UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

APRENDIZAJE FEDERADO PARA LA ESTIMACIÓN DE RENDIMIENTO EN REDES MÓVILES

GRADO EN INGENIERÍA DE
TECNOLOGÍAS DE TELECOMUNICACIÓN

CARLOS CANO GARRIDO

MÁLAGA, 2025

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

APRENDIZAJE FEDERADO PARA LA ESTIMACIÓN DE RENDIMIENTO EN REDES MÓVILES

Autor: Carlos Cano Garrido

Tutor: Matías Toril Genovés

Departamento: Ingeniería de Comunicaciones

Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación

Palabras clave: Aprendizaje automático, aprendizaje supervisado, aprendizaje federado, *throughput* y árboles de decisión

Resumen

En el contexto actual de las redes móviles, la estimación de la capacidad máxima de celda se ha convertido en una herramienta clave para mejorar la eficiencia de la operación y anticipar degradaciones de servicio. Sin embargo, la necesidad de preservar la privacidad de los datos supone un gran desafío para la elaboración de modelos eficaces a la hora de realizar estas estimaciones. Este trabajo introduce el uso del Aprendizaje Federado (*Federated Learning*) como alternativa para construir modelos descriptivos que permitan estimar el caudal de transmisión promedio de celda sin la necesidad de compartir datos sensibles en entornos de red móvil. Para ello, se ha implementado un sistema basado en árboles de decisión del tipo XGBoost, adaptado a un entorno distribuido, en el que colaboran distintos clientes con el fin de elaborar un modelo global, manteniendo la privacidad de los datos. Durante todo el trabajo se sigue una línea comparativa con la versión centralizada del modelo descriptivo basado en XGboost, demostrando la viabilidad y el potencial de esta prometedora variante de la inteligencia artificial. Los modelos desarrollados se validan con datos de rendimiento generados con un simulador dinámico de nivel de sistema LTE-A.

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

FEDERATED LEARNING FOR PERFORMANCE ESTIMATION IN MOBILE NETWORKS

Author: Carlos Cano Garrido

Supervisor: Matías Toril Genovés

Department: Communications Engineering

Degree: Bachelor´s degree in Telecommunication Technologies Engineering

Keywords: Machine Learning, Supervised Learning, Federated Learning, throughput and decision trees

Abstract

In the current context of mobile networks, estimating the maximum cell capacity has become a key tool for improving operational efficiency and anticipating service degradation. However, the need to preserve data privacy makes it difficult to build effective models for such estimations. This work introduces the use of Federated Learning as an alternative approach to develop descriptive models capable of estimating the average cell throughput without the need to share sensitive data within mobile network environments. To this end, a decision tree-based system using XGBoost has been implemented and adapted to a distributed setting, where multiple clients collaborate to build a global model while preserving data privacy. Throughout the project, a comparative analysis is conducted with the centralized version of the XGBoost-based model, demonstrating the feasibility and potential of this promising branch of artificial intelligence. The developed models are validated using performance data generated by a dynamic system-level LTE-A simulator.

Agradecimientos

A mi tutor, Matías, por acogerme y guiarme durante el desarrollo de este trabajo. Gracias por ayudarme a adentrarme en el apasionante mundo del aprendizaje automático.

A mis compañeros de universidad, que a día de hoy son mucho más que eso. Un grupo de amigos sin competencia ni envidias. Gracias por toda la ayuda y por todos los momentos vividos, dentro y fuera de clase.

A mi familia, que ha sufrido y disfrutado este recorrido tanto como yo (o incluso más). Gracias por vuestro apoyo incondicional, por la calma en los momentos difíciles y por ser siempre el mejor ejemplo que he podido tener.

Y, en general, a todas las personas que han estado, están y, seguro, estarán.

Acrónimos

CM	Configuration Management
CPU	Central Processing Unit
CQI	Channel Quality Indicator
CV	Cross Validation
DL	Downlink
eMBB	enhanced Mobile Broadband
FL	Federated Learning
FTL	Federated Transfer Learning
GPU	Graphics Processing Unit
HFL	Horizontal Federated Learning
IDD	Independent and Identically Distributed
IoT	Internet of Things
KNN	K-Nearest Neighbors
KPI	Key Performance Indicator
LTE	Long Term Evolution
LTE-A	Long Term Evolution Advanced
MAE	Mean Absolute Error
MANE	Mean Absolute Normalized Error
MAPE	Mean Absolute Percentage Error

MEC	Multi-access Edge Computing
ML	Machine Learning
mMTC	massive Machine Type Communications
NF	Network Function
non-IID	Non Independent and Identically Distributed
OSS	Operations Support System
PCA	Principal Component Analysis
PDCP	Packet Data Convergence Protocol
PM	Performance Management
PRB	Physical Resource Block
QoE	Quality of Experience
QoS	Quality of Service
RFE	Recursive Feature Elimination
RMSE	Root Mean Squared Error
SLA	Service Level Agreement
SVM	Support Vector Machine
SVR	Support Vector Regression
TFG	Trabajo de Fin de Grado
TTI	Transmission Time Interval
UE	User Equipment
URLLC	Ultra-Reliable Low-Latency Communications
VFL	Vertical Federated Learning
XGBoost	Extreme Gradient Boosting

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

Contenido

Sinopsis

Este capítulo introductorio describe el contexto, la motivación y los objetivos del Trabajo de Fin de Grado. A su vez, se expone un análisis del estado de la técnica actual del ámbito del estudio, así como la estructura general que seguirá la memoria.

1.1. Contexto y motivación

Las comunicaciones móviles han desempeñado un papel fundamental en la transformación digital que han experimentado nuestras sociedades en las últimas décadas. Desde la primera generación analógica (1G) hasta las redes de cuarta generación (4G) basadas en la tecnología *Long Term Evolution* (LTE), ampliamente desplegadas a nivel global, han revolucionado la forma en la que las personas acceden y comparten información. En la era actual del internet móvil y el *big data*, caracterizada por un crecimiento exponencial del volumen de

datos y de la demanda de conectividad, aparecen una serie de desafíos para las nuevas generaciones de estándares móviles, como el aumento del número de dispositivos conectados simultáneamente a una red o la amplia diversidad de casos de uso [9].

Ante este panorama, la quinta generación de redes móviles (5G) surge como una evolución necesaria. El objetivo fundamental de 5G es proporcionar capacidades mucho más allá de las generaciones anteriores, como velocidades de transmisión de datos del orden del gigabit (eMBB), ultra baja latencia y alta fiabilidad (URLLC) y comunicaciones masivas de tipo máquina a máquina (mMTC). Estos avances habilitan nuevos modelos de servicio, recogidos en la figura 1.1, como el Internet de las Cosas (IoT), servicios en la nube instantáneos, la conducción autónoma y las aplicaciones críticas en tiempo real [10]. Esta diversidad ilustra la complejidad de las nuevas generaciones de redes móviles, en las que la red debe adaptarse a requisitos y condiciones muy distintas, lo que supone un desafío para su diseño e implementación.

Figura 1.1: Casos de uso de tecnologías 5G [1].

En un escenario tan complicado, proporcionar una buena calidad de servicio es cada vez más dificil para los provedores. Los indicadores de calidad de servicio (*Quality of Service,* QoS) y la percepción subjetiva de los usuarios (*Quality of Experience*, QoE) se encuentran estrechamente relacionados. Entre los diversos parámetros que conforman la QoS, como latencia, *jitter* o la pérdida de paquetes, el caudal (*throughput*) ha sido identificado como el factor más determinante en la percepción de usuario final, resultando crítica su importancia tanto en usuarios individuales como en entornos industriales. El estado del arte ha explorado ampliamente el uso de técnicas de aprendizaje automático (*Machine Learning*) para estimar estas métricas en una gran variedad de escenarios. La monitorización continua del *throughput* y la capacidad de anticipación en la detección de caidas de rendimiento son claves para garantizar los acuerdos de nivel de servicio (*Service Level Agreement*, SLAs) [11].

