Escuela Técnica Superior de Ingeniería Informática

TRABAJO FIN DE GRADO

Implementación de una arquitectura de aprendizaje federado aplicada a una base de datos médica

---

Autor:

RAFAEL CASTILLO CEBOLLA

Tutores:

BETSAIDA ALEXANDRE BARAJAS

JUAN ANTONIO ORTEGA RAMÍREZ

MARÍA DEL CARMEN ROMERO TERNERO

Grado en Ingeniería Informática - Ingeniería del Software

Curso académico 2024-2025

Dedicado, por encima de todos, a mi familia, a quien siempre quiero y que siempre me quiere.

Dedicado, también, a Dios, por darme ánimo cuando la necesitaba, y a todas las personas buenas, que me inspiran a ser mejor.

Dedicado, además, a los profesores buenos que me he encontrado en la carrera; esos pocos que entendían que no se trataba de aprobar o suspender, o de echar horas, sino de enseñar y aprender.

TABLA DE CONTENIDOS

RESUMEN

Resumen

Este Trabajo de Fin de Grado presenta un desarrollo experimental centrado en la implementación de un prototipo de plataforma colaborativa basada en aprendizaje federado, una técnica innovadora de entrenamiento de modelos de aprendizaje de inteligencia artificial en la que los datos permanecen distribuidos en distintos dispositivos. Para ello, se ha desarrollado un sistema que simula un entorno federado compuesto por varios clientes y un servidor central orquestador, utilizando una base de datos médica como caso de estudio.

Como referencia comparativa, también se ha desarrollado una implementación equivalente basada en aprendizaje centralizado clásico, con los datos concentrados en un único punto para el entrenamiento. Se ha evaluado y comparado el rendimiento de ambos enfoques ante distintos escenarios de distribución de datos.

El trabajo abarca el diseño, desarrollo y validación del sistema de aprendizaje federado y centralizado, junto con un análisis experimental de los resultados. Las conclusiones permiten valorar la potencialidad de la viabilidad del aprendizaje federado en contextos sensibles como el médico, donde la privacidad es prioritaria. Esta memoria incluye también la planificación y el seguimiento realizados durante el desarrollo.

Palabras clave

Software, aprendizaje federado, inteligencia artificial, datos, aprendizaje automático, aprendizaje profundo, sanidad

Abstract

This Final Degree Project presents an experimental development focused on the study and implementation of a prototype of a collaborative platform based on federated learning, an innovative technique for collaborative training of artificial intelligence models while keeping data distributed across different devices or nodes. To this end, a system has been developed that simulates a federated environment composed of multiple clients and a central orchestrating server, using a medical dataset as a case study.

For comparison purposes, an equivalent version based on traditional centralized learning has also been implemented, where all data is aggregated in a single location for model training. The performance of both approaches has been evaluated and compared under different data distribution scenarios.

The work includes the design, development, and validation of both systems, as well as an experimental analysis of the results. The conclusions assess the viability of federated learning in sensitive contexts, such as the medical field, where data privacy is a key concern. This report also documents the planning and monitoring carried out throughout the development.

Keywords

Software, federated learning, artificial intelligence, data, machine learning, deep learning, health-care

1 – INTRODUCCIÓN

I. Introducción

En los últimos años, el avance de la inteligencia artificial (IA) ha abierto nuevas oportunidades en diferentes ámbitos, entre ellos, el de la salud, donde su aplicación apunta a mejorar los diagnósticos y tratamientos clínicos. Sin embargo, este potencial choca con limitaciones importantes relacionadas con la privacidad y seguridad de los datos.

El aprendizaje tradicional de modelos de IA ha seguido una estructura centralizada, es decir, que los datos de entrenamiento deben ser compartidos y recopilados en un silo centralizado. Esto presenta tres inconvenientes principales: un alto coste y una baja velocidad, debido a la gran cantidad de datos en transferencia, y un alto riesgo de seguridad, debido a la exposición de datos.

En este contexto surge el aprendizaje federado (AF; *federated learning* en inglés, FL), una técnica innovadora que permite entrenar los modelos de IA manteniendo los datos en su ubicación original, transmitiendo únicamente los modelos y no los datos.

Este Trabajo de Fin de Grado (TFG) surge en el contexto del proyecto de investigación ARTIFACTS[1], centrado precisamente en la exploración y aplicación del AF en el ámbito sanitario y la generación de datos sintéticos en espacios de datos seguros.

En este trabajo, se ha desarrollado un prototipo funcional de una plataforma de AF, el cual se ha implementado utilizado el framework *Flower*, simulando un entorno distribuido compuesto por varios clientes y un servidor central encargado de coordinar el proceso de entrenamiento. Como caso de estudio, se ha utilizado una base de datos médica[2]. Además, se ha implementado también un sistema equivalente basado en aprendizaje centralizado, con el fin de comparar ambos enfoques. Aunque los resultados no constituyen aún una validación formal, sí que ofrecen indicios útiles sobre su viabilidad y efectividad.

[1] This work was partially funded by Grant PID2022-141045OB- {C41,C42,C43} funded by MCIN/AEI /10.13039/ 501100011033/ and FEDER A way of making Europe in ARTIFACTS Project: generAtion of Reliable syntheTIc health data for Federated leArning in seCure daTa Spaces
[2] PI-CAI Data Splits, Datasets: Imaging, Labels. https://pi-cai.grand-challenge.org/DATA/

El código del sistema desarrollado se puede encontrar en el siguiente repositorio de GitHub: https://github.com/rafcasceb/Federated-learning

También se dispone de una guía completa del sistema desarrollado generada con DeepWiki. Se puede acceder a esta guía *online* a través del siguiente enlace: https://deepwiki.com/rafcasceb/Federated-learning

La memoria que a continuación se presenta se estructura en diez capítulos. En el primero, el actual, se introduce el trabajo y su memoria. El segundo capítulo describe el AF, analiza el estado actual de su investigación y uso, y, por último, se plantean los objetivos específicos del trabajo. En el tercer capítulo se detalla la metodología seguida para abordar la implementación. El cuarto capítulo recoge los requisitos del sistema. En el quinto capítulo se presenta la planificación del trabajo. El sexto capítulo recoge las principales decisiones de diseño tomadas durante el desarrollo. El séptimo capítulo constituye el núcleo de esta memoria, describiendo en detalle la implementación del sistema de AF y aprendizaje centralizado y la selección e integración del *framework* Flower. El octavo capítulo compara el rendimiento de las implementaciones federada y centralizada mediante algunas pruebas realizadas. El noveno capítulo presenta las conclusiones extraídas del trabajo. Finalmente, el décimo capítulo recoge las referencias bibliográficas utilizadas.

II. Glosario de abreviaturas

Por comodidad y legibilidad, se ha optado por emplear abreviaturas de los términos técnicos más recurrentes. Aunque en la primera aparición de cada término se declare su abreviatura, se incluye a continuación un glosario de abreviaturas para facilitar la búsqueda en caso de duda. Sigue un orden alfabético.

- **IA**: inteligencia artificial
- **AF**: aprendizaje federado
- **FL**: *federated learning*
- **TFG:** Trabajo de Fin de Grado

- **ML**: *machine learning*
- **IID:** *independent and identically distributed*

2 – JUSTIFICACIÓN Y OBJETIVOS

Esta sección empieza presentando el aprendizaje federado, haciendo indicación de sus orígenes, de sus aspectos técnicos y de una breve revisión de la situación actual de la investigación científica, y sus más importantes tendencias y aplicaciones prácticas.

El capítulo continúa presentando los objetivos de este TFG, con lo que se podrá justificar su existencia, destacando su relevancia y utilidad actual, no solo en aplicación, sino en conocimiento social también.

I. Aprendizaje federado y situación actual

Machine learning (ML) es un campo destacado dentro de la inteligencia artificial, el cual permite entrenar máquinas para aprender y predecir situaciones, simulando el comportamiento humano [1]. Para llevar a cabo eficientemente modelos de ML, las máquinas requieren una gran cantidad de datos para procesar y con los que entrenarse.

Tradicionalmente, el desarrollo de modelos de ML se ha llevado a cabo de manera centralizada, lo que significa que los datos de entrenamiento deben ser compartidos y recopilados en un silo centralizado. Esto presenta tres inconvenientes principales: un alto coste y una baja velocidad, debido a la gran cantidad de datos en transferencia, y un alto riesgo de seguridad, debido a la exposición de datos potencialmente sensibles en el proceso de transferencia [2] [3].

El aprendizaje federado (AF) es una técnica innovadora de ML que surge con el propósito de solventar dichos problemas. El AF permite desarrollar modelos de ML en un escenario distribuido y descentralizado. En vez de recoger en una localización centralizado todos los datos para el entrenamiento de un modelo de ML, se entrenan diferentes modelos por separado en cada uno de los silos descentralizados. Luego, solo se comparten los parámetros actualizados, que el servidor central se encarga de agregar estos modelos obteniendo uno solo unificado [4] [5].

Así pues, el AF permite evitar en buena medida los riesgos de seguridad al no compartir directamente los datos sensibles, y proporciona una mayor rapidez y un menor coste, al no tener que transmitir enormes cantidades de datos. Además, cuenta con una gran capacidad de escalamiento [6] [7]. La Figura 1: Ventajas del aprendizaje federado (elaboración propia) muestra un resumen de las ventajas del AF.

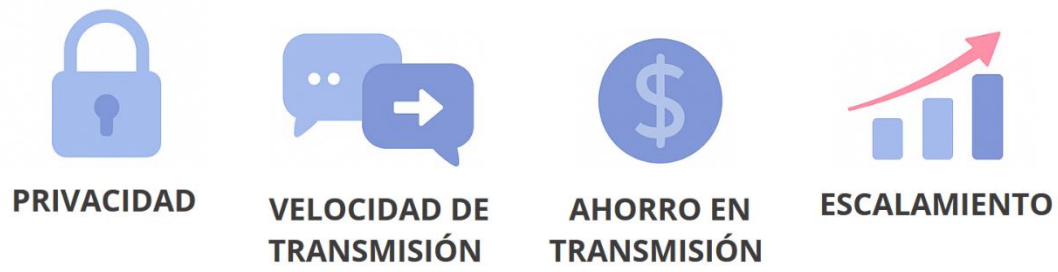


Figura 1: Ventajas del aprendizaje federado (elaboración propia)

Aunque el concepto de aprendizaje distribuido ya existía, fue en 2017 cuando miembros de la compañía Google propusieron formalmente el término "aprendizaje federado". Su objetivo inicial fue mejorar modelos predictivos en dispositivos Android sin comprometer el gran volumen de datos privados de los usuarios. De este modo propusieron un enfoque en el que "el modelo viaja a los datos" y no al revés [8].

A partir de esta propuesta inicial, el AF ha evolucionado hasta convertirse en un área activa y en boga de investigación en IA, seguridad informática y sistemas distribuidos, desarrollando un gran interés de la comunidad. Multitud de estudios y experimentos han demostrado las bondades de esta técnica y han dejado patente claros indicios de la efectividad del AF frente al aprendizaje centralizado [9].

Durante estos primeros años, han surgido nuevas líneas de desarrollo que han contribuido a sofisticar el paradigma. Entre ellas destacan las mejoras en los algoritmos de agregación, el desarrollo de técnicas para gestionar diferentes distribuciones de datos y mecanismos de comunicación más eficientes y seguros [10].

El AF ha comenzado a implementarse en múltiples dominios reales, especialmente en aquellos donde la privacidad de los datos y la distribución natural de los mismos suponen una barrera para los enfoques tradicionales. Destacan, por ejemplo, los entornos médicos y hospitalarios, donde los historiales clínicos no pueden compartirse libremente entre

centros; el sector financiero, y el ámbito de los dispositivos móviles, en el que millones de usuarios generan datos localmente [11] [12] [13] [14].
Este progreso técnico ha sedimentado el estatus del AF, no como una posible solución viable sino ya como una pieza clave de la actualidad de la IA.

II. Aspectos técnicos del aprendizaje federado

Un sistema de AF sigue un flujo de trabajo estructurado que permite coordinar la comunicación entre los distintos nodos participantes, ya sean clientes o dispositivos, y en muchos casos, un servidor central también. Los pasos de un ciclo estándar serían los siguientes [15] (ver también Figura 2 de la página siguiente):

1. Establecimiento de conexión: los nodos participantes establecen su conexión y se sincronizan para comenzar el ciclo de entrenamiento.
2. Entrenamiento local: cada cliente entrena de forma independiente un modelo con sus propios datos, conservando la privacidad de la información.
3. Agregación de modelos locales y actualización del modelo global: los modelos entrenados localmente son compartidos y agregados para formar un nuevo modelo global.
4. Actualización local con modelo agregado: el modelo global actualizado es redistribuido a los clientes, que actualizan sus modelos locales con la nueva versión y se preparan para una nueva ronda de entrenamiento.

Figura 2: Ciclo estándar de aprendizaje federado (elaboración propia)

Existen diferentes arquitecturas posibles para un sistema de AF en función de los roles de cada parte en su relación. Los dos más destacables son los siguientes [6]:

- Arquitectura servidor y clientes: es la más habitual en implementaciones reales. Un servidor central actúa como orquestador del proceso, encargándose de coordinar la comunicación, recopilar los modelos locales entrenados, realizar la agregación y redistribuir el modelo global a los clientes. Permite un control directo y estable del proceso de entrenamiento.
- Arquitectura *peer-to-peer*: en este modelo no existe un nodo central. Los dispositivos participantes se comunican directamente entre ellos, compartiendo modelos o actualizaciones de forma cooperativa. Aunque presenta desafíos técnicos adicionales para gestionar la sincronización y la agregación descentralizada, esta arquitectura elimina el posible cuello de botella que presenta el servidor.

Una propiedad clave a considerar en los sistemas de AF es si los datos locales de cada cliente mantienen una distribución similar entre sí, o no. Esta diferencia da lugar a dos

escenarios en función de si los datos son o no *Independent and Identically Distributed* (IID):

- Distribución IID: todos los clientes disponen de datos que comparten la misma distribución estadística. Es un escenario ideal puesto que facilita la convergencia del modelo global, pero en aplicaciones real, es poco habitual.
- Distribución non-IID: los datos de cada cliente provienen de distribuciones distintas, lo que es mucho más común en el mundo real. Esta heterogeneidad genera importantes dificultades técnicas, ya que puede provocar grandes diferencias entre los modelos locales y dificultades en la agregación del modelo global [16] [17].

También se puede clasificar el AF dependiendo de cómo estén repartidos los datos entre los participantes, en función de sus individuos u observaciones estudiadas, sus características de estudio para cada uno y los objetivos a predecir. Se encuentran tres clasificaciones principales [18]. Se considerará también un ejemplo con múltiples hospitales que estudian ciertos síntomas (por ejemplo, edad o presión arterial) de sus pacientes para detectar una cierta enfermedad [19].

- Aprendizaje federado horizontal (HFL): también conocido como *sampled-partitioned FL*, ocurre cuando los nodos tienen distintos individuos, pero estudian las mismas características e intentan predecir el mismo objetivo.

  En el ejemplo citado, cada hospital tendría pacientes diferentes, pero estudiaría los mismos síntomas para detectar la misma enfermedad.

- Aprendizaje federado vertical (VFL): también conocido como *feature-partitioned FL*, ocurre cuando los nodos comparten los mismos individuos y el mismo objetivo a predecir, pero cada uno estudia características diferentes.

  En el ejemplo mencionado, cada hospital tendría los mismos pacientes, pero estudiaría diferentes síntomas para detectar la misma enfermedad.

- *Transfer federated learning* (TFL): se aplica cuando los nodos no comparten ni los individuos ni las características de estudio ni los objetivos a predecir, pero,

aun así, buscan una transferencia de conocimiento que pueda resultares de utilidad.

En el caso de ejemplo, cada hospital tendría pacientes diferentes, estudiaría síntomas diferentes y buscaría detectar enfermedades diferentes.

En los últimos años, se han desarrollado diversas plataformas que facilitan el desarrollo y la experimentación con sistemas de AF, permitiendo a investigadores y profesionales aplicar estas técnicas de forma accesible y estandarizada y fomentando así la consolidación del AF.

Estas plataformas incluyen no solo las herramientas básicas necesarias para implementar el AF, como la comunicación entre clientes y servidor o la agregación de modelos, sino también componentes más avanzados como estrategias predefinidas de federación, simuladores de múltiples clientes, y compatibilidad con librerías de deep learning como PyTorch o TensorFlow. Se les considera *frameworks* porque encapsulan gran parte de la lógica común del sistema, permitiendo al desarrollador centrarse solo en definir las partes específicas de su caso de uso (como el modelo o los datos), sin tener que implementar toda la infraestructura desde cero.

Entre las más destacadas se encuentran TensorFlow Federated, PySyft, Flower y NVIDIA FLARE [20] [21].

III. Objetivos y justificación

La salud de las personas es siempre una inquietud central de la investigación científica con alto impacto social. Los hospitales y demás entidades sanitarias generan y almacenan un volumen inmenso de datos médicos, desde historiales de pacientes hasta imágenes clínicas de muy diversas pruebas médicas. Sin embargo, estos datos suelen estar separados entre múltiples instituciones y protegidos por rigurosas normativas de privacidad, como el RGPD en Europa.

Estas condiciones que presenta el ámbito de la salud crean un contexto donde la implementación del AF resulta muy conveniente, permitiendo el desarrollo de modelos colaborativos sin necesidad de centralizar los datos ni comprometer la confidencialidad de los pacientes. El análisis sistemático de estos datos mediante técnicas de IA tiene el potencial de transformar la calidad de los diagnósticos médicos.

Numerosos estudios e implementaciones prácticas han mostrado un alto potencial del AF en el ámbito sanitario, mostrando una precisión sólida y margen de mejora. Caben destacar estudios en el área de análisis de imágenes médicas, contando con estudios prometedores como la aplicación de AF a radiografías torácicas para detectar casos de COVID-19 o a resonancias magnéticas para segmentación de tumores [22].

