentralized manner without compromising data privacy and security. In vertical federated learning, data samples are distributed across different parties, each party having its own set of features or attributes. The goal of this technique is to train a global machine-learning model using these vertically partitioned data samples while ensuring that the raw data remains confidential and secure. To achieve this, each party trains its own local model based on its data and a central server then aggregates the results to build the global model. This approach is suitable for scenarios where organizations have data silos and

want to leverage machine learning to build better models without sharing the raw data with each other. In VFL, the data is partitioned in such a way that two nodes share the same user profiles but have different feature information. These nodes could be various health institutions or healthcare data application providers. The aim of VFL in this case is to construct a comprehensive model by combining patient features from multiple institutions without directly exchanging patient data. 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:

$$I_i = I_j = \begin{cases} X_i \neq X_j & i \neq j \\ Y_i \neq Y_j \end{cases} \tag{2.2}$$

For example, from each client for same data sample I can be denoted as $\{I_1^i, I_2^i, I_3^i\}$ and $\{I_1^j, I_2^j, I_3^j\}$ with different features $\{x_1^i, x_2^i, x_3^i\}$ and $\{x_3^j, x_4^j, x_5^j\}$. Supposedly, a hospital and a nearby immunisation centre are two separate healthcare institutions that operate in the same area. Because they are locals, the patients who use these two healthcare facilities may be largely similar. However, given that vaccine centres preserve users' immunisation histories and hospitals keep their medical treatment histories, the user features may not be related in any way. VFL safely integrates various feature sets to improve model performance while preserving data locality.

2.1.5 LSTM Neural Networks

A neural network (NN) is an interconnected collection of discrete processing "nodes", or units, whose operation is somewhat analogous to a biological neuron. The nodes or units are synthetic versions of biological neurons. Each input is multiplied by a weight before being delivered to the analogue of the cell body since synapses are represented by a single integer or weight. Here, basic arithmetic produces a node activation by adding the weighted signals together. The calculation is performed by an activation function, which may produce a result of zero or one. Defining the

model structure (such as the number of input features and outputs) and initialising the model's parameters before running them in a loop are the three basic processes in the construction of a neural network [10, 11]. After that, the current loss (forward propagation) or current gradient (backward propagation) calculations and parameter updates (gradient descent) are made. Preparing the dataset and tuning the learning rate can both have a significant impact on the algorithm.

Recurrent neural networks (RNNs) are a type of neural network that allows for information to be passed from one step of the network to the next. This makes them well-suited for tasks that involve sequential data, such as language translation and speech recognition. RNNs process input sequences element by element, maintaining an internal state that encodes the context of the sequence up to that point. This allows the network to use the information from earlier elements in the sequence when processing later ones, which is what gives RNNs the ability to capture dependencies between elements in the input sequence. There are various types of RNNs, such as long short-term memory (LSTM) networks and gated recurrent units (GRUs).

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) that is well-suited to model 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 encodes information about the past. LSTMs are a variant of RNNs that have an additional "memory cell" that can store information for an extended period of time, as well as three "gate" mechanisms (input, output, and forget gates) that control access to and modification of the cell. The equations for the forward pass of an LSTM cell can be denoted below:

Forget gate's compact forms:

$$\hat{c}^{<t>} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{2.3}$$

Update gate's activation vector:

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{2.4}$$

Forget gate's activation vector:

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{2.5}$$

Output gate's activation vector:

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{2.6}$$

Finally, cell state vector c^t:

$$c^t = \Gamma_u * \hat{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{2.7}$$

This is the standard set of equations used to update the hidden state and memory cell in an LSTM at each time step. $x^{<t>}$is the input at time step t, $a^{<t>}$ is the hidden state at time step t, $a^{<t-1>}$ is the hidden state at the previous time step. W is the weight matrices for the input, forget, output, and memory cell updates, respectively, and b is the bias. The function $\sigma(x)$ is the sigmoid function, which maps a value to the range [0, 1]. The function $tanh(x)$ is the hyperbolic tangent function, which maps a value to the range [-1, 1]. A detailed explanation of the LSTM algorithm can be found in [12–14].

2.2 Related Work

For the past few years, distributed machine learning research has been focused on data privacy and security in sensitive data analysis, e.g. *Healthcare Data*. In this part, we give a quick overview of a number of related works.

2.2.1 Federated Learning and Privacy Attacks

Federated learning indicates a distributed learning method where sharing sensitive data in raw format is impractical for preserving data privacy. The advantages of using FL techniques over traditional, centralised ML models for a number of sensitive data applications have recently been

demonstrated by studies. However, there are possible privacy attacks in this distributed framework, such as inference [4], reconstruction [15, 16], or backdoor [17, 18] attacks, where data can be regained back from the exposed information. Rajkumar et al. [19] showed how DP algorithm can effectively minimize the attack as well as maintain a plausible utility. Choudhury et al. [2] used this same DP approach in FL architecture for healthcare data, however, showed how the DP noise can adversely affect the data utility as well. The main advantage of their approach is that local models may be utilised to train the global server, obviating the requirement to send sensitive raw healthcare data outside of the institution. Their subsequent contribution [20] attempted to balance this privacy-utility trade-off by achieving a tailored *k-anonymity* to reduce DP noise utilisation as much as feasible. They demonstrated strong model performance based on an empirical assessment of one million patients' health records, producing better results than standard DP. Our latest work [7] was also implemented by the motivation of "data sensitization first, noise applied last" to minimize the DP noise in model data, and improve the overall data utility of the framework.

Encryption [21–23] or secret sharing [24–26] through secure multiparty communication is another method for privacy-preserving machine learning. Models are communicated using encrypted data in these frameworks. However, encryption requires more computational resources and cannot be used for all scenarios. Furthermore, frequent data transfers between clients are necessary for secure multiparty communication, which also introduces communication overhead.

It is observed that distributed ML techniques on horizontal data have been given more focus for the past few years. On the other hand, our work enables feature-parallel machine learning among nodes with vertically partitioned data which is equally significant and has not yet been investigated more extensively.

2.2.2 Privacy-Preserving Federated Learning on Horizontal Data

FL's privacy issues can be lessened by adopting Privacy-Preserving Federated Learning Models, which experts can use to lower the danger of re-identification. These techniques aim to teach a global model with the greatest degree of accuracy while maintaining the donors' data security. The main foundation of existing approaches is ε-differential privacy [6, 27] in a FL environment, which guarantees verified privacy regardless of an adversary's prior information.

With the help of iterative model averaging through stochastic gradient descent (SGD) and a thorough empirical evaluation that takes into account five distinct model architectures and four datasets, Mcmahan et al. [28] proposes a workable strategy for the federated learning of deep networks. Geyer et al. [27] introduced a DP-based approach for health applications in FL settings, but did not consider the scenario where servers can be honest-but-curious and breach the private data in the training process. In order to add an additional layer of privacy, authors in [2] presented a similar federated learning paradigm for health care data utilising DP. The quantity of noise added by the DP technique caused both implementations to perform significantly worse when the data dimension was relatively large. Choudhury et al. proposed customized anonymity as (k, k^m) anonymity [20] to mitigate the privacy-utility trade-off of [2] by proposing a modified k-anonymity model on set-valued dataset. They showed high model performance focused on an empirical study on over one million patients' health records which showed better output than traditional differential privacy. This method focused on performance optimization using an upgraded version of the data anonymity model to preserve privacy.

2.2.3 Privacy-Preserving Federated Learning on Vertical Data

There have been a number of research efforts focused on developing privacy-preserving techniques for vertical distributed learning. These techniques aim to protect the privacy of the data used for training while still allowing the model to be trained effectively. One common approach is to use

differential privacy [6, 27], which adds noise to the data in a way that preserves the privacy of individuals while still allowing the model to learn useful patterns. Other techniques include using secure multi-party computation (SMC) [29] to allow multiple parties to jointly compute the model without revealing their data to each other, and using homomorphic encryption to encrypt the data and allow the model to be trained on the encrypted data without revealing the underlying data.

Liu et al. [30] using SMC approach showed a Federated Stochastic Block Coordinate Descent (FedBCD) algorithm, where each party conducts multiple local updates to minimize communication overhead. Their approach works in vertically partitioned data, sharing a single value only instead of the model or raw data to maintain data privacy. Hu et al. [31] experimented in a similar way for VFL by only sharing a single value for model training by an *Alternating Direction Method of Multipliers* (ADMM) approach which is commonly used for distributed ML approach. In their approach, they also used the ε, δ-differential privacy algorithm to perturb the shared value by Gaussian noise. When dealing with distributed features, the perturbed method makes sure that the probability distribution of the values communicated is mostly insensitive to any modification to any one feature in a party's local dataset. With theoretical privacy guarantees, their method converges with very fewer epochs than the state-of-the-art SGD approach. Chen et al. [32] also showed a VFL approach in an asynchronous way, performing the communication between each party using the SMC method. They used *perturbed local embedding* with the Gaussian DP noise to add the extra layer of data privacy and experimented on both logistic regression and deep learning for healthcare data.

To jointly learn and train any classification algorithm in a VFL scenario, it is essential for each partner to communicate their own unique feature information. The majority of currently used technologies exchange feature information with clients via the SMC technique. Some strategies utilise homomorphic encryption or DP for data perturbation or sanitization to protect privacy while sharing this feature information. Hu et al. [33] described a feature-distributed collaborative learning strategy in which each client undergoes independent training and the final output of the prediction

is shared with a central server for the computation of the final score. By making predictions based purely on the available local features in a similar parallel computing method, this approach preserves data locality in an asynchronous SGD approach. Additionally, they used Laplacian DP noise to protect the shared prediction sent to the main server. Their technique, however, did not expressly target healthcare data. They utilised the NN deep learning technique, however, the LSTM-based approach fared better. In a similar way, we use a customized weighted feature algorithm and DP technique to securely exchange the prediction result within the central server in order to ensure data locality and privacy in a VDL context targeting healthcare data.

Chapter 3

Privacy-Preserving Federated Learning Model with Feature Selection

Federated Machine Learning (FL) can be used effectively in distributed datasets, where data owners hesitate to share their raw data, as a reliable approach to train an ML algorithm. However, in the case of sensitive healthcare datasets, additional privacy measures before feeding into machine learning mechanisms are also necessary as the exposed data in ML training may have privacy attacks. Our approach uses the federated learning framework, which removes the necessity of sharing patients' sensitive data in a raw format outside the premise. First, the data owners agree on a list of features selected by the correlation; then, after training the local models, the obtained local models are transmitted to the central server for aggregation. The differential privacy (DP) approach is adopted to perturb the local models before transmission to add an extra privacy layer. As a result, our framework achieves improved utility as the feature selection reduces the data dimension as well as noise usage by DP. Finally, based on the patient's genomic data, the framework establishes a practical healthcare application to privacy-predict certain heart failure/cancer diseases.

3.1 Introduction

Machine learning (ML) models have promising applications for healthcare data. For example, learning from this real-world health data can be utilized to build medical diagnostic tools, predict disease risk factors or analyze gene sequence data for medical treatments. Zheng et al. [1] showed how machine learning model can be utilized to identify type-2 diabetes by analyzing electronic health records(EHR). Choudhury et al. [34] also experimented with how successful the application can be using a federated machine learning model to predict patients' adverse drug reactions (ADR) by analyzing electronic health data. ML algorithms analyze these EHR to identify hidden patterns to be leveraged in diagnosing and predicting diseases that have considerable benefits, e.g. increasing treatment probability. ML is trained by the provided training data and gets ready to be used in real situations.

Several medical regulatory policies restrict healthcare data access due to containing sensitive patient information. Hence, rethinking data analytics methodologies for healthcare applications is required to employ such data while adhering to privacy policies. Generally, ML algorithms need to access the accumulated data to achieve the best accuracy. However, health data can expose sensitive information about individuals in its raw form, and publishing such data will violate their privacy. Therefore, in healthcare use cases, collecting all data in a centralized centre might not be feasible, hindering data owners from sharing their raw data. Furthermore, the shared data of a person releases some information about their relatives who may not give consent for sharing information about their healthcare data. A distributed ML algorithm can be an effective alternative solution in this regard.

Federated Learning (FL) is a new ML technique that proposes a solution to mitigate the necessity to share raw EHR outside of health premises. In this approach, a ML is trained on distributed data hosted by different data owners without passing raw data to a centralized manager. Raw data will stay at local storage providers. They can monitor the transmitted information and pursue their

own privacy policy, which can be modified or changed according to unforeseen circumstances. This is in contrast to centralized models. When the data is handed into a centralized manager, data owners cannot control what the manager will do with the accumulated data. However, model data transmitted in a federated learning approach can still have privacy attacks known as model inversion or reconstruction attacks [3,4]. The goal of a reconstruction attack is to rebuild the original training dataset using model parameters. Also, an intruder can infer if an individual's data was used for training the model. Therefore, incorporating an additional layer of privacy model to protect data from such attacks must also be managed. Our approach chooses the differential privacy model [5] to add a layer of protection to this FL framework by selecting a proper privacy budget.

