FLEA

APRENDIZAJE FEDERADO APLICADO

Reporte técnico

FLEA: Aprendizaje Federado aplicado a Analíticas de Aprendizaje

Autor(es)

Paola Bermolen, Germán Capdehourat, Lorena Etcheverry, Christian Fachola, María Inés Fariello, Agustín Tornaría

29 de diciembre de 2022

FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Facultad de Ingeniería,
Universidad de la República

Dirección:
Julio Herrera y Reissig 565

lorenae@fing.edu.uy

PALABRAS CLAVE:
Federated Learning, analíticas de aprendizaje

Reporte técnico

FLEA: Aprendizaje Federado aplicado a Analíticas de Aprendizaje

VERSION
1.0

FECHA
29 de diciembre de 2022

AUTOR(ES)
Paola Bermolen, Germán Capdehourat, Lorena Etcheverry, Christian Fachola, María Inés Fariello, Agustín Tornaría

PROYECTO
FLEA

CANTIDAD DE PÁGINAS Y ADJUNTOS
36

FLEA
APRENDIZAJE FEDERADO APLICADO

Índice general

Capítulo 1

Introducción

En los últimos años hemos experimentado el auge de la aplicación de técnicas de Aprendizaje Automático (AA) y técnicas de Inteligencia Artificial (IA) para la resolución de problemas en diferentes dominios. En su enfoque tradicional, la construcción, el entrenamiento y el despliegue de modelos de IA/AA implican protocolos de centralización e intercambio de datos. Los datos se fusionan, limpian e integran y, a continuación se utilizan para entrenar y evaluar los modelos. Sin embargo, este procedimiento se enfrenta a retos relacionados con la privacidad de las personas y la protección de los datos personales. Para que la aplicación de estas técnicas sea exitosa es necesario contar con datos abundantes y de buena calidad. Esto no es factible en muchas situaciones del mundo real, donde es frecuente que las organizaciones dispongan de volúmenes reducidos de datos o de baja calidad. Esto obstaculiza el uso de la tecnología de IA, haciendo que su aplicación sea más difícil de lo esperado. Podría pensarse que la solución es fusionar los datos en repositorios, tanto a la interna de las organizaciones como a lo largo de sectores de actividad, pero esto es inviable en muchos casos. La seguridad y privacidad de los datos, la competencia entre sectores, la distribución geográfica de datos, así como el costo de estos procesos y de almacenamiento, son algunos de los factores que hacen inviable la integración centralizada de datos en muchos contextos

Estas cuestiones de privacidad y ética son esenciales en la analítica del aprendizaje (LA, del inglés Learning Analytics), que es la aplicación de técnicas cuantitativas a los datos educativos para ayudar a resolver problemas como el diseño de trayectorias de aprendizaje o el desarrollo de alertas de abandono temprano. Los problemas de privacidad y el uso ético de los datos en actividades de ML en el contexto educativo han sido ampliamente documentados en la literatura [Dra+16; Ban+18; MG19].

En la literatura aparecen dos enfoques para garantizar la privacidad en el contexto de las Analíticas de Aprendizaje. Por un lado, el enfoque de publicación de datos con preservación de la privacidad, que consiste en aplicar técnicas de desidentificación y anonimización de datos (por ejemplo, satisfaciendo la definición de k-anonimity [Swe02]) y luego utilizar métodos convencionales de ML[KE16; Kyr+19]. Por otro lado, en el enfoque conocido como minería de datos que preserva la privacidad o control estadístico de la divulgación, el analista no accede directamente a los datos, sino que utiliza un mecanismo de consulta que añade ruido estadístico a la respuesta, implementando la privacidad diferencial [Dwo08]. Esta última estrategia puede ser más robusta y escalable, pero algunos autores sugieren que puede ser difícil de implementar y llevar a la práctica [Gur+17].

Sin embargo, estas estrategias de anonimizar los datos previo a la aplicación de técnicas de ML o IA, sólo son factibles cuando los datos ya han sido integrados y centralizados. El problema que busca estudiar este proyecto es **la realización de análisis, usando técnicas de ML/IA, sobre datos distribuidos, sin vulnerar la privacidad de los participantes y evitando la centralización de los datos.**

El Aprendizaje Federado (FL del inglés Federated Learning) es un enfoque descentralizado propuesto inicialmente por Google [Kon+16] 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, ML, 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 distribuido, hasta la extensión del concepto para cubrir escenarios de aprendizaje colaborativo entre organizaciones [Li+20].

El enfoque de FL parece ser adecuado para implementar modelos basados en IA/ML protegiendo la privacidad de los participantes. Este problema es particularmente relevante en el contexto de los datos educativos, y para el tipo de estudios que involucran a los datos que Plan Ceibal [Cei07] custodia. En este sentido, Plan Ceibal participa del proyecto identificando casos de uso de análisis relevantes que hoy no es posible realizar debido a limitaciones en el acceso a los datos por restricciones de privacidad. En particular se buscó identificar: casos que requieren del análisis conjunto de datos que custodia Plan Ceibal con datos sensibles provenientes de otras organizaciones y que por razones de protección de datos personales éstas no pueden compartir, y casos que requieran desarrollar modelos de ML sobre datos que residen en dispositivos manejados por Plan Ceibal y que por razones de privacidad no es deseable ni posible transferir a repositorios centralizados.

Este documento presenta el trabajo desarrollado en el marco del proyecto "FLEA: Uso de técnicas de Aprendizaje Federado para el análisis de datos sensibles y su aplicación al caso de Analíticas de Aprendizaje"financiado parcialmente por la ANII (Fondo María Viñas 2020, FMV_3_2020_1_162910)

1.1. Objetivos del proyecto

El objetivo general del proyecto es estudiar la posibilidad de desarrollar analíticas de aprendizaje basadas en técnicas de ML/IA que salvaguarden la privacidad y eviten la centralización, utilizando el enfoque de aprendizaje federado.

Los objetivos específicos del proyecto son los siguientes:

- Relevar y comprender los casos de uso de análisis a los que se enfrenta Plan Ceibal que no es posible realizar con técnicas tradicionales.
- Relevar las técnicas y herramientas de aprendizaje federado y estudiar la aplicabilidad a los casos de interés.
- Desarrollar al menos un caso de uso, haciendo particular énfasis en los aspectos metodológicos que son característicos del enfoque federado, en el desempeño de los métodos de ML, y en los niveles de privacidad alcanzados
- Evaluar experimentalmente el prototipo

1.2. Organización del documento

El resto del documento se organiza de la siguiente manera:

En el Capítulo 2 se definen conceptos preiliminares y se describen los casos seleccionados, que son dos problemas conocidos de clasificación de imágenes.

El Capítulo 3 presenta los experimentos realizados y los resultados de los mismos. Esto incluye experimentos en contexto centralizado y en el federado.

Por último, en el Capítulo 4 se discuten los resultados obtenidos y se presentan las conclusiones de los mismos.

Capítulo 2

Aprendizaje Federado: conceptos preliminares y relevamiento de herramientas

En la actualidad existen múltiples plataformas que permiten implementar soluciones basadas en Aprendizaje Federado. En esta Sección desarrollaremos algunos conceptos preliminares y luego presentamos un relevamiento de las plataformas existentes y su adecuación al escenario de estudio.

2.1. Conceptos Preliminares

El enfoque de Aprendizaje Federado se basa en realizar el entrenamiento de modelos de AA de forma distribuida. En este escenario, las entidades participantes suelen clasificarse como servidor y clientes, donde el servidor es quién orquesta el entrenamiento del modelo. Se lleva adelante un proceso iterativo, en el que en cada iteración (también denominada ronda) se eligen determinados clientes sobre los cuales se realizarán varias épocas de entrenamiento local del modelo. En estas épocas, cada cliente utiliza sólo los datos que son de su propiedad. Por último, el servidor agrega los resultados de todos los clientes para actualizar el estado del modelo, que será desplegado en nuevos clientes en la próxima iteración, repitiendo el proceso.

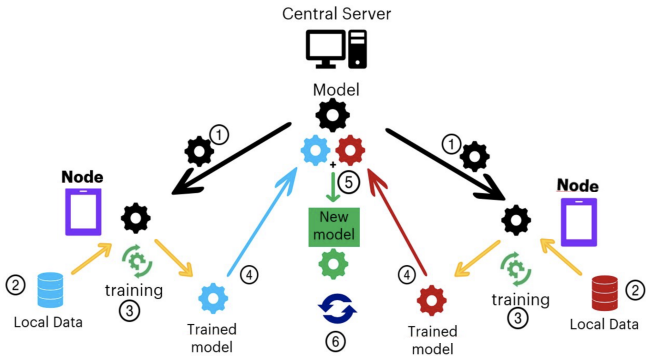

Figura 2.1: Arquitectura de una solución basada en Aprendizaje Federado. Fuente: Medium [Zam20].

En la Figura 2.1 se muestran los pasos necesarios para entrenar un modelo utilizando el esquema FL. Al comienzo, el servidor central envía los últimos parámetros del modelo a los nodos o parámetros iniciales en caso de ser la primer ejecución (paso 1). Luego, en el paso 2 se seleccionan datos en cada nodo y cada modelo local se entrena en función de los últimos parámetros (paso 3). Al finalizar el entrenamiento local cada cliente comunica los parámetros actualizados del modelo local al modelo global (paso 4) donde se combinan las actualizaciones de cada modelo generando un modelo nuevo (paso 5). Finalmente, el proceso se reinicia desde

el paso 1 (paso 6). El modelo generado en el paso 5 puede ser luego puesto en producción.

Presentaremos ahora una descripción más formal del escenario de Aprendizaje Federado para el caso de entrenamiento de una red neuronal, dado que este es el tipo de algoritmo más mencionado en la literatura del tema.

Una red neuronal para clasificación 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 de una red neuronal 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 $E \subset X$ para los cuales ya se conoce el valor de $f(x)$ para cada $x \in E$. 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 la red 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 E se particiona en un número de lotes de cierto tamaño fijo para todos, llamado *batch-size*[1]. Para procesar un lote B se calculan las salidas para todos los $x \in B$ 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. Las redes neuronales pueden entrenar por muchas épocas, es decir puede que vean el mismo dato muchas veces, tantas como épocas haya entrenado.

En la situación descrita anteriormente está implícito que el modelo tiene acceso a todos los datos del conjunto E 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 necesarios al mismo tiempo. Se plantea un esquema de servidor-cliente, en el que un servidor central aloja un modelo, que se despliega en un cliente, quien dispone de los datos, donde se entrena una versión del modelo. Este proceso se realiza en un conjunto de clientes a la vez y luego se agrega lo aprendido por cada modelo para actualizar la versión central alojada en el servidor. Diremos que cada uno de estos ciclos de despliegue, entrenamiento local en clientes seleccionados y actualización del modelo del servidor, constituye una *ronda*. Este esquema se implementa para el caso de redes neuronales mediante el algoritmo de *federated averaging* [McM+17], allí se describe una manera de ajustar los pesos del modelo del servidor a partir del ajuste que se tiene que hacer en los pesos en cada cliente.

