Data Privacy in Machine Learning: Differential Privacy and Federated Learning

Author: Aida Martirosyan
American University of Armenia
BS in Data Science

Author: Anna Misakyan
American University of Armenia
BS in Data Science

Supervisor: Elen Vardanyan
American University of Armenia

***Abstract*—The capstone project aims to train machine learning (ML) models with differential privacy (DP) algorithms to provide data privacy. We focus on a scenario where central DP is applied on models with black-box access to ensure user-level privacy. We train both Neural Network and Logistic Regression models by applying DP with the Gaussian mechanism, DP-SGD algorithm, on a sensitive Census *Adult* dataset to ensure DP guarantees. The project aims to compare the results of manual implementations of DP with the existing DP libraries, as well as to maintain high accuracy parallel to privacy. Furthermore, we explore federated learning with differential privacy to preserve model training privacy across decentralized devices. The project's target audience includes companies and individuals that provide ML services as an API or provide models with trained parameters to third parties.**

***Keywords*—Data Privacy, Machine Learning, Differential Privacy, Federated Learning, DP-SGD, Privacy Preservation**

I. Introduction

Data privacy is an essential concept in the modern technological world that refers to the protection of personal information, mainly including sensitive data such as financial and medical records. In fields such as data science and machine learning, where data is at the core of every project, ensuring the privacy guarantees of the data of any individual is of utmost importance. Implementing data privacy algorithms such as differential privacy in machine learning is necessary to ensure privacy between businesses and users.

DP is a powerful algorithm that provides robust privacy guarantees, but it is not a standard algorithm. The implementation of DP can vary depending on the complexity of the model and dataset characteristics. There are various approaches to applying DP in ML projects. One common way is to add noise during model training. This technique ensures data privacy guarantees for the model and provides accurate predictions for users without significantly affecting the model's performance [1].

Considering the increasing number of machine learning applications over the past years and their vulnerabilities, the importance of data privacy is also rising, which brings up the reason for this project's implementation.

In this capstone project, we aim to protect individuals' sensitive data while training ML models by ensuring data privacy. The project suggests using a differential privacy algorithm, which adds measured noise to the training process in a way that individual instances of the dataset are uninterpretable while the output of a model or statistical analysis remains nearly unchanged. This strategy ensures that adding or removing a single data point has minimal effect on the overall outcome. In addition to the differential privacy algorithm, we apply federated learning, another tool for preserving data privacy. By applying these methods, we want to train a well-performing model while maintaining data confidentiality.

In the next sections of the report, we provide a detailed explanation of the steps we took to complete this project. After introducing the main goal of the capstone project, we continue with discussing the research results on the topic in the **Literature Review** section. We provide more details about the differential privacy algorithm under the **Differential Privacy** section, including the definitions, properties, mechanisms, application cases, and its usage in ML applications, more specifically, introducing DP-SGD algorithm. In the next section, **Threat Scenarios and Possible Attacks**, we provide information on threat scenarios and define an attack against the DP model. The next section is **Federated Learning**, where we explain how federated learning works. The **Models and Methods** section presents our dataset description, data preprocessing steps, implementations of Neural Network and Logistic Regression models with DP, and their results. It also includes federated learning with differential privacy and attack implementation descriptions. Lastly, **Conclusions** section gives a summary of the project and ideas for future work.

II. Literature Review

One common vulnerability that ML models have is that they leak information about training data. One famous case is when Carlini et al. [2] organized an extraction attack on the GPT-2 model and proved that large language models memorize their training data. The experiment showed that searching particular keywords can reveal an individual's personal data like name, email address, and phone number. The situation would be much worse if the model included highly sensitive data, such as medical data. The article underlines the importance of using differentially private techniques while training to ensure data privacy.

Garfinkel et al. [3] presented the challenges that statistical agencies, like the U.S. Census Bureau face, considering the fact that they are working with individuals' data, that needs protection. They usually publish reports and statistical analyses

based on Census data, which are vulnerable to database reconstruction attacks (DRAs). DRAs can cause privacy risks by recovering personal data from published statistical tables. The paper discusses the importance of adding noise to the published statistics with the help of DP. As an example, the article notes that the 2020 Census decided to approve differential privacy usage to adequately add noise to data for confidential statistical results [3].

Medical data analyses with machine learning algorithms promote drug discoveries and disease predictions, which are crucial for healthcare. To protect the privacy of medical data prior to publishing and mining it, Liu et al. [4] proposed the application of differential privacy.

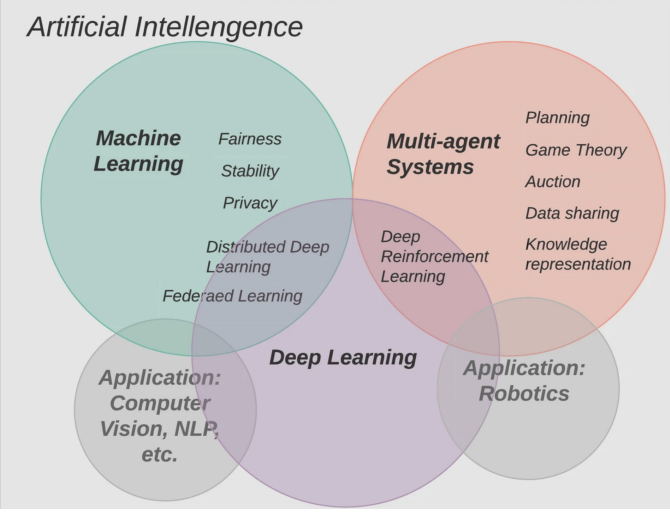


Fig. 1: Differential Privacy applications in different AI fields [5].

Zhu et al. [5] discussed the importance of differential privacy in different fields of Artificial Intelligence (AI), presented in Figure 1. The article underlines that differential privacy, besides guaranteeing data privacy, provides stability, security, fairness, and composition. Stability ensures that the model's outcome does not change when removing one entry from the data. Security and fairness create secure AI models that are not dependent on sensitive parameters such as gender. Finally, composition is another property of DP that ensures that in case of multiple operations on the same data, the privacy guarantees continue to hold.

Federated learning is another machine learning technique that ensures the training data stays private to local devices. However, this method alone cannot guarantee complete data privacy. To secure federated learning, Wei et al. [6] suggest adding differential privacy while training a model. According to their method, they add measured noise to the model results on each local device before combining with the central server, which preserves the privacy of sensitive information.

Banse et al. [7] implemented federated learning with differential privacy to train models on non-i.i.d datasets. The article experimented with the model on three different datasets. The first dataset is the famous MNIST, which contains images of handwritten digits. The second is the FEMNIST dataset, similar to the first, and the third dataset includes medical records from 120 patients. The paper results show that adding data privacy does not significantly harm the model's performance.

III. Differential Privacy

This section explains the mathematical background of the DP algorithm and provides information about its properties, guarantees, and different usage scenarios.

A. Definitions of DP

Dwork et al. [8] firstly introduced the DP algorithm in 2006 to preserve privacy in statistical databases. The algorithm suggested adding random noise to the query's result to ensure that the query's actual result would not leak data. To understand the algorithm better, the following assumptions and notations are defined:

- The data provider fully trusts the server that possesses the database or simply the dataset.
- The dataset $\mathcal{X}$ presents the input space. Neighboring datasets are $\mathcal{X}'$, which are different from $\mathcal{X}$ only by one record.
- $\mathcal{M}$ is a randomized mechanism.
- $\mathcal{Y}$ presents the output space of $\mathcal{M}$.
- ε is the privacy parameter, which is a positive scalar.

Definition 1: (Differential Privacy) [8]
A mechanism $\mathcal{M}$ is ε-differentially private if for any two neighboring datasets $\mathcal{X}$ and $\mathcal{X}'$, and for any output $Y \in \mathcal{Y}$:

$$P[\mathcal{M}(\mathcal{X}) \in Y] \leq e^{\varepsilon} \cdot P[\mathcal{M}(\mathcal{X}') \in Y]. \quad (1)$$

Definition 1 guarantees that the output of the same query from two similar datasets will not differ much. The difference between the two outputs is defined by ε, which is common as a *privacy parameter* or *privacy budget*. If ε is a small number, from Definition 1, it follows that adding or removing one piece of data from the dataset will not drastically change the query result. This method ensures that the results will not leak any individual information from the dataset after querying. In different use cases, the piece of data can be different, meaning that it can refer to a single row of information or a group of rows that include information about the same person. This project concentrates on user-level privacy, about which more details can be found in Section III-D2.

Depending on the task and how much data is present, the choice of ε can vary in different applications. With the help of ε, it is possible to control the desired size of privacy. Smaller values of ε ensure more privacy as they bring closer results in Equation 1. However, it is essential to note that taking a very small ε can negatively affect the quality of the query [9]. So, paying attention to the tradeoff between privacy and accuracy is vital.

In addition to the above-defined Definition 1, there is a more flexible version of the DP definition commonly used for ML

model privatization, called approximate DP. It is the relaxation of the standard definition, which offers improved utility [10].