In this thesis, we use the FL framework to analyze patients' genomic data to identify the potential risk in patients with various health conditions like heart failure with maintaining privacy. Genomic data are one key area of sensitive health data, revealing our well-being and disease susceptibility are also lawfully protected from public access. Transthyretin amyloid cardiomyopathy is such type of genomic data which contains information on the cause of heart failure. To identify the potential risk of such health conditions, ML model always comes as effective support. Therefore, our federated machine learning model will analyze the patient's genomic data to predict the risk of heart failure by identifying if a group of patients have the potential risk of wild-type transthyretin amyloid cardiomyopathy with known phenotypes. From this set of data, through privacy-preserving federated learning, our generated model should predict whether it will progress towards the disease of heart failure.

An example of the dataset is provided in Table 3.1. The data table shows that around **1874** phenotypes or columns are present from each genomic sample. Another dataset for our experiment is from BC-TCGA with almost **17814** genes.

Our approach builds the model locally with the raw genomic data and sends it to a central server with differential privacy to train the final framework. Only perturbed model data through differential

privacy mechanism is shared with the central aggregated server. The trusted aggregator server trains the global model based on the aggregated local data and builds the final model. The main structure of this FL model consists of two entities. Data owner(s) that provide raw data for training a model and a model manager that all data owners interact with. Notice that data provider(s) cannot communicate any messages with each other. Also, we assume the aggregator server is honest-but-curious and can not regain the raw data of a data owner from the generated model, which is protected by an additional privacy layer through differential privacy.

Table 3.1: Sample Genomic Dataset

Id	$Gene_1$	$Gene_2$	$Gene_3$	...	$Gene_m$	Disease
2356256	1	0	1		0	1
2456246	1	1	0		1	0
6575678	1	1	0		0	0
⋮				...		
n	0	1	0		0	0

The main challenge of our approach is the high-dimensional data with a large number of columns that can directly affect the accuracy and efficiency of any ML model. It can worsen the utility when noise is added to maintain privacy. More columns get less budget within a fixed privacy budget through equal distribution. According to the differential privacy theorem, the less the privacy budget, the more noise will be added to the data. Therefore, more noise can impact the data utility of the model. The current solutions do not address this high dimensionality problem with the privacy model. Our approach will use statistical methods using the highest correlation value to reduce the number of dimensions of the raw data to consume less noise in total and improve the model's accuracy.

Contributions. In this paper, we propose a generalized data-sharing framework for FL us-

ing differential privacy to predict certain heart failure diseases. Our framework reduces the whole data dimension through feature selection, reducing the noise addition for differential privacy while maintaining efficient scores. The contributions can be summarized as follows:

- We propose a feature selection mechanism based on the correlation value of the data to reduce dimension and maximize the utility of the ML models.
- We employ a differentially private technique to enable privacy-preserving data sharing among collaborating parties using FL infrastructure.
- We demonstrate the effectiveness of our proposed technique by experimenting on iDASH 2021 competition dataset [35]. Our proposed method achieves 79% and 98% accuracy with a maximum privacy budget of 30 and 10 for IQVIA and BC-TCGA datasets respectively and takes around 10 seconds for training on a federated training setting.

3.2 Methods

In this section, proposed methods detailing the privacy-preserving techniques will be discussed.

3.2.1 Problem Description

In this model (Fig. 3.1), we use the federated learning framework to analyze patients' gene data to identify the potential risk in patients with various health conditions like heart failure. Data is collected from different locations maintaining proper privacy as due to health data regulatory policies, these data cannot be shared across the premises in raw format. Therefore, we build the model locally and share those with the central server for training. In the proposed scheme, only perturbed model data is shared with the central aggregated server. The trusted aggregator server trains the global model based on the aggregated local data and builds the final model. However, model data

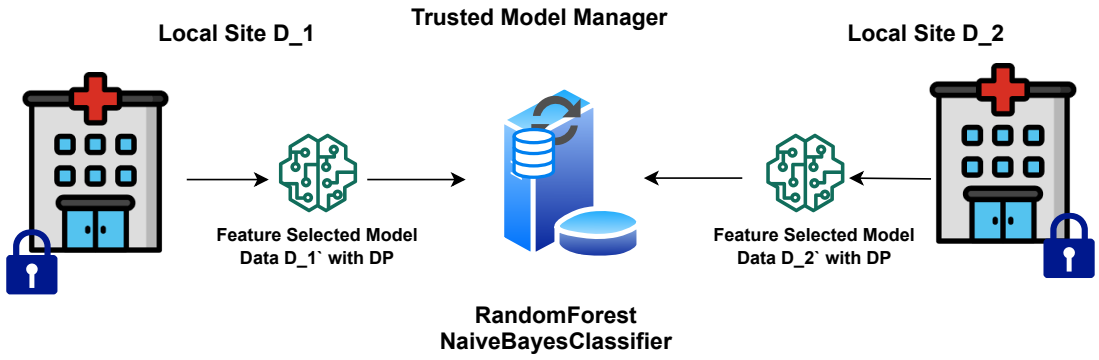


Figure 3.1: Multiple Data Owners are training a model collaboratively using federated machine learning algorithm providing a privacy guarantee over the data

transmitted in a federated learning approach can still have privacy attacks [3, 4, 36] like model inversion or reconstruction attacks. Therefore, an additional layer of privacy is needed. The privacy mechanism is achieved through the differential privacy algorithm by adding noise to the model data through the sharing process between sites and aggregator servers.

High-dimensional data with more columns can affect the model's accuracy if a noticeable amount of noise is added to the data. In order to achieve a shared list of the features that the training is performed a federated correlation-based feature selection is designed to reduce the number of data dimensions and improve the model's accuracy.

Transthyretin amyloid cardiomyopathy is a cause of heart failure. To identify the potential risk of such health conditions, ML model always comes as effective support. Therefore, our federated machine learning model will analyze the patient's gene data to predict the risk of heart failure through identifying if a group of patients have the potential risk of wild-type transthyretin amyloid cardiomyopathy with known phenotypes. The dataset is separated into two parties with a total of 1713 samples, where 855 samples are diagnosed as wild-type amyloidogenic TTR cardiomyopathy (ATTR-CM) (case group) and 858 samples (sample per row) and each sample have 1874 phenotypes (features per column). From this set of data, through privacy-preserving federated learning, our

generated model should predict whether it will progress towards the disease of heart failure. An example of the dataset is provided in Table 3.1

3.2.2 Summary of Methodology

The size of electronic health records and datasets is extensive, which can affect their usefulness when working with limited privacy resources. With a restricted privacy budget, more columns receive less budget, so we mitigate this by reducing the dimensionality through feature selection. Our approach to feature selection involves identifying the columns in the provided data that have the highest correlation value with the disease. To ensure privacy, we utilize Differential Privacy (DP) mechanisms and apply the *Laplace Transformation* to add noise to the derived statistics. Rather than transmitting the raw data, we send the resulting noisy data to the aggregator server. The central aggregator server then uses this data to train a machine learning framework that learns a global model used to predict the likelihood of heart failure.

3.2.3 Feature Selection

Many genes available in a genomic dataset are the first essential factor impacting the model's utility due to diversity in the whole dataset. Any machine learning system processing such vast data dimension often leads to less accuracy because some genes provide no value to the analysis. Therefore, proper methodologies for feature selection from the whole dataset constantly improve the score efficiently.

While reducing the data dimension, we focus on two things: Choosing genes based on their correlation with the disease and sending only summarized data as a model to the central Server. A correlation coefficient is a numerical measure of some type of correlation, meaning a statistical relationship between two variables.

$$corr(X,Y) = \rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu x)(Y-\mu y)]}{\sigma_X \sigma_Y} \tag{3.1}$$

Similarly, in our approach, each local server chooses a list of a maximum of 200-300 Genes that have the highest correlation with the disease as per the above equation to build up the local model. Finally, the central Server receives and matches each local model's columns and feeds them to the ML framework for training.

In our approach, reduced dimensions require less privacy budget (in terms of ε) while maintaining the model's utility. Within a limited budget, if the number of columns is high, the privacy budget will be distributed to each column with less amount, resulting in more noise to be added. Therefore, more noise can impact the accuracy of the score directly. After feature selection in our approach, the data dimension reduces, resulting in higher accuracy through less noise added.

According to Table 3.2, if any gene from the whole data set contains control value as *True*, refers to *Positive Prediction of Heart Failure* while *False* value represents negative prediction. We calculate the correlation value for each column and define it as (μ, σ) for *True* prediction label and (μ', σ') as the *False* prediction label. Then we select the top 200-300 columns based on the correlation value. In this way, we calculate the feature for each column and generate the two rows of data based on the control value as per shown in Table 3.2. Note that these two rows of data do not contain the actual raw value, rather the calculated model data based on the highest correlation for that column. Finally, after noise addition, Table 3.3 is sent to the model manager or central aggregation server with additional layer of protection using differential privacy.

3.2.4 Privacy Mechanism

Our approach solely depends on the differential privacy mechanism to maintain the additional privacy layer of the shared model data. In this approach, the noise will be added to the model data based on *Laplace Mechanism* [6]. The privacy budget, ε, is varied based on the data dimension

Table 3.2: Column Selection based on highest correlation value

$Gene_1$	$Gene_2$	$Gene_3$	Control Value
(μ_1, σ_1)	(μ_2, σ_2)	(μ_3, σ_3)	# Have Heart Disease
(μ'_1, σ'_1)	(μ'_2, σ'_2)	(μ'_3, σ'_3)	# Don't have Heart Disease

Table 3.3: Diffenrential Privacy mechanism applied to the model data

$Gene_1$	$Gene_2$	$Gene_3$	Control Value
$(\mu_1, \sigma_1) + Lap_\Delta f/\varepsilon_1$	$(\mu_2, \sigma_2) + Lap_\Delta f/\varepsilon_2$	$(\mu_3, \sigma_3) + Lap_\Delta f/\varepsilon_3$	Have Heart Disease
$(\mu'_1, \sigma'_1) + Lap_\Delta f/\varepsilon_1$	$(\mu'_2, \sigma'_2) + Lap_\Delta f/\varepsilon_2$	$(\mu'_3, \sigma'_3) + Lap_\Delta f/\varepsilon_3$	Don't have Heart Disease

to achieve the best utility. Data size will be reduced by dimension first based on feature selection rather than applying noise to the complete data sets. Finally, the noise will be added to the summarized data, as shown in Table 3.3. Note that the total privacy budget ε is distributed to each column based on the inverse correlation value. Therefore, the column with the highest correlation value will receive the most privacy budget and have less noise to be added. Also, according to the composition theorem [37] of the DP mechanism as detailed in the Background section, If $F_1(x)$ satisfies ε_1-differential privacy and $F_2(x)$ satisfies ε_2-differential privacy, then the mechanism $G(x) = (F_1(x), F_2(x))$ which releases both results satisfies $\varepsilon_1 + \varepsilon_2$-differential privacy satisfying the following equation:

$$\varepsilon_1 + \varepsilon_2 + \varepsilon_3 + ... + \varepsilon_n = \varepsilon, \varepsilon_i \propto (Corr(\mu_i, \sigma_i))^{-1} \tag{3.2}$$

Therefore, the total privacy budget will be equal to the summation of distributed budget for each column. In this way, the more relevant column with the disease prediction will add less noise value, resulting in an improved utility of the framework.

3.2.5 Federated Learning Mechanism

In our federated framework, after adding the privacy mechanism, only model data from each local data owner are sent to the aggregator server to train the ML framework. We have utilized two ML algorithms for the federated training: a) Naive Bayes Classifier and b) Random Forest in a federated setting to validate the efficacy of the proposed method.