En esta misma línea, investigaciones recientes se han centrado en aplicar técnicas de *Machine Learning* para estimar el rendimiento de redes móviles 5G. En particular, [5] propone un marco de estimación del *throughput* en redes 5G basado en la explotación de indicadores clave de desempeño (*Key Performance Indicators*, KPIs) provenientes del sistema de soporte a la operación (*Operations Support System*, OSS). En el estudio se consigue una estimación precisa del caudal agregado de celda y de *slice*, permitiendo anticipar escenarios críticos y evaluar el impacto de reconfiguraciones en parámetros de la red, contribuyendo a una gestión más inteligente del sistema.

No obstante, a medida que estas soluciones basadas en datos se extienden a entornos reales, cobra especial relevancia una preocupación clave: la privacidad. La mayoría de los modelos tradicionales de aprendizaje automático requieren centralizar la información para ser entrenados. Esto implica la transmisión de grandes volúmenes de datos a servidores centralizados, ocasionando costes computacionales elevados. Además, requiere compartir información detallada, que en ocasiones plantea problemas de privacidad. En este contexto, el aprendizaje federado (*Federated Learning*) se presenta como una solución prometedora. Esta técnica permite entrenar modelos de inteligencia artificial de forma colaborativa en distintos nodos de la red, sin necesidad de compartir los datos brutos con un servidor central. Así se consigue preservar la privacidad de los datos y reducir el tráfico de red, manteniendo las mejoras en la eficiencia de la gestión de las redes móviles que proporciona la aplicación del aprendizaje automático [12].

1.2. Objetivos

El presente trabajo tiene como objetivo principal desarrollar modelos de capacidad de celda empíricos construidos a partir de datos recopilados de una red móvil real. Para ello, dicho problema de estimación se formula como un problema de regresión, que se resuelve mediante árboles de decisión. Como punto de partida, se considera el trabajo [5], donde se desarrollaron distintos modelos centralizados de estimación de *throughput* en redes móviles basados en aprendizaje supervisado. Como principal novedad, se pretende ampliar dicho enfoque aplicando técnicas de aprendizaje federado. Se persigue preservar la privacidad de los datos, reducir el tráfico de red y adaptar el entrenamiento del modelo a escenarios distribuidos más realistas.

Para ello, se propone adaptar la implementación de árboles de decisión del tipo XGBoost a un entorno federado simulado, evaluando el impacto de esta nueva estrategia sobre la precisión y eficiencia computacional del modelo. El estudio se centrará en la estimación del *throughput* del enlace descendente, agregado a nivel de celda y promediado en el tiempo, estimado a partir de indicadores de rendimiento agregados por celda y hora. Los modelos desarrollados se validan con datos de rendimiento generados con un simulador de red móvil dinámico de nivel de sistema. Para ello, se plantean los siguientes objetivos:

- **Estudio del estado de la técnica.** Analizar las tendencias actuales en el uso de técnicas de aprendizaje automático para la predicción del rendimiento en redes móviles, así como el papel del *Federated Learning* como solución a los desafíos de privacidad y escalabilidad.
- **Presentación del marco teórico.** Aclarar los fundamentos teóricos necesarios para la adecuada comprensión del proyecto.
- **Construcción de un modelo base centralizado.** Entrenar un modelo de estimación de *throughput* en celdas móviles utilizando un modelo de aprendizaje automático, siguiendo la metodología del artículo de referencia [5]. Para su implementación se han usado herramientas y librerías de libre distribución. En concreto, se emplea el lenguaje Python junto con las librerías `scikit-learn` y `XGBoost`. Además, se utilizará `Pandas` y `Matplotlib` para la manipulación de datos y la generación de gráficos.
- **Construcción del modelo en un entorno federado.** Dividir el juego de datos en subconjuntos que representen diferentes emplazamientos de red, entrenando de forma local distintos modelos que serán combinados para elaborar un modelo global colaborativamente. La implementación se ha

realizado adaptando un esqueleto disponible en un repositorio público de GitHub, basado en la librería `FLOWER`, específicamente diseñada para el desarrollo de entornos de aprendizaje federado.

- **Evaluación comparativa.** Medir el rendimiento de los modelos centralizado y federado sobre un conjunto de test común, utilizando distintas cifras de mérito, así como técnicas de análisis gráfico.
- **Análisis crítico.** Extraer conclusiones sobre la viabilidad del aprendizaje federado para la estimación de *throughput* en redes móviles de última generación, identificando sus limitaciones y proponiendo líneas de investigación futura.

1.3. Estado de la técnica

En este apartado se presenta una revisión de la situación actual del área de conocimiento en la que se enmarca el estudio, introduciendo los modelos más ampliamente utilizados para la optimización de redes móviles hasta la fecha, así como el papel emergente del aprendizaje federado en este campo. Asimismo, se profundiza en los resultados obtenidos por el modelo elegido de referencia, XGBoost, en los estudios más actuales y en cómo se ha implementado en entornos federados.

La estimación precisa de KPIs, y más concretamente del *throughput* en redes móviles 5G y LTE, se ha convertido en un área de investigación activa debido a su importancia para la optimización de recursos, la planificación de la red y la garantía de la experiencia del usuario (*Quality of Experience*, QoE). El presente trabajo se centra en la aplicación de modelos descriptivos para esta tarea. Estos modelos establecen relaciones entre variables medidas en mismo instante de tiempo. Este enfoque se ha explorado ampliamente en trabajos recientes, la aplicación de técnicas de *Machine Learning* y *Deep Learning* ha demostrado una gran capacidad para captar patrones complejos, permitiendo estimaciones precisas y adaptativas del *throughput*. Diversas técnicas de aprendizaje supervisado han sido objeto de investigación para abordar este desafío: regresión lineal [13], K vecinos más cercanos (*K-Nearest Neighbors*, KNN) [5], árboles de decisión [5, 13, 14, 15, 16], máquinas de vectores de voporte para regresión (*Support Vector Regression*, SVR) [13, 5], y redes neuronales [5, 15], entre otros.

En el trabajo se emplean modelos basados en árboles de decisión, un tipo de modelo muy destacado por su simplicidad, interpretabilidad y eficiencia computacional. Los árboles de decisión realizan las estimaciones basándose en un es-

quema de decisiones jerárquicas. Cada nodo del árbol toma una decisión binaria que conduce hacia la estimación final. Estas decisiones individuales se toman en función de criterios que mejoran la precisión de la estimación [17]. A pesar de su buen rendimiento, cuando este tipo de modelos es entrenado de forma individual tiende a memorizar los datos, dando lugar a malas estimaciones al aplicarse sobre muestras nunca vistas previamente. Para solventar estas limitaciones, se proponen técnicas de ensamblaje como el ensamblado por agrupamiento (*bagging*) o el ensamblado secuencial (*boosting*) [18]. El *bagging* entrena múltiples árboles independientes en subconjuntos diferentes del conjunto de datos y luego promedia sus estimaciones. Por su parte, el *boosting* crea los árboles de forma secuencial, forzando que cada nuevo árbol aprenda de los errores del anterior.