A pesar de estos avances, la adopción a gran escala del AF en medicina aún está lejos de completarse. Por un lado, la precisión de los sistemas, las herramientas y las técnicas aún no alcanza los niveles deseables para un ámbito tan crítico como el sanitario. Además, se requiere de una inversión significativa en infraestructuras tecnológicas y una cuidada coordinación entre instituciones para construir ecosistemas federados funcionales, bien engranados y seguros [23] [24] [25] [26].

En este contexto científico, surge en España el proyecto ARTIFACTS (acrónimo de *generAtion of Reliable syntheTIc health data for Federated leArning in seCure daTa Spaces*). Se trata de un proyecto de investigación financiado por el Ministerio (2023-2026) y coordinado a nivel nacional con tres universidades cuyo objetivo es impulsar el desarrollo de tecnologías avanzadas de AF aplicadas al ámbito de la salud, con un enfoque especial en la privacidad y la seguridad de los datos clínicos y en el uso de datos sintéticos [27] [28].

ARTIFACTS está estructurado en varios subproyectos coordinados entre tres universidades españolas: la Universidad de Sevilla (USE), la Universidad Politécnica de Cataluña (UPC) y la Universidad Carlos III de Madrid (UC3M); y por varias instituciones hospitalarias, destacando el Hospital Universitario Virgen del Rocío (HUVR) de Sevilla.

Uno de los pilares fundamentales de ARTIFACTS es la creación de una plataforma federada para permitir a distintas instituciones sanitarias y centros de investigación

colaborar en el entrenamiento de modelos de IA sin necesidad de compartir los datos clínicos reales de sus pacientes.

En concreto, la aplicación práctica del proyecto se centra en el diagnóstico y pronóstico de cáncer. Para ello, se están aprovechando datos clínicos provenientes de varios hospitales españoles en conjunto con datos sintéticos generados a partir de ellos.

La Figura 3 muestra la arquitectura propuesta en ARTIFACTS para la plataforma federada a implementar, en la que cada hospital (cliente) puede entrenar sus datos de manera local y, además, puede asumir funciones de servidor en relación con otros clientes.

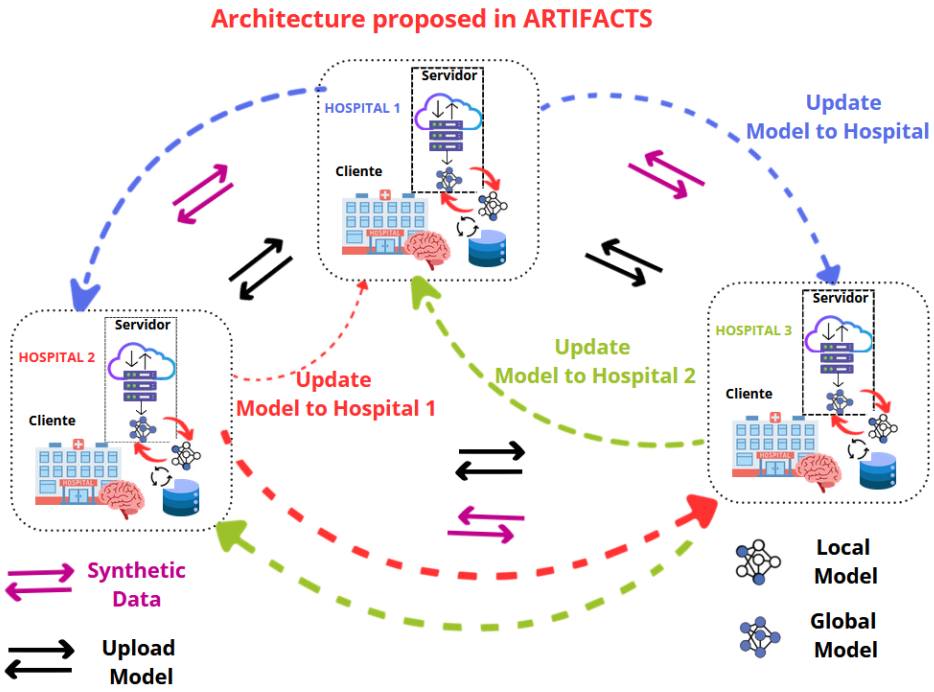


Figura 3: Arquitectura propuesta para ARTIFACTS (extraído de informe interno del proyecto)

No obstante, para llegar a ella, se plantea una primera versión basada en una arquitectura federada tradicional, como paso inicial. Esta primera aproximación se detalla en la Figura 4, donde se muestra un esquema clásico de AF compuesto por un servidor central y varios clientes. Cada cliente (hospital) entrena de forma independiente su propio modelo de *Deep Learning* utilizando únicamente sus datos locales. Las actualizaciones resultantes de estos entrenamientos se envían al servidor, el cual las agrega para actualizar el modelo

global y redistribuirlo posteriormente a los clientes. Este proceso se repite iterativamente hasta que el modelo converge y se alcanza un rendimiento satisfactorio.

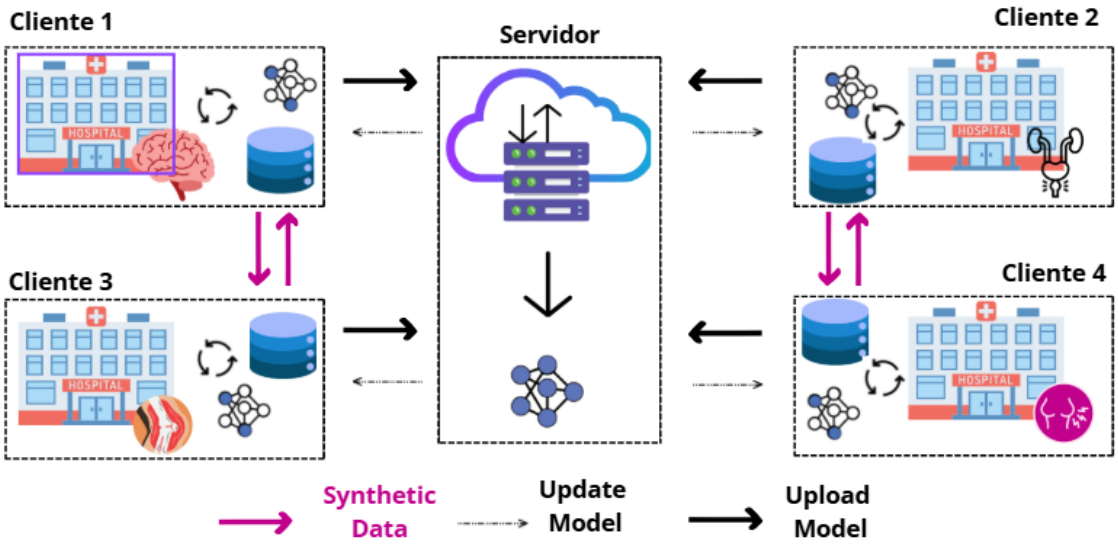


Figura 4: Esquema clásico de Aprendizaje Federado para ARTIFACTS (extraído de informe interno del proyecto)

Así pues, este TFG se ha desarrollado como parte de los trabajos realizados en el proyecto ARTIFACTS y con la supervisión y guía de los tutores, que son miembros de su equipo de investigación.

La implementación de un prototipo de esta arquitectura federada tradicional constituye el núcleo del TFG, sirviendo como base para futuras extensiones más avanzadas dentro del marco del proyecto ARTIFACTS. Está compuesto por una arquitectura básica de servidor y múltiples clientes, y se centra en validar el flujo funcional del proceso de federación, en la lógica de entrenamiento federado distribuido y en el código de su implementación.

Como objetivo adicional, se plantea comparar el rendimiento de un sistema de aprendizaje federado con el de un equivalente de aprendizaje centralizado. El fin es observar si, en un entorno controlado, la precisión del modelo entrenado mediante AF se mantiene cercana a la de un modelo entrenado con todos los datos centralizados. Aunque, en esta fase, estas pruebas aún no tienen valor clínico ni científico formal, pueden ofrecer una aproximación práctica y experimental sobre el potencial del AF en contextos reales.

La relevancia de este trabajo radica, en primer lugar, en formar parte de ARTIFACTS, un proyecto de investigación de envergadura, ambicioso y de alto impacto social. En segundo lugar, el trabajo se inscribe en un campo de investigación innovador y relevante: el AF aplicado al ámbito de la salud. Aportar un prototipo funcional en este contexto representa una contribución técnica modesta, pero significativa, considerando que se han puesto en práctica para su ejecución conocimientos, habilidades y destrezas que se han ido adquiriendo en distintas asignaturas a lo largo de toda la titulación y, asimismo, ha requerido un aprendizaje acerca del AF que el autor de este TFG no tenía al iniciar el trabajo.

Académicamente hablando, para mí, autor de este trabajo, este proyecto ha representado una oportunidad excepcional. Por un lado, formar parte de ARTIFACTS supone un reto técnico y organizativo de una envergadura muy superior a la de cualquier otro trabajo que haya realizado durante la carrera. También brinda la responsabilidad de participar, aunque sea en una pequeña parte, en un proyecto real y tan relevante junto a profesionales de alto nivel.

Pero toda dificultad es a su vez una oportunidad, vista desde los ojos que la quieren ver. Este proyecto es una oportunidad para consolidar los conocimientos aprendidos durante el grado y de confirmar todo lo aprendido durante los cuatro años que ha durado, entendiendo que son solo los primeros pasos de un aprendizaje continuo. Me ha permitido adentrarme en áreas poco o nada exploradas durante los estudios, como son la inteligencia artificial, el AF y levemente la ingeniería de datos, y también me ha permitido poder trabajar con tecnologías punteras como Python, PyTorch y Flower. Ha sido, además, una oportunidad de colaborar en un entorno técnico más profesional y de aprender de la experiencia de nuevos compañeros.

Este TFG es el broche a un largo grado universitario, que ha pasado a la vez deprisa y lento; una carrera trabajada desde la humildad y el esfuerzo, que espero poder y querer verla siempre con buenos ojos.

3 - METODOLOGÍA DE TRABAJO

I. Metodología seleccionada

La metodología seguida en este trabajo se basa en un enfoque iterativo e incremental, inspirado en los principios de las metodologías ágiles, adaptado al entorno individual y académico en el que se ha operado.

Las metodologías ágiles promueven la flexibilidad, la colaboración continua entre los interesados, la entrega frecuente de producto de valor y la capacidad de adaptación a cambios de requisitos. A diferencia de enfoques rígidos y lineales, como el ciclo de vida en cascada, el desarrollo ágil favorece la evolución progresiva del producto a través de ciclos de mejora constante [29].

El enfoque iterativo implica que el sistema se construye en sucesivas versiones, en las que se revisa y mejora lo ya desarrollado, permitiendo detectar errores de forma temprana y corregirlos en iteraciones posteriores. Por su parte, el enfoque incremental consiste en añadir funcionalidades de forma progresiva, construyendo sobre versiones anteriores. Cada incremento representa una versión funcional del sistema que contiene mejoras o nuevas capacidades, a menudo llamadas prototipos.

En la Figura 5 se puede observar una comparación visual entre la metodología en cascada y el ciclo típico de las metodologías ágiles.

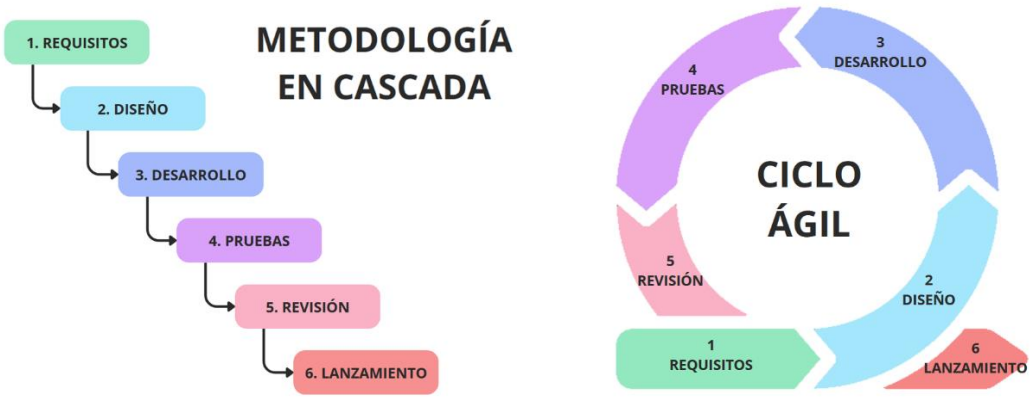


Figura 5: Metodología en cascada y ciclo ágil (elaboración propia)

Esta forma de trabajo ha permitido mantener un control constante sobre el desarrollo, garantizando la flexibilidad necesaria para adaptarse a las cambiantes necesidades y producir entregas parciales pero frecuentes de valor.

II. Organización del trabajo

El desarrollo del trabajo se ha organizado en distintas etapas, cada una con tareas y objetivos definidos. Aunque se ha planificado una estimación temporal, no se han empleado *sprints* en sentido estricto (periodos fijos de tiempo), sino que la planificación podía ajustarse en función del progreso. Este enfoque se inspira en los principios de metodologías ágiles, priorizando la adaptabilidad, la mejora continua y la entrega progresiva de valor.

Se han aplicado, específicamente, los siguientes principios clave de metodologías ágiles:

- Planificación iterativa del trabajo, dividiendo el proyecto en diferentes etapas y replanificando las tareas conforme fuese preciso.
- Revisión y supervisión frecuente y constante de avances y resultados con el resto del equipo del proyecto ARTIFACTS.
- Seguimiento permanente de los requisitos y adaptabilidad a sus cambios.
- Priorización de objetivos en cada etapa con el fin de obtener siempre producto de valor.
- Entrega incremental de funcionalidades.

Así pues, el desarrollo del trabajo se ha dividido en las siguientes etapas:

1. Análisis y planificación inicial: se han definido los objetivos, se ha revisado el estado del arte, se han recopilado los requisitos del sistema y se ha realizado una planificación inicial.
2. Diseño del sistema: se ha diseñado una arquitectura simple y clara que permitiese implementar el sistema y realizar la comparación del AF y el aprendizaje centralizado de forma controlada.

3. Implementación iterativa: a lo largo de diferentes etapas se han desarrollado las diversas funcionalidades y características del sistema, permitiendo reajustar la planificación. También se han realizado pruebas de rendimiento informales.
4. Experimentación: se han llevado a cabo pruebas formales con distintas configuraciones del sistema para ambos tipos de aprendizaje, evaluando el rendimiento y la precisión de ambos enfoques.
5. Análisis de resultados y documentación: se han analizado de forma global los resultados obtenidos y se ha preparado la documentación final.

En el capítulo *Planificación* se ahonda en cómo la estructuración cronológica del proyecto, cumpliendo con estas etapas.

III. Herramientas y recursos

Las herramientas y recursos empleados desde la perspectiva de la gestión han sido las siguientes:

- Git y GitHub: como sistemas de control del código y sus versiones de forma individual e informal.
- TeamGantt: para realizar la planificación temporal del proyecto y sus tareas mediante un diagrama de Gantt.
- GitHub: para definir las tareas mediante su funcionalidad de *issues* y *projects* ligada al control de versiones, y gestionar el progreso.
- Clockify: para registrar y analizar el tiempo invertido en cada tarea y etapa del proyecto.

Desde la perspectiva técnica, los recursos han sido los dos siguientes:

- Python como el lenguaje de programación principal del producto, debido a su flexibilidad, popularidad y gran compatibilidad con bibliotecas externas de IA.
- Un ordenador personal para realizar el desarrollo del producto de forma local.

Dado que el desarrollo del sistema se ha llevado a cabo de forma individual, no se ha implementado una gestión de la configuración formal ni compleja. Sin embargo, se han

seguido ciertas prácticas básicas para asegurar el control y la trazabilidad del código fuente y de las versiones generadas durante el desarrollo.

En particular, se ha utilizado Git y GitHub como sistema de control de versiones. Para organizar el trabajo, se han creado *issues* para cada tarea de desarrollo, permitiendo planificar y registrar las tareas en diferentes estados de progreso. Cada *commit* se ha nombrado siguiendo la convención Conventional Commits, incorporando además el número de la *issue* correspondiente para facilitar la trazabilidad entre cambios y tareas.

4 – REQUISITOS

I. Introducción

En esta sección se muestran los requisitos formalmente definidos para el producto de este trabajo. En primer lugar, se definen los objetivos principales; en segundo lugar, se recopilan los requisitos; y, en tercer lugar, se muestra la matriz de trazabilidad.

Los objetivos y requisitos han sido definidos agrupándolos en función de su tipo. Se ha seguido un estilo convencional de la industria, separando los requisitos en primero lugar entre funcionales y no funcionales y, posteriormente, en subtipos, como de usabilidad y de rendimiento. La clasificación usada es la siguiente:

- Objetivos (OBJ)
- Requisitos funcionales (RF)
- Requisitos no funcionales
    - Requisitos de seguridad y fiabilidad (RSF)
    - Requisitos de usabilidad e interfaz (RUE)
    - Requisitos de rendimiento y escalabilidad (RRE)
    - Requisitos de mantenibilidad y extensibilidad (RME)
    - Requisitos de compatibilidad e interoperabilidad (RCI)

Debido a que este TFG está enmarcado en las tareas técnicas del proyecto ARTIFACTS, los requisitos del trabajo han ido evolucionando para ajustarse a las necesidades del proyecto. Los requisitos aquí mostrados corresponden a sus estados finales.

II. Objetivos

A continuación, se definen los objetivos formales del sistema.

ID	OBJ-01	**Título**	Aprendizaje federado
Descripción	El sistema debe permitir realizar procesos de AF entre un servidor y varios clientes, con características configurables.		

ID	OBJ-02	**Título**	Aprendizaje centralizado
Descripción	El sistema debe permitir realizar procesos de aprendizaje de manera sencilla y directa con aprendizaje centralizado.		

ID	OBJ-03	**Título**	Gestor de pruebas
Descripción	El sistema debe permitir realizar pruebas de aprendizaje de manera sencilla y directa, tanto para la versión de AF como para la centralizada.		

