UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Uso de técnicas de Aprendizaje Federado para el análisis de datos sensibles

Agustín Tornaría

Proyecto de Grado presentado a la Facultad de Ingeniería de la Universidad de la República en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniería en Computación

Tutores

Lorena Etcheverry
Paola Bermolen

Tribunal

Juan Diego Campo
Diego Garat
Guillermo Trinidad

Montevideo, Uruguay
19 de diciembre de 2024

Resumen

En este trabajo se aborda el uso del Aprendizaje Federado (FL) como una solución emergente para el análisis de datos sensibles en escenarios donde la privacidad es una preocupación crítica. A diferencia de los enfoques tradicionales de Aprendizaje Automático (AA), FL permite entrenar modelos directamente en dispositivos o servidores locales, minimizando la necesidad de transferir datos a un servidor central. La investigación se enfoca en dos variantes principales de FL: el Aprendizaje Federado Horizontal (HFL) y el Aprendizaje Federado Vertical (VFL), que abordan diferentes formas de particionamiento de datos en entornos colaborativos.

El objetivo principal de este proyecto es evaluar la factibilidad y eficacia de FL en comparación con el AA centralizado, utilizando herramientas como TensorFlow Federated y Flower. Para ello, se realizaron experimentos en escenarios controlados con datasets estándar como MNIST y CIFAR-10, y se aplicaron técnicas de FL en un caso de uso realista dentro del contexto de analíticas de aprendizaje (Learning Analytics o LA), específicamente para la predicción de abandono escolar y la clasificación no supervisada de alumnos.

Los resultados indican que, bajo ciertas condiciones, el FL puede igualar al AA tradicional en términos de accuracy, especialmente cuando se configuran adecuadamente los hiperparámetros. Sin embargo, también se destacan las limitaciones del FL en escenarios del mundo real, como la latencia de red y la viabilidad de ejecutar múltiples rondas de entrenamiento en entornos con conectividad limitada.

En conclusión, el Aprendizaje Federado se presenta como una técnica prometedora para la construcción de modelos de AA en contextos donde la privacidad de los datos es prioritaria. No obstante, su implementación práctica requiere una cuidadosa consideración de los desafíos técnicos y operacionales, así como pruebas adicionales en entornos no experimentales para validar su aplicabilidad en escenarios reales.

Palabras clave: Aprendizaje Federado (Federated Learning), Aprendizaje Automático (Machine Learning), Redes Neuronales, TensorFlow Federated, Analíticas de Aprendizaje (Learning Analytics)

Índice general

Capítulo 1

Introducción

En este capítulo se proporciona el contexto y motivación del trabajo, se plantea y define el problema, se deja claro cuales son los objetivos, se plantean los resultados esperados, se establecen resumidamente las conclusiones y se describe la organización general del documento.

1.1. Contexto y Motivación

Hoy en día se genera una gran cantidad de datos que son recolectados y analizados con el fin de obtener información valiosa. Estos datos provienen de distintas fuentes como dispositivos móviles o IoT, redes sociales, sistemas de salud, sistemas financieros, centros educativos, entre otros. Pero muchos de estos datos son sensibles ya sea por la naturaleza de los mismos o gracias a que están bajo regulaciones de privacidad y seguridad. La protección de la privacidad de los datos se ha convertido en un desafío crucial en el análisis de los datos.

El Aprendizaje Federado (Federated Learning o FL) ha surgido como una solución para abordar este desafío. A diferencia de los enfoques tradicionales de Aprendizaje Automático (AA), en donde se centralizan los datos para su procesamiento, el Aprendizaje Federado permite entrenar modelos de Aprendizaje Automático directamente en dispositivos o servidores donde los datos se originan, quitando o minimizando la necesidad de transferir datos a un servidor central (Zhao y cols., 2024).

FL es un enfoque descentralizado propuesto inicialmente por Google (Konečný, McMahan, Ramage, y Richtarik, 2016) para construir modelos de AA utilizando conjuntos de datos distribuidos a través de múltiples dispositivos. El objetivo principal es entrenar modelos estadísticos en dispositivos remotos o centros de datos aislados sin transferir los datos a repositorios centralizados. FL incorpora ideas de múltiples áreas, incluyendo criptografía, Machine Learning, computación heterogénea y sistemas distribuidos. En los últimos años el concepto ha ido creciendo y consolidándose, en líneas que van desde mejoras en aspectos de seguridad, el estudio de problemas estadísticos que surgen en el escenario dis-

tribuido, hasta la extensión del concepto para cubrir escenarios de aprendizaje colaborativo entre organizaciones (T. Li, Sahu, Talwalkar, y Smith, 2020).

Existen dos escenarios principales denominados cross-device y cross-silo (Kairouz, McMahan, Avent, y cols., 2021). En el escenario cross-device, los datos están distribuidos a través de una gran cantidad de dispositivos personales, como teléfonos móviles o dispositivos IoT, y cada dispositivo contribuye con una pequeña cantidad de datos propios de cada usuario. Este escenario enfrenta desafíos particulares como la heterogeneidad de los dispositivos y las conexiones intermitentes. Por otro lado, el escenario cross-silo refiere a situaciones en las que los datos están distribuidos entre un número reducido de organizaciones o entidades, como diferentes hospitales, instituciones financieras o centros de educativos. En este caso cada entidad tiene un volumen significativo de datos pertenecientes a varios individuos, y el enfoque se centra en la colaboración entre estas entidades para mejorar el modelo global sin compartir datos sensibles directamente.

Dentro del escenario cross-silo, los datos pueden estar distribuidos de diferentes maneras, lo que da lugar a dos enfoques principales: Aprendizaje Federado Horizontal (Horizontal Federated Learning o HFL) y Aprendizaje Federado Vertical (Vertical Federated Learning o VFL). En el Aprendizaje Federado Horizontal, los datos están particionados horizontalmente, es decir, cada entidad tiene datos de diferentes individuos pero con las mismas características. Por ejemplo, varios centros educativos pueden tener información sobre distintos estudiantes pero cada centro tiene los mismos tipos de información (edad, cursos, notas, etc). En contraste, el Aprendizaje Federado Vertical se aplica cuando los datos están particionados verticalmente, es decir, cada entidad tiene diferentes características de los mismos individuos. Un ejemplo de esto sería cuando una institución financiera tiene información financiera (fondos, información crediticia, transacciones, etc.) de los clientes y un hospital tiene información médica (edad, diagnóstico, tratamientos, etc.) de los mismos clientes. Ambas instituciones pueden colaborar para construir un modelo sin intercambiar los datos completos de los clientes, sino solo las características específicas que poseen.

1.2. Objetivos de la Tesis

El objetivo principal de esta tesis es explorar el uso de técnicas de Aprendizaje Federado para el análisis de datos sensibles. En primer lugar, se busca proporcionar un marco teórico comprensivo sobre FL, incluyendo sus fundamentos, diferencias con el AA tradicional y las herramientas y datasets disponibles para su implementación. En segundo lugar, se pretende evaluar la factibilidad de FL mediante la realización de experimentos con herramientas específicas y datasets tanto estándar como más complejos y aplicados.

Los objetivos específicos de esta tesis son:

1. Introducir los fundamentos teóricos de FL, comparando los escenarios cross-device y cross-silo, describiendo un proceso de entrenamiento estándar

y presentando los enfoques utilizados para entrenar según el tipo de particionamiento de datos que se cuente, HFL y VFL.

2. Explorar el uso de herramientas como TensorFlow Federated y Flower en el contexto de HFL, comparando los resultados con enfoques tradicionales de AA.

3. Experimentar con un caso de uso aplicado al contexto de analíticas de aprendizaje (Learning Analytics o LA), utilizando técnicas de FL para la predicción de abandono y clasificación no supervisada de alumnos.

4. Analizar el enfoque de VFL, evaluando distintos algoritmos y técnicas de AA en este contexto.

Para cumplir con los objetivos se comienza proporcionando un marco teórico sobre Aprendizaje Federado, incluyendo sus fundamentos, los conceptos claves, el proceso de entrenamiento y los distintos desafíos que presenta. Además se exploran las diferencias con los enfoques tradicionales de Aprendizaje Automático, mecanismos para preservar la privacidad de los datos, así como las diferencias entre los escenarios horizontal y vertical. Se continúa presentando un estudio de factibilidad y comparación de herramientas realizado en el contexto del proyecto ANII - FLEA: Aprendizaje Federado aplicado a LA (informe de avance para la ANII disponible en (Bermolen y cols., 2022)[1]), realizando experimentos con datasets conocidos de manera de tener una linea base con la cuál comparar los resultados obtenidos con el uso de Aprendizaje Federado Horizontal. También se presenta un caso de uso de Aprendizaje Federado Horizontal en LA presentado en (Fachola y cols., 2023), usando técnicas de FL para la predicción de abandono y clasificación no supervisada de alumnos. Por último se revisan distintos algoritmos propuestos en la literatura, abarcando la aplicación de aprendizaje federado vertical sobre técnicas de aprendizaje automático tradicional, redes neuronales y regresión lineal.

1.3. Estructura de la Tesis

La tesis se organiza en seis capítulos, que se detallan a continuación:

- **Capítulo 1: Introducción** - Proporciona el contexto, motivación y objetivos de la tesis, así como una visión general de la estructura del documento.

- **Capítulo 2: Fundamentos Teóricos de Aprendizaje Federado** - Introduce los conceptos y definiciones fundamentales de FL, compara los escenarios cross-device y cross-silo, describe un proceso de entrenamiento estándar y destaca las diferencias con el AA tradicional. Además, se presentan las herramientas y datasets relevantes, proporcionando un panorama general del FL desde un punto de vista teórico.

[1] Repositorio del proyecto en: https://gitlab.fing.edu.uy/lorenae/flea/-/tree/main?ref_type=heads

- **Capítulo 3: Aprendizaje Federado Horizontal - Estudio de Factibilidad y Comparación de Herramientas** - Explora el uso de las herramientas TensorFlow Federated y Flower en el contexto del HFL, utilizando datasets estándar como MNIST y CIFAR-10. Se comparan los resultados obtenidos con los enfoques tradicionales de AA para estudiar la factibilidad de la técnica en un contexto controlado.
- **Capítulo 4: Caso de Aprendizaje Federado Horizontal en Analíticas de Aprendizaje** - Aplica las técnicas de FL en un caso de uso realista, utilizando un dataset de cursos educativos. Se simula un escenario con múltiples centros educativos y se experimenta con la predicción de abandono y la clasificación no supervisada de alumnos, explorando la aplicabilidad del FL en el contexto de LA.
- **Capítulo 5: Aplicaciones de Aprendizaje Federado Vertical** - Se centra en el análisis del VFL, explorando distintos algoritmos y técnicas de AA.
- **Capítulo 6: Conclusiones**- Resume los puntos clave de la tesis y discute las implicaciones de los resultados obtenidos.

Capítulo 2

Fundamentos Teóricos de Aprendizaje Federado

En este capítulo se proporcionará un marco teórico detallado de FL, explorando sus conceptos fundamentales y diferenciándolo de las técnicas de AA tradicionales. Se abordarán los distintos escenarios en los que se puede implementar el FL, como los escenarios cross-device y cross-silo, y se describirá el ciclo de vida y el proceso de entrenamiento que caracteriza a esta metodología. Además, se analizarán los mecanismos para preservar la privacidad de los datos, un aspecto crucial en FL, y se introducirán y explicarán las diferencias entre los particionamientos de datos horizontal y vertical, fundamentales para entender las distintas aplicaciones del FL que se presentan la tesis. Por último, se presentarán las herramientas y datasets más relevantes que facilitan la implementación de FL, proporcionando una visión integral de las capacidades y desafíos asociados con esta tecnología emergente. Este capítulo sienta las bases teóricas necesarias para los análisis y experimentos prácticos que se llevarán a cabo en los capítulos posteriores.

2.1. Concepto

FL es una técnica de AA que permite entrenar modelos utilizando datos distribuidos entre múltiples dispositivos o servidores, sin necesidad de centralizar los datos.

Origen de FL

El termino aprendizaje federado es introducido por Google en 2016 en (H. B. McMahan, Moore, Ramage, Hampson, y y Arcas, 2023). En este paper se destaca el aumento en el uso de dispositivos móviles, como celulares y tablets, que además de ser transportados con frecuencia, cuentan con sensores (como cámaras, micrófonos y GPS). Estos sensores proporcionan acceso a una gran cantidad

de información que podría utilizarse para entrenar modelos de AA. Sin embargo, debido a que estos datos son en su mayoría datos privados y sensibles, no se pueden centralizar. La técnica de Aprendizaje Federado se plantea como una solución que permite mejorar la usabilidad de aplicaciones, haciendolas más inteligentes sin la necesidad de centralizar los datos. La tarea de aprendizaje es realizada por la federación de algunos dispositivos llamados clientes, que son coordinados por un servidor central. Cada cliente entrena de manera local con sus datos y actualiza el modelo global, comunicando únicamente esta actualización y nunca los datos crudos.

Con el tiempo, surgió el interés de aplicar esta técnica en otros ámbitos, diferenciándose dos escenarios. El primer escenario se lo llama *cross-device*, donde se cuenta con muchos dispositivos, generalmente móviles, que no necesariamente están siempre disponibles. El segundo escenario, denominado *cross-silo*, involucra a menos clientes pero con mayor disponibilidad, cómo pueden ser múltiples organizaciones coordinandose para entrenar un modelo, a este escenarios se lo denomina *cross-silo*. Se discuten estos escenarios con más profundidad en la Sección 2.3.

Definición de FL

Dadas estas variaciones el paper *Advances and Open Problems in Federated Learning* (Kairouz, McMahan, Avent, y cols., 2021) da una definición más amplia de lo que es aprendizaje federado:

El Aprendizaje Federado es un entorno de aprendizaje automático donde múltiples entidades (clientes) colaboran en la resolución de un problema de aprendizaje automático, bajo la coordinación de un servidor central o proveedor de servicios. Los datos crudos de cada cliente se almacenan localmente y no se intercambian ni transfieren; en su lugar, se utilizan actualizaciones específicas destinadas a la agregación inmediata para lograr el objetivo de aprendizaje.

El Aprendizaje Federado representa un avance significativo en la forma en que se pueden entrenar modelos de AA utilizando datos distribuidos, ofreciendo un enfoque que prioriza la privacidad y la seguridad de los datos. Este capítulo continuará explorando en detalle los diferentes aspectos y aplicaciones del Aprendizaje Federado, proporcionando una base sólida para entender sus fundamentos teóricos.

2.2. Ciclo de Vida y Proceso de Entrenamiento

En esta sección se explora en detalle el ciclo de vida y el proceso de entrenamiento en el contexto del Aprendizaje Federado (FL). Se describen los pasos esenciales desde la identificación del problema hasta el despliegue del modelo final, subrayando las particularidades y desafíos propios del FL. Además, se presenta un proceso típico de entrenamiento basado en el método de agregación Federated Averaging, destacando las fases clave que permiten la coordinación y

actualización de los modelos en un entorno distribuido y descentralizado.

2.2.1. Ciclo de Vida

El ciclo de vida de un modelo de Aprendizaje Federado (FL) es un proceso iterativo y colaborativo que involucra varios pasos, desde la identificación del problema hasta el despliegue del modelo entrenado. A continuación, se describe en detalle este ciclo de vida, basado en la práctica común y las recomendaciones encontradas en la literatura (Kairouz, McMahan, Avent, y cols., 2021).

1. **Identificación del Problema**

   El ciclo de vida comienza con la identificación de un problema que puede ser abordado mediante FL. Un ingeniero de modelos o un experto en el dominio identifica la necesidad de un modelo específico.

2. **Configuración de Clientes**

   Una vez identificado el problema es posible que se necesite configurar a los clientes, estos son las entidades participantes (como dispositivos móviles, organizaciones o sensores IoT) cuyos datos se utilizan localmente. Esta configuración puede implicar la recopilación y almacenamiento de datos adicionales o metadatos, en muchos casos la aplicación ya almacena los datos necesarios, sin embargo en algunos casos puede ser necesario mantener datos adicionales, como datos de interacción del usuario para proporcionar etiquetas en una tarea de aprendizaje supervisado.

3. **Selección de hiperparámetros**

   Para entrenar el modelo se deben seleccionar hiperparámetros. En un contexto centralizado se cuentan con algunos hiperparametros que son necesarios ajustar, en el contexto federado se tienen los mismos hiperparámetros del centralizado y además otros propios de FL, en la Sección 2.2.3 se explican algunos de ellos. Estos hiperparámetros tienen un gran impacto en la calidad del módelo y son claves para obtener buenas métricas con buena convergencia, en el Capítulo 3 se realizan experimentos para ver como afectan algunos de estos hiperparámetros. Para seleccionar los hiperparámetros se puede realizar pruebas con los mismos mediante una simulación.

4. **Entrenamiento del Modelo Federado**

   Con los hiperparámetros ajustados, se inician múltiples tareas de entrenamiento federado para entrenar diferentes variaciones del modelo o utilizar diferentes hiperparámetros de optimización. Este proceso involucra la colaboración de múltiples clientes, que realizan el entrenamiento localmente y envían sus actualizaciones al servidor central, en la Sección 2.2.2 se detalle cómo es el proceso de entrenamiento en un escenario típico de FL.

5. **Evaluación del Modelo Federado**

   Después de que las tareas de entrenamiento hayan avanzado lo suficiente, los modelos entrenados se analizan para seleccionar los mejores candidatos. El análisis puede incluir métricas computadas en conjuntos de datos estándar en el servidor central o una evaluación federada en la que los modelos se envían a clientes reservados para evaluar su desempeño en datos locales del cliente.

6. **Despliegue**

   Finalmente, una vez seleccionado un buen modelo, este pasa por un proceso estándar de lanzamiento de modelos. Este proceso incluye aseguramiento de calidad manual, pruebas A/B en vivo (generalmente utilizando el nuevo modelo en algunos dispositivos y el modelo de generación anterior en otros dispositivos para comparar su desempeño en condiciones reales) y un despliegue gradual hasta que se haya comprobado su efectividad y fiabilidad.

Este ciclo de vida proporciona una estructura clara y repetitiva para desarrollar y desplegar modelos de FL, asegurando que el proceso sea eficiente y que los modelos resultantes sean de alta calidad y usabilidad.

2.2.2. Proceso de Entrenamiento

El proceso de entrenamiento en un escenario típico de FL sigue un ciclo iterativo en el cual un servidor central coordina la colaboración entre múltiples clientes que almacenan sus propios datos crudos, en una iteración a partir de datos locales y el modelo global se entrenan modelos de forma local, obteniendo nuevos parámetros que son transferidos al coordinador para ser agregados y actualizar el modelo global. Uno de los algoritmos más comunes de agregación es el algoritmo Federated Averaging (FedAvg), propuesto por McMahan en el paper que introduce el concepto de FL (H. B. McMahan y cols., 2023). En la sección 2.2.3 se entra en más detalle sobre el algoritmo. Este algoritmo y su proceso de entrenamiento se estructuran en varias etapas clave, que se repiten hasta que el modelo alcanza el nivel deseado de desempeño o se decide detener el entrenamiento. En la figura 2.1 se ilustra el proceso de entrenamiento, el cual se detalla a continuación (Kairouz, McMahan, Avent, y cols., 2021).

Proceso de Entrenamiento en Aprendizaje Federado

- **Selección de Clientes:** En cada iteración se utiliza una fracción de los clientes, el servidor central selecciona un subconjunto de clientes que además cumplen con ciertos requisitos de elegibilidad para que el proceso de entrenamiento no afecte negativamente la experiencia del usuario. Por ejemplo, en un entorno móvil, los dispositivos pueden participar en el entrenamiento solo si están conectados a una fuente de energía, tienen

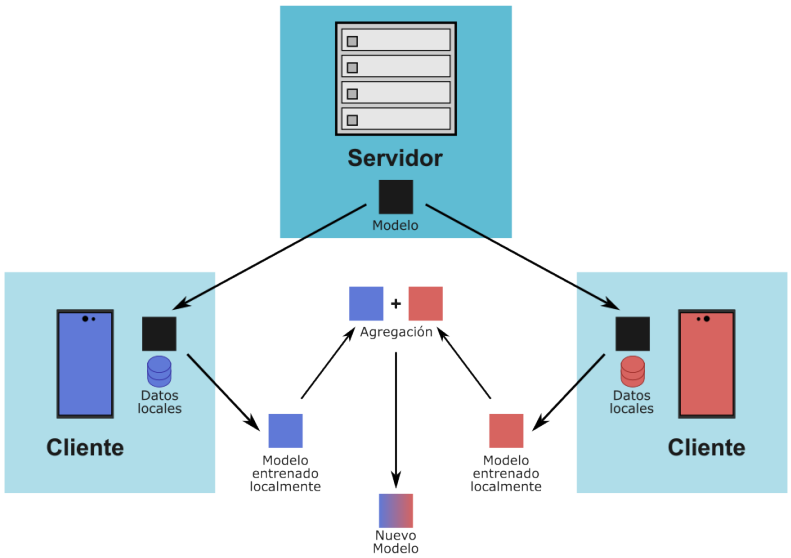

Figura 2.1: Proceso de entrenamiento en Aprendizaje Federado (Zaman, 2020).

una conexión Wi-Fi sin límite de datos y están inactivos. Estas limitantes pueden afectar negativamente al resultado generando sesgos en los resultados. En el capítulo 2.5.3 se discute en mayor profundidad como afectan estas limitantes.