Definition 2: (Approximate Differential Privacy) [8]
A mechanism $\mathcal{M}$ is (ε, δ)-differentially private if for any two neighboring datasets $\mathcal{X}$ and $\mathcal{X}'$, and for any output $Y \in \mathcal{Y}$:

$$P[\mathcal{M}(\mathcal{X}) \in Y] \leq e^{\varepsilon} \cdot P[\mathcal{M}(\mathcal{X}') \in Y] + \delta. \tag{2}$$

ε and δ are non-negative numbers in Definition 2 that control the system's privacy levels. The value of δ presents the relaxation in this algorithm. It quantifies the probability of an adversary being able to differentiate between the original and neighboring datasets using the model or query results. If δ equals to zero, the mechanism will be ε-differentially private. If δ is non-zero and has a small value, then there is less risk of privacy violations and stronger privacy guarantees [11].

B. Properties of DP

To provide data privacy, DP suggests multiple properties that make the algorithm robust and adjustable for different applications.

1) Sequential composition:

One of the important properties of DP is Sequential composition:

Theorem 1. ***Sequential composition***
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 $\epsilon_1 + \epsilon_2$-differential privacy.

This property gives a chance to apply multiple privacy analyses on the same input dataset. To estimate the total privacy result, one needs to sum individual mechanism's ε-s and δ-s [17].

2) Parallel composition:

The next property is called Parallel composition:

Theorem 2. ***Parallel composition***
If $F_1(x)$ satisfies ϵ-differential privacy and we split a dataset X into k disjoint chunks such that $x_1 \cup \ldots \cup x_k = X$, then the mechanism which releases all of the results $F_1(x), \ldots, F_k(x)$ satisfies ϵ-differential privacy.

Compared to Sequential composition, Parallel composition divides the dataset into disjoint subsets and separately applies private mechanisms. This method ensures that the mechanism is applied to each data point only once as the subsets are disjoint. In contrast to Sequential composition, which would suggest $k\varepsilon$-differential privacy, Parallel composition provides ε-differential privacy [17].

3) Post-processing:

The third important property of DP is Post-processing:

Theorem 3. ***Post-Processing***
If $F(x)$ satisfies ϵ-differential privacy, then for any deterministic or randomized function g, $g(F(x))$ satisfies ϵ-differential privacy.

The property ensures that it is impossible to undo the privacy protection provided by DP. So, any change to the output of DP will still preserve a differential privacy guarantee with the same level of privacy [17].

C. Mechanisms of DP

There are multiple mechanisms that the DP algorithm can use to provide data privacy. The most famous and widespread three mechanisms are Laplace, Gaussian, and Exponential [10]. The first two options work for queries that output numerical answers. As their names can tell, they add random noise to the query's result either from Laplace or Gaussian distribution. The third option, the Exponential mechanism, is used when the outcome is not numeric but categorical, and respectively, in this case, the added noise will be from Exponential distribution.

Before introducing each mechanism in more detail, it is important to understand the concept of sensitivity. Sensitivity shows how much the query's output can change when adding or removing one piece of data from the input dataset. There are two types of sensitivity [12]:

- **Global sensitivity** calculates the maximum change in the query's result when one piece of data changes in any dataset. This approach considers all possible datasets that differ by one data point.
- **Local sensitivity** again measures the maximum difference in the outcome, which is caused by one element change within a dataset. Compared with global sensitivity, the local sensitivity considers changes within a specific dataset but not all possible datasets.

In our project, we will focus on using local sensitivity and will calculate the changes within the chosen dataset, for which we will provide more details in Section VI-A. Below is the formal definition of sensitivity.

Definition 3: (L_1-Sensitivity) [13]
The L_1-sensitivity of a function $f : X^n \to \mathbb{R}^k$ is defined as:

$$\Delta(f) = \max \|f(X) - f(X')\|_1, \tag{3}$$

where X is the original dataset and X' are neighboring datasets.

Thus, the DP mechanisms have a general approach; they add statistical noise to the data to make it private while keeping the function or model performance unchanged. To decide the amount of noise, one needs to consider the four main components presented in Figure 2: sensitivity, desired epsilon, desired delta, and the type of noise to add. We defined sensitivity above and discussed epsilon and delta in Section III-A. It remains to introduce the three known mechanisms of DP to understand what possible types of noises can be added to the DP model.

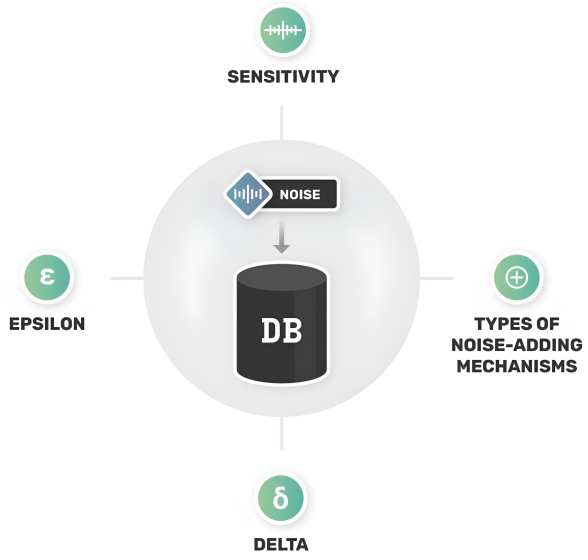


Fig. 2: Factors on which noise depends [12].

1) Laplace Mechanism:

The definition below represents the Laplace distribution, which is the basis of the Laplace mechanism.

Definition 4: (Laplace distribution) [13]
The Laplace distribution with location and scale parameters 0 and b, respectively, has the following density:

$$p(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right). \tag{4}$$

The variance of the Laplace distribution is $2b^2$. It is a symmetrical distribution with a higher peak compared to the normal distribution. Laplace distribution is a combination of positive and negative Exponential distributions as $x \in \mathbb{R}$.

Definition 5: (Laplace mechanism) [13]
Let $f : X^n \to \mathbb{R}^k$. The Laplace mechanism is defined as:

$$\mathcal{M}(x) = f(x) + (Z_1, \ldots, Z_k), \tag{5}$$

where the Z_i are independent Laplace$\left(\frac{\Delta f}{\varepsilon}\right)$ random variables.

The scale parameter of the mechanism grows larger as the sensitivity (Δ) of the function increases. That is intuitive, as the functions with higher sensitivity are likely to change hugely in case of adding or removing one piece of data, which means functions need more noise to provide data privacy. Additionally, the scale parameter decreases when the privacy parameter ε increases.

The Laplace mechanism provides $(\varepsilon, 0)$-differential privacy or is ε-differentially private [14].

2) Gaussian Mechanism:

An alternative to the Laplace mechanism is the Gaussian mechanism, which adds noise from Gaussian distribution. Compared to Laplace, the Gaussian mechanism does not preserve pure ε-differential privacy; it ensures (ε, δ)-differential privacy.

Definition 6: (Gaussian mechanism) [15]
Given a real-valued function $f(x)$. The Gaussian mechanism is defined as:

$$F(x) = f(x) + \mathcal{N}(0, \sigma^2), \tag{6}$$

where $\mathcal{N}(0, \sigma^2)$ denotes Gaussian noise with center 0 and variance $\sigma^2 = \frac{2\log(1.25/\delta)\cdot(\Delta f)^2}{\varepsilon^2}$.

In contrast to Laplace, the Gaussian mechanism provides weaker guarantees of DP; however, it suggests some advantages over the first one. The Laplace mechanism only accepts L1 sensitivity, while the Gaussian mechanism can accommodate both L1 and L2 sensitivities. This characteristic makes the Gaussian mechanism powerful, as in cases where L2 sensitivity is lower than L1 sensitivity, the mechanism can add less noise and achieve better results. That is why the DP-SGD algorithm is commonly used in ML applications, which is based on the Gaussian mechanism [10]. More details about the algorithm will be provided in Section III-F.

3) Exponential Mechanism:

The mechanisms mentioned above (Laplace and Gaussian) are for queries with numeric outputs, which add noise directly to the outputs. There is also the Exponential mechanism, which can handle non-numeric results and return the query result without noise while preserving privacy. The Exponential mechanism satisfies ϵ-differential privacy.

Definition 7: (Exponential mechanism) [16]
The Exponential mechanism is defined based on the following points:

- The analyst selects a set R of possible outputs.
- The analyst specifies a scoring function u with global sensitivity Δu.
- The exponential mechanism outputs r with probability proportional to:

$$\exp\left(\frac{\epsilon u(x, r)}{2\Delta u}\right) \tag{7}$$

The Exponential mechanism selects an output from the probability distribution that is defined in Equation 7. The selected output is the most appropriate output among the possible outputs that also preserves privacy [15].

D. Application Cases

This section will discuss various problem settings in which DP might be applicable, categorizing cases based on the following factors: whether the data providers trust a central server to aggregate their data, whether protection is needed for individual rows or groups within the dataset, and which part of the architecture is susceptible to becoming public.

1) Global vs Local DP:

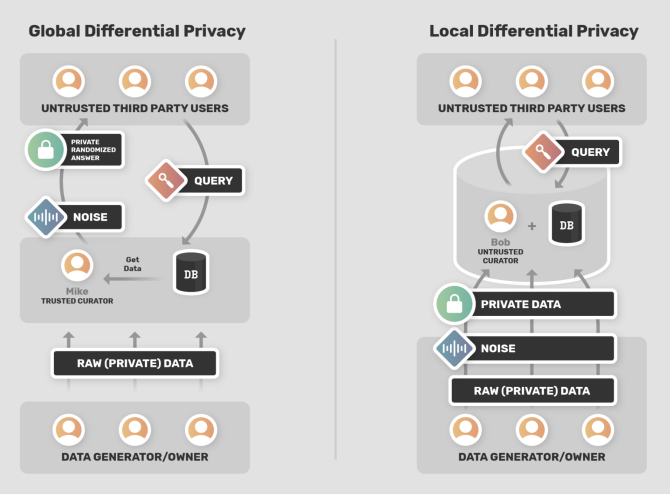


Fig. 3: Global Differential Privacy and Local Differential Privacy [18].