ID	OBJ-04	**Título**	Escalabilidad y sostenibilidad
Descripción	El sistema debe estar diseñado en previsión de un uso y mantenimiento futuro prolongado, valorando su capacidad de ampliación y extensión.		

III. Requisitos funcionales

A continuación, se definen los requisitos funcionales del sistema. Entrarían en el ámbito de las comúnmente llamadas “reglas de negocio”.

ID	RF-01	**Título**	Procesos de aprendizaje federado
Descripción	El sistema de AF debe permitir establecer un entorno de AF habilitando conexión entre un servidor y múltiples clientes, facilitando la ejecución de procesos de entrenamiento distribuido.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

ID	RF-02	**Título**	Procesos de aprendizaje centralizado
Descripción	El sistema centralizado debe permitir establecer un entorno de aprendizaje centralizado, en el que todos los datos se recopilan en un único nodo donde se lleva a cabo el entrenamiento completo del modelo.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

ID	RF-03	**Título**	Lectura de datos
Descripción	El sistema de AF debe permitir realizar los procesos de aprendizaje a partir de datos leídos por cada cliente.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

ID	RF-04	**Título**	Almacenamiento del modelo	
Descripción	El sistema de AF debe permitir guardar el modelo tras su entrenamiento.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RF-05	**Título**	Carga del modelo	
Descripción	El sistema de AF debe permitir cargar un modelo ya entrenado para que pueda servir de base a un nuevo entrenamiento.			
Importancia	☐ Esencial		☐ Deseable	☒ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RF-06	**Título**	Visualización de métricas	
Descripción	El sistema de AF debe mostrar, como resultado de los procesos de aprendizaje, varias métricas para determinar la calidad del entrenamiento.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☒ Alta		☐ Media	☐ Baja

ID	RF-07	**Título**	*Testing accuracy*	
Descripción	El sistema de AF debe mostrar, entre otras métricas, la precisión (*accuracy)* del aprendizaje con los datos de prueba.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☒ Alta		☐ Media	☐ Baja

ID	RF-08	**Título**	*Training accuracy*	
Descripción	El sistema de AF debe mostrar, entre otras métricas, la precisión (*accuracy)* de la fase de entrenamiento.			
Importancia	☐ Esencial		☒ Deseable	☐ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RF-09	**Título**	Validación del servidor	
Descripción	El sistema debe permitir realizar validación del modelo entrenado en el servidor con sus propios datos.			
Importancia	☐ Esencial		☒ Deseable	☐ Opcional
Prioridad	☐ Alta		☐ Media	☒ Baja

ID	RF-10	**Título**	*Cross validation*	
Descripción	El sistema de AF debe permitir realizar la técnica de validación cruzada para entrenar y probar el modelo.			
Importancia	☐ Esencial		☐ Deseable	☒ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RF-11	**Título**	Visualización de gráficos de datos	
Descripción	El sistema debe permitir mostrar gráficos de los datos iniciales, representando, a ser posible, información relevante como la distribución o correlación de los datos.			
Importancia	☐ Esencial		☐ Deseable	☒ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RF-12	**Título**	Visualización de gráficos de métricas	
Descripción	El sistema debe permitir la generación y visualización de gráficos de las métricas de resultado del aprendizaje del modelo.			
Importancia	☐ Esencial		☒ Deseable	☐ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RF-13	**Título**	Registro del proceso	
Descripción	El sistema debe conservar un registro claro y estructurado del proceso de entrenamiento, incluyendo métricas de rendimiento, parámetros utilizados y evolución de los resultados, con el fin de facilitar posteriores análisis o depuraciones.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☒ Alta		☐ Media	☐ Baja

ID	RF-14	**Título**	Ejecución de pruebas	
Descripción	El sistema debe permitir realizar pruebas de aprendizaje con configuraciones variables, como las relacionadas con la configuración del servidor (p.ej., número de rondas), la configuración de los clientes (como el número de clientes o la proporción de los conjuntos de entrenamiento y validación) y aspectos del propio aprendizaje (como la duración o los hiperparámetros utilizados).			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

IV. Requisitos no funcionales

A continuación, se definen los requisitos no funcionales del sistema, que están agrupados en subtipos.

Requisitos de seguridad y fiabilidad

ID	RSF-01	**Título**	Integridad de los datos	
Descripción	El sistema debe asegurar que los procesos de aprendizaje usen exclusiva y totalmente los datos disponibles, sin perder datos en el proceso ni manipular o inventar datos.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☒ Alta		☐ Media	☐ Baja

ID	RSF-02	**Título**	Confidencialidad de los datos	
Descripción	El sistema debe asegurar que se preserve la confidencialidad de los datos utilizados por cada nodo, sin que ninguna información de estos salga de él. No se considera aquella información posiblemente implícita en los pesos del modelo local entrenado como información confidencial transmitida.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☒ Alta		☐ Media	☐ Baja

ID	RSF-03	**Título**	Disponibilidad del sistema	
Descripción	El sistema debe asegurar su disponibilidad durante los procesos de aprendizaje, evitando interrupciones inesperadas.			
Importancia	☒ Esencial		☐ Deseable	☐ Opcional
Prioridad	☒ Alta		☐ Media	☐ Baja

Requisitos de usabilidad e interfaz

ID	RUI-01	**Título**	Facilidad de uso	
Descripción	El sistema debe ser fácil de usar por los usuarios, sin necesidad de interfaces de usuario, pero con cierta abstracción desde la consola de comandos.			
Importancia	☐ Esencial		☒ Deseable	☐ Opcional
Prioridad	☐ Alta		☒ Media	☐ Baja

ID	RUI-02	**Título**	Documentación
Descripción	El sistema debe disponer de alguna documentación que detalle su objetivo y cómo usarlo.		
Importancia	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☐ Media	☒ Baja

Requisitos de rendimiento y escalabilidad

ID	RRE-01	**Título**	Eficacia del sistema
Descripción	El sistema debe asegurar una eficacia alta del aprendizaje. Dado que el resultado depende en gran medida de los datos, la configuración y el entorno de cada experimento, se toma como referencia orientativa razonable una *accuracy* media superior a 70%.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

ID	RRE-02	**Título**	Rapidez de conexión del sistema
Descripción	El sistema debe asegurar unos tiempos razonables para el establecimiento de conexión y comienzo de cada ronda de aprendizaje. Dado que depende en gran medida de los clientes, la configuración y el entorno del experimento, se toma como referencia orientativa razonable un tiempo medio menor a 30 segundos.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

ID	RRE-03	**Título**	Escalabilidad del sistema
Descripción	El código del sistema deberá estar predispuesto a un incremento en el volumen de los datos, de los clientes y de las configuraciones del aprendizaje, sin perjudicar significativamente el rendimiento.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

Requisitos de mantenibilidad y extensibilidad

ID	RME-01	**Título**	Tecnologías modernas y con soporte
Descripción	El sistema debe estar creado con tecnologías modernas y seguras, y que cuenten con un amplio soporte, previendo un uso y mantenimiento futuro prolongado.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
ID	RME-02	**Título**	Código en inglés

Descripción	El código del sistema se hará en inglés para evitar confusiones con términos comunes del sector que habitualmente vienen en dicho idioma.		
Importancia	☐ Esencial	☐ Deseable	☒ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja

ID	RME-03	**Título**	Claridad de código
Descripción	El código se escribirá de forma ordenada, clara y modular, para así facilitar su entendimiento y futuro mantenimiento.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja

ID	RME-04	**Título**	Extensibilidad del código
Descripción	El código debe permitir ser ampliado y extendido en el futuro, por lo que su modularidad será importante.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja

Requisitos de compatibilidad e interoperabilidad

ID	RCI-01	**Título**	Sistemas operativos
Descripción	El sistema debe ser funcional en dos sistemas operativos: Windows y Linux.		
Importancia	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja

V. Matriz de trazabilidad de requisitos

A continuación, en la página siguiente, se muestra la matriz de trazabilidad de requisitos (Tabla 1), que relaciona los requisitos del sistema con los objetivos que satisfacen.

Tabla 1: Matriz de trazabilidad de requisitos

ID requisito	OBJ-01	OBJ-02	OBJ-03	OBJ-04
RF-01	X			
RF-02		X		
RF-03	X	X		
RF-04	X	X		
RF-05	X	X		
RF-06	X	X	X	
RF-07	X	X	X	
RF-08	X	X	X	
RF-09	X			
RF-10	X	X		
RF-11			X	
RF-12			X	
RF-13			X	
RF-14			X	
RSF-01	X	X	X	
RSF-02	X	X		
RSF-03	X	X		
RUI-01			X	X
RUI-02			X	X
RRE-01	X	X	X	
RRE-02	X	X	X	
RRE-03				X
RME-01	X			X
RME-02				X
RME-03				X
RME-04	X	X	X	X
RCI-01	X	X		

5 – PLANIFICACIÓN

I. Tareas

Una vez recopilados y analizados los requisitos del sistema, se ha procedido a planificar el desarrollo del proyecto a través de la definición de tareas concretas. Estas tareas se han organizado en diferentes bloques, coincidiendo con las etapas del proyecto.

Caben destacar dos consideraciones específicas. Primero, la fase de inicio ha sido planificada con anterioridad al resto, pues había de tener lugar cronológicamente antes que la planificación formal. Segundo, se han reservado algunas horas de carga de trabajo adicional para la fase de cierre, la cual involucra la entrega del producto y la preparación de la presentación.

En línea con la metodología ágil planteada, la lista de tareas no ha sido estática, sino que, a lo largo del desarrollo, han surgido nuevas necesidades, se han creado nuevas tareas, se han modificado previas y se ha ajustado la planificación de acorde.

A continuación, se muestra la lista final de tareas ordenadas según su etapa.

Inicio

ID	TI-01	**Título**	Reunión para definir trabajo
Descripción	Realizar reunión para definir el trabajo, aclarar el tema y establecer cualquier acuerdo.		

ID	TI-02	**Título**	Toma de contacto con AF
Descripción	Entender qué es AF, conocer el ambiente y estudiar el estado del arte.		

ID	TI-03	**Título**	Reunión con equipo de proyecto
Descripción	Reunión con equipo del proyecto para entender de qué va el proyecto ARTIFACTS, esclarecer objetivos del TFG y decidir por dónde comenzar.		

ID	TI-04	Título	Estudio de viabilidad
Descripción	Estudiar ligeramente posible viabilidad de las opciones propuestas para el trabajo.		

Planificación preliminar

ID	TPP-01	Título	Def. de objetivos preliminares
Descripción	Definir los objetivos preliminares del proyecto de manera formal, aunque provisional.		

ID	TPP-02	Título	Def. de requisitos preliminares
Descripción	Definir los requisitos preliminares del proyecto de manera formal, aunque provisional.		

ID	TPP-03	Título	Def. de metodologías
Descripción	Definir las metodologías a seguir en el proyecto de manera formal.		

ID	TPP-04	Título	Planificación preliminar
Descripción	Realizar una planificación preliminar del proyecto, incluyendo tareas y estimación temporal. Analizar riesgos.		

ID	TPP-05	Título	Prueba de conexión
Descripción	Realizar prueba de conexión entre la Universidad Sevilla, la Universidad Politécnica de Cataluña y la Universidad Carlos III de Madrid para comprobar viabilidad de traspaso de información y comprobar *firewalls*.		

ID	TPP-06	Título	Estudio del arte
Descripción	Estudiar estado del arte de AF para entender más del tema y ahondar en posibilidades.		

ID	TPP-07	Título	Análisis y selección de tecnologías
Descripción	Estudiar posibles opciones para las tecnologías. Analizar *frameworks*, bibliotecas y otros. Mantener Python en un principio.		

ID	TPP-08	**Título**	Reunión con equipo del proyecto
Descripción	Reunión con equipo para aclarar objetivo del proyecto ARTIFACTS y del TFG y compartir avances sobre las tecnologías.		

Desarrollo de prototipo funcional

ID	TDP-01	**Título**	Desarrollo de prototipo funcional
Descripción	Desarrollar un prototipo funcional de un sistema de AF entre un servidor y varios clientes. Probar conexión remota.		

ID	TDP-02	**Título**	Reunión con equipo del proyecto
Descripción	Reunión con equipo para iterar sobre los objetivos del TFG y compartir avances sobre el prototipo.		

Planificación formal

ID	TPF-01	**Título**	Estudio del arte
Descripción	Estudiar estado del arte de AF para entender más del tema y ahondar en posibilidades.		

ID	TPF-02	**Título**	Definición de requisitos formales
Descripción	Definir los requisitos del proyecto de manera formal.		

ID	TPF-03	**Título**	Reunión con equipo del proyecto
Descripción	Reunión con equipo para esclarecer dirección del proyecto ARTIFACTS.		

ID	TPF-04	**Título**	Planificación formal
Descripción	Realizar una planificación formal del proyecto, incluyendo tareas y estimación temporal. Actualizar y consolidar riesgos.		

ID	TPF-05	**Título**	Reunión de supervisión
Descripción	Reunión de supervisión para aprobar planificación del proyecto.		

Desarrollo inicial

ID	TDI-01	**Título**	Desarrollo del sistema AF inicial
Descripción	Desarrollar el sistema AF. Empezar a incluir funcionalidades. Usar BBDD provista.		

ID	TDI-02	**Título**	Desarrollo del sistema centralizado inicial
Descripción	Desarrollar versión centralizada análoga al sistema AF, para así poder comparar resultados.		

ID	TDI-03	**Título**	Def. de pruebas AF vs Centralizado
Descripción	Idear pruebas para comparar rendimiento entre el sistema AF y el sistema centralizado.		

ID	TDI-04	**Título**	Implementación y ejecución de pruebas AF vs Centralizado
Descripción	Implementar y ejecutar las pruebas comparativas definidas. Recoger resultados.		

ID	TDI-05	**Título**	Realización de memoria TFG (transversal)
Descripción	Redacción de la memoria del TFG. Incluir lo realizado hasta el momento. Utilizar estilo de documento formal y cercano al definitivo. Se pulirá más adelante.		

ID	TDI-06	**Título**	Reuniones de supervisión (transversal)
Descripción	Varias reuniones para realizar seguimiento del proyecto.		

Desarrollo final

ID	TDF-01	**Título**	Estudio del arte AF vs Centralizado
Descripción	Estudiar estado del arte para entender qué comparaciones realizar. Especial atención a casos de modelos diferentes en clientes y en servidor.		

ID	TDF-02	**Título**	Desarrollo del sistema AF mejorado
Descripción	Desarrollar el sistema AF. Incluir más funcionalidades y mejorar la calidad de vida de su utilización.		

ID	TDF-03	**Título**	Desarrollo de versión centralizada mejorada
Descripción	Desarrollar versión centralizada análoga al sistema AF, para así poder comparar resultados. Incluir funcionalidades céntricas del rendimiento, no adicionales que no afecten.		

ID	TDF-04	**Título**	Def. de pruebas AF vs Centralizado
Descripción	Idear pruebas para comparar rendimiento entre el sistema AF y el sistema centralizado.		

ID	TDF-05	**Título**	Implementación. y ejecución de pruebas AF vs Centralizado
Descripción	Implementar y ejecutar las pruebas definidas. Recoger resultados.		

ID	TDF-06	**Título**	Realización de memoria TFG (transversal)
Descripción	Finalizar la redacción de la memoria del TFG. Incluir lo realizado hasta el momento. Actualizar la planificación, requisitos y riesgos con las versiones más recientes. Añadir el análisis de coste con las horas invertidas reales. Conclusiones. Pulir ya de forma definitiva y atención a la consistencia.		

ID	TDF-07	**Título**	Reuniones de supervisión (transversal)
Descripción	Varias reuniones para realizar seguimiento del proyecto.		

Cierre

ID	TC-01	**Título**	Entrega de la memoria y código
Descripción	Entregar memoria del TFG y código y resto de artefactos del trabajo. Finalizar todo antes de hacerlo. Fijarse bien en las fechas.		

ID	TC-02	**Título**	Realización de la presentación
Descripción	Realizar la presentación del TFG.		

ID	TC-03	**Título**	Entrega de la presentación
Descripción	Entregar la presentación del TFG.		

ID	TC-04	**Título**	Aprendizaje de la presentación
Descripción	Finalizar aprendizaje de la presentación del TFG.		

II. Planificación y seguimiento temporal

Tras la definición inicial de las tareas del proyecto, se ha elaborado una planificación temporal utilizando un diagrama de Gantt. Esta herramienta permite representar gráficamente la distribución temporal de las actividades, su secuenciación y duración estimada. Esto facilita, además, una visión global del desarrollo del proyecto.

El cronograma inicial ha servido como guía para organizar el trabajo y establecer una hoja de ruta de trabajo clara, indicando los hitos principales y las dependencias entre tareas. Sin embargo, como es natural, con el transcurrir del proyecto, se han requerido ciertos ajustes respecto la planificación original. Estos cambios se han debido a diversos factores, tales como:

- La aparición de tareas no previstas inicialmente.
- La reestructuración de tareas conforme han ido evolucionando.
- Cambios en las prioridades o retrasos puntuales por dificultades técnicas o sucesos externos.

Consecuentemente, el diagrama de Gantt ha ido evolucionando junto al proyecto para reflejar estos cambios. Este aspecto es vital en todo proyecto, pues permite hacer un seguimiento realista del progreso y detectar desviaciones de forma temprana. En esta memoria se incluyen tanto la planificación original como la versión actualizada con los tiempos reales dedicados a cada tarea, con el objetivo de evidenciar la evolución temporal del trabajo y contrastar lo planificado con lo ejecutado. Se muestran ambos a nivel de etapa, ocultando el detalle a nivel de tarea.

La Figura 6 presenta el diagrama de Gantt con la planificación inicial, mientras que la Figura 7 muestra el diagrama de Gantt final plenamente actualizado.