Entre los árboles de decisión, XGBoost (*Extreme Gradient Boosting*) ha sobresalido como un modelo versátil y potente, logrando resultados altamente competitivos en tareas de estimación. XGBoost es un modelo de árboles de decisión que pertenece a la familia de los métodos de *boosting*. Este enfoque introduce mecanismos que mejoran la generalización, como la regularización explícita de la función de pérdida, el escalado de la contribución de cada árbol y la selección aleatoria de columnas durante el entrenamiento [14]. Estas características lo convierten en una opción especialmente adecuada para entornos complejos, como es el caso de las redes 5G. En el contexto específico de redes móviles, varios estudios respaldan su rendimiento superior respecto a otros algoritmos tradicionales. En [5], se comparan siete modelos de inteligencia artificial aplicados a la estimación de *throughput* en distintos escenarios reales: redes con *Network Slicing* y *slices* de servicio único, redes con *slices* multi-servicio y redes sin *slicing*. XGBoost mostró un rendimiento consistente, muy cercano al de las redes neuronales, superando ampliamente a modelos como SVR, regresión lineal o KNN. Es amplia la bibliografía en la que este modelo ha demostrado ser uno de los más precisos y eficientes, en algunos casos incluso superando a arquitecturas de aprendizaje profundo [16, 14].

Si bien las técnicas de aprendizaje automático, incluyendo XGBoost, han demostrado su eficacia en la estimación del *throughput* en redes móviles bajo un enfoque centralizado, esta tarea presenta desafíos inherentes en escenarios donde los datos están distribuidos entre múltiples entidades. Este es el caso de las redes móviles, donde existen distintos emplazamientos, celdas, portadoras, *slices*, operadores o incluso dispositivos de usuario. En este contexto, la privacidad de la información puede ser una preocupación primordial. En las redes móviles, los datos de rendimiento y comportamiento espacio temporal de la demanda de tráfico pueden ser sensibles, y no siempre es deseable ni factible centralizar grandes volúmenes de datos para el entrenamiento de modelos.

Ante esta situación, el Aprendizaje Federado (*Federated Learning*, FL) [19] emerge como una rama prometedora de la inteligencia artificial para entornos donde los datos están distribuidos y no conviene centralizarlos por razones de privacidad, coste o latencia. De esta forma, el aprendizaje federado permite entrenar modelos de forma colaborativa entre distintos clientes, manteniendo los datos locales privados sin ser enviados a ningún servidor central.

Estudios como [12] profundizan en la aplicación del *Federated Learning* en el ámbito de las redes móviles, un campo de investigación relativamente reciente pero que ha despertado un creciente interés. El enfoque de aprendizaje colaborativo se adapta especialmente bien a la arquitectura de las redes 5G, donde los datos relevantes para la gestión de la red se generan de forma natural en diferentes funciones de red virtualizadas (*Network Functions*, NFs), estaciones base o entornos de computación en el borde (*Multi-access Edge Computing*, MEC). Gracias a su estructura distribuida, además de mejorar la privacidad, el aprendizaje federado permite que los modelos se entrenen cerca del origen de los datos, lo que contribuye a minimizar la latencia. Asimismo, se reduce el ancho de banda requerido, al transmitirse únicamente actualizaciones de los modelos en lugar del volumen completo de datos en bruto [12].

La aplicación de XGBoost en entornos de aprendizaje federado presenta desafíos particulares debido a la naturaleza de los árboles de decisión, los cuales no están definidos por parámetros continuos como ocurre en las redes neuronales, sino por estructuras discretas basadas en reglas de decisión y umbrales. Esta diferencia impide aplicar directamente técnicas de agregación como *Federated Averaging*, comúnmente utilizadas en modelos paramétricos, que combinan la información de los modelos promediando sus pesos en el servidor central. Además, en XGBoost, cada árbol se construye de forma secuencial para corregir los errores acumulados por los árboles anteriores, lo que complica aún más el diseño de estrategias de entrenamiento y combinación de modelos en escenarios distribuidos. Para abordar estas limitaciones, se han propuesto distintos enfoques [20] como la agregación tipo *bagging*, en la que cada cliente entrena árboles localmente y el servidor central los combina en un ensamblado global. Este enfoque ha mostrado ser efectivo incluso en entornos con datos no independientes e idénticamente distribuidos (*non-IID*) y distribuciones heterogéneas entre clientes. Otra alternativa es el entrenamiento cíclico, en el que los clientes participan secuencialmente en la construcción del modelo global, transmitiendo el conjunto actualizado de árboles al siguiente nodo en cada ronda. Aunque esta técnica puede requerir un mayor número de rondas para converger, ofrece ventajas en cuanto a simplicidad en su implementación.

En este contexto, el proyecto se centrará en la implementación simulada de

XGBoost en un escenario federado. Esta es un área de investigación activa que busca combinar las ventajas de rendimiento de XGBoost con los beneficios de privacidad y descentralización del *Federated Learning*.

1.4. Estructura del documento

A continuación se presenta la estructura de la memoria.

- **Capítulo 1. Introducción.**

  El presente capítulo pretende aportar una visión general de la motivación y los objetivos del trabajo.

- **Capítulo 2. Marco teórico.**

  Se explican conceptos teóricos de *Machine Learning*, introduciendo al lector en los principios básicos del aprendizaje supervisado para poder profundizar posteriormente en los árboles de decisión y el paradigma del Aprendizaje Federado.

- **Capítulo 3. Diseño e implementación del modelo de estimación.**

  Se detalla el proceso seguido para el diseño del modelo. Se introduce el juego de datos utilizado, la creación de los modelos y los criterios de evaluación utilizados. Así como aspectos de implementación, como el entorno de desarrollo.

- **Capítulo 4. Pruebas.**

  Se presenta la metodología experimental, incluyendo los procesos de entrenamiento y las configuraciones definitivas de los modelos. Finalmente, se exponen los resultados obtenidos en todas las pruebas.

- **Capítulo 5. Conclusiones y líneas futuras.**

  Se presenta un análisis final del trabajo por parte del autor, junto a una serie de mejoras y extensiones en posibles trabajos futuros.

Capítulo 2

Marco Teórico

Contenido

Sinopsis

El presente capítulo explica los fundamentos conceptuales y técnicos necesarios para comprender el enfoque desarrollado en este trabajo. Se introducen primero los principios básicos del aprendizaje automático, con especial atención a los modelos de estimación con aprendizaje supervisado. Posteriormente, se profundiza en los algoritmos de árboles de decisión, detallando su funcionamiento interno. Finalmente, se abordan los fundamentos del aprendizaje federado, explicando su arquitectura, sus beneficios frente al aprendizaje centralizado y los desafíos técnicos que conlleva su implementación. Este marco teórico establece la base necesaria para entender las decisiones tomadas a lo largo del desarrollo del proyecto.

2.1. Aprendizaje Automático

En este apartado se introducen los fundamentos del aprendizaje automático. Se enumerarán los distintos tipos de técnicas de *Machine Learning*, así como conceptos claves para la correcta implementación de modelos de aprendizaje automático como la generalización y el control de complejidad.

2.1.1. Introducción al aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial que estudia algoritmos capaces de aprender patrones a partir de datos y tomar decisiones sin estar explícitamente programados para ello. En lugar de definir un conjunto fijo de reglas, los algoritmos de ML construyen modelos matemáticos que generalizan el comportamiento observado en conjuntos de entrenamiento y los aplican a nuevas muestras. Dependiendo de la naturaleza de los datos usados para el entrenamiento, se distinguen tres tipos principales de aprendizaje [21]:

- **Aprendizaje supervisado:** las observaciones se presentan en forma de pares entrada-salida. El objetivo es aprender una función que modele la relación entre las variables de entrada (predictoras) y la variable de salida (etiqueta). El conjunto de datos original se divide típicamente en varios subconjuntos: el conjunto de entrenamiento, el conjunto de validación y el conjunto de evaluación. El algoritmo se entrena para generar una función o estructura capaz de predecir la salida esperada a partir de nuevas entradas, identificando patrones en los datos etiquetados del conjunto de entrenamiento. Este proceso continúa hasta que se alcanza un nivel de precisión aceptable sobre el conjunto de validación.