Entonces ,en el caso federado para redes neuronales, hay que considerar los párametros que definen los comportamiento de los modelos en los clientes (épocas y *batch-size*), pero también los que son propios de la federación, a saber: la cantidad de rondas, la cantidad de clientes elegidos por ronda, la cantidad de clientes total y cómo se distribuyen los datos entre ellos.

Los parámetros antes mencionados, como la cantidad de rondas de entrenamiento, la cantidad de clientes a seleccionar por ronda, los criterios de selección de los clientes que participan de las rondas de entrenamiento, los mecanismos para particionar los datos localmente, la cantidad de épocas de entrenamiento local, etc. pueden influir, a priori, en el desempeño de los modelos obtenidos. Uno de los objetivos de este proyecto es experimentar en este sentido para entender los efectos de estos parámetros. En el Capitulo [**capitulo4**] se elaborará sobre este tópico.

A continuación presentaremos un panorama general sobre las herramientas y plataformas para implementar soluciones con Aprendizaje Federado.

[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 $batch - size$

2.2. Herramientas y plataformas

Existen múltiples consideraciones a la hora de evaluar las herramientas disponibles para implementar soluciones mediante Aprendizaje Federado. 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. Más allá de esto, al momento de este relevamiento existe una serie de herramientas y plataformas que cuentan con un grado de desarrollo y madurez considerable.

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 [Kai+21; Zha+21a]. A continuación reseñaremos algunos de los que nos parecen más relevantes.

Modelos y algoritmos disponibles Un aspecto importante al tener en cuenta a la hora de seleccionar una plataforma en particular son los algoritmos de ML 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 [McM+17], 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 Otra punto en el cuál se tendrá que tomar una decisión es 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. Este escenario, también denominado cross-silo busca resolver el problema de integración de datos de diferentes orígenes. Para este enfoque no hay actualmente versiones de todos los modelos, el más estudiado es la regresión logística [Zha+21a] .

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 mencionamos: 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 y 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 Aprendizaje Federado involucran un amplio espectro de problemas y áreas de investigación, no sólo vinculadas al Machine Learning 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.

2.3. Herramientas relevadas

En esta sección presentamos las herramientas relevadas, analizando los aspectos mencionados en la sección anterior. Además, compilamos 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)

2.3.1. 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 reduciento 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 [Uni21] [Coh+17] o Extended MNIST es un conjunto de datos creado a partir del dataset MNIST [Yan21], 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.

CIFAR-10 y CIFAR-100 CIFAR-10 y CIFAR-100 [Ale21] [KH+09] 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 [BP21] 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.

Shakespeare Este conjunto de datos, publicado en la plataforma Kaggle, contiene transcripciones de todas las obras de William Shakespeare[Kag16]. 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[Sta18] 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 [Wey+20] 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 [Cal+18] 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. n el sitio web del proyecto hay una guía para instalarlo, tutoriales, documentación de la API [LEA21]. El código es abierto y está alojado en un repositorio GitHub de acceso público [LEA22].

2.3.2. TensorFlow Federated

TensorFlow Federated (TFF) [Goo21k] [Goo21a] es un framework desarrollado por Google basado en TensorFlow [Goo21m]. Al momento de este relevamiento solo puede ejecutar en una máquina por lo que la parte federada se hace mediante simulaciones que soportan un gran volúmen de clientes y datos.

TFF está organizado en dos capas Federated Learning[Goo21f] y Federated Core[Goo21e]. 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[MIT21] 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 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 [Goo21d], aunque tiene la posibilidad de implementar más mediante Federated Core.

TFF incluye módulos que permiten probar el escenario federado con diferentes conjuntos de datos [Goo21c]. Dentro de los conjuntos disponibles se destacan Federated CIFAR-100, Federated EMNIST, Federated Google Landmark v2, Shakespeare y StackOverflow, mencionados previamente.

Esta plataforma ciene acompañada de una extensa documentación [Goo21b], junto con guías[Goo21h] y diversos tutoriales en formato de notebooks[Goo21j] 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 [Goo21l] como en plataformas como Stack Overflow [Goo21i] y en el repositorio de GitHub una entrada para reportar problemas o *issues* [Goo21g].

2.3.3. Flower

Flower [Beu+20] 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 ML, como por ejemplo PyTorch, TensorFlow, e inclusive NumPy entre otras. Además, Flower permite migrar procesos de entrenamiento de ML 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 [Flo22a] y código disponible en su repositorio público [Flo22b].

Capítulo 3

TFF vs Flower

El objetivo de este capítulo es presentar los experimentos realizados para comparar, en diferentes sentidos las plataformas TFF y Flower. El esquema federado trae ventajas, pero es importante saber si existe una pérdida de desempeño con respecto a los algoritmos centralizados. Por eso la pregunta principal a responder es: ¿Pueden los modelos federados alcanzar desempeños similares a su equivalente centralizado? y además, ¿existen diferencias en el desempeño relativo de las diferentes plataformas?.

Cabe también preguntarse si siempre es posible alcanzar un desempeño similar al centralizado ¿Se puede lograr en la misma cantidad de épocas totales o de tiempo? También interesa realizar cierto análisis de sensibilidad de los parámetros.¿Qué balance existe entre la cantidad de rondas de entrenamiento y la cantidad de épocas que cada cliente entrena localmente? ¿Cuál afecta más al desempeño? Partimos de la hipótesis de que el desempeño debería ser similar entre ejecuciones con la misma cantidad de épocas totales.

Comenzaremos por presentar los dos casos de prueba elegidos para evaluar las herramientas elegidas para la federación: Tensorflow Federated(TFF) y Flower (Flower). Utilizaremos los mismos conjuntos de datos y reproduciremos cada escenario en ambas plataformas, evaluando los resultados obtenidos, el desempeño general, así como la evolución de diferentes parámetros (ej: cantidad de rondas necesarias para alcanzar ciertos umbrales de precisión).

El escenario utilizado para comparar las plataformas es el de un problema de clasificación supervisada de imágenes. Se eligen como conjuntos MNIST y CIFAR10 porque definen problemas de clasificación conocidos que son ampliamente usados cómo linea base para la prueba de modelos en el área de la clasificación de imágenes. Ambos conjuntos fueron presentados en la Sección 2.

El problema de clasificación dado por el conjunto de datos MNIST consiste en clasificar imágenes de dígitos del 0 al 9. Este conjunto de datos sirve como primera aproximación a las herramientas de Aprendizaje Federado a utilizar y fue elegido especialmente porque define un problema que actualemnte es sencillo de resolver en el contexto centralizado. El conjunto de datos CIFAR10 consiste en imágenes que pertenecen a diez clases de objetos distintos (ejemplo: aviones, autos, etc). De acuerdo con la literatura especializada, conseguir buenos resultados en este caso presenta un desafío mayor que en el caso del conjunto de datos MNIST .

En la Sección 3.1 se describe la lógica de los experimentos realizados, y luego en las Secciones 3.2 y 3.3 se presentan los resultados obtenidos. Para culminar, la Sección 3.4 discute los resultados obtenidos y presenta algunas conclusiones.

3.1. Diseño experimental

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 dos soluciones federadas, una usando TFF y la otra Flower.

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 utilizaremos 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 medirán 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 corrieron 20 épocas totales. En cada caso se presentaran primero los resultados del caso centralizado, que servirá de referencia y a continuación de cada uno se dan los resultados del algoritmo federado.

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

Caso MNIST

Como ya se mencionó, el conjunto de datos elegido es una de las versiones del conjunto MNIST [Yan21], 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.

Los datos fueron 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 las Figuras 3.2 y 3.2.

Figura 3.1: Ejemplo de algunos dígitos del conjunto MNIST

Experimentaremos con una arquitectura de red convolucional (CNN, por sus siglas en inglés), que es de uso común en los problemas de imágenes. Tenemos 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. 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 fueron 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

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

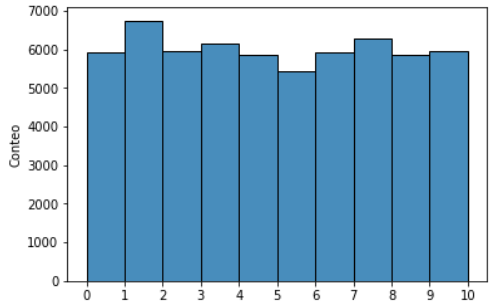

Figura 3.2: MNIST: Distribución por dígito en el conjunto de entrenamiento

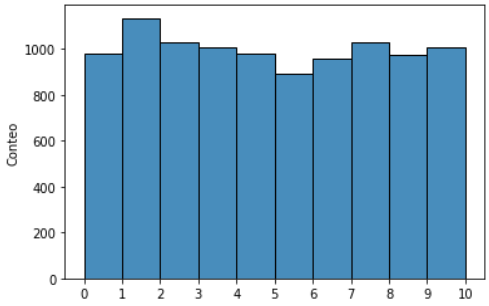

Figura 3.3: MNIST: Distribución por dígito en el conjunto de test

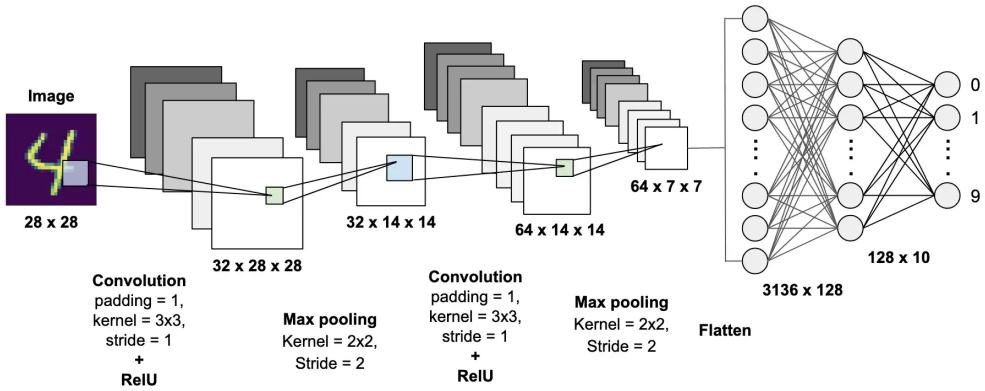

Figura 3.4: Arquitectura de la red convolucional usada para clasficar los dígitos de MNIST. Fuente: TowardsDataScience

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. 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 aprecia en las Figuras 3.6 y 3.7

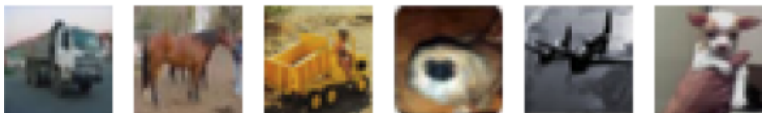

Figura 3.5: Ejemplo de algunas imágenes del conjunto CIFAR10

En este caso se decidió experimentar con distintas arquitecturas en cada herramienta. En TFF se utilizó un modelo convolucional, del mismo tipo que se uso para MNIST, pero con más capas de convoluciones y una capa densa con más neuronas. En el caso de Flower se utilizó la arquitectura MobileNetV2 [San+18], una arquitectura más compleja que la anterior y con más parámetros entrenables.

