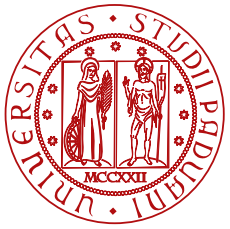


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Tesi in Ingegneria dell'Informazione

Valutazione di metodi di ϵ-Differential Privacy in Federated Learning

Laureando

Davide Brisolin

Matricola 2009735

Relatore

Prof. Giovanni Perin

Università di Padova

Anno Accademico
2022/2023

Introduzione

Negli ultimi anni, l'Intelligenza Artificiale ha acquisito sempre più notorietà e le sue applicazioni in ambiti quotidiani ricoprono sempre più maggiore importanza; esempi significativi sono certamente gli utilizzi di tool di intelligenza artificiale per ricreare immagini o testi, come ad esempio Dall-E e ChatGPT, i quali vengono addestrati utilizzando informazioni provenienti da molte fonti, come siti web, libri, articoli, forum, e molti altri, anche se la vera fonte di ispirazione siamo noi e i nostri dati sensibili: a titolo d'esempio infatti, recentemente, in Italia è stato bloccato temporaneamente il servizio di ChatGPT per *l'assenza di una base giuridica che giustifichi la raccolta e la conservazione massiccia di dati personali, allo scopo di addestrare gli algoritmi sottesi al funzionamento della piattaforma* [11].

Infatti, di pari passo, la tecnologizzazione globale che viviamo quotidianamente può portare con sé numerosi problemi di sicurezza: basti considerare gli innumerevoli dati personali che abbiamo salvati nei nostri device, come password di account bancari o altre informazioni sensibili. Più semplicemente, ogni qualvolta digitiamo una nostra preferenza ad un sondaggio o accettiamo i "cookie" dei siti che visitiamo regolarmente, stiamo offrendo, a fini di marketing, le nostre abitudini a grandi aziende che devono avere il compito di mantenere segreti i dati che trasmettiamo: in una parola devono garantire la privacy.

In questa tesi mostreremo una tecnica utilizzata per far sì che l'addestramento di una macchina di Intelligenza Artificiale non necessiti in maniera assoluta di uno scambio diretto di dati tra gli utenti e la macchina (Federated Learning). A questa tecnica assoceremo un ulteriore algoritmo per creare ancora più sicurezza nello scambio di dati, che deriva dalla definizione di ϵ-differential privacy. Valuteremo infine queste due tecniche appaiate per stabilire la reale accuratezza dell'analisi effettuata, in termini di efficienza del modello e di privacy ottenuta.

Indice

1 Machine Learning

1.1 Definizione e obiettivi del ML

Il Machine Learning (ML) è una branca dell'Intelligenza Artificiale (AI) che si focalizza, come suggerisce il nome stesso, nello sviluppare algoritmi e modelli utilizzabili dai sistemi informatici per generare soluzioni a problemi sempre nuovi: il concetto di apprendimento infatti si basa sull'acquisizione di esperienze del mondo che successivamente possano essere convertite in conoscenze da applicare in campi non noti [12].

L'apprendimento si articola dunque in varie componenti:

- **Dati di Input:**
  - **Dominio X:** è l'insieme arbitrario dei dati iniziali che vorremmo classificare per poi produrre una risposta in output.
    Generalmente, questi punti del dominio saranno rappresentati da un vettore di caratteristiche (*vector of features*);
  - **Etichette Y:** l'insieme di tutte le possibili risposte per cui vogliamo addestrare il nostro sistema informatico;
  - **Dati di addestramento S:** $S = ((x_1, y_1)...(x_m, y_m))$ è una sequenza finita di coppie in X x Y: di fatto è l'input a cui la macchina deve poter aver accesso per iniziare il processo di addestramento.

- **Dati di Output:** la macchina deve possedere una regola di predizione $h{:}X \rightarrow Y$ che permette di mappare un elemento in X in un suo elemento in Y. La funzione h viene solitamente chiamata ipotesi o classificatore. La funzione h potrà poi essere usata per classificare un elemento x, sconosciuto, nel suo valore y.

- **Target Function:** è la funzione bersaglio, sconosciuta, che permette di classificare correttamente ogni valore di x nel suo esatto valore y; formalmente, $f: X \longrightarrow Y$ tale che $y_i = f(x_i)$ per ogni valore di i.
- **Errore della predizione:** viene espresso come la probabilità che il classificatore non predica correttamente il valore della etichetta y per un generico dato x generato da una distribuzione di probabilità casuale D.

  Formalmente è definita come:

$$L_{D,f}(h) = \mathop{\mathbb{P}}_{x \sim D}[h(x) \neq f(x)] \tag{1.1}$$

Il compito di un algoritmo di apprendimento automatico consiste nel trovare la funzione h più simile possibile alla funzione target f, in modo da poter classificare correttamente ogni valore x che viene fornito al sistema, con la minor probabilità di errore possibile.

1.1.1 Fasi di un processo di ML

La creazione di un modello di Machine Learning può essere riassunta in due punti fondamentali:

- **Addestramento del modello:** il processo di addestramento di un modello di Machine Learning consiste nel trovare i parametri ottimali che possono descrivere accuratamente la relazione tra X e Y. Per raggiungere questo obiettivo, è necessario un set di dati di allenamento. Quindi, successivamente, viene adottata una funzione di perdita per quantificare la differenza tra due output, ovvero quello reale e quello previsto. L'obiettivo dell'addestramento di un modello è ridurre al minimo questa funzione di perdita.
- **Test del modello:** dopo che l'addestramento del modello è stato completato, ottenendo i parametri ottimali, dato un generico input è possibile calcolarne l'output attraverso l'utilizzo dei parametri ottenuti in precedenza. Possiamo inoltre calcolare l'accuratezza della previsione del modello su un set di dati di test per misurare le prestazioni del modello stesso.

1.2 La Regressione

La Regressione è una tecnica di analisi statistica che permette di indagare relazioni di dipendenza di tipo causa-effetto tra variabili quantitative. Questa tecnica viene associata spesso a modelli di ML perché, attraverso la relazione che intercorre tra due o più variabili, è possibile creare un modello di predizione valido.

Esistono diverse tipologie di regressione utilizzabili dagli algoritmi di Machine Learning; in questo capitolo ne esaminiamo due: Regressione Lineare e Regressione Logistica.

Prima di passare oltre, definiamo la **classe delle funzioni affini**, una grandezza che tornerà utile successivamente. Essa viene definita come:

$$L_d = \{h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{1.2}$$

dove

$$h_{\mathbf{w},b} = \langle \mathbf{w}, x \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b \tag{1.3}$$

in cui $\langle \mathbf{w}, x \rangle$ rappresenta il prodotto scalare di due vettori e $\mathbf{w}$ è il vettore che parametrizza la funzione h; d'ora in poi, per semplicità, indicheremo $\mathbf{w}$ come il vettore che rappresenta i parametri del modello e, di conseguenza, il modello.

1.2.1 La Regressione Lineare

La Regressione Lineare è un semplice metodo statistico per modellare la relazione che intercorre tra variabili indipendenti e alcune variabili reali in output. In questo caso, quindi, la Regressione Lineare si occupa di trovare la migliore funzione lineare h: $\mathbb{R}^d \to \mathbb{R}$ che descrive al meglio la relazione tra le variabili in esame. In particolare la classe di ipotesi dei predittori di regressione lineare corrisponde alla classe delle funzioni affini descritta precedentemente in (1.2).

Per identificare la bontà di un modello lineare, è necessario definire di quanto la funzione ipotesi h si discosta dalla funzione target sconosciuta: da qui deriva la misura del *Mean Square Error*, definita come:

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 \tag{1.4}$$

L'obiettivo, per trovare il miglior modello possibile, è quello di minimizzare la grandezza appena descritta: in altre parole si tratta di trovare quella funzione h che minimizza la distanza tra i valori della predizione e i valori reali, sconosciuti, della funzione bersaglio f.

1.2.2 La Regressione Logistica

La Regressione Logistica è la seconda tipologia di regressione usata nel Machine Learning che verrà trattato in questo capitolo. A differenza dei modelli utilizzati nella Regressione Lineare (impiegati per identificare la relazione tra una variabile dipendente continua e una o più variabili indipendenti), la Regressione Logistica viene solitamente utilizzata per fare una previsione circa una variabile categoriale - true o false, sì o no, 1 o 0, ... - rispetto ad una continua. Inoltre, la Regressione Logistica viene utilizzata anche per obiettivi di classificazione: possiamo infatti interpretare *h(x)* come la probabilità che l'output di un dato valore x sia 1.

La classe di ipotesi per la Regressione Logistica si ottiene con la composizione di una funzione sigmoidea $\phi_{\text{sig}} : \mathbb{R} \to [0, 1]$ con la classe di funzioni lineari L_d. In particolare la funzione sigmoidea viene chiamata **funzione logistica** ed è definita come:

$$\phi_{\text{sig}}(z) = \frac{1}{1 + exp(-z)} \tag{1.5}$$

Quindi la classe di ipotesi per la Regressione Logistica è:

$$H_{\text{sig}} = \phi_{\text{sig}} \circ L_{\text{d}} = \{\mathbf{x} \mapsto \phi_{\text{sig}}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathbb{R}^d\} \tag{1.6}$$

Ora non resta che definire la funzione che ci permetterà di valutare l'errore della predizione: nel caso della Regressione Logistica, il problema si traduce nel quantificare l'errore nella predizione di una certa $h_{\mathbf{w}} \in [0, 1]$ rispetto al reale valore di $y = \{\pm 1\}$.