- **Cálculo en el Cliente:** Los clientes seleccionados descargan los parámetros actuales del modelo con los que junto con sus datos locales entrenan el modelo. Este entrenamiento local puede involucrar algoritmos como el Stochastic Gradient Descent (SGD), que ajusta los parámetros del modelo basándose en los datos propios del cliente.

- **Agregación:** El servidor central recopila las actualizaciones de los modelos de los diferentes clientes. Para mejorar la eficiencia, es posible que se omitan los resultados de aquellos clientes que se retrasan. Durante esta fase, se pueden implementar técnicas adicionales para mejorar la privacidad y la eficiencia de la comunicación, como la agregación segura, la compresión con pérdida de las actualizaciones y la adición de ruido o el recorte de actualizaciones para aplicar differential privacy.

- **Actualización del Modelo:** El servidor actualiza el modelo global utilizando la agregación de las actualizaciones recibidas de los clientes participantes en la ronda actual. Este modelo actualizado se difundirá en la siguiente iteración.

2.2.3. Descripción Formal de FL para Predicción de Datos Supervisados

En esta sección se presenta una descripción más formal del escenario de Aprendizaje Federado para el caso de entrenamiento de un modelo de predicción de datos supervisados, dado que este es el tipo de algoritmo más mencionado en la literatura del tema. Además este tipo de algoritmo se utiliza en el Capítulo 3 y el Capítulo 4. Se empieza describiendo el esquema centralizado

Esquema Centralizado

Una modelo de aprendizaje automático supervisado para predicción M puede entenderse como un modelo para una función $f : X \to Y$, donde X es un conjunto de datos e Y es un conjunto de etiquetas. Dados ciertos parámetros w de M y un $x \in X$ con etiqueta y, se calcula la estimación $M(x; w) = \hat{y}$ para el valor real $f(x) = y$.

El entrenamiento del modelo no es otra cosa que el proceso mediante el cual los parámetros w se ajustan, cambian, para estimar mejor la función f. Esto se logra utilizando un conjunto de datos de entrenamiento $train \subset X$ para los cuales ya se conoce el valor de $f(x)$ para cada $x \in train$. Dicho proceso es de carácter iterativo. En la iteración inicial los pesos se inicializan de cierta forma, por ejemplo al azar, luego se recibe una entrada x con etiqueta conocida $f(x) = y$ y el modelo calcula su salida $\hat{y}$. Estos valores son insumo para una función de coste $L = L(y, \hat{y}) = L(y, M(x; w))$, que, como se hizo explícito, depende de los pesos w. La función L se minimiza con respecto a los pesos w mediante un proceso de optimización para encontrar los nuevos pesos con los cuales $M(x; w)$ estima mejor a y. En la siguiente iteración se dará un nuevo x y se repetirá el proceso, pero con los pesos ajustados a partir de la iteración anterior.

Típicamente los datos no se procesan de a uno en uno, sino de a lotes o *batches*. El conjunto $train$ se particiona en un número de lotes de cierto tamaño fijo B , llamado *batch-size*[1]. Para procesar un lote B_i se calculan las salidas para todos los $x \in B_i$ y se encuentran los mejores pesos teniendo en cuenta información de todo el lote. Una época constituye el procesamiento de todo el conjunto de entrenamiento, es decir el procesamiento de todos los lotes. Los modelos pueden entrenar por muchas épocas, es decir puede que vean el mismo dato muchas veces, tantas como épocas haya entrenado, se denota E la cantidad de épocas entrenadas.

Esquema Federado

En la situación descrita anteriormente está implícito que el modelo tiene acceso a todos los datos del conjunto $train$ al mismo tiempo para entrenar. El desafío que se plantea en el contexto federado es el poder entrenar un modelo sin tener acceso a todos los datos al mismo tiempo.

[1] El tamaño es el mismo para todo los lotes a excepción del último, que tendrá el tamaño correspondiente al resto de los elementos E mod $batch - size$

En el esquema federado definimos nuevamente un modelo de aprendizaje automático supervisado para predicción M como un modelo para una función $f : X \to Y$, donde X es un conjunto de datos e Y es un conjunto de etiquetas. Dados ciertos parámetros w de M y un $x \in X$ con etiqueta y, se calcula la estimación $M(x; w) = \hat{y}$ para el valor real $f(x) = y$.

Al igual que en el esquema centralizado el entrenamiento del modelo es el proceso mediante el cual los parámetros w se ajustan para estimar mejor la función f. Esto se logra utilizando los datos de K clientes, donde cada cliente i cuenta con su propio conjunto de datos de entrenamiento $train_i \subset X$ para los cuales conocen el valor de $f(x)$ para cada $x \in train_i$, y la unión de estos conjuntos sería equivalente al conjunto de entrenamiento completo $train = \bigcup_{1 \leq i \leq K} train_i$. Además se cuenta con un servidor que coordina el entrenamiento y guarda el modelo global entrenado. El entrenamiento se realiza en R rondas, donde en cada ronda se elige una fracción C de los clientes para participar en dicha ronda. Una vez elegidos los clientes a participar el servidor comparte el modelo global con cada uno de estos participantes quienes se encargan de entrenar un modelo local con su conjunto de datos $train_i$.

El entrenamiento en cada uno de los clientes es similar al entrenamiento en en el esquema centralizado, es un proceso iterativo que consiste de E épocas, donde se recibe una entrada x con etiqueta conocida $f(x) = y$ y el modelo calcula su salida $\hat{y}$. Estos valores son insumo para una función de coste $L = L(y, \hat{y}) = L(y, M(x; w))$. La función L se minimiza con respecto a los pesos w mediante un proceso de optimización para encontrar los nuevos pesos con los cuales $M(x; w)$ estima mejor a y. En la siguiente época se dará un nuevo x y se repetirá el proceso, pero con los pesos ajustados a partir de la iteración anterior.

Una vez más los datos no se procesan de a uno en uno, sino de a lotes o *batches*. El conjunto $train$ se particiona en un número de lotes de cierto tamaño fijo B , llamado *batch-size*[2]. Para procesar un lote B_i se calculan las salidas para todos los $x \in B_i$ y se encuentran los mejores pesos teniendo en cuenta información de todo el lote.

Este proceso se realiza en cada uno de los clientes elegidos para participar en la ronda, obteniendo así un modelo local actualizado para cada uno de estos clientes, generado a partir del modelo global brindado por el servidor. La intención es utilizar lo aprendido en cada uno de los clientes para actualizar el modelo global, para esto se comparten los modelos locales con el servidor y ser realiza una agregación de los mismos, nunca compartiendo los datos crudos. Existen diferentes algoritmos de agregación que pueden ser utilizados, en el algoritmo 1 se presenta Federated Averaging (H. B. McMahan y cols., 2023), es uno de los algoritmos más utilizados y mencionado en la literatura, y el que será utilizado en los capítulos 3 y 4. El mismo está basado en que cada cliente aplique Stocastic Gradiend Descent (SGD) y el servidor se encargue de realizar el promedio del modelo.

Una vez agregados los modelos este pasa a ser el nuevo modelo global y se

[2]El tamaño es el mismo para todo los lotes a excepción del último, que tendrá el tamaño correspondiente al resto de los elementos E mod $batch - size$

Algorithm 1 Federated Averaging.

Parámetros:

C la fracción de clientes usados
R la cantidad de rondas globales
E es el numero de epocas locales
B es el tamaño del minibatch
η es el learning rate

Notación:

K es la cantidad de clientes totales
w_t^i son los pesos del cliente i en una ronda t
P_i son los ejemplos que tiene el cliente i
n_i es la cantidad de ejemplos locales para el cliente i

Resultado: Pesos actualizados w_R

Ejecución del servidor:

inicializar w_0
for each ronda r de 0 a $R-1$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_r \leftarrow$ (conjunto aleatorio de m clientes)
  for each cliente $i \in S_r$ **in parallel do**
    $w_{r+1}^i \leftarrow \text{ClientUpdate}(i, w_r)$
  $m_r \leftarrow \sum_{i \in S_r} n_i$
  $w_{r+1} \leftarrow \sum_{i \in S_r} \frac{n_i}{m_t} w_{r+1}^i$
return w_R

ClientUpdate(i,w)

$B_i \leftarrow$ (particionar P_i en batches de tamaño B)
for each epoca local e de 1 a E **do**
  for each batch $b \in B_i$ **do**
    $w \leftarrow w - \eta \nabla \text{ loss}(w, b)$
return w

procede con la siguiente ronda, eligiendo $C \times K$ nuevos clientes, a los que se les comparte el nuevo modelo para que entrenen localmente y así siga el proceso iterando por las R rondas definidas anteriormente. Es importante agregar que en la primer ronda aún no se cuenta con un modelo global y por esto es necesario inicializar los pesos al igual que se realiza en el esquema centralizado, pudiendo utilizar algún criterio o simplemente al azar.

En el caso federado se describieron varios hiperparámetros que hay que considerar y que influyen significativamente en el desempeño de los modelos obtenidos. En el Capítulo 3 se realizan pruebas sobre como afectan la cantidad de rondas R, la fracción de clientes elegidos por ronda C y la cantidad de épocas

de entrenamiento local E .

2.3. Cross-Device y Cross-Silo

Como ya se mencionó, existen dos escenarios en FL : *cross-device* y *cross-silo*. Cada uno de estos escenarios tiene características y aplicaciones específicas que los hacen adecuados para diferentes tipos de entornos y problemas de aprendizaje automático. Ambos escenarios comparten el principio fundamental de mantener los datos localmente para proteger la privacidad, pero difieren significativamente en sus configuraciones y desafíos específicos. A continuación, se detallan las particularidades de cada escenario (Kairouz, McMahan, Avent, y cols., 2021).

2.3.1. Escenario Cross-Device

El escenario *cross-device* involucra una gran cantidad de dispositivos, típicamente móviles o dispositivos del Internet de las Cosas (IoT), que pueden no estar siempre disponibles. Los datos se generan y permanecen localmente en cada dispositivo, y estos dispositivos colaboran en el entrenamiento de un modelo global sin compartir sus datos crudos. La figura 2.2 presenta una representación de este escenario.

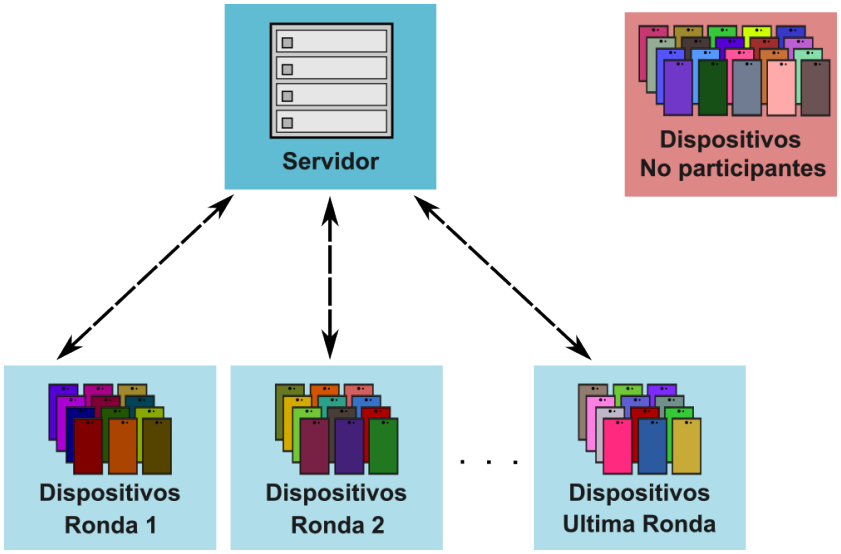

Figura 2.2: Escenario Cross-Device. Se cuenta con mayor cantidad de clientes, generalmente dispositivos móviles o IoT. En cada ronda cambian los dispositivos utilizados y algunos dispositivos no se usan.

Características del Escenario Cross-Device:

- **Escala de Distribución:** Puede incluir hasta 10^{10} dispositivos, lo que implica una escala masiva y una alta variabilidad en la disponibilidad de los dispositivos.
- **Disponibilidad de Datos:** Solo una fracción de los dispositivos está disponible en un momento dado, lo que puede variar con patrones diurnos u otros factores.
- **Comunicación:** La comunicación suele ser el cuello de botella principal, ya que los dispositivos se conectan a través de redes Wi-Fi o conexiones más lentas.
- **Estado de los Clientes:** Los clientes no mantienen estado, típicamente participan solo una vez en cada tarea de entrenamiento, lo que implica que en cada ronda se utilizan conjuntos de dispositivos nuevos.
- **Confiabilidad de los Clientes:** Se espera que un alto porcentaje de los clientes (5 % o más) falle o se desconecte durante el entrenamiento debido a problemas de batería, red o disponibilidad.
- **Partición de Datos:** La partición de datos es fija, siempre es partición horizontal (por ejemplos), donde cada dispositivo tiene un subconjunto diferente de datos con las mismas características.

El escenario *cross-device* es especialmente útil para mejorar aplicaciones móviles y servicios basados en IoT, donde la privacidad de los datos y la eficiencia de la comunicación son cruciales. Este escenario solo funciona con Aprendizaje Federado Horizontal ya que para utilizar Aprendizaje Federado Vertical es necesario contar con todos los dispositivos disponibles al mismo tiempo y se debe coordinar la ejecución de los algoritmos, en la sección 2.4 se explica en más detalle los tipos de particionamientos.

En el Capítulo 3 se explora el uso de herramientas para HFL en el contexto de escenarios *cross-device*.

2.3.2. Escenario Cross-Silo

El escenario *cross-silo* involucra un número menor de clientes que son organizaciones o centros de datos geográficamente distribuidos, como instituciones médicas, financieras o educativas. En este escenario, cada cliente (silo) almacena y gestiona su propio conjunto de datos de manera descentralizada. La figura 2.3 presenta una representación de este escenario.

Características del Cross-Silo Federated Learning:

- **Escala de Distribución:** Típicamente incluye de 2 a 100 clientes, lo que implica una menor escala en comparación con el escenario *cross-device*.

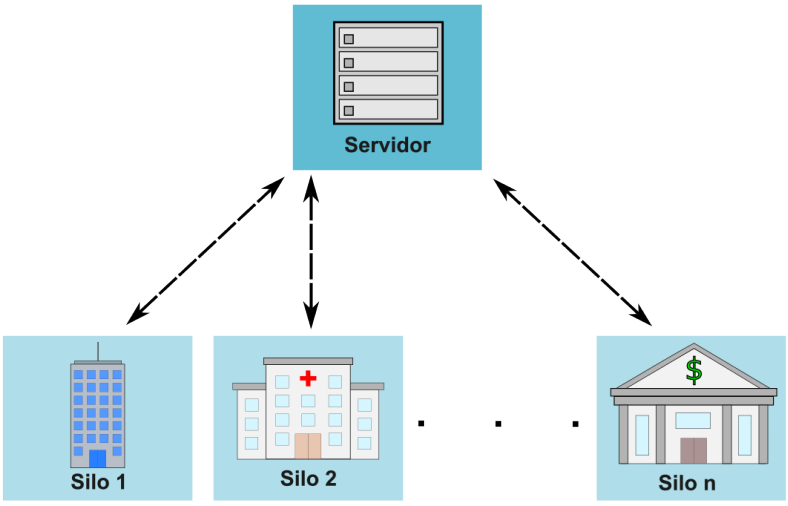

Figura 2.3: Escenario Cross-Silo. Se cuenta con menor cantidad de clientes, generalmente organizaciones como instituciones, hospitales o bancos. Los clientes pueden participar en todas las rondas.

- **Disponibilidad de Datos:** Los clientes están casi siempre disponibles, lo que permite una colaboración más estable y predecible.
- **Comunicación:** Puede ser un cuello de botella, aunque menos crítico que en el escenario *cross-device*. Normalmente existe un servidor central que coordina las actualizaciones con los clientes aunque existen otras variantes.
- **Estado de los Clientes:** Los clientes pueden participar en múltiples rondas de entrenamiento, manteniendo el estado entre estas rondas.
- **Confiabilidad de los Clientes:** Hay relativamente pocos fallos, lo que hace que el entorno sea más fiable y robusto para el entrenamiento colaborativo.
- **Partición de Datos:** La partición de datos es fija y puede ser tanto horizontal (por ejemplos) como vertical (por características), dependiendo de cómo se distribuyan los datos entre los clientes.

El escenario *cross-silo* es ideal para aplicaciones en las que múltiples organizaciones desean colaborar para entrenar modelos de AA sin compartir datos sensibles, como en el sector de la salud, finanzas y educación. Este escenario funciona con Aprendizaje Federado Horizontal, al igual que el escenario cross-device, pero además se puede aplicar Aprendizaje Federado Vertical gracias a

que se cuenta con menos clientes más confiables a partir de los cuales se pueden coordinar la ejecución de los algoritmos. En la sección 2.4 se explica en más detalle los tipos de particionamientos.

En el Capítulo 4 se plantea un caso de uso en LA en el que se aplica FL con un escenario cross-silo y particionamiento de los datos horizontal. En el Capítulo 5 se hace un análisis de FL en escenarios cross-silo con particionamiento de los datos vertical.

2.4. Particionamiento de los Datos: Horizontal y Vertical

En el contexto de FL, el particionamiento de los datos es un aspecto crucial que afecta significativamente la manera en que se entrena el modelo y se gestionan los datos. Existen dos enfoques principales de particionamiento de datos: horizontal y vertical. Ambos enfoques presentan diferentes desafíos, que tipo de particionamiento se utiliza depende de la naturaleza de los datos, el caso de uso y el contexto de los clientes.

En esta sección se comienza presentando la estructura de los datos para AA, introduciendo los concepto de ejemplos, características y etiquetas . Para luego explorar los dos tipos de particionamiento, horizontal y vertical.

2.4.1. Estructura de los Datos de Entrenamiento

En el AA, los datos constituyen la base sobre la cual se construyen y entrenan los modelos. Para entender cómo funcionan estos modelos, es fundamental comprender la estructura de los datos, específicamente los conceptos de **ejemplos**, **características** y **etiquetas** (Google, 2024).

- Un **ejemplo**, es una unidad individual dentro del conjunto de datos que representa una entidad específica del problema que se está abordando. Cada ejemplo contiene información completa sobre una entidad particular y se utiliza para entrenar el modelo.
- Las **características** son las propiedades o atributos que describen cada ejemplo. Cada característica aporta información específica que el modelo utilizará para aprender patrones y realizar predicciones.
- Una **etiqueta** es el valor que se desea predecir o clasificar para cada ejemplo mediante el modelo de aprendizaje automático.

Los conjuntos de datos se pueden representar en una **tabla** donde las **filas** representan los **ejemplos** y las **columnas** representan las **características**:

- **Filas**: Representan los **ejemplos**.
- **Columnas**: Representan las **características**.

Por ejemplo, consideremos un conjunto de datos para predecir el riesgo de enfermedad cardíaca:

ID	Edad	Género	Colesterol	Presión Arterial	Historial Familiar	Riesgo (etiqueta)
1	55	M	230	140	Sí	Alto
2	40	F	180	120	No	Bajo
3	65	M	250	160	Sí	Alto
4	50	F	190	135	Sí	Medio
5	35	F	175	115	No	Bajo
6	60	M	240	150	No	Alto
7	30	F	180	120	Sí	Bajo

Tabla 2.1: Conjunto de datos para predecir el riesgo de enfermedad cardíaca.

En el ejemplo de la Tabla 2.1:

- Cada fila (ID: 1, 2, 3, 4, 5, 6, 7) es un **ejemplo** que representa a un individuo.
- Las columnas Edad, Género, Colesterol, Presión Arterial, y Historial Familiar son las **características** que describen cada individuo.
- La columna Riesgo es la variable objetivo o **etiqueta** que el modelo intentará predecir.

En el aprendizaje automático tradicional, todo el conjunto de datos está centralizado y accesible para el entrenamiento del modelo. Sin embargo, en escenarios de FL, los datos están distribuidos entre múltiples clientes que no comparten sus datos directamente, dando a lugar a los distintos tipos de particionamientos de datos. A continuación se presentan los enfoques **horizontal** y **vertical**, que determinan cómo se distribuyen las **características** y los **ejemplos** entre los clientes en FL.

2.4.2. Particionamiento Horizontal

El **particionamiento horizontal**, también conocido como particionamiento por ejemplos, se refiere a la división de los datos de tal manera que cada cliente posee un subconjunto de datos. En este esquema, cada cliente tiene acceso a todas las características de un grupo de ejemplos de datos, la figura 2.4 representa este particionamiento de datos.

Por ejemplo, en un escenario en el que varios hospitales desean colaborar para entrenar un modelo de predicción de riesgo de enfermedad cardíaca utilizando historiales médicos. Cada hospital tiene acceso a los registros de sus propios pacientes, es decir, a un conjunto de ejemplos completo, con todas las características (edad, genero, colesterol, etc.), pero limitado solo a sus pacientes. En este caso, los datos están particionados horizontalmente, ya que cada institución posee una fracción de las filas del conjunto de datos global.

