The Application of Privacy Preserving Machine Learning Models Using Federated Learning and Differential Privacy on Healthcare Data

Comp 430 Term Project

GROUP MEMBERS: Atalay Görgün, Çınar Eren Arslan, Duru Tandoğan, Yamaç Ömür

I. Introduction

In this project, we have implemented several privatized models using a variety of Python libraries to operate on a healthcare dataset. We aimed to use decentralized training of ML models across hospitals and provide privacy by adding noise to the inputs during the model training using the Opacus library's [4] differentially private stochastic gradient descent (DP-SGD) approach. By this approach, the patient data of each hospital remains local and secure as no raw data leaves the premises, and only perturbed model parameters are shared for aggregation. Through our approach and generated model, we have achieved a secure analysis framework with high-accuracy predictions for early disease detection.

Stroke is one of the most important leading causes of long-term disability and mortality. According to the World Health Organization (WHO), stroke is responsible for approximately 11% of total deaths, [2]. With early detection and prediction, accurate medical treatment can be applied to prevent death and fast recovery with better outcomes.

With the correct Machine Learning (ML) model, we can predict and detect such diseases faster and take action early. A crucial part of training an ML model with health datasets is preserving privacy. From the health data that we use to train our model, adversaries can learn/infer sensitive information of the patients and their relatives [1]. Thus we need to secure the privacy of the information that we have in our dataset.

We have assessed the performance of Differential Privacy (DP) on a linear regression and a neural network setting, and examined Federated Learning (FL) on a neural network setting, as most research on FL is done on this model type. The study conducted by [1], trained a machine learning model using the heart failure dataset and reached an accuracy between %79 and %98 with their method. In our project, we used a dataset that will allow us to predict patients who are likely to get strokes [2]. While working on our dataset, we observed that the stroke instances were underrepresented and have a distribution of 4% over the entire dataset. To balance out our dataset and overcome the problem of underrepresentation, we have applied the Synthetic Minority Oversampling Technique (SMOTE) to the minority class. With our implementation, we have observed the tradeoff between privacy and utility which will be discussed in the following parts of our report.

II. Background and Implementation

A. Data Preparation

The dataset we used in this project, healthcare-dataset-stroke-data.csv, contains several features relevant to predicting the likelihood of stroke, such as age, glucose level, and body mass index (BMI). Our target variable is a binary class which holds 0 to indicate no stroke and 1 to indicate stroke. The dataset includes a mix of numerical and categorical features. We have preprocessed the categorical features using one-hot encoding to make them suitable for training ML models. For some entries in the dataset, some features such as the BMI are unknown, and thus an initial data preprocessing step was required. Data cleaning steps include the removal of missing values and redundant columns, such as the unique ID column, to ensure a clean and efficient dataset.

While we were investigating our dataset, we observed an imbalance. The stroke instances were underrepresented with a distribution of 4% throughout the dataset. We have researched and found an oversampling technique called Synthetic Minority Oversampling Technique (SMOTE) and applied it to our minority class in the dataset to address the imbalance and to improve our model's performance. After applying SMOTE, we split the dataset into training and testing sets with the ratio 80%-20%, ensuring an appropriate division for model training and evaluation.

As we want our model to mimic a real-world scenario in healthcare and implement it using FL, we wanted to divide the original dataset into multiple subsets, each representing a local hospital's dataset. To do so, we have split the original dataset into three subsets. Each subset has a different ratio of positive labelled entries to negative labelled entries. These subsets are designed to reflect and simulate the heterogeneous real-world medical data distribution across different institutions.

We were able to assess the performance of two methods in our project:

B. Federated Learning (FL)

Federated learning is the concept of training decentralized machine learning models and combining them in an aggregate server without sharing the raw data that the decentralized models were trained on. This approach will be very useful in our aforementioned hospitals scenario, where protecting the privacy of their patients is of utmost importance. In brief, we simulated the training of three different local machine

learning models, where each hospital trained their own stroke prediction model with their own data, without sharing their raw data with the central server. Subsequently, the three separate models were combined in a simulated "central server" [1]. A common method to combine the local models is by simply averaging their model parameters, known as FedAvg [3]. Thus, the most common implementations of federated learning are through parametric methods, such as neural networks.