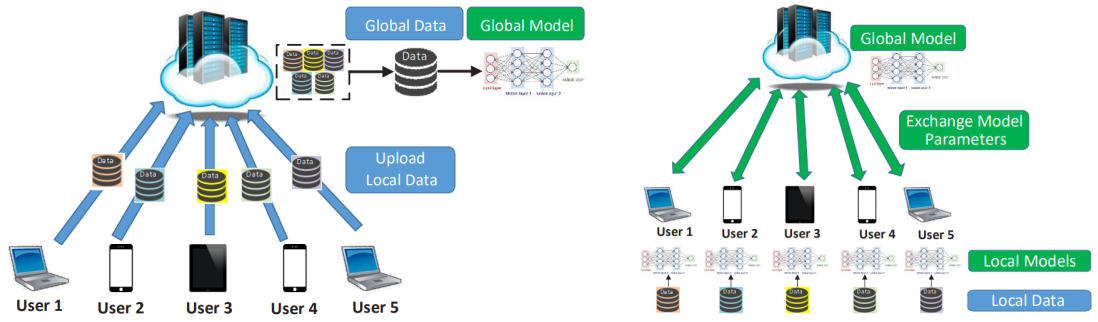
Per quanto riguardava la Regressione Lineare, la misura dell'errore della predizione veniva calcolato con il modello dei minimi quadrati, descritto in precedenza. Non avendo la stessa distribuzione di dati, nella Regressione Logistica, il metodo utilizzato in precedenza non può essere applicato: diversamente viene applicato il metodo della **maggior verosimiglianza**: questo metodo consiste nel massimizzare la funzione di verosimiglianza, definita in base alla probabilità di osservare una data realizzazione campionaria, rispetto ai valori assunti dai parametri dell'oggetto di stima.

1.3 Tipologie di ML

Come si può notare dalla Figura 1.1, i modelli di Machine Learning possono essere distinti in due categorie, a seconda della propria architettura [8]:

- **Apprendimento Centralizzato:** I dati di allenamento sono centralizzati in una macchina o in un *data center*, la quale si occupa di addestrare e ospitare i modelli. Ad esempio, un ricercatore potrebbe utilizzare una piattaforma cloud, per ospitare set di dati e addestrare un modello di intelligenza artificiale basato su di essi. Va da sé che la disponibilità di tutti i dati in un metodo così centralizzato porta ad un'elevata efficienza e precisione. Tuttavia, poiché l'operatore centralizzato ha accesso diretto ai dati sensibili, la privacy dell'utente potrebbe essere violata agevolmente.
- **Apprendimento distribuito:** L'apprendimento centralizzato spesso non è una buona opzione per diversi motivi:
  - i dati sono intrinsecamente distribuiti in alcuni scenari;
  - i dati sono troppo grandi per essere archiviati in un unico file macchina;
  - gli utenti non sono disposti a condividere dati "grezzi" (cioè dati sensibili da utilizzare per addestrare la macchina);
  - gli utenti vorrebbero addestrare la macchina con diverse istanze, per ottenere una migliore precisione della previsione.

  Ecco che, un modello di ML può essere costruito utilizzando un apprendimento distribuito. In generale, l'apprendimento distribuito si ottiene avendo a disposizione uno scenario di dati di addestramento distribuiti e un server centralizzato. Esistono varie tipologie di addestramento distribuito, ad esempio il Featured Learning, che vedremo approfondito nel Capitolo 2.



(a) centralized learning (b) distributed learning

Figura 1.1: Tipologie di ML

2

Federated Learning

Il Federated Learning (FL) è una tipologia di architettura di Machine Learning che si basa sull'apprendimento distribuito - già descritto in 1.3.

Il Federated Learning si pone come alternativa alla classica concezione di modelli di Machine Learning, per cui debba esistere un server centralizzato a cui inviare tutti i dati per l'addestramento. Come suggerisce il nome, infatti, l'apprendimento del modello di ML è risolto da una federazione libera di dispositivi partecipanti (che chiamiamo *clients*) che sono coordinati da un server centrale: in questo modo ogni *client* avrà a disposizione un dataset da poter utilizzare per l'addestramento che non condividerà mai con il server centrale.

Utilizzando questa tecnica, infatti, i dati di addestramento vengono lasciati distribuiti nei vari dispositivi che localmente svolgeranno un addestramento per ottenere i parametri del modello; una volta ottenuti i parametri, questi verranno aggregati in un server centralizzato che li aggiornerà e restituirà ai singoli dispositivi, i quali si occuperanno a loro volta di aggiornare il proprio modello in locale [10].

Nella sezione 2.2 verranno mostrate in maniera più accurata le fasi di cui si compone un classico meccanismo di Federated Learning:

1. **Addestramento locale**
2. **Aggregazione del modello**
3. **Trasmissione dei parametri**
4. **Aggiornamento dei parametri**

2.1 I vantaggi di Privacy nel FL

Utilizzando l'approccio descritto precedentemente, è possibile ottenere molti vantaggi, soprattutto nell'ambito della privacy. Infatti, il classico scambio di dati tra client e server viene eliminato, rendendo più sicuri i nostri dati personali che altrimenti avrebbero corso il rischio di essere intercettati; al contrario, addestrare localmente un modello di ML e inviare successivamente i parametri ottenuti, crea una maggior sicurezza in quanto è praticamente impossibile risalire ai dati forniti nel dataset locale.

Le informazioni trasmesse per l'apprendimento federato sono i minimi aggiornamenti necessari per migliorare un particolare modello (naturalmente, la forza del vantaggio in termini di privacy dipende dal contenuto degli aggiornamenti). Inoltre, gli aggiornamenti stessi possono (e dovrebbero) essere effimeri.

Ciò che verrà trasmesso non conterrà mai più informazione di un dataset completo, ma anzi generalmente ne conterrà molto meno.

Infine, l'origine degli aggiornamenti non è necessaria per l'aggregazione dei parametri del modello, quindi gli aggiornamenti possono essere trasmessi senza identificazione dei meta-dati.

2.2 Fasi del Federated Learning

Precedentemente sono state identificate le fasi principali che caratterizzano un classico meccanismo di Federated Learning.

Consideriamo un sistema di FL in cui sia presente un server centrale S e N clients. Sia D_i un database locale contenuto nel client C_i, dove $i \in \{1, 2, \dots N\}$, come quello che in Figura 2.1.

Successivamente verranno mostrate in maniera più approfondita come si articolano e come vengono sviluppate queste quattro fasi [15].

2.2.1 Addestramento locale

L'addestramento locale avviene internamente ad ogni client C_i, i quali avranno a disposizione un database di addestramento D_i. L'addestramento può avvenire utilizzando vari modelli di regressione, già descritti nel Capitolo 1. Nel

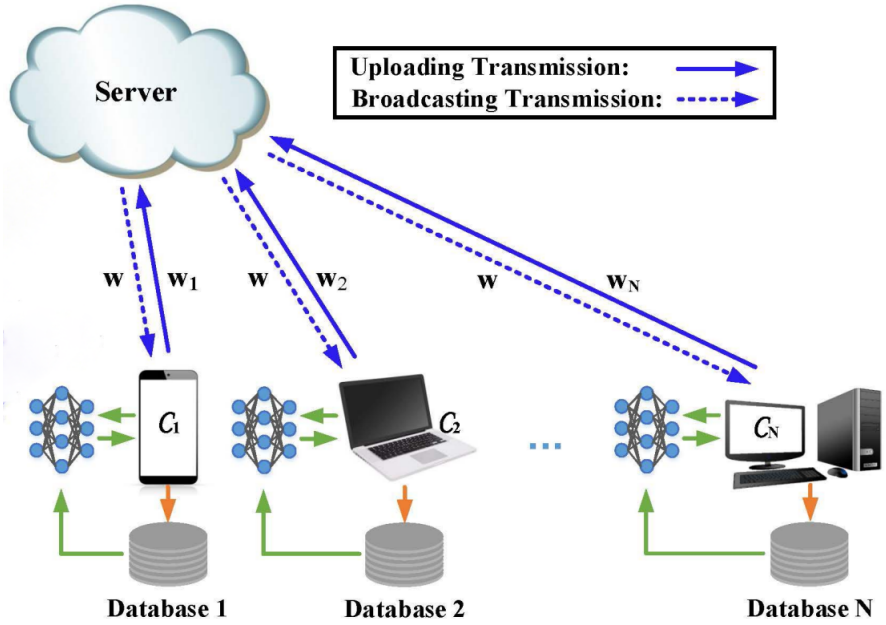


Figura 2.1: Sistema di Federated Learning

nostro caso verrà utilizzato un modello di Regressione Logistica, il cui funzionamento è stato presentato in 1.2.2. Il risultato dell'addestramento produrrà un vettore $\mathbf{w}_i$ che rappresenta i parametri ottimi del modello - che quindi minimizzano la funzione di perdita - ottenuti dall'addestramento locale.

2.2.2 Aggregazione del modello

L'aggregazione del modello è la seconda fase in cui si articola il Federated Learning. In questo stadio, i singoli clients C_i inviano al server centralizzato S i propri parametri $\mathbf{w}_i$ ottenuti in seguito al loro addestramento locale.

Il server ha il compito di raccogliere questi dati e aggregarli, in modo tale da formare un unico vettore dei parametri, $\mathbf{w}$, definito come:

$$\mathbf{w} = \sum_{i=1}^{N} p_i \cdot \mathbf{w}_i \tag{2.1}$$

dove $p_i = \frac{|D_i|}{|D|} \geq 0$ in cui $|D| = \sum_{i=1}^{N} |D_i|$ rappresenta la dimensione totale di tutti i campioni di dati. Di conseguenza, si ottiene una proprietà sui pesi p_i tale per cui $\sum_{i=1}^{N} p_i = 1$.

Una volta che i dati vengono aggregati in un unico vettore $\mathbf{w}$, si procede all'ottimizzazione di questi parametri, in base alla funzione di perdita adottata;

in generale per trovare il vettore ottimo dei parametri **w*** si utilizza le seguente equazione:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} p_i F_i(\mathbf{w}, D_i) \tag{2.2}$$

dove $F_i(\cdot)$ è lo locale funzione di perdita del client i-esimo.

2.2.3 Trasmissione dei parametri

Una volta ottenuto il vettore ottimo dei parametri **w***, il server S si occupa di trasmetterlo ai vari client C_i, in broadcast, in modo che tutti i client connessi al server possano ricevere il nuovo vettore dei parametri che andranno poi ad utilizzare nella fase successiva, per aggiornare i propri parametri ottimi locali. In questo stadio, può svolgersi il test dell'efficienza del modello rispetto ad un set di dati utilizzabili per questo scopo.