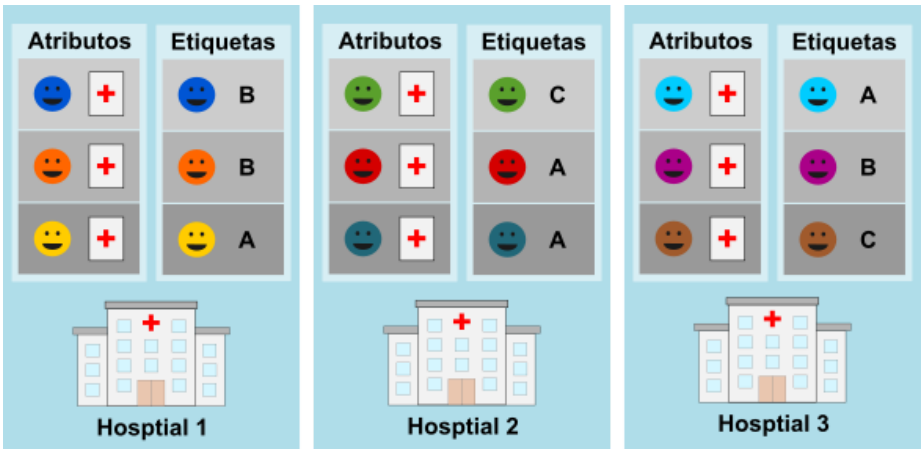

Figura 2.4: Particionamiento Horizontal - Cada cliente posee distintos ejemplos con el mismo tipo de atributos.

Cliente (Hospital) 1						
ID	Edad	Género	Colesterol	Presión Arterial	Historial Familiar	Riesgo (etiqueta)
1	55	M	230	140	Sí	Alto
2	40	F	180	120	No	Bajo
3	65	M	250	160	Sí	Alto

Tabla 2.2: Conjunto de datos para predecir el riesgo de enfermedad cardíaca.

Cliente (Hospital) 2						
ID	Edad	Género	Colesterol	Presión Arterial	Historial Familiar	Riesgo (etiqueta)
3	65	M	250	160	Sí	Alto
4	50	F	190	135	Sí	Medio
5	35	F	175	115	No	Bajo

Tabla 2.3: Conjunto de datos para predecir el riesgo de enfermedad cardíaca.

Cliente (Hospital) 3						
ID	Edad	Género	Colesterol	Presión Arterial	Historial Familiar	Riesgo (etiqueta)
6	60	M	240	150	No	Alto
7	30	F	180	120	Sí	Bajo

Tabla 2.4: Conjunto de datos para predecir el riesgo de enfermedad cardíaca.

Las Tablas 2.2, 2.3 y 2.4 tienen el mismo conjunto de datos que en la Tabla 2.1 pero cada tabla contiene los datos de un hospital distinto.

Al enfoque utilizado al entrenar FL cuando los datos están particionados horizontalmente se lo denomina **Aprendizaje Federado Horizontal** (HFL), es el enfoque utilizado para escenarios cross-device pero también se utiliza para cross-silo. HFL sigue un flujo similar al de un proceso de aprendizaje automático

tradicional, pero distribuido entre múltiples entidades. Cada cliente entrena localmente un modelo utilizando solo los ejemplos a los que tiene acceso, y luego comparte las actualizaciones del modelo (pero no los datos en sí) con un servidor central. Este servidor se encarga de agregar las actualizaciones de todos los clientes para formar un modelo global que represente mejor los datos distribuidos. La figura 2.5 presenta un esquema HFL para predecir el riesgo de enfermedad cardíaca.

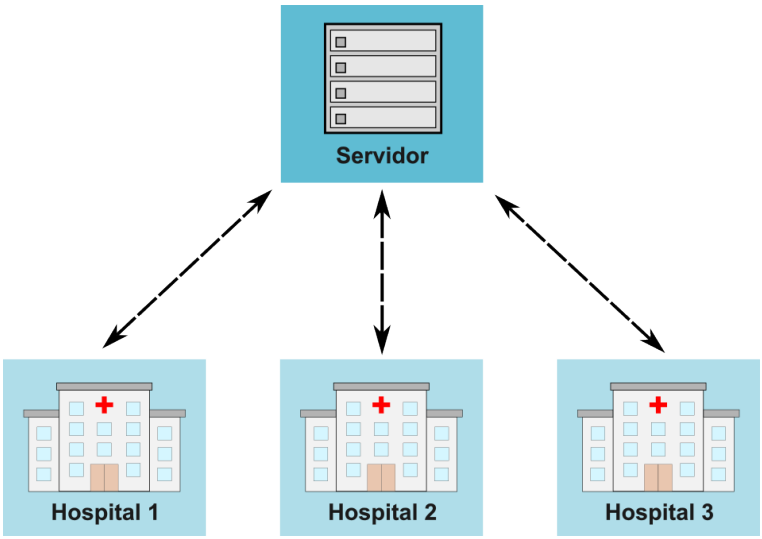

Figura 2.5: Esquema de Aprendizaje Federado Horizontal para predecir el riesgo de enfermedad caríadaca.

Este enfoque permite una colaboración eficaz entre entidades que tienen datos sobre diferentes individuos u objetos, pero todos comparten el mismo conjunto de características. Esto permite a los clientes entrenar localmente modelos completos, que luego se combinan mediante algoritmos de agregación como Federated Averaging, sin necesidad de compartir datos sensibles directamente, lo que preserva la privacidad y cumpliendo con normativas de protección de datos.

En el Capítulo 3 se explora el uso de herramientas para HFL en el contexto de escenarios *cross-device*. En el Capítulo 4 se plantea un caso de uso en LA en el que se aplica FL con un escenario cross-silo y particionamiento de los datos horizontal

2.4.3. Particionamiento Vertical

El **particionamiento vertical**, también conocido como particionamiento por características, se refiere a la división de los datos en columnas, donde cada cliente posee diferentes características de los mismos ejemplos, la figura 2.6 representa este particionamiento de datos.

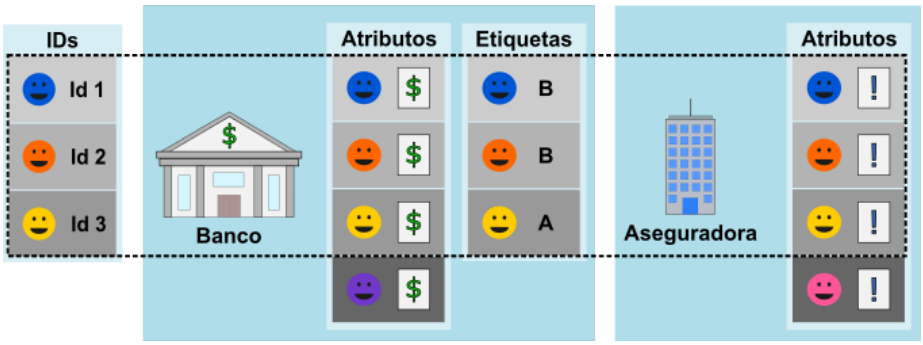

Figura 2.6: Particionamiento Vertical - Cada cliente posee distintos tipos de atributos de los mismos ejemplos.

Por ejemplo, en un escenario en el que una empresa financiera y una compañía de seguros quieren colaborar para construir un modelo predictivo que evalúe el riesgo crediticio de los clientes. La empresa financiera puede tener acceso a las características financieras de los clientes, como ingresos y deuda, mientras que la compañía de seguros tiene acceso a datos relacionados con la salud. Aquí, ambos tienen información sobre los mismos individuos, pero cada uno posee diferentes tipos de características. En este caso, los datos están particionados verticalmente.

Cliente 1 (Empresa Financiera)				
ID	Ingresos Anuales	Deuda	Historial de Pagos	Riesgo Crediticio (etiqueta)
1	$50,000	$10,000	Bueno	Bajo
2	$70,000	$20,000	Atrasado	Alto
3	$45,000	$15,000	Regular	Medio
4	$80,000	$5,000	Excelente	Bajo
5	$60,000	$30,000	Malo	Alto
6	$90,000	$25,000	Excelente	Bajo
7	$55,000	$12,000	Regular	Medio

Tabla 2.5: Conjunto de datos para predecir el riesgo crediticio.

Las Tablas 2.5 y 2.6 contienen un conjunto de datos para predecir el riesgo crediticio de los clientes particionado verticalmente. La Tabla 2.5 contiene los datos de la empresa financiera mientras que la Tabla 2.6 contiene los datos de la aseguradora. Se observa que contienen los mismos ejemplos (identificados por la columna ID, del 1 al 7) con distintas características.

Es importante destacar que no necesariamente se cuenta con los mismos ejemplos desde un principio, la empresa financiera podría tener varios clientes

Cliente 2 (Aseguradora)		
ID	Edad	Historial Médico
1	35	Sin problemas
2	42	Enfermedad Crónica
3	29	Sin problemas
4	50	Sin problemas
5	47	Antecedentes Familiares
6	33	Sin problemas
7	40	Sin problemas

Tabla 2.6: Conjunto de datos para predecir el riesgo crediticio.

que la aseguradora no tiene y viceversa, en esos casos solo se utiliza la intersección de ambos conjuntos de ejemplos para entrenar, ya que se precisa tener todas las características de cada ejemplo. Para realizar esto se utiliza un algoritmo criptográfico denominado PSI que permite encontrar la intersección de los datos sin revelar ninguna otra información información, en la sección 5.2 se presenta PSI en el contexto de un proceso de entrenamiento para redes neuronales con datos particionados verticalmente.

El proceso de aprendizaje en un entorno con particionamiento vertical se denomina **Aprendizaje Federado Vertical** (VFL), este enfoque solo es utilizado para escenarios cross-silo. Este proceso implica la colaboración entre las entidades para entrenar un modelo que integre todas las características sin compartir los datos brutos. Para lograrlo, cada cliente realiza cálculos locales utilizando sus características y envía resultados intermedios (como gradientes parciales o predicciones) al servidor central, o intercambia esta información con otros clientes. El servidor o los clientes luego combinan estas informaciones parciales para ajustar un modelo global que considere todas las características disponibles. La figura 2.7 presenta un esquema VFL para predecir el riesgo crediticio.

VFL es particularmente útil en escenarios donde diferentes organizaciones poseen distintas partes de datos, pero ninguna de ellas tiene una visión completa del conjunto de características necesarias para entrenar un modelo eficaz. Este enfoque permite la colaboración sin comprometer la privacidad de los datos, ya que cada cliente conserva el control de las características que posee y solo comparte resultados intermedios o agregados.

En VFL, el desafío principal es la necesidad de una coordinación y comunicación eficiente entre los clientes para poder reconstruir los ejemplos completos durante el proceso de entrenamiento del modelo, ya que cada cliente cuenta con parte de las características. Esto requiere técnicas avanzadas de criptografía y protocolos de comunicación seguros para garantizar la privacidad y la integridad de los datos.

En el Capítulo 5 se hace un análisis de FL en escenarios con particionamiento de los datos vertical, explorando distintos algoritmos para distintas técnicas de aprendizaje automático.

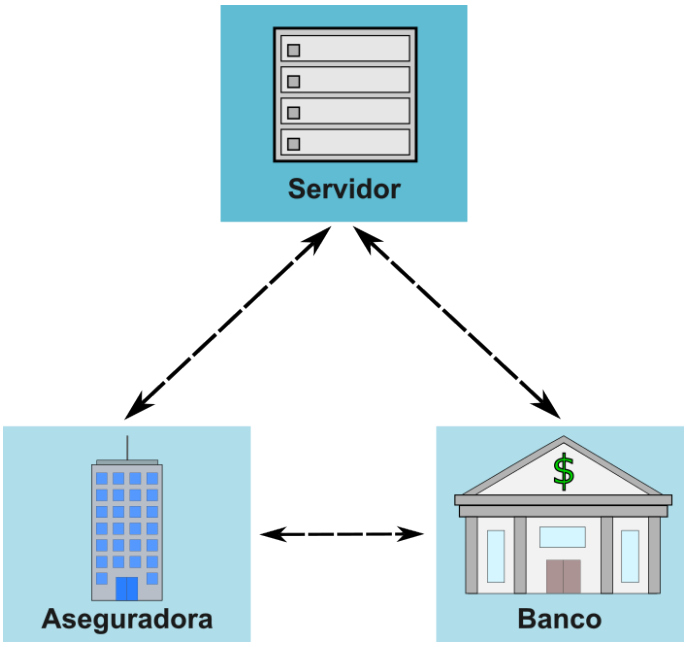

Figura 2.7: Esquema de Aprendizaje Federado Vertical para predecir el riesgo crediticio - Los clientes realizan cálculos locales utilizando sus datos y envía resultados intermedios al servidor central o los intercambia con otros clientes.

2.5. Desafíos de Aprendizaje Federado

FL ofrece beneficios en términos de privacidad, pero también introduce una serie de desafíos que no se encuentran en los enfoques tradicionales de AA (Bharati, Mondal, Podder, y Prasath, 2022). Estos desafíos son críticos para garantizar que los modelos entrenados de manera federada sean efectivos, justos y respeten la privacidad de los datos. En esta sección, se abordan tres de los desafíos más destacados en el ámbito del Aprendizaje Federado: la comunicación, los mecanismos para preservar la privacidad de los datos y la equidad (*fairness*). Cada uno de estos aspectos presenta dificultades únicas que deben ser manejadas cuidadosamente para aprovechar al máximo las ventajas del Aprendizaje Federado sin comprometer la calidad, la equidad y la seguridad de los modelos resultantes.

2.5.1. Comunicación

En el Aprendizaje Federado, la comunicación juega un rol fundamental, ya que los datos permanecen distribuidos en los dispositivos de los clientes, y es necesario intercambiar información entre estos y el servidor central para entrenar el modelo global. A diferencia del aprendizaje centralizado, donde los datos se

recopilan y procesan en un solo lugar, en el Aprendizaje Federado, los modelos se entrenan localmente en los dispositivos de los clientes, y solo los parámetros actualizados del modelo (o sus gradientes) se envían al servidor central. Esto plantea varios desafíos:

- **Volumen de Datos Transferidos:** Durante el proceso de aprendizaje, los parámetros del modelo deben ser comunicados desde y hacia el servidor central en cada ronda de entrenamiento. Si el modelo es grande, el volumen de datos transferidos puede ser significativo, lo que genera una carga de red considerable, especialmente cuando participan miles o millones de dispositivos.

- **Latencia y Ancho de Banda:** La comunicación entre los clientes y el servidor puede verse afectada por la latencia y la disponibilidad de ancho de banda. En entornos con conectividad inestable o limitada, como áreas rurales o en dispositivos con restricciones de datos, la sincronización entre los modelos locales y el servidor puede retrasarse, afectando el ritmo del entrenamiento global y no considerar tales dispositivos puede llevar a sesgos en el modelo.

- **Costo de Comunicación:** En escenarios donde los dispositivos dependen de conexiones móviles o de datos pagados, el costo asociado al intercambio constante de información puede ser elevado. Esto puede desincentivar la participación de algunos usuarios o requerir la implementación de estrategias para minimizar el volumen de datos transmitidos, como la compresión de modelos o el uso de técnicas de comunicación eficiente.

- **Sincronización:** No todos los dispositivos participantes tienen la misma capacidad de procesamiento o acceso a la red, lo que puede llevar a problemas de sincronización. Los dispositivos más lentos, conocidos como "stragglers", pueden retrasar el proceso de agregación de modelos en el servidor, afectando la eficiencia del entrenamiento.

- **Optimización de la Frecuencia de Comunicación:** Encontrar un equilibrio entre la frecuencia de comunicación y la eficiencia del entrenamiento es otro desafío. Comunicar los modelos locales con demasiada frecuencia puede saturar la red y aumentar los costos, mientras que hacerlo con poca frecuencia puede resultar en un modelo global que no converja adecuadamente.

Para abordar estos desafíos, se han propuesto diversas soluciones, como la compresión de gradientes, la actualización asíncrona, y la reducción del número de rondas de comunicación. Sin embargo, la búsqueda de soluciones óptimas sigue siendo un área activa de investigación, dado que la eficiencia de la comunicación es crucial para el éxito del Aprendizaje Federado en aplicaciones del mundo real.

2.5.2. Mecanismos para Preservar la Privacidad de los Datos

La principal razón para utilizar FL es su capacidad para entrenar modelos de aprendizaje automático sin necesidad de centralizar los datos de los usuarios, lo que en teoría permite mantener la privacidad. Sin embargo, este enfoque distribuido introduce nuevos desafíos relacionados con la preservación de la privacidad, ya que, aunque los datos en sí no se comparten, las actualizaciones de los modelos (pesos o gradientes) pueden filtrar información sensible si no se manejan adecuadamente.

Tecnologías

Para mitigar estos riesgos, se pueden utilizar diversos mecanismos de preservación de la privacidad que buscan minimizar la exposición de información sensible mientras se entrena un modelo eficaz. A continuación, se describen algunas de las tecnologías más relevantes (Hosseini, Sikaroudi, Babaei, y Tizhoosh, 2022):

- **Privacidad Diferencial (DP) (Ji, Lipton, y Elkan, 2014)**: Es un framework matemático que garantiza que la salida de un algoritmo (en este caso, la actualización del modelo) no revele información significativa sobre ningún individuo en particular en el conjunto de datos, incluso si se conocen todos los demás datos, a partir de agregar ruido a los mismos. En el contexto de FL (Banse, Kreischer, y i Jürgens, 2024) el servidor recibe de cada cliente actualizaciones de los parámetros a partir de entrenar con sus propios datos, si bien no se manejan los datos crudos se puede inferir información comparando los parámetros anteriores con los nuevos por ejemplo haciendo lo que se conoce como Gradient Inversion citegradient-inversion. Hay múltiples ataques que se pueden realizar (Zhao y cols., 2024) para obtener distinta información, como determinar si un ejemplo fué usado en los datos del entrenamiento (Membership inference), identificar algunas características de los datos (Property inference) y hasta reconstruir los datos completos o una fracción de los datos (Data reconstruction). Para evitar esto con DP se aintroduce ruido aleatorio en las actualizaciones de los modelos antes de enviarlas al servidor, esto dificulta que un adversario pueda inferir información sobre datos específicos del cliente, manteniendo así la privacidad.

- **Secure Multi-Party Computation (MPC) (Yao, 1986)**: Esta tecnología permite a múltiples participantes colaborar para realizar cálculos sobre sus datos combinados sin revelar la información subyacente a ninguna de las partes. En el contexto de FL (Hosseini y cols., 2022), los clientes cifran sus actualizaciones locales antes de enviarlas al servidor. Este último puede calcular la suma o el promedio de las actualizaciones cifradas, pero no puede acceder a las contribuciones individuales de cada cliente.

Este enfoque asegura que el servidor sólo puede ver el resultado agregado, protegiendo así la privacidad de cada cliente.

- **Encriptación Homomórfica (HE) (Cheon, Kim, Kim, y Song, 2017)**: Esta técnica criptográfica permite realizar operaciones matemáticas directamente sobre datos cifrados, sin necesidad de descifrarlos. En el contexto de FL (Fang y Qian, 2021), esto significa que el servidor puede llevar a cabo agregaciones y otras operaciones necesarias en las actualizaciones del modelo sin acceder a los parámetros de los clientes. Aunque es más costosa computacionalmente, ofrece un nivel muy alto de seguridad, garantizando que la privacidad de los datos se mantenga incluso si el servidor central es malicioso.

- **Entornos de Ejecución Confiable (TEEs) (X. Li y cols., 2023)**: Los TEEs son tecnologías que permiten la ejecución segura de código en un entorno aislado, donde ni siquiera el administrador del sistema puede interferir o espiar la ejecución. En el contexto de FL (Mo y cols., 2021), los TEEs permiten que las actualizaciones del modelo se procesen en un entorno seguro, asegurando la integridad y confidencialidad del proceso. Estos entornos garantizan que el estado del código de ejecución se mantenga secreto, protegiendo tanto los datos como los resultados intermedios de posibles ataques o accesos no autorizados.

Cada una de estas tecnologías presenta enfoques diferentes para mantener la privacidad de los datos durante el entrenamiento federado, y su elección dependerá de las necesidades específicas de la aplicación y del nivel de seguridad requerido.

Verificabilidad

No solo alcanza con tener mecanismos para preservar la privacidad de los datos, además se precisa poder garantizar que estos mecanismos están funcionando y que las partes involucradas están siguiendo los protocolos correctamente. De manera de poder asegurar que no existan manipulaciones y compromisos de seguridad que afecten la privacidad de los datos o la accuracy del modelo.

La **verificabilidad** es un aspecto crucial en la preservación de la privacidad dentro del Aprendizaje Federado (Kairouz, McMahan, Avent, y cols., 2021), ya que permite garantizar que los procesos y comportamientos esperados se lleven a cabo correctamente, sin comprometer la confidencialidad de los datos sensibles.

Un mecanismo destacado en este contexto es el uso de **Zero-Knowledge Proofs (ZKPs)** (Sun, Li, y Zhang, 2024). Estas pruebas permiten que una parte (el *prover*) demuestre a otra parte (el *verifier*) que posee cierto conocimiento o que ha realizado un cálculo específico, sin revelar ningún detalle sobre el conocimiento o los datos en cuestión. En el contexto del Aprendizaje Federado, esto podría utilizarse para garantizar que los clientes sigan correctamente el protocolo de entrenamiento y envío de modelos sin revelar sus datos subyacentes, o para verificar el comportamiento del servidor en un entorno de Aprendizaje

Federado (Wang, Dong, Sun, Knottenbelt, y Guo, 2024). Con ZKPs, los clientes pueden estar seguros de que el servidor no ha manipulado los datos ni ha introducido errores en el proceso, todo sin necesidad de que el servidor revele los datos agregados o las actualizaciones individuales. Este enfoque fortalece la confianza en todo el sistema de aprendizaje federado, asegurando que tanto el servidor como los clientes actúan de manera honesta y conforme al protocolo, sin comprometer la privacidad o seguridad de los datos.