Local DP (LDP) accounts for settings where data contributors do not trust the centralized server or a data aggregator. The solution LDP suggests for such cases is that individuals add noise to the raw input data before sharing it with a centralized aggregator. The noise addition preserves the privacy of the original data by ensuring that the aggregator has no access to the original data. However, the LDP approach increases the overall noise level in the dataset, which can negatively affect the quality and accuracy of the model. Consequently, people usually use higher epsilons in the case of LDP to achieve meaningful model results, which limits the practical applicability of the method [18].

Global or Central DP (CDP) addresses the scenario when data providers trust their sensitive data to a centralized server and a data aggregator, usually referred to as a trusted curator, who is responsible for conducting statistical analysis or running ML models following privacy constraints. Consequently, any outcomes released to third parties must respect the privacy of data providers. In this paper, we concentrate on Global DP. In our scenario, we are in the role of the trusted curator, whom the raw dataset is given on which we will implement the model and use DP to ensure data privacy.

2) User-level privacy vs group-level privacy:

The setting might also differ based on whose privacy preservation is under question. The conventional concept of differential privacy accounts for ensuring privacy for each data instance taken separately, as it ensures that the outcome of the mechanism does not vary a lot as a result of adding or removing a single instance of the dataset. Ensuring differential privacy in this setting is referred to as user-level individual privacy.

In other settings, the problem definition might require ensuring privacy not for individuals but for groups of individuals where groups form a partition of the dataset. Examples might be people who share a common ethnic background, a company of employment, etc. In such a setting, adjacent datasets are defined as:

Definition 8: (Group-level adjacent data sets) [19] Two data sets D_1 and D_2 are group-level adjacent data sets of each other if $\exists\ \mathcal{G}_i \in \mathcal{G}$ such that $\mathcal{D}_1 = \mathcal{D}_2 \cup \mathcal{G}_i$.

In the context of group-level differential privacy, a differentially private mechanism A is defined as:

Definition 9: (Group differential privacy) [19] A randomized algorithm A guarantees ϵ_g-group differential privacy if for all adjacent data sets D_1 and D_2 differing by at most one group $G_i \in G$, and for all possible results $S \subseteq \text{Range}(A)$,

$$P[A(D_1) = S] \leq e^{\epsilon_g} \times P[A(D_2) = S]$$

In the scope of this project, we implement user-level privacy.

3) DP Settings:

Depending on what part of the model the attacker has access to, the level and methods needed for protection may change. There are different parts of an ML system that can be at risk and can be violated:

1) If the raw data, which includes sensitive information, needs to be published, then the training dataset itself needs privatization.
2) The attacker might have access to the updates from individual users, precisely, the intermediate or aggregated model information. This can be a common concern in the case of federated learning, which we will discuss in more detail in Section V.
3) Another scenario suggests that the final model structure and parameters are publicly available, which is the case with ML as a service platforms that provide with white-box access to the model.
4) The final scenario infers that only the predictions of the model are public, but not the model structure or parameters. These are the black-box model access examples. [20].

We can limit access to trusted people for some parts, like the raw data and training dataset. However, for other parts, like the final model parameters and predictions, we need to use data privacy techniques to protect them, as they may be at risk when using the model [10]. In the scope of this project, we consider the fourth setting.

E. DP in ML

There are three possible applications of the DP algorithm in machine learning applications [10]:

- The first variant suggests using DP at the input level. Using the Post-processing property of DP explained in Section III-B3, after making input data differentially private, any model trained using that data and outputs of that model will also be differentially private.
- The second approach is adding DP during the training ML model. There are two possible ways based on the application. Either data privacy is applied only on labels by treating independent features as public, or both features and labels are private and need protection. The second one is the most common case in ML when people need

privacy to train ML by considering features and labels. In these cases, the DP-SGD algorithm is commonly used, which we will discuss in Section III-F.

- The third case is when the model is not public, but only its publications are, so there is a need to apply DP to the ML predictions. This approach is applicable when users only have access to the model predictions via a trusted server by providing their own input.

Our project focuses on applying DP while training the ML model using the DP-SGD algorithm to preserve the private model training process.

F. DP-SGD

Stochastic gradient descent (SGD) is an optimization method commonly used in ML applications [21]. It serves as the basis for the DP-SGD algorithm, which is an application of the Gaussian Mechanism, as shown in Algorithm 1. Section VI will discuss our experiments and implementations using DP-SGD in more detail.

Algorithm 1: (Differentially private SGD) [22]
(Dataset $\mathcal{D}$, loss function $L_{\mathcal{D}}(w)$, learning rate r, batch size b, noise scale σ)

1) **for** $t \in [T]$ **do**
2) Randomly sample a batch B_t with $|B_t| = b$ from D
item **for** each sample $x_i \in B_t$ **do**
3) $g_i = \nabla L_i(w_t)$
4) **if** $g_i >$ threshold
5) $g_i = g_i *$ threshold $/||g||_2$
6) $\tilde{G}_B = \frac{1}{b}\left(\sum_i \bar{g}_i + \mathcal{N}\left(0, \frac{2\log\left(\frac{1.25}{\delta}\right)\cdot(\Delta f)^2}{\varepsilon^2}\right)\right)$
7) $\tilde{w}_{t+1} = \tilde{w}_t - r\tilde{G}_B$
8) Return $\tilde{w}_T$ and accumulated (ϵ, δ)

As Algorithm 1 suggests, a machine learning algorithm will be made differentially private if each step in the process of gradient descent is preceded by gradient clipping and noise addition.

Gradient clipping involves scaling down the gradient if its L_2 norm is above a predefined threshold. The purpose of gradient clipping is to prevent minor changes in the input from causing significant changes in the output function, thus ensuring that data points that are slightly different from the rest of the training dataset do not affect the output. This method also handles outliers. As a result, the contribution of individual datapoints to the output function is decreased, unless the datapoint is similar to the rest of the dataset.

Noise addition entails generating a vector of the size of theta of independent sample from the Gaussian distribution with a variance of $\frac{2\log(1.25/\delta)\cdot(\Delta f)^2}{\varepsilon^2}$, and adding the two vectors. Δf refers to the L_2 sensitivity of the mechanism, which is defined in Definition 10. As sensitivity accounts for the maximum contribution of a single data point, the greater value implies a greater variance of the added noise is needed to conceal the contribution of individual data points to the output function.

Definition 10: (L_2-Sensitivity) [12]
The L_2-sensitivity of a function $f : X^n \to \mathbb{R}^k$ is defined as:

$$\Delta(f) = \max \|f(X) - f(X')\|_2, \tag{8}$$

where X is the original dataset and X' are neighboring datasets. The function calculates all L_2 norms between each neighboring dataset and the original dataset and, in the end, takes the maximum norm.

IV. Threat Scenarios and possible attacks

In the scope of this paper, we discuss the Membership Inference Attack, or MIA, which presents a scenario where the adversary, who has access to a data record and an ML model, intends to infer whether the given record has been part of the training set of the model. MIA poses a threat to the privacy of the individual to whom the given data record belongs. Firstly, the information about whether he participated in the study or not may inherently be confidential. Secondly, considering a scenario where the adversary has partial access to an individual's data record, for example, having knowledge about some of the variable values of the individual that the model takes as input, as well as knowing that the individual's data record is in fact part of the training set, the adversary may experiment with the values of remaining variables, until the attack mechanism indicates that the given vector of records has been part of the training set, thus, revealing the individual's information regarding these remaining variables, which may potentially be private.

MIA is often used to attack ML models that have been created using Machine Learning as a Service platform, where any individual is able to upload a dataset, specify the type of the ML task to be performed on the dataset, and receive a corresponding model. Some ML as a service platforms provide the user with the model structure and its trained parameters. In this case, the user has white-box access to the model. Other ML-as-a-service platforms only provide API access to the model, meaning the user has no access to the structure and the parameters of the model; however, he is able to query it through the provided API. In this case, the user has black-box access to the model.

Moreover, some ML as a service platforms, regardless of whether they provide white-box or black-box access to their models, allow not only the client who requested the development of a model but also third parties to have access to it. Thus, third parties may perform MIA to attack the training dataset of the models, gaining access to any private information present in them.

In this paper, we examine MIA with black-box access to the model, as this scenario offers a broader insight into possible attack methods.

In the setting of MIA, the model being attacked is referred to as the target model. Additionally, shadow models must be developed. The purpose of the shadow models is to mimic the behavior of the target model. The conditions of the shadow models are rather relaxed, allowing them to be of any structure as long as they solve the same ML task as the target model.