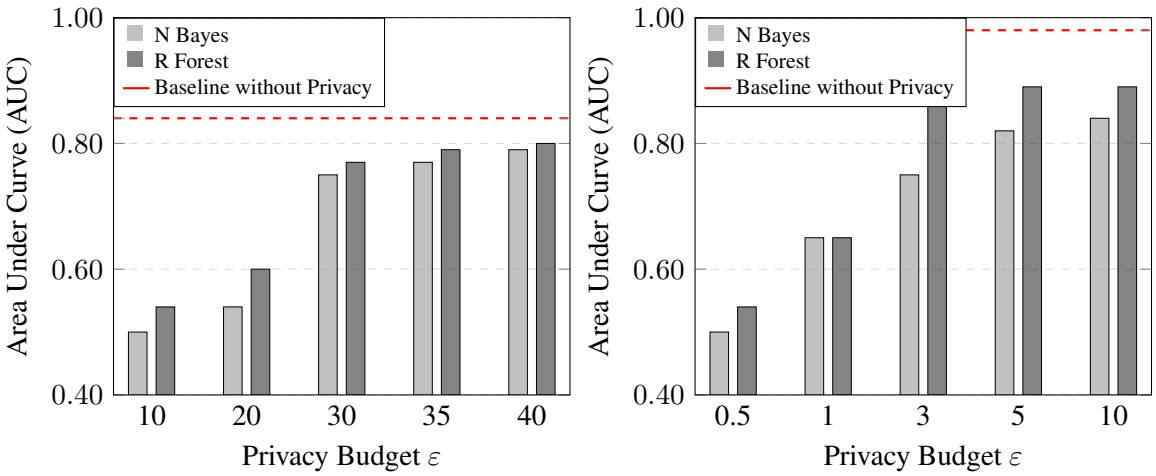
The data owners remain solely responsible for building their local model, while the aggregator server builds the results in a collaborative learning setting. The statistics required for Random Forest and Naive Bayes, for example, are first completed at the data owner's location and then shared with an additional privacy layer (DP) to be robust against further model inversion attacks. For RandomForest, after reducing the column with the highest correlation values, only the two rows with control value *True* or *False* is calculated as (μ, σ) and (μ', σ') respectively. Finally, Table 3.3 is generated at each data owner and sent to the aggregator server after adding the noise according to the Laplace mechanism. The same approach is applied for RandomForest, except the model is built on the Tree from raw data and noise is added afterwards.

3.3 Results

In this section, the experimental result is described. Since the proposed method is a generalized data-sharing mechanism for federated ML applications, we experiment with different settings as portrayed in Table 3.4. We utilized multiple machines at our lab as server-client settings to conduct the experiments. The average latency between the servers was minimal.

3.3.1 Experimental Setup

The experimental data were taken from the iDASH 2021 competition [35] which tested the proposed solutions with a single dataset: IQVIA Inc, for predicting causes of certain heart failure. We utilized



(a) AUC for IQVIA Inc. Dataset

(b) AUC for BC-TCGA Dataset

Figure 3.2: Accuracy difference with different privacy budgets and methods

Table 3.4: Different experimental parameters considered in this approach

Dataset	ML Methods	budget ε	Dimensions
IQVIA Inc.	Naive Bayes	[20, 30, 40, 50, 60]	1874
BC-TCGA	Random Forest	[1, 3, 5, 10, 20]	17814

additional datasets from BC-TCGA [38] for cancer prediction alongside to compare our proposed method:

1. **IQVIA Inc.:** 1713 samples, where 855 samples are diagnosed as wild-type amyloidogenic TTR cardiomyopathy (ATTR-CM) as well as positive cases of heart failure, and 876 negative controls

2. **BC-TCGA:** 17814 genes with 424 positive labels and 50 negative labels

The training data for the ML models were chosen at random in an 80:20 split, with 80% of the data being used in training. In a two-party setup where the data is split into two, the identical training data was used in both Naive Bayes and Random Forest. The training procedure was repeated ten

times, with the Area Under the Curve (AUC) values from each test set being averaged. We also experimented with different parameters, privacy budget ϵ varying data dimension to 200-350 for IQVIA and 20-50 for BC-TCGA. These are outlined in Table 3.4.

3.3.2 Accuracy

We utilised the Area Under Curve (AUC) to evaluate both ML models as the significant accuracy metric. AUC accurately describes the binary classifier since it uses True Positive and False Positive rates to generate the curve (receiver operating characteristic). Furthermore, it chooses several thresholds from $[0, 1]$, with 1 being the most accurate, indicating that the model successfully predicts all data occurrences for all thresholds. However, for every binary classification, an AUC ≤ 0.5 demonstrates that the model is no better than a coin toss. In our scenario, the proposed model can only categorise positive records accurately and not classify negative ones. In Fig. 3.2, we depict the relation between privacy budget, ε and Area Under Curve (AUC) for different methods (Naive Bayes and Random Forest). It shows that the privacy budget of 30 and more provides higher AUC values for the competition (IQVIA Inc.) dataset with the Random Forest algorithm. The reduced dimension is set to be $m' = 250$ for this experiment. Similarly in Fig. 3.2, BC-TCGA datasets were evaluated with higher accuracy within privacy budget 10. The reduced dimension is set to be $m' = 20$ for this case. Finally, in terms of accuracy, we see a similar tendency, with greater ε values resulting in higher AUCs.

Fig. 3.2 also shows similar behaviour in terms of AUC, with Random Forest providing greater AUCs even with various m' values. However, we have noticed that, besides the AUC discrepancy, the Random Forest technique takes longer to execute than Naive Bayes because it requires more computations. It also depicts the behaviour for multiple data dimensions m' as both larger and smaller values adversely affect the results. Smaller m's cause much data loss whereas larger m's consume much ε that adds excessive noise to the data. Therefore, $m' \in \{250, 300, 350\}$ performs

well for $\varepsilon = 30$.

We applied our solution with BC-TCGA data. Note that the overall score seems near the value of 98% for the BC-TCGA data, though IQVIA data can reach up to a maximum of 80% only with DP setting. This is because the maximum baseline score for IQVIA data is 84% without any privacy protection in a central architecture, whereas BC-TCGA data is 98% for this similar settings. Also, IQVIA data contains only binary value, which is known as *Haplotype Data* as well as more refined removing details. Therefore, this data set with dimension reduction affects the overall score requiring more privacy budget than BC-TCGA to achieve the data utility. We believe that this scenario happened due to data loss in this haplotype gene data, which directly impacted the framework's accuracy. In contrast, BC-TCGA data had minor effects on the score as more relevant data were chosen through feature selection for dimension size reduction.

3.4 Conclusion

Most of the existing approaches using the federated learning model for analyzing healthcare data generally do not ensure privacy along with performance. This paper builds a healthcare data analysis framework using federated learning in a private manner. In addition, differential privacy mechanisms are applied through feature selection based on statistical methods to improve scalability and accuracy. In future, we plan to integrate data anonymization with differential privacy to improve the score further through an improved feature selection algorithm.

Chapter 4

Privacy-Preserving Federated Learning Model with Data Sanitization

Federated Learning (FL) is an efficient way to train Machine Learning (ML) algorithms on distributed datasets where data owners are restricted by policies to share their raw data. Through local training and model aggregation to a central server, this method reduces the need to communicate raw data with parties outside of the premises. However, FL raises serious privacy concerns. Therefore, additional privacy measures are required. The differential privacy (DP) approach is a cutting-edge privacy method used to perturb the local models prior to transmission and add an additional layer of privacy. However, this technique can affect the utility of the framework. To balance the privacy-utility trade-off, we implement a private technique to sanitize raw data using a combination of a top-down taxonomy tree and DP noise. The generalized data using DP noise is used to train local models to be shared in the FL architecture. The proposed framework achieves improved utility with a moderate privacy budget.

4.1 Introduction

Machine learning (ML) models offer a lot of prospects in the analysis of sensitive and crucial data. Learning from real-world healthcare data can be applied, for example, to the development of medical diagnostic tools, the identification of disease risk factors, or the evaluation of gene sequence data for medicinal therapy. Zheng et al. [1] demonstrated how examining Electronic Health Records (EHR) may be used to identify type-2 diabetes using machine learning. Choudhury et al. [34] also tested the efficacy of the application for predicting Adverse Drug Reactions (ADR) in patients using EHR and a distributed machine learning algorithm. ML algorithms examine these sensitive data to find underlying patterns that can be used in disease diagnosis and prediction, which has significant advantages, such as boosting the effectiveness of therapy.

Important patient information is contained in sensitive data like healthcare, and various regulatory laws prevent direct access to it. Therefore, it is necessary to reevaluate data analytics approaches for these data while upholding privacy policies. For instance, posting health data will breach people's privacy because it can reveal critical information about them in their raw form. It also might not be feasible to gather all the data from different healthcare clinics in one place because the policies forbid sharing the information outside of the clinic's walls. Additionally, a person's supplied data may reveal details about their family who might not agree to the sharing of their medical information. An efficient alternative approach in this case is a distributed ML algorithm, where model sharing reduces the risk of raw data sharing outside of the premises.

A new approach for iteratively training machine learning models on distributed data is provided by federated learning. The sites typically employ the gradient descent method to train a global model on their local data at each iteration. The local models' parameter updates are transmitted to an aggregation server and incorporated into the overall model. For the upcoming training cycle, the sites are once again given access to the revised global model. This process is repeated

until a certain criterion is met up. The whole process suggests a way to reduce the need to transmit raw data outside of the facility. Without sending raw data to a centralised server, ML is trained on distributed data held by several locations. Only locally created ML models' data will be shared; raw data will remain with local storage providers. Model inversion or reconstruction attacks [3, 4] on the model data supplied in a federated learning approach are still a possibility. Reconstructing the initial training dataset using the model parameters is the goal of a reconstruction attack. Additionally, an intruder can determine whether a specific person's data was utilised to train the model. As a result, it is necessary to manage the inclusion of a second layer of privacy to safeguard data from such attacks. Most of the current approaches choose DP [2, 5, 6] to add a layer of protection to this FL framework by selecting a proper privacy budget. The DP method works by introducing noise to the training model data, which might directly impact the framework's utility. We suggest a method using a differentially private strategy, a combination of data generalisation and DP approaches to improve the framework utility as much as feasible, to balance this privacy-utility trade-off. Our contribution is to instrument the DP generalization algorithm [39] in the domain of federated learning to countermeasure the aforementioned approach.

Our model consists of two parts for data privacy, generalization and noise addition. One of the most widely used methods for transforming raw data to meet a particular privacy criterion is generalization [40, 41]. Generalization reduces the precision of information while maintaining its "truthfulness" by substituting a specific value for a more general one. An example of the dataset sanitization is provided in Table 4.1. Table 4.1b shows the generalized contingency table of Table 4.1a. However, applying this generalized contingency data directly to ML algorithms is vulnerable to minimality attack [36] and does not provide the claimed privacy guarantee. Therefore, we use a unique approach, where raw data is sanitized first through a generalization-based algorithm *DiffGen* [39, 42], which uses top-down taxonomy tree and publishes the leaf node data with DP noise. The robust privacy mechanism DP is adapted to this generalized contingency data like Table 4.2 before

sending it to FL architecture. *Differential privacy* [6] is a strict privacy model that does not assume the knowledge of an adversary. A differentially-private mechanism guarantees that all outputs are insensitive to any individual's data by ensuring that the likelihood of any output (released data) is equal from all near identical input datasets. The utility of the data can be greatly reduced when the added noise is rather considerable compared to the count, hence employing solely DP may not be appropriate for high-dimensional data with a wide domain. In our approach, noise is added to each count of published data to satisfy the privacy requirement and the noisy sanitized data can be fed into the FL model. In this way, noise usage is reduced significantly to improve classification scores.

In summary, our approach converts the raw data table to a general contingency table and applies it to a top-down taxonomy tree-like Fig. 4.1 through specialization. The leaf nodes of the tree are published with noisy counts which mitigates noise usage as much as possible to preserve the data utility. The sanitized released data like Table 4.2 are finally converted to a model and sent to a central server to train the global framework. By using a differential privacy approach, only data from perturbed models are shared with the central aggregated server in this manner. On the basis of the local model data aggregated, the trusted aggregator server creates the final model and trains the global model. Two entities make up the main framework of this FL model. A central server with which all data owners communicate and the data owner(s) who supply the raw data used to train the model. It is assumed that data cannot be exchanged across data providers in our model. Additionally, since the constructed model is shielded by a second privacy layer created by differential privacy, we suppose the aggregator server is honest but curious and cannot recover the raw data of a data owner from it.

Contributions. In this paper, we combine data sanitization and differential privacy to present a generic framework for FL-sensitive data exchange. The contributions can be summarized as follows:

- To enable privacy-preserving data sharing among cooperating parties using FL infrastructure, we use a differentially private approach.

- We used generalization-based algorithm using top-down taxonomy tree to sanitize raw data first and then apply differentially private count based noise to reduce the amount of noise in every level of data for utility improvement
- We demonstrated the effectiveness of the proposed technique by experimenting on two real datasets using three different ML classifications in FL architecture and comparing them with the baseline centralized model. Our approach achieves at most 77% accuracy against the baseline model with 84% maximum score.

(a) Raw Data Table

Education	Age	$Income$
College	20	<=50k
Secondary	17	<=50k
College	31	>50k
Secondary	34	<=50k
Undergrad	20	<=50k
Postgrad	30	>50k
Postgrad	41	>50k
Undergrad	37	<=50k

(b) Contingency Table

Education	Age	Count
Secondary	[17-33)	2
Secondary	[33-70)	1
College	[17-33)	3
College	[33-70)	0
Undergrad	[17-33)	2
Undergrad	[33-70)	0
Postgrad	[17-33)	3
Postgrad	[33-70)	3

Table 4.1: A raw data table and its generalized versions

4.2 Methods

This section will explain the suggested strategies for training a model using local data sanitization and privacy-preserving procedures.