  El aprendizaje supervisado incluye principalmente dos tipos de tareas: regresión, cuando la variable de salida es continua (como en nuestro caso, donde se busca estimar el *throughput* de una red móvil), y clasificación, cuando la variable es categórica (por ejemplo, identificar si un correo electrónico es *spam* o no). Algunos algoritmos comunes de SL son: las redes neuronales, la regresión lineal, los árboles de decisión, *k*-vecinos más cercanos, regresión logística, máquinas de soporte vectorial (*Support Vector Machines*, SVM) y *Naive Bayes*. Asimismo, ciertos métodos de reducción de dimensionalidad guiada, como *Recursive Feature Elimination* (RFE) [22], también se consideran técnicas supervisadas, dado que utilizan la variable objetivo para seleccionar las características más relevantes.

- **Aprendizaje no supervisado:** las observaciones contienen únicamente variables de entrada, sin etiquetas asociadas. El objetivo es descubrir estructuras ocultas o patrones en los datos, como la agrupación de muestras similares en clústeres. Este enfoque resulta útil para segmentación de datos, detección de anomalías o visualización. Entre las técnicas más utilizadas se encuentran métodos de reducción de dimensionalidad como PCA (*Principal Component Analysis*), así como algoritmos de agrupamiento como *k*-means, agrupamiento jerárquico o *fuzzy clustering*.
- **Aprendizaje semi-supervisado:** combina aspectos de los dos anteriores. Parte de los datos de entrenamiento incluye etiquetas, mientras que una proporción significativa no las contiene. El modelo aprovecha los datos no etiquetados para mejorar la generalización en la predicción de aquellos que sí tienen etiquetas [21].

Aunque los tres tipos de aprendizaje tienen aplicaciones prácticas en tareas de modelado descriptivo en ingeniería, el presente trabajo se enmarca dentro del aprendizaje supervisado, ya que se dispone de un conjunto de datos etiquetado. En este caso, diversas métricas representativas del comportamiento de la red móvil actúan como variables de entrada (predictores) que permiten estimar la variable objetivo (*throughput*), que será la etiqueta de salida del modelo.

2.1.2. Generalización y control de complejidad en aprendizaje automático

Un aspecto esencial en cualquier modelo de aprendizaje automático es su capacidad de generalización, que determina su habilidad para describir correctamente nuevas muestras no vistas durante el entrenamiento. La figura 2.1 ilustra el rendimiento de modelos de aprendizaje automático en función de su capacidad de generalización o ajuste. La gráfica de la derecha corresponde a un modelo excesivamente complejo, que puede memorizar los datos de entrenamiento incluyendo el ruido y las irregularidades específicas del conjunto, perdiendo la capacidad de extrapolarse a nuevos datos. Este fenómeno se conoce como sobreajuste (*overfitting*) y es crucial evitarlo para que un modelo pueda ser útil en entornos reales, donde se enfrenta a situaciones diferentes a las contenidas en el conjunto de entrenamiento. Por el contrario, si el modelo es demasiado simple, puede ser incapaz de capturar la complejidad de los datos y cometer errores sistemáticos, este fenómeno se denomina subajuste (*underfitting*), escenario que se visualiza en la gráfica de la izquierda de la imagen. Por su parte, la gráfica central muestra un modelo con correcta generalización.

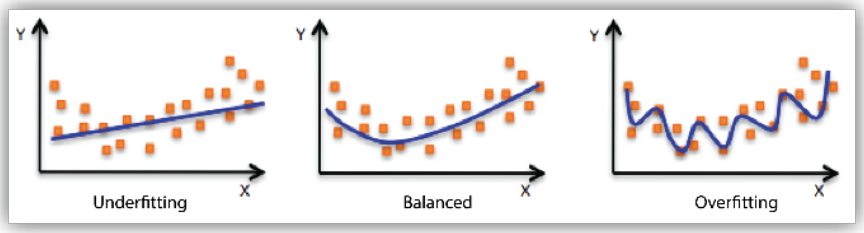

Figura 2.1: Subajuste, sobreajuste y buen ajuste en un modelo de aprendizaje automático [2].

Este equilibrio entre simplicidad y complejidad es conocido en mucha bibliografía como el principio de *Goldilocks* [3]. La figura 2.2 permite visualizar esta situación, ilustrando el compromiso entre dos conceptos fundamentales del aprendizaje automático: el sesgo (*bias*) y la varianza. El sesgo representa la tendencia del modelo a simplificar en exceso la relación entre variables, generando errores por aproximaciones demasiado optimistas. Mientras que la varianza indica la sensibilidad del modelo a pequeñas variaciones en los datos. Un modelo con alto sesgo y baja varianza tiende a subajustar, mientras que uno con bajo sesgo y alta varianza sobreajusta. Encontrar el punto óptimo entre ambos, la llamada "Zona *Goldilocks*", es clave para maximizar la capacidad de generalización del modelo y lograr buenas estimaciones de forma robusta.

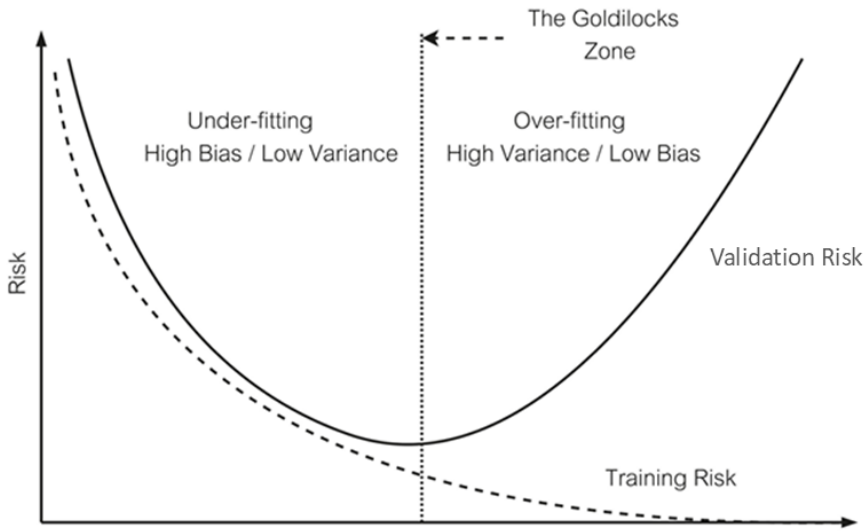

Figura 2.2: Gráfica de equilibrio sesgo-varianza [3].

Para entender los métodos que se utilizan en la optimización de este compromiso se debe entender cómo se evalúa si un modelo es preciso o no. Para ello, es clave introducir el concepto de función de pérdida. Esta función cuantifica el error cometido al realizar predicciones, midiendo la diferencia entre las salidas predichas por el modelo y los valores reales esperados en el conjunto de validación. Durante el entrenamiento, el modelo ajusta sus parámetros internos mediante algoritmos de optimización numéricos con el fin de minimizar esta función de pérdida. Entre las funciones de pérdida más comunes se encuentran [23]:

- La **pérdida absoluta** o *Mean Absolute Error (MAE)*, que penaliza de forma lineal los errores:
$$\mathcal{L}_{\text{MAE}} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$
- La **pérdida cuadrática** o *Mean Squared Error (MSE)*, mide la diferencia al cuadrado entre estimaciones y valores reales, penalizando más fuertemente los errores grandes:
$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
- La **pérdida de Huber**, que combina las ventajas de las dos anteriores: se comporta como la cuadrática para errores pequeños y como la lineal para errores grandes, siendo más robusta frente a valores atípicos. Un umbral δ determina cuándo se pasa de penalización cuadrática a lineal:
$$\mathcal{L}_{\text{Huber}} = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{si } |y_i - \hat{y}_i| \leq \delta \\ \delta \cdot \left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) & \text{en otro caso} \end{cases}$$

Para encontrar el equilibrio entre sesgo y varianza una de las técnicas más usadas consiste en modificar la función de pérdida añadiendo un término adicional, llamado regularizador, que penaliza modelos excesivamente complejos. De este modo, el modelo se ve forzado a optar por soluciones más simples, priorizando la adaptación a nuevos datos frente a la precisión absoluta en el entrenamiento. El resultado es una función objetivo, que combina la función de pérdidas con los términos de regularización, guiando el aprendizaje hacia un modelo más generalizable [3].