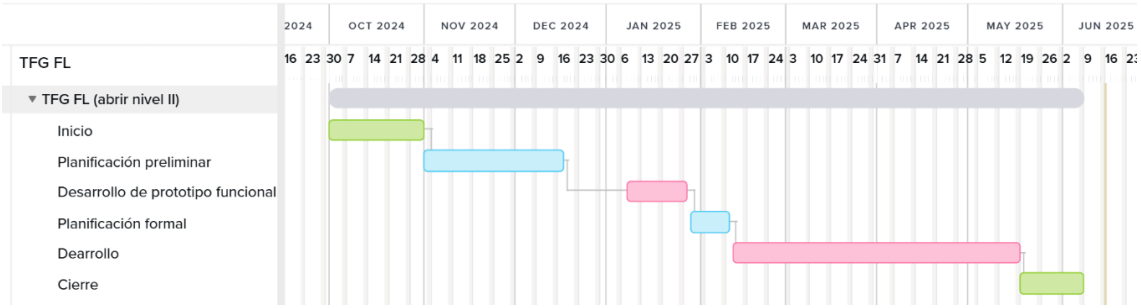


Figura 6: Diagrama de Gantt inicial a nivel de etapas (elaboración propia)

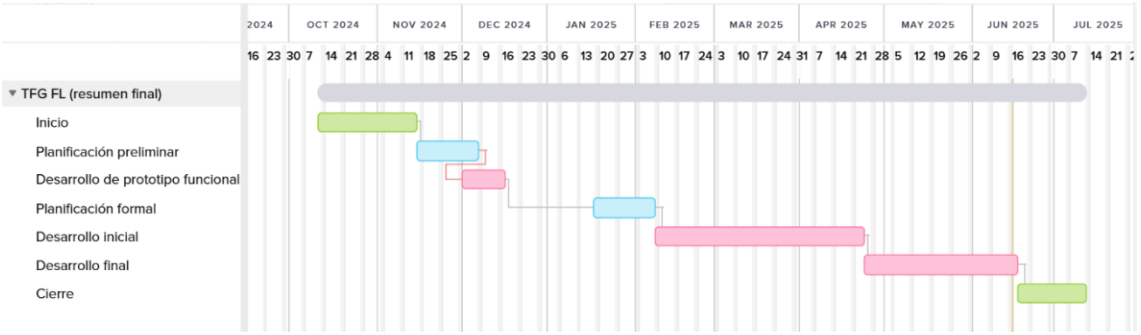


Figura 7: Diagrama de Gantt final a nivel de etapas (elaboración propia)

Como se puede observar, incluso a este nivel de detalle, el cronograma ha sido objeto de varios cambios. A continuación se comentarán los casos más destacables.

En primer lugar, la planificación preliminar ha sido más breve de lo previsto por lo que el prototipo funcional se pudo realizar antes de las vacaciones de Navidad. Esto ha permitido adelantar la planificación formal casi un mes.

Otro cambio relevante ha sido dividir la etapa de desarrollo en dos etapas, cada una con sus tareas de pruebas y revisión. De este modo, se ha aprovechado la capacidad iterativa de este proyecto para al final de la etapa de desarrollo inicial poder hacer una retrospectiva y ajustar la planificación de cara a la etapa de desarrollo final.

Por último, el proyecto ha sufrido un retraso de prácticamente un mes en su fecha de finalización. Estaba previsto que el desarrollo del sistema acabara a mediados del mes de abril, y la memoria para mediados de mayo. Sin embargo, ambos se acabaron retrasando a mitad de junio. La causa principal de este retraso ha sido la complejidad que han conllevado la implementación de ciertas funcionalidades del sistema, en especial, el intento de mejora de la precisión del aprendizaje. Adicionalmente, la planificación también se ha visto afectada por sucesos externos al TFG. Estos riesgos se identificaron y gestionaron de la forma adecuada.

III. Resultados y estimación de costes

Aunque este proyecto no ha implicado un coste económico directo al tratarse de un trabajo académico individual, sí es posible realizar una estimación teórica del coste basada en los recursos invertidos. Las unidades de tiempo han sido redondeadas a la hora para mayor legibilidad.

Se incluyen tres apartados: los costes de los recursos de *hardware* y *software* empleados, los costes de personal, basado en el tiempo invertido, y el coste total del proyecto.

La etapa de Cierre no se incluye en la estimación de costes ni en la comparación entre tiempo planificado y tiempo real, ya que en el momento de redactar esta memoria dicha etapa aún no ha comenzado, por lo que no ha sido posible registrar el tiempo efectivamente invertido. Cabe comentar que en la planificación se estimaron 23 horas a esta fase, principalmente para la preparación de la presentación final.

Costes de hardware y software

Aunque este proyecto se ha desarrollado como un trabajo académico sin incurrir en costes económicos, la Tabla 2: Costes de hardware y software muestra una estimación completa del coste total que tendría en un contexto profesional, basada en los costes reales de cada producto utilizado.

Tabla 2: Costes de hardware y software

Recurso	Tipo	Coste
Ordenador portátil	Hardware	1.100 €
Sistema operativo: Windows 10	Software	125 €
Microsoft 365	Software	99 €
IDE: Visual Studio Code	Software	0 €
Frameworks y librerías: Flower, Python, PyTorch, etc.	Software	0 €
Otros: Git, TeamGantt, etc.	Software	0 €
TOTAL		1.324 €

Costes de personal

En base a los datos recopilados por la Junta de Andalucía en el año 2018 [30] y contemplando el aumento natural de los precios, se considera un salario estimado de 35 euros la hora para el personal de este TFG. La matriz mostrada en la Tabla 3 resume el tiempo y los costes estimados, y muestra la diferencia entre las previsiones y los resultados reales.

Tabla 3: Comparación de tiempo y costes previstos y reales

Etapa	Horas previstas	Horas reales	Diferencia de horas	Coste estimado previsto	Coste estimado real	Diferencia de coste
Inicio	34	24	- 10	1.190 €	840 €	- 350 €
Planificación preliminar	49	44	- 5	1.715 €	1.540 €	- 175 €
Desarrollo de prototipo	22	16	- 6	770 €	560 €	- 210 €
Planificación formal	29	28	- 1	1.015 €	980 €	- 35 €
Desarrollo inicial	89	86	- 3	3.115 €	3.010 €	- 105 €
Desarrollo final	114	134	+ 20	3.990 €	4.690 €	+ 700 €
TOTAL	337	332	- 5	11.795 €	11.620 €	- 175 €

Coste total del proyecto

Sumando el coste de los recursos de *hardware* y *software*, y el coste real de personal, la Tabla 4 muestra el coste total del proyecto.

Tabla 4: Coste total del proyecto

Concepto	Coste del proyecto
Recursos materiales	1.324 €
Coste personal	11.620 €
TOTAL	12.944 €

Observaciones

El coste de personal representa la mayor parte del presupuesto estimado (cerca del 90%). El tiempo previsto fue ligeramente superior al real en cada una de las etapas, salvo en el desarrollo final, en el que se subestimó en gran manera el tiempo a dedicar a la presente memoria. En el cómputo global, el resultado final fue, quizás sorprendentemente, muy cercano a la realidad.

IV. Gestión de riesgos

Durante la planificación y el mismo desarrollo del proyecto, se han identificado una serie de riesgos potenciales que podrían afectar al desarrollo. Estos riesgos se han evaluado en función de su probabilidad de ocurrencia y del impacto que podrían tener sobre el cumplimiento de los objetivos, los plazos y la calidad del sistema.

Los riesgos analizados incluyen aspectos técnicos, organizativos y personales, teniendo en cuenta que el trabajo fue desarrollado de forma individual y en un entorno académico. Además, se contemplaron tanto riesgos negativos como positivos. Se entiende por riesgos positivos aquellas oportunidades que podrían beneficiar al desarrollo o mejorar los resultados esperados.

A lo largo del proyecto se ha realizado un seguimiento de estos riesgos para que, en caso de ocurrencia, decidir cómo actuar.

A continuación, se muestra la lista de riesgos.

ID	Riesgo-01	**Título**	Mala definición de alcance	
Probabilidad	☐ Alta		☐ Media	☒ Baja
Impacto	**Alcance**	☒ Alto	☐ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☒ Alto	☐ Medio	☐ Bajo
Mitigación preventiva	Revisar y validar el alcance al principio y de forma recurrente.			
Mitigación reactiva	Redefinir alcance y replanificar cronograma.			

ID	Riesgo-02	**Título**	Cambios frecuentes en alcance	
Probabilidad	☐ Alta		☒ Media	☐ Baja
Impacto	**Alcance**	☒ Alto	☐ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☐ Alto	☒ Medio	☐ Bajo
Mitigación preventiva	Revisar y validar el alcance al principio y de forma recurrente.			
Mitigación reactiva	Revisar y validar alcance, y limitar cambios a lo estrictamente necesario.			

ID	Riesgo-03	**Título**	Falta de tiempo por carga lectiva	
Probabilidad	☒ Alta		☐ Media	☐ Baja
Impacto	**Alcance**	☐ Alto	☒ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☐ Alto	☐ Medio	☒ Bajo
Mitigación preventiva	Identificar periodos críticos y planificar cronograma con holgura.			
Mitigación reactiva	Replanificar cronograma y redefinir alcance si es preciso.			

ID	Riesgo-04	**Título**	Enfermedad	
Probabilidad	☒ Alta		☐ Media	☐ Baja
Impacto	**Alcance**	☐ Alto	☒ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☒ Alto	☐ Medio	☐ Bajo
Mitigación preventiva	Planificar cronograma con holgura.			
Mitigación reactiva	Retomar el ritmo progresivamente. Replanificar cronograma y redefinir alcance si es preciso.			

ID	Riesgo-05	**Título**	Cambio de tecnologías	
Probabilidad	☐ Alta	☒ Media	☐ Baja	
Impacto	**Alcance**	☐ Alto	☒ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☐ Alto	☐ Medio	☒ Bajo
Mitigación preventiva	Estudiar tecnologías candidatas y valorar estabilidad.			
Mitigación reactiva	Evaluar rápidamente las nuevas herramientas y ajustar el diseño si es viable.			

ID	Riesgo-06	**Título**	Tecnologías conocidas	
Probabilidad	☐ Alta	☐ Media	☒ Baja	
Impacto	**Alcance**	☐ Alto	☒ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☐ Alto	☒ Medio	☐ Bajo
Mitigación preventiva	Diseñar el cronograma con tareas opcionales de mejora.			
Mitigación reactiva	Usar el tiempo ganado para mejorar la calidad o ampliar funcionalidades.			

ID	Riesgo-07	**Título**	Mayor eficacia de la esperada	
Probabilidad	☐ Alta	☒ Media	☐ Baja	
Impacto	**Alcance**	☒ Alto	☐ Medio	☐ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☒ Alto	☐ Medio	☐ Bajo
Mitigación preventiva	Planificar tareas opcionales de mejora.			
Mitigación reactiva	Usar el tiempo ganado para mejorar la calidad o ampliar las funcionalidades.			

ID	Riesgo-08	**Título**	Tiempo sobrante por poca carga lectiva	
Probabilidad	☐ Alta	☐ Media	☒ Baja	
Impacto	**Alcance**	☐ Alto	☐ Medio	☒ Bajo
	Tiempo	☒ Alto	☐ Medio	☐ Bajo
	Calidad	☐ Alto	☐ Medio	☒ Bajo
Mitigación preventiva	Anticipar momentos de poca carga y planificar tareas opcionales de mejora.			
Mitigación reactiva	Usar el tiempo ganado para mejorar la calidad o ampliar las funcionalidades.			

6 – DISEÑO

Esta sección presenta el diseño de las dos arquitecturas que conforman el núcleo del trabajo: la arquitectura federada y la arquitectura centralizada, donde se describen con detalle las principales decisiones de diseño tomadas para el desarrollo del producto.

I. Arquitectura federada

El sistema de AF está basado en una arquitectura clásica de clientes-servidor, con múltiples clientes y un único servidor.

Los componentes principales del sistema son:

- Clientes: diferentes nodos que poseen y procesan localmente sus propios datos, con los que entrarán un modelo local. Se conectarán con el servidor para el proceso federado.
- Datos: cada cliente cuenta con un conjunto exclusivo de datos.
- Servidor: nodo central encargado de coordinar el proceso federado. Su principal función es orquestar las rondas de entrenamiento, recibir los modelos locales y construir el modelo agregado.
- Modelos: tanto los clientes como el servidor disponen de una instancia del modelo de aprendizaje.
- Estrategia de agregación: define cómo se combinan los modelos locales en el servidor para obtener un modelo global actualizado.
- *Framework* de aprendizaje federado: se emplea un *framework* especializado para facilitar la implementación del sistema. Este proporciona herramientas para la orquestación del flujo de AF, la comunicación de los clientes con el servidor y la gestión de rondas de entrenamiento y sus resultados. Se han evaluado diferentes alternativas.

La Figura 8 conceptualiza la arquitectura federada implementada.

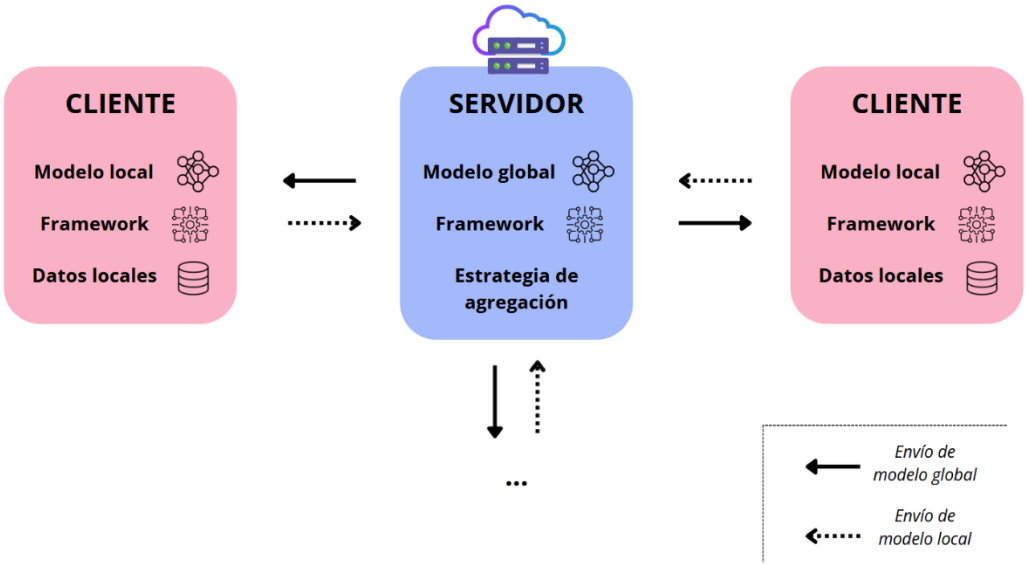


Figura 8: Arquitectura federada conceptual implementada (elaboración propia)

II. Arquitectura centralizada

Para el sistema de aprendizaje centralizado tradicional, todos los datos se encuentran reunidos en un único nodo central. Este enfoque difiere sustancialmente del AF al prescindir de servidores y múltiples clientes.

No se requiere ninguna estrategia de agregación ni *framework* federado, ya que el entrenamiento se realiza de forma directa y secuencial sobre el conjunto de datos completo.

Sin embargo, se reutiliza parte del código común con la implementación del AF, como los módulos de lectura de datos, carga de la configuración, definición del modelo e incluso el propio entrenamiento y evaluación del modelo. Esto permite mantener la coherencia en los experimentos y facilitar la comparación entre ambos enfoques, a la vez que reducir redundancia de código.

III. Pruebas

Para la evaluación sistemática del AF en el sistema, se ha desarrollado un módulo de pruebas dedicado, que automatizará el proceso de experimentación. Este módulo es responsable de las siguientes funcionalidades:

- Ejecutar múltiples épocas de entrenamiento tanto de AF como centralizado.
- Coordinar la ejecución de los distintos clientes y del servidor.
- Registrar las métricas obtenidas durante el entrenamiento.

IV. Interfaz

Aunque el sistema no cuenta con una interfaz gráfica de usuario, sí se ha diseñado una interfaz de interacción clara y estructurada principalmente a través de la línea de comandos. Esta elección se debe a la naturaleza técnica del proyecto y a la necesidad de facilitar su uso por parte de usuarios objetivos: desarrolladores e investigadores.

Para ello, se han desarrollado distintos puntos de entrada al sistema que abstraen la complejidad de la implementación subyacente:

- Fichero de configuración: permite ajustar todos los parámetros clave del sistema (estructura del modelo, número de rondas, etc.) sin necesidad de modificar el código fuente. Esto proporciona flexibilidad y separación entre lógica y configuración.
- *Scripts* principales de ejecución: permiten ejecutar los procesos de aprendizaje, como lanzar un experimento federado, ejecutar un entrenamiento centralizado o gestionar los clientes. Funcionan como interfaz funcional directa para el usuario.
- Interfaces de línea de comandos (*CLIs*) básicas: los *scripts* se ejecutan desde la terminal de forma flexible empleando argumentos simples adicionales.

V. Consideraciones adicionales

Se han tenido en cuenta diversas buenas prácticas de diseño con el objetivo de mejorar la mantenibilidad, extensibilidad y portabilidad del sistema:

- Parámetros de configuración: todas las variables clave (hiperparámetros, número de rondas, estructura del modelo, etc.) están centralizadas en archivos de configuración externos, facilitando la experimentación y los cambios sin necesidad de modificar el código fuente.
- Registro del proceso: durante la ejecución de los programas se generan archivos de *log* que almacenan información detallada del progreso, incluyendo los pasos, tiempos, métricas y posibles errores. Esto permitirá auditar el comportamiento del sistema y analizar los resultados *a posteriori* y de forma ordenada.
- Compatibilidad multiplataforma: todo el código está diseñado para funcionar tanto en sistemas operativos Windows como Linux. Se utiliza la biblioteca estándar "os" de Python para gestionar rutas y operaciones del sistema de forma segura.