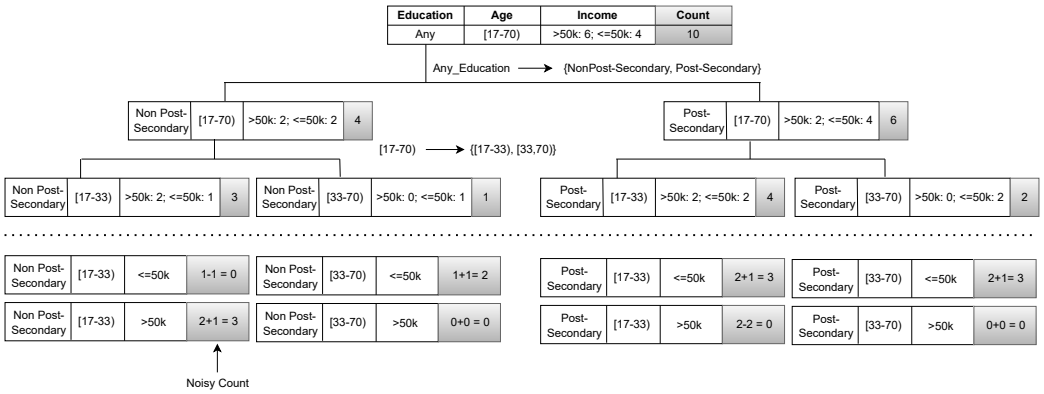


Figure 4.1: Noisy Count added to Leaf Node after partitioning records

4.2.1 Problem Description

In our strategy (Fig. 5.1), we analysed sensitive data using the federated learning architecture to identify or forecast through categorization. Since they cannot be disseminated across the premises in raw format due to regulatory restrictions, data is gathered from various sites while keeping appropriate privacy. As a result, the *DiffGen* [39] method is used through the DP mechanism to first sanitize the raw data. Then the model is built locally on those sanitized data and shared with the central server for training. Only model data that has been perturbed is shared with the central aggregated server under the proposed system. Based on the combined local data, the trustworthy aggregator server creates the final model after training the global model. We use the state-of-the-art stochastic

Table 4.2: Sanitized Representation of Raw Data

Education	*Age*	Count (>50k + <=50k)	Noisy Count
Non Post-Secondary	[17-33)	2+1=3	3+0=3
Non Post-Secondary	[33-70)	0+1=1	0+2=2
Post-Secondary	[17-33)	2+2=4	0+3=3
Post-Secondary	[33-70)	0+2=2	0+3=3

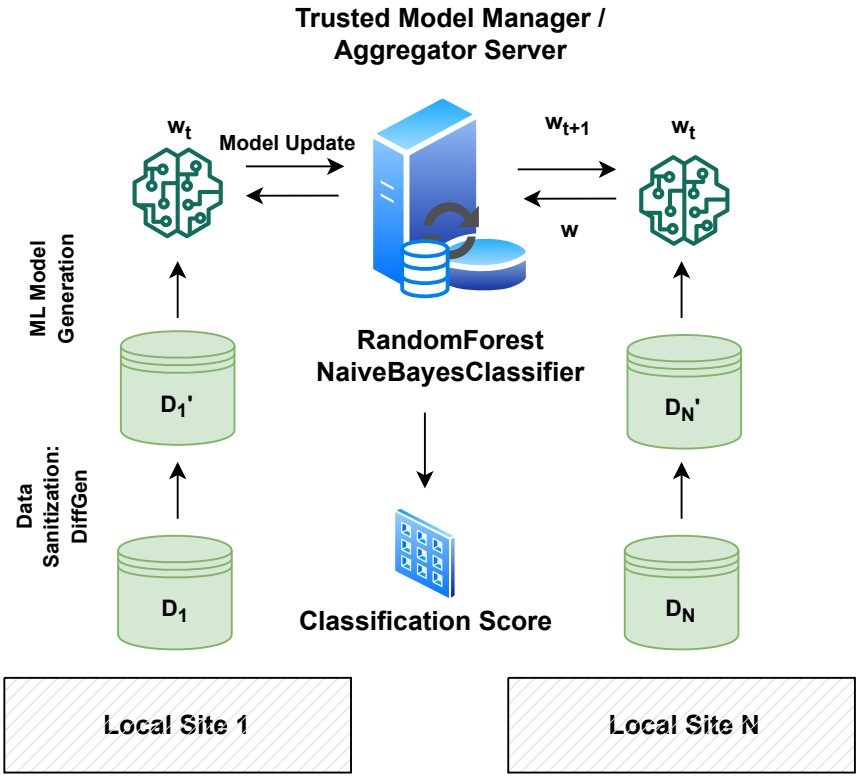


Figure 4.2: Structure of our collaborative federated machine learning algorithm

gradient descent (SGD) [27,28,34] for model optimization of the federated training process. In each iteration of this process, the sites use the gradient descent method to optimize the global model on their local data. The local models' parameter updates are then forwarded to an aggregation server and incorporated into the global model. For the following round of training, the new global model is shared with the sites again for further optimization. Until a convergence requirement, such as loss function minimization, is satisfied, the process is repeated. The paragraphs that follow provide more information on this procedure.

4.2.2 Local Data Sanitization

Before feeding into our federated ML framework, raw data are sanitized through generalization and specialization using top-down taxonomy tree. Any sanitization algorithm will work for our

Algorithm 1: Algorithm for Data Site

Input : Raw data set D, privacy budget ε, and number of specializations h, global model w_t

Output: Private model parameter of classification Algorithm

```
// Raw data D is sanitized using DiffGen algorithm
```

$D' \leftarrow DiffGen(D, \varepsilon, h)$;

```
// Initialize local model with global initial model of
   classification algorithm
```

$w \leftarrow w_t$;

```
// For each local epoch
```

for $e \in [E]$ **do**

```
// For bacth of Sanitized Data D'
```

  for $b \in B$ **do**

    $w \leftarrow w - \eta \nabla F(w)$; `// Gradient Descent optimization`

$w_{t+1} \leftarrow w - w_t$; `// Update local model`

return w_{t+1} *for further global update*

approach, however, to achieve the best performance, we follow the algorithm DiffGen [39] for this purpose. The algorithm first divides the raw data into numerous equivalence groups by generalising the predictor attributes each with the same attribute values. Then it releases the noisy group counts. The general idea is to sanitize the raw data by a series of specializations, starting from the topmost general state as shown in Fig. 4.1. Considering the raw data in Table 4.1a, if it is only converted to a contingency table like 4.1b, the noise amount will be excessive for high dimensional data as more rows with less generalization will consume more noise to add. Therefore, a more generalized version is needed to have a balanced utility. Table 4.2 shows a more generalized version contain-

Algorithm 2: Algorithm for Aggregator Server

Input : Total number of data parties P, Total number of training rounds R

Output: Updated Global Model

Initialize Global Model w_t;

`// Set the model with classification parameter`

$w_t \leftarrow ModelParamter(Classificationalgorithm)$;

`// for each round of training`

for $r \in R$ **do**

  `// for each client in parallel, model update`

  for $p \in P$ **do**

    $w_{t+1} \leftarrow LocalDataSiteUpdate(w_t)$;

`// Aggregate global model through SGD for minimum loss function`

$w_{t+1} \leftarrow w_t - \eta \sum_{p=1}^{P} \nabla F_p(w_t)$

return w_{t+1} *for ML classification*

ing attributes *Education, Age and Income*. The *Education* attribute provides information such as *elementary, secondary, college, undergrad, and postgrad* etc. All values have been generalised to *Non Post-secondary and Post-secondary*. Similarly, the *Age* attribute ranges from 17 to 70, thus it may be summed up as [17-70]. The dependant variable or the label attribute here for the ML classification training is the *Income* attribute, which has values of either > 50k or $<= 50$k. In this example, the topmost tree serves as a root partition, including all records with count values of *Income* attribute that are generalised to {Any Education, [17-70]}.

Then the specialization process starts, as $v \longrightarrow child(v)$, where $child(v)$ denotes the set of child values of v, replaces the parent value v with a child value. The specialisation process is analo-

gous to the downward "cut" of each taxonomy tree. Each root-to-leaf path in a cut of the taxonomy tree for an attribute has exactly one value. For example, as our root of the tree is now {"Any Education", [17-70)} can be denoted as root attribute and the "cut" based on *Education* can push the tree downwards with two leaf-node as $child(v)$. Let the first specialization be Any Education $\longrightarrow$ {Non Post-secondary, Post-secondary}. The algorithm creates two new partitions and splits data records between them denoted as {Non Post-secondary, [17-70)} and {Post-secondary, [17-70)}. By specialising a few split values in the current cut, the specialisation begins at the topmost cut and iteratively pushes the cut down. Similarly, as shown in Fig. 4.1, the next cut is executed using age group [17-70) $\longrightarrow$ {[17-33), [33-70)} and the leaf is specialized into two children. At each iteration *DiffGen* probabilistically selects a candidate $v \in Cut$ using an exponential algorithm. The exponential mechanism ensures privacy as well as it also exponentially favours a candidate with a high score. This satisfies ε-differential privacy because the probability of choosing any value is proportional to $exp(\frac{\varepsilon' u(D,v)}{2 \triangle u})$. Detailed proof for this theorem is given in [39].

Each node stores the count value, as seen in Fig. 4.1. In the root node, it is shown as 10, however, notice that after specialisation through *Education* attribute, each child node gets a count value of 4 and 6 correspondingly. It can be observed that it is the real count in each node throughout the specialisation process if the *Income* attribute values are noticed for each node. When the specialisation process is completed at the leaf node, however, the real count is not released since disclosing the actual values of these groups violates differential privacy. According to the Laplace process, these values are countered by adding noise to each group's count. Each true count of the groups is given Laplace noise via this *DiffGen* algorithm. Because the sensitivity of the count query is 1, it also satisfies $\varepsilon-$differential privacy [6]. The output is a list of each leaf partition's equivalence groups, together with their noisy counts. Finally, by letting the specialisation continue until it reaches the leaf level of the attribute domains, a contingency table similar to Table 4.2 is produced.

In the following section, we explain how Table 4.2 is converted to the respective classification

model and sent to the central aggregation server for the federated machine learning training.

4.2.3 Building the Local Model on Sanitized Data

The framework starts training from a central server by sharing a global model across all sites, following the FL standard. The model is built using sanitized local datasets after *DiffGen* mechanism, and parameter modifications are then included in the global model. This technique is repeated until the global model converges. Here, the data owners perform their computations and the aggregator server aggregates the results needed for the ML operation. We use three classification models that can be trained using gradient descent optimization: Logistic Regression, Random Forest, and Naive Bayes for this federated training process.

Initially, the global model is shared with each site, which trains the model on its local data. After computing the average gradient, the updated local parameter *w* is aggregated to the shared global model and sent back to the central server. The process is continued till the minimization of loss function $F_n(w)$ is reached. If there are *N* sites, for example, we divide the training data into *N* disjoint subsets of the feature set $\{X^i_{train}\}^N_{i=1}$ and the corresponding label set $\{Y^i_{train}\}^N_{i=1}$. Let *R* stands for the number of rounds in which local model updates are aggregated, *E* number of epochs, η the learning rate and *batch* based on a given batch size B, respectively, for stochastic gradient descent. At a fixed learning rate η, for each data site n, average gradient ($\nabla F_n(w)$) [27,28] is computed with respect to its current model parameter *w*. Here, $F_n(w)$ is the local loss function of the n^{th} site with respect to its model parameter *w*. The model update will follow the equation: $w^n \leftarrow w^n - \eta \nabla F_n(w^n)$. Epoch *R*, the number of training runs each client does on its local dataset every round, and **B**, the local minibatch size from the sanitised dataset D' used for client updates, are the key factors affecting the amount of computation. To indicate that the entire local dataset is processed as a single minibatch rather than a portion of it, we write $B = \infty$. This step is required, as the *Diffgen* algorithm already generalizes the raw data with the best specialization,

further partitioning in the training process with batch might lose the utility of the overall framework.

Algorithm 1 refers to the process in each client site. A global classification ML model w_t is initialized (Naive Bayes, Random Forest or LR in our approach) and shared to each local site at the startup of the federated training process. Initially, at Line 3, the DiffGen algorithm converts the raw data D to sanitized data D' using DP algorithm and the output is similar to Table 4.2. After that, the global model w_t is updated in the local site using the local model created by the local sanitized dataset. Each site updates its local model to the shared global model based on the epoch, the maximum number of passes over the training data, as indicated in Line 5. As previously stated, the batch is obtained from the sanitised data D' and prepares an updated model parameter at Line 6 using the SGD method. Finally, for aggregation, this model is returned to the central aggregator server.