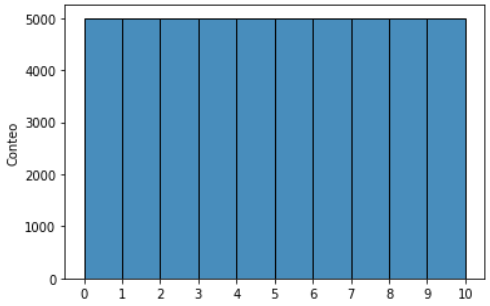

Figura 3.6: CIFAR10: Distribución en el conjunto de entrenamiento

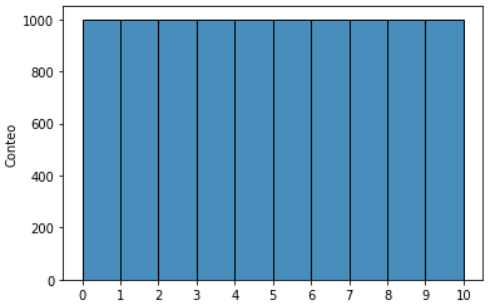

Figura 3.7: CIFAR10: Distribución en el conjunto de test

3.2. Resultados para el Caso MNIST

En primer lugar se entrenó la red de manera centralizada. Se intentó 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 entrenó 50 épocas, luego de cada época se calcularon 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. Tomamos entonces 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.

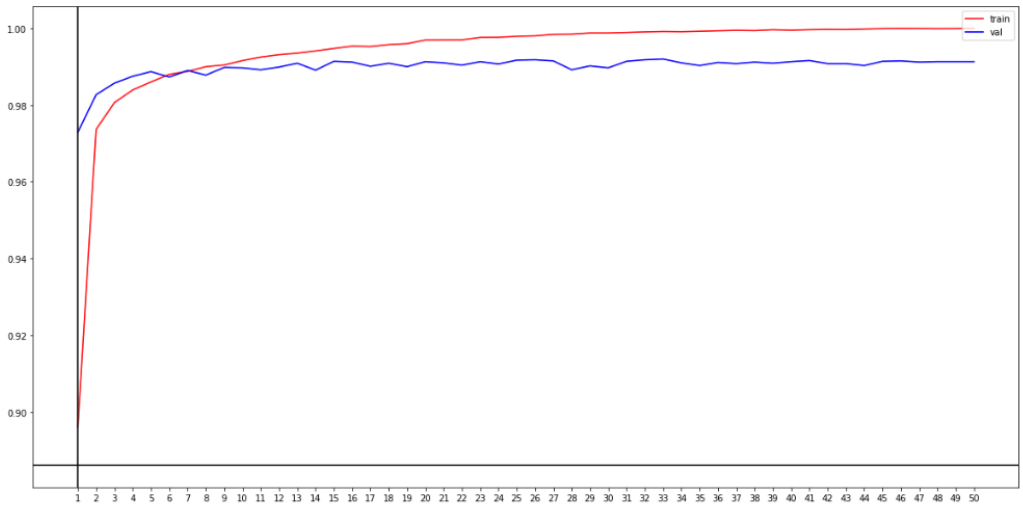

Figura 3.8: MNIST: Comparación de *accuracy* entre entrenamieniento y test en el caso centralizado. Épocas en el eje de las abscisas, la métrica en las ordenadas

Para el caso federado utilizamos un esquema de entrenamiento con 10 clientes, durante 20 rondas, en las cuales participaron siempre todos los clientes, cada uno entrenando durante una época local, dando un total de 20 épocas totales por cliente. Se obtuvo una *accuracy* de 0.98 % con TFF en 205 segundos y una de 0.99 % en Flower.

3.3. Resultados CIFAR10

Como se adelantó, para este caso se utilizó una arquitectura distinta en cada herramienta. Se presentan los resultados por cada plataforma.

3.3.1. Caso CIFAR10 con TFF

Empezando por la red convolucional elegida para TFF, el primer resultado es el centralizado. Se entrenó durante 120 épocas con el fin de encontrar la cantidad óptima. La gráfica de la Figura 3.9 presenta la métrica de *accuracy* en el conjunto de entrenamiento y de test. Se tomó como punto de corte la época 60, donde se obtuvo una *accuracy* de 78 % en test en un tiempo de 6300 segundos.

Se fija entonces en 60 el número de épocas totales, y realizamos ejecuciones con distintas cantidades de rondas y épocas para evaluar el caso federado, cambiando 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 x/y clientes significa que para ese experimento se seleccionaron aleatoriamente en cada ronda x clientes de los y totales. Resumimos los resultados en la Tabla 3.1.

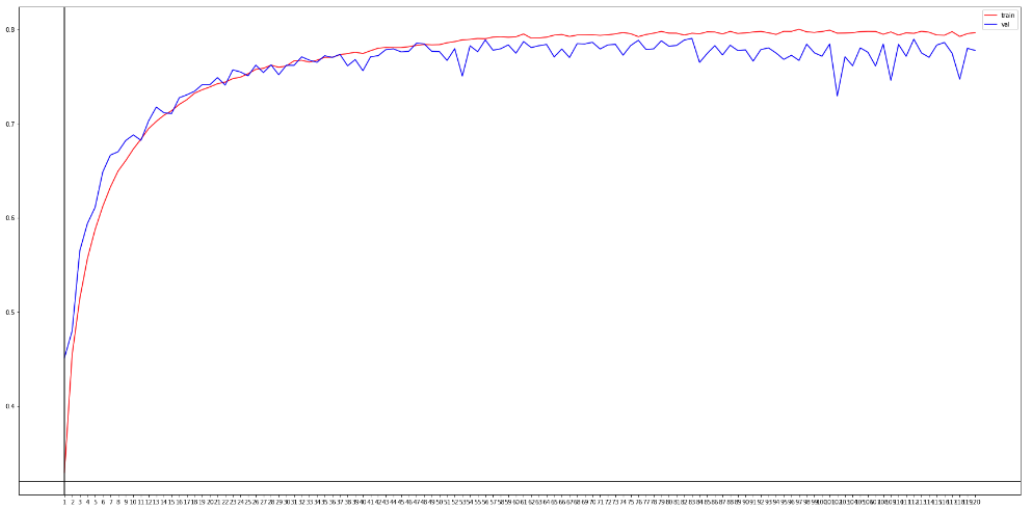

Figura 3.9: CIFAR10 con TFF: Comparación de *accuracy* entre entrenamieniento (rojo) y test (azul) en el caso centralizado. Épocas en el eje de las abscisas, la métrica en las ordenadas

3.3.2. Caso CIFAR10 con Flower

Para este caso, como ya comentamos, se utilizó la arquitectura MobileNetV2. Así como en los casos anteriores, primero tratamos de encontrar el numero óptimo de épocas utilizando un esquema centralizado. Se entrenó durante 120 épocas y evaluando análogamente a los experimentos mostrados anteriormente, se encontró que la cantidad óptima de iteraciones está en el entorno de las 60 épocas. Con este número se logró un desempeño en el conjunto de test de 76 % de *accuracy* en un tiempo de 7385 segundos.

Proseguimos a realizar los experimentos federados correspondientes, con un esquema idéntico al de MNIST. Los resultados obtenidos se muestran en la Tabla 3.2.

Cuadro 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.

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

Cuadro 3.2: CIFAR10 con Flower. 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.

Clientes	Rondas	Epocas	Accuracy	Tiempo(s)
2/2	15	4	75 %	4967
2/2	30	2	75 %	5013
2/7	15	4	64 %	1507
2/7	30	2	63 %	2542
5/7	15	4	68 %	3698
5/7	30	2	67 %	5574

3.4. Discusión y conclusiones

En cuanto a los resultados asociados al caso MNIST, vemos que 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.

Centrémonos ahora en el caso CIFAR10. En primer lugar se pude decir que la CNN desempeña un poco mejor en términos de *accuracy* que la MobileNetV2, lo cual ya se puede ver al comparar los resultados centralizados y se replica en los federados, donde sistemáticamente se tienen de 1 a 2 puntos porcentuales de diferencia. Por otro lado, utilizando menos clientes se obtienen mejores resultados, ver los experimentos con 2/2 clientes en las Tablas 3.1 y 3.2. Es posible que esto se deba a que en el resto de los casos se están utilizando menos datos (recordemos que los datos se distribuyen entre la cantidad de clientes considerada en cada caso). Sería interesante realizar los experimentos con 7/7 para ver si efectivamente lo que esta pasando es que falta usar todos los datos.

Por último, en cuanto al análisis de sensibilidad de los parámetros, 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, así que aumentar la cantidad de épocas locales implícitamente dice que se esperan clientes que no se den de baja rápidamente.

Capítulo 4

Relevamiento de casos de uso de aplicación de Aprendizaje Federado en el contexto de Learning Analytics

Como se mencionó anteriormente, uno de los objetivos de este proyecto es identificar posibles casos de uso de FL que podrían servir a las necesidades de Plan Ceibal [Cei07], en el contexto de *Learning Analytics*. En base a la revisión del trabajo reciente en esta área emergente dentro del aprendizaje automático (ML), hemos seleccionado algunos casos de uso donde se considera que un esquema federado podría ser aplicado. Si bien la descripción de cada caso se aborda de manera genérica, se busca ilustrar cada una con una aplicación concreta.

4.1. Perfiles de uso de dispositivos o navegación web

Dentro de las responsabilidades de Plan Ceibal, la entrega de dispositivos a los beneficiarios es sin lugar a dudas una de las más relevantes. Si bien el dispositivo una vez entregado pasa a ser de propiedad de cada beneficiario, es relevante para Plan Ceibal el monitoreo y mantenimiento de los dispositivos, en tanto y en cuanto serán la herramienta fundamental para el uso de las distintas plataformas y acceso al contenido educativo.

En tal sentido, suele ser de interés conocer el uso que los beneficiarios dan a los dispositivos (ej. cantidad de días a la semana que los usan, cantidad de horas, aplicaciones que usan, consumo de batería, entre otras). Sin embargo, resulta también claro que todos estos datos son privados y sería de suma utilidad analizarlos sin necesidad de recolectarlos en forma centralizada.

Algo similar ocurre con los datos de navegación web que los navegadores suelen recolectar. Es información muy valiosa saber las distintas páginas a las que acceden para generar distintos *perfiles* de usuario y poder luego usar dicha información para *personalizar* distintos contenidos que Ceibal brinda.

En ambos casos, se trata de generar un perfil de cada usuario, pero tratando de evitar la recolección centralizada de los datos crudos. Un problema similar es el que está analizando Google para su navegador Chrome, quienes están trabajando en lo que denominaron *Federated Learning of Cohorts*[1]. En ese caso se busca cubrir una demanda de los anunciantes en medios digitales (que actualmente se resuelve en base a *cookies*) que quieren conocer el perfil del usuario que está accediendo a un sitio web [Epa+21] para personalizar sus campañas publicitarias. Por el momento esta es una propuesta en discusión [Lan21], y este debate sobre el compromiso óptimo entre privacidad y personalización parece estar aún lejos de su fin [Kut+21].