2.2.4 Aggiornamento dei parametri

Tutti i client, una volta raggiunti dal messaggio broadcast inviato dal server S contenente il vettore **w***, aggiornano i propri parametri con quelli ricevuti dal server e, opzionalmente, testano la performance del modello aggiornato.

In seguito ad un numero sufficiente di scambi di aggiornamenti tra il server e i suoi clients associati, la soluzione del problema di ottimizzazione, descritta in 2.2, converge all'unico modello di addestramento ottimo globale.

3

Differential Privacy

La privacy dei dati personali è un tema che sempre più sentiamo vicino in questo periodo storico: infatti con il termine privacy - che può essere tradotto con riservatezza - si intende il diritto alla riservatezza della vita privata di una persona, vita privata che ormai viene riversata in maniera massiva in network globali, e, in maniera particolare, nella grande rete dell'Internet. Ecco che diventa di primaria importanza garantire la privacy anche in questo contesto, proteggendo e mantenendo segreti i dati personali più sensibili.

Di seguito, in questo capitolo, valuteremo il concetto di privacy nel Machine Learning e introdurremo una tecnica per salvaguardare i nostri dati personali durante un processo di Machine Learning, la Differential Privacy.

3.1 Privacy e Machine Learning

Come abbiamo trattato nel Capitolo 1, gli algoritmi di Machine Learning si basano sulla raccolta di alcuni dati iniziali, provenienti dall'esperienza, chiamati dati di addestramento, che vengono utilizzati per addestrare una macchina in modo che essa possa produrre una risposta ad un problema, e che la risposta sia la più corretta possibile.

Ecco che in questo processo avvengono scambi di informazioni continui tra l'utente che vuole comunicare con la macchina e la macchina stessa, che è in grado di produrre, e di seguito comunicare, una risposta per l'utente. In particolare, se volessimo creare un algoritmo di ML che preveda lo scambio

di informazioni utili per il suo addestramento, ma sensibili per gli utenti - ad esempio dati sanitari o bancari -, queste informazioni devono poter essere scambiate senza alcun tipo di rischio, mantenendo l'anonimato e la riservatezza.

Una tecnica molto utilizzata per la privacy dei dati personali consiste nel modificarli, aggiungendo del "rumore" ai dati stessi; è questa l'idea che sta alla base della Differential Privacy.

3.2 Differential Privacy: una promessa

La Differential Privacy (DP) viene originariamente connotata come una promessa, che viene fatta da parte del titolare nei confronti dei soggetti che condividono i loro dati personali; la promessa è che l'utente non dovrà essere influenzato, negativamente o in altro modo, consentendo ai suoi dati di essere utilizzati in qualsiasi studio o analisi [5]. Da questa promessa derivano gli obiettivi e la definizione della DP: nel passaggio di informazioni tra due utenti, è vivo il rischio che queste possano essere intercettate e finire nelle mani sbagliate; la DP opera creando degli algoritmi che permettano l'aggiunta di rumore (*noise*) ai dati prima di essere inviati, in modo da mascherare il reale valore che essi assumono.

3.2.1 Esempio di randomizzazione nella DP

In questa sezione, viene mostrato un esempio molto semplice di come i dati possano essere modificati [5]. Poniamo che venga posta una domanda ad una popolazione, la cui risposta può essere solamente **SI** oppure **NO**. Invece di inviare direttamente la risposta, si lancia una moneta e si valuta il risultato:

- **Esce croce:** si comunica in modo corretto la risposta;
- **Esce testa:** si lancia una seconda moneta:
  - **Esce testa:** si comunica la risposta SI;
  - **Esce croce:** si comunica la risposta NO.

In tutto questo processo viene garantito inoltre l'anonimato, in quanto diventa superflua la relazione tra risposta e utente che l'ha fornita. Ecco che un utente malintenzionato che riesce a carpire queste informazioni, non è in grado

di interpretarle in quanto non è a conoscenza di come quella risposta è stata generata, e tantomeno a chi si riferisce: infatti il lancio della monetina costituisce la chiave di volta dell'intero sistema: è l'elemento randomico.

Questo meccanismo lascia emergere un ulteriore problema: il rumore aggiunto potrebbe ridurre l'attendibilità dell'indagine statistica qualora il numero di dati raccolti fosse in somma inconsistente: infatti, dall'esempio precedente, è chiaro che le risposte avrebbero una affidabilità del 75% in caso di moneta non truccata; se invece si truccasse la moneta favorendo l'uscita di testa o di croce, anche l'affidabilità del risultato sarebbe modificata. Ecco che la DP raggiunge un ottimo *trade-off* tra accuratezza e privacy solo nel caso di databases molto ampi.

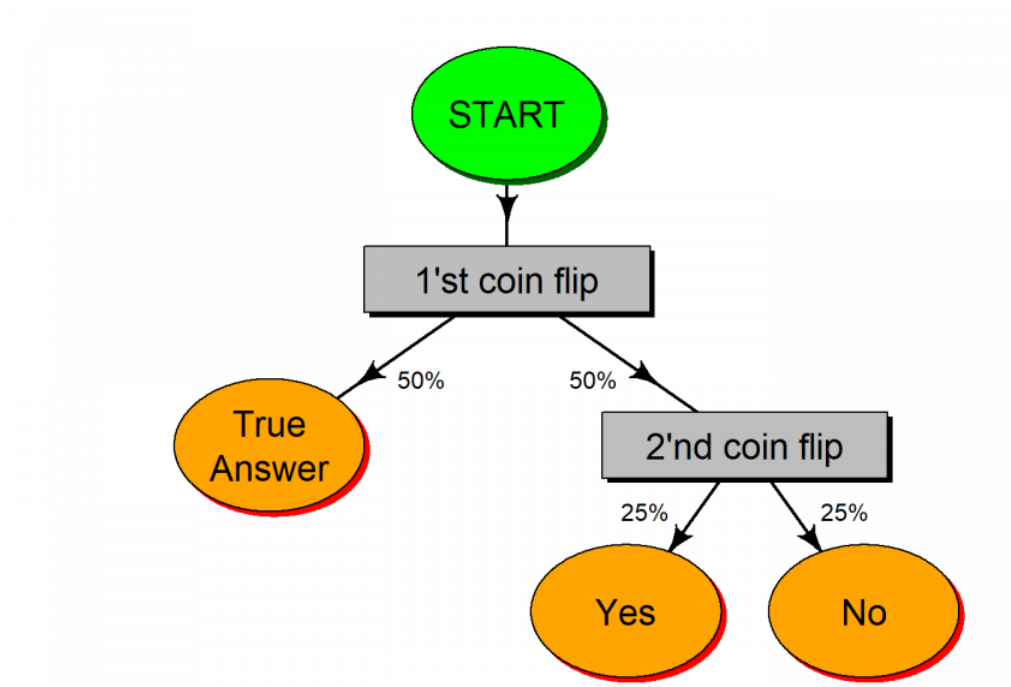


Figura 3.1: Esempio di Differential Privacy

3.3 Formalizzare la Differential Privacy

Siamo ora in grado di dare una definizione formale di Differential Privacy.
Differential Privacy. Un algoritmo randomizzato $\mathcal{M}$, con dominio $\mathbb{N}^{|\mathcal{X}|}$, è (ϵ, δ)-*differentially private* se per ogni $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ e per ogni coppia di dataset adiacenti $x, y \in \mathbb{N}^{|\mathcal{X}|}$ tali che $|x - y| \leq 1$, vale:

$$\mathbb{P}[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\epsilon)\, \mathbb{P}[\mathcal{M}(y) \in \mathcal{S}] + \delta \tag{3.1}$$

dove due dataset vengono detti adiacenti se differiscono al massimo per una riga, che corrisponde ai dati di un individuo [3].

Nella definizione 3.1 compaiono due valori, ϵ e δ, che sono dei parametri utili alla definizione di Differential Privacy. In particolare, essi rappresentano:

- δ : è un termine che introduce una maggior differenza tra i due dataset x e y; per semplicità d'ora in poi considereremo $\delta = 0$ e quindi parleremo di $(\epsilon, 0)$-*differential privacy* o più semplicemente ϵ-*differential privacy*;

- ϵ : è la misura della *privacy loss*, ossia quantifica a che livello la privacy viene raggiunta o perduta.

Sappiamo che quando ϵ è piccola, $\exp(\epsilon) \approx 1 + \epsilon$; in questa forma è facile valutare che se $\epsilon = 0$ significa che le due probabilità devono essere le stesse: quindi avendo 0 *privacy loss*, si hanno inoltre 0 informazioni sul dataset. Ma se ϵ è molto piccola, e diversa da zero, possiamo affermare che se un evento è molto improbabile che si verifichi quando non appartiene a un dataset (y) allora è ancora molto improbabile che si verifichi quando invece appartiene al dataset (x).

3.4 Come ottenere la Differential Privacy

Abbiamo descritto la DP come una promessa che viene raggiunta tramite degli algoritmi che operano aggiungendo del rumore ai dati prima di essere inviati. Ora ci chiediamo che tipologia di rumore possa venire aggiunto in modo tale da rendere la definizione 3.1 valida per il dataset in questione.

I due rumori che solitamente vengono utilizzati per preservare la DP sono il rumore Gaussiano e il rumore Laplaciano.

3.4.1 Il rumore Laplaciano

La distribuzione di Laplace è una distribuzione di probabilità continua che prende il nome dal matematico Pierre-Simon de Laplace. Come per la distribuzione Gaussiana, il rumore Laplaciano è un rumore, la cui distribuzione statistica corrisponde ad una Laplaciana, che viene aggiunto ad una distribuzione di dati che si intende modificare [2].

La distribuzione di Laplace è definita dalla seguente equazione:

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right) \tag{3.2}$$

Definiremo ora il Meccanismo di Laplace utilizzato per l'aggiunta del rumore sopracitato. Come suggerisce il nome, il meccanismo di Laplace calcolerà semplicemente f e perturberà ciascuna coordinata con il rumore ricavato dalla distribuzione di Laplace descritta in 3.2. Data una qualsiasi funzione $f : \mathbb{N}^{|\mathcal{X}|} \mapsto \mathbb{R}^k$, il Meccanismo di Laplace è definito come

$$M_L(x, f(\cdot), \varepsilon) = f(x) + (Y_1, \ldots, Y_k) \tag{3.3}$$

dove Y_i sono delle variabili indipendenti e identicamente distribuite (i.i.d.) ricavate da $\text{Lap}(\Delta f/\epsilon)$.