Table 4.2, for example, shows that the sanitised representation of raw data now comprises four rows. For the federated training process, we can consider two local data sites and one central aggregator server as shown in our architecture in Fig. 5.1. Each of the data sites can have a portion of this sanitized data. We divided the data horizontally into two parties for this example. As a result, after the split, each party can have two rows, as shown in Table 4.3a & 4.3b.

Table 4.3: Example of Splitting data into Local Party

(a) After splitting data into Party 1

Education	Age	Noisy Count
Non-Post Secondary	[17-33)	3
Post Secondary	[17-33)	3

(b) After splitting data into Party 2

Education	Age	Noisy Count
Non-Post Secondary	[33-70)	2
Post Secondary	[33-70)	3

For each data site or party, the feature column for the model of classification algorithm is the attribute {*Education, Age*} and can be referred to X^i_{train} and the classification label is the column

Table 4.4: Sample Naive Bayes Model for Party 1

Income	μ(Educ.)	σ(Educ.)	(μ)(Age)	σ(Age)
>50k	5.4	0.0035	36	1.57
<=50k	6.2	0.0091	55	0.43

Noisy Count referring Y^i_{train} for i^{th} site. The model w is prepared as $w \leftarrow \{X^i_{train}, Y^i_{train}\}$ for each classification algorithm in each party.

As per Naive Bayes theorem, mean (mu) and variance ($sigma$) is calculated to prepare the model data for the Naive Bayes classifier. Table 4.4 shows a sample model representation for the above sample data. For the Random Forest classifier, a model is constructed using a combination of decision trees with the above sample data. The data is used to generate a number of decision trees, and the class with the most votes is chosen as the final solution for the classification problem. As a result, for the aforementioned sample data, Fig. 4.3, displays the model of the Random Forest classifier for this sample data of party 1, which shows a combination of multiple Decision trees. Due to the fact that party 1's Feature attribute *age* contains only 17-33 values, two decision trees are created with the same root node. Similar models are constructed in each party for logistic regression using the model parameters, and these models are then shared with the central server for training.

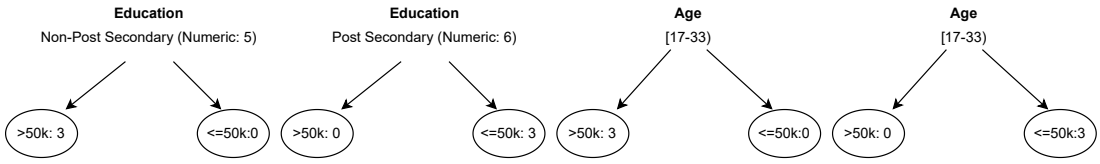


Figure 4.3: Sample Random Forest Model for Party 1 - Combination of Decision Trees

Finally, the model parameter w based on respective classification algorithm is prepared and ready to be sent to the aggregator server through the SGD optimization process as mentioned in Algorithm 1 for a round of iteration and continuous update. The process stops once the convergence

criterion is met. Lastly, the aggregator server builds the final model and trains the ML classifier. It is important that just the model representation displayed in Table 4.4 and Fig. 4.3 is conveyed during this training phase. Here, the original data is covered up using generalisation and given an extra layer of privacy protection by adding DP noise. Due to the use of a noisy count, there are extremely few opportunities to recover data from this model. This protects against any form of membership inference attacks during FL training and preserves privacy.

4.2.4 Aggregator Server Execution

The statistics required for the LR, Random Forest, and Naive Bayes are updated at the local sites and then shared with the aggregator server for the continuous iteration to the final result. Finally, the aggregator server updates the model to feed into the classification training, as described in Algorithm 3. The training approach for each classification method begins with each participant in the FL architecture sharing a model skeleton. After that, each local party modifies their local model and updates to send it back to the central server. The iteration continues until the epoch, which is the number of passes over the training data that must be made until the convergence condition is reached. The global model is updated with all of the local models after the final iteration. The model is aggregated with optimization to minimise the loss function on line 6 using the SGD approach:

$$w_{t+1} \leftarrow w_t - \eta \sum_{p=1}^{P} \nabla F_p(w_t) \tag{4.1}$$

Finally, the model is given into the machine learning algorithm, which generates a classification score. Privacy is protected throughout the training process: raw data is never exchanged with the server from the client. Sanitized data also meets the ε-DP requirement using DP noise. As a result, the entire federated training process strictly adheres to data privacy. Whether the server is honest-but-curious, it will not be able to learn anything from the sanitised model data supplied by the clients.

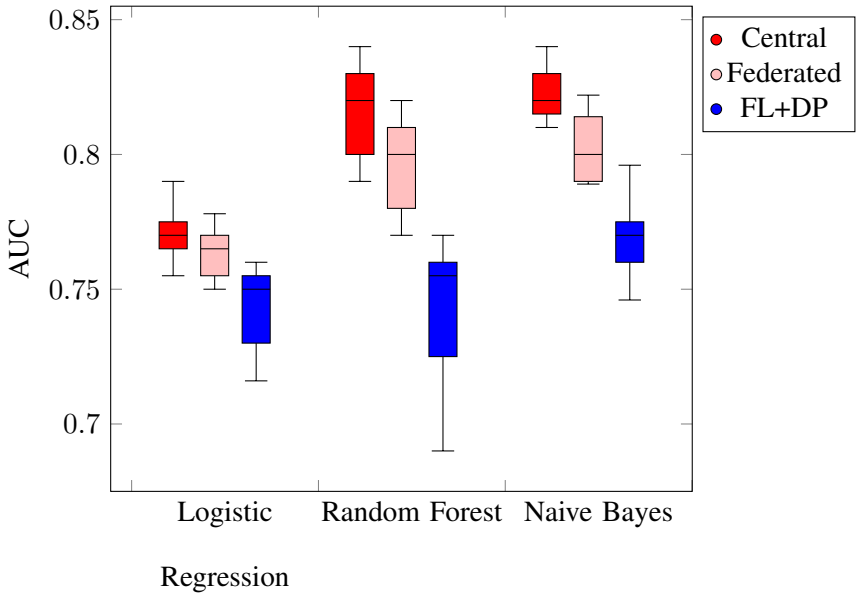


Figure 4.4: AUC for Adult Dataset with Privacy Budget ε : 1 to 5

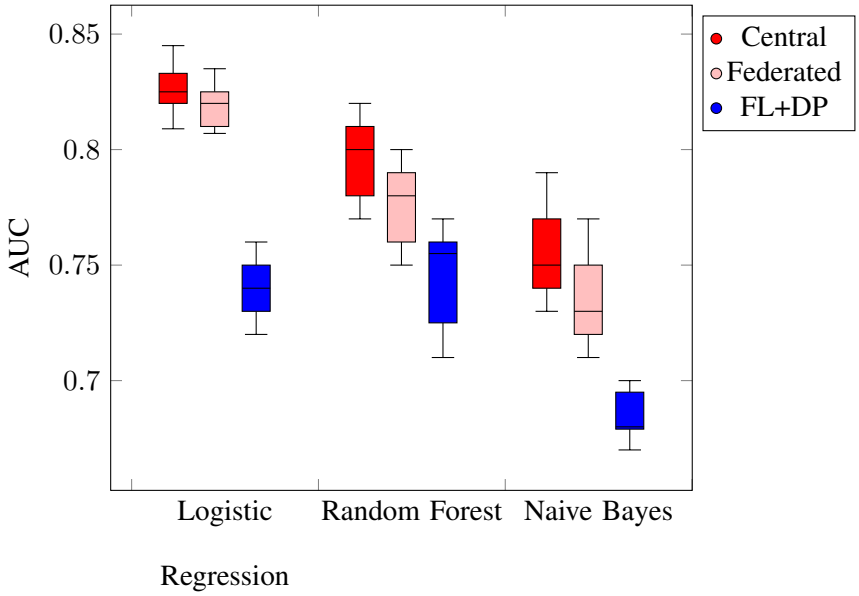


Figure 4.5: AUC for MIMIC III Dataset with Privacy Budget ε : 1 to 5

4.3 Results

The experimental outcome is described in this section. We test various privacy budget ε values because the suggested solution is a broad data sharing mechanism for federated ML applications. To conduct the research, we used a number of machines in our lab as server-client settings.

4.3.1 Experimental Setup

To establish benchmark results, we first developed centralized learning models and FL models without applying sanitization and privacy. Thereafter, we employ DP and our approach to make a proper comparative evaluation. We used three classification algorithms, namely Naive Bayes, Logistic Regression and Random Forest. To evaluate the models, prior to and after employing sanitization and privacy-preserving mechanisms, we measure their utility in terms of Area Under Curve (AUC) score. We used two datasets, Adult Census data from UCI repository and Medical Information Mart for Intensive Care (MIMIC III) dataset [43], a publicly available benchmark data. According to [44], we chose 17 physiological variables, including demographic information from MIMIC III dataset.

An 80:20 split of the training data for the ML models were made at random, with 80% of the data being used for training. The Area Under the Curve (AUC) values from each test set were averaged after the training algorithm ran for ten epochs. For both datasets, we also tried out various privacy budget ε values ranging from 1 to 5.

4.3.2 Accuracy

We plotted AUC within a defined privacy range, which depicts a binary classifier accurately because it creates the curve using True Positive and False Positive rates (receiver operating characteristic). In Fig. 5.2 and Fig. 4.5, we depict the relation within a fixed privacy budget (1 to 5). The red, pink and blue boxes denote non-private central, non-private federated settings and our customized

approach respectively. It demonstrates that as compared to the non-private centralised model, an effective AUC value for classification is provided with a privacy budget of at least 5, which is close to the baseline.

For the three classifiers, we first compared our solution to the Adult Census data in Fig. 5.2. For all of the classifiers, the overall score was close to the range of 79% percent. We feel that the score decline is the result of the data suppression provided by sanitization and DP noise in our sanitized model. For MIMIC III data, the training was implemented similarly and the overall score was recorded near the value of 76% for a budget of 5 or more. The maximum baseline score for MIMIC data is 84% for Logistic Regression without any privacy protection in a central or FL architecture. Fig. 4.5 shows the box plot depicting this.

In summary, our approach requires a minimum privacy budget to maintain an acceptable level of accuracy for both datasets. We assume that this scenario occurred as a result of projected data loss caused by noise disturbance and sanitization in the federated settings. This also generates the discussion on how the privacy-utility trade-off is related in this setting. The predictive performance of the federated models combined with the sanitised technique is plausible because our strategy provides sufficient privacy while maintaining an acceptable amount of data utility. However, FL suffers considerably more severely from performance reduction when using only ε-differential privacy. Therefore, further experiments on different datasets for different privacy budgets are our future work direction.

4.4 Conclusion

Existing methods that use the federated learning model to analyse sensitive data typically do not provide optimum performance and privacy protection. With a private data analysis framework being built on federated learning, our method focuses on utility improvement. To ensure the best accuracy,

we adopt data sanitization techniques with differential privacy mechanisms to minimise the use of noise at every stage of the training process. Through experimental evaluation using two real-world datasets and different parameter settings, we demonstrated that our technique offers good model performance while providing a justifiable level of data privacy. To further strengthen the score with a better privacy algorithm, we intend to incorporate vertically partitioned data analysis on different datasets in a deep learning setting.

Chapter 5

Privacy Preserving Vertical Distributed Learning

Federated learning is becoming a key tool for mining remote datasets where data owners are unable to share raw data due to privacy concerns. Data is often analyzed using records from different locations that are trained and combined into one central server. In this distributed setting, data can be vertically and horizontally partitioned. In our approach, we choose the specific vertical partition learning process in which the data is segmented by characteristics or columns for the same record across all local sites. This framework can be called Vertical Distributed Learning (VDL) as well as features distributed machine learning. Our design follows the Stochastic Gradient Descent (SGD) approach for this scenario to collaboratively learn from each local site and merge the final result into a central server. Throughout the training phase, no raw data or model are shared, only local prediction results are shared for aggregation. However, sharing the local prediction raises some privacy issues, which we address by introducing noise into the local results using Differential Privacy (DP) algorithm. Thus, our ideas suggest a robust vertical distributed learning system that respects user privacy. We conducted experiments using the sensitive healthcare data *MIMIC-III*

for its four types of applications in this architecture and the publicly accessible *Adult* data with deep learning. The findings of our experiment, which were compared to training only from local features and a completely centralised architecture, showed accuracy that was almost as effective as the centralised model. Therefore, our solution offers an effective federated learning solution maintaining data locality and privacy while utilising vertically separated data.

5.1 Introduction

Machine learning (ML) research is becoming more prominent in the healthcare sector, and it has the potential to save lives through case prediction in addition to providing useful healthcare applications. For instance, the development of medical diagnostic tools, the detection of disease risk factors, or the assessment of gene sequence information for therapeutic purposes can all benefit from learning from real-world healthcare data. Kwekha et al. [45] demonstrated how machine learning may be used in healthcare and plans to investigate, predict, and discriminate among COVID-19 instances to analyse and triage them. Choudhury et al. [34] used health data and a federated machine learning algorithm to examine the effectiveness of the application for anticipating adverse drug reactions (ADR) in patients. Numerous other studies have already shown that ML is capable of doing important healthcare jobs including disease diagnosis.