Además, los **Entornos de Ejecución Confiable (TEEs)**, mencionados anteriormente, también juegan un papel importante en la verificabilidad. Los TEEs permiten que el código se ejecute de forma segura y aislada, garantizando que ni siquiera el administrador del sistema pueda alterar o inspeccionar el proceso. Esto asegura que los cálculos y las actualizaciones del modelo se realicen de manera verificada, protegiendo al mismo tiempo la privacidad de los datos del cliente.

Estos mecanismos permiten no solo mantener la privacidad de los datos, sino también verificar que los procesos se ejecuten correctamente y de manera segura, sin comprometer la confidencialidad de la información.

Esta es un área activa de investigación, donde se busca constantemente mejorar la seguridad y encontrar el equilibrio adecuado entre privacidad, accuracy del modelo y eficiencia computacional.

2.5.3. Sesgos y Equidad

Los **sesgos** (Mikołajczyk-Bareła y Grochowski, 2023) en el Aprendizaje Automático son desviaciones sistemáticas en los datos o en los modelos que pueden llevar a resultados injustos. Estos sesgos pueden ser el resultado de datos históricos que reflejan desigualdades sociales, económicas o culturales. Por ejemplo, si los datos utilizados para entrenar un modelo contienen un sesgo de género o racial, es probable que el modelo reproduzca o incluso exacerbe ese sesgo, tomando decisiones que discriminen a ciertos grupos de personas.

La **equidad** (o fairness) (Oneto y Chiappa, 2020) en el Aprendizaje Automático tiene como objetivo reducir o eliminar estos sesgos, asegurando que los modelos operen de manera justa y no discriminatoria para todos los individuos o grupos involucrados. Esto puede implicar el ajuste de algoritmos, la modificación de datos de entrenamiento, o la implementación de medidas adicionales para evaluar y corregir posibles desigualdades. A medida que los sistemas de AA se integran más en la toma de decisiones críticas, como la aprobación de créditos, la asignación de recursos médicos o la selección de candidatos para un trabajo, garantizar que estos sistemas no generen ni amplifiquen sesgos existentes se ha convertido en un tema de gran importancia.

En el Aprendizaje Federado (Kairouz, McMahan, Avent, y cols., 2021), los datos no se encuentran en un único repositorio centralizado, sino que están distribuidos entre numerosos clientes, cada uno con su propio conjunto de datos

local. Esta distribución puede resultar en una variedad significativa en la calidad y cantidad de datos entre diferentes clientes.

Por ejemplo, un cliente puede tener datos ricos y bien balanceados, mientras que otro puede tener un conjunto de datos pequeño o sesgado. Este desbalance puede llevar a que el modelo aprenda más de ciertos subconjuntos de la población, lo que genera sesgos y afecta la equidad del modelo para otros grupos.

La disponibilidad de los clientes es otro punto donde se puede generar sesgo. Por ejemplo, si algunos clientes están disponibles en distintos horarios debido a diferentes husos horarios o simplemente distintas rutinas, la participación de estos clientes en las rondas de actualización puede variar significativamente. Esto puede llevar a que los datos de ciertos grupos estén sobrerrepresentados en el modelo, mientras que otros estén subrepresentados, afectando la generalización del modelo.

Además, la capacidad de procesamiento de los dispositivos puede influir en los sesgos. Los dispositivos con procesadores más rápidos pueden completar las tareas de actualización más rápidamente, lo que podría hacer que sus datos se utilicen con mayor frecuencia en el proceso de entrenamiento. Esto podría resultar en un modelo que refleje más las características de los usuarios con dispositivos más modernos, lo cual podría estar correlacionado con factores socioeconómicos.

Evaluar y mejorar la equidad en el Aprendizaje Federado presenta desafíos únicos. En un sistema centralizado, los desarrolladores pueden evaluar el modelo en un conjunto de pruebas diverso para detectar y corregir sesgos. Sin embargo, en un entorno federado, el acceso limitado a los datos completos hace que sea difícil realizar evaluaciones exhaustivas de la equidad. Además, los diferentes participantes pueden tener diferentes definiciones de lo que constituye una salida justa o equitativa del modelo, lo que complica aún más el desarrollo de un modelo que sea percibido como justo por todos.

Para garantizar que los datos recogidos por los clientes sean más diversos, se seleccionan de manera deliberada un conjunto representativo de clientes para las rondas de entrenamiento, asegurando que el modelo no se incline hacia un subconjunto específico de la población.

También se pueden aplicar técnicas de ajuste de pesos que aseguren que las contribuciones de los clientes con menos datos o con datos más diversos tengan un impacto significativo en el modelo final. Esto ayuda a equilibrar la influencia de diferentes grupos y a reducir el sesgo.

Desarrollar y aplicar métricas específicas para evaluar la equidad de los modelos en un entorno federado es esencial. Estas métricas pueden ayudar a detectar sesgos y a guiar el ajuste de los modelos para mitigar la inequidad.

El Aprendizaje Federado introduce nuevos sesgos y desafíos a la hora de intentar mejorar la equidad. En la sección 4.1.2 se realizan experimentos con la distribución de los datos, comparando resultados de un entrenamiento donde los datos son homogéneos con otro entrenamiento en donde los datos son heterogéneos, analizando lo que sucede con los clientes infrarepresentados.

2.6. Herramientas y Datasets

En la actualidad existen múltiples plataformas que permiten implementar soluciones basadas en FL. En esta Sección se presenta un relevamiento de las plataformas existentes y su adecuación al escenario de estudio.

2.6.1. Aspectos Importantes a la Hora de Elegir Herramientas

Existen múltiples consideraciones a la hora de evaluar herramientas disponibles para implementar soluciones FL. Lo primero a tener en cuenta es que se trata de un paradigma que surgió recientemente, del cuál quedan muchos aspectos por investigar. Siendo un área de investigación muy activa, el espectro de soluciones existente cambia constantemente. Al momento de este relevamiento existe una serie de herramientas y plataformas que cuentan con un grado de desarrollo y madurez considerable para HFL.

Existen varios trabajos que reseñan y recopilan herramientas, de los cuales es posible extraer una serie de ejes o aspectos en función de los cuales organizar la comparación (Kairouz, McMahan, Song, y cols., 2021; Zhang y cols., 2021). Se reseñan algunos de los más importantes.

Modelos y algoritmos disponibles Un aspecto importante al tener en cuenta a la hora de seleccionar una plataforma en particular son los algoritmos de AA disponibles en cada solución. Existirán diversas variaciones en los modelos federados dependiendo del caso de uso que se desee implementar, lo que implica que distintas herramientas permitirán resolver distintas aplicaciones. Al igual que en el caso más tradicional se deberá elegir un modelo para el aprendizaje, con la diferencia que no todos los modelos seguirán sirviendo, aún hay muchos que no fueron adaptados al contexto federado y algunos directamente no son compatibles. Asimismo, existen diferentes algoritmos que pueden usarse para que el servidor agregue los resultados parciales. El más común es Federated Averaging (B. McMahan, Moore, Ramage, Hampson, y y Arcas, 2017), pero la disponibilidad de otros algoritmos implementados puede ser un criterio relevante a la hora de elegir y evaluar las plataformas existentes.

Esquema de comunicación/distribución Otro punto que tomar una decisión sobre el tipo de comunicación que se tendrá, pudiendo variar entre lo que se conoce como Aprendizaje Federado Vertical y Horizontal. En el caso horizontal, existe un cierto grado de solapamiento entre las características de los datos repartidos entre varios nodos, mientras que los datos son bastante diferentes en el espacio muestral. La solución de modelo federado para la actualización del teclado predictivo en teléfonos móviles Android planteada por Google es un ejemplo de FL horizontal, ya que los datos tienen las mismas dimensiones o variables.

El caso vertical es adecuado cuando los datos se particionan en sentido vertical según sus variables o características, es decir que cada cliente tiene información sobre variables diferentes para los mismos elementos de la realidad. Para este enfoque no hay actualmente versiones de todos los modelos, el más estudiado es la regresión logística (Zhang y cols., 2021) .

Madurez de las herramientas Los aspectos vinculados al uso y madurez de las herramientas, así como los requerimientos no funcionales que puedan existir, también son relevantes a la hora de seleccionar herramientas y plataformas. A modo de ejemplo se mencionan:

- La facilidad de utilizar determinado framework
- La existencia de tutoriales y ejemplos que faciliten la comprensión y el desarrollo de prototipos
- La información y documentación técnica disponible
- La existencia de medios de comunicación disponibles con los equipos de desarrollo de las herramientas para poder evacuar dudas
- Los planes a futuro del equipo de desarrollo, tanto para nuevas funcionalidades como para resolver errores existentes.

Cabe mencionar que en la mayoría de los casos, las herramientas disponibles son de código abierto.

En resumen, las soluciones basadas en FL involucran un amplio espectro de problemas y áreas de investigación, no sólo vinculadas al AA si no también a aspectos de comunicación de datos, privacidad en el cómputo y encriptado (como la aplicación de técnicas de Multi Party Computation o Differential Privacy). involucrando aspectos como la no disponibilidad de los datos por parte de algunos de los clientes en ciertos momentos, sesgos en los modelos generados, entre otros. Las funcionalidades provistas en cada uno de estos sentidos por las herramientas disponibles es muy variable, y evoluciona con el tiempo. Es importante establecer un marco de comparación mínimo para evaluarlas.

2.6.2. Herramientas Relevadas

En esta sección se presentan algunas herramientas relevadas, analizando los aspectos mencionados en la sección anterior. Además, se compila una serie de conjuntos de datos que típicamente son utilizados para evaluar y armar ejemplos de aplicación de FL. Estos conjuntos de datos, típicamente analizados usando redes neuronales, son en muchos casos estándares de facto para algunos problemas clásicos (por ejemplo, clasificación de imágenes)

TensorFlow Federated

TensorFlow Federated (TFF) (Google Brain Team, 2021j) (Google, 2021) es un framework desarrollado por Google basado en TensorFlow (Google Brain Team, 2021l). Al momento del relevamiento no se pueden utilizar múltiples dispositivos y ejecutar el algoritmo de manera distribuida tal como se haría en un escenario real, sino que la parte federada se hace en una sola máquina mediante simulaciones que soportan un gran volumen de clientes y datos.

TFF está organizado en dos capas Federated Learning (Google Brain Team, 2021e) y Federated Core (Google Brain Team, 2021d). La primera cuenta con interfaces de alto nivel que proveen herramientas para entrenar y evaluar con aprendizaje federado, las mismas incluyen los algoritmos federados. TFF permite el uso de modelos Keras (MIT, 2021) por medio de una función ya implementada, pero también cuenta con una interfaz que se puede adaptar para usar cualquier otro modelo. La segunda capa, Federated Core, provee interfaces de bajo nivel que permiten crear algoritmos federados personalizados.

En la versión relevada en este trabajo TFF no cuenta con mecanismos extra de aseguramiento de la privacidad y solo tiene una función de agregación disponible en la API de Federated Learning, Federated Averaging (Google Brain Team, 2021c), aunque tiene la posibilidad de implementar más mediante Federated Core.

TFF incluye módulos que permiten probar el escenario federado con diferentes conjuntos de datos (Google Brain Team, 2021b). Dentro de los conjuntos disponibles se destacan Federated CIFAR-100, Federated EMNIST, Federated Google Landmark v2, Shakespeare y StackOverflow, mencionados previamente.

Esta plataforma viene acompañada de una extensa documentación (Google Brain Team, 2021a), junto con guías (Google Brain Team, 2021g) y diversos tutoriales en formato de notebooks (Google Brain Team, 2021i) que facilitan su uso. Tiene un equipo de desarrolladores y contribuidores que están activos trabajando en mantener y seguir incorporando funcionalidades a la aplicación, además de contar con foros tanto en su página (Google Brain Team, 2021k) como en plataformas como Stack Overflow (Google Brain Team, 2021h) y en el repositorio de GitHub una entrada para reportar problemas o *issues* (Google Brain Team, 2021f).

En el Capítulo 3 se utiliza para realizar un estudio de factibilidad y comparación de herramientas. En el Capítulo 4 se utiliza en el contexto de LA para predicción de abandono.

Flower

Flower (Beutel y cols., 2020) es un entorno de trabajo para FL diseñado para experimentar nuevas soluciones y algoritmos en sistemas en FL. Este entorno, surgido del ámbito académico, ofrece una implementación de los componentes principales de un sistema FL, y proporciona abstracciones de alto nivel para permitir a los investigadores experimentar e implementar nuevas ideas. Es posible integrar sobre Flower implementaciones realizadas con diferentes bibliotecas

de AA, como por ejemplo PyTorch, TensorFlow y NumPy entre otras. Además, Flower permite migrar procesos de entrenamiento de AA existentes a una configuración basada en FL. Este entorno fue diseñado para soportar múltiples tipos de clientes, móviles e inalámbricos, con recursos heterogéneos de computación, memoria y red.

Este proyecto se encuentra muy activo, cuenta con buena documentación disponible en su sitio web (Flower, 2022a) y código disponible en su repositorio público (Flower, 2022b).

En el Capítulo 3 se utiliza para realizar un estudio de factibilidad y comparación de herramientas.

Conjuntos de Datos para Evaluación

Es importante contar con un buen abanico de conjuntos de datos que permitan simular distintos casos de uso y así experimentar con distintos modelos. En el contexto de este nuevo paradigma de aprendizaje federado es especialmente importante, tanto a la hora de desarrollar nuevas herramientas y modelos como para evaluar las existentes. Estos conjuntos de datos permiten simular un entorno controlado reduciendo la cantidad de interrogantes, y así enfocarse en evaluar y probar los aspectos estrictamente vinculados al despliegue federado, y comparar las soluciones federadas con mecanismos tradicionales. A continuación se describen algunos de los conjuntos de datos que suelen usarse.

EMNIST EMNIST (United States Goverment, 2021) (Cohen, Afshar, Tapson, y van Schaik, 2017) o Extended MNIST es un conjunto de datos creado a partir del dataset MNIST (Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond, 2021), compuesto por 671,585 imágenes de letras y números, los cuales fueron escritos a mano y luego convertidos en imágenes de 28x28 píxeles. Se puede acceder a todos los datos o separarlos en dígitos y letras. En la Sección 3.1 se utiliza para simular un entorno controlado y comparar resultados obtenidos con distintas herramientas de FL y un modelo tradicional.

CIFAR-10 y CIFAR-100 CIFAR-10 y CIFAR-100 (Alex Krizhevsky, 2021) (Krizhevsky, Hinton, y cols., 2009) recopilan y clasifican algunas de las imágenes que contiene el 80 conjunto de datos "80 million tiny images". Este último conjunto de datos se ha retirado de la web debido a que se ha encontrado que contenía imágenes ofensivas (Birhane y Prabhu, 2021) CIFAR-10 cuenta con 60000 imágenes de 32x32 píxeles clasificadas en 10 tipos de objetos distintos. CIFAR-100 sigue una idea similar teniendo la misma cantidad de imágenes pero esta vez etiquetadas según 100 tipos de objetos. En la Sección 3.1 se utiliza este dataset para probar distintos hiperparámetros en un ambiente controlado.

Shakespeare Este conjunto de datos, publicado en la plataforma Kaggle, contiene transcripciones de todas las obras de William Shakespeare (Kaggle, 2016).

Contiene todas las frases, indicando para cada una la obra a la que pertenece, el lugar dentro de la obra y el personaje que le corresponde.

Stack Overflow El conjunto de datos de Stack Overflow (Stack Overflow, 2018) contiene publicaciones, comentarios, votos, etiquetas, usuarios, y otros elementos de la plataforma . Está basado en parte de un conjunto más grande de Stack Exchange que contiene una recopilación similar a lo comentado anteriormente pero de toda la red de Stack Exchange y los distintos sitios que forman parte.

Google Landmark v2 Google Landmark v2 (Weyand, Araujo, Cao, y Sim, 2020) es la segunda versión de un conjunto de datos que contiene imágenes de paisajes, etiquetadas según si se trata de un paisaje natural o uno creado por el ser humano. En este caso se cuenta con 5 millones de imágenes de 200.000 paisajes distintos.

LEAF LEAF (Caldas y cols., 2018) es una plataforma de benchmarking para escenarios federados. Su objetivo es recopilar conjuntos de datos preprocesados para facilitar su uso en el contexto federado. Está diseñado para que se pueda utilizar junto con TensorFlow y NumPy. Este benchmark consta de seis conjuntos de datos diferentes, etiquetados por distintos usuarios simulando aplicaciones de clasificación de imágenes, predicción de caracteres, análisis de sentimiento, clasificación y procesamiento de lenguaje. Entre estos conjuntos de datos se incluye FEMNIST: una versión federada de MNIST donde se particiona el conjunto original separándolo en 3500 clientes según el autor original de cada trazo, así como una versión federada del conjunto de datos Shakespeare. Tiene un propósito muy específico lo que lo convierte en una herramienta sencilla de utilizar en combinación con otros frameworks. En el sitio web del proyecto hay una guía para instalarlo, tutoriales, documentación de la API (LEAF, 2021). El código es abierto y está alojado en un repositorio GitHub de acceso público (LEAF, 2022).

Capítulo 3

Aprendizaje Federado Horizontal - Estudio de Factibilidad y Comparación de Herramientas

En este capítulo se presenta un estudio de factibilidad y comparación de herramientas realizado en el contexto del proyecto ANII - FLEA: Aprendizaje Federado aplicado a LA (Bermolen y cols., 2022). Se presenta la metodología de los casos de estudio, introduciendo los datasets utilizados, los problemas de clasificación a resolver y el enfoque utilizado en cada caso. Luego se presentan los resultados y análisis obtenidos en cada caso. Por último se presentan conclusiones a partir de los resultados. El primer caso de estudio se realiza para validar el funcionamiento de las herramientas TFF y Flower, y del esquema federado, para este propósito se realizan los experimentos sobre el dataset MNIST, el mismo no presenta grandes dificultades lo que permite fácilmente comparar métricas entre el esquema centralizado y el esquema federado, tanto con el uso de TFF y Flower, además de poder conocer las herramientas y su facilidad de uso. Una vez probadas las herramientas y validado el funcionamiento de las mismas y del esquema federado, se elige la herramienta TFF, con la que se realiza un segundo caso de estudio. Este segundo caso de estudio es un poco más complejo, se desea realizar pruebas controladas pero un poco más desafiantes, para esto se utiliza el dataset CIFAR10 y se realizan varios experimentos con distintos hiperparámetros, probando como afecta el número de clientes, la cantidad de rondas y de épocas.

3.1. Metodología de los Casos de Estudio

El estudio en ambos casos se divide en distintas etapas. Primero se despliega una solución centralizada para el problema definido en cada caso, implementada en Tensorflow, para ser usada como línea base. En segundo lugar se despliegan las soluciones federadas, en el primer caso un solucion usando TFF y otra usando Flower, y para el segundo caso una usando TFF.

Se realiza la misma cantidad de ejecuciones total en ambos casos (en el federado y en el centralizado), de forma que las comparaciones sean justas. Los resultados se centran en: comparar el desempeño y tiempo entre cada solución federada y el modelo centralizado y en comparar desempeño y tiempo entre ellas.

La métrica que se utiliza para evaluar el desempeño es *accuracy*, típicamente utilizada en problemas de clasificación. Esta métrica computa el porcentaje de observaciones que el modelo clasifica correctamente, y se calcula en base a los resultados de la ejecución del modelo sobre un conjunto de datos que no fue utilizado en el proceso de entrenamiento, usualmente denominado datos de evaluación o *test*. Además, se miden los tiempos de entrenamiento en cada caso.

Los experimentos se realizan de la siguiente manera, se eligen los parámetros a probar en el contexto federado: los clientes, la cantidad de rondas, la cantidad de clientes por ronda y la cantidad de épocas. Luego, a partir de la cantidad de rondas y épocas, se determina el "número de épocas totales" que corre ese experimento, de manera de poder compararlo con el algoritmo centralizado ejecutando ese número de épocas totales.

Debido a que en el contexto federado entran en juego las rondas, hay que tener en cuenta que para un mismo experimento centralizado hay varios experimentos similares que se pueden comparar en el contexto federado. A modo de ejemplo, si en un experimento centralizado se corren 20 épocas, entonces dos experimentos federados similares podrían ser 20 rondas y 1 época o 10 rondas y 2 épocas; En los tres experimentos se toma como que se corren 20 épocas totales. En cada caso se presentan primero los resultados del caso centralizado, que sirven de referencia y a continuación de cada uno se dan los resultados del algoritmo federado.

A continuación se detallan los problemas de clasificación a resolver y el enfoque seguido en cada caso.

Caso MNIST

El conjunto de datos elegido es una de las versiones del conjunto MNIST (Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond, 2021), constituida por imagenes de dígitos del 0 al 9 escritos a mano. En la Figura 3.1 se muestran algunos ejemplos. En este conjunto las imágenes miden 28×28 pixeles y están en blanco y negro, por lo que cada una se representa por medio de una matriz de dimensión 28×28 codificando con 0 o 1 el blanco o negro.

Figura 3.1: Ejemplo de algunos dígitos del conjunto MNIST.