[1] `https://en.wikipedia.org/wiki/Federated_Learning_of_Cohorts`

4.2. Clustering de usuarios

Este caso es similar al anterior, pero donde el objetivo es algo diferente, o podría considerarse como paso previo. En diversos casos de interés se sabe poco sobre los datos antes de tomar contacto con ellos, y deben ser explorados antes de poder utilizarlos para hacer algún tipo de segmentación de los beneficiarios. Por otro lado, hay casos donde no se busca usarlos con dicho fin, sino simplemente para hacer un análisis de lo observado. En Plan Ceibal se ha trabajado en este tema, tanto con datos del uso de la red, como con datos de plataformas educativas. Si bien este punto está muy relacionado con el anterior, aquí la idea no está enfocada en la personalización de servicios, sino más bien en el análisis exploratorio de datos mediante clustering.

Existen diversos casos donde los perfiles buscados no son conocidos a priori, o ni siquiera está claro qué acciones se podrían tomar luego de conocerlos. Es decir que este problema podría ser una etapa previa, más de descubrimiento de los datos, donde se busca conocer los distintos patrones de comportamiento observados. En tal sentido, lo que se pretende hacer mediante aprendizaje federado, no es otra cosa que clustering tradicional, pero evitando nuevamente la recolección y centralización de los datos. Es decir que se busca implementar de manera distribuida en un esquema federado, algoritmos clásicos de clustering como *k-means* [Rod+20] (ver sección 4.2 y 5.3, así como las referencias que allí se citan y el notebook de ejemplo en Github), SOM [Kho+21] y otros [WC20].

Al igual que cuando se hace clustering de manera centralizada, hay una etapa posterior, cuando las clases ya fueron encontradas, de identificar a qué patrones corresponde cada una (ej. los que tienen mucha, poca o nula actividad). Para ello, es necesario contar con los datos, porque lo que se hace habitualmente es calcular estadísticas sobre los datos de los distintos clústers. Aquí nuevamente se agrega un desafío en el enfoque federado, ya que esta etapa también es necesario hacerla en forma distribuida. Es decir que no alcanza con completar la segmentación, sino que luego es necesario hacer esta pasada de *explicabilidad* sobre las clases resultantes para que los hallazgos sean relevantes.

4.3. Detección de anomalías

Otro tipo aplicaciones de aprendizaje automático de relevancia en la educación, son aquellas donde se busca identificar o predecir comportamientos anómalos, que en algunos casos podrían ser indicios de alguna dificultad en el aprendizaje. Por ejemplo, podemos pensar en el caso de la dislexia o la disgrafía, que suelen ser detectables a tempranas edades mediante diagnósticos adecuados [Ass+18; DD20; RD16; Kar+19].

En este tipo de casos, asumimos que los datos locales que se podrían recolectar en cada dispositivo no cuentan con ningún tipo de etiqueta[2]. Por lo tanto, dado que no podemos centralizar los datos, y que además no tenemos etiquetas, no es posible en este caso *entrenar* un modelo de forma federada.

La alternativa que sí podemos implementar de forma federada en dicha situación, es un detector de anomalías [Sin+21]. Según la aplicación habrá que ver la utilidad práctica que esto podría tener, ya que a priori lo único que podría indicar el modelo entrenado, es qué datos (y por ende qué beneficiarios) corresponden a un comportamiento que se aparta de la *normalidad*. Este tipo de enfoques ha sido utilizado en otras áreas, como pueden ser la detección de anomalías en redes de datos [Zha+19] , así como aplicaciones de IoT en el ámbito industial [Liu+21b].

Si bien en este escenario federado es difícil profundizar en la explicabilidad de los resultados ante la imposibilidad de acceder a los datos, tal vez según el caso sea posible además de detectar una anomalía, identificar *para qué lado* se da dicho apartamiento. Esto es algo que podría tener implicancias relevantes según el caso de uso específico (por ejemplo, identificar estudiantes con dificultades por un lado y estudiantes muy avanzados por otro). Otra posibilidad para lidiar con la falta de etiquetas, es usar un esquema de aprendizaje semi-supervisado [Zha+21b]. Es este caso, si bien a nivel local cada dispositivo no cuenta con etiquetas, a nivel

[2] Para etiquetarlos sería necesario contar con conocimiento de expertos, que sean capaces de diagnosticar en base a lo que hacer cada usuario.

centralizado sí se tiene un conjunto de datos curado con etiquetas asignadas por expertos, que pueden ser aprovechados en el desarrollo del modelo.

4.4. Corrección automática o asistida

Como mencionamos en el ejemplo anterior, en la mayoría de las aplicaciones que podrían resultar de interés para Plan Ceibal, es necesario que algún actor externo participe en el etiquetado de los datos que generan los beneficiarios (en su mayoría estudiantes de primaria y secundaria). En el caso previo, esto no era posible que lo realice un docente, ya que se trataba de problemas de difícil diagnóstico, que requieren en general de la evaluación de un experto del área (ej. psicólogo, psicomotricista, fonoaudiólogo, etc). Sin embargo, existen muchos otros casos, donde es el propio docente referente de cada estudiante quien podría ser el encargado de etiquetar los datos. Además, en ese caso no habría problemas de privacidad, ya que el docente sí accede a los datos de cada estudiante (ej. las tareas que realiza, lo que escribe, los ejercicios que hace, etc).

Por lo tanto, este último caso de uso, plantea la necesidad de agregar un actor adicional al esquema federado clásico. Se asume que los problemas de privacidad que estamos abordando, no son tales si la situación se plantea en términos locales, a nivel de cada grupo, en particular en la relación docente-estudiante. En tal sentido, podría haber casos donde los beneficiarios generan datos locales en sus dispositivos, que no tienen que ser privados para sus respectivos docentes, quienes podrían *etiquetarlos* con conocimiento experto según la aplicación específica. Esta *corrección* que brinda el docente, podría aprovecharse en un esquema federado, para entrenar un sistema de corrección automática, que pueda servir para asistir a los docentes en la corrección.

Incluso podría ser útil para detectar estudiantes con ciertos problemas particulares, en base a esta corrección asistida, que luego está claro puede ser siempre validada por un docente que mira los datos crudos de su estudiante que pertenece a su clase (sin violar por tanto temas de privacidad).

En base a los casos identificados previamente, en esta sección se busca avanzar un paso más, y definir al menos un ejemplo concreto para cada uno. Esto implica describir brevemente con qué datos se trabajaría, cómo podrían recolectarse, qué algoritmos de aprendizaje o modelos sería posible entrenar, y finalmente qué aplicación y forma de despliegue del modelo entrenado se propone.

4.5. Segmentación de beneficiarios

En base al caso descrito en 4.1, se puede definir un ejemplo concreto para Plan Ceibal, que involucre ambos datos allí mencionados. Es decir, se puede pensar en una forma posible de segmentación de los beneficiarios, en base a dos fuentes de datos a recolectar en sus dispositivos:

- Sitios web a los que accede: es posible recolectarlos del navegador.
- Aplicaciones que utiliza: es posible recolectarlos con el *launcher* en tablets, o a nivel del sistema operativo en laptops.

El objetivo es que el modelo sea capaz de inferir un perfil asociado a cada beneficiario, que luego permita una atención más personalizada y con una oferta de servicios acorde a sus intereses y necesidades.

Según el caso, se deberá analizar previamente cómo se puede definir el modelo, y si se usa algún tipo de taxonomía previa para generar los agrupamientos. También es posible que sea necesario hacer antes una etapa exploratoria de los datos, para usar los resultados obtenidos en la generación de los perfiles.

4.6. Clustering distribuido

El segundo caso presentado en 5.1.3 refiere al análisis de patrones comunes que puedan formar conjuntos o *clases* bien definidos. Estos agrupamientos podrían ser útiles para luego ser usados en un caso como el

presentado en la sección anterior, o simplemente como forma de analizar el comportamiento en distintas plataformas y servicios que Plan Ceibal ofrece a los beneficiarios.

Un ejemplo concreto sería el análisis de datos en las distintas plataformas, como CREA, PAM, Matific y la Biblioteca País. En diversas situaciones, es de interés analizar una o varias de dichas fuentes, y hacer un estudio basado en clustering para identificar patrones de comportamiento típicos. Estos datos, en vez de centralizarse como se hace actualmente, podrían mantenerse en forma local sin ser accedidos, implementando algoritmos distribuidos para la búsqueda de agrupamientos.

Para este caso Ceibal ya cuenta con datos de diversas plataformas, por lo que se podría evaluar algoritmos distribuidos y comparar los resultados con sus versiones tradicionales. De esa forma es posible identificar los posibles compromisos que se puedan dar entre las garantías de privacidad y la precisión de los resultados que arroja el análisis.

4.7. Detección temprana de problemas como disgrafía o dislexia

Existen diversos ejemplos asociados al caso de la sección 4.3, enfocados en masificar el screening de distintos problemas típicos que afectan el aprendizaje. En general se basan en el desarrollo de aplicaciones específicas, trabajando junto a expertos en la temática en cuestión (ej. psicólogos, psicomotricistas, fonoaudiólogos, otorrinolaringólogos, etc). Un ejemplo de este tipo es la línea de trabajo de Guillermo Sapiro en Duke University, que busca desarrollar una aplicación para hacer campañas de screening para la detección de autismo [SHD19; Bov+21].

En este caso la recolección de los datos se hace en la propia aplicación desarrollada para un fin específico. Si bien podría haber otros, dos ejemplos concretos que podrían considerarse son la detección de dislexia y disgrafía [Ass+18; DD20; RD16; Kar+19]. En este caso la aplicación permitiría a los docentes, no expertos en estos temas, visualizar alarmas que la aplicación podría generar de forma auotmática, y usar esto como disparador para solicitar la intervención de un profesional en la materia. Se podría pensar un tercer caso asociado al habla, algo análogo a los anteriores, pero donde los estudiantes en vez de escribir en la aplicación se graben y sea el audio lo que se analiza para detectar posibles dificultades.

4.8. Corrección asistida de la lectura

Finalmente tenemos el caso de la corrección asistida 4.4, donde el esquema sería similar al del caso anterior, ya que la recolección de datos se haría en la aplicación específica a desarrollar. Lo que difiere en este caso es el esquema de comunicación con el docente responsable, quien tendría permisos para ver los datos de su grupo y poder calificar las tareas de cada estudiante, logrando de esa forma generar datos etiquetados.

Por lo tanto, un esquema posible sería que los nodos del aprendizaje federado en este caso sean los docentes, siendo ellos quienes recolectan los datos de su grupo y asignan las etiquetas correspondientes. Además, también es para ellos el resultado del modelo aprendido, por lo que tiene sentido que luego el despliegue se haga también en sus dispositivos.

En este caso se pueden considerar varios ejemplos, como puede ser la corrección automática de redacciones. Este es un punto que ya se ha analizado en el caso de Ceibal en inglés, ya que es el único punto de las evaluaciones que no se realiza de manera automatizada (el resto es múltiple opción). Otra posible aplicación tiene que ver con el desarrollo de la lectura, donde Ceibal ha trabajado en una propuesta para evaluar de manera automática el aprendizaje y las dificultades de cada estudiante, mediante el procesamiento del audio grabado de los alumnos leyendo textos conocidos previamente. Se puede pensar en un esquema donde el docente recibe dicha grabación y hace una evaluación, identificando puntos positivos y negativos de la lectura en voz alta, que son luego utilizados como etiquetas para el entrenamiento de modelos automáticos.