Data distribution is the main challenge in healthcare machine learning (ML) research since most clinics are unwilling to share their raw Electronic Health Records (EHR) due to various healthcare privacy regulations. Due to this, a patient's data is not confined to a central repository, rather distributed through several sites or clinics. This data can be distributed or separated in two ways, horizontally or vertically. When each client has a unique collection of records, yet their data has features in common is known as horizontally partitioned data silos. In general, horizontal partitioning typically maintains a table's rows across several database clusters. Many current contexts, such

as healthcare, demand processing data across numerous sites for the same set of records on various sets of attributes, in contrast to the horizontally partitioned arrangement. Vertical partitioning is a sort of data distribution where the same set of records can have distinctively different attributes on each site. As an illustration, one clinic might have patient data with the following attributes: name, age, and disease code, whereas another clinic might have data with the following attributes: name, age, and timestamp of ICU stays for the same group of patients. The case of training on horizontally partitioned data is considered in most current ML research. The vertically partitioned data has received little attention in this regard. ML training on this vertically partitioned data distribution for the healthcare industry is the focus of our research. A healthcare organisation may use this type of data segregation if it wants to evaluate the patient's health using clinical data from various locations.

Central machine learning architecture is not a workable solution to distributed data silos. Federated Learning (FL) [46,47] is a distributed machine-learning approach that addresses this problem. FL permits several data owners to jointly develop and make use of a shared prediction model while maintaining the privacy of all local training data. This method iteratively improves model accuracy by allowing each site to update its local model and exchange locally computed gradients or model parameters with a central server. The entire procedure points to a strategy to lessen the requirement to send raw data outside of the facility. ML is trained using distributed data stored across multiple sites rather than delivering raw data to a centralised server. Raw data will remain with local data providers, and will only be shared with locally constructed ML models. The sharing of model data in distributed learning creates several privacy concerns as well. An attacker may be able to replicate sensitive data from the shared information [3, 4]. As a result, the FL method needs an additional layer of privacy to protect it further.

In a horizontal setting, most recent approaches use data anonymization or differential privacy algorithm to add the privacy layer. Choudhury et al. [2, 20] showed how customized *k-anonymity* or differential privacy can be utilized in FL setting to preserve privacy with an acceptable utility.

They chose the healthcare data to predict mortality rates based on patients' ICU stays. In a similar way, the majority of the current research in this area discussed how to integrate appropriate privacy algorithms into this federated learning to conform to healthcare regulatory policies. Our earlier research [7] also suggested a framework that only communicates model data after sufficient data sanitization to protect the privacy of the patients with data utility. For vertically separated data, the FL setting differs from the reasonably well-researched horizontal FL configuration in that it has its special characteristics and challenges. Local models in vertical FL need the data from the models at the other sites to learn about all the features and jointly train the ML model. The existing methods [30, 32] mostly rely on encrypted multiparty communication to learn about each other's features. However, these cryptographic techniques reduce privacy concerns at the expense of computational overhead. Our layout lessens the necessity to learn about the features of other sites for training. No model or feature sets, just local prediction is exchanged with the server for score aggregation.

In our vertical architecture, models are trained locally based on the available features. Only local prediction from each site is shared with the central server. The server merely serves as an aggregator for the total predictions; it does not perform any training. Using a mini-batched stochastic gradient descent (SGD) technique carried out in the sense of distributed computation, the full model is trained from beginning to end. The final result is shared when local training is complete. Different parties are permitted to execute different iterations of parameter updates, which asynchronously simulate our design. The local feature set is mapped to a local prediction for classification using Logistic Regression (LR) and long short-term memory (LSTM) neural networks, and the local predictions are shared for aggregating the final prediction in the central server.

Examples of how feature sets can be dispersed for the same patients are presented in Tables 5.1a and 5.1b. A group of patients' ICU stays are listed in the first table as $P_1(ID, InTime, OutTime)$, and their disease codes are listed in the second table as $P_2(ID, Age, DiseaseCode)$. Each site locally determines the prediction of the mortality rate or probability of a particular acute disease

respectively and shares the result with the server. On the server side, no model parameters are exchanged; instead, only local predictions are exchanged, as indicated in Table 5.3. The server determines the final prediction for each classification process using a weighted feature approach. There is still a privacy concern as attackers can still regain original data using only model predictions [26,48]. Therefore, privacy attacks may still occur if the direct prediction is shared. We use the well-known privacy method Differential Privacy (DP) [6] to perturb the local prediction before sharing to mitigate this concern. To determine the final prediction, the server only receives the perturbed prediction. In this way, we still have a robust vertical distributed learning framework with data privacy even though we lose some accuracy owing to DP noise addition.

Maintaining privacy and an efficient level of accuracy without incurring a substantial communication overhead is the major challenge in vertical distributed learning. Our approach maintains data privacy with very low communication overhead through sharing prediction results only. After sending each local prediction to the central server, we apply a weighted feature approach to use in order to build its final accuracy. Hu et al. [33] suggested a comparable feature distributed collaborative learning technique using a continuously differentiable function to aggregate local intermediate predictions as a weight parameter similar to nonlinear transformations. Our contribution follows in a similar way to randomly shuffle and weight local prediction which effectively influences the ultimate score. For instance, if a local prediction has a feature set x_1 and x_2 that can contribute greater accuracy if it is trained in a central architecture with all the feature sets, our weighted feature algorithm will give it more weight. Sample weighted matrix is shown in Table 5.2.

Contributions. The contributions can be summarized as follows:

- We demonstrated a vertical partitioned FL framework maintaining proper privacy.
- We targeted the healthcare sector with four different types of applications.
- We demonstrated the effectiveness of the proposed technique by experimenting with *Adult*

and *MIMIC-III* datasets with LR and LSTM-based deep neural networks and compared them with the baseline centralized model. Our approach achieves accuracy almost near to the baseline model with privacy. For *Mimic-III* dataset, our solution acquires up to 80% and 82% accuracy and for *Adult* dataset, 90.3% and 90.4% accuracy, for LR and LSTM methods respectively. In summary, our model loses at most 5% of accuracy due to noise addition in a distributed learning framework.

(a) Raw Data At Site 1 (ICU Stays)

Patient Id	In Time	Out Time
901	2155-02-25	2155-02-27
902	2157-12-02	2173-12-31
903	2157-12-02	2173-12-31
904	2173-12-31	2173-12-31
905	2173-12-31	2174-01-14
906	2129-03-21	2164-02-06
907	2129-03-21	2164-02-06
908	2129-03-21	2164-02-06

(b) Raw Data At Site 2

Patient Id	Age	Disease Code
901	34	110
902	70	380
903	33	14
904	55	28
905	47	483
906	67	109
907	77	107
908	58	486

Table 5.1: Feature Distribution over sites and Server Data Representation

5.2 Methods

This section will explain the proposed approach for training a model using vertically separated data and privacy-preserving procedures.

Table 5.2: 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

Table 5.3: Data Available In Central Server

Classification	Site 1 (Prediction)	Site 2 (Prediction)	Final Prediction
Logistic Regression	0.67	0.75	0.84
LSTM	0.56	0.70	0.86

5.2.1 Problem Overview

In the proposed vertical federated learning (VFL) architecture, as shown in Fig. 5.1, models are trained locally based on the features available at each site. The central server serves as an aggregator for the overall predictions but does not perform any training. The training of the complete model is carried out using mini-batched stochastic gradient descent (SGD) [27, 28, 34] at each local party. After local training is completed, the final results are shared with the central server. Logistic Regression (LR) and long short-term memory (LSTM) neural networks are used to classify the local feature set into a local prediction, which is then aggregated by the central server to produce the final prediction. To determine the final score, an appropriate feature weight is required for each feature, which is calculated using Logistic Regression with the SGD technique. This weight matrix, along with the local prediction results, is aggregated by the central server. This design eliminates the need for communication between each party and reduces the risk of data exposure. The local predictions are also perturbed with differential privacy (DP) noise to provide a more robust privacy-preserving model, although this may result in a slight loss of accuracy in the final result.

We will evaluate the performance of the model using an evaluation metric and aim to minimize

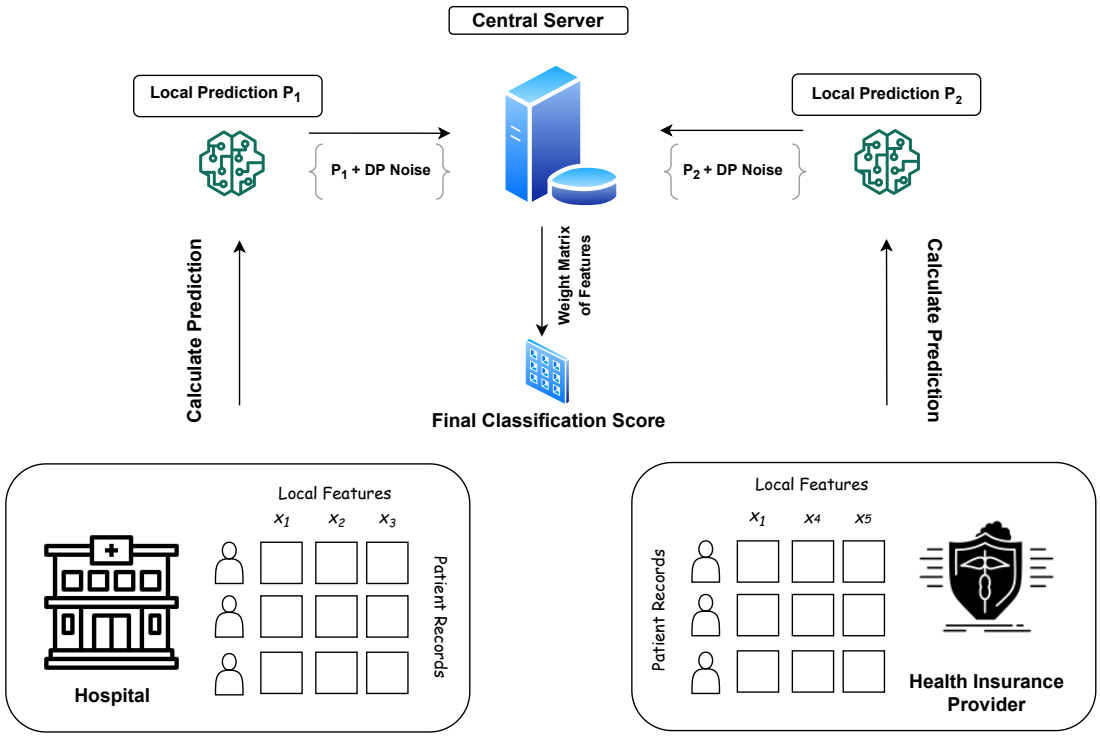


Figure 5.1: Multiple Data Owners are training a model collaboratively using distributed feature learning

the error between the predicted outputs and the true labels. Our objective is to build a model that generalizes well to unseen data and can be deployed in a real-world setting to solve the healthcare problem.

5.2.2 Local Prediction Calculation

In our model, each data source or site contains a unique feature set, therefore, we train our model based on the available local features only in this step. Hu et al. [33] use LR and CNN approaches in mobile app datasets for similar settings. We use both logistic regression and Long Short-Term Memory (LSTM) models for local feature-based training to improve the utility using health care data. Logistic regression is a simple, yet powerful linear model that is often used for classification tasks. It works by using a linear combination of the input features to predict the probability of a

given example belonging to a particular class. On the other hand, LSTM is a type of recurrent neural network (RNN) that is particularly well-suited for modelling sequential data, such as time series or natural language. LSTMs can capture long-term dependencies in data by using gating mechanisms to control the flow of information through the network.

To use logistic regression for local feature-based training, we define a set of local features, compute the local feature representation for each feature vector, and then use these local features as input to train a logistic regression 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.

On the other hand, to use LSTM for local feature-based training, we follow a similar process, however, we need to reformat the local feature representation as a sequence that can be processed by the LSTM. For the same set of local features $f_1, f_2, \ldots, f_m$ and for each feature vector x_i, we create a sequence of length by concatenating the local features. Then, we use this sequence as input to an LSTM model following the equations described in the *Background* section.

To simulate our methodology in the healthcare industry, we train the model with a sensitive *MIMIC-III* dataset. Before beginning training, we divide the whole dataset into two independent sets with the same records but different attributes, as shown in Fig. 5.1. We *cut* the full dataset

into two separate sets with the same records but different features to mimic the VFL scenario. The model training is then run in two distinct clients to compute the local prediction result for both LR and LSTM. In this scenario, we can assume one party is Hospital and another party is a local health insurance provider. After proper preprocessing of the data, For the same set of records, two parties could have had different features. As an example in Fig. 5.1, a hospital has features x_1, X_2, X_3 and a health insurance provider has features x_1, X_4, X_5, therefore only one feature x_1 as the unique one. For benchmarking, we use the tasks defined by Harutyunyan et al. [49] as below:

In-hospital Mortality/IHM

This is a binary classification problem that predicts in-hospital mortality based on the first 48 hours of an ICU stay. Our LSTM model takes as input a sequence of vital signs and other patient data collected over time and predicts the probability of death within the hospital.