Los datos son extraídos usando la biblioteca `keras` de Python, a partir del módulo `keras.datasets` [1]. En estos datos hay 60.000 imágenes destinadas al conjunto de entrenamiento y 10.000 para el conjunto de test. La distribución de los dígitos del 0 al 9 en los conjuntos de entrenamiento y de test puede verse en la Figura 3.2 y la Figura 3.2.

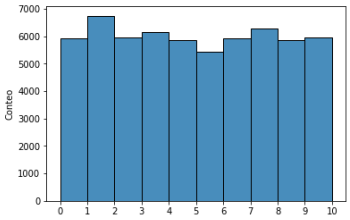

Figura 3.2: MNIST: Distribución por dígito en el conjunto de entrenamiento.

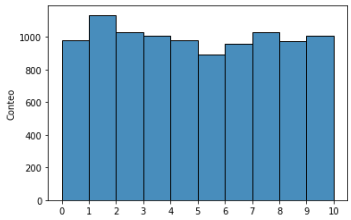

Figura 3.3: MNIST: Distribución por dígito en el conjunto de test.

Se experimenta con una arquitectura de red convolucional (CNN, por sus siglas en inglés), que es de uso común en los problemas de imágenes. Se tienen dos bloques de convolución, cada uno con su capa de convolución seguido de una de pooling. Sobre estas hay una capa densa de 32 neuronas y por último la capa de salida de 10 neuronas (una por clase). La Figura 3.4 representa gráficamente la arquitectura de la red.

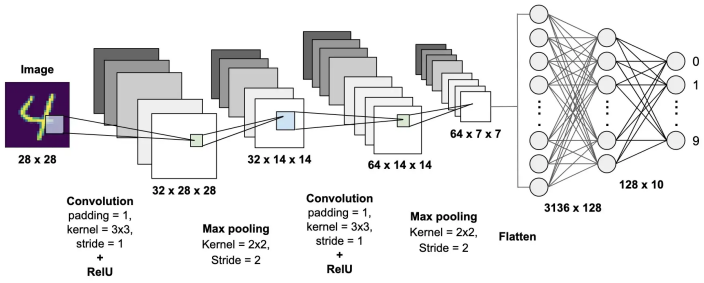

Figura 3.4: Arquitectura de la red convolucional usada para clasficar los dígitos de MNIST (Krut Patel, 2019).

[1] Los detalles sobre este módulo pueden consultarse en https://keras.io/api/datasets/mnist/

Esta es la arquitectura utilizada en la implementación centralizada así como en las implementaciones federadas utilizando TFF y Flower.

Caso CIFAR10

En el caso de CIFAR10 los datos también son extraídos usando la biblioteca `keras` de Python y el módulo `keras.datasets`. El conjunto de entrenamiento tiene 50.000 elementos y el de test 10.000. Las imágenes tienen en este caso 32×32 pixeles, pero a diferencia de MNIST están en colores. Cada color se representa con 3 canales y por lo tanto cada una de estas imágenes tiene dimensión (32, 32, 3).

Las imágenes se clasifican en 10 clases donde 4 corresponden a medios de transporte (aviones, autos, barcos y camiones) y 6 a animales (pájaros, gatos, ciervos, perros, ranas y caballos). En la Figura 3.5 pueden verse algunas imágenes de ejemplo.

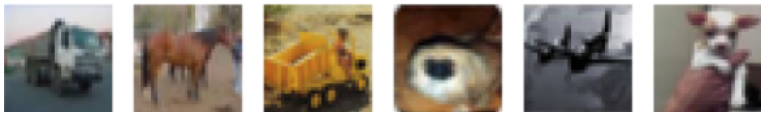

Figura 3.5: Ejemplo de algunas imágenes del conjunto CIFAR10.

Existe la misma cantidad de elementos de cada clase, tanto en los conjuntos de entrenamiento como de test, por lo que la distribución es uniforme tal como se muestra en la Figura 3.6 y la Figura 3.7

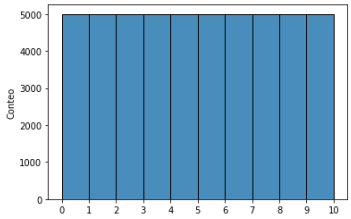

Figura 3.6: CIFAR10: Distribución en el conjunto de entrenamiento.

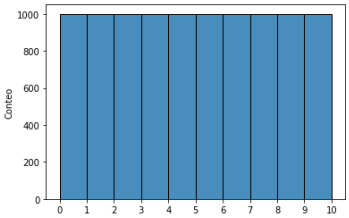

Figura 3.7: CIFAR10: Distribución en el conjunto de test.

En este caso se sigue trabajando con imgenes y se vuelve a utilizar un modelo convolucional, del mismo tipo que se usa para MNIST, pero con más capas de convoluciones y una capa densa con más neuronas.

3.2. Resultados y Análisis del Caso MNIST

En primer lugar se entrena la red de manera centralizada. Se intenta entender cuál es la cantidad óptima de épocas, es decir cuál es la cantidad mínima

de épocas necesarias para alcanzar un buen desempeño, y a partir de la cuál el modelo no mejora en el conjunto de test, incluso aunque lo haga en el de entrenamiento (*overfitting*). Con este fin se entrena 50 épocas, luego de cada época se calculan las métricas tanto en el conjunto de entrenamiento como en el de test. La Figura 3.8 presenta los resultados obtenidos, donde se puede ver que la mayor parte del aprendizaje toma lugar en las primeras épocas, y luego el desempeño se estanca. Se toma como punto de corte la época 20, en la que se logra una *accuracy* de 0.99 en test en un tiempo de 240 segundos.

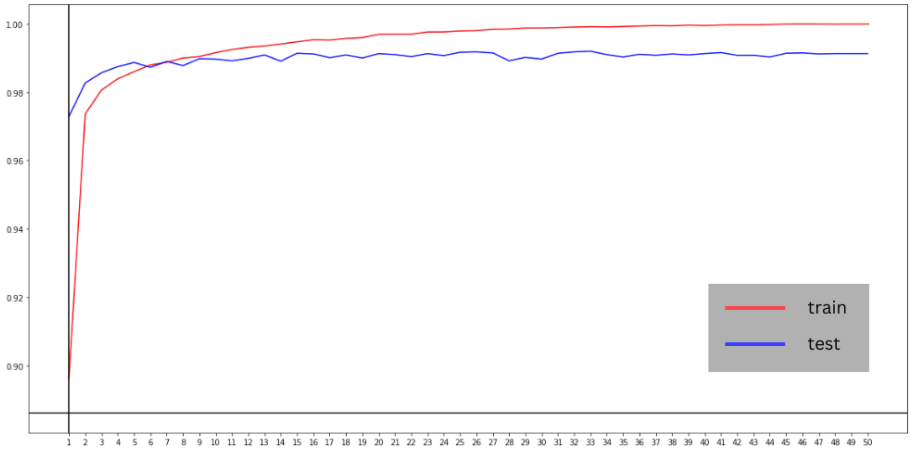

Figura 3.8: MNIST: Comparación de *accuracy* entre entrenamieniento y test en el caso centralizado. Épocas en el eje de las abscisas, la métrica en las ordenadas.

Para el caso federado se utiliza un esquema de entrenamiento con 10 clientes, durante 20 rondas, en las cuales participan siempre todos los clientes, cada uno entrenando durante una época local, dando un total de 20 épocas totales por cliente. Se obtiene una *accuracy* de 0.98 % con TFF y una de 0.99 % en Flower.

3.3. Resultados CIFAR10

Al igual que en el caso MNIST primero se intenta entender cuál es la cantidad óptima de épocas. Se entrena el modelo centralizado durante 120 épocas con el fin de encontrar la cantidad óptima. La gráfica de la Figura 3.9 muestra la métrica de *accuracy* en el conjunto de entrenamiento y de test. Se toma como punto de corte la época 60, donde se obtiene una *accuracy* de 78 % en test en un tiempo de 6300 segundos.

Se fija entonces en 60 el número de épocas totales, y se realizan ejecuciones con distintas cantidades de rondas y épocas para evaluar el caso federado, se cambian además el número de clientes: primero un escenario con 2 clientes y luego con 7. En todos los casos los datos originales del conjunto de entrenamiento se particionan en tantas particiones como cantidad de clientes. La nomenclatura

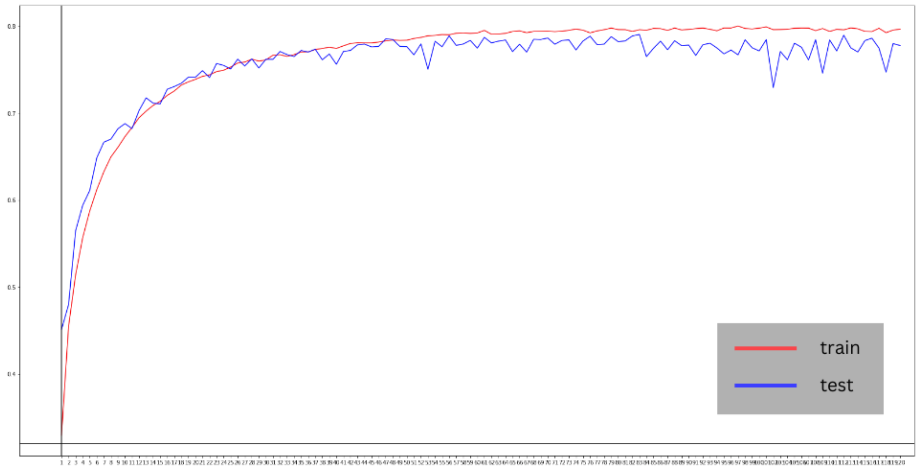

Figura 3.9: CIFAR10: Comparación de *accuracy* entre entrenamieniento (rojo) y test (azul) en el caso centralizado. Épocas en el eje de las abscisas, la métrica en las ordenadas.

x/y clientes significa que para ese experimento se seleccionaron aleatoriamente en cada ronda x clientes de los y totales. Se resumen los resultados en la Tabla 3.1.

Clientes	Rondas	Epocas	Accuracy	Tiempo(s)
2/2	15	4	78 %	5387
2/2	30	2	77 %	5335
2/7	15	4	66 %	1564
2/7	30	2	67 %	1617
5/7	15	4	69 %	3483
5/7	30	2	69 %	3550

Tabla 3.1: CIFAR10 con TFF. Accuracy y tiempo de cada ejecución variando cantidad de clientes, rondas y épocas. La nomenclatura x/y clientes significa que para ese experimento se seleccionaron aleatoriamente en cada ronda x clientes de los y totales.

3.4. Conclusiones

Se presentan las conclusiones para ambos casos, MNIST y CIFAR10.

MNIST

En cuanto a los resultados asociados al caso MNIST, tanto en TFF como en Flower se obtienen resultados similares en *accuracy* y en tiempo. Además, se consiguen en un tiempo menor que en la contraparte centralizada. Por lo tanto se valida el buen funcionamiento de ambas herramientas y se concluye que este problema se puede federar sin pérdida de desempeño. Además podemos concluir que la dificultad en implementar este caso siguiendo el esquema federado en ambas plataformas fue similiar, y también es similar el comportamiento de ambas, por esta razón se sigue experimentando solo con TFF.

CIFAR10

En primer lugar se puede ver que si bien el escenario federado desempeña peor que el centralizado, no está lejos incluso alcanzando los mismos resultados con algunas configuraciones de hiperparámetros. A su vez los tiempos están por abajo del tiempo que tomó en el escenario centralizado, acercandose en las ejecuciones que dieron resultados parecidos.

Respecto a la diferencia en los resultados respecto a ejecuciones con distinta cantidad de clientes se puede observar que utilizando una mayor fracción de clientes se obtienen mejores resultados. Esto se muestra en la Tabla 3.1, en los experimentos con 2/7 clientes comparados con los de 5/7 clientes, o los de 2/2 clientes. Esto es debido a que se están utilizando menos datos con una fracción menor de clientes (recordar que los datos se distribuyen entre la cantidad de clientes considerada en cada caso).

Por último, en cuanto a la cantidad de rondas y cantidad de épocas, comparando ejecuciones con la misma cantidad de clientes, en las dos implementaciones parece ser que es indistinto aumentar la cantidad de rondas y disminuir las épocas locales en cada cliente, o viceversa, siempre y cuanto se mantenga fija la cantidad total de épocas. La decisión entonces se debería tomar según las restricciones del caso particular tomando en cuenta las consecuencias de cada configuración, por ejemplo, que el cliente debe estar disponible durante el entrenamiento, aumentar la cantidad de épocas locales implícitamente dice que se esperan clientes que no se den de baja rápidamente.

Capítulo 4

Caso de Aprendizaje Federado Horizontal en Analíticas de Aprendizaje

En el Capítulo 3 se presentan casos sencillos para estudiar la factibilidad de HFL y comparar y validar el uso de las herramientas TFF y Flower. En este capítulo se presenta un caso de estudio de HFL en LA presentado en (Fachola y cols., 2023) más exigente sobre el cuál se realizan experimentos que pretenden comparar performance entre el esquema tradicional y el federado. Se realizan dos tareas: la predicción del abandono escolar y la clasificación no supervisada de estudiantes.

En la Sección 3.1 se realizan pruebas que muestran que con diferentes hiperparámetros y distintas decisiones de diseño a tomar afectan el desempeño de los modelos federados, como son la cantidad de rondas, la cantidad de clientes a seleccionar por ronda, cómo seleccionar los clientes, cuántas épocas entrenar localmente el modelo en cada cliente, etc. Por esta razón se ejecuta un mismo algoritmo de FL en cada modelo predictivo con distintas configuraciones de hiperparámetros con el fin de medir el desempeño en cada caso, tanto en cuanto a calidad de las predicciones (evaluando métricas estadísticas), como en cuanto al tiempo que tarda en entrenarse el modelo en cada caso.

4.0.1. Caso de Estudio y Conjuntos de Datos

Independientemente del esquema de gobierno y organización del sistema educativo de cada país, es habitual que existan entidades gubernamentales por encima de los centros escolares. Una de las principales tareas que llevan a cabo estas instituciones es la recopilación y el análisis de datos del sistema educativo. En este contexto, se ve una clara oportunidad para aprovechar las ventajas de FL. Los sistemas educativos suelen estar compuestos por diferentes centros educativos. En cada centro participan alumnos y profesores que interactúan

diariamente en diversas actividades de aprendizaje. Toda la interacción en el proceso de aprendizaje genera información valiosa, tanto a nivel local para cada centro como a nivel global para todo el sistema educativo. Cuando estas interacciones se producen a través de plataformas educativas electrónicas, se genera un volumen potencialmente masivo de datos que puede ser aprovechado para diversos fines académicos y pedagógicos.

El objetivo es proporcionar mecanismos que permitan estudiar la información generada en cada centro educativo de forma global evitando compartir los datos crudos generados en cada centro. Este esquema mejora la gestión de los datos en términos de preservación de la privacidad. Se evalúa el impacto que esto tiene en los resultados del análisis de los datos. El primer paso para lograr los objetivos es definir un marco de aprendizaje federado adecuado que se adapte adecuadamente a las relaciones típicas entre escuelas y una forma conveniente de evaluar los resultados.

Se pueden diferenciar dos escenarios de aprendizaje federado: cross-device y cross-silo, en la Sección 2.3 se desarrolló más de ambos escenarios. El escenario cross-device corresponde al entrenamiento entre miles de dispositivos y presenta problemas de comunicación que juegan un papel relevante en este caso, ya que los dispositivos sólo están disponibles en ocasiones, lo que dificulta las rondas de entrenamiento de los modelos de aprendizaje automático. El segundo corresponde al intercambio de información entre distintas instituciones, que suele implicar la comunicación de datos entre centros de datos bien establecidos. En cuanto al caso educativo mencionado parece claro que se corresponde con un escenario cross-silo

En el caso de escenarios cross-silo se pueden distinguir dos tipos de particionamiento de datos: horizontal y vertical, en la Sección 2.4 se presentó en más profundidad estos tipos de particionamiento. En el particionamiento horizontal cada institución tiene su propio conjunto de datos, pero todos los conjuntos comparten los mismos atributos o variables. Los registros de cada conjunto de datos tienen los mismos campos pero para distintos participantes. Por el contrario, en la partición vertical diferentes conjuntos de datos comparten identificadores comunes (por ejemplo, información de los mismos usuarios). Sin embargo, cada conjunto de datos incluye campos diferentes en sus registros. Un caso típico de esto último podría ser un esquema de cross-silo en el que distintos organismos gubernamentales comparten información sobre sus ciudadanos.

En la propuesta, cada centro educativo gestiona toda la información relativa a sus profesores y alumnos. Esta situación no implica necesariamente que cada centro educativo tenga su centro de datos on-premise. También se podría tener plataformas educativas en la nube, donde los datos se alojan en servidores de terceros. Sin embargo, se supone que cada centro educativo tiene los derechos de administración de todos los datos de sus profesores y alumnos. Así, cada centro del sistema educativo se corresponde con un silo en el esquema federado propuesto. A su vez cada centro tiene el mismo tipo de datos, por lo que estamos hablando de HFL.

En la Figura 4.1, se ilustra el esquema cross-silo propuesto y su aplicación en el sistema educativo.

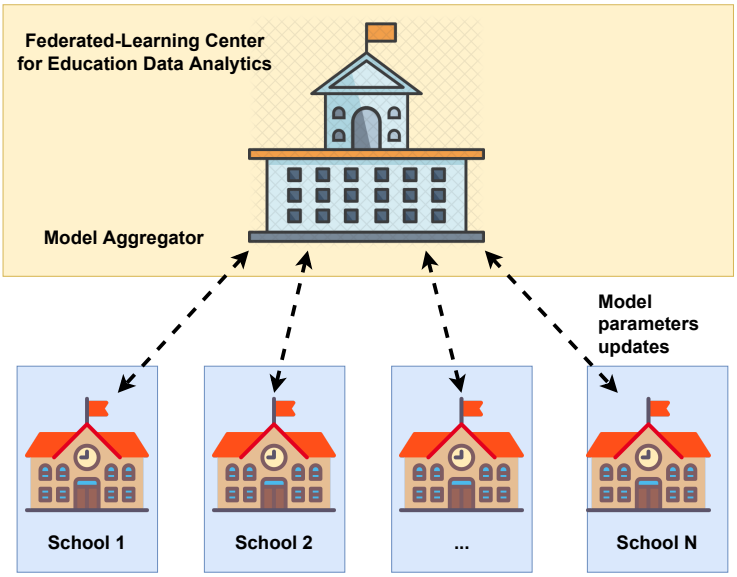

Figura 4.1: Esquema FL cross-silo propuesto en (Fachola y cols., 2023) para el análisis de información centralizado del sistema de educación.

El objetivo es utilizar el paradigma del aprendizaje federado para permitir un análisis centralizado de los datos del sistema educativo, evitando al mismo tiempo la correspondiente centralización de los datos crudos. Esto permite a las instituciones gubernamentales superiores encargadas del sistema educativo llevar a cabo análisis de datos empleando modelos de aprendizaje automático, preservando al mismo tiempo la privacidad de los profesores y estudiantes implicados.

Conjunto de Datos

Se utilizan datos públicos de educación utilizados en un desafío de la KDD-Cup 2015 (KDD, 2015). El conjunto de datos KDDCup2015 contiene registros de actividad de XuetangX, una plataforma china de aprendizaje MOOC (Massive Online Open Course). Se proporciona información sobre el curso y la actividad de los estudiantes a lo largo del tiempo. La información de los estudiantes incluye un registro de participación en diversos aspectos de cada curso (foro de debate, cuestionario, uso de medios, etc.). Hay 21 aspectos o actividades definidos, y su disponibilidad varía según los cursos, que se identifican por *course_id*. Además, se proporciona un ID de estudiante que puede utilizarse para vincular los registros de un estudiante determinado a través de los cursos. Esto puede utilizarse para calcular métricas como las tasas de finalización a nivel de estudiante en los distintos cursos. Los registros tienen 42 millones de entradas individuales y un tamaño total de aproximadamente 2,1 GB. Hay aproximadamente 77.000

estudiantes distintos y 247 cursos.

Se realizan tareas de preparación de datos que transforman las entradas individuales de los registros de actividad en bruto, agregando datos para cada actividad, curso y nombre de usuario y contando el número de entradas para cada grupo. El resultado final es una matriz 225642×21 en la que cada entrada corresponde a un par distinto $(course\_id, username)$, que también se identifica por un número $enroll\_id$. Las características son el número de actividades realizadas por $enroll\_id$. El código de preparación de datos está disponible en el repositorio del proyecto FLEA (FLEA, 2022).

4.1. Predicción de Abandono

Para esta tarea se utiliza el enfoque presentado en (Feng, Tang, y Liu, 2019) para predecir el abandono de los estudiantes. Para cada $enroll\_id$ en el conjunto de datos, se sabe si el estudiante abandonó el curso. Estas etiquetas se utilizan para entrenar y probar un modelo de deep learning que predice el abandono. Se utiliza una arquitectura CNN que consiste en una capa de entrada, 3 capas ocultas de tamaño 100, y una capa de salida de 1 neurona con una sigmoide como función de activación. Se utiliza el optimizador Adam (Kingma y Ba, 2014) y la entropía cruzada binaria como función de pérdida.