This flexibility makes MIA suitable for black-box attacks. Another requirement for the shadow models is that they have the same input format as the target model. This entails having the same number of variables as the target model, with each variable being of the data type corresponding to the target model. The output of the shadow models lies within the continuous set [0:1], representing the confidence of the value 1 in the case of binary classification. Synthetic datasets may be used for training the shadow models

Having trained the shadow models, the attack model must be trained. It can be of any structure that enables solving binary classification tasks, as the purpose of the attack model is to distinguish whether the given data point was present in the training set of the target model or not. During the training process, the attack model takes a vector of length two as an input, where the first element is the true label of a data point that has or has not been in the training set of any of the shadow models, and the second input is the output of the shadow model. If the datapoint has been part of the training process of any shadow model, the corresponding shadow model must be used to get the predicted value. Otherwise, any shadow model may be used. Once the attack model is trained, it can be implemented to make predictions on the target model. Having a data instance, the second element of the input vector can be obtained using the black-box API, while the actual label of the datapoint may be either known to the adversary or can be experimented with different values. The purpose of the attack model is to find any differences between the way the shadow models perform on the data that has been in their training set, as opposed to the ones that have not, and apply this knowledge to the target model. The standard metrics used to evaluate the performance of the attack model are precision and recall [23].

V. Federated Learning

The term federated learning (FL) was first introduced by Google in 2016. During that time, people started to pay considerable attention to personal data privacy. There were broad discussions on this topic, and big companies like Facebook were under suspicion if they provided necessary privacy while working with massive datasets of individuals. As a solution, federated learning provides a new, efficient technique for training ML models. Federated learning has various applications in healthcare and finance and has massive usage as it develops AI models while preserving the privacy of individuals' sensitive information [24].

Before explaining federated learning in more detail, let us introduce the difference between centralized and decentralized machine learning. In the case of traditional ML, all users' data is located in one place, usually called a data warehouse. The model training process is done on the central server, and users can access it. Figure 4 shows centralized ML.

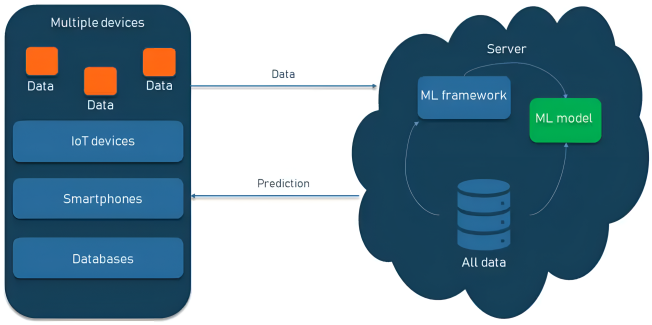


Fig. 4: Centralized Machine Learning [25].

On the other hand, in the case of decentralized ML, the training process is done locally for each device. So, there is no need to communicate with the central server to complete local training [25]. Figure 5 illustrates decentralized ML.

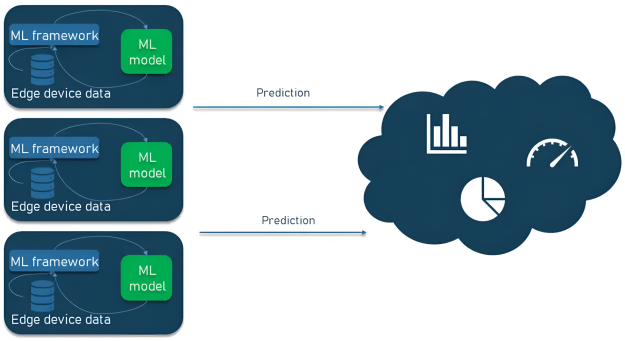


Fig. 5: Decentralized Machine Learning [25].

In federated learning, multiple users collaborate to train the same model using the decentralized approach. Each user has access to the model, trains it locally on an individual device, and sends the local training updates to the central server, where all separate model updates are aggregated. The steps of federated learning are presented in Figure 6.

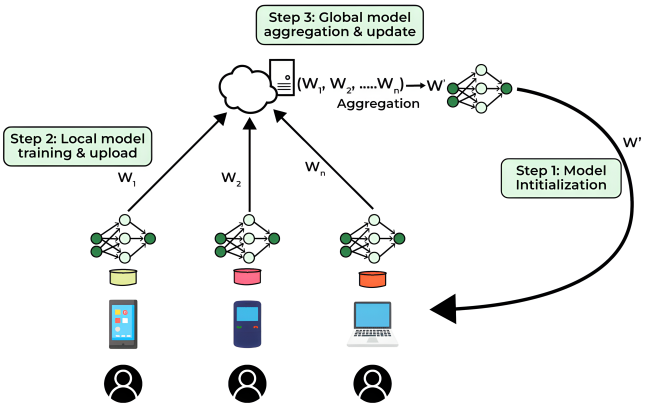


Fig. 6: Federated Learning [26].

Federated learning avoids sharing raw data centrally; it does calculations on isolated datasets and only gathers local results into the central server. Compared to distributed learning, in the case of federated learning, local datasets do not need to be independent and identically distributed, meaning that data distributions can vary across devices [27]. Thus, federated learning is another tool that preserves data privacy by reducing the risk of centralized vulnerabilities.

FL offers multiple advantages while training models, including reduced power consumption due to smaller data sizes in local devices and privacy preservation due to keeping user data locally without sharing with the central aggregator. Additionally, federated learning provides better model performance by considering multiple devices' real-time training updates. There are also some limitations besides all these advantages, as in FL, user devices and datasets can be different, which in some scenarios can create inconsistency or slow down the overall training process [26].

Federated learning is a powerful tool that powers AI models while ensuring data privacy during training. Its importance is also evidenced by the fact that big companies like Google [24], Apple [28], Amazon [29], Facebook [30], and Microsoft [31] use the tool. Large companies pay huge attention to the privacy of sensitive user information, which makes them more successful. The companies use federated learning along with differential privacy to ensure complete data privacy.

VI. Models and Methods

A. Data

The publicly available Adult Census income dataset was chosen for our experiments. It not only involves sensitive information about the data providers, such as their salary range, but also contains demographic information, which is the key factor enabling the adversary to infer the identity of data providers.

The row dataset consists of 14 columns that contain demographic information and the target column that indicates the yearly salary of the individuals, categorizing it as either above or below 50,000 USD. One downside of the dataset is that it is imbalanced, with the two unique values of the target column comprising the 75% and 25% of it. As a result of data preprocessing, the column education-num was dropped. Each unique value of the column education-num corresponded to only one value in the column education, meaning that the two columns carried the same information. Similarly, the column relationship was dropped for its statistically significant (Chi-squared test for independence, p = 0.0) similarity to the column marital-status. The columns with continuous values were scaled to the range $[0; 1]$, while the columns with categorical values were dummified. When splitting the data into train, validation, and test sets, the stratified sampling method was applied to the column native-country, for its imbalanced nature.

B. Neural Network Model

To solve the classification problem of our chosen dataset described in Section VI-A, we implement a PyTorch neural network (NN) model. The implementation includes classes and functions to define the data loading process, model architecture, training, and evaluation steps.

Our NN uses two fully connected (dense) layers with a dropout regularization layer. The activation function ReLU is applied after the first linear transformation to introduce non-linearity to the model. After the activation function, the dropout layer is called to provide regularization and avoid overfitting. The model uses cross-entropy as a loss function, which is commonly used for classification tasks. We use SGD as our optimization method to implement DP further.

After defining the model, we first train it using standard steps without DP. During each training epoch, the function responsible for training iterates over the training data, computing the loss, performing backpropagation, and updating the model parameters. Additionally, the function evaluates the model performance based on the validation data. After each iteration, the loss and accuracy of validation and training datasets are monitored. The test accuracy of the trained model we are reaching is approximately 86%, which is rather good in the case of our imbalanced dataset. As in the case of imbalanced datasets, it is expected that it is challenging for models to learn data patterns and show high-accuracy results.

Next, we train the model using the steps mentioned above and add DP to the training process. There is a PyTorch library called Opacus, and we use this existing library to preserve our data privacy. Opacus library is applying DP to PyTorch models [32]. The library's primary goal is to protect individuals' sensitive information while training the model without affecting the model's accuracy much. Opacus defines a privacy engine that modifies the PyTorch optimizer to add DP. The library focuses on stochastic gradient descent optimizer and, as a result, provides a DP-SGD method for protecting data privacy. The Opacus method does not directly affect the original data but modifies the gradients of parameters during model weight updates. The library undertakes two main steps to implement DP-SGD:

1) First is the gradient clipping. The library is clipping the L2 norm of gradients up to the predefined threshold during the training process. It is done to prevent the gradient exploding problem, which happens when gradients become too large. So, to ensure that model parameters are stable and do not affect the optimization process, Opacus clips the gradients up to the given clipping threshold. To decide on our clipping threshold, we researched other projects' used cases and did empirical testing to define it. For our model, we choose the maximum gradient norm to be 1.0.
2) The second step is adding noise to the gradients, a crucial step in the DP-SGD algorithm. It is essential to add the right size of noise to data, as adding extra noise can affect the model performance, and adding very little

noise will fail to provide complete data privacy. Opacus uses noise multiplier parameter to decide the amount of noise to add the gradients. Higher values, as mentioned, can provide strong privacy but will affect the model's accuracy, so it is vital to find the right amount to balance privacy and utility tradeoffs. For our model, we choose the noise multiplier equal to 1.1.

The below Figure 7 shows the visual representation of the above-explained two steps.

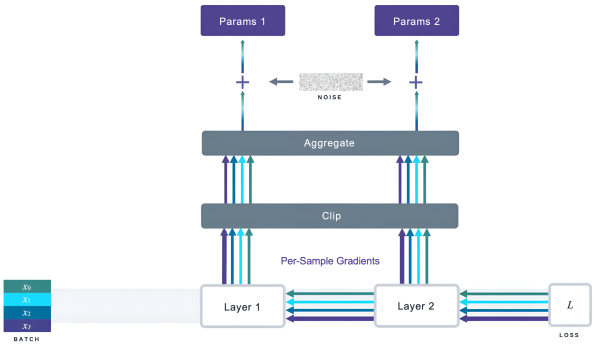


Fig. 7: DP-SGD algorithm by Opacus library. [33].