Existen múltiples técnicas de regularización, entre las que destacan las denominadas L1 (*Lasso*) [24] y L2 (*Ridge*) [25]. Estas dos formas de regularización

son especialmente relevantes en este proyecto, ya que se emplean para controlar la complejidad de los modelos. En el caso de L1, se añade a la función de pérdida un término proporcional a la norma ℓ_1 del vector de parámetros del modelo. Este término penaliza los valores grandes de dichos parámetros y promueve la dispersión (*sparsity*), es decir, tiende a que muchos de los coeficientes del modelo se reduzcan exactamente a cero. Como resultado, se obtienen modelos más simples y, a menudo, más interpretables. Este enfoque se representa matemáticamente mediante la siguiente función objetivo:

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + \lambda \|\mathbf{x}\|_1. \tag{2.1}$$

donde $L(\mathbf{x})$ es la función de pérdida, λ es un hiperparámetro que controla la intensidad de la regularización, y $\|\mathbf{x}\|_1$ es la norma ℓ_1, que suma el valor absoluto de cada parámetro del modelo. Al ajustar λ, es posible controlar cuánto se penaliza la complejidad del modelo.

Por su parte, la regularización L2 (*Ridge*) aplica una penalización basada en la norma cuadrática de los coeficientes, conocida como norma ℓ_2. En este caso, en vez de anular completamente ciertos coeficientes, se les reduce progresivamente su magnitud, distribuyendo de forma más equilibrada sus pesos. Esto tiende a generar modelos más estables y con menor varianza. Matemáticamente, se expresa como:

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + \lambda \|\mathbf{x}\|_2^2. \tag{2.2}$$

Aquí, $\|\mathbf{x}\|_2^2$ representa la suma de los cuadrados de los parámetros del modelo. Igual que en el caso de L1, el hiperparámetro λ, controla el grado de regularización aplicado.

Parámetros de un modelo

Durante el proceso de entrenamiento de un modelo de aprendizaje automático, se pueden distinguir dos tipos principales de parámetros. Por un lado, están los parámetros ajustables, que el propio modelo aprende a partir de los datos mediante algoritmos de optimización, como los pesos en una red neuronal o los umbrales de decisión en un árbol. Son estos los parámetros que se modifican iterativamente con el objetivo de minimizar la función de pérdida.

Por otro lado, existen los hiperparámetros, que son valores predefinidos antes del entrenamiento y que no son aprendidos automáticamente durante este. Su configuración afecta directamente al comportamiento del modelo. La elección

adecuada de los hiperparámetros es fundamental para garantizar la generalización del modelo, valores lejanos del punto óptimo pueden conducir a sobreajuste o subajuste. Para ello existen diversas técnicas de ajuste de hiperparámetros (*tuning*), como la búsqueda en cuadrícula (*grid search*) o la búsqueda aleatoria (*random search*), que exploran combinaciones posibles dentro de un espacio predefinido y evalúan su impacto en el rendimiento del modelo.

Reducción de dimensionalidad

La reducción de dimensionalidad es una técnica fundamental en Machine Learning para mejorar la capacidad de generalización de los modelos y reducir su complejidad. Consiste en disminuir el número de variables de entrada (o *features*) utilizadas para entrenar el modelo, eliminando las que sean más irrelevantes para la predicción o conservando todas las características pero reduciendo su dimensión teniendo en cuenta relaciones estadísticas.

Este proceso no solo ayuda a evitar el sobreajuste en conjuntos de datos de alta dimensionalidad, sino que también mejora la eficiencia computacional y la interpretabilidad del modelo. En contextos reales, la optimización del coste computacional es un requisito primordial, y reducir la cantidad de datos a almacenar y procesar contribuye a esta tarea notablemente.

Existen múltiples técnicas para reducir la dimensionalidad, que utilizan tanto métodos supervisados como no supervisados. Los métodos no supervisados, como el análisis de componentes principales (*Principal Component Analysis*, PCA), transforman el conjunto de datos en un espacio de menor dimensión sin tener en cuenta la variable objetivo, basándose generalmente en criterios estadísticos como la varianza. En cambio, los métodos supervisados, como *Recursive Feature Elimination* (RFE), sí consideran la variable de salida para identificar y seleccionar las características más relevantes. En este trabajo se emplea RFE, que será descrito en detalle en el Capítulo 3.

Evaluación del entrenamiento y convergencia

El proceso de control de complejidad encargado de favorecer la capacidad de generalización del modelo debe ejecutarse durante el propio entrenamiento. Para ello, una técnica común es la división del juego de datos en 3 subconjuntos:

- **Conjunto de entrenamiento:** ajusta los parámetros del modelo para que aprenda patrones de los datos.

- **Conjunto de validación:** se emplea durante el entrenamiento para monitorizar el rendimiento del modelo, aplicar técnicas de regularización como el *early stopping*, ajustar los hiperparámetros o controlar técnicas de reducción de dimensionalidad. La introducción de este subconjunto es clave para poder evaluar si el modelo está generalizando correctamente sin necesidad de recurrir al subconjunto reservado para comprobar la precisión final del modelo.
- **Conjunto de test:** está completamente aislado del proceso de entrenamiento. Su función es evaluar el rendimiento del modelo de forma objetiva, simulando su comportamiento en un entorno real sobre datos nunca vistos previamente.

Además, esta división de los datos permite el uso de otra técnica clave para supervisar el aprendizaje del modelo, como son las curvas de convergencia. Estas curvas permiten visualizar cómo evolucionan las métricas de error durante el entrenamiento. La figura 2.2 del inicio de este apartado es también un ejemplo de este tipo de gráfica. Las curvas de convergencia constituyen una herramienta muy útil para monitorizar la adaptación del modelo a los datos durante la fase de entrenamiento sin necesidad de recurrir al conjunto de datos reservado para el test. Estas gráficas permiten observar el comportamiento del modelo durante el entrenamiento y detectar señales de sobreajuste en función de cómo evolucionan las métricas en los conjuntos de entrenamiento y validación. La curva de *Validation Risk* en la imagen se corresponde con las pérdidas evaluadas sobre el subconjunto de validación y la de *Training Risk* sobre el subconjunto de entrenamiento. Cuando se detecta que la curva de validación empieza a despegarse de la de entrenamiento de forma sostenida, como ocurre en la mencionada figura a partir de la línea discontínua vertical, se está ante un claro indicio de sobreajuste. Este comportamiento sugiere que el modelo está memorizando los datos del conjunto de entrenamiento, sobre el que cada vez muestra mayor precisión, dejando de generalizar bien en nuevas muestras [3].

2.1.3. Árboles de decisión

En el trabajo se ha utilizado un modelo de aprendizaje automático supervisado basado en árboles de decisión. A continuación, se introduce al lector este tipo de modelos.

Estructura de un modelo basado en árboles de decisión

Los árboles de decisión [17] son uno de los enfoques más ampliamente utilizados en el aprendizaje automático, tanto para tareas de clasificación como de regresión. Destacan por ser fácilmente interpretables, eficientes y capaces de captar patrones no lineales.