Los experimentos tienen dos objetivos principales 1) evaluar si los modelos federados pueden alcanzar o no la accuracy de la configuración centralizada, y 2) evaluar la influencia de los parámetros sobre la accuracy y el tiempo total de entrenamiento de los modelos federados. En primer lugar, se despliega una solución centralizada utilizando Tensorflow, que funciona como línea base para los modelos federados. Para federar, se utiliza el algoritmo Federated Averaging (B. McMahan y cols., 2017) implementado en TFF. Nótese que además de los parámetros locales habituales de cada cliente, como las épocas y el tamaño de lote, en la versión federada, se necesitan manejar parámetros adicionales, como el número de rondas de entrenamiento por cliente, el número de clientes elegidos en cada ronda, el número total de clientes y cómo se distribuyen los datos entre ellos.

4.1.1. Experimentos con Hiperparámetros

Se entrena un modelo centralizado durante 20 épocas. El número de épocas se elige empíricamente; se busca un número lo suficientemente grande como para permitir el ajuste de parámetros del enfoque federado que también mantuviera un nivel aceptable de accuracy sin demasiado sobreajuste. Se toma como muestra el 70 % de los nombres de usuario de todos los estudiantes y se recopilan sus datos para construir el conjunto de datos de entrenamiento, utilizando el resto para construir el conjunto de pruebas. El modelo se evalúa utilizando 50 divisiones aleatorias diferentes de los estudiantes, alcanzando una accuracy media del 81,7 % y un tiempo medio de ejecución de 105 segundos.

En el modelo federado, cada cliente se compone de muestras de 1.000 alumnos que representan a una escuela, lo que hace un total de 77 escuelas (clientes). Los clientes no comparten alumnos, pero pueden compartir cursos. El proceso de formación-evaluación consiste en 50 divisiones aleatorias diferentes en una proporción 70/30. Utilizando 1000 alumnos por cliente, ese 70 % de los datos se convierte en 54 clientes diferentes. Los datos restantes se utilizan para las pruebas; esto se hace de forma centralizada, donde un modelo con la misma arquitectura se inicializa con los pesos del modelo federado al final de cada ronda.

El proceso se repite en cada uno de los experimentos, en los que se prueban diferentes combinaciones en el número de rondas (R) y épocas locales (E), dejando un número fijo de épocas totales $R \times E = 20$, y variando el número de clientes por ronda utilizando: 1 cliente (disponibilidad mínima de clientes), 14 clientes (25 % de disponibilidad), 27 (50 %), 43 (75 %) y 54 (disponibilidad máxima). Los resultados en términos de accuracy se muestran en la Figura 4.2.

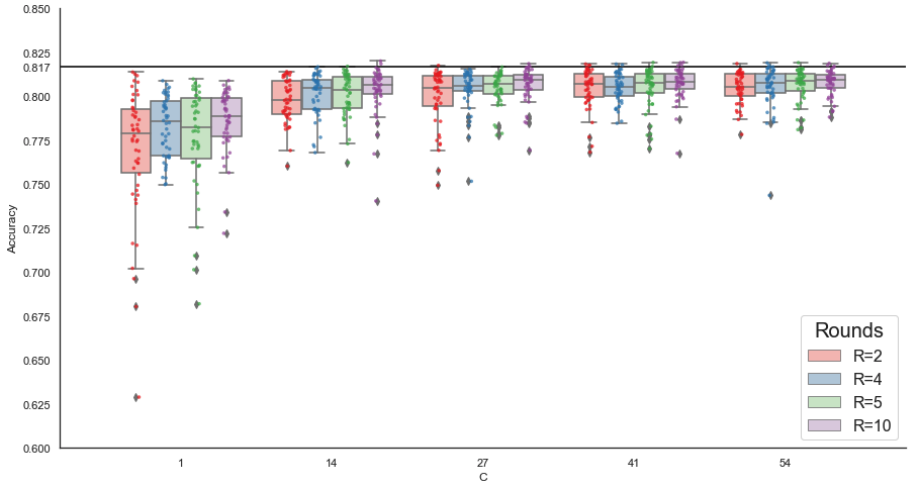

Figura 4.2: Resultados de accuracy de la predicción de abandono (versión federada), promediando 50 ejecuciones aleatorias con diferentes cantidades de clientes por ronda (C), número de rondas (R) y épocas locales de clientes (E) donde $R \times E = 20$. La línea negra marca la accuracy promediada por el modelo centralizado.

Aumentar el número de clientes de 1 a 14 provoca un salto del 2 %-3 % en la accuracy media, mientras que incrementos adicionales en el número de clientes sólo causan un aumento marginal en la accuracy media, pero también producen un aumento en el tiempo de ejecución. Si el número de clientes es fijo, se puede ver que favorecer el número de rondas R sobre las épocas locales E tiende a producir una mejor accuracy en general (las casillas en cada grupo suben), pero esto causará un incremento en el tiempo. También cabe destacar que la varianza disminuye a medida que se aumentan los clientes y las rondas.

El rendimiento en esta configuración federada es similar al del modelo centralizado, con una accuracy media superior al 76 % en todos los experimentos (que aumenta hasta el 80 % si se excluyen los experimentos con 1 cliente por ronda), una accuracy máxima del 82 % (alcanzada en una ejecución con 14 clientes y 10 rondas) y con alrededor del 63 % de todas las ejecuciones individuales, en todos los experimentos, con una accuracy superior al 80 %. Sin embargo, algunas ejecuciones siguen teniendo una accuracy relativamente baja.

Se pregunta entonces si es posible alcanzar *consistentemente* los resultados del entorno centralizado. Por lo tanto, se repiten los experimentos ejecutando tantas rondas como fueran necesarias para alcanzar el 81,7 % de accuracy. Este método de evaluación está inspirado en (B. McMahan y cols., 2017). La Figura 4.3 muestra los resultados; se puede ver que es posible alcanzar la accuracy del modelo centralizado en todos los casos, con la salvedad de que pueden ser necesarias muchas rondas.

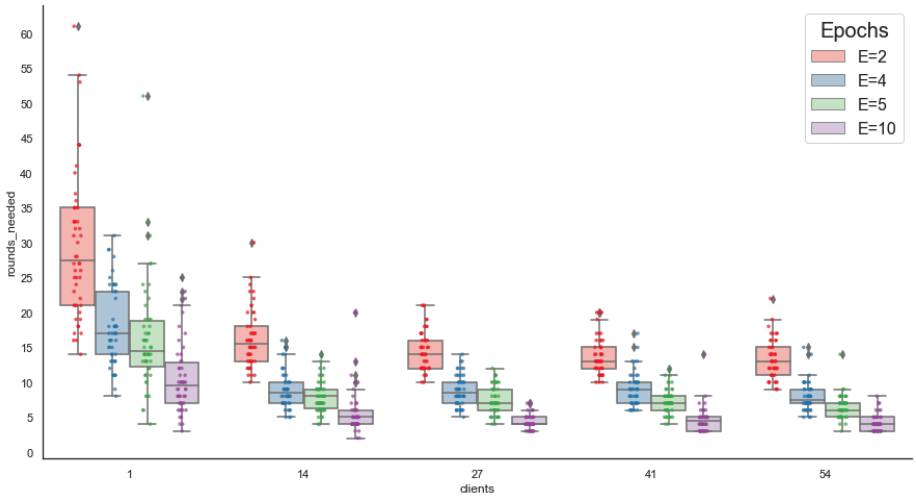

Figura 4.3: Número de rondas R necesarias para alcanzar la accuracy centralizada de referencia (81,7 %), promediando 50 ejecuciones aleatorias con diferentes cantidades de clientes por ronda (C) y épocas locales en los clientes (E).

El número máximo de rondas es necesario cuando se entrena con un cliente por ronda, y la accuracy resultante presenta una gran varianza. A partir de 14 clientes, los resultados no varían significativamente; es decir, aumentar el número de clientes no mejora necesariamente la convergencia. Aumentar la cantidad de épocas (E) reduce la cantidad de Rondas (R) promedio necesarias para alcanzar la accuracy de referencia (81,7 %) en todos los casos. Sin embargo, no existe una relación inversa 1:1: por ejemplo, con 14 clientes, si la cantidad de épocas es 2 (E=2) se necesita una media de 16 rondas, pero si la cantidad de épocas es 10 (E=10), se necesita 6 rondas, es decir, una relación de aumento x5 en E pero sólo una relación de disminución x2,6 en R.

4.1.2. Experimentos con la Distribución de Datos. Homogénea Frente a Heterogénea

Los experimentos descritos en la Sección 4.1.1 prueban la interacción entre diferentes hiperparámetros y cómo afectan al rendimiento, dada una configuración experimental fija. En esta sección, se fijan los parámetros pero se varían las configuraciones. Se comparan tres esquemas de entrenamiento: 1) cada institución entrena un modelo utilizando sólo sus datos locales, 2) una configuración federada, y 3) un enfoque centralizado con el entrenamiento que tiene lugar en los datos recogidos de todas las instituciones. En cada caso, las pruebas se realizan localmente con los datos de prueba de cada institución.

Por otro lado se varía la hipótesis de distribución de datos utilizando (a) una distribución de datos homogénea (los datos de los clientes se eligen aleatoriamente del conjunto de datos inicial) y (b) una heterogénea en la que el criterio de distribución de datos se basa en la tasa de abandono.

Distribución de Datos Homogenea

La Figura 4.4 muestra los resultados de 50 ejecuciones independientes utilizando la hipótesis de distribución homogénea de datos en la que los alumnos se distribuyen aleatoriamente entre los clientes.

En esta figura, cada punto representa la accuracy media alcanzada en los datos de prueba retenidos presentes en todos los clientes. En la diagonal, donde se sitúan los histogramas, se observa que bajo una federación los resultados tienen una varianza mayor que en los otros esquemas. En los cuadrantes centro-izquierda y arriba-centro se muestra como los resultados de las instituciones que se forman solas tienen mejores resultados de media que en una federación. Por otro lado en los cuadrantes abajo-izquierda y arriba-derecha se muestra que los resultados de las instituciones tienden a tener peores resultados que en un esquema centralizado. Por último los cuadrantes abajo-centro y centro-derecha muestran que la federación obtiene en general peores resultados que el modelo centralizado.

Distribución de Datos Heterogénea

Es interesante estudiar el efecto de la distribución de los datos en el rendimiento del modelo, por lo que se aplican otros criterios para distribuir los datos entre los clientes. En este caso, se estudia la distribución de la tasa de abandono teniendo en cuenta todo el conjunto de datos (véase la Figura 4.5).

Se utiliza para dividir la población de estudiantes entre las instituciones en función de la tasa de abandono de cada estudiante. La Tabla 4.1 presenta las categorías definidas y el número de alumnos por categoría.

La Figura 4.6 muestra la accuracy media tras 50 ejecuciones independientes distribuyendo los datos según la tasa de abandono (cada institución cliente tiene un número fijo de estudiantes = 1000).

Dado que el número de estudiantes con una tasa de abandono baja (11,32 % de toda la población) es menor, están infrarepresentados en el conjunto total

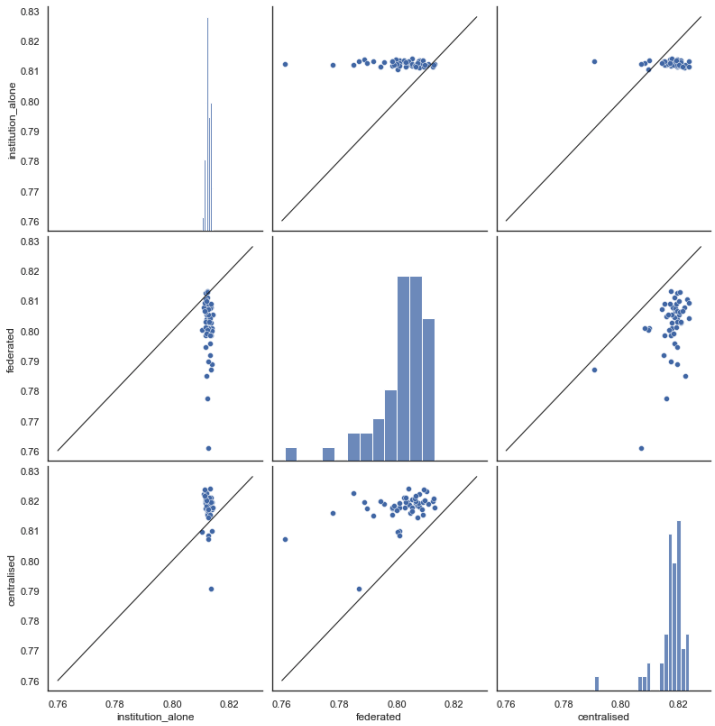

Figura 4.4: Comparación de la accuracy media tras 50 ejecuciones independientes utilizando tres esquemas de formación: instituciones solas, federadas y centralizadas.

Tabla 4.1: Categorías de distribución de los alumnos, según la tasa de abandono.

Alumnos con tasa de abandono baja (inferior a 0,2):	8723	11,32 %
Alumnos con tasa de abandono media (entre 0,2 y 0,8):	20567	26,68 %
Alumnos con tasa de abandono alta (superior a 0,8):	46687	60,57 %

de datos, lo que probablemente provoque un rendimiento deficiente en el caso del entrenamiento federado. En consecuencia, se observa que se forman pocas instituciones cliente en esta categoría (sólo nueve). En el caso de las instituciones cliente con una tasa de abandono media, los modelos entrenados mediante los enfoques federado y centralizado tienen un rendimiento similar. La hipótesis que se maneja es que estas instituciones se benefician del uso de más datos. Por último, dado que la mayoría de los estudiantes tienen una tasa de abandono alta, se obtienen buenos resultados para las versiones centralizada y federada en los casos de instituciones cliente pertenecientes a esta categoría de abandono.

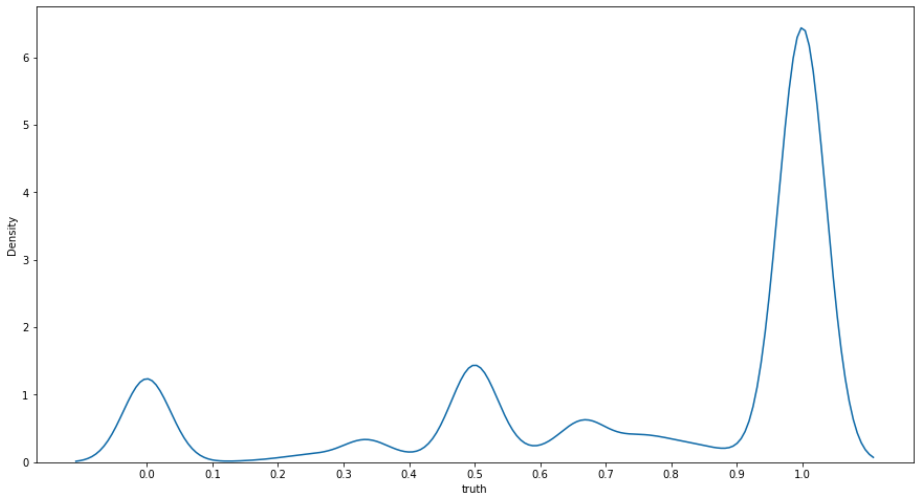

Figura 4.5: Distribución de la tasa de abandono entre todos los alumnos.

Discusión

Si la institución pertenece a una "clase favorecida", es decir, aquellas para las que hay más datos, entonces no hay mucha diferencia entre utilizar un esquema u otro. Si la institución tiene una distribución aleatoria, es mejor utilizar enfoques que aprovechen los datos de otros clientes, como los esquemas centralizados o federados. En este caso, no hay pruebas de pérdida de rendimiento utilizando la federación. Para completar este análisis, si la institución pertenece a una de las categorías con menos datos, es más conveniente utilizar un modelo personalizado entrenado sólo con sus datos.

4.2. Clasificación No Supervisada de Alumnos

La segunda tarea corresponde a la clasificación no supervisada de alumnos. Esta tarea puede realizarse de forma centralizada utilizando el conocido algoritmo k-means (Lloyd, 1982) (Hartigan y Wong, 1979). También existen versiones distribuidas de este algoritmo (Oliva, Setola, y Hadjicostis, 2014). La versión para Federated Learning de k-means (algoritmo 2) combina ideas del algoritmo distribuido con el concepto de utilizar sólo una fracción de los clientes en cada iteración (Kumar, Karthik, y Nair, 2020).

Criterio de Evaluación de Resultados

Es necesario comentar cómo se evalúan los resultados experimentales en este caso. Para comparar el rendimiento de los algoritmos de agrupación, no es factible seguir el mismo enfoque presentado en la Sección 4.1, donde se utilizó la métrica de accuracy para comparar el rendimiento de los diferentes modelos. Los

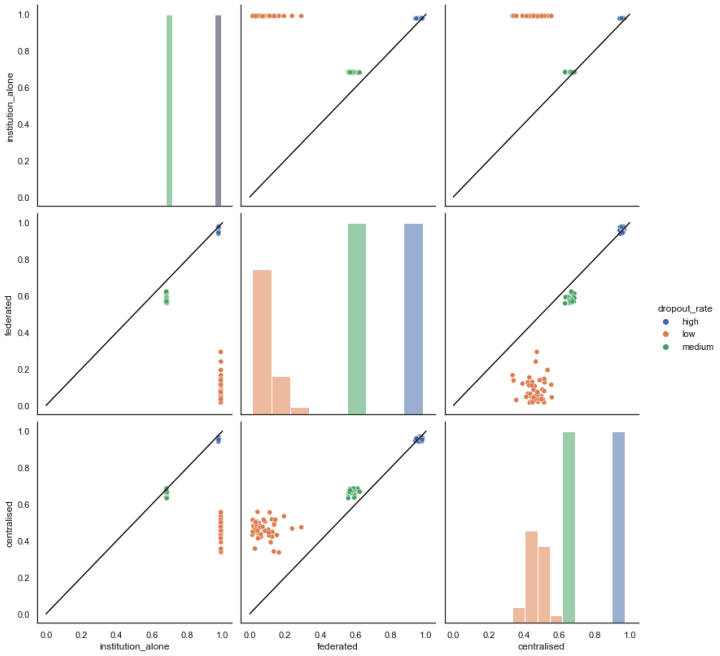

Figura 4.6: Comparación de la accuracy media tras 50 ejecuciones independientes utilizando tres esquemas de distribución: instituciones solas, federadas y centralizadas. La tasa de abandono varía entre los clientes según las categorías definidas en la Tabla 4.1.

algoritmos de agrupación se utilizan para agrupar puntos de datos basándose en las similitudes que comparten entre ellos. Pero no existe un criterio único para determinar lo que se considera un buen resultado. Se pueden hacer diferentes suposiciones a la hora de definir qué hace que diferentes puntos sean similares, y para cada suposición se obtendrán diferentes clusters, cada uno igualmente válido. Los supuestos y clusters finales que se utilicen dependerán de los intereses, y aun así, suele ser necesario analizar los clusters obtenidos e interpretar los resultados.

Como no se puede simplemente comparar las accuracies obtenidas con los algoritmos de clustering centralizado y federado, se comparan los clusters obtenidos con cada enfoque utilizando el Índice de Similitud de Jaccard. Cuanto mayor sea el valor, más similares serán los resultados entre sí. Utilizando este índice, la similitud entre los resultados obtenidos se calcula sin ningún juicio de valor sobre las soluciones particulares. Se asume que si son suficientemente similares, permitirán llegar a las mismas conclusiones e interpretarlas de forma similar, independientemente de que su aplicación sea centralizada o federada.

Así, el experimento no se centrará en la utilidad del algoritmo k-means en general o para esta tarea, sino en si k-means centralizado y federado pueden

Algorithm 2 Federated K-Means (Kumar y cols., 2020).

Parámetros:

C la fracción de clientes usados
R la cantidad de rondas globales
E es el numero de epocas locales
B es el tamaño del minibatch
α es el learning rate del servidor
β es el learning rate del cliente
K es la cantidad de clusters

Notación:

P_i son los ejemplos que tiene el cliente i
Γ es el conjunto de K centroides $(\mu_1, ..., \mu_k \in \mathbb{R}^m)$
Λ es el conjunto de los tamaños de los K clusters $(\lambda_1, ..., \lambda_k \in \mathbb{R})$

Ejecución del servidor:

inicializar los centroides Γ_0 aleatoriamente
$m \leftarrow \max(C * K, 1)$
for each ronda r de 1 a R **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (conjunto aleatorio de m clientes)
  for each cliente $i \in S_t$ **in parallel do**
    $(\Lambda_r^{(i)}, \Gamma_r^{(i)}) \leftarrow \text{ClientKMeans}(i, \Gamma_{r-1})$
  $\Lambda_r = \sum_{i=1}^{m} \Lambda_r^{(i)}$
  $\Gamma_r^* = \frac{1}{\Lambda_r} \cdot \sum_{i=1}^{m} \Lambda_r^{(i)} \cdot \Gamma_r^{(i)}$
  $\Gamma_r = \Gamma_{r-1} + \alpha \cdot (\Gamma_r^* - \Gamma_{r-1})$
return Γ_R

ClientKMeans(i,Γ)

$B_k \leftarrow$ (particionar P_i en batches de tamaño B)
for each epoca local e de 1 a E **do**
  inicializar tamaño de clusters Λ en 0
  for each batch $b \in B_k$ **do**
    for each dato $x_j \in b$ **in parallel do**
      $c_j = \arg\min_k ||x_j - \mu_k||^2$
    $\Lambda^b = \sum_{j=1}^{B} 1_{c_j=k}$
    $\Gamma^b = \frac{1}{\Lambda^b} \cdot \sum_{j=1}^{B} 1_{c_j=k} \cdot x_j$
    $\Lambda = \Lambda + \Lambda^b$
    $\Gamma = \Gamma + \beta \cdot \frac{\Lambda^b}{\Lambda} \cdot (\Gamma^b - \Gamma)$
return Λ, Γ

utilizarse indistintamente, proporcionando la misma información que el centra-

lizado pero con todas las ventajas de preservación de la privacidad de la versión federada.