Opacus library also takes input value for the delta parameter, defined based on the desired privacy level. As we want to provide more robust data privacy guarantees, we define the delta parameter to a small number of 1×10^{-5}.

The library suggests accountants with three different possible mechanisms. As in our project, we are focusing on the DP-SGD algorithm, we fix the Opacus accountant to use *gdp* mechanism, which corresponds to the Gaussian mechanism. One advantage of the Opacus library is that it, during the training process, estimates the ε or *privacy budget* based on the Dual and Central Limit Theorem (CLT) [34]. We achieve nearly 84.6% accuracy with the Opacus DP library.

Finally, we manually implement the DP-SGD algorithm to make our model differetially private. In the manual implementation, the logic of the DP-SGD algorithm is identical, as explained above; however, in this case, we manually implement the gradient clipping and noise addition function without using any existing library. We also manually define the sensitivity function, which is vital to calculate for determining the amount of noise to add gradients; however, as our dataset is rather extensive and the computation power is not so enormous, our computers could not compute these heavy functions of sensitivity that is why we estimate the sensitivity based on some subset of neighboring datasets instead of iterating over all neighboring datasets.

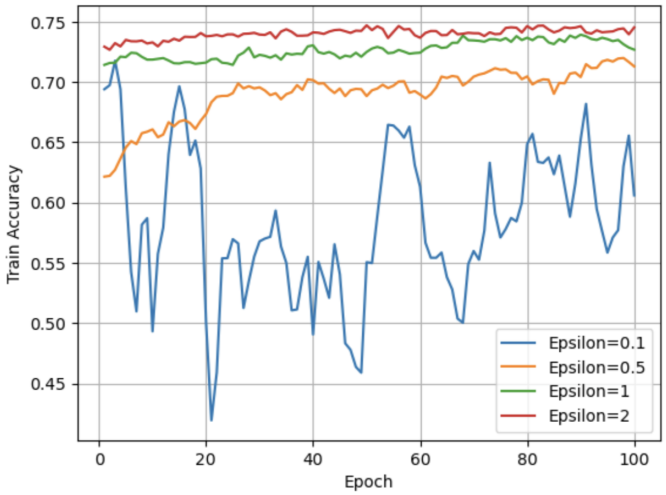


Fig. 8: Training accuracy results with different data privacy values.

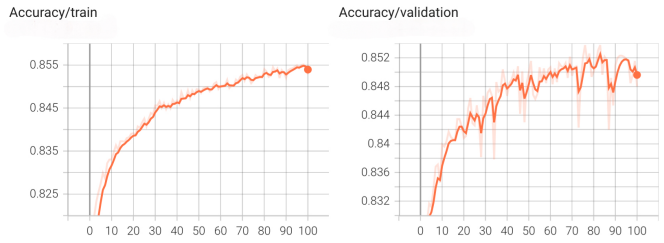
In the manual implementation, similar to the Opacus case, we define delta as a small number to have a robust private model. Based on empirical tests, we choose the gradient clipping parameter to be equal to 30. To select the ε parameter, we train our model with different ε values as shown in Figure 8. The plot shows that increasing ε provides better accuracy results, which is logical as larger ε-s ensure less data privacy but better model performance. To balance the privacy and accuracy tradeoffs, we set ε to be equal to 0.8 to have the best accurate and secure model choice.

For this approach, we achieve less accuracy compared to our previous approach with Opacus, nearly 76.6%. This is logical because the PyTorch Opacus library uses different optimization techniques, which improve the overall result quality.

During the whole training process, we use training and validation datasets for model performance checking and, in the end, use the test dataset for the final model evaluation. We use Tensorboard to track the whole training process. Tensorboard results from our standard training without DP are shown in Figure 9.

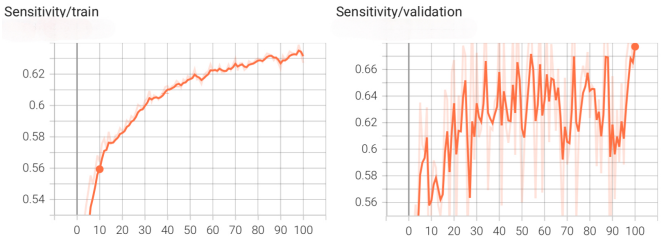
Accuracy plots show a consistent increasing trend while the loss gradually decreases, which indicates that the model learning process is effective. Other performance metrics such as sensitivity, specificity, precision, and F1 score demonstrate values greater than 0.5 and closer to 1, which is also a good sign of training results. However, the validation plots of specified metrics display fluctuating patterns, constantly increasing and decreasing. It possibly suggests that the model faces challenges while generalizing unseen data.

Considering that the dataset is imbalanced, we emphasize the F1 score, which balances precision and sensitivity and is a valuable metric for imbalanced classes. In our case, the F1 score is approximately 0.7, which assures that the model performs rather well while training.

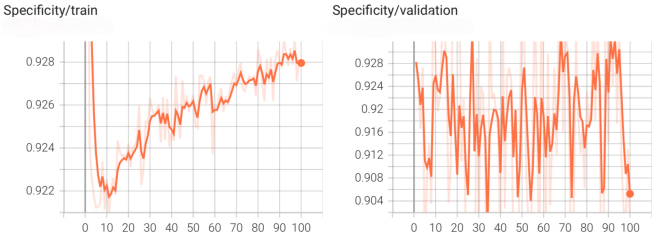


(a) Train and validation accuracy.

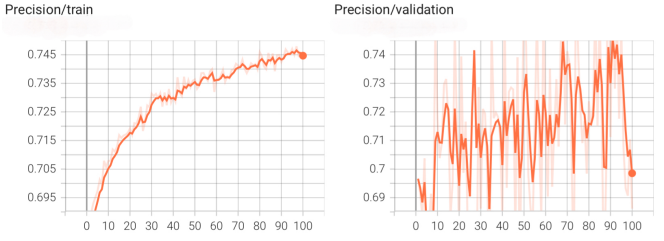
(b) Train and validation loss.



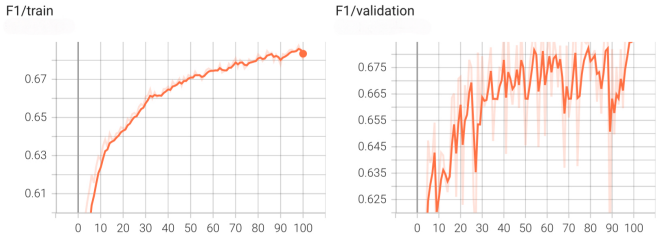
(c) Train and validation sensitivity (recall).



(d) Train and validation specificity.



(e) Train and validation precision.



(f) Train and validation F1 score.

Fig. 9: Tensorboard train and validation metrics monitoring over 100 epochs (without DP NN model).

The table below shows a summary of the performance metrics on the test dataset for the three training experiments explained in this section.

TABLE I: Performance metrics on test dataset

	Without DP	**With DP (Opacus)**	**With DP (manual)**
Accuracy	0.8596	0.8461	0.7663
Loss	0.3035	0.4855	14681610.4717
Sensitivity	0.7212	0.5479	0.3958
Specificity	0.9041	0.9419	0.8854
Precision	0.7075	0.7521	0.5262
F1 Score	0.7143	0.6339	0.4517

The table displays that without Differential Privacy (DP) model shows higher accuracy (0.8596) than DP models, which is natural, as adding data privacy affects model performance. Notably, the loss rate for the model with manual DP implementation is high, which may be due to the complexity of the model architecture. Despite the high loss, it is vital to note that the loss function decreases during training, ensuring the learning of the model.

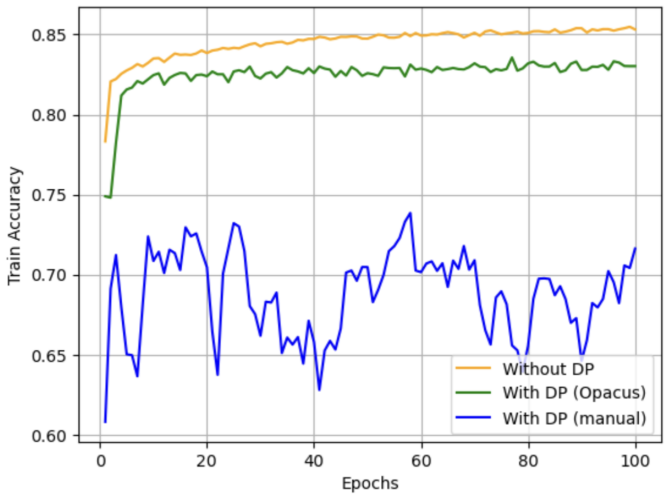


Fig. 10: Comparison of training accuracy between models.

The visualization in Figure 10 compares the training accuracy among three models. The PyTorch Opacus library presents quite similar results to the model without DP. Compared to the existing PyTorch library, the manually implemented model achieves lower accuracy, considering the fewer optimization techniques and tricks.

C. Logistic Regression Model

Our aim is to compare the performance of four models - our implementation of logistic regression without DP, our implementation of logistic regression with the DP SGD algorithm, our implementation of logistic regression with a variation of the DP SGD algorithm that introduces an adjustable clipping threshold, and the logistic regression with DP of the diffprivlib package.