C. Differential Privacy (DP)

As discussed in the course lectures, differential privacy protects the integrity of the data by using the notion that there should be little difference between the application results of neighboring datasets, in our case, meaning that adding or removing a patient's record should not change the results of the ML model significantly. For the context of ML models, the most common ways of adding differential privacy is through the input data or the output labels. In our case, since we used the Opacus [4] library, the noise was added during the training step during the calculation of the model parameter gradients.

D. Model Architecture

Before DP integration, the preprocessed dataset is first tested with a simple logistic regression (LR) model from SciKit. The model was trained and showed promising evaluation results, such as 87% accuracy and 88% F1 score.

The next goal was to test the preprocessed dataset with DP integrated. This implementation of the ML model began with LR again. For this case, PyTorch's torch.nn.Linear layer was utilized, which serves as a fully connected linear layer. A sigmoid activation function was applied to the output of the linear layer to transform predictions into probabilities for the binary classification task. While the LR model offered a straightforward starting point, it provided valuable insights into the dataset's baseline performance, enabling comparisons with more complex models.

Subsequently, for the application of federated learning, we first trained three separate local neural network models on the three now-split datasets, with the three individual models achieving satisfactory results. Finally, we used Python's Flower [5] library in order to implement federated learning, which trains the three models in a decentralized manner and then aggregates them into one final model using Federated Averaging.

E. Differential Privacy Integration

To integrate DP into the LR model, Opacus' PrivacyEngine class was utilized. Opacus' make_private_with_epsilon method was used for applying DP on the LR with a given privacy budget (ε) and then training it. We evaluated the LR model trained with DP on different privacy budgets, ranging from ε=0.1 to ε=0.9. These privacy budgets are passed as arguments to field target_epsilon. Other fields such as target_delta were kept constant with appropriate values. Results showed that privacy budget did not significantly affect the performance of a single LR model, with metrics accuracy, precision, recall, and F1 score all being 70% on average. These results were expected compared to higher scores with no DP application, as there was a natural utility loss after DP was applied.

F. Federated Learning Integration

To implement FL on the three subsets of the dataset with DP integrated, Flower Framework's [5] FlowerClient class was utilized. Three FL clients were configured, each responsible for training a previously obtained local LR model with DP. These clients represented three hospitals, with each hospital data being processed and converted into PyTorch tensors. The training procedure of local LR models was the same as the previous part, except it was run on a FL client. The FL server was configured to train these models and merge them with FedAvg for three rounds. For each of the three rounds, local models were trained separately on their respective datasets. After that, the FL server merged the models using FedAvg to produce a global model that incorporated updates from all clients and the process was simulated on a single server using the run_simulation method. The results of the simulation and performance metrics were obtained with expected and promising results.

III. Results

A. Evaluation Metrics

For our model evaluation we used accuracy, precision, recall, F1 score metrics. These metrics are evaluated as follows:

$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$	$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$
$Recall = \frac{TP}{TP+FN}$	$Precision = \frac{TP}{TP+FP}$

In the above table, TP denotes True Positives, FP denotes False Positives, FN denotes False Negatives, and TN denotes True Negatives.

These metrics provide detailed information about the performance of the classification model, and for our dataset which has class imbalance these evaluations are important. *Accuracy* gives the ratio of correctly classified samples out of the total samples, we used accuracy_score function from sklearn.metrics for the computation. *Precision* gives the ratio of correctly predicted positives out of all positive predictions, we used precision_score from sklearn.metrics for the computation. *Recall* gives the ratio of correctly predicted positives out of all actual positives, we used recall_score from sklearn.metrics for the computation. *F1 Score* gives the harmonic average of precision and recall, balances the trade-off between these two metrics, we used f1_score from sklearn.metrics for the computation. The metrics were calculated assuming "1" or "stroke" to be the positive class, and "0" or "no stroke" to be the negative class.