Come afferma il teorema 3.6 contenuto in [5], è provato che il Meccanismo di Laplace garantisce che venga rispettata la *ϵ-differential privacy*.

3.4.2 Il rumore Gaussiano

La distribuzione normale (o distribuzione di Gauss dal nome del matematico tedesco Carl Friedrich Gauss), nella teoria della probabilità, è una distribuzione di probabilità che viene utilizzata molto spesso. Il rumore gaussiano, analogamente a quanto descritto nella sezione precedente, è un rumore, la cui distribuzione statistica corrisponde ad una gaussiana, che viene aggiunto ad una distribuzione di dati che si intende modificare.

La distribuzione di Gauss è definita dalla seguente equazione:

$$f(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.4}$$

Come in precedenza, il Meccanismo di Gauss è molto simile al Meccanismo di Laplace per l'aggiunta di rumore: questo metodo, che permette di aggiungere rumore ad ogni componente della funzione, garantisce infatti che venga rispettata la (ϵ, δ)-*differential privacy* .

3.5 Differential Privacy applicata al FL

La Differential Privacy, descritta precedentemente, sta alla base di un utilissimo meccanismo per cui i nostri dati personali vengono mascherati e modificati, aggiungendo ad essi del rumore, ad esempio del rumore laplaciano o gaussiano.

Questo meccanismo può essere invece utilizzato in modo più competitivo affiancandolo al Federated Learning: infatti, in questo modo, il rumore aggiuntivo non viene applicato ai dati da trasferire, ma invece viene sommato al vettore che contiene i parametri del modello; l'aggiunta del rumore va effettuata per ogni client C_i, modificando il valore di $\mathbf{w}_i$ in questo modo:

$$\tilde{\mathbf{w}}_i = \mathbf{w}_i + \mathbf{n}_i \tag{3.5}$$

dove $\tilde{\mathbf{w}}_i$ indica il nuovo vettore dei parametri "rumoroso" per ogni client C_i e il vettore $\mathbf{n}_i$ indica il rumore aggiunto per garantire la Differential Privacy. Dopo aver aggiunto il rumore, è possibile terminare la prima fase del FL inviando il vettore $\tilde{\mathbf{w}}_i$ al server S.

Nella seconda fase di aggregazione del modello, allora, il vettore unico dei parametri $\mathbf{w}$ viene calcolato utilizzando i valori ricevuti da i client; ecco che $\mathbf{w}$ è definito ora in questo modo:

$$\mathbf{w} = \sum_{i=1}^{N} p_i \cdot \tilde{\mathbf{w}}_i \tag{3.6}$$

Prima della terza fase del FD, in cui il Server S trasmette i parametri in broadcast a tutti i client ad esso connessi, si procederà ulteriormente ad aggiungere rumore al vettore $\mathbf{w}$, con la stessa modalità descritta in 3.5; ai client arriverà dunque un vettore $\tilde{\mathbf{w}}$ descritto come:

$$\tilde{\mathbf{w}} = \mathbf{w} + \mathbf{n} \tag{3.7}$$

dove il vettore $\mathbf{n}$ indica il rumore aggiunto per garantire la Differential Privacy.

Successivamente, gli aggiornamenti dei parametri, effettuati da parte dei clients utilizzando i loro rispettivi dataset, avverrà considerando il vettore "rumoroso" $\tilde{\mathbf{w}}$ ottenuto dal server.

I vantaggi dell'applicazione della Differential Privacy al Federated Learning sono principalmente legati alla privacy dei dati sensibili che sono presenti nel dataset dei clients C_i: utilizzando questi due metodi insieme, è possibile, infatti, evitare di scambiare direttamente dati personali tra client e server, inviando solo i parametri del modello di Machine Learning addestrato in locale; inoltre, a questi parametri viene aggiunto del rumore (laplaciano o gaussiano) che li rende meno vulnerabili ad attacchi da parte di malintenzionati.

D'altro canto, un eccessivo rumore potrebbe provocare, specialmente per dataset ridotti, una riduzione drastica dell'efficienza del modello di Machine Learning che viene adottato. E' dunque necessario creare un giusto accordo tra efficienza del modello e privacy dei dati.

3.5.1 Contributi rilevanti

Il Federated Learning offre un framework di apprendimento automatico collaborativo e sicuro per diversi dispositivi, senza condividere i loro dati privati e, per questo motivo, vi è una ricerca estensiva in corso in questo campo. Il Federated Learning può essere applicato in diversi settori come il trasporto o la sanità: infatti, il FL è stato utilizzato per prevedere le future richieste di ossigeno dei pazienti sintomatici affetti da COVID-19 utilizzando dati di segni vitali, dati di laboratorio e radiografie del torace come input [4].

Anche se i framework di FL offrono una migliore garanzia della privacy rispetto ad altri framework di machine learning, sono comunque vulnerabili a diversi attacchi: l'aggiunta di rumore è un fattore che garantisce una ancor maggiore sicurezza [9]. Se garantire la privacy diventa una priorità, di pari passo non si può trascurare l'efficienza dei risultati: sono stati condotti molti test per valutare l'effettivo grado di efficienza di modelli di ϵ-DP in FL [6] [13] [16] [17] [7] [10]; da queste analisi è possibile dedurre alcuni comportamenti rilevanti e comuni, riportati in seguito.

Per modelli con un numero di clients sempre maggiori , l'accuratezza del modello resta significativamente sotto la performance che si ottiene senza l'utilizzo

dei concetti di DP; in ogni caso la performance è comunque migliore di quanto si otterrebbe se i singoli client riuscirebbero ad ottenere utilizzando solo i loro dati di addestramento. Con un elevato numero di client che partecipano all'addestramento, il modello, in generale, presenta una maggior sicurezza; avendo meno client a disposizione, questi però avrebbero un set di dati maggiori, che potrebbe portare ad una maggior accuratezza del modello globale.

Per distribuzioni di dati non-IID (indipendenti e identicamente distribuiti), la qualità dell'addestramento del modello degrada se ciascuno dei dispositivi periferici (ossia i client) vede una distribuzione unica dei dati; come soluzione, viene proposta una strategia per migliorare l'addestramento su dati non IID, creando un piccolo subset di dati condivisi globalmente tra tutti i dispositivi periferici, in modo da garantire una performance del modello maggiore.

4

Valutazione sperimentale

In questa sezione si procederà all'analisi sperimentale di un sistema di Federated Learning utilizzando un dataset conosciuto [1], attraverso un linguaggio di programmazione come Python [14] . Si rendono note le modalità operative che hanno portato alla creazione del sistema in questione e alla verifica sperimentale di quanto descritto nei capitoli precedenti.

4.1 Caratterizzazione del Dataset

Nell'analisi sperimentale che segue, viene utilizzato il dataset denominato **Maternal Health Risk**, un dataset noto e impiegato nell'ambito dell'apprendimento automatico come esempio di classificazione statistica. Esso consiste in 1013 istanze, classificate secondo tre categorie:

- Alto livello di rischio (*high risk*)
- Medio livello di rischio (*mid risk*)
- Basso livello di rischio (*low risk*)

Le classi delle istanze indicano l'intensità del rischio di mortalità materna durante una gravidanza.

Le sei variabili considerate sono:

- l'età di una donna durante la gravidanza (*Age*)
- valore superiore della pressione sanguigna in mmHg (*SystolicBP*)
- valore inferiore della pressione sanguigna in mmHg (*DiastolicBP*)

- livello di glucosio nel sangue in mmol/L (*BS*)
- temperatura corporea in Fahrenheit (*BodyTemp*)
- frequenza cardiaca a riposo (*HeartRate*)

Si riportano di seguito una rappresentazione del dataset descritto precedentemente, in modo tale da poter verificare le relazioni che intercorrono tra le variabili utilizzate per la classificazione dell'intensità del rischio durante una gravidanza.

In particolare, in Figura 4.1 viene riportato il *boxplot* del dataset, ossia una rappresentazione grafica utilizzata per descrivere la distribuzione di un campione (nel nostro caso, il livello di rischio) rispetto ad una variabile (nel nostro caso, ognuna delle 6 variabili descritte in precedenza).

Come si può notare dal grafico, le ultime due variabili non permettono una distinzione netta tra i 3 livelli di rischio; per semplicità, in questa analisi, si decide di non prenderle in considerazione e, di conseguenza, di restringere il dataset.

Dai grafici riportati, è chiaro che il livello di rischio maggiore è il più riconoscibile, in quanto si discosta in maniera netta rispetto alle altre due classi, soprattutto se vengono considerati le variabili riguardanti la pressione massima del sangue e il livello di glucosio nel sangue; le altre due classi, invece, spesso si confondono l'un l'altra e non sembra facile trovare una netta distinzione tra di esse rispetto alle variabili in gioco. Vedremo nelle sezioni successive come questa caratteristica del dataset andrà a condizionare l'accuratezza del modello globale.