First, we implement a logistic regression without DP. Considering that the loss of the logistic regression, binary cross-entropy, is a convex function and that theta, the vector of final

parameters, will converge to the global minima despite its initialization, we initialize theta as the 0 vectors and assign the step size to the constant 0.9 for the sake of simplicity. The number of gradient descent steps to be performed is chosen based on the Tensorboard plots in Figure 11 depicting the progression of accuracy, sensitivity, specificity, precision, F1 score, and loss on train and validation sets over 5000 iterations. The graphs illustrate how all metrics keep improving at a high rate until 4500 iterations, after which the rate of change decreases. The pattern is more visible on the plots for the train set, as they are more stable. Thus, 4500 is chosen as the number of iterations.

Training the model without DP, we get an accuracy of 85.35% on the test set. Information about the other metrics can be found in Table IV.

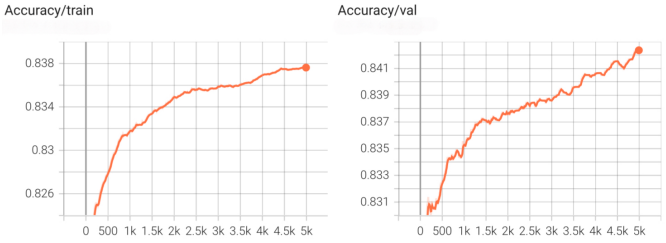
We proceed to add differential privacy to the model. We estimate the local L2 sensitivity for the lack of computational power, calculating it on randomly chosen 2000 entries of the dataset, and get the estimate of the sensitivity as 5.6.

Next, we implement functions for clipping the gradients above the defined threshold and for noise addition, which depends on epsilon, delta, and sensitivity. The clipping threshold is set to 0.9, considering the range of the L2 norms of theta during the training process, which can be observed by uncommenting the print statement in the definition of the logistic regression function without DP. Finally, we implement a function for training the logistic regression with DP that initializes theta at the 0 vectors and applies gradient clipping and noise addition as described in Algorithm 1 on every step of the gradient descent. The number of gradient descent steps is explicitly given as a hyperparameter, the value of which, along with the values of other hyperparameters that are also present in the function of logistic regression without DP, are set equal to the ones selected for the training of logistic regression without DP. In this way, we are able to compare the two algorithms.

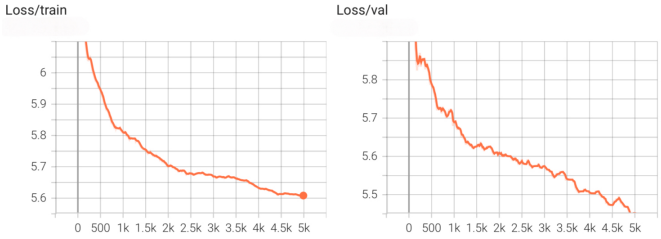
To choose optimal values for the hyperparameters epsilon and delta, we experiment with several pairs and track the progression of the accuracy on the train set over the number of iterations.

The Figure 12 suggests that the epsilon-delta pairs of (2, 0.9), (0.9, 0.1), and (0.1, 0.1) result in the highest accuracy on the train set. We plot them on Figure 13 for a more thorough understanding of their performance, which suggests selecting the pair (2, 0.9) for further consideration.

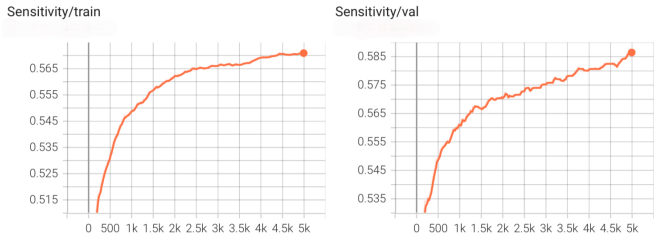
The output of the training of the logistic regression with DP includes adding random noise. Thus, the output varies with every training, even though the hyperparameters and the initialization of theta are fixed. A larger variance of the noise results in more varied output parameters. To take this variation of outputs into account while examining the performance of the algorithm, we train it 10 times and record the summary statistics of the performance metrics on train and validation sets. The results can be found in Table II.



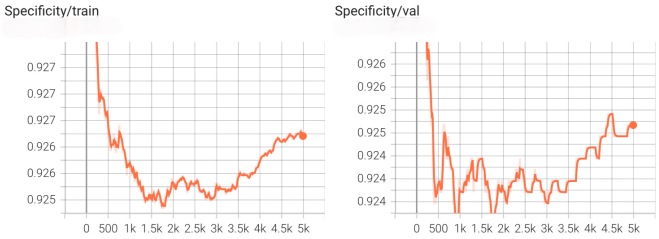
(a) Train and validation accuracy.



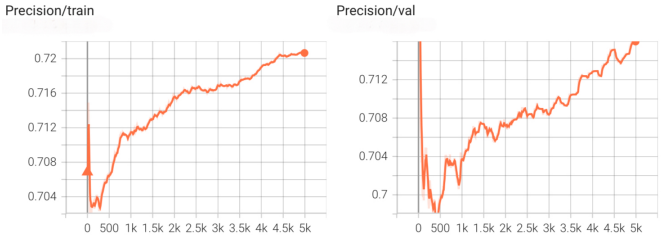
(b) Train and validation loss.



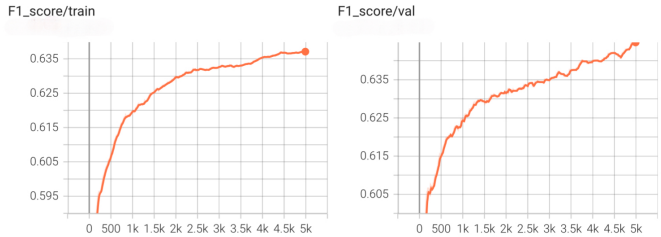
(c) Train and validation sensitivity (recall).



(d) Train and validation specificity.



(e) Train and validation precision.



(f) Train and validation F1 score.

Fig. 11: Tensorboard train and validation metrics monitoring over 5000 iterations (without DP Logistic Regression model).

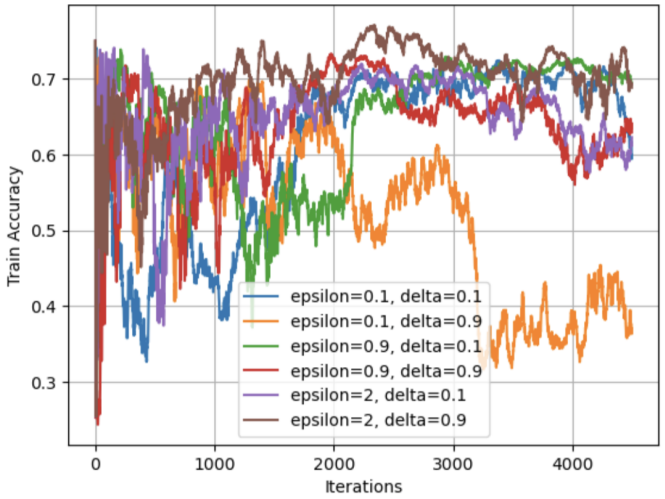


Fig. 12: Training accuracy results with different epsilon and delta values.

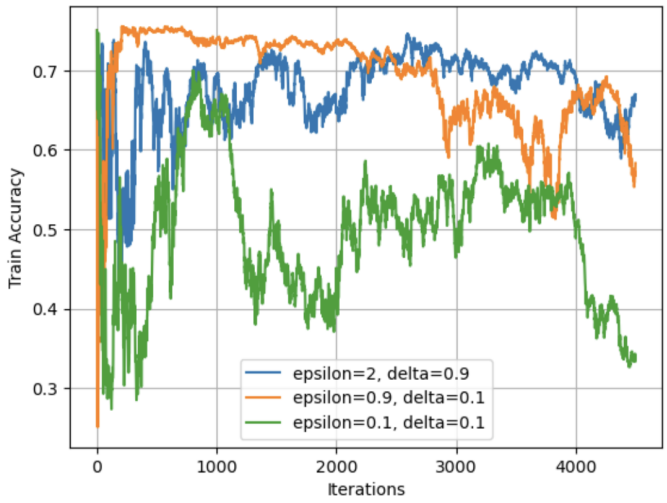


Fig. 13: Training accuracy results with different epsilon and delta values.

TABLE II: DP SGD, Performance metrics on train and validation datasets

		Train	Validation
Accuracy	Mean	0.7154	0.7189
	Min	0.6911	0.6947
	Max	0.7579	0.7583
Sensitivity	Mean	0.4904	0.4946
	Min	0.3764	0.3782
	Max	0.6503	0.6562
Specificity	Mean	0.7903	0.7913
	Min	0.7318	0.7362
	Max	0.8467	0.8417
Precision	Mean	0.4391	0.4348
	Min	0.3934	0.3907
	Max	0.5158	0.5046
BCE loss	Mean	9.8303	9.7074
	Min	8.3635	8.3496
	Max	10.6691	10.5439

We trained the model once again to fix the output theta. The logistic regression with DP obtains an accuracy of 72.09% on the test set. Information about the remaining metrics can be found in Table IV.