Forecasting length of stay/LOS

The benchmark is defined as a multiclass classification issue prediction of bucketed remaining ICU stay. The remaining ICU stay duration is divided into ten classes/buckets ($< 1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-13, > 14$). This is only done for individuals who did not die in the intensive care unit.

phenotype classification/PH

Phenotyping is a collection of 25 discrete binary classification tasks that determine which of 25 acute care conditions exist in a specific patient's ICU stay record.

Decompensation/DC

The goal is to identify patients who are physically deteriorating. Decompensation is characterised as a sequential prediction job in which the model must forecast 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 within the local feature set, with time series observations we capture patient's length of ICU stay of *T* hours, with x_t at each time step t (1-hour interval). To model the time series observations, we define the feature as $[x_t]_{t=1}^{T}$ where previous hidden state $a^{<t-1>}$ and current hidden state $a^{<t>}$. According to the LSTM equation mentioned in *Background* section, the model equation for all the above fours task will be as follows:

$$\hat{IHM} = \sigma(W_o[a^{<48>}, x^{<48>}] + b_o) \tag{5.1}$$

$$\hat{LOS} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{5.2}$$

$$\hat{DC} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{5.3}$$

$$\hat{PH} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{5.4}$$

Where o is a binary label at $t = 48$ hours for in-hospital mortality, and $t = 5 \ldots T$ for the other three tasks. It is worth noting that in-hospital death is predicted at the end of 48 hours, whereas decompensation, phenotyping, and LOS tasks are anticipated at each time step after the first four hours of ICU stay, as benchmarked in [49]. We trained our LSTM model using the Adam optimizer with a learning rate of D and a batch size of E. We used a categorical cross-entropy loss function and evaluated the model's performance on the validation set at the end of each epoch. We trained the model for a maximum of 20 epochs and selected the best model based on the validation loss.

Similarly, along with the *MIMIC-III* dataset, we performed the training on the *Adult* dataset to benchmark and compare the performance. In this dataset, the dependent variable for the ML classification training is the *Income* attribute, which has values of either > 50k or $<= 50$k. We split the full dataset into two separate sets with the same records but different features to mimic the VFL scenario. Then we performed the same model training as described above for the *MIMIC-III* dataset. We also compared the performance of our LSTM model to a baseline logistic regression model and found that the LSTM model outperformed the logistic regression model for both datasets, which will be described in the *Experiment* section.

5.2.3 Feature Weight Mechanism

In machine learning, feature weights are values that are learned during the training process of a model to predict the target variable. These weights indicate the importance of each feature, or input, in the model's prediction. For example, if a model is trained to predict the mortality rate of patients in the ICU based on length of stay, drug history, and age, the feature weights could indicate that the length of stay is the most important factor, followed by the age, and then the drug history. In vertical federated learning, a weighted feature algorithm is used to determine the importance of each feature in the final prediction. Each party locally trains a logistic regression model on their own set of features and labels by randomly shuffling them. The coefficients of each feature in the logistic regression model are calculated and sent to the central server. The central server combines the coefficients received from each party and uses them to train a new logistic regression model on the complete set of features. The coefficients obtained from the trained logistic regression model are used as the feature weights. By using the feature weights, the central server can aggregate the local predictions without needing to share the raw feature data among the parties, thereby preserving privacy.

Finally, the weight matrix is built as shown in the example in Table 5.2. For calculating the

mortality of hospital patients, the length of stay has a greater effect on the classification than *Age* or *Disease Code*. That is why *In Time* and *Out Time* have a weight of 85%, while *Age* and *Disease Code* have weights of 25% and 40%, respectively. On the other hand, for phenotyping acute diseases, staying in the hospital has very little effect, whereas *Disease Code* has more effect. That is why, in this scenario, *In Time* and *Out Time* have a weight of 15%, while *Age* and *Disease Code* have weights of more than 85

5.2.4 Aggregator Server Execution

The LR and LSTM local predictions are calculated at the local locations and then shared with the aggregator server for the final aggregation. After each client has shared its results, the aggregator server applies the weight matrix to calculate the final output, as explained in Algorithm 3. The score is then aggregated, as illustrated in Table 5.3.

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

$$\mathcal{Y} = \delta \left(\sum_{p=1}^{P} w^p y(x^p) \right) \tag{5.5}$$

While only the local predictions $y(x^p)$ are shared to produce the final prediction with the weight matrix w^p for each party. Here, $\delta : \mathbb{R} \to \mathbb{R}$ is a continuously differentiable function used to aggregate local predictions weighted by w^p.

Note that the server does not perform any ML classifications; rather, it only calculates the aggregation based on the provided local predictions and weight matrix.

For example, for the LR training, the local score from site 1 and site 2 is 0.67 and 0.75, respectively, as shown in Table 5.3. Now, after applying the weight matrix from Table 5.2, multiplication and aggregation are performed, and finally, the score is returned as 0.84, which is above the value of each local party as well as below the value of the training that could be performed in a central

architecture. This is due to the noise addition and weight multiplication in the calculation process.

Algorithm 3: Algorithm for Aggregator Server

Input : Total number of data parties P, Local Predictions Y, Weight Matrix W

Output: Updated Score

Initialize Score $\mathcal{Y}$;

`// for each round of training`

for $p \in P$ **do**

  $\mathcal{Y} \leftarrow (w^p y(x^p))$;

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

return $\mathcal{Y}$

5.2.5 Privacy Mechanism

One of the most important concerns in our design is maintaining the privacy of local data. Only the local prediction results are transferred, which are formed from the local data and features. As a result, if these data are exchanged in raw format, the original functionality may be revealed to curious servers or other parties. By perturbing the local predictions to be uploaded, we apply differential privacy algorithm to better protect the feature data at each party from hostile servers and parties. Differential privacy [6, 27, 50] is a generally accepted norm for ensuring the privacy of algorithms that works with aggregated data. To ensure the anonymity of all feature characteristics, we add laplacian noise to the local prediction result $y(x^p)$ at party p. Therefore, the shared prediction with DP noise becomes $y(x^p) + \varepsilon$ to be aligned with privacy.

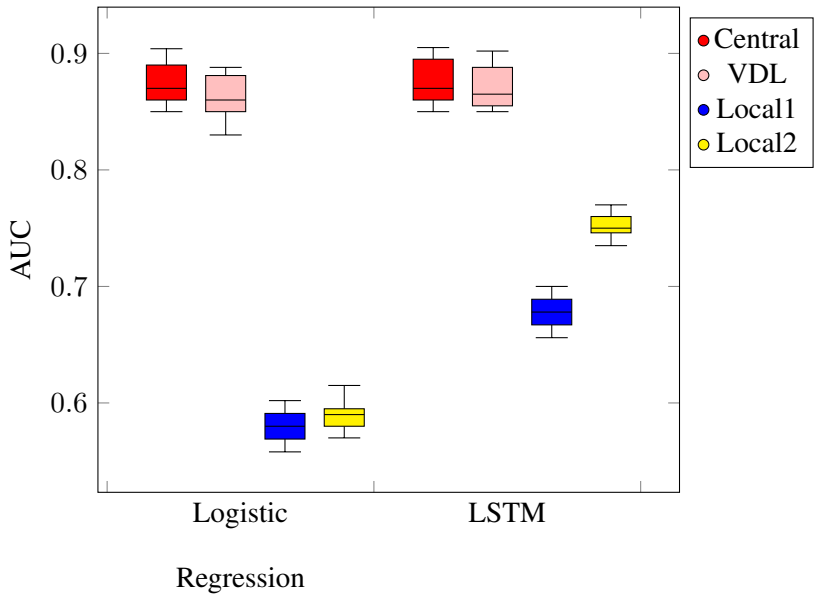


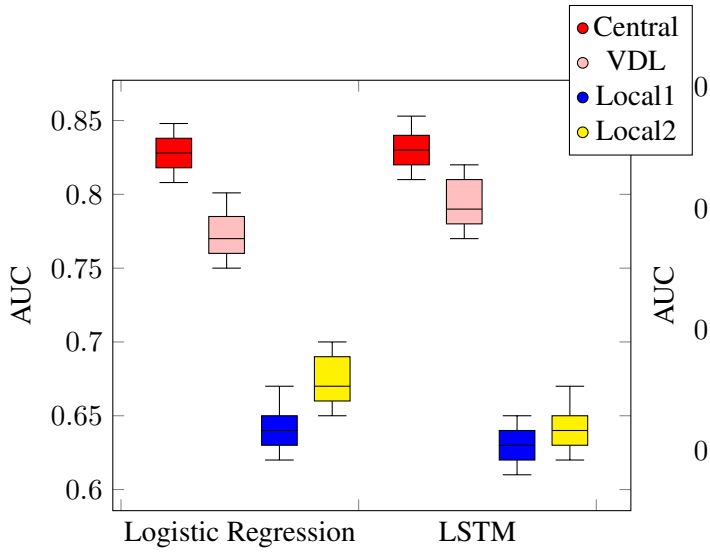
Figure 5.2: AUC for Adult Dataset with Privacy Budget ε : 1 to 5

5.3 Results

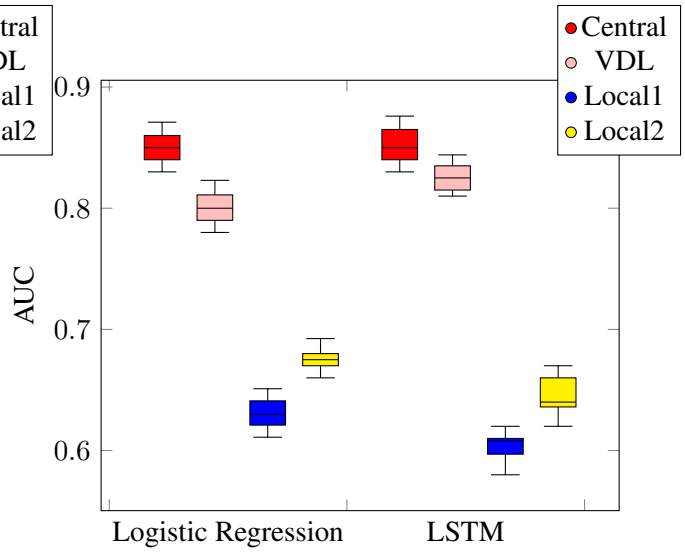
This section describes the experimental results. To benchmark and compare the results, we tested with *MIMIC-III* and *Adult* datasets in various settings. We specified a fixed privacy budget of DP algorithm as ε: 1-5 for each instance of the experiment. To perform the study, we employed multiple computers in our lab to imitate multi-client environments.

5.3.1 Experimental Setup

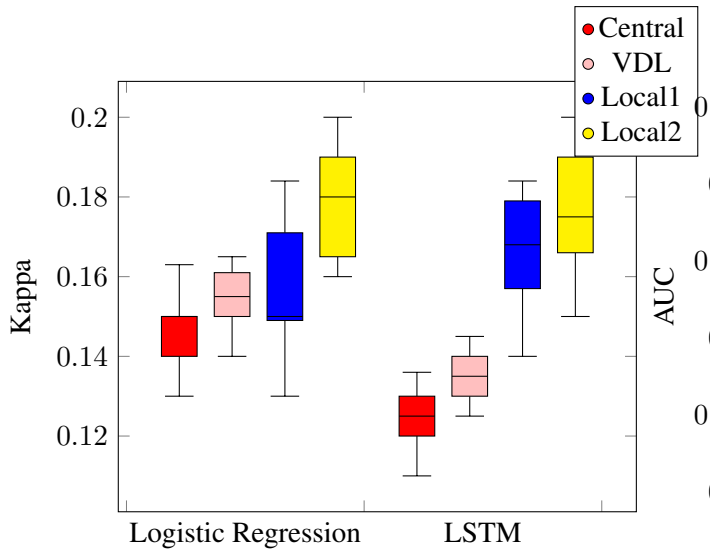
For our tests, we first used the Medical Information Mart for Intensive Care (MIMIC III) dataset and followed Harutyunyan et al. [49] benchmark configuration for processing time series signals from ICU devices. We used the same test set as in the benchmark and 15% of the remaining data as the validation set. We performed all four benchmarking tasks as discussed in the methodology section. Only patients hospitalized in the ICU for at least 48 hours are considered for the in-hospital mortality task. However, we removed all clinical notes that did not have a chart time linked with



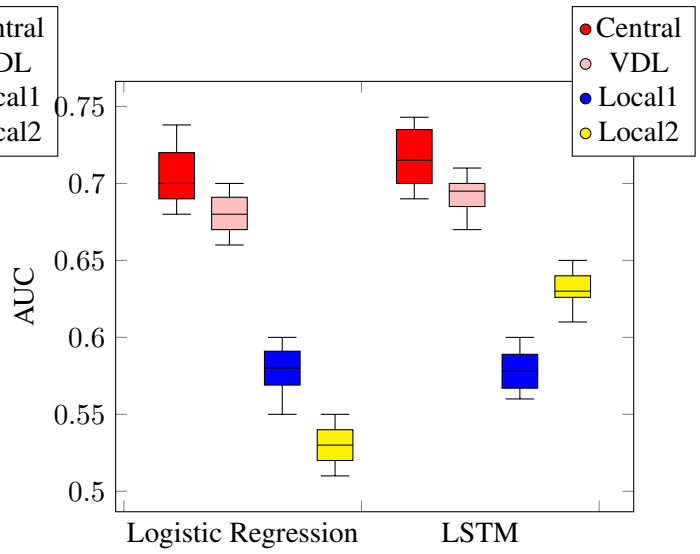
(a) AUC for MIMIC-III Dataset (In Hospital Mortality)



(b) AUC for MIMIC-III Dataset (Decompensation)



(c) Cohen's (Kappa) for MIMIC-III Dataset (Length of Stay Forecasting)



(d) AUC for MIMIC-III Dataset (Phenotyping)

Figure 5.3: AUC for MIMIC-III Dataset

them, as well as any patients who did not have any notes. We employed 64 hidden units in LSTM for decompensation and LOS prediction. We used a mix of 25 distinct binary classification tasks based on ICU phenotypic data for phenotyping classification.