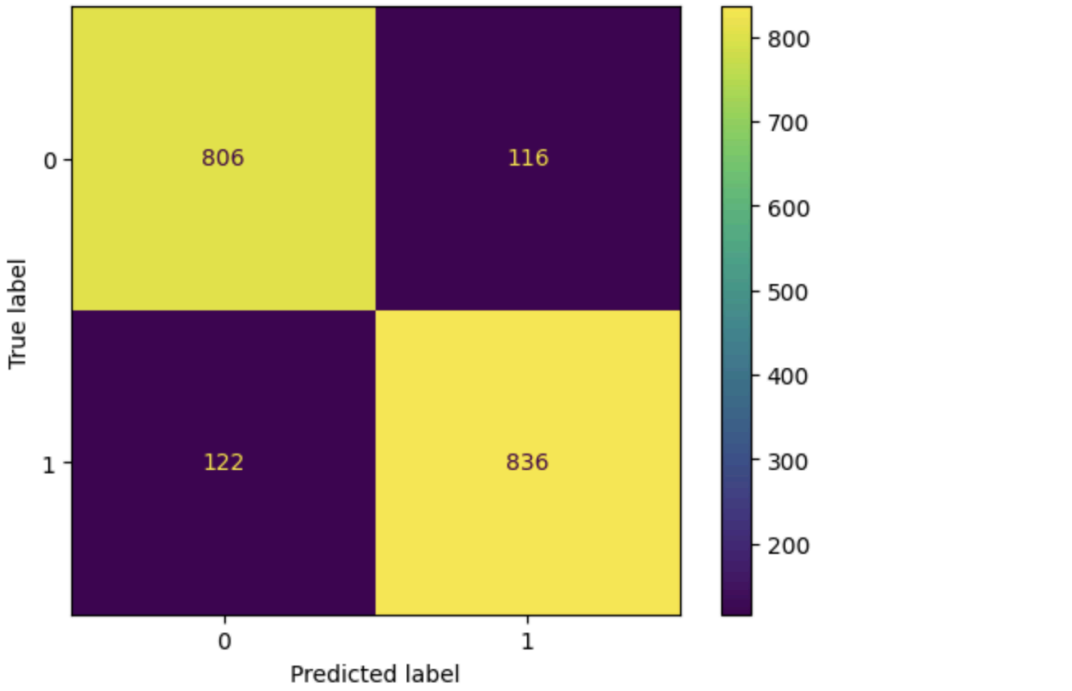
We used these metrics to evaluate model accuracy in four steps: first, after the logistic regression model was trained with no Differential Privacy (DP) and no Federated Learning (FL); second, after applying Differential Privacy; third, after separating the dataset into three simulated hospitals and creating three separate neural networks, and finally after implementing FL.

We present our results as follows:

B. Results

Step 1: Logistic Regression

```
Accuracy:  0.8734042553191489
Recall:  0.872651356993737
Precision:  0.8781512605042017
F1 Score:  0.875392670157068
Out[9]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7e0d8faa4710>
```



Confusion Matrix (0 = No Stroke, 1 = Stroke)

The results of the simple logistic regression model show that it is possible to apply machine learning models on health care applications such as stroke prediction, achieving an approximately 87% accuracy, recall, precision, and F1 score. We would also like to note that without oversampling, the model was mostly classifying every data point as "non-stroke", indicating a heavy bias towards the negative class. After SMOTE, the model began predicting positives as well.

Step 2: Logistic Regression with DP

ε	Accuracy	Precision	Recall	F1 Score
0.09	0.6851	0.7007	0.6670	0.6834
0.20	0.6947	0.7069	0.6848	0.6957
0.30	0.7096	0.7556	0.6357	0.6905
0.39	0.6941	0.7477	0.6033	0.6678
0.49	0.7101	0.7034	0.7453	0.7238
0.60	0.7128	0.7370	0.6785	0.7065
0.70	0.7117	0.7476	0.6555	0.6986
0.80	0.6910	0.7869	0.5397	0.6402
0.90	0.7085	0.7248	0.6900	0.7070

Table 1: Privacy Budget (ε) vs. Accuracy, Precision, Recall, and F1 Score

Table 1 shows our results for the logistic regression model with differential privacy case in PyTorch over different privacy budgets, ranging from 0.09 to 0.90. The results show that the overall metrics have decreased compared to the basic logistic regression model with zero DP, as expected. Furthermore, although the metrics seem to be non-monotonic with respect to the privacy budget ε, we observed an overall increase in accuracy as ε increased. This phenomenon was expected, since low ε corresponds to more privacy, resulting in more noise being added to the gradients.