Diseño Experimental

El objetivo del experimento es comparar los dos enfoques. Se parten de los mismos centroides iniciales en cada caso. Luego, en la versión centralizada, se ejecuta el algoritmo hasta que todos los centroides se mueven menos que un valor específico ϵ. Para las versiones federadas, se ejecutan un número determinado de rondas r, y se calcula el índice de Jaccard utilizando los clusters centralizados finales y los federados en cada ronda. De este modo, se obtiene la evolución de la puntuación de Jaccard para saber si los clusters se están pareciendo a los clusters centralizados finales.

El número de clusters k se fijará en 3 para el experimento. Para la versión centralizada, se utiliza una condición de parada de $\epsilon = 0,001$, y se ejecutan 4 versiones federadas con 5, 10, 20 & 40 clientes por ronda durante 100 rondas. La Figura 4.7 muestra los resultados de este experimento. Como era de esperar, las versiones que utilizaron más clientes por ronda tienden a tener mayores puntuaciones. Sin embargo, después de 10 rondas, todas las versiones de k-means federado tienen una puntuación Jaccard superior a 0,96, incluso la que utilizó sólo 5 clientes.

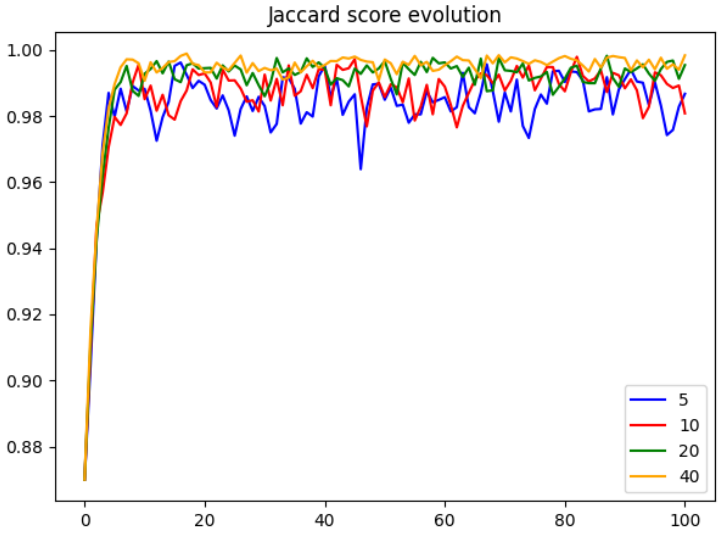

Figura 4.7: Evolución de la puntuación Jaccard para kmeans federado con 5, 10, 20 & 40 clientes disponibles por ronda y clusters kmeans centralizados para $\epsilon = 0,001$.

4.3. Discución y Conclusiones

Los resultados muestran que aumentando el número de clientes y favoreciendo más rondas se obtiene una mayor accuracy. Sin embargo, es crucial tener en cuenta que se trata de una simulación, y que no se han considerado los problemas que entraña la transferencia de información a través de la red en el mundo real. Por ejemplo, cuando la conectividad es un problema (como en instituciones de zonas rurales), es posible que no se pueda utilizar la mayoría de los clientes simultáneamente en la misma ronda. La latencia también puede ser un factor a tener en cuenta, en cuyo caso un aumento del tiempo de comunicación podría hacer prohibitivo ejecutar muchas rondas, en cuyo caso uno preferiría favorecer las épocas locales, incluso cuando se demostró que no es óptimo en términos de accuracy. También merece la pena señalar que no se han considerado esquemas que preserven la privacidad (Privacidad Diferencial y otros (Liu y cols., 2021)) en los experimentos.

Es esencial no olvidar que todos los experimentos se basan en datos de MOOCs; esto debe tenerse en cuenta a la hora de extrapolar los resultados al contexto de una institución física. Algunas variables tienen un equivalente (cantidad de cursos realizados, por ejemplo), pero otras sin duda difieren.

Por último, se ha centrado en evaluar si un modelo puede arrojar resultados similares en los escenarios de formación federada y centralizada. Sin embargo, no se ha explorado en qué medida se beneficia cada cliente de la federación. La pregunta es: ¿se beneficia la institución de los patrones aprendidos por el modelo en otras instituciones, o estaría mejor simplemente entrenando un modelo centralizado propio? La respuesta a esta pregunta probablemente varía y depende en gran medida de la cantidad de datos que posea la institución. También podría ocurrir que un modelo entrenado en una única institución funcionara bien con datos no vistos anteriormente del mismo lugar, pero no se adaptara a un cambio en la distribución (por ejemplo, nuevos estudiantes con un comportamiento muy diferente y no visto se matriculan en la institución). Por el contrario, un modelo federado enriquecido con datos de varias otras instituciones podría ser más robusto. Un experimento que introduzca artificialmente una deriva en los datos podría arrojar luz sobre este tema.

Sin limitaciones de red, concentrar los recursos en más rondas, en lugar de en épocas locales de los clientes, aporta mejores resultados. Si el tiempo y la conectividad no son un problema, también sería óptimo utilizar tantos clientes como sea posible por ronda, sin embargo la ganancia no es sustancial y se podrían alcanzar resultados razonables utilizando muchos menos datos (como en el experimento al utilizar el 25 % y el 50 % de todos los clientes). Se conluye que FL tiene el potencial de alcanzar los mismos resultados que el ML tradicional en entornos del mundo real, como lo hace en los experimentos, pero es necesario realizar pruebas en un contexto no experimental para confirmarlo.

Capítulo 5

Aplicaciones de Aprendizaje Federado Vertical

Se comienza presentando las diferencias y desafíos adicionales que presenta este enfoque comparado con HFL. Luego se presenta un proceso de entrenamiento para redes neuronales para VFL presentado en (Wei y cols., 2024). Por último se presenta un algoritmo VFL para regresión logística utilizando encriptado homomórfico, presentado en (Hardy y cols., 2017).

5.1. Diferencias y Desafíos

En VFL los datos están particionados verticalmente en lugar de horizontalmente como en HFL, lo que introduce grandes diferencias a la hora de entrenar los modelos y desafíos adicionales. En esta sección se verán las principales diferencias y desafíos a la hora de entrenar con VFL en lugar de HFL.

Una gran diferencia es cómo está definido el modelo, en HFL los clientes tienen datos completos con todas las características y pueden entrenar el modelos locales de manera independiente, que luego serán agregados para contribuír al modelo global. En VFL cada cliente tiene solo una parte de los datos, precisando del resto de los clientes para completar todas las características de cada dato, lo que presenta desafíos adicionales.

A la hora de diseñar modelos HFL alcanza con diseñar el modelo global tal cómo se haría en AA tradicional, y luego se entrenan los clientes con el mismo modelo localmente. Pero para VHL el modelo global deberá ser repartido entre el servidor y los clientes, teniendo varios modelos locales, donde cada cliente entrenará con su modelo local, por lo que se deben tener en consideración los datos que posee cada cliente al diseñar modelos VFL. En modelos con varias capas, es crucial definir como se repartirá el modelo, ya que esto tendrá un

gran peso en costos de comunicación, de cómputo y en temas de seguridad y privacidad.

En VFL se precisan el resto de clientes para entrenar un modelo, por lo que se deben compartir las salidas intermedias en cada parte del entrenamiento, lo que implica un aumento en los costos de comunicación ya que se tienen que mandar mensajes con las salidas intermedias mientras que en HFL solo se comparten las actualizaciones al terminar de entrenar localmente.

Estas salidas intermedias no son datos crudos, pero de todas formas no son suficientemente seguras como para garantizar que se preserve la privacidad. Un cliente podría inferir las características de otros participantes utilizando lo que conoce del modelo y las salidas intermedias. Es por esto que es necesario utilizar mecanismos para presevar la privacidad de los datos, cómo pueden ser DP, MPC, HE o TEE's presentados en la sección 2.5.2, en la sección 5.3 se presenta un algoritmo para regresión logistica en VFL utilizando HE.

La disponibilidad de los participantes es otro punto donde se generan diferencias. En HFL los clientes que entrenan se seleccionan entre un pool de participantes disponibles, por lo que si un participante no está disponible simplemente no se selecciona, en la sección 2.5.3 se discute como esto puede generar sesgos, pero el entrenamiento funciona con normalidad. En VFL cada participante tiene características propias que son esenciales a la hora de entrenar cada ronda, el modelo no está completo sin todos los participantes, por lo que es necesario que todos los clientes estén disponibles al mismo tiempo.

Además en VFL un cliente con limitaciones de hardware para realizar cálculos o con problemas de conectividad será cuello de botella, atrasando el entrenamiento, ya que se precisan las salidas intermedias de todos los clientes antes de poder avanzar a la siguiente parte. Mientras que en HFL los clientes pueden enviar sus actualizaciones del modelo a destiempo, por lo que no es un problema si un cliente demora más en entrenar, ni tampoco se precisa una conexión estable hasta no terminar el entrenamiento porque no hay comunicaciones intermedias.

5.2. Usando Redes Neuronales

En la sección 5.1 se presentaron algunos de los principales desafíos de VFL y diferencias respecto a HFL. En esta sección se ilustra el proceso de entrenamiento de VFL para redes neuronales.

A continuación, se presenta el proceso de entrenamiento para redes neuronales presentado en (Wei y cols., 2024), que detalla las etapas clave involucradas en cada época de aprendizaje. En la figura 5.1 se observan estas etapas numeradas del 1 al 7.

1. Private Set Intersection (PSI) Antes de que comience el entrenamiento del modelo, es esencial que el framework identifique las muestras de datos comunes entre todos los participantes, ya que estos comparten el mismo espacio de muestras pero tienen diferentes atributos. Este proceso se denomina Private

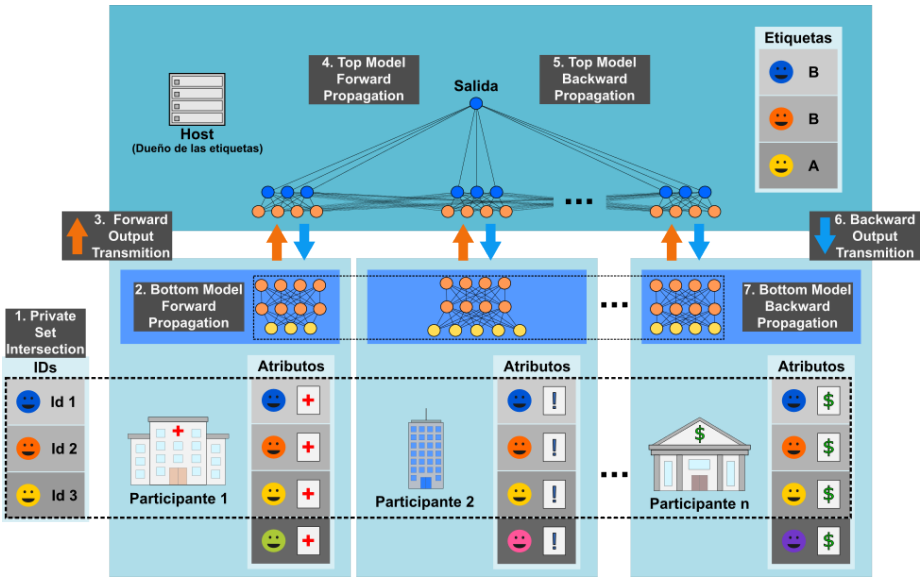

Figura 5.1: Etapas involucradas en cada época de aprendizaje para el proceso de entrenamiento VFL con redes neuronales presentado en (Wei y cols., 2024).

Set Intersection (PSI). PSI es un protocolo criptográfico interactivo que permite a múltiples participantes encontrar los IDs comunes sin revelar ninguna otra información.

2. Bottom Model Forward Propagation Una vez que se han determinado las muestras de datos alineadas entre todos los participantes, cada uno de ellos completa un proceso de forward propagation utilizando los datos locales en su propio modelo inferior. Este proceso es similar al forward propagation convencional en el entrenamiento de redes neuronales, con la excepción de que no se calcula el valor de pérdida en esta etapa. Aquí, cada participante transforma sus atributos originales en características que servirán de insumo para la siguiente etapa.

3. Forward Output Transmission Después del forward propagation, cada participante debe transmitir su salida al propietario de las etiquetas. Esta salida contiene resultados intermedios del modelo local, que son las características transformadas a partir de los atributos originales. Lo que implica que este proceso de transmisión podría exponer información privada de los participantes. Por lo tanto, es crucial implementar métodos avanzados de preservación de la privacidad para mitigar este riesgo, aunque esto podría aumentar los costos de comunicación y la complejidad computacional.

4. Top Model Forward Propagation El propietario de las etiquetas utiliza las salidas recopiladas de todos los participantes para calcular el valor de la función de pérdida en su modelo superior. Este modelo superior integra las características proporcionadas por los modelos inferiores de cada participante y utiliza las etiquetas para evaluar el rendimiento del modelo combinado.

5. Top Model Backward Propagation En esta etapa, el propietario de las etiquetas realiza el backward propagation, calculando los gradientes de:

- los parámetros del modelo superior
- los parámetros de la última capa de cada participante

Con los gradientes del modelo superior, el propietario puede calcular los gradientes promedio para cada lote y actualizar su modelo.

6. Backward Output Transmission Los gradientes se envían de vuelta a cada participante. Es importante destacar que la cantidad de bits necesarios para esta transmisión suele ser mucho menor que en la etapa de forward propagation, ya que solo se transmiten gradientes y no salidas intermedias.

7. Bottom Model Backward Propagation Finalmente, cada participante utiliza los gradientes recibidos del propietario de las etiquetas para calcular los gradientes de sus propios parámetros del modelo inferior, basándose en los datos locales. Después de calcular estos gradientes, cada participante actualiza su modelo inferior, completando así el ciclo de aprendizaje.

5.3. Usando Encriptado Homomórfico en Regresión Logística

En la sección anterior se presentó el proceso de entrenamiento de VFL para redes neuronales, en esta sección se presenta un algoritmo VFL para regresión logística basado en comunicaciones entre los clientes utilizando encriptado homomórfico, el algoritmo fué presentado en (Hardy y cols., 2017). En este caso no solo se introduce un algoritmo VFL para otra técnica de AA, la regresión logística, sino que además se presenta cómo es utilizado el encriptado homomórfico para preservar la privacidad, en la sección 2.5.2 se presentó el encriptado homomórfico y la importancia de utilizar mecanismos de preservación de la privacidad para minimizar la exposición de información.

Es importante destacar que en este algoritmo es indispensable contar con el encriptado homomórfico si se pretende preservar la privacidad. Esto sucede porque se precisa realizar el calculo del gradiente utilizando los datos de ambos clientes y estos datos se compartirían prácticamente crudos (a excepción de un par de cálculos lineales) si no fuera por el encriptado. En el caso presentado en la sección anterior se pueden agregar mecanismos adicionales para preservar la

privacidad pero no son estrictamente necesarios ya que los datos sufren varias transformaciones en las capas del modelo inferior que le agregan ruido, siendo un el resultado intermedio de una capa lo que se transmite al servicio, quién no conoce los pesos del modelo inferior e incluso podría llegar a no conocer como es el modelo inferior (solo sus salidas).

Para este algoritmo se asume que los participantes son "curiosos-pero-honestos" un termino que se utiliza normalmente para referirse a participantes que no van a intentar interferir en el entrenamiento, es decir que van a seguir los protocolos tal cómo tienen que seguirlo y no van a introducir errores, pero que son curiosos y van a intentar inferir tanta información como les sea posible. En el contexto de VFL tiene sentido suponer esto ya que los clientes son organizaciones que desean colaborar y desean que el entrenamiento funcione para lograr obtener un modelo lo más preciso posible, lo que les genera beneficio propio, y en cambio se hacen algo para que no entrene bien se estarían perjudicando.

Existen 3 participantes, A y B son los clientes que tienen la información y C es el servidor que participa como intermediario que tiene la clave con la que desencripta las actualizaciones. A y B se comparten datos encriptados que no pueden desencriptar porque no poseen la clave, y C solo recibe actualizaciones de los modelos, los cuales no se consideran privados en este contexto.

5.3.1. Encriptado Homomórfico

Este algoritmo se basa fuertemente en el encriptado homomórfio, en la sección 2.5.2 se introduce el concepto, ahora se presentaran las propiedades principales que serán utilizadas en el algoritmo.

Un **encriptado homomórfico** es una técnica criptográfica que permite hacer operaciones sobre los datos encriptados sin necesitar decifrarlos. Para esto es necesario tener una operación que dados dos números encriptados, devuelva la suma de los dos números pero encriptado. Es decir, si $[[u]]$ es u encriptado y $+$ es el operador entonces:

$$[[u]] + [[v]] = [[u + v]] \tag{5.1}$$

A su vez sumando varias veces tenemos para v valor no encriptado que:

$$v \cdot [[u]] = [[vu]] \tag{5.2}$$

Lo que también se puede extender para vectores y matrices. Por ejemplo si u y v son vectores sin encriptar, el producto interno es:

$$v^\top [[u] = [[v^\top u]]] \tag{5.3}$$

Y el producto de matrices:

$$v \circ [[u]] = [[v \circ u]] \tag{5.4}$$

Esto permite usar encriptado homomórfico para implementar primitivas de álgebra lineal que son útiles para AA.

5.3.2. Adaptación de SGD para Usar con HE

Para utilizar decenso por gradiente estocástico con encriptado homomórfico es necesario adapatarlo para que funcione con las máscaras y se pueda calcular solo con operaciones que cumplan las propiedades de HE. Para esto se utiliza la aproximación de Taylor de segundo orden para aproximar la loss y el gradiente, obteniendo aproximaciónes que se pueden calcular usando solo las operaciones permitidas sobre los datos encriptados.

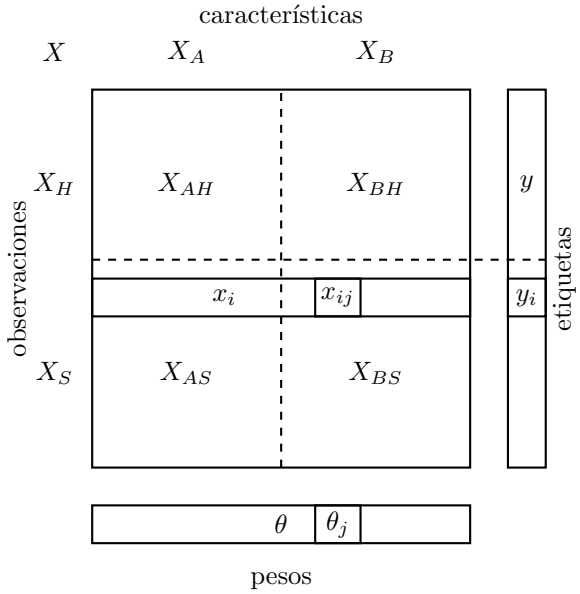

Figura 5.2: Notación utilizada en (Hardy y cols., 2017).

En la figura 5.2 se puede observar la notación utilizada en (Hardy y cols., 2017), que es la misma que se utiliza en el cálculo del gradiente como en los algoritmos.

Cálculo de Gradiente con Aproximación de Taylor y Máscara

Loss: El promedio de la loss computada en el conjunto de entenamiento es:

$$\ell_S(\theta) = \frac{1}{n} \sum_{i \in S} log(1 + e^{-yi\theta^\top x_i}) \tag{5.5}$$

Gradiente: Y el gradiente estocástico calculado para un mini-batch $S' \subseteq S$ de tamaño s' es:

$$\nabla \ell_{S'}(\theta) = \frac{1}{s'} \sum_{i \in S'} (\frac{1}{1 + e^{-yi\theta^\top x}} - 1) y_i x_i \tag{5.6}$$

Serie de Taylor: La serie de Taylor de $log(1 + e^{-z})$ alrededor de $z = 0$ es:

$$log(1 + e^{-z}) = log2 - \frac{1}{2}z + \frac{1}{8}z^2 - \frac{1}{192}z^4 + O(z^6) \tag{5.7}$$

Taylor loss: Usando que $y_i^2 = 1, \forall i$ La aproximación de segundo orden de la ecuación 5.5 evaluada en H es:

$$\ell_H(\theta) \approx \frac{1}{h}\sum_{i \in H} log2 - \frac{1}{2}y_i\theta^\top x_i + \frac{1}{8}(\theta\top x_i)^2 \tag{5.8}$$

Taylor Gradient: Y el gradiente para un mini-batch S' es:

$$\nabla\ell_{S'}(\theta) \approx \frac{1}{s'}\sum_{i \in S'}(\frac{1}{4}\theta^\top x_i - \frac{1}{2}y_i)x_i \tag{5.9}$$

Taylor Gradient con Máscara: Al aplicar la máscara a la ecuación 5.9 obtenemos el gradiente enmascarado para un mini-batch S':

$$[[\nabla\ell_{S'}(\theta)]] \approx \frac{1}{s'}\sum_{i \in S'}[[m_i]](\frac{1}{4}\theta^\top x_i - \frac{1}{2}y_i)x_i \tag{5.10}$$

La ecuación 5.10 es la aproximación de taylor del gradiente con la máscara aplicada, la cuál se puede calcular usando operaciones de HE, esta ecuación se utiliza para el algoritmo 4 que calcula el gradiente.