El análisis de las posibles aplicaciones que el aprendizaje federado tendría en Plan Ceibal, muestra que hay un abanico de opciones donde podría comenzar a evaluarse su aplicación y tener resultados concretos. Se

plantean opciones tanto para casos de uso más inmediatos, como puede ser la segmentación e identificación de perfiles de beneficiarios, como otros a más largo plazo, que involucran el desarrollo de aplicaciones específicas que atienden diversos aspectos relacionados con los aprendizajes.

En base a este relevamiento se plantea un intercambio con Plan Ceibal, para definir en conjunto cuál de estas líneas priorizar a la hora de avanzar con el proyecto. Se decide profundizar en el caso de clustering distribuído, y de la implementación de algoritmos de predicción de deserción.

Capítulo 5

Implementación y Evaluación de casos de Aprendizaje Federado en el contexto de Learning Analytics

Uno de los objetivos del presente proyecto consiste en evaluar si existe una pérdida de desempeño en las soluciones basadas en FL en comparación con las soluciones tradicionales que operan sobre los datos en forma centralizada. Para ello se eligieron dos casos de prueba, un primer caso sencillo para validar el uso de las herramientas seleccionadas, y el segundo más exigente sobre el cuál se realizan todos los experimentos que pretenden comparar la performance entre el caso tradicional y el federado.

Como ya vimos en los Capítulos 2 y 3, existen diferentes parámetros y decisiones de diseño a tomar que afectan el desempeño de los modelos federados, a saber, 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 es que ejecutamos un mismo algoritmo de Aprendizaje Federado en cada modelo predictivo con variadas configuraciones de parámetros, 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.

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, vemos una clara oportunidad para aprovechar las ventajas del aprendizaje federado. Los sistemas educativos suelen estar compuestos por diferentes centros educativos (por ejemplo, guarderías, institutos, etc.). 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.

Nuestro objetivo es proporcionar mecanismos que permitan estudiar la información generada en cada centro educativo de forma global pero evitando consolidar los datos brutos generados en cada centro. Este esquema mejora la gestión de los datos en términos de preservación de la privacidad. Evaluaremos el impacto que esto tiene en los resultados del análisis de los datos. El primer paso para lograr nuestros 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.

Como ya se ha mencionado, podemos diferenciar dos esquemas de aprendizaje federado: cross-device u horizontal y cross-silo o vertical. El primero se refiere a aplicaciones similares a la que acuñó el aprendizaje federado: el teclado predictivo de los smartphones. Los problemas de comunicación juegan un papel relevante en este caso, ya que los dispositivos sólo están disponibles en ocasiones, lo que dificulta las rondas de

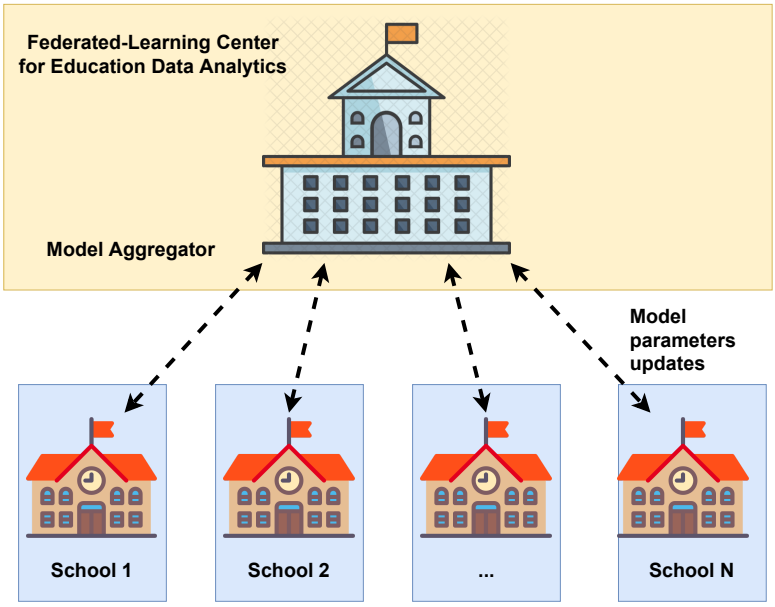

Figura 5.1: Our proposal for a cross-silo federated learning scheme for centralized data analysis of the educational system.

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.

Otra diferencia significativa entre los esquemas de aprendizaje federado es la forma en que se organizan los datos. En el caso entre dispositivos, se suele hablar de partición horizontal, ya que la estructura de datos en los distintos dispositivos es la misma. Cada dispositivo 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, la partición vertical corresponde al caso en que 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 cuanto al caso educativo que nos interesa, parece claro que se corresponde con un esquema cross-silo. En nuestra 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 podríamos tener plataformas educativas en la nube, donde los datos se alojan en servidores de terceros. Sin embargo, suponemos 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.

En la figura 5.1, ilustramos el esquema cross-silo propuesto y su aplicación en el sistema educativo. 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 brutos. Esto permitiría 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.

5.1. Diseño experimental y resultados

Como ya se ha mencionado, nuestro trabajo se centra en estudiar la aplicabilidad del Aprendizaje Federado a dos tareas diferentes: la predicción del abandono escolar y la clasificación no supervisada de estudiantes. Esta sección presenta nuestra implementación de cada tarea utilizando marcos de Aprendizaje Federado y los experimentos llevados a cabo en cada caso.

Debido a que no pudimos disponer en tiempo y forma de datos locales proporcionados por Plan Ceibal, utilizamos para evaluar nuestro enfoque datos públicos de educación utilizados en un desafío de la KDDCup 2015[KDD15]. Comenzaremos por describir los datos utilizados en la Sección 5.1.1. Luego, describiremos las dos tareas implementadas y luego pasaremos a mostrar los resultados experimentales y conclusiones.

5.1.1. Descripción del conjunto de datos

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.

Nuestras tareas de preparación de datos 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 nuestro repositorio [FLE22].

5.1.2. Tarea 1: predicción de abandono

Para esta tarea utilizamos el enfoque presentado en [FTL19] para predecir el abandono de los estudiantes. Para cada $enroll\_id$ en el conjunto de datos, se sabe si el estudiante abandonó el curso. A continuación, utilizamos estas etiquetas para entrenar y probar un modelo de aprendizaje profundo que predice el abandono. Utilizamos una arquitectura DNN 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. Utilizamos el optimizador Adam [KB14] 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 precisión de la configuración centralizada, y 2) evaluar la influencia de los parámetros mencionados sobre la precisión 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, utilizamos el algoritmo Federated Averaging [McM+17] 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, necesitamos 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.

Ejecución de los experimentos

Se entrena un modelo centralizado durante 20 épocas. El número de épocas se eligió empíricamente; buscamos un número lo suficientemente grande como para permitir el ajuste de parámetros del enfoque federado que también mantuviera un nivel aceptable de precisión sin demasiado sobreajuste. Tomamos como muestra el 70 % de los nombres de usuario de todos los estudiantes y recopilamos 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 precisión 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 probamos 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 precisión se muestran en la figura 5.2.

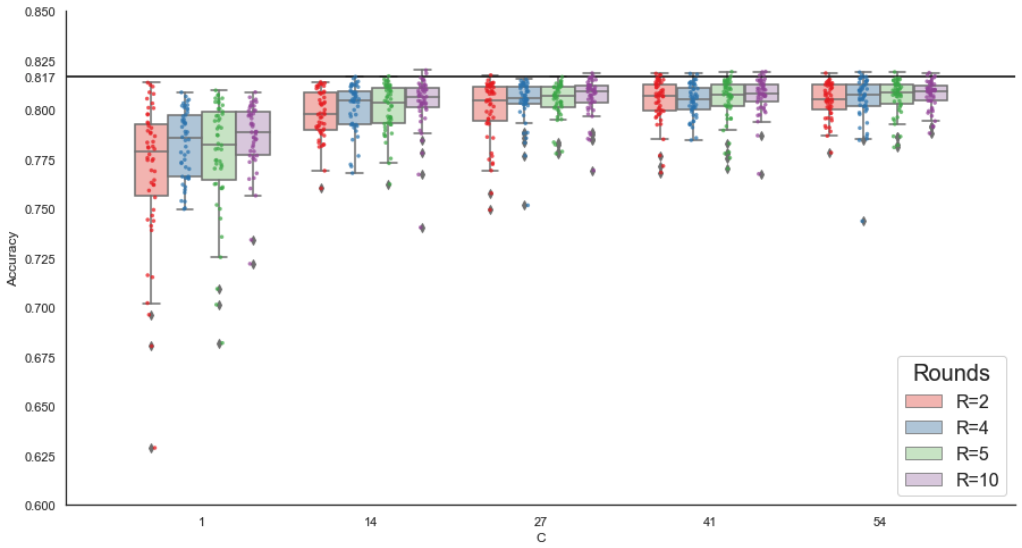

Figura 5.2: Resultados de precisión de la predicción de abandono (versión federada), promediando 50 ejecuciones aleatorias con diferentes cantidades de clientes por ronda (C), número de rondas (R) y épocas locales de clientes (E) donde $R \times E = 20$. La línea negra marca la precisión promediada por el modelo centralizado

.

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

El rendimiento en esta configuración federada es similar al del modelo centralizado, con una precisión

media superior al 76 % en todos los experimentos (que aumenta hasta el 80 % si se excluyen los experimentos con 1 cliente por ronda), una precisión 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 precisión superior al 80 %. Sin embargo, algunas ejecuciones siguen teniendo una precisión relativamente baja.

Nos preguntamos si es posible alcanzar *consistentemente* los resultados del entorno centralizado. Por lo tanto, repetimos los experimentos ejecutando tantas rondas como fueran necesarias para alcanzar el 81,7 % de precisión. Este método de evaluación está inspirado en [McM+17]. La figura 5.3 muestra nuestros resultados; podemos ver que es posible alcanzar la precisión del modelo centralizado en todos los casos, con la salvedad de que pueden ser necesarias muchas rondas. El número máximo de rondas es necesario cuando se entrena con un cliente por ronda, y la precisión 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 E reduce la R media necesaria para alcanzar nuestra precisión de referencia (81,7 %) en todos los casos. Sin embargo, no existe una relación inversa 1:1: por ejemplo, con 14 clientes, si E=2 se necesita una media de 16 rondas, pero si E=10, necesitamos 6 rondas, es decir, una relación de aumento x5 en E pero sólo una relación de disminución x2,6 en R.