We also tested on another public dataset, Adult, a traditional census dataset in which the prediction objective is to assess whether a person earns more than $50,000 per year. There are 48,842 samples with 124 characteristics apiece.

We ran the experiment with LR and LSTM in four distinct situations for these two datasets. Our test arrangement included a central server and two clients. For benchmarking, we first examined the prediction in a central architecture, where entire datasets with all attributes are available in a single location. The forecast was then calculated independently by the local clients for two of them. Finally, as the VDL architecture, we simulated distributed learning with two clients and a central server. We used the Linux *Cut* command to split the data columns-wise for the identical set of records in order to separate the whole dataset by feature.

5.3.2 Evaluation

We used the Area Under Precision-Recall (AUC) measure for in-hospital mortality, phenotyping, and decompensation tasks in the MIMIC dataset. To assess LOS, we used Cohen's linear weighted kappa, which evaluates the correlation between anticipated and real multi-class buckets, as proposed by Harutyunyan et al. [49]. In each chart, we box-plotted the comparison for both LR and LSTM, with red, pink, blue, and yellow boxes denoting non-private central, vertical distributed learning (VDL), and local prediction for parties 1 and 2, respectively. Figs. 5.3a, 5.3b, and 5.3d show the AUC values with the central scenario, vertically distributed learning scenario with privacy, and local scenario for two parties for the three tasks of the MIMIC dataset. It is observed that the central design performs best in all circumstances since it is the benchmarking reference with a full dataset accessible with all characteristics and no privacy. However, compared to the central design, Local

1 or Local 2 have relatively low scores. This is due to the training being conducted on only the available local features rather than categorizing using all available features.

5.4 Conclusion

We proposed a distributed machine learning framework while guaranteeing privacy on vertically segregated data. In our method, each client uses LR and LSTM neural networks to make local predictions based solely on local features. Then, to offer an extra layer of privacy, a certain amount of noise is added to the prediction results using the DP algorithm. In addition, the weighted feature function, which is computed based on local feature sets, is applied to the final prediction. The central server then receives the perturbed scores along with a proper weight to calculate the final prediction. No raw data, features or model parameters are shared in any phase of the training. In this way, our method mitigates the exposure of sensitive records in any form maintaining an effective utility with the localization of the data. To demonstrate the application of our system in the healthcare sector, we apply our technique to publicly available health data: *MIMIC-III* and *Adult* data from US-Census. Comparison with other health data using a more effective weighted feature can be the future direction of this research.

Chapter 6

Conclusion

In this study, we present three approaches for conducting a secure analysis of sensitive healthcare data that is distributed among various participants. These frameworks are designed to perform ML classification in both horizontally and vertically partitioned data with an adequate level of utility while maintaining privacy.

The first two frameworks employ horizontal federated learning with differential privacy to strike a balance between privacy and utility. In contrast, the third approach is based on vertical distributed learning and is applied to real-life health data to establish a secure healthcare application.

6.1 Summary

The first approach involves reducing the whole data dimension through feature selection in horizontal federated learning, which minimizes the noise added by differential privacy while maintaining efficient scores. This model provides efficient data utility with a moderated privacy budget.

The second approach involves generalizing the full dataset by records and applying differential privacy techniques to summarized count-based statistical data in horizontal federated learning. By

reducing noise in every level of data, this model offers efficient data utility with a moderated privacy budget.

The third approach is a privacy-preserving distributed ML technique for vertically partitioned data. This approach uses a generalized weighted feature algorithm in the server to aggregate local clients' results, avoiding the need to share feature information and offering a secure healthcare framework that preserves data privacy and utility.

In conclusion, the main contribution of this thesis is the secure, efficient, and privacy-preserving federated machine learning on healthcare data considering different data distributions.

6.2 Future Work

There are several potential research directions in federated learning with different privacy algorithms to balance the privacy-utility trade-off. Some of the possibilities include:

- Enhancing the effectiveness and expandability of federated learning with differential privacy. It could entail creating innovative methods for communication and distributed optimization that are tailored to the limitations brought about by differential privacy.

- Developing new privacy-preserving techniques for federated learning. This could include exploring new forms of differential privacy, such as zeroth-order [51] or R'enyi differential privacy [52], or developing new methods for composing different forms of privacy to achieve stronger guarantees.

- Examining the compromises between privacy, utility, and stability in federated learning. This may involve exploring the impact of various privacy techniques on the precision and broad applicability of federated models, as well as examining their resistance to malicious attacks.

Bibliography

[1] Tao Zheng, Wei Xie, Liling Xu, Xiaoying He, Ya Zhang, Mingrong You, Gong Yang, and You Chen. A machine learning-based framework to identify type 2 diabetes through electronic health records. *International Journal of Medical Informatics*, 97:120–127, January 2017.

[2] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. Differential privacy-enabled federated learning for sensitive health data, 2020.

[3] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 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, pages 1322–1333, New York, NY, USA, October 2015. Association for Computing Machinery.

[4] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, May 2017. ISSN: 2375-1207.

[5] Anand D. Sarwate and Kamalika Chaudhuri. Signal Processing and Machine Learning with Differential Privacy: Algorithms and Challenges for Continuous Data. *IEEE Signal Processing Magazine*, 30(5):86–94, September 2013.

[6] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, STOC '09, pages 371–380, New York, NY, USA, May 2009. Association for Computing Machinery.

[7] Tanzir Ul Islam, Reza Ghasemi, and Noman Mohammed. Privacy-preserving federated learning model for healthcare data. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0281–0287. IEEE, 2022.

[8] Tanzir Ul Islam, Noman Mohammed, and Dima Alhadidi. Private federated framework for health data. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1919–1926. IEEE, 2022.

[9] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv:1610.02527 [cs]*, October 2016. arXiv: 1610.02527.

[10] Chris M Bishop. Neural networks and their applications. *Review of scientific instruments*, 65(6):1803–1832, 1994.

[11] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018.

[12] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[13] Understanding LSTM Networks – colah's blog.

[14] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[15] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE*

INFOCOM 2019-IEEE Conference on Computer Communications, pages 2512–2520. IEEE, 2019.

[16] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2019.

[17] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[18] Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2430–2444, 2020.

[19] Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Artificial Intelligence and Statistics*, pages 933–941. PMLR, 2012.

[20] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, and Issa Sylla. Anonymizing data for preserving privacy during use for federated machine learning, November 30 2021. US Patent 11,188,791.

[21] Hiroaki Kikuchi, Chika Hamanaga, Hideo Yasunaga, Hiroki Matsui, Hideki Hashimoto, and Chun-I Fan. Privacy-preserving multiple linear regression of vertically partitioned real medical datasets. *Journal of Information Processing*, 26:638–647, 2018.

[22] Haokun Fang and Quan Qian. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4):94, 2021.

[23] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506, 2020.

[24] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[25] Huafei Zhu, Rick Siow Mong Goh, and Wee-Keong Ng. Privacy-preserving weighted federated learning within the secret sharing framework. *IEEE Access*, 8:198275–198284, 2020.

[26] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 253–261, 2020.

[27] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[29] Hyunghoon Cho, David J Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature Biotechnology*, 36(6):547–551, July 2018.

[30] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A Communication Efficient Collaborative Learning Framework for Distributed Features, July 2020. arXiv:1912.11187 [cs, stat].

[31] Yaochen Hu, Peng Liu, Linglong Kong, and Di Niu. Learning Privately over Distributed Features: An ADMM Sharing Approach, July 2019. arXiv:1907.07735 [cs, stat].

[32] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. VAFL: a Method of Vertical Asynchronous Federated Learning, July 2020. arXiv:2007.06081 [cs, math, stat].

[33] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. 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, pages 2232–2240, New York, NY, USA, July 2019. Association for Computing Machinery.

[34] Olivia Choudhury, Yoonyoung Park, Theodoros Salonidis, Aris Gkoulalas-Divanis, Issa Sylla, and Amar K. Das. Predicting Adverse Drug Reactions on Distributed Health Data using Federated Learning. *AMIA Annu Symp Proc*, 2019:313–322, 2019.

[35] IDASH PRIVACY & SECURITY WORKSHOP 2021. Idash privacy & security workshop 2021, 2021.

[36] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, pages 543–554, 2007.

[37] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The Composition Theorem for Differential Privacy. *arXiv:1311.0776 [cs, math]*, December 2015. arXiv: 1311.0776.

[38] Ahsan Huda, Adam Castaño, Anindita Niyogi, Jennifer Schumacher, Michelle Stewart, Marianna Bruno, Mo Hu, Faraz S Ahmad, Rahul C Deo, and Sanv J Shah. A machine learning model for identifying patients at risk for wild-type transthyretin amyloid cardiomyopathy. *Nature communications*, 12(1):1–12, 2021.

[39] Noman Mohammed, Rui Chen, Benjamin C.M. Fung, and Philip S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international confer-*

ence on Knowledge discovery and data mining - KDD '11, page 493, San Diego, California, USA, 2011. ACM Press.

[40] Latanya Sweeney. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, October 2002.

[41] Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[42] Noman Mohammed, Xiaoqian Jiang, Rui Chen, Benjamin C M Fung, and Lucila Ohno-Machado. Privacy-preserving heterogeneous health data sharing. *Journal of the American Medical Informatics Association*, 20(3):462–469, May 2013.

[43] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[44] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):1–18, 2019.

[45] Ameer Sardar Kwekha-Rashid, Heamn N Abduljabbar, and Bilal Alhayani. Coronavirus disease (covid-19) cases analysis using machine-learning applications. *Applied Nanoscience*, pages 1–13, 2021.

[46] Jie Xu, Benjamin S. Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated Learning for Healthcare Informatics. *J Healthc Inform Res*, 5(1):1–19, March 2021.

[47] Akhil Vaid, Suraj K Jaladanki, Jie Xu, Shelly Teng, Arvind Kumar, Samuel Lee, Sulaiman Somani, Ishan Paranjpe, Jessica K De Freitas, Tingyi Wanyan, Kipp W Johnson, Mesude Bicak,

Eyal Klang, Young Joon Kwon, Anthony Costa, Shan Zhao, Riccardo Miotto, Alexander W Charney, Erwin Böttinger, Zahi A Fayad, Girish N Nadkarni, Fei Wang, and Benjamin S Glicksberg. Federated Learning of Electronic Health Records to Improve Mortality Prediction in Hospitalized Patients With COVID-19: Machine Learning Approach. *JMIR Med Inform*, 9(1):e24207, January 2021.

[48] Ligeng Zhu and Song Han. Deep Leakage from Gradients. In Qiang Yang, Lixin Fan, and Han Yu, editors, *Federated Learning*, volume 12500, pages 17–31. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Computer Science.

[49] Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, December 2019.

[50] Zhanglong , Xiaoqian ang, Shuang Wang, Li Xiong, and Lucila Ohno-Machado. Differentially private distributed logistic regression using private and public data. *BMC medical genomics*, 7(S1):S14, 2014.

[51] Cristiano Gratton, Naveen KD Venkategowda, Reza Arablouei, and Stefan Werner. Privacy-preserving distributed zeroth-order optimization. *arXiv preprint arXiv:2008.13468*, 2020.

[52] Ilya Mironov, Kunal Talwar, and Li Zhang. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.