7 – IMPLEMENTACIÓN

El presente capítulo se divide en 15 secciones, abordando los distintos aspectos clave de la implementación del sistema para ambos tipos de aprendizaje, y comentando las justificaciones detrás de las decisiones tomadas. Debido a la extensión de este capítulo, cada sección comenzará en una página nueva, con el objetivo de mejorar la claridad visual y facilitar la lectura.

I. Selección de *framework*

El primer paso en el diseño del sistema de AF ha sido la elección del *framework* de desarrollo.

Dado que un sistema de AF implica múltiples componentes (coordinación entre clientes y servidor, agregación de modelos, integración con bibliotecas de aprendizaje automático, etc.), contar con un *framework* adecuado es esencial, pues favorece acelerar el desarrollo, asegurar la compatibilidad y la implementación de técnicas de privacidad, comunicación y escalabilidad.

A pesar de ser un campo relativamente reciente, existen ya numerosos *frameworks* en el ecosistema actual, cada uno con enfoques, fortalezas y limitaciones diferentes. Para este trabajo, se han analizado en base a las prioridades del proyecto los *frameworks* más relevantes del panorama actual: Flower, PySyft, FedPerl, Fed-BioMed, NVIDIA FLARE y TensorFlow Federated [31] [32] [33].

En la Tabla 5 se resume la comparativa entre los tres principales candidatos analizados en mayor profundidad. El *framework* finalmente elegido ha sido Flower [34].

Tabla 5: Comparativa entre Flower, PySyft y FedPerl

Criterio	Flower	PySyft	FedPerl
Enfoque principal	Flexible, orientado a producción e investigación	Alta seguridad y colaboración P2P	Escenarios experimentales
Bibliotecas ML	Alta compatibilidad: PyTorch, TensorFlow, Keras, Scikit-learn, etc.	Principalmente PyTorch y TensorFlow	Principalmente PyTorch
Arquitectura base	Cliente-servidor con comunicación gRPC o HTTP	Cliente-servidor y P2P	Cliente-servidor
Soporte P2P	No nativo; emulable con sockets o APIs externas	Soporte nativo y real	No nativo; emulable en entornos controlados
Herramientas de privacidad	Soporte avanzado no nativo, pero configurable con herramientas externas	Soporte avanzado nativo; privacidad diferencial y encriptación homomórfica	Soporte avanzado configurable manualmente
Documentación y comunidad	Comunidad muy activa, documentación clara, ejemplos variados	Comunidad sólida centrada en la seguridad	Comunidad reducida, orientada a investigación experimental

Tras la evaluación, se ha elegido Flower como *framework* para el desarrollo del sistema. Esta decisión se ha basado en su equilibrio entre flexibilidad, facilidad de uso, compatibilidad y extensibilidad. Es un *framework* orientado tanto a investigación como a producción, y cuenta con una alta compatibilidad con herramientas externas, amplia documentación y una comunidad activa.

Aunque Flower cuenta con puntos negativos, como la falta de soporte nativo de arquitectura P2P y las limitaciones para la aplicación de tecnologías avanzadas de seguridad, privacidad y encriptación, aspectos en los que PySyft destaca, estos no han cobrado especial peso al no ser objetivos prioritarios de este trabajo.

Los otros tres *frameworks* se han descartado por los siguientes motivos:

- Fed-BioMed: orientado específicamente al entorno biomédico, con soporte de privacidad y modelos médicos. Se ha descartado por su rigidez fuera de ese contexto y su complejidad en entornos genéricos.

- NVIDIA FLARE: de capacidades avanzadas y con buen mantenimiento. Sin embargo, cuenta con una curva de aprendizaje considerable y está mayoritariamente orientado a soluciones empresariales.
- TensorFlow Federated: potente pero limitado principalmente a TensorFlow, lo que restringe la interoperabilidad con PyTorch y otros frameworks requeridos para este trabajo.

II. Estructura y dependencias del producto

Estructura del producto

El sistema se ha implementado de forma modularizada para mayor legibilidad, reusabilidad y cohesión, y menor acoplamiento, con el fin de mejorar su mantenimiento futuro. La estructura principal de carpetas y archivos del código del producto es la siguiente:

- */aggregated_models*: carpeta en la que se guardarán los pesos de los modelos agregados por el servidor.
- */data*: carpeta en la que se guardarán los datos locales de cada cliente y del nodo centralizado.
- */logs*: carpeta en la que se guardarán los logs, los resultados de los experimentos y otros ficheros temporales.
- */plots*: carpeta en la que se guardarán los gráficos generados, tanto para visualizar propiedades de los datos de entrada como para ver los resultados de los procesos de aprendizaje.
- *centralized.py*: código para el aprendizaje centralizado.
- *client_app.py*: código para los clientes del AF; común entre todos.
- *client_1.py*, *client_2.py*, etc.: código para lanzar las diferentes instancias de cada cliente, cada una con su identificador. Ejecutarán *client_app.py*.
- *client_manager.py*: código para gestionar automáticamente la ejecución de clientes, proveyendo una abstracción mayor al usuario, incluyendo el lanzamiento de clientes de forma automática, el estado de los clientes en ejecución y la detención de los estos.
- *model.py*: código para el modelo de red neuronal a utilizar por los clientes, el servidor y el nodo centralizado.
- *strategy.py*: código para la estrategia de agregación usada por el servidor del AF.
- *server.py*: código para el servidor del AF.
- *task.py*: código para funcionalidades misceláneas como los logs, las gráficas o el contexto.

- *config.yaml*: archivo de configuración con el contexto y los hiperparámetros a usar por los clientes, el servidor y el nodo centralizado.
- *experiment.py*: código para la ejecución sistemática y automatizada de pruebas de AF.

Dependencias externas del producto

La implementación ha sido desarrollada usando como lenguaje de programación Python 3.10.11, asegurando compatibilidad con las bibliotecas más actuales a la vez que la estabilidad de las dependencias. A continuación, se detallan las principales dependencias utilizadas, junto con su propósito principal:

- Flower (1.18.0): *framework* de AF utilizado.
- Matplotlib (3.6.2): para la representación de gráficas.
- Numpy (1.26.4): para operaciones numéricas de bajo nivel y gestión de *arrays*.
- Pandas (2.2.3): para la gestión de datos en estructuras de tipo DataFrame.
- PyYAML (6.0.2): para la gestión de archivos en formato YAML en Python.
- Scikit-learn (1.6.1): para el cálculo de métricas y la validación cruzada.
- Seaborn (0.13.2): para visualizaciones estadísticas avanzadas.
- PyTorch (2.7.0): para la construcción, entrenamiento y evaluación de redes neuronales.
- TorchMetrics (1.7.0): para gestionar métricas de PyTorch.

III. Datos

Para el trabajo se han utilizado los datos de una variación de la base de datos proporcionada por el desafío PI-CAI (*Prostate Imaging: Cancer AI*), un reto internacional centrado en el desarrollo de algoritmos de IA para el diagnóstico automatizado del cáncer de próstata clínicamente significativo (csPCa) mediante resonancia magnética biparamétrica (bpMRI) [35].

La versión pública de la base de datos contiene cerca de 1.500 estudios de provenientes de múltiples centros médicos y fabricantes de equipos de imagen, lo que garantiza una alta diversidad de casos clínicos. Para su uso en este trabajo, se han mantenido para cada registro los siguientes campos: el id del estudio, la edad del paciente, el antígeno prostático específico (PSA), el volumen de la próstata y el diagnóstico de csPCa.

Dentro ya del ámbito de este TFG, se ha dividido el conjunto de datos en diferentes subconjuntos para repartir entre los clientes. En el caso del nodo centralizado se sigue usando el conjunto completo. Previo a la carga de los datos por el nodo respectivo, se aplica un ligero proceso de preprocesamiento, destacando la transformación de valores booleanos a numéricos, la gestión de otros valores no numéricos (NaN) y la reordenación aleatoria de los datos.

Durante la carga de los datos, también se realiza un breve análisis estadístico con el objetivo de comprender mejor la distribución y las relaciones entre las variables. Para ello, se han utilizado las bibliotecas Seaborn y Matplotlib, tan populares como sencillas de usar.

La primera visualización generada es un diagrama de cajas (*boxplot*), que permite observar la distribución de cada variable en términos de su rango, valores extremos, rango intercuartílico, mediana y valores atípicos.

La segunda visualización corresponde a una matriz de correlación, en la que se muestran los coeficientes de correlación entre todas las variables numéricas del conjunto de datos. Esta matriz es útil para identificar relaciones lineales fuertes o redundancias de significado, lo que puede influir en el diseño del modelo y en el proceso de selección de variables.

En la Figura 9 se puede observar un ejemplo de las dos gráficas.

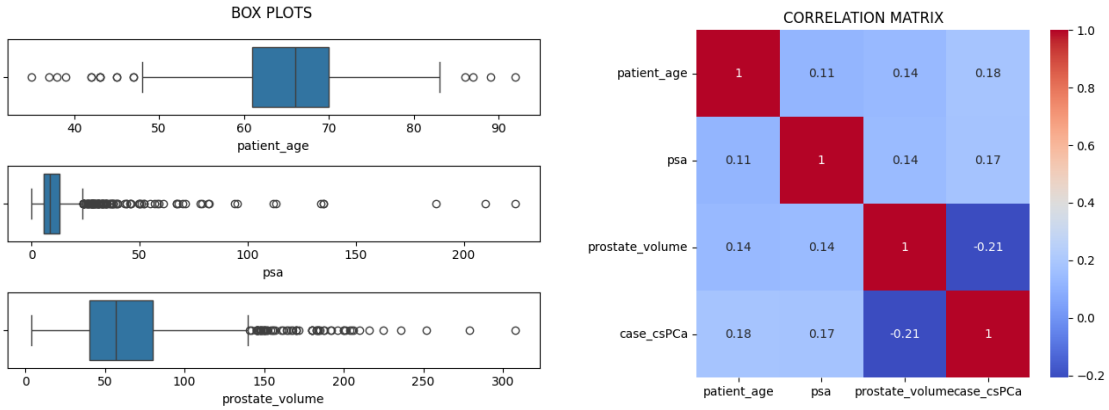


Figura 9: Diagrama de cajas y matriz de correlación de los datos iniciales

IV. Métricas de rendimiento

Para evaluar el rendimiento del aprendizaje, tanto federado como centralizado, y comprender su comportamiento, se han empleado las siguientes métricas:

- **Exactitud (*accuracy*):** proporción de casos correctamente clasificados frente al total de casos clasificados.

$$exactitud = \frac{predicciones\ correctas}{predicciones\ totales}$$

- **Precisión (*precision*):** proporción de casos correctamente clasificados como positivos frente al total de casos clasificados como positivos.

$$precisión = \frac{positivos\ correctos}{positivos\ clasificados}$$

  En otras palabras, de entre los casos predichos como positivos, cuántos casos eran de verdad positivos.

- **Exhaustividad (*recall*):** proporción de casos correctamente clasificados como positivos frente al total real de casos positivos.

$$exhaustividad = \frac{positivos\ correctos}{positivos\ reales}$$

  En otras palabras, de entre los casos positivos de verdad, cuántos casos se han identificado.

- **Puntuación F1 (*F1 score*):** media armónica entre la precisión y la exhaustividad.

$$F1 = \frac{2 \cdot precisión \cdot exhaustividad}{precisión + exhaustividad}$$

- **Exactitud equilibrada (*balanced accuracy score*):** variación de la exactitud normal pero ajustada a conjuntos de datos no equilibrados. Cuanto mayor sea la diferencia de la exactitud equilibrada con respecto a la exactitud normal, más desequilibrado estarán los datos.

- **Coeficiente de correlación de Matthews (MCC):** indica la aleatoriedad de las predicciones para un conjunto de datos desequilibrado. El rango del coeficiente se encuentra entre los valores -1 y 1. Un valor de 0 indicaría una aleatoriedad completa en las predicciones. Cuanto más se alejen los valores de 0, menor es la presencia de aleatoriedad en las predicciones, llegando a indicar ambos extremos, -1 y 1, la ausencia total de aleatoriedad. Los valores negativos apuntan a predicciones incorrectas, y los positivos, a predicciones correctas.

V. Modelo de red neuronal

Para el modelo de aprendizaje se utiliza una red neuronal completamente conectada (*feedforward*) diseñada de forma flexible. Su implementación se llevó a cabo mediante PyTorch, y permite parametrizar tanto el número como el tamaño de las capas ocultas, lo que facilita la experimentación con diferentes arquitecturas.

El modelo actualmente usado representa un equilibrio entre rendimiento y simplicidad, con una arquitectura robusta frente a distintas configuraciones y conjuntos de datos. Si bien la dedicación a pulir los hiperparámetros del modelo se vio limitada por el enfoque del trabajo, las decisiones no se han tomado de forma arbitraria, sino que son producto de experimentación y estudio.

La red neuronal implementada sigue una arquitectura *feedforward* compuesta por una capa de entrada, varias capas ocultas y una capa de salida. Todas emplean una transformación lineal (Lineal). La capa de entrada y cada capa oculta emplea va seguida de una normalización por lotes (BatchNorm1d), una función de activación ReLU y una capa de *dropout* como mecanismo de regularización.

Para la inicialización de los pesos, las capas ocultas usan *kaiming_uniform_*, recomendada para funciones de activación ReLU [36], mientras que la capa de salida utiliza xavier_uniform_.

El modelo se entrena con un tamaño de lote (*batch size*) de 16, una tasa de aprendizaje de 0.002, el optimizador Adam y la función de pérdida *BCEWithLogitsLoss*, ideal para clasificación binaria.

En la Figura 10, de la página siguiente, se puede observar la implementación del modelo de red neuronal.

```python
class NeuralNetwork(nn.Module):

    def __init__(self, input_size: int, hidden_sizes: List[int], output_size: int, dropout) -> None:
        super(NeuralNetwork, self).__init__()

        layers = []
        layers.append(nn.Linear(input_size, hidden_sizes[0]))

        init.kaiming_uniform_(layers[-1].weight, nonlinearity='relu')
        layers.append(nn.BatchNorm1d(hidden_sizes[0]))
        layers.append(nn.ReLU())
        layers.append(nn.Dropout(dropout))

        for in_size, out_size in zip(hidden_sizes[:-1], hidden_sizes[1:]):
            layers.append(nn.Linear(in_size, out_size))

            init.kaiming_uniform_(layers[-1].weight, nonlinearity='relu')
            layers.append(nn.BatchNorm1d(out_size))
            layers.append(nn.ReLU())
            layers.append(nn.Dropout(0.1))

        layers.append(nn.Linear(hidden_sizes[-1], output_size))
        init.xavier_uniform_(layers[-1].weight)

        self.model = nn.Sequential(*layers)


    def forward(self, x: torch.Tensor) -> torch.Tensor:
        return self.model(x)
```

Figura 10: Código de modelo de red neuronal

En versiones anteriores del modelo, la estructura de las capas ocultas de la red neuronal estaba definida de forma rígida y manual, por lo que cualquier cambio que se desease realizar, como tan solo añadir una capa más o variar el número de neuronas, requería modificar manualmente el modelo, lo que dificultaba la experimentación en gran medida. Para solventar este problema, se ha implementado una arquitectura completamente dinámica y parametrizada: basta con especificar una lista de tamaños (por ejemplo, '*[64, 32]*') y la red se construye automáticamente añadiendo tantas capas como elementos haya en la lista, y con el tamaño indicado para cada una.

Durante el diseño de la red neuronal se han realizado numerosos experimentos con distintas variantes del modelo con el objetivo de mejorar su rendimiento y su estabilidad. Un caso destacable ha sido la prueba con LeakyReLU en lugar de ReLU para mitigar el riesgo de neuronas "muertas" (con salidas cero constantes). Sin embargo, las mejoras observadas no han sido consistentes, por lo que se ha mantenido ReLU por su simplicidad y velocidad de convergencia.

VI. Clientes

En el código del cliente se cargan los datos de cada cliente, se inicializa un modelo, se instancia una clase de cliente, se definen los procesos de entrenamiento y de validación del modelo y se establece la conexión con el servidor. Con todo esto, el *framework* puede orquestar el proceso de AF, entrenando y actualizando el modelo a lo largo de las diferentes rondas de aprendizaje. Se ha utilizado PyTorch como herramienta de entrenamiento de los modelos.

Para facilitar la reutilización del código y permitir la configuración específica de cada cliente, se ha adoptado una estructura modular:

- Un archivo común define la clase base del cliente, que implementa la lógica compartida por todos.
- Cada cliente específico se define en un archivo separado, desde el cual se crea su propia instancia de la clase común, permitiendo un identificador único y un contexto personalizado.

Flujo de ejecución

El flujo de ejecución del cliente es el siguiente:

1. Carga de datos: se cargan los datos locales del cliente.
2. Preparación de datos:
    a. Se separan las características de entrada (*features*) y las etiquetas (*labels*).
    b. Los datos se dividen en subconjuntos de entrenamiento y validación según una proporción dada (por defecto, 80% para entrenamiento y 20% para validación).
3. Inicialización del modelo: se instancia el modelo local.
4. Instanciación del cliente: se crea el objeto cliente, que alberga el modelo y los datos locales y define el entrenamiento y la validación.
5. Conexión con el servidor y AF: el cliente se conecta al servidor usando Flower y comienza a participar en las rondas de entrenamiento federado.

Clase FlowerClient

La clase personalizada *FlowerClient* representa a cada cliente dentro del entorno federado. Esta clase extiende *NumPyClient* de Flower y debe sobrescribir cuatro métodos clave para que el *framework* pueda interactuar con ella:

- *get_parameters():* devuelve los parámetros actuales del modelo.
- *set_parameters()*: actualiza los parámetros del modelo.
- *fit():* entrena el modelo localmente utilizando los datos de entrenamiento.
- *evaluate()*: evalúa el aprendizaje del modelo utilizando los datos de validación y devuelve las métricas correspondientes.

Entrenamiento

El entrenamiento local se realiza mediante la función *train*, que sigue los pasos típicos de un bucle de entrenamiento en PyTorch:

1. Se define el optimizador (Adam) y la función de pérdida (BCEWithLogitsLoss).
2. Para cada época:
    a. Se inicializan métricas de pérdida y precisión.
    b. Se recorren los lotes del conjunto de entrenamiento.
    c. Se actualizan los pesos mediante retropropagación.