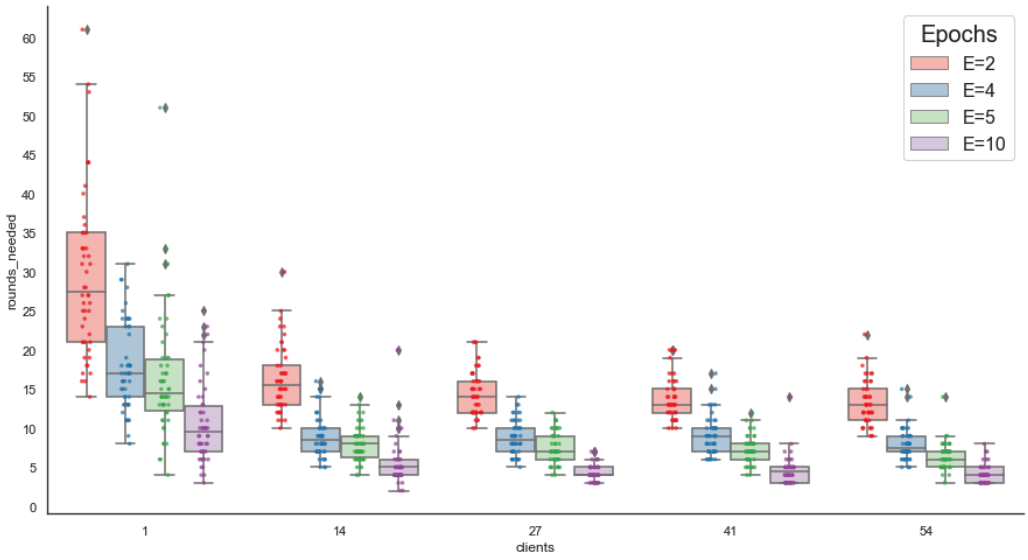

Figura 5.3: Número de rondas R necesarias para alcanzar la precisión centralizada de referencia (81,7 %), promediando 50 ejecuciones aleatorias con diferentes cantidades de clientes por ronda (C) y épocas locales en los clientes (E).

Otros experimentos: Distribución de datos homogénea frente a heterogénea

Los experimentos descritos en la sección tarea1:experimentos prueban la interacción entre diferentes parámetros y cómo afectan al rendimiento, dada una configuración experimental fija. En esta sección, seleccionaremos los parámetros pero variaremos las configuraciones. Básicamente, compararemos 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. También variamos 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.

La figura 5.4 muestra los resultados de 50 ejecuciones independientes utilizando la hipotesis de distribución homogénea de datos en la que los alumnos se distribuyen aleatoriamente entre los clientes. En esta figura,

cada punto representa la precisión media alcanzada en los datos de prueba retenidos presentes en todos los clientes. Para empezar con la discusión, si prestamos atención a la diagonal donde se sitúan los histogramas, podemos ver que bajo una federación los resultados tienen una varianza mayor que en los otros esquemas. A continuación, los resultados de las instituciones que se forman solas tienen mejores resultados de media que en una federación (centro-izquierda y arriba-centro). Sin embargo, tienden a tener peores resultados que en un esquema centralizado (abajo-izquierda y arriba-derecha). Por último, la federación obtiene peores resultados que el modelo centralizado en general (abajo-centro y centro-derecha).

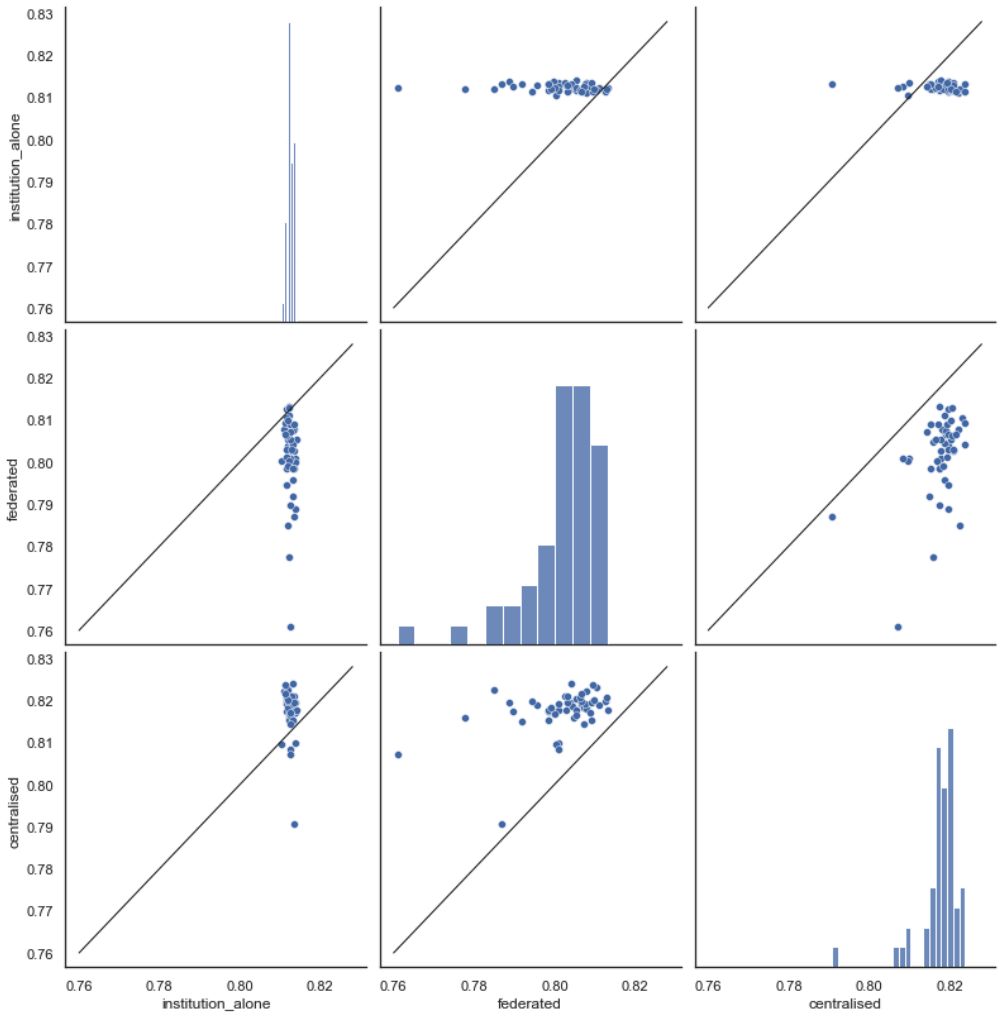

Figura 5.4: Comparación de la precisión media tras 50 ejecuciones independientes utilizando tres esquemas de formación: instituciones solas, federadas y centralizadas.

Dado que es interesante estudiar el efecto de la distribución de los datos en el rendimiento del modelo, aplicamos otros criterios para distribuir los datos entre los clientes. En este caso, estudiamos la distribución de la tasa de abandono teniendo en cuenta todo el conjunto de datos (véase la figura **??**) y la utilizamos para dividir la población de estudiantes entre las instituciones en función de la tasa de abandono de cada estudiante. La tabla 5.1 presenta las categorías definidas y el número de alumnos por categoría.

La figura 5.6 muestra la precisión 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 infrarrepresentados en el conjunto total de datos, lo que probablemente provoque un rendimiento deficiente en el caso

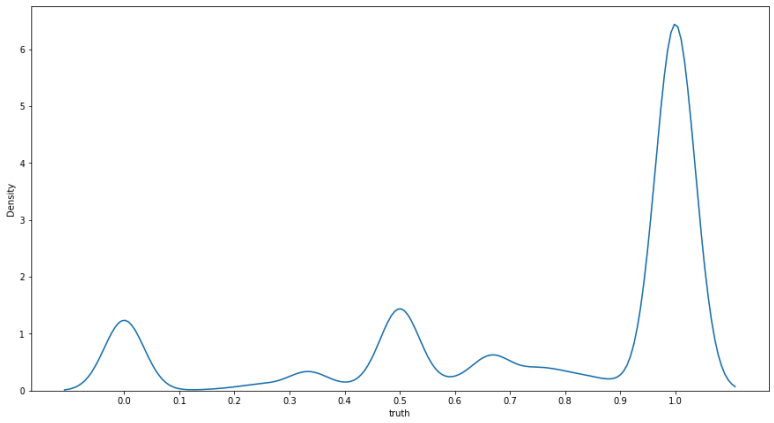

Figura 5.5: Distribución de la tasa de abandono entre todos los alumnos

Cuadro 5.1: 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 %

del entrenamiento federado. En consecuencia, obsérvese 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. Nuestra hipótesis 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.

Podemos concluir que si la institución pertenece a una çlase 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.

5.1.3. Tarea 2: clasificación no supervisada de alumnos

Nuestra segunda tarea corresponde a la clasificación no supervisada de alumnos. Esta tarea puede realizarse de forma centralizada utilizando el conocido algoritmo k-means. También existen versiones distribuidas de este algoritmo. La versión Federated Learning de k-means combina ideas del algoritmo distribuido con el concepto de utilizar sólo una fracción de los clientes en cada iteración [KKN20].

Es necesario comentar cómo evaluamos 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 5.1.2, donde se utilizó la métrica de precisión para comparar el rendimiento de los diferentes modelos. Los 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 podemos simplemente comparar las precisiones obtenidas con los algoritmos de clustering cen-

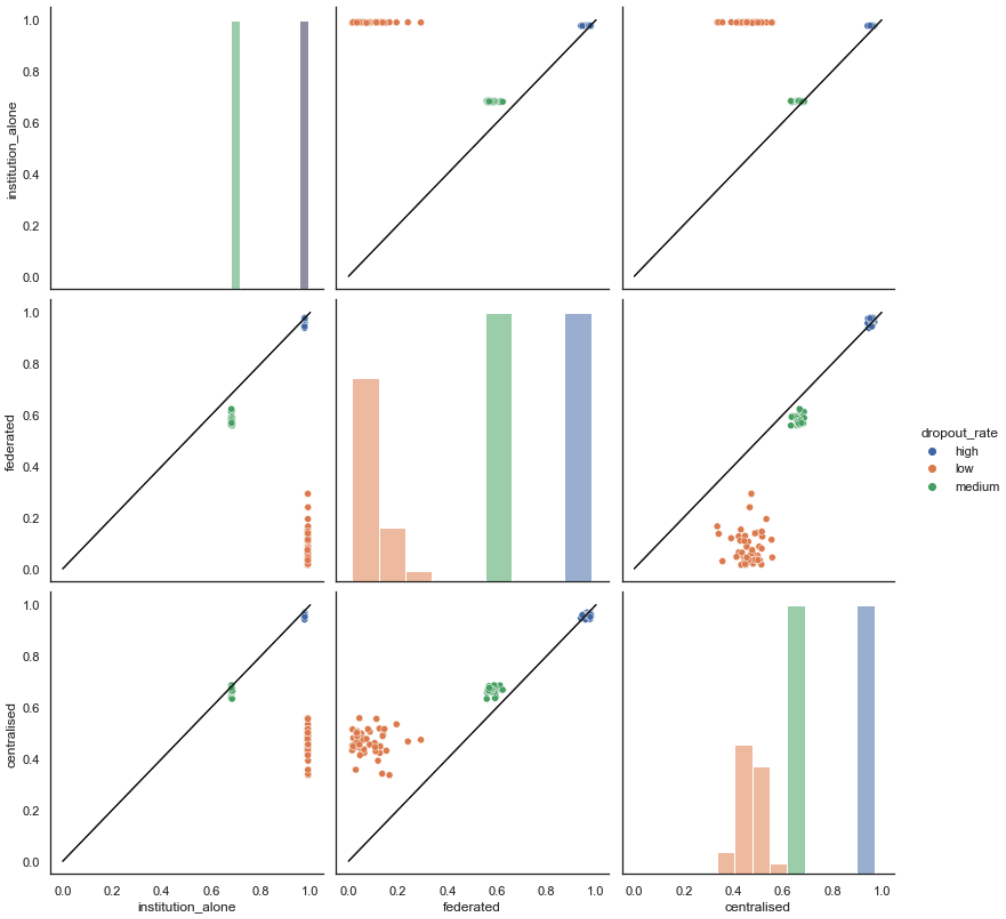

Figura 5.6: Comparación de la precisión media tras 50 ejecuciones independientes utilizando tres esquemas de distribución: instituciones solas, federadas y centralizadas. La tasa de abandono varía entre los clientes según las categorías definidas en la Tabla 5.1

tralizado y federado, compararemos 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. Suponemos que si son suficientemente similares, nos 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 utilizarse indistintamente, proporcionando la misma información que el centralizado 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. Partimos de los mismos centroides iniciales en cada caso. Luego, en la versión centralizada, ejecutamos el algoritmo hasta que todos los centroides se mueven menos que un valor específico ϵ. Para las versiones federadas, ejecutamos un número determinado de rondas r, y calculamos el índice de Jaccard utilizando los clusters centralizados finales y los federados en cada ronda. De este modo, obtenemos la evolución de la puntuación de Jaccard para saber si los clusters se están pareciendo a los clusters centralizados finales.