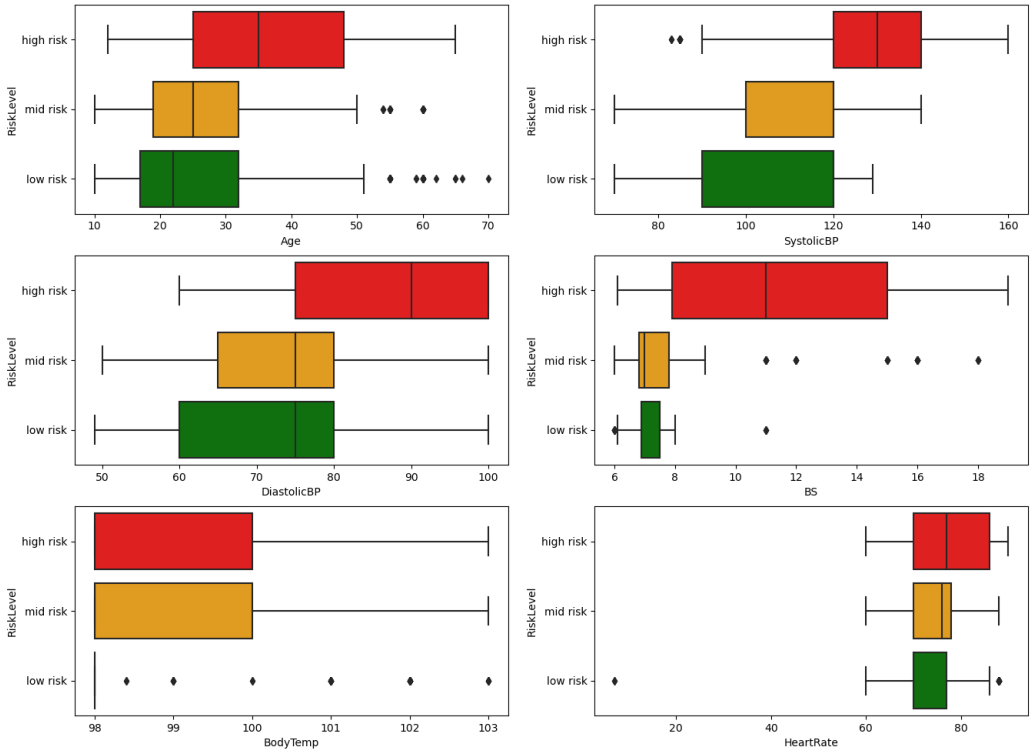


Figura 4.1: Boxplot di Maternal Health Risk

4.1.1 Partizione del Dataset

Inizialmente verrà ricreato un sistema di Federated Learning con un numero arbitrario di *client* e un singolo *server* centralizzato e condiviso tra tutti i *client*: in questo modo sarà possibile ottenere un canale di comunicazione tra ogni *client* e il *server*.

Una volta operata questa divisione tra *client* e *server* , il dataset originale verrà partizionato in due sotto-dataset, in modo tale da dare la possibilità ai *client* di sviluppare un addestramento dei dati, e al *server* di sviluppare un'analisi di accuratezza del modello aggregato ottenuto. Il sotto-dataset assegnato ai *client* verrà poi successivamente ulteriormente partizionato tra tutti i client presenti.

Nella Figura 4.2 viene riportato uno schema che rappresenta la divisione operata del dataset rispetto ad un numero arbitrario di *client* - che verrà indicato con la variabile *num_client* - e il singolo *server*.

Ogni dataset assegnato ai *client* verrà poi ulteriormente diviso in due parti, rappresentanti il set di dati da usare per l'addestramento (*Training Set*) e un set di dati da usare per la validazione (*Validation Set*), dedicato alla verifica della bontà dell'addestramento.

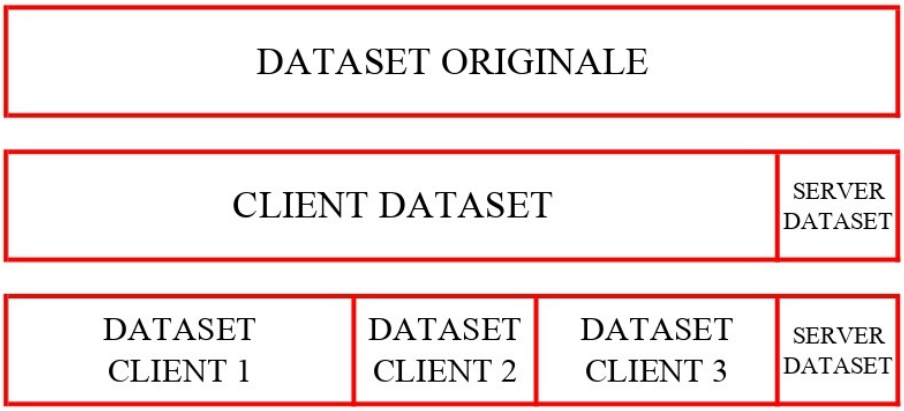


Figura 4.2: Partizione del dataset con *num_client* = 3

Da notare che la dimensione dei dataset assegnati ad ogni *client* non è uniforme tra di essi, ma può variare in maniera casuale; come abbiamo descritto nel Capitolo 2, le dimensioni dei vari dataset dei *client* influiscono nel valore dei vari pesi p_i assegnati ad ogni *client*, e, di conseguenza, anche nel vettore dei parametri ottenuto dall'aggregazione del modello da parte del server, come riportato in (2.1).

Dopo aver effettuato questa partizione, si può procedere alla valutazione e all'analisi di un sistema di Federated Learning. Vengono proposte tre tipologie di analisi: la prima riguarda un modello di Federated Learning classico, mentre la seconda e la terza sviluppano un modello di Federated Learning a cui viene applicato del rumore, rispettivamente di tipo laplaciano e gaussiano, in accordo con la definizione di ϵ-Differential Privacy.

4.2 Analisi sperimentale: Apprendimento centralizzato

Si propone, in prima analisi, il test di accuratezza di un modello di Machine Learning addestrato utilizzando un classico metodo di regressione logistica.

Si riportano i valori dei coefficienti e dell'intercetta del modello in Figura 4.3. Inoltre, si riportano la percentuale di efficienza del modello rispetto ai dati di test e la matrice di confusione del modello (Figura 4.4). La matrice di confusione restituisce una rappresentazione dell'accuratezza di classificazione statistica, attraverso cui è osservabile se vi è "confusione" nella classificazione di diverse classi. Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. L'elemento sulla riga i e sulla colonna j è il numero di casi in cui il classificatore ha categorizzato la classe "vera" i come classe j; di conseguenza, più elementi si trovano nella diagonale principale, più il modello sarà accurato.

Age	SystolicBP	DiastolicBP	BS	intercepts
2.5645	-37.3476	17.6675	195.9827	-15.7645
-3.723	-4.783	17.4673	-119.2748	15.0145
1.878	24.2854	-36.6126	-87.0104	1.1004

Figura 4.3: Tabella dei parametri

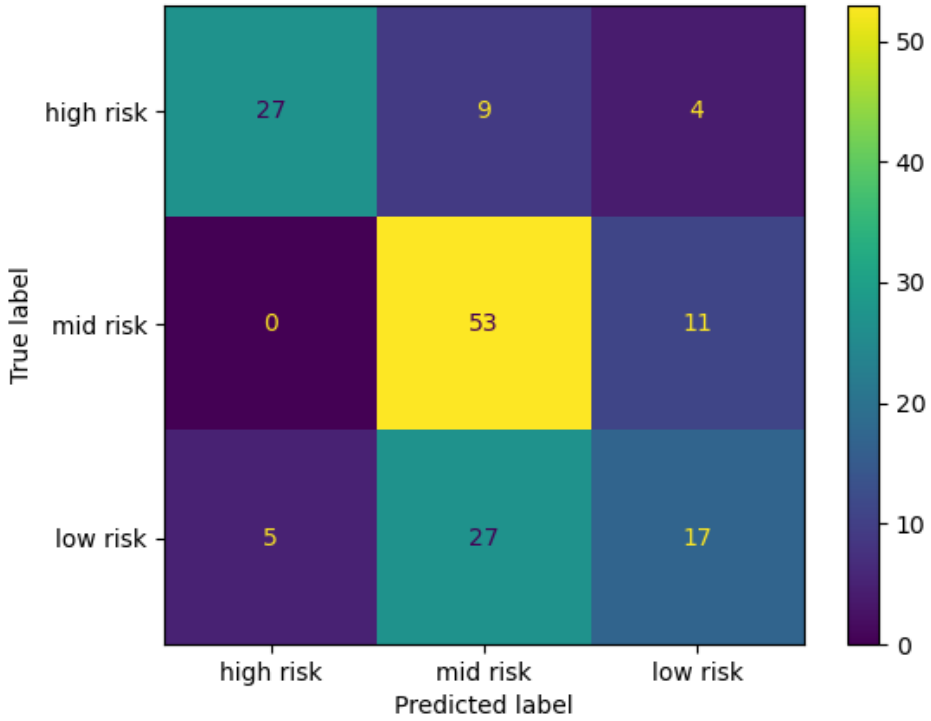


Figura 4.4: Matrice di confusione.
Accuratezza del modello = 63.40 %

Nelle sezioni successive, utilizzeremo questi dati per confrontare i modelli di Federated Learning con questo modello di addestramento centralizzato, in modo da poter confrontare e commentare l'efficienza di queste due tipologie di apprendimento di Machine Learning.

4.3 Analisi sperimentale: Federated Learning

Viene ricreato un sistema di Federated Learning, composto da tre client, ognuno dei quali dispone un set di dati di dimensione (276, 4), (210, 4), (375, 4) rispettivamente, e da un server avente a disposizione un dataset di dimensione (153, 4). Viene fissato una variabile T che indica il numero di interazioni che avverranno tra il server e i client, dopo il quale, viene restituito il modello globale elaborato dal server e una analisi di efficienza del modello stesso rispetto al set di dati di test contenuti nel server.

Inizialmente ogni client effettua un trading locale utilizzando un mini batch di dati rispetto al set a loro disposizione, in modo che il modello venga addestrato all'interno del client; successivamente i parametri ottenuti (parametri di coefficienti e di intercetta) vengono inviati al server, che procederà con l'aggregazione del modello, come riportato in (2.1), ottenendo un vettore aggregato per i coefficienti e per le intercette, che verrà inviato ai client, i quali aggiorneranno i loro parametri con i vettori appena ottenuti dal server e riprenderanno il trading locale. Questo processo di scambio dei parametri tra client e server perdurerà finché non si raggiungerà il numero di interazioni prefissato (nel nostro caso viene fissato T=60).

Si riportano i valori dei coefficienti e dell'intercetta del modello aggregato al server al tempo t=1 (prima interazione tra i client e il server, Figura 4.5) e al tempo t=T=60 (ultima interazione tra client e server, Figura 4.6).

Age	SystolicBP	DiastolicBP	BS	intercepts
3.0341	-28.8425	23.202	166.7379	-11.1984
-5.6584	-10.6421	19.7095	-95.6333	10.9311
-3.871	26.5918	-39.487	-71.9242	0.8168

Figura 4.5: Tabella dei parametri per t=1

Age	SystolicBP	DiastolicBP	BS	intercepts
1.9915	-37.0439	17.3305	195.6814	-15.452
-3.1493	-4.3779	17.4151	-118.4268	15.0212
1.5527	24.3272	-36.0637	-86.9821	1.098

Figura 4.6: Tabella dei parametri per t=T=60

Inoltre, si riportano la percentuale di efficienza del modello rispetto ai dati di test del server e la matrice di confusione del modello aggregato, entrambe al tempo t=1 (Figura 4.7) e t=T=60 (Figura 4.8).