Having trained a logistic regression with DP, we implement one more algorithm, which was our variation of DG SGD. The motivation to develop a variation of DP SGD came from our observation of decreasing norms during the training process of the logistic regression without DP. While DP SGD suggests having a constant gradient clipping threshold, which, in case of decreasing gradients, results in clipping the gradients of the first steps of the gradient descent and keeping the gradients that appear on the later steps unchanged, we experiment with an adjustable clipping threshold that would allow the gradients to be clipped regardless of in which step of the gradient descent they emerge. Specifically, starting the 4th step of the gradient descent, we scale the current gradient, treating the maximum norm of the previous 3 gradients as a threshold. The algorithm is presented below.

Algorithm 2: (Differentially private SGD our implementation) (Number of iterations $\mathcal{N}$, loss function $L_{\mathcal{D}}(w)$, learning rate r, empty list of gradient norms p,)

1) Initialize θ as zero-vector of size of columns of X train
2) **for** i in $\mathcal{N}$ **do**
3) $g_i = \nabla L_i(w_t)$
4) p.append($||g||_2$)
5) **if** $\mathcal{N} > 2$
6) Remove first element of p
7) threshold = max(p)[:-1]
8) **if** $g >$ threshold
9) $g = g*$ threshold $/||g||_2$
10) $g = g + \mathcal{N}\left(0, \frac{2\log\left(\frac{1.25}{\delta}\right)\cdot(\Delta f)^2}{\varepsilon^2}\right)$
11) $\theta = \theta - r * g$
12) Return θ and accumulated (ϵ, δ)

The model is trained 10 times to capture the variability of the performance metrics. The results of the train and validation sets are shown in Table III.

TABLE III: DP SGD adaptive clipping, performance metrics on train and validation datasets

		Train	Validation
Accuracy	Mean	0.7004	0.7065
	Min	0.5774	0.5786
	Max	0.7370	0.7409
Sensitivity	Mean	0.4233	0.4378
	Min	0.2667	0.2725
	Max	0.6344	0.6562
Specificity	Mean	0.7926	0.7933
	Min	0.6612	0.6576
	Max	0.8516	0.8563
Precision	Mean	0.4076	0.4100
	Min	0.2422	0.2392
	Max	0.4694	0.4644
BCE loss	Mean	10.3480	10.1361
	Min	9.0826	8.9503
	Max	14.5961	14.5558

To fix an output theta, we train the algorithm once again and get an accuracy of 76.01% on the test set, while the values of the other metrics can be found in IV.

TABLE IV: Performance metrics on test set

No DP	DP SGD	DP adaptive clipping	
Accuracy	0.8535	0.7209	0.7601
Sensitivity	0.5982	0.5152	0.4545
Specificity	0.9355	0.7871	0.8584
Precision	0.7489	0.4375	0.5078
F1 score	0.6651	0.4731	0.4797
BCE loss	5.0614	9.6391	8.2846

As the tables suggest, the performance metrics are rather similar when training the logistic regression with DP, whether with or without adaptive clipping.

Lastly, we utilized the diffprivlib library to run a logistic regression with and without DP. Without DP, the accuracy is 86.08%. Implementing logistic regression with DP, we set the epsilon and delta to 2 and 0.9, respectively, to be able to compare it with the results of previous algorithms. The library uses an algorithm that is more complex than DP SGD and has more hyperparameters, which we will not discuss in this paper. However, as our aim is to compare the library with our implementations, we did not specify any additional parameters, allowing them to be set to their default values. As a result, we got an accuracy of 82.09%.

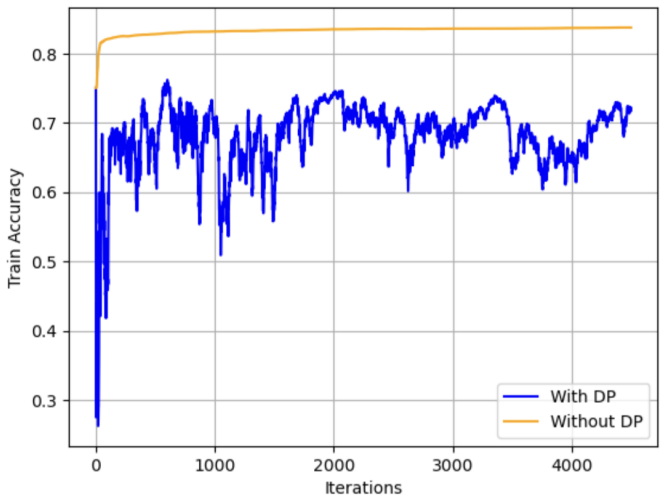


Fig. 14: Comparison of training accuracy between models.

D. Federated Learning with Differential Privacy using Support Vector Machines

After implementing different scenarios for the binary classification for our chosen dataset, including neural network and logistic regression with DP and without DP, we also implement the federated learning (FL) model with DP. The goal of implementing the FL model with DP is to allow multiple users to train a differentially private model collaboratively. We choose the famous Support Vector Machines (SVM) model to solve the binary classification task. SVM is a supervised machine learning algorithm. The main task that the algorithm solves is to find the hyperplane that best separates the data points of different classes. SVM tries to maximize the margin, which is the region between the support vectors, to differentiate between the classes better. SVM also includes a regularization term that helps avoid overfitting by balancing the tradeoff between maximizing the margin and minimizing classification mistakes [35]. Figure 15 illustrates the SVM model.

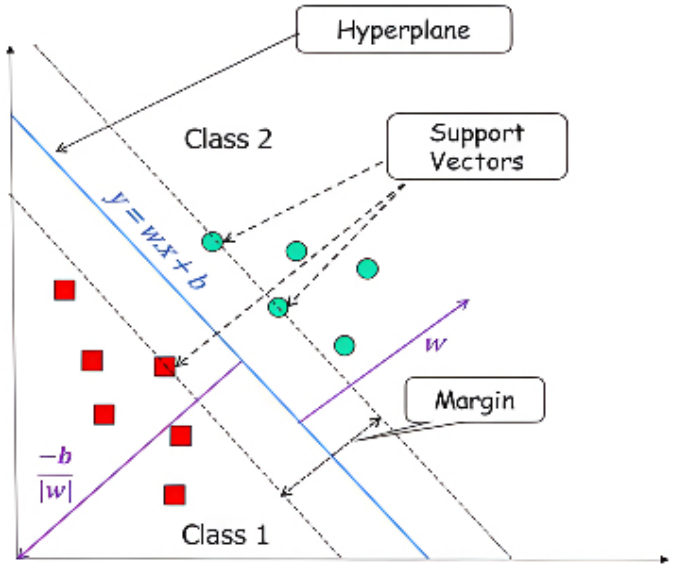


Fig. 15: Support Vector Machines. [35].

The implementation includes functions that load the data, preprocess it, if necessary, and calculate the SVM model's gradient, loss, and accuracy for model performance validation. Besides the general functions for the training process, we defin the functions of gradient clipping and noise adding to apply differential privacy while training.

Our training process for the FL model with DP includes the following main steps:

- First step is loading the Adult dataset and transforming the boolean label column into 1 and -1 for the SVM model.
- Next, we define 50 clients, which are the individual users or devices that participate in the training process. We fix the number of clients selected for each iteration to be equal to the number of overall clients so that each training round includes updates from all the clients.
- During each iteration, the dataset is divided into multiple clients, so that each client can train the global model locally. The local training applies either mini-batch SGD or full-batch SGD, depending on the given batch size.
- To ensure that each local client's update remains differentially private, we clip gradients of the SVM model and perturb them with Gaussian noise. These steps are similar to the DP-SGD algorithm explained in Section III-F. The only difference is that the added noise and clipping threshold are adjusted based on the amount of processed data and the number of clients to be applicable to the FL model.

- After applying DP to all local solutions, we aggregate them into the global solution. The aggregated solution is calculated by averaging the local solutions. Thus, the global solution is computed at the end of each epoch, which is the updated model parameters we received from all participating clients. The global solution becomes the updated model weights for the next training iteration.
- Similar steps are repeated several times during each epoch, following the goal of having a more robust model. As real-world datasets include noise and variability, it is always better to run multiple experiments to see how the model performs during different iterations with different subsets of data. This ensures the model results are not random and are stable throughout the training process.
- The accuracy and loss metrics are calculated for training and test datasets to monitor the model performance during the training process.

Our implemented FL model with DP used similar parameter values of data privacy, clipping threshold, and delta as in NN manual implementation explained in VI-B. The model results in an accuracy of 74.94%, which is quite close to the accuracy of the with DP manual implementation's model.

E. Membership Inference Attack

To implement an MIA attack, we first delete some rows of the dataset to make the target column balanced, which becomes more important given the dataset will be further divided into several parts. We drop the column native-country for the same reason. Then, we proceed to split the dataset. 20% of the dataset is allocated for training the target model and is assigned as X_target_in and y_target_in, 10% of the dataset is used as a validation set for the target model, which is stored as X_target_val and y_target_val, 20% is set aside to be used in the testing the attack model, as datapoint that have not been part of the target model. They are referred to as X_target_out and y_target_out. The rest of the dataset is used for the shadow models. This section is divided into 4 equal parts for training 4 shadow models, while each of them is divided into 2 equal parts, when one part will be used to train the model, while the other part will be used in the training process of the attack model, as data points that have not been part of the training set. For the i'th shadow model, the respective datasets are named X_shadow_in_i, y_shadow_in_i, X_shadow_out_i, and y_shadow_out_i.