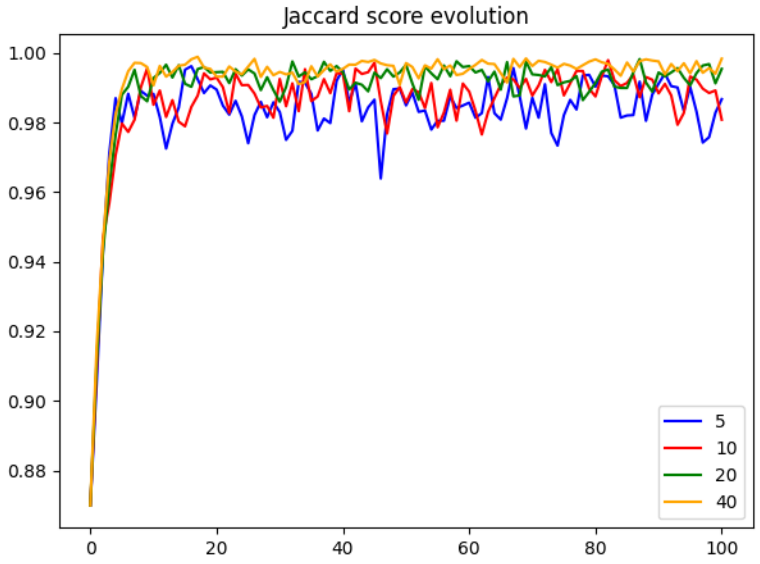

Figura 5.7: Evolución de la puntuación Jaccard para kmeans federado con 5, 10, 20 & 40 clientes disponibles por ronda y clusters kmeans centralizados para $\epsilon = 0,001$

El número de clusters k se fijará en 3 para el experimento. Para la versión centralizada, utilizamos una condición de parada de $\epsilon = 0,001$, y ejecutamos 4 versiones federadas con 5, 10, 20 & 40 clientes por ronda durante 100 rondas. La figura 5.7 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.

5.2. Discución y conclusiones

Los resultados muestran que aumentando el número de clientes y favoreciendo más rondas se obtiene una mayor precisión. Sin embargo, es crucial tener en cuenta que se trata de una simulación, y que no hemos

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 el experimento de la Figura 2 demostró que no es óptimo en términos de precisión. También merece la pena señalar que no hemos considerado esquemas que preserven la privacidad (Privacidad Diferencial y otros [Liu+21a]) en nuestros 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, nos hemos centrado en evaluar si un modelo puede arrojar resultados similares en los escenarios de formación federada y centralizada. Sin embargo, no hemos 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 nuestro experimento al utilizar el 25 % y el 50 % de todos los clientes). Concluimos que FL tiene el potencial de alcanzar los mismos resultados que el ML tradicional en entornos del mundo real, como lo hace en nuestros experimentos, pero es necesario realizar pruebas en un contexto no experimental para confirmarlo.

Capítulo 6

Conclusiones y trabajo a futuro

En este informe se presentaron los resultados del proyecto de investigación FLEA: Aprendizaje Federado para Learning Analytics. Se puede concluir que se cumplió con todos los objetivos del proyecto.

Se realizó un relevamiento de herramientas existentes para desarrollar soluciones de aprendizaje Federado. Específicamente, se decidió evaluar Tensor Flow Federated y Flower.

En ambos casos se logró implementar y experimentar casos de clasificación de imágenes con redes neuronales. Las pruebas comparando las soluciones federadas con la centralizada permiten afirmar que, para los casos evaluados, es posible conseguir un desempeño de la solución federada comparable a la centralizada.

Luego, se planteó trasladar lo aprendido en esta etapa preliminar a la implementación de casos específicos de Learning Analytics. Se seleccionaron dos tareas: la predicción de abandono y la clasificación distribuída de estudiantes. En ambos casos, se implementaron y testearon soluciones federadas. Se evaluó la influencia de diferentes parámetros como cantidad de clientes, distribución de los datos, tamaño de los batches y cantidad de épocas en los resultados obtenidos. Si bien queda por delante realizar evaluaciones más exhaustivas del enfoque, usando datos reales, y haciendo particular énfasis en las repercusiones que podría tener habilitar mecanismos como Differential Privacy, los resultados obtenidos son muy auspiciosos.

En todos los casos se arriba a conclusiones interesantes, que no sólo demuestran la factibilidad de este enfoque, si no que además permiten augurar su aplicación a nivel institucional e industrial en muchos escenarios. Todo el código generado está disponible en el repositorio del proyecto [FLE22]

Bibliografía

[Ale21] Alex Krizhevsky. *Sitio Web del proyecto CIFAR*. 2021. URL: https://www.cs.toronto.edu/~kriz/cifar.html (vid. pág. 8).

[Ass+18] Thibault Asselborn y col. «Automated human-level diagnosis of dysgraphia using a consumer tablet». En: *NPJ digital medicine* 1.1 (2018), págs. 1-9 (vid. págs. 17, 19).

[Ban+18] Seyyed Kazem Banihashem y col. «Learning Analytics: A Systematic Literature Review». En: *Interdisciplinary Journal of Virtual Learning in Medical Sciences* 9.2 (jun. de 2018), pág. 63024. ISSN: 2476-7263. DOI: 10.5812/IJVLMS.63024 (vid. pág. 3).

[Beu+20] Daniel J. Beutel y col. «Flower: A Friendly Federated Learning Research Framework». En: (2020). arXiv: 2007.14390. URL: http://arxiv.org/abs/2007.14390 (vid. pág. 9).

[Bov+21] Matthieu Bovery y col. «A Scalable Off-the-Shelf Framework for Measuring Patterns of Attention in Young Children and Its Application in Autism Spectrum Disorder». En: *IEEE Transactions on Affective Computing* 12.3 (2021), págs. 722-731. DOI: 10.1109/TAFFC.2018.2890610 (vid. pág. 19).

[BP21] Abeba Birhane y Vinay Uday Prabhu. «Large image datasets: A pyrrhic win for computer vision?» En: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2021, págs. 1536-1546 (vid. pág. 8).

[Cal+18] Sebastian Caldas y col. «LEAF: A Benchmark for Federated Settings». En: *CoRR* abs/1812.01097 (2018). arXiv: 1812.01097. URL: http://arxiv.org/abs/1812.01097 (vid. pág. 8).

[Cei07] Ceibal. *Website del Plan Ceibal*. 2007. URL: https://www.ceibal.edu.uy (vid. págs. 4, 16).

[Coh+17] Gregory Cohen y col. «EMNIST: an extension of MNIST to handwritten letters». En: *CoRR* abs/1702.05373 (2017). arXiv: 1702.05373. URL: http://arxiv.org/abs/1702.05373 (vid. pág. 8).

[DD20] Peter Drotár y Marek Dobeš. «Dysgraphia detection through machine learning». En: *Scientific reports* 10.1 (2020), págs. 1-11 (vid. págs. 17, 19).

[Dra+16] Hendrik Drachsler y col. «Ethical and privacy issues in the design of learning analytics applications». En: *ACM International Conference Proceeding Series* 25-29-Apri (abr. de 2016), págs. 492-493. DOI: 10.1145/2883851.2883933. URL: http://dx.doi.org/10.1145/2883851.2883933 (vid. pág. 3).

[Dwo08] Cynthia Dwork. «Differential privacy: A survey of results». En: *International conference on theory and applications of models of computation*. Springer. 2008, págs. 1-19 (vid. pág. 3).

[Epa+21] Alessandro Epasto y col. «Clustering for Private Interest-Based Advertising». En: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, págs. 2802-2810. ISBN: 9781450383325. DOI: 10.1145/3447548.3467180. URL: https://doi.org/10.1145/3447548.3467180 (vid. pág. 16).

[FLE22] FLEA. *FLEA project public repository*. https://gitlab.fing.edu.uy/flea/flea. 2022. URL: https://gitlab.fing.edu.uy/lorenae/flea (vid. págs. 23, 31).

[Flo22a] Flower. *Flower A Friendly Federated Learning Framework*. https://flower.dev/. 2022 (vid. pág. 9).

[Flo22b] Flower. *Repositorio del proyecto Flower*. https://github.com/adap/flower. 2022 (vid. pág. 9).

[FTL19] Wenzheng Feng, Jie Tang y Tracy Xiao Liu. «Understanding dropouts in MOOCs». En: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, págs. 517-524 (vid. pág. 23).

[Goo21a] Google. *TensorFlow Federated: GitHub*. 2021. URL: https://github.com/tensorflow/federated (vid. pág. 9).

[Goo21b] Google Brain Team. *TensorFlow Federated: API Documentation*. 2021. URL: https://www.tensorflow.org/federated/api_docs/python/tff (vid. pág. 9).

[Goo21c] Google Brain Team. *TensorFlow Federated: Datasets*. 2021. URL: https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets (vid. pág. 9).

[Goo21d] Google Brain Team. *TensorFlow Federated: Federated Averaging*. 2021. URL: https://www.tensorflow.org/federated/api_docs/python/tff/learning/ClientFedAvg (vid. pág. 9).

[Goo21e] Google Brain Team. *TensorFlow Federated: Federated Core*. 2021. URL: https://www.tensorflow.org/federated/federated_core (vid. pág. 9).

[Goo21f] Google Brain Team. *TensorFlow Federated: Federated Learning*. 2021. URL: https://www.tensorflow.org/federated/federated_learning (vid. pág. 9).

[Goo21g] Google Brain Team. *TensorFlow Federated: GitHub Issues*. 2021. URL: https://github.com/tensorflow/federated/issues (vid. pág. 9).

[Goo21h] Google Brain Team. *TensorFlow Federated: guides*. 2021. URL: https://www.tensorflow.org/federated/get_started (vid. pág. 9).