Además, se realiza una evaluación simple durante el entrenamiento con los propios datos de entrenamiento, lo que permite analizar la calidad del aprendizaje de una perspectiva diferente a la que aporta la evaluación con los datos de validación. Solo se han estudiado la exactitud (*accuracy*) y la pérdida.

Validación

Tras el entrenamiento, el cliente evalúa el modelo con el conjunto de datos de validación utilizando la función *test*. El proceso es, a rasgos generales, el siguiente:

1. Se desactiva el cálculo de gradientes para no afectar al modelo entrenado.

2. Se recorren los lotes del conjunto de validación:
    a. Se calcula la salida del modelo.
    b. Se aplica la función sigmoide y se binarizan las predicciones según un umbral.
3. Se calcula el promedio de las métricas de evaluación recolectadas durante la validación.

Conexión con servidor

Cada cliente se conecta al servidor mediante el *framework* Flower. Para ello, basta con instanciar la clase cliente (*FlowerClient* en este caso) y utilizar la dirección IP del servidor para establecer la comunicación. A partir de ese momento, Flower se encarga de orquestar las rondas de AF entre el servidor y los clientes.

VII. Servidor

El servidor se encarga de dar inicio y fin a las rondas de entrenamiento federado, orquestando la comunicación con los clientes y gestionando la lógica de agregación y distribución del modelo global. Se ha definido un archivo propio para el código de la implementación del servidor.

Flujo de ejecución

El flujo de ejecución del servidor es el siguiente:

1. Configuración del servidor: se configuran las características del servidor.
2. Inicialización del modelo: se instancia el modelo base sobre el que se irá realizando la agregación de los modelos locales para formar el modelo global. Si existen datos guardados de pesos del modelo, se cargan.
3. Definición de la estrategia: se instancia el modelo local.
4. Conexión con los clientes y AF: el servidor se inicializa mediante Flower, a espera de clientes. Cuando se dan las condiciones, comienza a orquestar las rondas de entrenamiento federado, encargándose de agregar los modelos locales recibidos y distribuir el modelo global agregado.

El *framework* Flower se encarga de por sí de gran parte de la gestión de la coordinación entre servidor y clientes, por lo que la implementación del servidor ha sido simple, siendo los aspectos más destacables la configuración del servidor y la inicialización del modelo.

Se ha incluido la posibilidad de guardar y cargar pesos del modelo. Para lograrlo se ha tenido que realizar una implementación personalizada de la estrategia a usar, FedProx en este caso. Se detalla en la siguiente sección, *VIII. Guardado y carga de modelos*.

Estrategia

El servidor utiliza la estrategia FedProx, una extensión de FedAvg que introduce un término de regularización para controlar las desviaciones de los modelos locales respecto al modelo global. Esto se ha mostrado útil en escenarios donde los datos de los clientes

presentan alta heterogeneidad, mejorando la estabilidad y la convergencia del proceso de entrenamiento [37].

El rendimiento de cualquier estrategia de agregación, como el de tantos otros componentes en un sistema de aprendizaje, está fuertemente condicionado por el resto de los hiperparámetros (como la arquitectura de la red, la tasa de aprendizaje o el número de clientes) y por las características concretas de los datos.

Por este motivo, evaluar una estrategia de forma aislada resulta complicado; una misma estrategia puede mostrar un rendimiento muy diferente bajo pequeñas variaciones del entorno de entrenamiento. En este proyecto se ha intentado realizar pruebas imparciales, pero aun así los resultados deben interpretarse con cautela, sin considerarse excesivamente concluyentes.

Además de FedProx, se ha experimentado con varias otras estrategias como FedAdagrad [38] [39] [40], la cual adapta la tasa de aprendizaje en función de la magnitud del gradiente. Sin embargo, en las pruebas realizadas, sus resultados no han ofrecido mejoras significativas o concluyentes frente a FedAvg. Por el contrario, FedProx sí ha aportado ligeras mejoras en la exactitud y otras métricas, y una reducción estable de la pérdida, por lo que finalmente se ha optado por utilizar esta estrategia.

VIII. Guardado y carga de modelos

En cualquier proceso de aprendizaje automático es fundamental preservar el conocimiento adquirido. Por ello, el sistema desarrollado incluye mecanismos para guardar los pesos del modelo una vez entrenado, así como cargarlos posteriormente para continuar el entrenamiento desde el último punto guardado o para evaluarlo.

Guardado del modelo

El framework Flower, utilizado para el AF, no proporciona por defecto métodos para guardar los pesos del modelo, por lo que se ha implementado esta funcionalidad utilizando directamente las capacidades de PyTorch.

Para poder guardar los modelos después de cada ronda de agregación se ha tenido que crear una estrategia de agregación personalizada, FexProxSaveModel, que extiende la implementación por defecto de FedProx. Esta clase redefine el método *aggregate_fit*, el cual es llamado durante la agregación de cada ronda, y ahora extrae los pesos del modelo actual y los guarda en un archivo de extensión PyTorch (.pt).

Carga del modelo

Para continuar desde el modelo más reciente, también se ha implementado la carga automática de pesos al inicio de una nueva ejecución. El sistema busca el último archivo de guardado y, en caso de encontrarlo, lo carga en el modelo. Por elección, el sistema elimina los archivos .pt tras cargar el modelo, pero esta lógica puede modificarse fácilmente para conservar los historiales si se desea.

Consideraciones técnicas

PyTorch ofrece dos enfoques principales para guardar y cargar modelos [41] [42]:

1. Guardar y cargar el modelo completo: esta opción guarda tanto la arquitectura como los pesos. Limita la aplicación del modelo al requerir

tener exactamente la misma definición del modelo en el entorno donde se carga.

El modelo se guarda con:

torch.save(the_model, PATH)

Y se carga con:

torch.load(PATH)

2. Guardar y cargar solo los pesos: permite mayor flexibilidad al no guardar la arquitectura del modelo, permitiendo que los pesos pueden cargarse en cualquier modelo compatible con la misma arquitectura.

   El modelo se guarda con:

   torch.save(the_model.state_dict(), PATH)

   Y se carga con:

   the_model.load_state_dict(torch.load(PATH))

La implementación actual sigue la segunda aproximación para asegurar la correcta agregación entre los diferentes clientes en casos de disparidad de modelos.

IX. Logs

Durante la ejecución de los programas se generan archivos de log que almacenan información detallada del progreso, incluyendo los pasos, tiempos, métricas y posibles errores. Esto permite auditar el comportamiento del sistema y analizar los resultados de forma ordenada.

Para la gestión de los logs se ha utilizado la biblioteca estándar *logging* de Python, que permite configurar diferentes niveles de severidad (info, warning, error, etc.) y estructuras de escritura. Se ha optado también por el uso de logs rotatorios mediante *RotatingFileHandler*, lo que permite limitar el tamaño máximo de cada archivo de log y continuar automáticamente en nuevos archivos cuando se alcanza dicho límite. Esto evita el crecimiento descontrolado de los archivos de registro y facilita su consulta posterior.

Cada cliente y el servidor instancian un *logger* propio, que es el objeto responsable de gestionar la escritura de mensajes en los archivos de log.

En la Figura 11 se puede observar un fragmento de un log de un cliente. En la Figura 12, de la página siguiente, un fragmento de un log del servidor.

```
2025-06-15 20:32:36,620 - INFO:  Starting FL client...
2025-06-15 20:32:36,622 - INFO:  Running in Production mode (non-deterministic).
2025-06-15 20:32:36,623 - INFO:  Loading data from data\PI-CAI_3__part1.xlsx
2025-06-15 20:32:36,830 - INFO:  Data loaded. Shape: (360, 5)
2025-06-15 20:32:36,842 - INFO:  Data preprocessing completed. Final shape: (360, 4)
2025-06-15 20:32:37,516 - INFO:  Client initialized.
2025-06-15 20:32:37,538 - INFO:
2025-06-15 20:32:37,539 - INFO:  === [NEW TRAINING ROUND] ===
2025-06-15 20:32:37,539 - INFO:  Starting local training...
2025-06-15 20:32:37,539 - INFO:  Updating model parameters...
2025-06-15 20:32:37,541 - INFO:  Starting local model training...
2025-06-15 20:32:40,100 - INFO:  Hyperparameters - LR: 0.000100, Batch Size: 16, Epochs: 10
2025-06-15 20:32:40,101 - INFO:  Starting training for 10 epochs...
2025-06-15 20:32:40,166 - INFO:  Epoch 1/10 -- Loss: 0.7723, Accuracy: 0.6875
2025-06-15 20:32:40,244 - INFO:  Epoch 2/10 -- Loss: 0.6852, Accuracy: 0.6979
2025-06-15 20:32:40,315 - INFO:  Epoch 3/10 -- Loss: 0.6373, Accuracy: 0.7049
2025-06-15 20:32:40,389 - INFO:  Epoch 4/10 -- Loss: 0.6332, Accuracy: 0.6979
2025-06-15 20:32:40,457 - INFO:  Epoch 5/10 -- Loss: 0.6046, Accuracy: 0.7118
2025-06-15 20:32:40,528 - INFO:  Epoch 6/10 -- Loss: 0.6016, Accuracy: 0.7049
2025-06-15 20:32:40,590 - INFO:  Epoch 7/10 -- Loss: 0.6048, Accuracy: 0.6944
2025-06-15 20:32:40,651 - INFO:  Epoch 8/10 -- Loss: 0.5745, Accuracy: 0.7049
2025-06-15 20:32:40,716 - INFO:  Epoch 9/10 -- Loss: 0.5500, Accuracy: 0.7049
2025-06-15 20:32:40,786 - INFO:  Epoch 10/10 -- Loss: 0.5593, Accuracy: 0.7292
2025-06-15 20:32:40,786 - INFO:  Local training complete.
2025-06-15 20:32:40,787 - INFO:  Fetching model parameters...
2025-06-15 20:32:40,807 - INFO:  === [VALIDATION REPORT] ===
2025-06-15 20:32:40,808 - INFO:  Starting local model validation...
2025-06-15 20:32:40,808 - INFO:  Updating model parameters...
2025-06-15 20:32:40,810 - INFO:  Using threshold 0.40 for binarization.
2025-06-15 20:32:40,815 - INFO:  Testing loss: 0.5979
2025-06-15 20:32:40,830 - INFO:  Testing metrics -- Accuracy: 0.708, Precision: 0.615, Recall: 0.593, F1 s
2025-06-15 20:32:40,840 - INFO:
2025-06-15 20:32:40,840 - INFO:  === [NEW TRAINING ROUND] ===
```

Figura 11: Fragmento de log de cliente

```
2025-06-15 20:11:53,428 - INFO:  Starting FL server...
2025-06-15 20:11:53,436 - INFO:  Loaded global model from the latest checkpoint: aggregated_models\server_model_weights_r5.pt
2025-06-15 20:11:53,437 - INFO:  Initial model parameter means: {'model.0.weight': -0.014902, 'model.0.bias': 0.009067, 'model.1
2025-06-15 20:11:53,437 - INFO:  Deleted old checkpoint: aggregated_models\server_model_weights_r5.pt
2025-06-15 20:11:53,438 - INFO:  Deleted old checkpoint: aggregated_models\server_model_weights_r4.pt
2025-06-15 20:11:53,438 - INFO:  Deleted old checkpoint: aggregated_models\server_model_weights_r3.pt
2025-06-15 20:11:53,438 - INFO:  Deleted old checkpoint: aggregated_models\server_model_weights_r2.pt
2025-06-15 20:11:53,439 - INFO:  Deleted old checkpoint: aggregated_models\server_model_weights_r1.pt
2025-06-15 20:11:53,439 - INFO:  Server configuration complete. Listening on 192.168.18.12:8081
2025-06-15 20:11:54,587 - INFO:
2025-06-15 20:11:54,587 - INFO:  [ROUND 1]
2025-06-15 20:12:10,685 - INFO:  Aggregated parameter means: {'model.0.weight': 0.003329, 'model.0.bias': -0.038015, 'model.1.we
2025-06-15 20:12:10,688 - INFO:  Saved aggregated global model after round 1 at aggregated_models\server_model_weights_r1.pt.
2025-06-15 20:12:10,718 - INFO:  Round metrics -- Accuracy: 0.771, Precision: 0.523, Recall: 0.566, F1 score: 0.543, Balanced ac
2025-06-15 20:12:10,718 - INFO:
2025-06-15 20:12:10,718 - INFO:  [ROUND 2]
```

Figura 12: Fragmento de log de servidor

X. Parametrización y contexto

Tanto el sistema de AF como centralizado cuentan con un alto número de parámetros que permiten configurar la arquitectura del modelo, controlar el flujo del entrenamiento, las conexiones, la evaluación y otros aspectos críticos. Con el objetivo de facilitar esta configuración y el mantenimiento y extensión del sistema, se ha optado por un enfoque completamente parametrizado, basado en un archivo externo de configuración y una estructura de objetos en los que agrupar y almacenar estos parámetros a la que se le ha denominado "contexto".

Archivo de configuración

Para la extensión del archivo de configuración se han evaluado cuatro alternativas: un texto plano, XML, JSON y YAML. El formato XML ha sido descartado por su dificultad de escritura y legibilidad. De igual forma, el texto plano se ha rechazado debido a la poca flexibilidad y programabilidad que ofrecía. La decisión final se redujo a JSON o YAML, y, si bien JSON era una excelente opción, se ha optado por YAML por los siguientes motivos [43]:

- Ofrece una mejor legibilidad y simplicidad sintáctica, especialmente en configuraciones de baja o media complejidad, como las de este trabajo.
- Permite una mayor complejidad y potencia en caso de ser necesaria en un futuro.
- Tiene mayor compatibilidad con herramientas de control de versiones, siendo compatible con el *diff* de Git, al contrario que JSON.
- Permite comentarios.

Algunos de los parámetros incluidos en el archivo de configuración YAML son la arquitectura de la red neuronal, la tasa de aprendizaje (*learning rate*), el número de rondas de AF y el tamaño de los lotes (*batch size*) o el número mínimo de clientes para comenzar una ronda de aprendizaje.

Contexto

Además de los hiperparámetros, existen ciertos objetos y valores que resultan útiles durante múltiples fases del ciclo de entrenamiento y evaluación, como por ejemplo el identificador del cliente, el registrador de eventos (*logger*) o las métricas acumuladas. Para evitar tener que pasar cada uno de estos objetos de forma individual entre funciones, se ha diseñado un sistema centralizado denominado contexto, que agrupa y encapsula toda esta información de manera coherente y reutilizable.

El sistema de contexto está compuesto por cinco clases principales:

- HyperParameters: contiene los hiperparámetros del archivo de configuración.
- RandomState: permite controlar la reproducibilidad mediante semillas y banderas de aleatoriedad en cargas de datos y particiones.
- MetricsTracker: almacena de forma ordenada las métricas de entrenamiento y validación a lo largo del proceso.
- ClientContext: contiene la información relevante para un cliente, agregando las tres anteriores.
- ServerContext: contiene la información relevante para el servidor.

El uso del contexto no solo reduce la complejidad del código y su repetición, sino que, además, garantiza coherencia entre módulos y facilita la extensión del sistema, ya que nuevos parámetros o componentes pueden añadirse al contexto sin modificar el resto de las funciones del sistema.

XI. Ejecución del sistema de aprendizaje federado

Modos de ejecución

Inicialmente, el sistema se desarrolló utilizando el modo tradicional (*legacy*) de Flower, basado en funciones como *start_server* y *start_client* para lanzar, respectivamente, el servidor y los clientes. Este enfoque, aunque considerado obsoleto desde Flower v1.7, sigue siendo oficialmente funcional por razones de retrocompatibilidad. La ejecución se realiza lanzando el archivo directamente mediante el intérprete de Python; por ejemplo, '*python server.py*'.

Este método ofrece simplicidad y control manual de parámetros como la dirección IP y la gestión manual del contexto de la ejecución.

Sin embargo, con el objetivo de modernizar la ejecución, se ha estudiado el nuevo sistema de Flower basado en los conceptos *superlink* (para servidores) y *supernode* (para clientes):

- Superlink: representa un servidor, instanciado con *ServerApp*.
- Supernode: representa un cliente federado, instanciado con *ClientApp*.

La Figura 13 muestra una plantilla para los comandos de lanzamiento publicada por Flower en su repositorio de GitHub con la salida de la versión 1.13 [44].

```
# Start a secure SuperLink
$ flower-superlink \
    --ssl-ca-certfile <your-ca-cert-filepath> \
    --ssl-certfile <your-server-cert-filepath> \
    --ssl-keyfile <your-privatekey-filepath>

# In a new terminal window, start a long-running SuperNode
$ flower-supernode \
     --superlink 127.0.0.1:9092 \
     --clientappio-api-address 127.0.0.1:9094 \
     --root-certificates <your-ca-cert-filepath> \
     <other-args>

# In another terminal window, start another long-running SuperNode (at least 2 SuperNodes are required)
$ flower-supernode \
     --superlink 127.0.0.1:9092 \
     --clientappio-api-address 127.0.0.1:9095 \
     --root-certificates <your-ca-cert-filepath> \
     <other-args>
```

Figura 13: Ejemplos de comandos de superlink y supernodes de Flower

Este nuevo modelo está diseñado pensando en el despliegue en entornos de producción, permitiendo una integración más fluida con herramientas de orquestación, contenedores, o despliegues distribuidos. Sin embargo, durante el desarrollo se han identificado importantes limitaciones prácticas para el alcance de este trabajo, las cuales han acabo motivando la decisión de mantener el enfoque tradicional. En concreto:

- Documentación insuficiente: la documentación del nuevo sistema es todavía confusa, dispersa y escasa.
- Rigidez en el despliegue: el nuevo modelo favorece despliegues estandarizados, pero a expensas de la flexibilidad que ofrece el enfoque tradicional, como la en la definición de la dirección IP.
- Complejidad innecesaria: las nuevas funcionalidades y herramientas que provee y fuerza este nuevo enfoque, más allá de los propios comandos, no aportan beneficios al enfoque y alcance de este trabajo, sino que, en realidad, aumentan la complejidad innecesariamente.