We define a model that trains a logistic regression without DP with the same architecture and hyperparameters as in the notebook for logistic regression. X_target_in, y_target_in are used for training, X_target_val, y_target_val are used for validation, while X_target_out and y_target_out are used as a test set. This method does not prohibit us from using X_target_out and y_target_out in the testing process of the attack model as data points that have not been part of the training process, as they are only used in the testing process of the target model and do not affect the learned parameters.

We go on to define and train a logistic regression model with DP with the same structure and hyperparameters as in the notebook for the logistic regression. The model is trained and tested on the same train, validation, and test datasets as the one with DP.

For the next step, we implement 4 neural networks as shadow models. Although the shadow models could have any architecture that performs the same task as the target model, we chose NNs, for their flexibility. The shadow models vary in the number of layers, the number of nodes in them, the activation functions, the optimizers, and the number of epochs.

Lastly, we define a function for the attack model, which is a small NN, similar to the shadow models. It uses shadow models to create a training set, as described in the section Threat Scenarios and Possible Attacks. Similarly, the information from the 2 target models, with and without DP, is used to create the test set of the attack model. The goal is to determine whether the attack model performs differently on the target model, depending on whether DP has been employed in it.

Prior to implementing the attack model on the target models, we tested it on the data that it had been trained on, which resulted in low-performance metrics. From here, we could conclude that the attack model did not learn. We did not proceed to test the attack model on the target models, as the results would more likely be based on random chance than logical inference.

The performance metrics of the target, shadow, and attack models can be found in the notebook called Membership Inference Attack.

The time constraints prevented us from further refining the code for MIA. We plan to address it in future work. Specifically, we plan to do more research on different MIA approaches and increase the performance metrics on the shadow models.

VII. Conclusions

In the scope of this project, we implemented a differentially private logistic regression in three ways. Firstly, we applied the DP SGD algorithm, then we created our modification of it to introduce adaptive clipping, and lastly, we utilized the diffprivlib library for training a differentially private logistic regression. Additionally, we implemented a neural network and added DP to it in two ways - manually, using the DP SGD algorithm, and with the help of the Opacus library. We proceeded to implement a membership inference attack model, which incorporated training 2 target models - with and without DP, 4 shadow NNs, and the attack model. Lastly, we manually implemented federated learning with DP on an SVM model.

In our future work, we plan to conduct further research on the attack methods and refine the current architecture.

VIII. Acknowledgements

We would like to express our gratitude to our supervisor, Elen Vardanyan, for her invaluable guidance and support, as well as the Data Science faculty of the American University of Armenia for their contribution to our academic journey and for the knowledge shared with us over the years.

REFERENCES

[1] Devaux, E. (2022, December 21). *What is Differential Privacy: definition, mechanisms, and examples*. https://www.statice.ai/post/what-is-differential-privacy-definition-mechanisms-examples#:~:text=Differentially%20private%20machine%20learning%20algorithms,make%20accurate%20predictions%20or%20decisions.

[2] Carlini et al. (2021). *Extracting Training Data from Large Language Models*. https://www.usenix.org/system/files/sec21-carlini-extracting.pdf

[3] Garfinkel, S., Abowd, J. M., & Martindale, C. (2018). *Understanding Database Reconstruction Attacks on Public Data*. https://queue.acm.org/detail.cfm?id=3295691

[4] Liu, W., Zhang, Y., Yang, H., & Meng, Q. (2023). *A Survey on Differential Privacy for Medical Data Analysis*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10257172/

[5] Zhu, T., Ye, D., Wang, W., Zhou, W., & Yu, P. S. (2022). *More Than Privacy: Applying Differential Privacy in Key Areas of Artificial Intelligence*. https://www.computer.org/csdl/journal/tk/2022/06/09158374/1m1eAPbg4JW

[6] Wei et al. (2023). *Federated Learning With Differential Privacy: Algorithms and Performance Analysis*. https://ieeexplore.ieee.org/document/9069945

[7] Banse et al. (2024, February 3). *Federated Learning with Differential Privacy*. https://arxiv.org/pdf/2402.02230

[8] Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). *Calibrating Noise to Sensitivity in Private Data Analysis*. https://link.springer.com/chapter/10.1007/11681878_14

[9] Elamurugaiyan, A. (2018). *A Brief Introduction to Differential Privacy*. https://medium.com/georgian-impact-blog/a-brief-introduction-to-differential-privacy-eacf8722283b

[10] Ponomareva, N., Hazimeh, H., Kurakin, A., Xu, Z., Denison, C., McMahan, H. B., Vassilvitskii, S., Chien, S., & Thakurta, A. (2023). *How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy*. https://arxiv.org/pdf/2303.00654.pdf

[11] Fathima, S. (2020). *Differential Privacy Definition*. https://medium.com/@shaistha24/differential-privacy-definition-bbd638106242

[12] Fathima, S. (2020, September 15). *Query "Sensitivity" types and effects on Differential Privacy Mechanism*. https://becominghuman.ai/query-sensitivity-types-and-effects-on-differential-privacy-mechanism-c94fd14b9837

[13] Kamath, G. (2020). *Lecture 4 — Intro to Differential Privacy, Part 2*. http://www.gautamkamath.com/CS860notes/lec4.pdf

[14] Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*. https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf

[15] Fathima, S. (2020, October 2). *Differential Privacy — Noise adding Mechanisms*. https://becominghuman.ai/differential-privacy-noise-adding-mechanisms-ede242dcbb2e

[16] Near, J. S., & Abuah, C. (2024). Programming Differential Privacy. *The Exponential Mechanism*. https://programming-dp.com/ch9.html

[17] Near, J. S., & Abuah, C. (2024). Programming Differential Privacy. *Properties of Differential Privacy*. https://programming-dp.com/ch4.html

[18] Fathima, S., & Pudari, R. (2021). Differential Privacy. *LOCAL VS GLOBAL DIFFERENTIAL PRIVACY*. https://blog.openmined.org/basics-local-differential-privacy-vs-global-differential-privacy/

[19] Palanisamy, B., Li, C., & Krishnamurthy, P. (2017, July 17). *Group Differential Privacy-preserving Disclosure of Multi-level Association Graphs*. https://www.sis.pitt.edu/bpalan/papers/GroupDP-ICDCS2017.pdf

[20] Wu, X., Fredrikson, M., Jha, S., & Naughton, J. A. (2016). *A Methodology for Formalizing Model-Inversion Attacks*. https://www.cs.cmu.edu/~mfredrik/papers/wfjn_csf16.pdf

[21] Srinivasan, A. V. (2019, September 7). *Stochastic Gradient Descent — Clearly Explained !!*. https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

[22] Zhang, X. et al. (2021). *Adaptive Privacy Preserving Deep Learning Algorithms for Medical Data*. https://openaccess.thecvf.com/content/WACV2021/html/Zhang_Adaptive_Privacy_Preserving_Deep_Learning_Algorithms_for_Medical_Data_WACV_2021_paper.html

[23] Shokri, R. et al. (2017). *Membership Inference Attacks Against Machine Learning Models*. https://ieeexplore.ieee.org/document/7958568

[24] Martineau, K. et al. (2022, August 24). *What is federated learning?*. https://research.ibm.com/blog/what-is-federated-learning

[25] Altexsoft. (2022, February 21). *Federated Learning: The Shift from Centralized to Distributed On-Device Model Training*. https://www.altexsoft.com/blog/federated-learning/

[26] GeeksforGeeks. (2024). *Collaborative Learning – Federated Learning*. https://www.geeksforgeeks.org/collaborative-learning-federated-learning/

[27] Truong, N. et al. (2021, November). Computers & Security. *Privacy preservation in federated learning: An insightful survey from the GDPR perspective*. https://www.sciencedirect.com/science/article/pii/S0167404821002261

[28] Hao, K. (2019, December 11). ARTIFICIAL INTELLIGENCE. *How Apple personalizes Siri without hoovering up your data*. https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/#:~:text=In%20addition%20to%20federated%20learning,a%20local%20machine%2Dlearning%20model.

[29] Devansh. (2023, October 26). *What is federated learning and why do Big Tech companies like Amazon love it?*. https://machine-learning-made-simple.medium.com/what-is-federated-learning-and-why-do-big-tech-companies-like-amazon-love-it-6cf56b3dc4b1

[30] Stojkovic, B. et al. (2022, June 14). Engineering at Meta. *Applying federated learning to protect data on mobile devices*. https://engineering.fb.com/2022/06/14/production-engineering/federated-learning-differential-privacy/

[31] Omhover, J. (2023, May 30). *Federated Learning with Azure Machine Learning: Powering Privacy-Preserving Innovation in AI*. https://techcommunity.microsoft.com/t5/ai-machine-learning-blog/federated-learning-with-azure-machine-learning-powering-privacy/ba-p/3824720

[32] Testuggine, D., & Mironov, I. (2020, August 31). *Introducing Opacus: A high-speed library for training PyTorch models with differential privacy*. https://ai.meta.com/blog/introducing-opacus-a-high-speed-library-for-training-pytorch-models-with-differential-privacy/

[33] Yousefpour et al. (2022). *Opacus: User-Friendly Differential Privacy Library in PyTorch*. https://arxiv.org/pdf/2109.12298.pdf

[34] Opacus. (2021). *Opacus: User-Friendly Differential Privacy Library in PyTorch*. https://github.com/pytorch/opacus/blob/main/opacus/accountants/analysis/gdp.py#L18

[35] Rani et al. (2022). *Support Vector Machine*. https://www.sciencedirect.com/topics/computer-science/support-vector-machine