[Goo21i] Google Brain Team. *TensorFlow Federated: Stack Overflow*. 2021. URL: https://stackoverflow.com/questions/tagged/tensorflow-federated (vid. pág. 9).

[Goo21j] Google Brain Team. *TensorFlow Federated: Tutorials*. 2021. URL: https://www.tensorflow.org/federated/tutorials/tutorials_overview (vid. pág. 9).

[Goo21k] Google Brain Team. *TensorFlow Federated: Web*. 2021. URL: https://www.tensorflow.org/federated (vid. pág. 9).

[Goo21l] Google Brain Team. *TensorFlow: Forum*. 2021. URL: https://discuss.tensorflow.org/ (vid. pág. 9).

[Goo21m] Google Brain Team. *TensorFlow: Web*. 2021. URL: https://www.tensorflow.org (vid. pág. 9).

[Gur+17] Mehmet Emre Gursoy y col. «Privacy-Preserving Learning Analytics: Challenges and Techniques». En: *IEEE Transactions on Learning Technologies* 10 (1 ene. de 2017), págs. 68-81. ISSN: 19391382. DOI: 10.1109/TLT.2016.2607747 (vid. pág. 3).

[GZ20] Song Guo y Deze Zeng. «Pedagogical Data Federation toward Education 4.0». En: *Proceedings of the 6th International Conference on Frontiers of Educational Technologies*. ICFET '20. Tokyo, Japan: Association for Computing Machinery, 2020, págs. 51-55. ISBN: 9781450375337. DOI: 10.1145/3404709.3404751. URL: https://doi.org/10.1145/3404709.3404751.

[Hak+20] Saqib Hakak y col. «A framework for edge-assisted healthcare data analytics using federated learning». En: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, págs. 3423-3427.

[Kag16] Kaggle. *Shakespeare plays*. 2016. URL: https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays (vid. pág. 8).

[Kai+21] Peter Kairouz y col. «Practical and Private (Deep) Learning without Sampling or Shuffling». En: (2021), págs. 1-34. arXiv: 2103.00039. URL: http://arxiv.org/abs/2103.00039 (vid. pág. 7).

[Kar+19] Ruchira Kariyawasam y col. «Pubudu: Deep learning based screening and intervention of dyslexia, dysgraphia and dyscalculia». En: *2019 14th Conference on Industrial and Information Systems (ICIIS)*. IEEE. 2019, págs. 476-481 (vid. págs. 17, 19).

[KB14] Diederik P Kingma y Jimmy Ba. «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980* (2014) (vid. pág. 23).

[KDD15] KDD. *KDDCup*. http://moocdata.cn/challenges/kdd-cup-2015. 2015. URL: http://moocdata.cn/challenges/kdd-cup-2015 (vid. pág. 23).

[KE16] Mohammad Khalil y Martin Ebner. «De-Identification in Learning Analytics». En: *Journal of Learning Analytics* 3 (1 abr. de 2016), págs. 129-138. ISSN: 1929-7750. DOI: 10.18608/jla.2016.31.8. URL: https://learning-analytics.info/index.php/JLA/article/view/4519 (vid. pág. 3).

[KH+09] Alex Krizhevsky, Geoffrey Hinton y col. «Learning multiple layers of features from tiny images». En: (2009) (vid. pág. 8).

[Kho+21] Ivan Kholod y col. «Parallelization of the self-organized maps algorithm for federated learning on distributed sources». En: *The Journal of Supercomputing* 77 (2021), págs. 6197-6213. ISSN: 1573-0484. DOI: 10.1007/s11227-020-03509-2. URL: http://doi.org/10.1007/s11227-020-03509-2 (vid. pág. 17).

[KKN20] Hemant H Kumar, VR Karthik y Mydhili K Nair. «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)*. IEEE. 2020, págs. 52-56 (vid. pág. 27).

[Kon+16] Jakub Konečný y col. *Federated Optimization: Distributed Machine Learning for On-Device Intelligence*. 2016. URL: https://arxiv.org/abs/1610.02527 (vid. pág. 3).

[Kut+21] Reshma Narayanan Kutty y col. *Personalization, Privacy, and Me*. 2021. arXiv: 2109.06990 [cs.CY] (vid. pág. 16).

[Kyr+19] Kyriaki H. Kyritsi y col. «The pursuit of patterns in educational data mining as a threat to student privacy». En: *Journal of Interactive Media in Education* 2019 (1 mayo de 2019), págs. 1-10. ISSN: 1365893X. DOI: 10.5334/JIME.502/METRICS/. URL: http://jime.open.ac.uk/articles/10.5334/jime.502/ (vid. pág. 3).

[Lan21] Marc Langheinrich. «To FLoC or Not?» En: *IEEE Pervasive Computing* 20.2 (2021), págs. 4-6. DOI: 10.1109/MPRV.2021.3076812 (vid. pág. 16).

[LEA21] LEAF. *Website del proyecto LEAF*. 2021. URL: https://leaf.cmu.edu/ (vid. pág. 9).

[LEA22] LEAF. *Repositorio del proyecto LEAF*. 2022. URL: https://github.com/TalwalkarLab/leaf (vid. pág. 9).

[Li+20] Tian Li y col. «Federated Learning: Challenges, Methods, and Future Directions». En: *IEEE Signal Processing Magazine* 37.3 (2020), págs. 50-60. DOI: 10.1109/MSP.2020.2975749 (vid. pág. 4).

[Liu+21a] Bo Liu y col. «When Machine Learning Meets Privacy: A Survey and Outlook». En: *ACM Computing Surveys* 54.2 (2021). ISSN: 15577341. DOI: 10.1145/3436755. arXiv: 2011.11819 (vid. pág. 30).

[Liu+21b] Yi Liu y col. «Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach». En: *IEEE Internet of Things Journal* 8.8 (2021), págs. 6348-6358. DOI: 10.1109/JIOT.2020.3011726 (vid. pág. 17).

[McM+17] Brendan McMahan y col. «Communication-efficient learning of deep networks from decentralized data». En: *Artificial intelligence and statistics*. PMLR. 2017, págs. 1273-1282 (vid. págs. 6, 7, 23, 25).

[MG19] Katerina Mangaroska y Michail Giannakos. «Learning Analytics for Learning Design: A Systematic Literature Review of Analytics-Driven Design to Enhance Learning». En: *IEEE Transactions on Learning Technologies* 12.4 (oct. de 2019), págs. 516-534. ISSN: 19391382. DOI: 10.1109/TLT.2018.2868673 (vid. pág. 3).

[MIT21] MIT. *Keras: Web*. 2021. URL: https://keras.io/ (vid. pág. 9).

[Ngu+22] Dinh C Nguyen y col. «Federated learning for smart healthcare: A survey». En: *ACM Computing Surveys (CSUR)* 55.3 (2022), págs. 1-37.

[RD16] Sara Rosenblum y Gideon Dror. «Identifying developmental dysgraphia characteristics utilizing handwriting classification methods». En: *IEEE Transactions on Human-Machine Systems* 47.2 (2016), págs. 293-298 (vid. págs. 17, 19).

[Rie+20] Nicola Rieke y col. «The future of digital health with federated learning». En: *NPJ digital medicine* 3.1 (2020), págs. 1-7.

[Rod+20] Nuria Rodríguez-Barroso y col. «Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy». En: *Information Fusion* 64 (dic. de 2020), págs. 270-292. ISSN: 15662535. DOI: 10.1016/j.inffus.2020.07.009. arXiv: 2007.00914 (vid. pág. 17).

[San+18] Mark Sandler y col. «Mobilenetv2: Inverted residuals and linear bottlenecks». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, págs. 4510-4520 (vid. pág. 12).

[SHD19] Guillermo Sapiro, Jordan Hashemi y Geraldine Dawson. «Computer vision and behavioral phenotyping: an autism case study». En: *Current Opinion in Biomedical Engineering* 9 (2019). Futures of BME: Digital Health and BME Biomedical Imaging: Cardiovascular Imaging, págs. 14-20. ISSN: 2468-4511. DOI: https://doi.org/10.1016/j.cobme.2018.12.002. URL: https://www.sciencedirect.com/science/article/pii/S246845111830059X (vid. pág. 19).

[Sin+21] Shubham Singh y col. «Anomaly Detection Using Federated Learning». En: *Proceedings of International Conference on Artificial Intelligence and Applications*. Ed. por Poonam Bansal y col. Singapore: Springer Singapore, 2021, págs. 141-148. ISBN: 978-981-15-4992-2 (vid. pág. 17).

[Sta18] Stack Overflow. *Stack Overflow Data (BigQuery Dataset)*. 2018. URL: https://www.kaggle.com/datasets/stackoverflow/stackoverflow (vid. pág. 8).

[Swe02] Latanya Sweeney. «k-anonymity: A model for protecting privacy». En: *International journal of uncertainty, fuzziness and knowledge-based systems* 10.05 (2002), págs. 557-570 (vid. pág. 3).

[Uni21] United States Goverment. *Sitio Web del proyecto EMNIST*. 2021. URL: https://www.nist.gov/itl/products-and-services/emnist-dataset (vid. pág. 8).

[WC20] Shuai Wang y Tsung-Hui Chang. *Federated Matrix Factorization: Algorithm Design and Application to Data Clustering*. 2020. arXiv: 2002.04930 [cs.LG] (vid. pág. 17).

[Wey+20] Tobias Weyand y col. «Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval». En: *CoRR* abs/2004.01804 (2020). arXiv: 2004.01804. URL: https://arxiv.org/abs/2004.01804 (vid. pág. 8).

[Yan+19] Qiang Yang y col. «Federated Machine Learning: Concept and Applications». En: *ACM Trans. Intell. Syst. Technol.* 10.2 (ene. de 2019). ISSN: 2157-6904. DOI: 10.1145/3298981. URL: https://doi.org/10.1145/3298981.

[Yan21] Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond. *MNIST: Web*. 2021. URL: http://yann.lecun.com/exdb/mnist/ (vid. págs. 8, 11).

[Zam20] Faisal Zaman. *Instilling Responsible and Reliable AI Development with Federated Learning*. https://medium.com/the-dock/instilling-responsible-and-reliable-ai-development-with-federated-learning-d23c366c5efd. 2020 (vid. pág. 5).

[Zha+19] Ying Zhao y col. «Multi-Task Network Anomaly Detection Using Federated Learning». En: *Proceedings of the Tenth International Symposium on Information and Communication Technology*. SoICT 2019. Hanoi, Ha Long Bay, Viet Nam: Association for Computing Machinery, 2019, págs. 273-279. ISBN: 9781450372459. DOI: 10.1145/3368926.3369705. URL: https://doi.org/10.1145/3368926.3369705 (vid. pág. 17).

[Zha+21a] Chen Zhang y col. «A survey on federated learning». En: *Knowledge-Based Systems* 216 (2021), pág. 106775 (vid. pág. 7).

[Zha+21b] Zhengming Zhang y col. *Improving Semi-supervised Federated Learning by Reducing the Gradient Diversity of Models*. 2021. arXiv: 2008.11364 [cs.LG] (vid. pág. 17).