A pesar del gran potencial de esta nueva estrategia de ejecución, se ha optado por mantener el enfoque tradicional. Flower sigue evolucionando para lograr proporcionar arquitecturas cada vez más modernas y escalables, y es posible que en el futuro transitar sea más fácil y beneficioso.

Client Manager: automatización de clientes

Para evitar tener que definir un archivo propio para cada cliente y lanzar manualmente cada uno de forma independiente, se ha implementado una herramienta auxiliar denominada *Client Manager*, cuyo propósito es facilitar la gestión de múltiples clientes.

El *Client Manager* permite tres funcionalidades:

- Lanzar automáticamente el número de clientes deseado.
- Listar los clientes en ejecución activa.
- Detener la ejecución de los clientes activos.

Además, se ha integrado una interfaz de línea de comandos (*CLI*) sencilla, utilizando la biblioteca estándar *argparse*, lo que permite lanzar el código con diferentes opciones de manera flexible. La opción '*--help*' proporciona un resumen claro de los comandos y parámetros disponibles, como se puede observar en la Figura 14.

```
PS C:\Users\rafae\git_repos_my_own\TFG\Federated_learning\fl-four-clients> python client_manager.py --help
usage: client_manager.py [-h] {start,stop,list} ...

description: manage federated clients (start/stop/list)

positional arguments:
  {start,stop,list}
    start            Start the specified number of clients (start <NUM_CLIENTS>)
    stop             Stop all clients
    list             List running clients

options:
  -h, --help         show this help message and exit
```

Figura 14: Ayuda de CLI de Client Manager

XII. Experimentos

Con el fin de automatizar y organizar la ejecución de múltiples pruebas de AF, se ha desarrollado un archivo dedicado exclusivamente a la gestión de experimentos. Este *script* permite definir los parámetros clave del experimento, como el número de ejecuciones o la cantidad de clientes, y se encarga de lanzar automáticamente tanto el servidor como los clientes en cada ejecución.

Durante cada ejecución, las métricas globales calculadas por el servidor tras cada ronda de AF son almacenadas temporalmente en un archivo JSON. Tras cada ejecución del experimento, se extraen y agregan progresivamente a un archivo CSV que consolida todos los resultados. Al finalizar el conjunto de ejecuciones, el sistema calcula de forma automática la media y la desviación típica de cada métrica, ofreciendo así una visión agregada del rendimiento del sistema. La Figura 15 muestra un ejemplo del archivo CSV generado.

run_id	Accuracy	Precision	Recall	F1 score	Balanced accuracy	MCC
1	74	52	4	45	63	29
2	7	65	37	46	62	3
3	72	43	57	49	66	3
avg	72	53	45	47	64	3
std	2	11	11	2	2	1

Figura 15: CSV de resultados de experimento

Interfaz de línea de comandos

Tal y como se hizo con client_manager.py, para facilitar su uso y parametrización, se ha implementado una pequeña interfaz de línea de comandos con la biblioteca argparse. En la Figura 16 se muestra el resumen provisto por la opción '*--help'*.

```
PS C:\Users\rafae\git_repos_my_own\TFG\Federated_learning\fl-four-clients> python experiment.py --help
usage: experiment.py [-h] [--test]

description: run Federated Learning experiments

options:
  -h, --help  show this help message and exit
  --test      Run in Test mode (some lightweight configs for non-deterministic reproducibility)
```

Figura 16: Ayuda de CLI de experiment.py

Semillas de aleatoriedad

Dado que ciertos componentes del sistema implican aleatoriedad, como la partición de los conjuntos de datos de entrenamiento y validación en los clientes, el sistema cuenta con la posibilidad de establecer semillas (*seeds en inglés*) para asegurar la reproducibilidad de los experimentos.

Cuando lanza un experimento con el argumento --test, se fijan semillas para los generadores aleatorios, de forma que las condiciones del experimento puedan repetirse exactamente.

XIII. Sistema centralizado

El segundo sistema desarrollado corresponde al aprendizaje centralizado. A diferencia del AF, no existen múltiples clientes distribuidos ni servidor que actúe como coordinador del entrenamiento, por lo que no se requiere ninguna arquitectura federada, estrategia de agregación ni *framework* especializado.

El entrenamiento se realiza de forma directa y secuencial sobre el conjunto de datos completo. El modelo se entrena localmente, y el flujo de trabajo es considerablemente más simple: carga de datos, partición (si aplica; ver siguiente sección *XIV. Validación cruzada*), inicialización del modelo, entrenamiento y validación. Además, no se emplea el concepto de rondas, ya que todo el aprendizaje ocurre en una única ejecución continua.

Esta sencillez otorga la ventaja significativa de poder reutilizar directamente la lógica desarrollada para el cliente federado. Las funciones de entrenamiento y validación, al igual que la gestión de hiperparámetros, registro de métricas y demás, han sido diseñados para ser reutilizables, lo que evita duplicación de código, facilita el mantenimiento y permite una comparación más controlada entre el enfoque federado y el centralizado. Gracias a esta estructura modular, no ha sido necesario implementar nuevos componentes y ha bastado con secuenciar adecuadamente el flujo de trabajo.

Sin embargo, a pesar de no ser necesario, sí se ha implementado una nueva funcionalidad: la validación cruzada. Se explica en detalle en la siguiente sección, *XIV. Validación cruzada*.

XIV. Validación cruzada

La validación cruzada (*cross-validation*) es una técnica comúnmente utilizada en aprendizaje automático supervisado para evaluar el aprendizaje de un modelo. A diferencia del enfoque tradicional (conocido como *hold-out*), el cual divide el conjunto de datos en dos partes fijas (una para entrenamiento y otra para validación), la validación cruzada propone una estrategia más robusta: dividir el conjunto completo en k particiones (folds), y realizar k entrenamientos distintos. En cada uno, se utiliza un fold diferente como conjunto de validación y el resto para entrenamiento [45]. El aprendizaje no se acumula de un entrenamiento a otro, por lo que se instancia un nuevo modelo para cada entrenamiento.

La validación cruzada cuenta con la ventaja de poder evaluar el modelo de forma más representativa con respecto al conjunto de datos, ya que todas las muestras son utilizadas tanto para entrenar como para validar en algún momento. Por esta razón, la validación cruzada es especialmente útil durante el ajuste de hiperparámetros (hyperparameter tuning) [19].

No obstante, la validación cruzada presenta serios conflictos con el AF:

- Pérdida del contexto*:* en validación cruzada, el modelo se entrena desde cero en cada partición. Esto entra en conflicto con la filosofía de FL, donde el modelo se entrena de forma acumulativa y progresiva a través de rondas, por lo que la evaluación no sería representativa del aprendizaje de cara a la agregación. Del mismo modo, tampoco reflejaría el rendimiento del aprendizaje sobre el modelo agregado.
- Coste computacional y temporal elevado: aplicar validación cruzada para un número k de *folds* implicaría multiplicar el tiempo de entrenamiento por k, y repetirlo en cada ronda de entrenamiento federado. Este sobrecoste es difícilmente viable en un entorno federado.

Ante estos hechos, se ha descartado implementar la validación cruzada como evaluación en el ciclo de AF. Sin embargo, la validación cruzada sigue siendo útil como técnica de validación centralizada de datos locales. Puesto que conceptualmente ya no está ligado

de forma alguna al AF, se ha implementado en el sistema como parte del aprendizaje centralizado.

Adicionalmente, para flexibilizar el proceso, se ha parametrizado el uso de validación cruzada en su implementación, siendo la alternativa la validación tradicional.

XV. Posibles mejoras no realizadas

Durante el desarrollo del sistema se han identificado varias mejoras potenciales que, al no considerarse no prioritarias, no han llegado a implementarse, quedando fuera del alcance del TFG. A continuación, se enumeran algunas de las más destacadas:

- Decadencia de tasa de aprendizaje (*learning rate decay*): el ajuste dinámico de la tasa de aprendizaje a lo largo del entrenamiento favorecería una mejor convergencia.
- Tiempo de espera dinámico: adaptar los tiempos de espera del servidor en función del comportamiento de los clientes permitiría una gestión más robusta ante retrasos o desconexiones.
- Parada temprana (*early stopping*): introducir un mecanismo para detener el entrenamiento automáticamente al detectar estancamiento en la mejora del rendimiento ayudaría a reducir los tiempos del proceso y evitar el sobreajuste.
- Validaciones: incorporar comprobaciones automáticas del formato y consistencia de los datos y parámetros contribuiría a mejorar la robustez y fiabilidad del sistema y a facilitar la depuración.

Éstas y otras mejoras quedan como posibles futuras ampliaciones del sistema actual.

8 – PRUEBAS COMPARATIVAS

El objetivo de esta fase es analizar el comportamiento del sistema de AF implementado y compararlo con un enfoque de entrenamiento centralizado. Se pretende evaluar la eficacia, estabilidad y robustez del sistema en distintos contextos de uso, incluyendo variaciones en el número de clientes, la distribución de los datos y el volumen de información disponible. Aunque las pruebas no pretenden tener una validez estadística estricta, sí buscan ofrecer una perspectiva sedimentada y razonada.

I. Entorno de pruebas común

Todas estas pruebas se han realizado con el conjunto de datos y el modelo especificado en el capítulo *6 – Implementación*, en las secciones *III. Datos* y *V. Modelo de red neuronal*, respectivamente. Se han mantenido los hiperparámetros para asegurar una comparación efectiva. Se resumen a continuación:

- Framework: PyTorch
- Optimización: Adam
- Función de pérdida: BCEWithLogitsLoss
- Tasa de aprendizaje: 0.002
- *Batch size*: 16
- Número de épocas (*epochs*): 150
- Umbral de binarización: 0.4
- Tamaño de entrada: 3
- Capas ocultas: [128, 128]
- Tamaño de salida: 1
- *Dropout*: 0.1
- Funciones de activación: ReLU
- Número de rondas (federado): 5

Los resultados se evalúan en términos de las métricas definidas en el capítulo *6 – Implementación*, en la sección *IV. Métricas de rendimiento*. Empleando sus nombres en inglés, éstas son: *accuracy*, *precision*, *recall*, *F1 score*. La pérdida (*loss*) también se ha considerado.

II. Escenarios de pruebas

A continuación, se describen los distintos escenarios evaluados, explicando brevemente las condiciones particulares y los resultados obtenidos:

Escenario 1: Centralizado completo

Entrenamiento tradicional, utilizando todos los datos en un único nodo central.

- *Accuracy*: 76 %
- *Precision*: 60 %
- *Recall*: 58 %
- *F1 Score*: 58 %
- *Balanced Accuracy*: 70 %
- *MCC*: 0.50

Este escenario presenta un buen resultado y muestra una muy buena estabilidad.

Escenario 2: Federado con 1 cliente

Simula un entrenamiento federado con un solo cliente que posee todos los datos. A nivel teórico es idéntico al escenario 1, pero muestra la influencia del entorno federado.

- *Accuracy*: 73 %
- *Precision*: 57 %
- *Recall*: 56 %
- *F1 Score*: 56 %
- *Balanced Accuracy*: 69 %

- *MCC*: 0.37

Los resultados son positivos, pero muestran un descenso notable respecto a la versión centralizada, debido a la agregación distribuida.

Escenario 3: Federado con 4 clientes (datos equilibrados)

Se distribuyen los datos equitativamente entre cuatro clientes, manteniendo una proporción similar de clases y de tamaño del conjunto de datos.

- *Accuracy*: 75 %
- *Precision*: 59 %
- *Recall*: 56 %
- *F1 Score*: 57 %
- *Balanced Accuracy*: 69 %
- *MCC*: 0.40

El rendimiento general es levemente inferior al caso centralizado, debido al proceso federado. Aun así, los resultados son consistentes, y el modelo parece aprender adecuadamente.

Escenario 4: Federado con 4 clientes (datos desequilibrados)

Los datos se distribuyen de forma desigual: algunos clientes reciben datos predominantemente de una clasificación, y el tamaño del conjunto de datos varía significativamente entre los clientes. Es un entorno interesante por su realismo.

- *Accuracy*: 73 %
- *Precision*: 53 %
- *Recall*: 51 %
- *F1 Score*: 51 %
- *Balanced Accuracy*: 65 %
- *MCC*: 0.37

Los resultados indican una ligera bajada en el rendimiento con respecto al escenario equilibrado.

Escenario 5: Centralizado (pocos datos)

Se entrena el modelo centralizado con solo el 15% del volumen de datos total. Se busca analizar la capacidad de aprendizaje en condiciones de escasez de datos.

- *Accuracy*: 70 %
- *Precision*: 54 %
- *Recall*: 55 %
- *F1 Score*: 54 %
- *Balanced Accuracy*: 68 %
- *MCC*: 0.32

La reducción de datos repercute negativamente en el rendimiento, como era de esperar. Se observa una mayor variabilidad entre épocas y menor estabilidad.

Escenario 6: Federado con 4 clientes (pocos datos)

Se entrena cada cliente con solo el 15% del volumen de sus datos disponible. Se busca analizar la capacidad de aprendizaje en condiciones de escasez de datos.

- *Accuracy*: 71 %
- *Precision*: 63 %
- *Recall*: 53 %
- *F1 Score*: 57 %
- *Balanced Accuracy*: 69 %
- *MCC*: 0.35

Los resultados han mostrado una disparidad considerable entre los diferentes experimentos, mayor que en el escenario 5. El rendimiento promedio no muestra un descenso notable, pero se han dado casos de un alta precisión y casos de resultados muy

pobres. La desviación estándar ha rondado el 10% para cada métrica. Este es el escenario más sensible debido a la baja cantidad de datos sumada a la distribución federada.

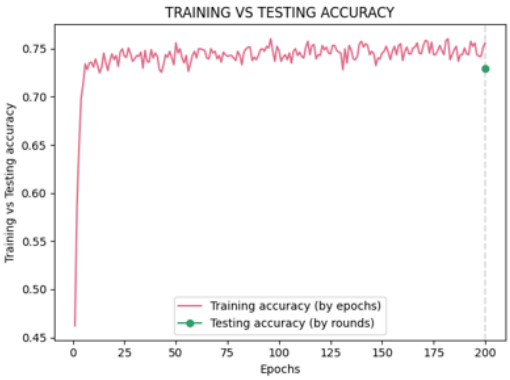
III. Conclusiones de las pruebas

La Tabla 6 resume los resultados promedios obtenidos en cada escenario.

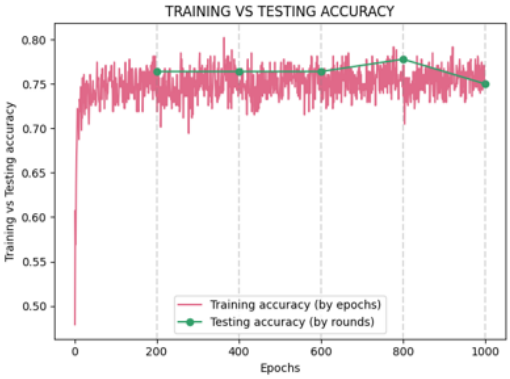
Tabla 6: Comparativa de resultados promedio de las pruebas

Escenario	***Accuracy***	***Precision***	***Recall***	***F1 Score***	***Balanced Accuracy***	***MCC***
Centralizado completo	76%	60%	58%	58%	70%	0.40
Federado con 1 cliente	73%	57%	56%	56%	69%	0.37
Federado con 4 clientes (datos equilibrados)	75%	59%	56%	57%	69%	0.40
Federado con 4 clientes (datos desequilibrados)	73%	53%	51%	51%	65%	0.37
Centralizado (pocos datos)	70%	54%	55%	54%	66%	0.32
Federado con 4 clientes (pocos datos)	71%	60%	55%	57%	69%	0.35

La Figura 17 muestra tres gráficas de rendimiento. La primera representa al aprendizaje centralizado. La segunda, refleja el AF con 4 clientes con datos equilibrados, en la que se puede observar una variabilidad ligeramente mayor. La tercera muestra la evolución de la pérdida (*loss*) en el escenario anterior. La pérdida siempre se ha mantenido estable en todos los



1. Accuracy - Centralizado



2. Accuracy - AF (datos equilibrados)

3. Accuracy - AF (datos equilibrados)

Figura 17: Gráficas de resultados de pruebas

Los resultados obtenidos reflejan un comportamiento coherente con lo esperado: el entrenamiento centralizado con todos los datos ofrece el mejor rendimiento global, pero el AF se muestra muy competitivo, especialmente cuando los datos están equilibradamente distribuidos entre los clientes.

El escenario con un solo cliente confirma que la arquitectura federada, aunque introduzca cierta sobrecarga, no afecta gravemente al rendimiento. Con cuatro clientes y datos bien repartidos, el modelo federado obtiene resultados muy cercanos al enfoque centralizado, lo que indica la solidez de su aplicabilidad práctica.

Los entornos desequilibrados y con escasez de datos revelan algunas de las limitaciones del aprendizaje: mayor variabilidad, sensibilidad a la distribución y descenso ocasional del rendimiento. Aun así, los resultados del aprendizaje federado y del aprendizaje centralizado han sido muy similares, manteniendo una capacidad de generalización aceptable.

En conjunto, el sistema de aprendizaje federado implementado ha demostrado claros indicativos de ser una alternativa viable, robusta y escalable. Con un ajuste estudiado de sus características y con técnicas adicionales de compensación o personalización, se puede esperar que su rendimiento puede ser mejorado.

9 – CONCLUSIONES

I. Objetivos y resultado

Este Trabajo de Fin de Grado se originó con dos objetivos. En primer lugar, la implementación de un sistema de AF que actuase como prototipo funcional para el proyecto ARTIFACTS. En segundo lugar, una comparación no formal pero real entre el AF y el aprendizaje centralizado. Concluido este trabajo, se puede confirmar que se han cumplido con los objetivos; el proyecto ha sido un éxito.