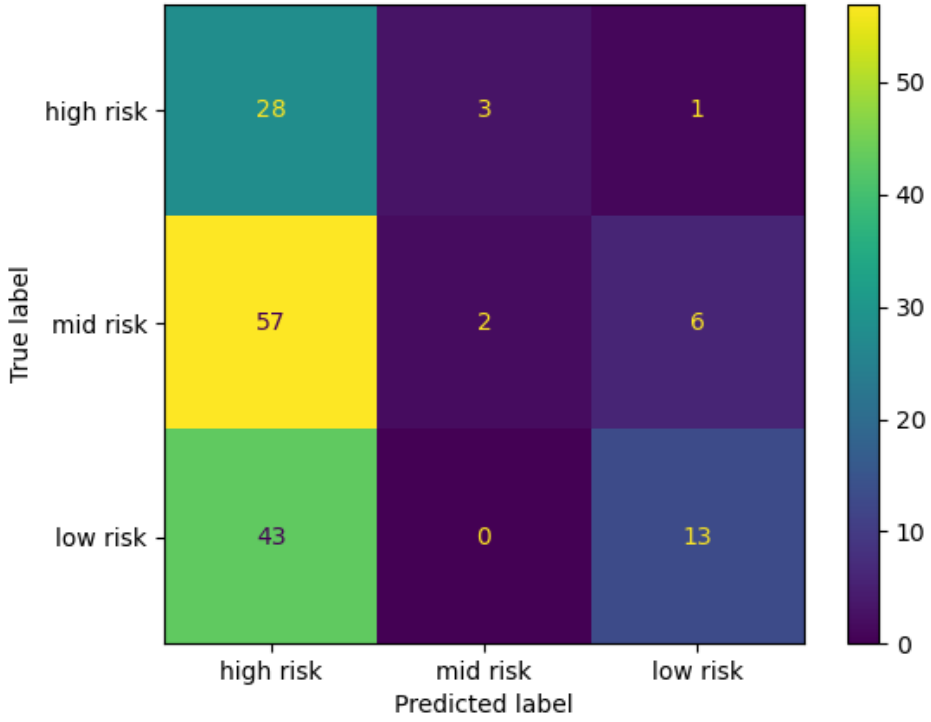


Figura 4.7: Matrice di confusione per t=1.
Accuratezza del modello = 28.10 %

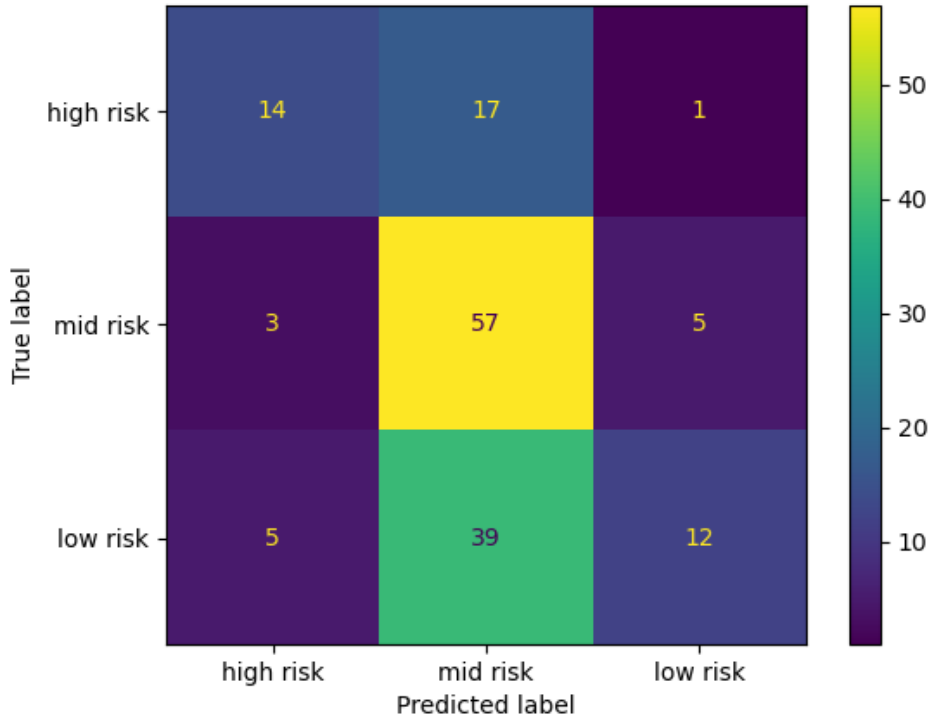


Figura 4.8: Matrice di confusione per t=T=60.
Accuratezza del modello = 54.25 %

4.3.1 Valutazione dei risultati ottenuti

Come si può notare dai risultati ottenuti, in particolare dal valore dell'efficienza del modello e dalla matrice di confusione riportate in Figura (4.7) e (4.8), l'accuratezza del modello è migliorata: infatti, terminato il processo di Federated Learning, il modello riesce a classificare correttamente molte più istanze di quanto non riesca a fare all'inizio del processo. Da notare certamente è la classificazione tra i livelli di rischio medio e basso, che crea "confusione" al modello finale.

Questo problema di classificazione, aumentando il numero di interazioni tra i client e il server, permane, in quanto, come già anticipato nella sezione 4.1 riguardante la caratterizzazione del dataset, le classi *mid risk* e *low risk* si somigliano molto considerando i parametri forniti dal dataset: non è quindi facile per il modello riuscire a creare una distinzione netta tra le due tipologie di rischio.

4.4 Analisi sperimentale: ϵ-Differential Privacy applicata al Federated Learning

Viene ricreato lo stesso modello di Federated Learning proposto nella sezione 4.3, con delle piccole modifiche, che permettono l'applicazione dei concetti di ϵ-Differential Privacy; in particolare, l'aggiunta di rumore laplaciano o gaussiano, secondo le seguenti modalità:

- Una volta dopo aver effettuato l'addestramento dei modelli locali, viene aggiunto del rumore, laplaciano o gaussiano, al vettore dei parametri ottenuti, che poi verranno inviati al server, in accordo con 3.5;
- Il server, dopo aver ottenuto un vettore aggregato dei parametri, prima di inoltrare i nuovi parametri ottenuti ed effettuare un test di efficienza del modello, aggiunge anch'esso del rumore laplaciano o gaussiano al vettore aggregato, e successivamente avanzerà con il processo di Federated Learning, in accordo con 3.7.

4.4.1 Federated Learning con rumore laplaciano

Vengono riportati, come nella sezione precedente, i valori dei coefficienti e dell'intercetta del modello aggregato (successivi all'aggiunta di rumore laplaciano) ai tempi t=1 e t=T=60, rispettivamente in Figura 4.9 e 4.10; inoltre si riportano le matrici di confusione e la percentuale di efficienza del modello aggregato, entrambe al tempo t=1 e t=T=60, rispettivamente in Figura 4.11 e 4.12.

Age	SystolicBP	DiastolicBP	BS	intercepts
12.1034	-22.2602	8.4507	234.7686	-36.3653
-8.1755	-10.2672	29.3742	-185.4219	37.4139
-3.5286	19.7681	-37.963	-109.8053	4.5218

Figura 4.9: Tabella dei parametri per t=1

Age	SystolicBP	DiastolicBP	BS	intercepts
2.7796	-37.2384	17.5315	195.8817	-15.4507
-3.762	-4.4021	17.4326	-118.947	15.0096
1.8869	24.2961	-36.6036	-87.0054	1.0967