Un árbol de decisión representa una serie de decisiones jerárquicas en forma de grafo. La figura 2.3 muestra un ejemplo sencillo de árbol de decisión para una tarea de regresión. El proceso comienza en el nodo raíz, donde se evalúa la primera condición, sobre una variable de entrada, que suele ser la más discriminante (en este caso, $X_1 > 0{,}5$). Si esta condición se cumple, se sigue la rama izquierda; en caso contrario, se toma la derecha. Cada nodo interno evalúa una nueva condición. En el ejemplo, si se ha seguido la rama izquierda, se llega a un nodo que comprueba si $X_2 < 0{,}8$. A partir de ahí, según el resultado de esta nueva comparación, se accede a una de las hojas del árbol, donde se encuentra el valor estimado de la variable objetivo. Por ejemplo, si ambas condiciones se cumplen ($X_1 > 0{,}5$ y $X_2 < 0{,}8$), el modelo estima un valor de $Y = 0{,}3$. En cambio, si $X_1 \leq 0{,}5$, se accede directamente a una hoja con un valor estimado de $Y = 2$.

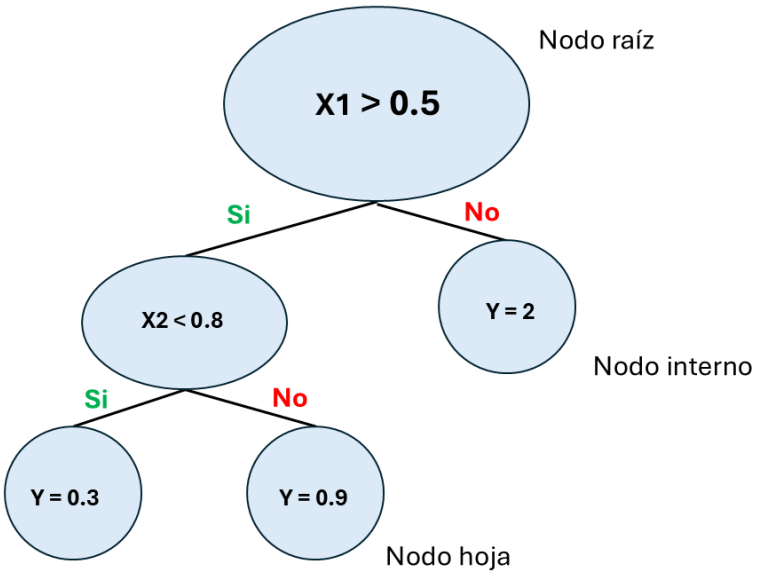

Figura 2.3: Ejemplo de un árbol de decisión.

Este tipo de estructura permite dividir el espacio de entrada en regiones homogéneas, en cada una de las cuales el modelo predice un valor constante (en el caso de regresión). La simplicidad de su lógica basada en decisiones binarias hace que los árboles de decisión sean especialmente útiles e interpretables.

Gracias a estas características, los árboles de decisión resultan eficaces para muchas tareas. No obstante, una de sus principales limitaciones es su tendencia al sobreajuste cuando crecen demasiado. Para evitarlo, es necesario un control adecuado del proceso de entrenamiento.

Construcción del modelo

En los problemas de regresión, los árboles de decisión se construyen de forma jerárquica mediante una estrategia conocida como particionamiento binario recursivo [17]. Este proceso comienza agrupando todos los datos de entrenamiento en un nodo raíz y, a partir de ahí, el algoritmo va añadiendo nodos de forma progresiva. Cada nuevo nodo introduce una división del conjunto de datos con el objetivo de mejorar la capacidad del modelo para estimar la variable de salida. Para ello, se intenta que los ejemplos agrupados en cada subconjunto tengan valores de la variable objetivo relativamente similares entre sí, reduciendo error global en las estimaciones. Esta reducción del error al generar subconjuntos más homogéneos se debe a que los árboles de decisión asignan una única predicción a cada subconjunto o rama (normalmente la media de las variables objetivo contenidas en el subconjunto). Cuanto más parecidos sean los valores reales de las etiquetas de un subconjunto, más representativa será su media y, por tanto, menor será la pérdida asociada.

Estas divisiones se realizan mediante comparaciones de variables predictoras con valores umbrales. Por ejemplo, en el árbol de la figura 2.3, la primera división se basa en la condición $X_1 > 0.5$. Si esta división es adecuada, cabría esperar que las muestras que cumplen esta condición (rama izquierda) tengan valores de salida similares entre sí (por ejemplo, $Y < 1$), y que las que no la cumplen (rama derecha) también presenten cierta homogeneidad en sus etiquetas (en este caso, homogeneidad total entre ellas tomando todas $Y = 2$). Esta separación inicial permite que, conforme se añadan nuevos nodos, las predicciones sean más precisas y específicas para cada subconjunto.

Para determinar cuál es la mejor división en cada nodo, el algoritmo evalúa muchas posibles combinaciones de variable predictora y umbral antes de introducir un nuevo nodo. Posteriormente, selecciona aquella que separa los datos de forma que se minimice la suma de las desviaciones cuadráticas respecto a la media de ambos subconjuntos generados en esa divisón binaria:

$$\arg\min_{t} \left(\sum_{i \in R_{\text{izq}}(t)} (y_i - \bar{y}_{\text{izq}})^2 + \sum_{i \in R_{\text{der}}(t)} (y_i - \bar{y}_{\text{der}})^2 \right) \tag{2.3}$$

donde $R_{\text{izq}}(t)$ y $R_{\text{der}}(t)$ son los subconjuntos generados por la división con umbral t, y $\bar{y}_{\text{izq}}$, $\bar{y}_{\text{der}}$ son las medias de las etiquetas en cada subconjunto.

Este procedimiento se repite en cada nuevo nodo generado, creando una estructura en forma de árbol. La generación de divisiones continúa mientras se sigan cumpliendo ciertos criterios, como alcanzar un número mínimo de muestras en un nodo, una profundidad máxima del árbol, o no encontrar ninguna división que reduzca significativamente el error. Cuando se alcanza una condición de parada, el nodo se convierte en una hoja. En los árboles de regresión, cada hoja contiene una estimación que corresponde generalmente a la media de los valores de la variable objetivo de las muestras que han llegado hasta ese nodo. De esta forma, el árbol divide el espacio de entrada en regiones y asigna a cada una un valor estimado constante.

Este método resulta eficaz siempre que se empleen mecanismos para mitigar la tendencia al sobreajuste. Si se permite que el árbol siga añadiendo nodos sin limitaciones, puede memorizar los datos de entrenamiento, perdiendo su capacidad de generalización. Para reducir este riesgo, se utilizan técnicas como la poda (*pruning*), que consiste en limitar el tamaño del árbol, por ejemplo, estableciendo una profundidad máxima durante su construcción o eliminando ramas completas una vez finalizada la elaboración del árbol, si estas no aportan mejoras significativas al rendimiento.

Estas limitaciones empeoran la capacidad predictiva individual de cada árbol. En muchas ocasiones, para compensarlo, se recurre al ensamblaje [18]. Esta técnica consiste en combinar múltiples árboles débiles para conseguir modelos más precisos, a la vez que se controla el riesgo de sobreajuste de cada arbol. Las dos estrategias principales son:

- ***Bagging***, que entrena los árboles en paralelo con distintos subconjuntos del juego de datos y promedia las predicciones. Un ejemplo es la técnica de bosques aleatorios (*Random Forest*).
- ***Boosting***, que entrena árboles de forma secuencial, y cada nuevo árbol trata de corregir los errores del anterior. El modelo de referencia del proyecto, XGBoost, utiliza este enfoque.

En este trabajo se hace uso de ambas técnicas. La variante centralizada se basa en la técnica de *boosting*, mientras que la variante federada añade *bagging*.