En relación con el primer objetivo, se ha implementado una arquitectura de AF funcional compuesta por un servidor y múltiples clientes capaces de participar en sesiones reales de entrenamiento distribuido. El sistema desarrollado no solo cumple con las funcionalidades mínimas necesarias, sino que además incorpora características adicionales como el guardado y carga de modelos y la parametrización flexible de las configuraciones de entrenamiento. El código ha sido diseñado con especial atención a la claridad, modularidad y extensibilidad, facilitando su uso, reutilización, ampliación y mantenimiento futuro.

También se ha priorizado la facilidad de uso. Se han creado scripts para simplificar la ejecución de pruebas y se ha desarrollado un gestor de clientes que permite operar el sistema sin necesidad de modificar el código directamente, lanzando, siguiendo y parando la ejecución de forma sencilla. Estas decisiones facilitan tanto el uso como la evaluación del sistema por parte de otras personas que desarrollen o investiguen.

Respecto al segundo objetivo, se ha implementado un sistema centralizado equivalente al federado y se han realizado diversas pruebas comparativas en distintos escenarios de distribución de datos. Aunque estas pruebas aún no tienen un carácter definitivo ni científica ni estadísticamente formal, los resultados muestran con claridad que el AF puede alcanzar niveles de precisión muy similares a los del enfoque centralizado. En particular, se ha observado que cuando los datos están razonablemente balanceados entre los nodos, las diferencias en el rendimiento del modelo son mínimas. Incluso en casos con cierta heterogeneidad en la distribución, el modelo federado ha mostrado una

capacidad notable para adaptarse y converger, lo que refuerza su viabilidad para contextos donde la centralización de datos no es posible.

El sistema de AF desarrollado constituye, por tanto, un ejemplo de una contribución real y tangible al AF, representando una prueba del potencial de la aplicación de esta tecnología en ámbitos sensibles como el sanitario. Este prototipo servirá como una base teórica sobre la que desarrollar futuras extensiones más complejas del proyecto ARTIFACTS y quedará público para servir de referencia a cualquiera que lo necesite. Con suerte, contribuirá a que ARTIFACTS logre sus ambiciosos objetivos científicos y su impacto social en la mejora de la atención médica basada en IA.

II. Limitaciones

Este TFG ha estado condicionado por varias limitaciones que conviene señalar. En primer lugar, el AF es una técnica reciente y en evolución. La escasez de documentación consolidada y de ejemplos prácticos completos unido al desconocimiento inicial del tema por parte del autor, han constituido la principal dificultad del trabajo.

En segundo lugar, el tiempo y alcance propio de un TFG han supuesto una limitación inherente. El tiempo y los recursos han limitado el alcance. Una parte importante del tiempo del proyecto ha estado dedicado al estudio previo. Las pruebas realizadas, si bien útiles como indicios, no tienen el rigor necesario para extraer conclusiones científicas definitivas. El objetivo ha sido ofrecer una demostración funcional y realizar una primera comparación exploratoria.

Por último, el hecho de que este trabajo se haya desarrollado en el contexto de un proyecto de la envergadura de ARTIFACTS y que se haya desarrollado dentro de una etapa inicial ha influido en la definición de los requisitos. Los requisitos del trabajo no estaban completamente definidos desde el inicio y, naturalmente, han ido evolucionando. Esto ha exigido una continua adaptación del enfoque y del desarrollo técnico a lo largo del proceso.

III. Futuro

Este trabajo sienta unas bases claras para su continuidad y evolución dentro del proyecto ARTIFACTS. El sistema desarrollado puede servir tanto como punto de partida para futuras versiones más avanzadas como referencia teórica útil para otras personas que desarrollen o investiguen. Su modularidad y claridad lo hacen adecuado para ser ampliado, adaptado a otros tipos de datos y probado en contextos más complejos o simplemente diferentes.

En términos generales, el futuro del AF es prometedor y responde a la necesidad crítica de preservar la privacidad de los datos, reduciendo además los costes y aumentar la velocidad. La comunidad científica y tecnológica está activamente involucrada en su evolución, especialmente en sectores sensibles como el sanitario.

El siguiente paso a seguir sería empezar a crear arquitecturas reales. Una línea especialmente interesante de investigación futura es la integración del AF con datos heterogéneos como los *Electronic Health Records* (EHR). También resulta de gran interés explorar la conectividad en tiempo real de los diferentes elementos del sistema, estudiando la asincronía de la comunicación [22].

Desde una perspectiva educativa, también se puede prever un impacto significativo. La creciente relevancia de la inteligencia artificial en todos los sectores está motivando la aparición de nuevas titulaciones, cursos y programas especializados. Es previsible que en los años venideros la comprensión de conceptos como el aprendizaje automático, la gestión de datos y la privacidad de los mismos se conviertan en conocimientos básicos y transversales independientemente del ámbito profesional. En este sentido, el AF representa un cambio de paradigma que la sociedad deberá asimilar: no solo entrenamos modelos con datos, sino que también debemos entender cómo hacerlo sin renunciar a la protección de nuestra propia información personal.

10 – REFERENCIAS

I. Referencias bibliográficas

[1] Zhu, H. (2021). *Communication Efficient and Secure Federated Learning*. University of Surrey, Department of Computer Science.

https://doi.org/10.15126/thesis.900230

[2] Zhong, H., Chen, D. y Jiang, X. (2023). *Building Trusted Federated Learning: Key Technologies and Challenges.* J. Sens. Actuator Netw. 2023, 12(1), 13.

https://www.mdpi.com/2224-2708/12/1/13

[3] Mäenpää, D., (2021). *Towards Peer-to-Peer Federated Learning: Algorithms and Comparisons to Centralized Federated Learning*. Linköping University, Department of Computer and Information Science. Recuperado el 08 de junio de 2025 de: https://liu.diva-portal.org/smash/get/diva2:1569479/FULLTEXT01.pdf

[4] Konečný, J., McMahan, H. y Ramage, D. (2015). *Federated Optimization: Distributed Optimization Beyond the Datacenter.* arXiv:1511.03575.

https://doi.org/10.48550/arXiv.1511.03575

[5] Agencia Española de Protección de Datos (AEPD) (2023). *Federated Learning: Inteligencia Artificial sin comprometer la privacidad.* https://www.aepd.es/ prensa-y-comunicacion/blog/federated-learning-inteligencia-artificial-sin-comprometer-la-privacidad

[6] Tan, Z., Yu, H., Cui, L. y Yang, Q. (2023). *Towards Personalized Federated Learning.* IEEE Transactions on Neural Networks and Learning Systems, vol. 34, núm. 12, pp. 9587-9603.

https://ieeexplore.ieee.org/document/9743558

[7] Letaief, K., Shi, Y., Lu, Jianmin y Lu, Jianhua (2024). *Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications*. IEEE Journal on Selected Areas in Communications, vol. 40, núm. 1, pp. 5-36.

https://ieeexplore.ieee.org/abstract/document/9606720

[8] McMahan, H., Moore, E., Ramage, D., Hampson, S. y Agüera y Arcas, B (2016). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. arXiv:1602.05629.

https://doi.org/10.48550/arXiv.1602.05629

[9] McMahan, H. y Ramage, D. (2017). *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. Google Research. https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/

[10] Chaudhary, R., Kumar, R. y Saxena, N. (2024). *A systematic review on federated learning system: a new paradigm to machine learning*. ResearchGate, Knowledge and Information Systems (2025) 67:1811–1914.

https://www.researchgate.net/publication/385709639_A_systematic_review_on_federated_learning_system_a_new_paradigm_to_machine_learning

[11] Li, L., Fan, Y., Tse, M. y Lin, K. (2020). *A review of applications in federated learning*. Computers & Industrial Engineering, vol. 149, 106854.

https://doi.org/10.1016/j.cie.2020.106854

[12] Ghimire, B. y Rawat, D. (2022). *Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things*. IEEE Internet of Things Journal, vol. 9, núm. 11, pp. 8229-8249.

https://ieeexplore.ieee.org/document/9709603

[13] Nguyen, D., Ding, M., Pathirana, P., Seneviratne, A., Li, J. y Vincent Poor, H. (2020). *Federated Learning for Internet of Things: A Comprehensive Survey*. IEEE communications surveys & tutorials, vol. 23, núm. 3, pp. 1622-1658.

https://ieeexplore.ieee.org/document/9415623

[14] Katyayani, M., Keshamoni, K., Rama Chandra Murty, A., Usha Rani, K., Reddy L., S. y Kumar Alapati, Y. (2025). Federated Learning: Advancements, Applications, and Future Directions for Collaborative Machine Learning in Distributed Environments. Journal of Electrical Systems.

https://doi.org/10.52783/jes.1900

[15] Luzón, M., Rodríguez-Barroso, N., Argente-Garrido, A., Jiménez-López, D., Moyano, J.M., Del Ser, J., Ding, W. y Herrera, F. (2024). A Tutorial on Federated Learning from Theory to Practice: Foundations, Software Frameworks, Exemplary Use Cases, and Selected Trends. IEEE/CAA Journal of Automatica Sinica, vol. 11, núm. 4, pp. 824-850.

https://ieeexplore.ieee.org/document/10475194

[16] Ma, X., Zhu, J., Lin, Z., Chen, S. y Qin, Y. (2022). A state-of-the-art survey on solving non-IID data in Federated Learning. Elsevier BV. Future Generation Computer Systems, vol. 135, pp. 244-258.

https://doi.org/10.1016/j.future.2022.05.003

[17] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D. y Chandra, V. (2022). *Federated Learning with Non-IID Data.* arXiv:1806.00582.

https://doi.org/10.48550/arXiv.1806.00582

[18] Ye, M., Fang, X., Du, B., Yuen, P. y Tao, D. (2023). *Heterogeneous Federated Learning: State-of-the-art and Research Challenge.* ACM computing surveys, 2024-03, vol.56 (3), pp.1-44, Article 79.

https://doi.org/10.1145/3625558

[19] Lyashenko, V. y Jha, A. (2025). Cross-Validation in Machine Learning: How to Do It Right. Neptune.ai. https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

[20] Chandorikar, K. (2020). *Introduction to Federated Learning and Privacy Preservation using PySyft and PyTorch*. OpenMinded. https://openmined.org/blog/federated-learning-additive-secret-sharing-pysyft/

[21] Braungardt, A. (2023). *Flower, FATE, PySyft & Co. — Federated Learning Frameworks in Python*. ELCA IT, Medium. https://medium.com/elca-it/flower-pysyft-co-federated-learning-frameworks-in-python-b1a8eda68b0d

[22] Darzidehkalani, E., Ghasemi-rad, M. y van Ooijen, P.M.A. (2022). *Federated Learning in Medical Imaging: Part I: Toward Multicentral Health Care Ecosystems*. JACR, vol. 19, núm. 8, pp. 969-974.

https://doi.org/10.1016/j.jacr.2022.03.015

[23] Antunes, R., da Costa, C., Küderle, A., Yari, I. y Eskofier, B. (2023). *Federated Learning for Healthcare: Systematic Review and Architecture Proposal*. ACM Transactions on Intelligent Systems and Technology, vol. 13, núm. 4, pp. 1-23.

https://dl.acm.org/doi/10.1145/3501813

[24] Li, H., Li, C., Wang, J., Yang, A., Ma, Z., Zhang, Z. y Hua, D. (2023). *Review on security of federated learning and its application in healthcare*. ScienceDirect, vol.144, pp. 271-290.

https://doi.org/10.1016/j.future.2023.02.021

[25] Mouhni, N., Elkalay, A., Chakraoui, M., Abdali, A., Ammoumou, A. y Amalou, I. (2022). *Federated Learning for Medical Imaging: An Updated State of the Art*. Ingénierie des Systèmes d'Information, vol. 27, núm. 1, February, 2022, pp. 143-150.

https://doi.org/10.18280/isi.270117

[26] Rauniyar, A., Haileselassie Hagos, D., Jha, D., Håkegård, J., Bagci, U. y Rawat, D. (2023). Federated Learning for Medical Applications: A Taxonomy, Current Trends, Challenges, and Future Research Directions. IEEE Internet of Things Journal, vol. 11, núm. 5, pp. 7374-7398.

https://ieeexplore.ieee.org/document/10304218

[27] Universidad Carlos III de Madrid (2025). *generAtion of Reliable syntheTIc health data for Federated leArning in seCure daTa Spaces-UC3M. - ARTIFACTS-UC3M.* https://researchportal.uc3m.es/display/act560824

[28] Vicerrectorado de Investigación US. *GenerAtion of Reliable syntheTIc health data for Federated leArning in seCure daTa Spaces – USE*. Vicerrectorado de Investigación, Universidad de Sevilla.
https://investigacion.us.es/sisius/sis_proyecto.php?idproy=37834

- Datos del proyecto: Proyectos de Generación de Conocimiento 2022
- Área: Tecnologías de la información y de las comunicaciones
- Subárea: Ciencias de la computación y tecnología informática
- Referencia: PID2022-141045OB-C42

[29] Agile Alliance (s.f.). *The 12 Principles behind the Agile Manifesto.* Agile Alliance, Agile Essentials. https://agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

[30] Junta de Andalucía (2018). *Perfiles profesionales ámbito informático*. Junta de Andalucía. https://www.juntadeandalucia.es/haciendayadministracionpublica/apl/pdc-front-publico/perfiles-licitaciones/consultas-preliminares/detalle?idExpediente=000000078484

[31] Riedel, P., Schick, L., von Schwerin, R., Reichert, M., Schaudt, D. y Hafner, A. (2024). *Comparative analysis of open-source federated learning frameworks - a literature-based survey and review*. International Journal of Machine Learning and Cybernetics, vol. 15, pp. 5257–5278.

https://doi.org/10.1007/s13042-024-02234-z

[32] Braungardt, A. (2024). *Flower, FATE, PySyft & Co.—Federated Learning Frameworks in Python.* https://medium.com/elca-it/flower-pysyft-co-federated-learning-frameworks-in-python-b1a8eda68b0d

[33] Stuecke, J. (2025). *Top 7 Open-Source Frameworks for Federated Learning*. Apheris. https://www.apheris.com/resources/blog/top-7-open-source-frameworks-for-federated-learning

[34] Flower Labs GmbH (2025). *Flower Framework Documentation*. Flower Framework. https://flower.ai/docs/framework/

[35] Twilt, J., Saha, A., Bosma, J., van Ginneken, B., Bjartall, A., Padhani, A., Bonekamp, D., Villeirs, G., Salomon, G., Giannarini, G., Kalpathy-Cramer, J., Barentsz, J., Maier-Hein, K., Rusu, M., Rouvière, O., van den Bergh, R., Panebianco, V., Kasivisvanathan, V., Obuchowski, N., …, y de Rooij, M (2025). *Evaluating Biparametric Versus Multiparametric Magnetic Resonance Imaging for Diagnosing Clinically Significant Prostate Cancer: An International, Paired, Noninferiority, Confirmatory Observer Study.* ScienceDirect, vol. 87, pp. 240-250.

https://doi.org/10.1016/j.eururo.2024.09.035

[36] Mishkin, D. y Matas, J. (2016). *All you need is a good init*. ICLR.

https://doi.org/10.48550/arXiv.1511.06422

[37] Subramanian, M., Rajasekar, V., Sathishkumar, V.E., Shanmugavadivel, K. y Nandhini, P. (2022). *Effectiveness of Decentralized Federated Learning Algorithms in Healthcare: A Case Study on Cancer Classification*. MDPI, Electronics, Artificial Intelligence in Healthcare: Theory, Methods and Applications.

https://doi.org/10.3390/electronics11244117

[38] Kahenga, F., Nyanyukweni, J. y Bigomokero, A. (2023). *A Comprehensive Security Analysis and Comparison of Federated Learning Algorithms: Resilience against Data and Model Poisoning Attacks*. Intelligent Systems and Advanced Telecommunication Lab,Computer Science department,University of Western Cape, South Africa.

[39] Katz, R. (2023). Performance comparison of different federated learning aggregation algorithms. https://resolver.tudelft.nl/6b7f0876-3f11-4095-aff3-2a59cf640250

[40] Sharma, I. (2025). *Federated Learning Strategies: An Overview*. Medium. https://medium.com/@theivision/federated-learning-strategies-an-overview-9c3e7d552249

[41] Yashin, A. (2024). *Saving and Loading Models in PyTorch: Best Practices*. Medium. https://yassin01.medium.com/saving-and-loading-models-in-pytorch-best-practices-cc4ce58e5bd7

[42] de Armas, J. (2018). *How do I save a trained model in PyTorch? Stack* Overflow. https://stackoverflow.com/a/49078976

[43] Amazon Web Services, Inc. o sus empresas afiliadas (2024). *¿Cuál es la diferencia entre YAML y JSON?* AWS. https://aws.amazon.com/es/compare/the-difference-between-yaml-and-json/

[44] Flower Labs GmbH (2025). *Upgrade to Flower 1.13*. GitHub, adap/flower. https://github.com/adap/flower/blob/040aa26e9a0b21a3254243c130a2c9761abff2c7/framework/docs/source/how-to-upgrade-to-flower-1.13.rst#L99

[45] Alhamid, M. (2025). *What is Cross-Validation?* Medium. https://medium.com/ data-science/what-is-cross-validation-60c01f9d9e75

II. Atribuciones de imágenes utilizadas en figuras

HAJICON (s.f.). *Framework*. Flaticon. https://www.flaticon.com/free-icon/framework_12119067

Becris (s.f.). *Neural*. Flaticon. https://www.flaticon.com/free-icon/neural_2103620

Smashicons (s.f.). *Database*. Flaticon. https://www.flaticon.com/free-icon/database_149206

Freepik (s.f.). *People*. Flaticon. https://www.flaticon.com/free-icon/people_3136102

Freepik (s.f.). *Circular Refreshment Arrow*. Flaticon. https://www.flaticon.com/free-icon/circular-refreshment-arrow_20666

orvipixel (s.f.). *Agile*. Flaticon. https://www.flaticon.com/free-icon/agile_16784040

fin