Figura 4.10: Tabella dei parametri per t=T=60

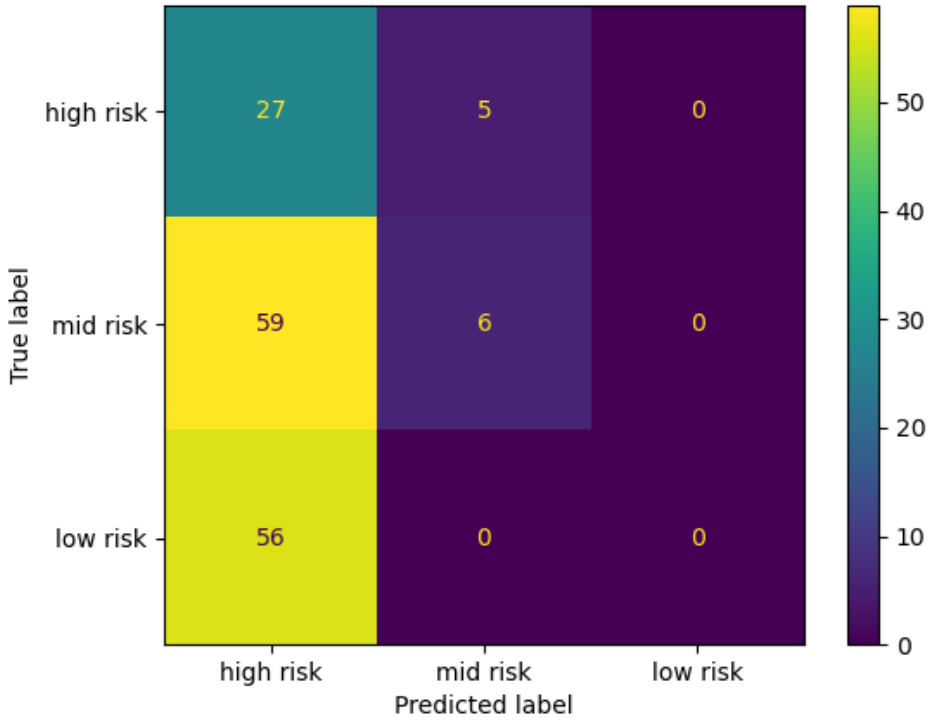


Figura 4.11: Matrice di confusione per t=1.
Accuratezza del modello = 21.56 %

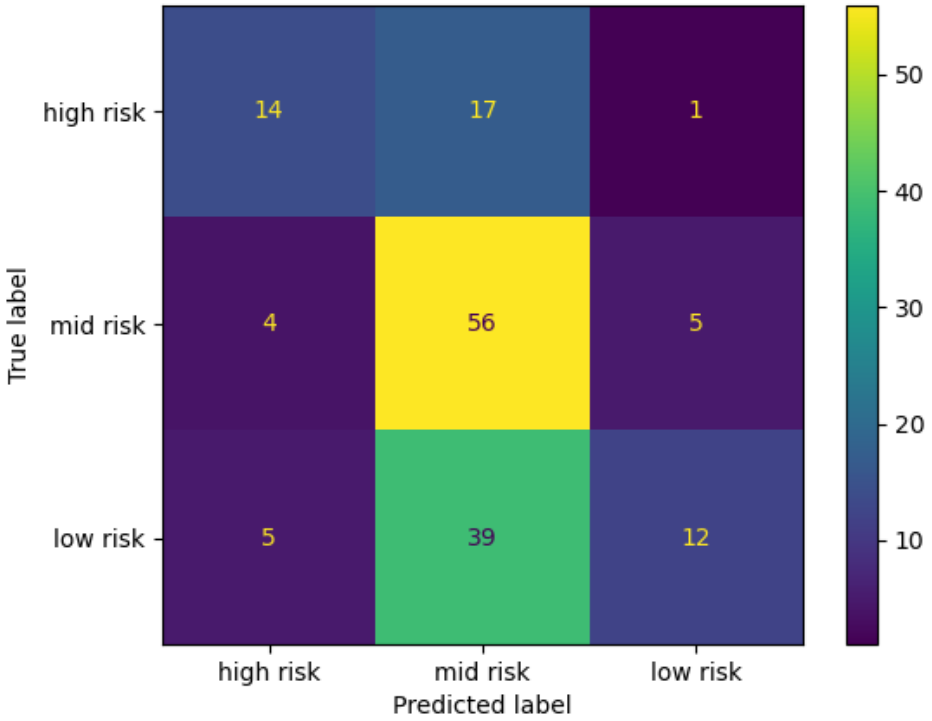


Figura 4.12: Matrice di confusione per t=T=60.
Accuratezza del modello = 53.59 %

4.4.2 Valutazione dei risultati ottenuti

Come si può notare dai risultati ottenuti, in particolare dal valore dell'efficienza del modello e dalla matrice di confusione riportate in Figura (4.11) e (4.12), l'accuratezza del modello generale è migliorato: a differenza del caso precedentemente illustrato in 4.3.1, l'aggiunta di rumore al vettore dei parametri comporta però un effetto di maggior incertezza nella predizione; ciononostante, il processo di Federated Learning garantisce una efficienza che sia avvicina molto al caso precedente, senza aggiunta di rumore.

Se da un lato questo modello non è competitivo con il precedente, dall'altro lato, però, non si può non notare l'apporto che l'aggiunta di rumore laplaciano comporta, in termini di privacy: è vero che la classificazione è meno precisa, ma questo lo si fa coscienziosamente, in modo da garantire maggior sicurezza nel passaggio dei parametri, che, con l'aggiunta di rumore, vengono modificati e non possono essere così interpretati correttamente da un utente esterno.

4.4.3 Federated Learning con rumore gaussiano

Vengono riportati, come nella sezione precedente, i valori dei coefficienti e dell'intercetta del modello aggregato (successivi all'aggiunta di rumore gaussiano) ai tempi t=1 e t=T=60, rispettivamente in Figura 4.13 e 4.15; inoltre si riportano le matrici di confusione e la percentuale di efficienza del modello aggregato, entrambe al tempo t=1 e t=T=60, rispettivamente in Figura 4.14 e 4.16.

Age	SystolicBP	DiastolicBP	BS	intercepts
12.1034	-22.2602	8.4507	234.7686	-36.3653
-8.1755	-10.2672	29.3742	-185.4219	37.4139
-3.5286	19.7681	-37.963	-109.8053	4.5218

Figura 4.13: Tabella dei parametri per t=1

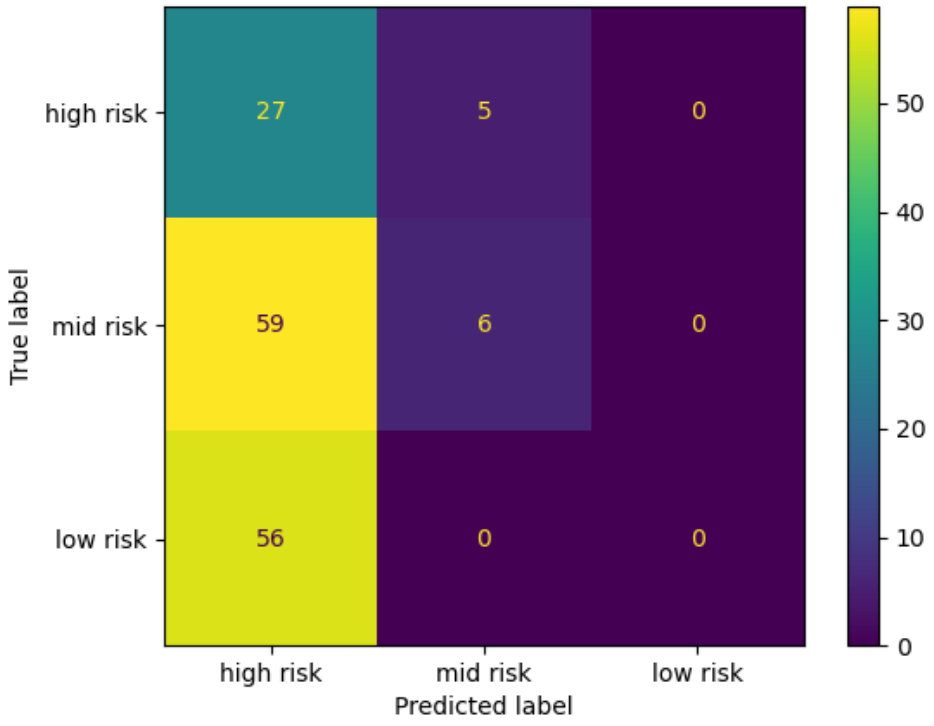


Figura 4.14: Matrice di confusione per t=1.
Accuratezza del modello = 21.56 %

Age	SystolicBP	DiastolicBP	BS	intercepts
-1.8007	-37.487	19.65	271.668	-39.2856
2.5522	-8.8353	26.3424	-172.4281	40.7878
-1.1545	30.125	-41.9099	-144.3096	3.1757

Figura 4.15: Tabella dei parametri per t=T=60

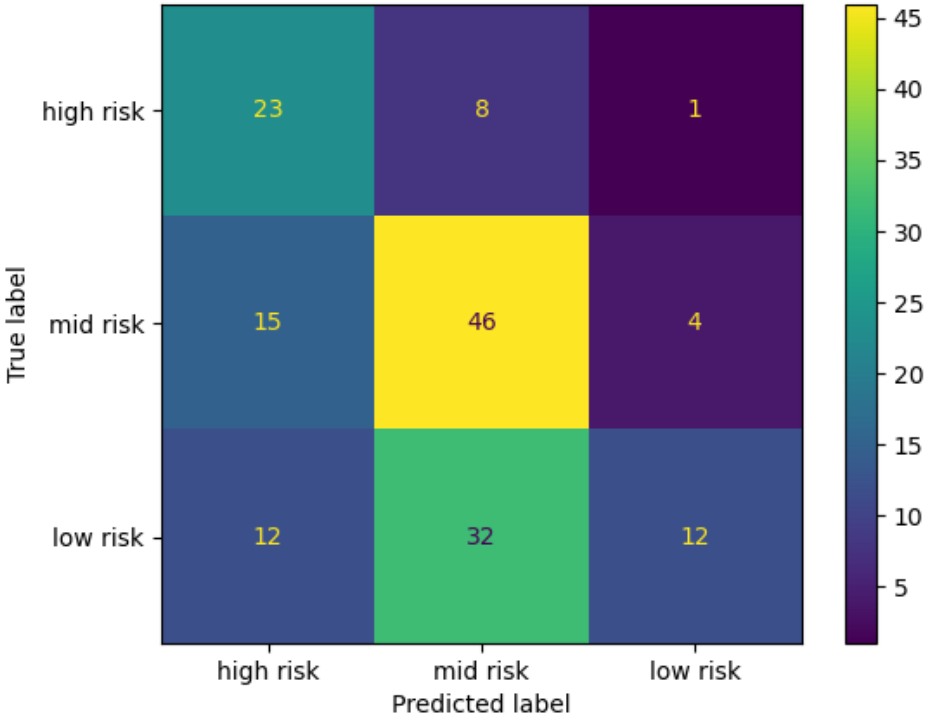


Figura 4.16: Matrice di confusione per t=T=60.
Accuratezza del modello = 52.94 %

4.4.4 Valutazione dei risultati ottenuti

Come si può notare dai risultati ottenuti, in particolare dal valore dell'efficienza del modello e dalla matrice di confusione riportate in Figura (4.14) e (4.16), l'accuratezza del modello generale è migliorato: dopo le prime interazioni, il modello riesce a classificare correttamente molte più istanze fornite come dataset di test. Come per il modello precedente, in cui era stato aggiunto del rumore laplaciano, l'efficienza migliora, cercando di raggiungere l'accuratezza ottenuta nel Federated Learning classico, senza l'aggiunta di rumore; dall'esperimento, l'accuratezza però rimane sempre al di sotto, seppur si avvicini molto.

Dall'altro lato, però, come già anticipato nella sezione precedente, in termini di privacy, l'aggiunta di rumore gaussiano è estremamente preziosa: è vero che la classificazione non sarà competitiva nel tempo, ma questo è il risultato della volontà di garantire maggior sicurezza nel passaggio dei parametri, che, con l'aggiunta di rumore, vengono modificati e di conseguenza diventano non più affidabili.