2.2. Aprendizaje Federado

El aprendizaje federado (*Federated Learning*, FL) es una técnica de aprendizaje automático diseñada para entrenar los algoritmos a través de una arquitectura distribuida. La figura 2.4 muestra un esquema con la idea básica de aprendizaje federado. A diferencia del enfoque clásico, que centraliza los datos en un único servidor para entrenar el modelo, el FL propone que cada cliente entrene el modelo localmente sobre sus propios datos. De este modo, durante todo el proceso de construcción del modelo global se mantiene la privacidad, ya que los datos en crudo nunca se comparten. En su lugar, cada nodo transmite al servidor central únicamente actualizaciones del modelo, como matrices de pesos en redes neuronales o los árboles generados en modelos basados en árboles de decisión.

Se trata de un enfoque novedoso que busca dar respuesta a los desafíos relacionados con la privacidad en el entrenamiento de modelos de aprendizaje automático. Estos modelos, para alcanzar un alto nivel de precisión, suelen necesitar grandes volúmenes de datos, que en muchas ocasiones proceden de clientes distintos que no desean compartir directamente su información. El *Federated Learning* permite preservar la confidencialidad, ya que los datos nunca abandonan el entorno local.

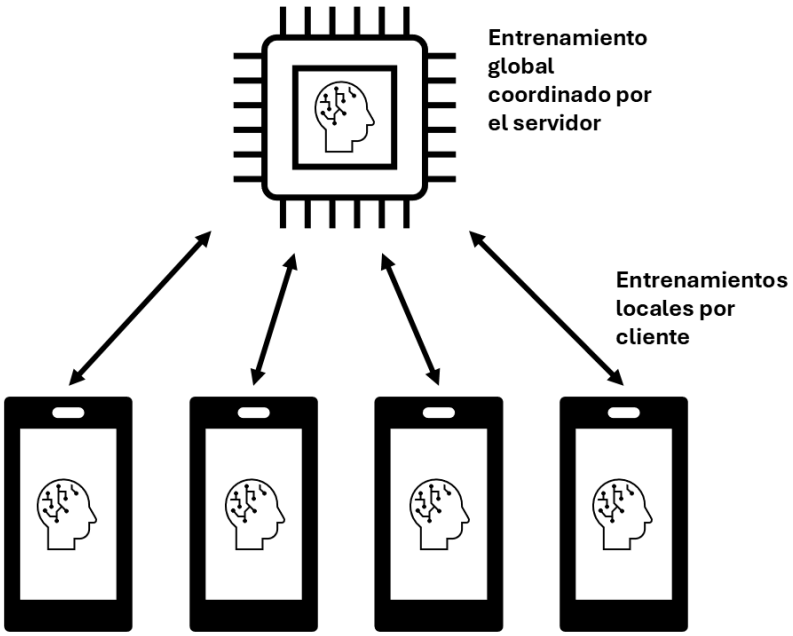

Figura 2.4: Ilustración de aprendizaje federado.

Además de mejorar la privacidad, esta técnica contribuye a una mayor eficiencia en la comunicación, ya que en lugar de distribuir los datos completos, solo se comparten parámetros del modelo, mucho más ligeros. Por último, al permitir que el entrenamiento ocurra cerca del origen de los datos, el aprendizaje federado también puede mejorar la latencia de los sistemas en los que se implemente [19].

El proceso típico de construcción de un modelo de aprendizaje federado se basa en un ciclo iterativo de entrenamiento local y agregación en un servidor central. Aunque existen múltiples variantes, el funcionamiento básico del aprendizaje federado se compone de las siguientes fases:

1. **Entrenamiento local:** los modelos se entrenan localmente en cada nodo con los datos descentralizados.

2. **Comunicación entre modelos:** los participantes transmiten los parámetros resultantes al servidor central. En este paso, es crucial tomar medidas de seguridad como el cifrado, ya que, aunque no se envíen los datos en bruto, los parámetros pueden revelar información sensible.

3. **Agregación:** los modelos entrenados localmente son combinados mediante alguna técnica de agregación para generar un modelo global. El método más común es el *Federated Averaging* (*FedAvg*), que consiste en combinar los modelos entrenados localmente en cada cliente mediante un promedio de sus parámetros. Esta técnica solo es aplicable a modelos paramétricos (como redes neuronales). En el caso de los árboles de decisión, donde no tenemos parámetros que promediar, sino que el modelo basado en reglas de decisión de cada cliente solo puede aportar su estimación final al servidor de agregación, se utilizan técnicas de ensamblaje. Como se introdujo en la sección anterior, estas técnicas consisten en combinar múltiples árboles de decisión, en este caso, los generados por cada cliente. En este trabajo, la agregación se realizará mediante *bagging*, y consistirá en promediar las predicciones realizadas por cada árbol. Es decir, para cada valor a estimar, se obtendrá una estimación de cada uno de los árboles locales y se calculará la media de todas ellas como estimación final.

$$\hat{y}_{\text{bag}} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}^{(m)} \tag{2.4}$$

donde M es el número total de árboles agregados, $\hat{y}^{(m)}$ es la predicción realizada por el árbol m, y $\hat{y}_{\text{bag}}$ representa la predicción final del modelo global.

4. **Actualización local:** cada cliente actualiza su modelo con los parámetros del modelo global generado por el servidor tras la agregación y vuelve a realizar el entrenamiento, permitiendo un aprendizaje colaborativo sin compartir explícitamente los datos. El aprendizaje finaliza cuando se alcanza el número de rondas fijadas o se cumple algún criterio de parada.

Por otro lado, los modelos de aprendizaje federado pueden clasificarse en distintas categorías en función de cómo están distribuidos los datos entre los distintos clientes [19]:

- **Aprendizaje Federado Horizontal (Horizontal Federated Learning, HFL):** los clientes comparten las variables predictoras, pero tienen distintas muestras. Es lo más común y el escenario en el que se sitúa este proyecto, donde los distintos emplazamientos de la red móvil entrenarán modelos basándose en los mismos indicadores de red como variables de entrada (predictores).
- **Aprendizaje Federado Vertical (Vertical Federated Learning, VFL):** ocurre cuando los clientes comparten las mismas muestras, pero cada uno tiene variables predictoras distintas. Típico en escenarios realistas, como colaboración entre bancos u hospitales que tienen datos distintos sobre los mismos clientes o pacientes.
- **Aprendizaje Federado Transferido (Federated Transfer Learning, FTL):** es el escenario más complejo de aprendizaje federado. Se aplica cuando no hay coincidencia ni en las muestras ni en las características entre los participantes. Este enfoque pretende fusionar el aprendizaje entre dominios que, en principio, no se solapan. Suele aplicarse sobre modelos previamente entrenados con grandes conjuntos de datos centralizados.

En cuanto a la distribución de los datos, otro aspecto clave, además de conocer si los clientes comparten muestras o variables predictoras, es la naturaleza estadística de estos a nivel local.

En muchos entornos reales, los datos no están distribuidos de forma independiente e idénticamente distribuida (*Independent and Identically Distributed*, IID), sino que presentan distribuciones heterogéneas entre los distintos nodos (non-IID). En el escenario en el que se enmarca el proyecto, cada emplazamiento puede tener condiciones radio distintas o configuraciones dispares. Esta heterogeneidad supone un desafío para la agregación de los modelos, que pueden haber aprendido patrones completamente contradictorios. Diseñar modelos robustos frente a datos non-IID es uno de los principales retos del aprendizaje federado [19].

Capítulo 3

Diseño e implementación del modelo de estimación

Contenido

Sinopsis

En este capítulo se expondrá de forma general el problema concreto que aborda el trabajo, describiendo los diseños que se implementarán. A continuación, se introducirá el conjunto de datos sintético y su preprocesamiento, así

como el proceso de construcción de ambos modelos, centralizado y federado. Además, se explicarán los criterios de evaluación utilizados y el entorno, tanto *hardware* como *software*, en los que se han desarrollado las pruebas.