Step 3: 3 Separate Neural Networks

Model1

ε	Loss	Accuracy	Precision	Recall	F1 Score
Epoch=1: 0.04	**1.977095**	**80.00%**	**0.64**	**0.80**	**0.71**
Epoch=2: 0.06	**1.876979**	**80.00%**	**0.64**	**0.80**	**0.71**
Epoch=3: 0.07	**1.986079**	**80.00%**	**0.64**	**0.80**	**0.71**
Epoch=4: 0.08	**2.113304**	**80.00%**	**0.64**	**0.80**	**0.71**
Epoch=5: 0.09	**1.891421**	**80.00%**	**0.64**	**0.80**	**0.71**

Table 2: Privacy budget (ε) vs. Loss, Accuracy, Precision, Recall, and F1 Score for the first hospital.

Model2

ε	Loss	Accuracy	Precision	Recall	F1 Score
Epoch=1: 0.05	**2.016454**	**85.00%**	**0.72**	**0.85**	**0.78**
Epoch=2: 0.06	**2.055433**	**85.00%**	**0.72**	**0.85**	**0.78**
Epoch=3: 0.07	**1.917398**	**85.00%**	**0.72**	**0.85**	**0.78**
Epoch=4: 0.09	**1.782484**	**85.00%**	**0.72**	**0.85**	**0.78**
Epoch=5: 0.10	**1.783064**	**85.00%**	**0.72**	**0.85**	**0.78**

Table 3: Privacy budget (ε) vs. Loss, Accuracy, Precision, Recall, and F1 Score for the second hospital.

Model3

ε	Loss	Accuracy	Precision	Recall	F1 Score
Epoch=1: 0.05	**1.165600**	**90.00%**	**0.81**	**0.90**	**0.85**
Epoch=2: 0.06	**1.081403**	**90.00%**	**0.81**	**0.90**	**0.85**
Epoch=3: 0.07	**1.231627**	**90.00%**	**0.81**	**0.90**	**0.85**
Epoch=4: 0.08	**1.153760**	**90.00%**	**0.81**	**0.90**	**0.85**
Epoch=5: 0.09	**1.259737**	**90.00%**	**0.81**	**0.90**	**0.85**

Table 4: Privacy budget (ε) vs. Loss, Accuracy, Precision, Recall, and F1 Score for the third hospital.

The above three tables show the performance of the three separate local hospital models with the target epsilon value set to 0.1 measured in 5 epochs. Since the initial dataset became significantly smaller as it was split into three, it is likely that it became very easy for the machine learning process to occur, achieving a satisfactory loss even with the first epoch. Since the training data was easier to learn, we did not observe much difference in the classification metrics as the privacy budget changed with different epochs. We also tried privacy budgets ranging from 0.1 to 0.9, but since their results were very similar, we decided to simply model the average loss as a graph for each model:

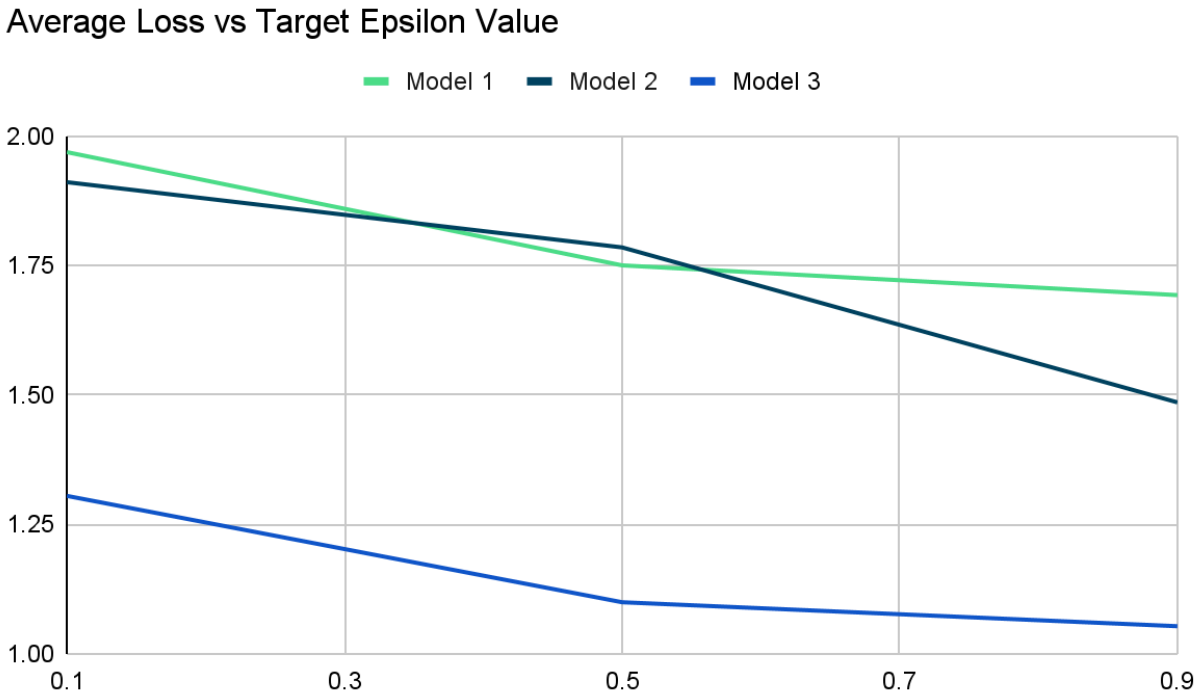


Figure 1: Privacy budget (ε) vs Average Loss across all models.

The line graph above demonstrates that the average loss generally drops as the privacy budget increases, which is expected since less noise is added to the model.

Step 4: Neural Network with FL (Results of the Aggregated Model Across 3 Rounds)

Round	Accuracy	Loss	Precision	Recall	F1 Score
1	0.9	0.2647	0.0	0.0	0.0
2	0.9216	0.2580	0.6734	0.4187	0.5164
3	0.9394	0.2068	0.7404	0.6062	0.6667

Table 5: The accuracy, loss, precision, recall and F1 Score results across three rounds of FL training.

Table 5 illustrates that federated learning can achieve an accuracy that is even higher than differential privacy.

Overall, our results demonstrate that differential privacy and federated learning are both very useful methods that can be applied to privatize machine learning models trained on health care data. Even with high privacy, the worst model, which was basic logistic regression with differential privacy, attained a result of at least 68% accuracy.

BIBLIOGRAPHY

[1] T. U. Islam, R. Ghasemi and N. Mohammed, "Privacy-Preserving Federated Learning Model for Healthcare Data," *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2022, pp. 0281-0287, doi: 10.1109/CCWC54503.2022.9720752.

[2] fedesoriano, "Stroke Prediction Dataset," *Kaggle*. [Online]. Accessed: Dec. 28, 2024. Available: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download

[3] Educative, "What is federated averaging (FedAvg)?," *Educative.io*. [Online]. Accessed: Dec. 28, 2024. Available: https://www.educative.io/answers/what-is-federated-averaging-fedavg.

[4] Meta, "Introducing Opacus: A high-speed library or training PyTorch models with differential privacy," *Meta Blog*, Aug. 31, 2020. [Online]. Available: https://ai.meta.com/blog/introducing-opacus-a-high-speed-library-for-training-pytorch-models-with-differential-privacy/

[5] "Flower," *PyPi*, [Online]. Available: https://pypi.org/project/flower/