4.5 Confronto tra i modelli e considerazioni finali

Vengono effettate diverse valutazioni dei modelli, modificando ogni volta la permutazione dei dati contenuti nei vari dataset dei client e del server, mantenendo però inalterate le dimensioni degli stessi, come già descritte in 4.3. In seguito a questa operazione, si procede al calcolo della media e della deviazione standard, in modo da ottenere una lista di valori di accuratezza che siano mediati tra i vari test effettuati.

Si riporta in Figura 4.17 il grafico dell'andamento dell'efficienza, in funzione del tempo, dei tre modelli di Federated Learning proposti nelle sezioni precedenti.

Dal grafico 4.17 si possono osservare alcune caratteristiche importanti.

Tutte e tre le curve, corrispondenti ai tre modelli di Federated Learning, tendono a migliorare ogni round, portandosi verso un livello di efficienza che si avvicina molto al livello di accuratezza che si avrebbe addestrando il modello utilizzando un apprendimento centralizzato, come riportato nella sezione 4.2.

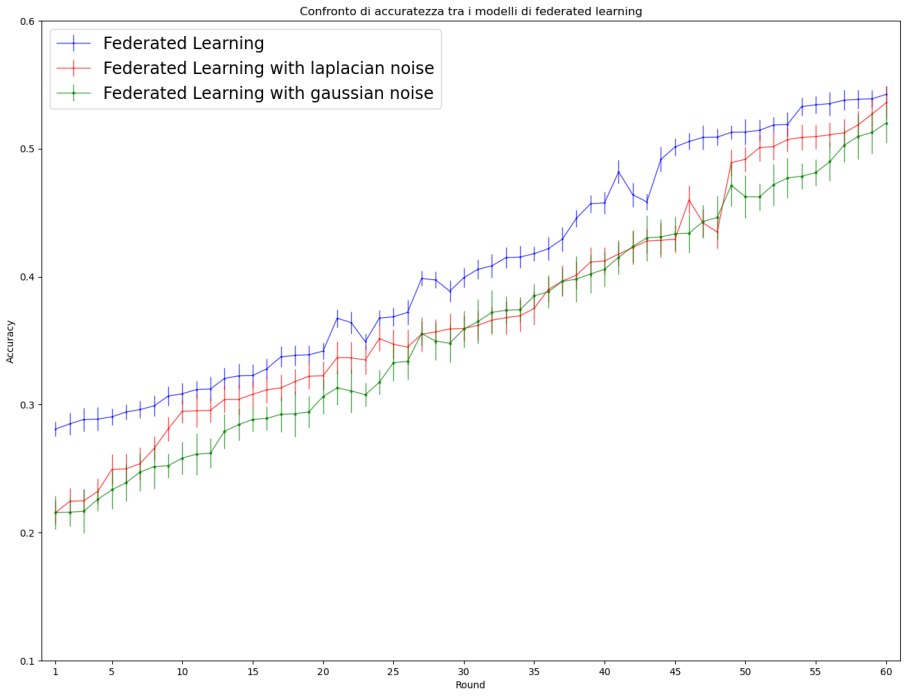


Figura 4.17: Confronto di accuratezza tra i modelli di FL

L'andamento della curva blu, corrispondente al modello di Federated Learning classico senza aggiunta di rumore, è senza alcun dubbio la curva che garantisce maggior efficienza nel tempo: dopo 60 interazioni, l'accuratezza del modello, in media, quasi raggiunge l'efficienza ottenuta con l'apprendimento centralizzato.

Gli andamenti delle altre due curve, corrispondenti ai modelli di Federated Learning con aggiunta di rumore laplaciano o gaussiano, invece, si trovano sempre al di sotto della curva blu, ma comunque tendono a migliorare a quasi ogni interazione, se non per qualche caso in cui si verifica una perdita di qualità: la presenza di questa leggera incostanza è proprio la conseguenza dell'aggiunta di un elemento randomico, ossia il rumore, che rende imprevedibile qualsiasi tipo di previsione successiva.

L'aggiunta di rumore gaussiano e laplaciano non ha le stesse conseguenze: infatti, seppur in entrambi i modelli in considerazione la prestazione è minore rispetto a quella ottenuta utilizzando il solo Federated Learning, l'influenza del rumore gaussiano è molto più marcata e crea maggior instabilità di prestazione

rispetto a ciò che viene prodotto dal rumore laplaciano: infatti, la curva dell'efficienza del modello laplaciano è spesso al di sopra della curva riferita al modello di FL con rumore gaussiano, a parità di round.

Nonostante nel grafico vengano riportati solo 60 cicli di interazione tra client e server, e seppur il dataset utilizzato per la valutazione sperimentale sia di modeste dimensioni, si può affermare in generale che il modello di FL, senza aggiunta di rumore, ma anche i modelli di FL con aggiunta di rumore laplaciano e gaussiano, forniscono una accettabile misura di efficienza medio-alta. Ovviamente l'efficienza raggiunta da un modello di Machine Learning centralizzato, rispetto ai modelli di Federated Learning, sarà in generale maggiore, come è stato confermato in questa analisi; analogamente, lo stesso si può affermare per il confronto tra modelli di FL senza e con aggiunta di rumore (qualsiasi esso sia): ad un incremento di sicurezza, quindi ad un incremento di rumore, corrisponde sempre una perdita di efficienza dei modelli di Federated Learning.

Inoltre, la scelta dell'aggiunta di rumore gaussiano o laplaciano, ha delle conseguenze sull'efficienza: nell'analisi condotta, in accordo con quanto descritto in [2] e in [5], sono state utilizzate due distribuzioni di probabilità in modo che le deviazioni standard coincidano tra le due e che quindi sia fornito, di conseguenza, lo stesso valore di ϵ; in particolare, applicare la ϵ-Differential Privacy utilizzando il rumore laplaciano permette, a parità di ϵ, ossia a parità di sicurezza, di ottenere una efficienza maggiore rispetto all'utilizzo del rumore gaussiano.

5

Conclusioni

In questa tesi abbiamo descritto una tecnica di apprendimento automatico, il Federated Learning, che permette di addestrare un algoritmo attraverso l'utilizzo di dispositivi decentralizzati, senza la necessità di scambiare dati sensibili, in modo da garantire la privacy degli stessi. A questa tecnica, abbiamo affiancato la Differential Privacy, un approccio per fornire maggior privacy durante la condivisione di informazioni in fase di addestramento di un modello di Machine Learning: questo viene raggiunto apportando piccole modifiche ai singoli parametri, aggiungendo del rumore, in modo che i dati non possano essere utilizzati per dedurre informazioni sensibili sugli individui coinvolti.

Successivamente, abbiamo realizzato un'analisi sperimentale, utilizzando un dataset noto, in modo da poter valutare, in modo pratico, le prestazioni di questa tipologia di approccio. Dall'analisi, è emerso che il Federated Learning tende a raggiungere, nel tempo, le stesse performance che l'algoritmo realizzerebbe utilizzando un apprendimento centralizzato; con l'aggiunta di rumore (nel nostro caso, di tipo laplaciano o gaussiano), in accordo con la definizione di privacy differenziale, viene a crearsi maggior incertezza nella predizione, ma, le prestazioni, seppur leggermente minori, rimangono comunque in linea con quelle ottenute nei casi precedenti; anche la tipologia del rumore scelto influenza l'efficienza dell'algoritmo, e, in particolare, a parità di livello di privacy, la performance nel caso laplaciano risulta migliore rispetto al caso gaussiano.

In conclusione, si può affermare che la combinazione del Federated Learning con la Differential Privacy è una potente tecnica per garantire buone prestazioni e, allo stesso tempo, garantire la privacy dei dati personali.

Bibliografia

[1] Marzia Ahmed. *Maternal Health Risk*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DP5D. 2023.

[2] March Boedihardjo, Thomas Strohmer e Roman Vershynin. «Privacy of Synthetic Data: A Statistical Framework». In: *IEEE Transactions on information theory*. 2023.

[3] March Boedihardjo, Thomas Strohmer e Roman Vershynin. «Private Sampling: A Noiseless Approach for Generating Differentially Private Synthetic Data». In: *University of California*. 2022.

[4] Ittai Dayan et al. «Federated learning for predicting clinical outcomes in patients with COVID-19». In: *Nature Medicine, volume 27*. 2021.

[5] Cynthia Dwork e Aaron Roth. «The Algorithmic Foundations of Differential Privacy». In: *Now Foundations and Trends*. 2014.

[6] Robin C. Geyer, Tassilo Klein e Moin Nabi. «Differentially Private Federated Learning:A Client Level Perspective». In: *arXiv preprint arXiv:1712.07557*. 2017.

[7] Hegedus István e Márk Jelasity. «Distributed differentially private stochastic gradient descent: An empirical study». In: *2016 24th Euromicro international conference on parallel, distributed, and network-based processing (PDP), IEEE*. 2016.

[8] Bo Liu et al. «When Machine Learning Meets Privacy: A Survey and Outlook». In: *ACM Computing Surveys (CSUR)*. 2021.

[9] Priyanka Mary Mammen. «Federated Learning: Opportunities and Challenges». In: *University of Massachusetts, Amherst*. 2021.

[10] H. Brendan McMahan et al. «Communication-Efficient Learning of Deep Networks from Decentralized Data». In: *Artificial intelligence and statistics*. 2017.

[11] Garante per la Protezione dei Dati Personali. «Provvedimento n. 112 del 30 marzo 2023». In: *doc. web n. 9870832*. 2023.

[12] S. Shalev-Shwartz e Shai Ben-David. «Understanding machine learning: From theory to algorithms». In: *Cambridge University Press*. 2014.

[13] L. Shi et al. «HFL-DP: Hierarchical Federated Learning with Differential Privacy». In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021.

[14] Jake VanderPlas. «A Whirlwind Tour of Python». In: *O'Reilly Media, Inc*. 2016.

[15] K. Wei et al. «Federated Learning With Differential Privacy: Algorithms and Performance Analysis». In: *IEEE Transactions on Information Forensics and Security*. 2020.

[16] Y. Zhao et al. «Federated Learning with Non-IID Data». In: *arXiv preprint arXiv:1806.00582*. 2018.

[17] Hangyu Zhua et al. «Federated Learning on Non-IID Data: A Survey». In: *arXiv preprint arXiv:2106.06843*. 2021.