5.3.3. Algoritmo

El algoritmo que se plantea en (Hardy y cols., 2017) utiliza como fundamento el gradiente calculado en la sección 5.3.2, el mismo se presenta en el algoritmo 3. El cálculo del gradiente utilizado en el mismo se presenta en el algoritmo 4.

Algorithm 3 Secure logistic regression (Ejecutado por C) (Hardy y cols., 2017).

Datos: Mascara m, learning rate η, regularización Γ, batch size s'
Resultado: Modelo θ
crear claves de encriptado homomórfico
enviar clave pública a A y B
encriptar m con la clave publica, enviar $[[m]]$ a A y B
inizializar loss
$\theta \leftarrow 0, \ell_H \leftarrow \infty$
repeat
  for cada mini-batch S' **do**
  $\nabla \ell_{S'}(\theta) \leftarrow$ Secure Gradient(θ, s') 4
  $\theta \leftarrow \theta - \eta(\nabla \ell) S'(\theta) + \Gamma\theta)$
  $\ell_H(\theta) \leftarrow$ calcular loss
  if $\ell_H(\theta)$ no disminuyó por un tiempo **then break**
until maximas iteraciones
return θ

El algoritmo 4 se utiliza para calcular el gradiente en el algoritmo 3.

Algorithm 4 Secure Gradient (Hardy y cols., 2017).

Datos: Modelo θ, tamaño del batch s'
Resultado: $\nabla \ell_{S'}(\theta)$ de un mini-batch S' (sin revelar)
C
  enviar θ a A
A
  seleccionar el siguiente batch $S' \subset S, |S'| = s'$
  $u = \frac{1}{4} X_{AS'} \theta_A$
  $[[u']] = [[m]]_{S'} \circ (u - \frac{1}{2} y_{S'})$
  enviar θ, S' y $[[u']]$ a B
B
  $v = \frac{1}{4} X_{BS'} \theta_B$
  $[[w]] = [[u']] + [[m]]_{S'} \circ (v)$
  $[[z]] = X_{BS'}[[w]]$
  enviar $[[w]]$ y $[[z]]$ a A
A
  $[[z']] = X_{AS'}[[w]]$
  enviar $[[z']]$ y $[[z]]$ a C
C
  obtener $[[\nabla \ell_{S'}(\theta)]]$ concatenando $[[z']]$ y $[[z]]$
  obtener $\nabla \ell_{S'}(\theta)$ desencriptando con la clave privada

El algoritmo que se presenta en esta sección es un algoritmo VFL que utiliza encriptado homomórfico y para esto una adaptación del cálculo de gradiente con aproximación de Taylor permitiendo utilizar una máscara de encriptado homórfico. Mediante este algoritmo dos participantes pueden colaborar para

entrenar un modelo utilizando su información pero sin compartirla directamente, mediante la ayuda de un intermediario C.

5.4. Estado del Arte y Herramientas para VFL

El Aprendizaje Federado es un enfoque reciente en el campo de la inteligencia artificial; esta novedad le suma complejidad, teniendo una cantidad de desafíos nuevos sin resolver. En un área que ya es complicada de por sí, en donde hay muchos parámetros que ajustar, los cuales dependen del caso específico, que además se obtienen a partir de prueba y error, donde no hay soluciones absolutas, sino que se requiere de mucho tiempo y esfuerzo para implementar soluciones a medida para lo que se quiere resolver, en donde se está avanzando continuamente y se presentan nuevos trabajos a diario con lo que se está en continuo avance.

Este paradigma se originó teniendo en mente el escenario cross-device, con todos los clientes contando con los mismos tipos de datos (H. B. McMahan y cols., 2023), más adelante se extiende el concepto para incluir el escenario cross-silo y tener en cuenta no solo el tipo de particionamiento horizontal sino que también el vertical, en donde distintos participantes cuentan con distintos tipos de datos (Kairouz, McMahan, Avent, y cols., 2021).

VFL no es el escenario base de un enfoque que ya es novedoso de por sí, lo que dificulta aún más el desarrollo de investigación de este escenario, contando con menor cantidad de información y menos trabajos realizados, ya que el mayor foco está en HFL. En la literatura la mayoría de trabajos son a nivel teórico, y en los casos que incluyen partes prácticas es frecuente encontrar trabajos que reportan resultados prometedores, pero que carecen de detalles sobre su implementación y reproducibilidad, lo que dificulta realizar pruebas y validar los algoritmos propuestos.

Debido a contar con distintos tipos de datos, VFL presenta desafíos adicionales, ya que requiere coordinación entre los clientes para garantizar que cada ronda se pueda llevar a cabo sin problemas, utilizando todos los atributos de los datos y así entrenando al modelo global en conjunto. Esto hace que los algoritmos deban ser adaptados y no se puedan utilizar los mismos que para HFL. Esto también impacta en las herramientas a utilizar, ya que las mismas soluciones que servirían para HFL no necesariamente tienen implementaciones para VFL.

En la sección 2.6.2 se presentaron algunas herramientas relevadas, TensorFlowFederated y Flower, las mismas fueron probadas y se concluyó que son útiles para HFL, utilizandolas para un estudio de factibilidad en el capítulo 3, y luego en el capítulo 4 se utilizó TensorFlowFederated para abordar un caso más complejo en el ambito de LA.

Estas herramientas, además de ser de gran utilidad para HFL, cuentan con documentación completa, presentan ejemplos claros y tutoriales, tienen un equipo de desarrollo activo que resuelven errores y tienen métodos de comunicación disponibles, lo que las hace tener gran facilidad a la hora de usarlas. Pero no cuentan con soporte para VFL, por lo que al querer implementar algoritmos o

realizar pruebas con VFL se precisan utilizar otras soluciones.

Para este trabajo se investigaron si había herramientas disponibles específicamente para VFL, la oferta era menor que para HFL pero se lograron encontrar algunas que decían contar con distintas funcionalidades para VFL, encontrando para distintos algoritmos como redes neuronales, regresiones logísticas, arboles de decisión, entre otros. En general no contaban con mucha documentación o directamente no tenían, y las que tenían era poco clara.

Se realizaron múltiples pruebas con la intención de hacer funcionar algunas para poder compararlas y realizar pruebas con ellas. Tras muchos intentos y bastante esfuerzo, no se logró hacer funcionar ninguna de las herramientas que se probaron, con todas hubo diversos problemas a la hora de hacerlas correr, tampoco contaban con métodos de comunicación con el equipo de desarrollo, el único canal que se encontró eran las issues del repositorio GitHub, pero en general si tenían interacciones no había respuestas a los problemas planteados por la comunidad. Hubo una herramienta con la que se logró contactar al desarrollador a través de mail y se tuvieron varios intercambios intentando solucionar los problemas que surgían para hacerla correr pero finalmente se desistió tras no lograrlo.

Esto permite apreciar cómo técnicas nuevas que parecen prometedoras tienen complicaciones adicionales por no tener suficiente investigación y desarrollo, dificultando en gran medida realizar pruebas y en especial si se quisiera implementar una solución en producción. En un ambiente de investigación en dónde no todo tiene que funcionar perfecto y se puede aislar que se quiere probar ya es complicado, en el mundo real donde nada puede fallar, especialmente cuando se manejan datos sensibles, hace que esta técnica (VFL) por más prometedora que sea no esté apta para su uso al día de hoy.

Es de gran interés realizar pruebas además del estudio teórico que se hizo sobre VFL, pero implementar un algoritmo de cero está fuera del alcance de este proyecto, y al no contar con herramientas no se pudieron realizar pruebas. Para un trabajo futuro sería de gran importancia realizar diversas pruebas sobre algoritmos VFL, incluso implementando una solución propia.

Capítulo 6

Conclusiones

En esta tesis se ha explorado el uso de Aprendizaje Federado (FL) en distintos contextos de Aprendizaje Automático, comparando su desempeño con el aprendizaje centralizado tradicional. A lo largo de los capítulos, se han realizado estudios de factibilidad y análisis detallados de herramientas y técnicas aplicadas en escenarios de aprendizaje federado tanto horizontal como vertical.

Los resultados obtenidos en los estudios de casos presentados en el Capítulo 3 confirman que el FL es una alternativa viable al enfoque centralizado, especialmente en contextos donde la privacidad de los datos es una preocupación crítica. En el caso del dataset MNIST, se logró validar el buen funcionamiento de las herramientas TFF y Flower, demostrando que es posible federar el problema sin perder accuracy en las predicciones.

El estudio realizado con el dataset CIFAR10 destacó cómo la configuración de los hiperparámetros, como la cantidad de rondas y la fracción de clientes, afecta significativamente el rendimiento del modelo. A pesar de las variaciones en la complejidad del problema, los resultados obtenidos indican que, bajo las configuraciones adecuadas, FL puede aproximarse al rendimiento del enfoque centralizado.

El Capítulo 4 exploró un caso de estudio más exigente en el contexto de la predicción de abandono escolar y la clasificación no supervisada de estudiantes, siendo esta una aplicación de interés para el uso de FL. Se demostró que, en condiciones controladas, FL puede alcanzar niveles de accuracy comparables a los del aprendizaje centralizado. Sin embargo, también hay que tener en cuenta las limitaciones prácticas del FL en escenarios del mundo real, donde la latencia de red y la conectividad pueden afectar la viabilidad de ejecutar múltiples rondas de entrenamiento. Además, se plantea la cuestión de si los modelos federados realmente ofrecen ventajas a nivel institucional, en comparación con entrenar modelos centralizados propios, destacando la necesidad de futuros estudios para evaluar la robustez de los modelos federados ante cambios en la distribución de los datos.

En conjunto, los resultados obtenidos en esta tesis sugieren que el Aprendizaje Federado Horizontal tiene el potencial de igualar al aprendizaje automático

tradicional en términos de accuracy, pero al mismo tiempo preservando la privacidad de los datos. Sin embargo, es esencial realizar pruebas adicionales en entornos no experimentales y considerar factores como la conectividad de red, la disponibilidad de los clientes y analizar en mayor profundidad sobre la privacidad de los datos para confirmar su aplicabilidad en escenarios reales.

Las técnicas de Aprendizaje Federado Vertical tienen muchas aplicaciones en dónde instituciones y organizaciones se podrían ver beneficiadas mutuamente de su utilización. En el Capítulo 5 se presentaron técnicas prometedoras a nivel teórico, pero aún presenta varios desafíos sin resolver, siendo necesario más estudio y desarrollo de más herramientas para hacer viable su uso.

Finalmente, se concluye que FL es una técnica prometedora, especialmente en aplicaciones donde la privacidad es una prioridad. Sin embargo, su implementación práctica requiere una cuidadosa consideración de las limitaciones y desafíos asociados.

Referencias

Alex Krizhevsky. (2021). *Sitio Web del proyecto CIFAR.* Descargado de https://www.cs.toronto.edu/~kriz/cifar.html

Banse, A., Kreischer, J., y i Jürgens, X. O. (2024). *Federated learning with differential privacy.* Descargado de https://arxiv.org/abs/2402.02230

Bermolen, P., Capdheourat, G., Etcheverry, L., Fachola, C., Fariello, M. I., y Tornaría, A. (2022). *Flea: Aprendizaje federado aplicado a analíticas de aprendizaje.* Universidad de la República. Facultad de Ingeniería. Descargado de https://hdl.handle.net/20.500.12381/3126

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., ... Lane, N. D. (2020). Flower: A Friendly Federated Learning Research Framework. Descargado de http://arxiv.org/abs/2007.14390

Bharati, S., Mondal, M. R. H., Podder, P., y Prasath, V. S. (2022, mayo). Federated learning: Applications, challenges and future directions. *International Journal of Hybrid Intelligent Systems*, *18*(1–2), 19–35. Descargado de http://dx.doi.org/10.3233/HIS-220006 doi: 10.3233/his-220006

Birhane, A., y Prabhu, V. U. (2021). Large image datasets: A pyrrhic win for computer vision? En *2021 ieee winter conference on applications of computer vision (wacv)* (pp. 1536–1546).

Caldas, S., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., y Talwalkar, A. (2018). LEAF: A benchmark for federated settings. *CoRR*, *abs/1812.01097*. Descargado de http://arxiv.org/abs/1812.01097

Cheon, J. H., Kim, A., Kim, M., y Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. En T. Takagi y T. Peyrin (Eds.), *Advances in cryptology – asiacrypt 2017* (pp. 409–437). Cham: Springer International Publishing.

Cohen, G., Afshar, S., Tapson, J., y van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. *CoRR*, *abs/1702.05373*. Descargado de http://arxiv.org/abs/1702.05373

Fachola, C., Tornaría, A., Bermolen, P., Capdehourat, G., Etcheverry, L., y Fariello, M. I. (2023). Federated learning for data analytics in education. *Data*, *8*(2). Descargado de https://www.mdpi.com/2306-5729/8/2/43 doi: 10.3390/data8020043

Fang, H., y Qian, Q. (2021). Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, *13*(4). Des-

cargado de https://www.mdpi.com/1999-5903/13/4/94 doi: 10.3390/fi13040094

Feng, W., Tang, J., y Liu, T. X. (2019). Understanding dropouts in MOOCs. En *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 517–524).

FLEA. (2022). *FLEA project public repository.* https://gitlab.fing.edu.uy/flea/flea. Descargado de https://gitlab.fing.edu.uy/lorenae/flea

Flower. (2022a). *Flower a friendly federated learning framework.* https://flower.dev/.

Flower. (2022b). *Repositorio del proyecto flower.* https://github.com/adap/flower.

Google. (2021). *TensorFlow Federated: GitHub.* Descargado de https://github.com/tensorflow/federated

Google. (2024). *Supervised learning — machine learning — google for developers.* https://developers.google.com/machine-learning/intro-to-ml/supervised#:~:text=Features%20are%20the%20values%20that,wind%20direction%2C%20and%20atmospheric%20pressure. (Accessed: 2024-8-25)

Google Brain Team. (2021a). *TensorFlow Federated: API Documentation.* Descargado de https://www.tensorflow.org/federated/api_docs/python/tff

Google Brain Team. (2021b). *TensorFlow Federated: Datasets.* Descargado de https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets

Google Brain Team. (2021c). *TensorFlow Federated: Federated Averaging.* Descargado de https://www.tensorflow.org/federated/api_docs/python/tff/learning/ClientFedAvg

Google Brain Team. (2021d). *TensorFlow Federated: Federated Core.* Descargado de https://www.tensorflow.org/federated/federated_core

Google Brain Team. (2021e). *TensorFlow Federated: Federated Learning.* Descargado de https://www.tensorflow.org/federated/federated_learning

Google Brain Team. (2021f). *TensorFlow Federated: GitHub Issues.* Descargado de https://github.com/tensorflow/federated/issues

Google Brain Team. (2021g). *TensorFlow Federated: guides.* Descargado de https://www.tensorflow.org/federated/get_started

Google Brain Team. (2021h). *TensorFlow Federated: Stack Overflow.* Descargado de https://stackoverflow.com/questions/tagged/tensorflow-federated

Google Brain Team. (2021i). *TensorFlow Federated: Tutorials.* Descargado de https://www.tensorflow.org/federated/tutorials/tutorials_overview

Google Brain Team. (2021j). *TensorFlow Federated: Web.* Descargado de https://www.tensorflow.org/federated

Google Brain Team. (2021k). *TensorFlow: Forum.* Descargado de https://discuss.tensorflow.org/

Google Brain Team. (2021l). *TensorFlow: Web.* Descargado de https://www.tensorflow.org

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., y Thorne, B. (2017). *Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption.* Descargado de https://arxiv.org/abs/1711.10677

Hartigan, J. A., y Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108. Descargado 2024-10-22, de http://www.jstor.org/stable/2346830

Hosseini, S. M., Sikaroudi, M., Babaei, M., y Tizhoosh, H. R. (2022). *Cluster based secure multi-party computation in federated learning for histopathology images.* Descargado de https://arxiv.org/abs/2208.10919

Ji, Z., Lipton, Z. C., y Elkan, C. (2014). *Differential privacy and machine learning: a survey and review.* Descargado de https://arxiv.org/abs/1412.7584

Kaggle. (2016). *Shakespeare plays.* Descargado de https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays

Kairouz, P., McMahan, B., Song, S., Thakkar, O., Thakurta, A., y Xu, Z. (2021). Practical and Private (Deep) Learning without Sampling or Shuffling. , 1–34. Descargado de http://arxiv.org/abs/2103.00039

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, *14*(1–2), 1-210. Descargado de http://dx.doi.org/10.1561/2200000083 doi: 10.1561/2200000083

KDD. (2015). *KDDCup.* http://moocdata.cn/challenges/kdd-cup-2015. Descargado de http://moocdata.cn/challenges/kdd-cup-2015

Kingma, D. P., y Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konečný, J., McMahan, H. B., Ramage, D., y Richtarik, P. (2016). *Federated Optimization: Distributed Machine Learning for On-Device Intelligence.* Descargado de https://arxiv.org/abs/1610.02527

Krizhevsky, A., Hinton, G., y cols. (2009). Learning multiple layers of features from tiny images.

Krut Patel. (2019). *MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN).* Descargado de https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

Kumar, H. H., Karthik, V., y Nair, M. K. (2020). Federated k-means clustering: A novel edge AI based approach for privacy preservation. En *2020 ieee international conference on cloud computing in emerging markets (ccem)* (pp. 52–56).

LEAF. (2021). *Website del proyecto LEAF.* Descargado de https://leaf.cmu.edu/

LEAF. (2022). *Repositorio del proyecto LEAF.* Descargado de https://github.com/TalwalkarLab/leaf

Li, T., Sahu, A. K., Talwalkar, A., y Smith, V. (2020). Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, *37*(3), 50–60. doi: 10.1109/MSP.2020.2975749

Li, X., Zhao, B., Yang, G., Xiang, T., Weng, J., y Deng, R. H. (2023). *A survey of secure computation using trusted execution environments.* Descargado de https://arxiv.org/abs/2302.12150

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., y Lin, Z. (2021). When Machine Learning Meets Privacy: A Survey and Outlook. *ACM Computing Surveys*, *54*(2). doi: 10.1145/3436755

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*(2), 129-137. doi: 10.1109/TIT.1982.1056489

McMahan, B., Moore, E., Ramage, D., Hampson, S., y y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. En *Artificial intelligence and statistics* (pp. 1273–1282).

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., y y Arcas, B. A. (2023). *Communication-efficient learning of deep networks from decentralized data.*

Mikołajczyk-Bareła, A., y Grochowski, M. (2023). *A survey on bias in machine learning research.* Descargado de https://arxiv.org/abs/2308.11254

MIT. (2021). *Keras: Web.* Descargado de https://keras.io/

Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., y Kourtellis, N. (2021). *Ppfl: Privacy-preserving federated learning with trusted execution environments.* Descargado de https://arxiv.org/abs/2104.14380

Oliva, G., Setola, R., y Hadjicostis, C. N. (2014). *Distributed k-means algorithm.* Descargado de https://arxiv.org/abs/1312.4176

Oneto, L., y Chiappa, S. (2020). Fairness in machine learning. En *Recent trends in learning from data* (p. 155–196). Springer International Publishing. Descargado de http://dx.doi.org/10.1007/978-3-030-43883-8_7 doi: 10.1007/978-3-030-43883-8_7

Stack Overflow. (2018). *Stack Overflow Data (BigQuery Dataset).* Descargado de https://www.kaggle.com/datasets/stackoverflow/stackoverflow

Sun, H., Li, J., y Zhang, H. (2024). *zkllm: Zero knowledge proofs for large language models.* Descargado de https://arxiv.org/abs/2404.16109

United States Goverment. (2021). *Sitio Web del proyecto EMNIST.* Descargado de https://www.nist.gov/itl/products-and-services/emnist-dataset

Wang, Z., Dong, N., Sun, J., Knottenbelt, W., y Guo, Y. (2024). *zkfl: Zero-knowledge proof-based gradient aggregation for federated learning.* Descargado de https://arxiv.org/abs/2310.02554

Wei, K., Li, J., Ma, C., Ding, M., Wei, S., Wu, F., ... Ranbaduge, T. (2024). *Vertical federated learning: Challenges, methodologies and experiments.*

Descargado de https://arxiv.org/abs/2202.04309

Weyand, T., Araujo, A., Cao, B., y Sim, J. (2020). Google landmarks dataset v2 - A large-scale benchmark for instance-level recognition and retrieval. *CoRR*, *abs/2004.01804*. Descargado de https://arxiv.org/abs/2004.01804

Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond. (2021). *MNIST: Web.* Descargado de http://yann.lecun.com/exdb/mnist/

Yao, A. C.-C. (1986). How to generate and exchange secrets. En *27th annual symposium on foundations of computer science (sfcs 1986)* (p. 162-167). doi: 10.1109/SFCS.1986.25

Zaman, F. (2020). *Instilling responsible and reliable ai development with federated learning.* https://medium.com/accenture-the-dock/instilling-responsible-and-reliable-ai-development-with-federated-learning-d23c366c5efd.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., y Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*, *216*, 106775.

Zhao, J. C., Bagchi, S., Avestimehr, S., Chan, K. S., Chaterji, S., Dimitriadis, D., ... Roth, H. R. (2024). *Federated learning privacy: Attacks, defenses, applications, and policy landscape - a survey.* Descargado de https://arxiv.org/abs/2405.03636