3.1. Descripción del diseño

El presente trabajo tiene como fin simular el flujo completo de un sistema de estimación del *throughput* medio de celda en una red móvil, comparando el rendimiento de un modelo entrenado de forma centralizada con el de su adaptación al enfoque federado. Para abordar este problema, se ha seguido una metodología estructurada en varias fases, que abarcan desde la obtención de los datos hasta la construcción y validación de los modelos. El flujo se ha implementado en dos variantes paralelas: una basada en el entrenamiento centralizado de la totalidad de los datos disponibles y otra distribuida mediante técnicas de aprendizaje federado. A continuación, se presentan los esquemas de los dos flujos de trabajo diseñados para el proyecto.

En primer lugar, la figura 3.1 muestra el enfoque clásico de aprendizaje automático, que sigue una estrategia de entrenamiento centralizado. Todos los datos disponibles recopilados por el OSS dentro de una red (en el experimento generados por un simulador) se someten a una fase de preprocesamiento para que sean compatibles con el entrenamiento. Además, se separa el juego de datos en tres subconjuntos, 2 serán utilizados para entrenar el modelo y controlar el proceso entrenamiento (subconjuntos de entrenamiento y validación), y 1 subconjunto será aislado para poder medir la calidad del modelo posteriormente sin sesgos (subconjunto de test). A continuación, empieza la elaboración del modelo, se trata de optimizar el entrenamiento mediante ajuste de hiperparámetros y selección de variables para reducir la dimensionalidad antes de entrenar. Finalmente, el modelo entrenado es evaluado sobre el subconjunto reservado para test, que contiene datos nunca vistos en entrenamiento.

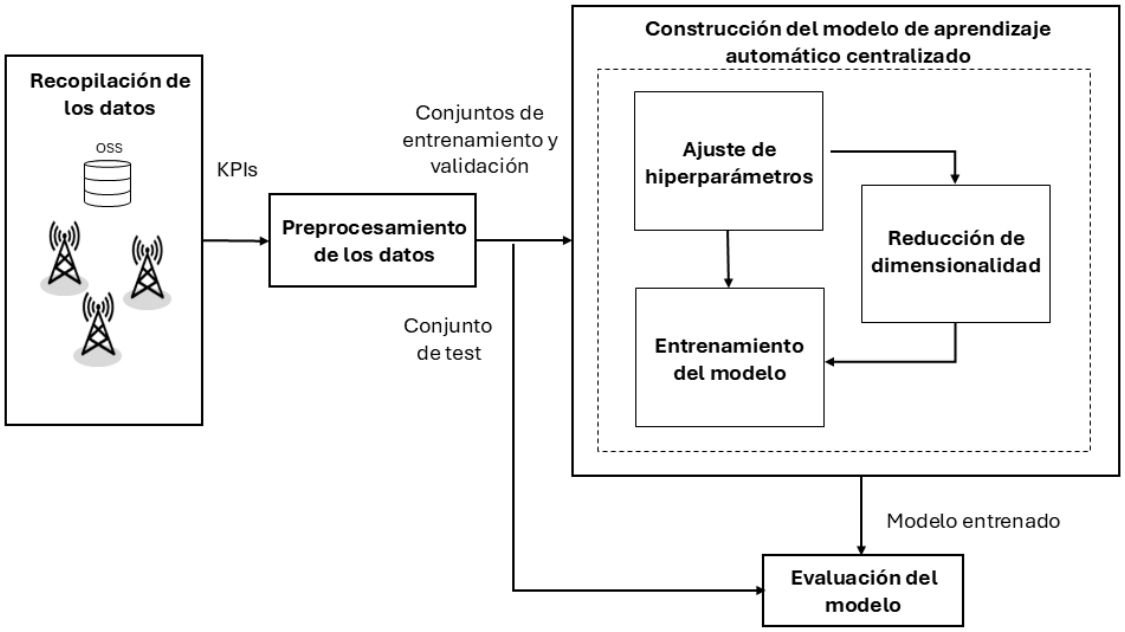

Figura 3.1: Diagrama de flujo del modelo centralizado (adaptado de [4]).

Por otro lado, la figura 3.2 recoge el flujo seguido por la versión descentralizada. En el esquema se representa un caso reducido de tres clientes, aunque el experimento completo se ha llevado a cabo con treinta y seis clientes. En este enfoque, los datos permanecen en su ubicación original, asociados en nuestro caso a emplazamientos de estaciones base. Cada cliente, que agrega la información de la estaciones base ubicadas en el mismo emplazamiento, preprocesa los datos y entrena el modelo localmente. Este proceso de entrenamiento, a pesar de ser representado en el diagrama como caja negra, encapsula los mismos pasos previos de ajuste de hiperparámetros y reducción de dimensionalidad que el modelo clásico. Los modelos entrenados por cada nodo se transfieren a un servidor central que los agrega, construyendo así un único modelo federado que ha aprendido de todos los clientes sin necesidad de que estos compartan información entre ellos. En cuanto a la evaluación, debido a la naturaleza académica del proyecto, se ha optado por combinar los conjuntos de test de cada cliente en una único subconjunto global. Aunque esta decisión está en contra de los principios clásicos del aprendizaje federado, al romper el aislamiento total entre

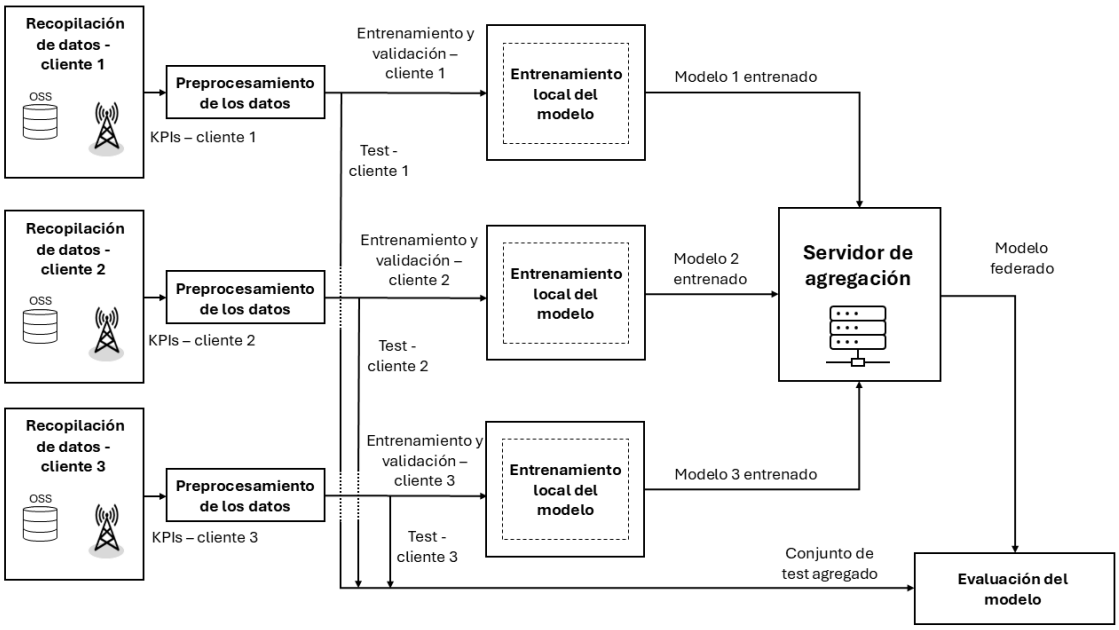

Figura 3.2: Diagrama de flujo del modelo federado.

clientes, permite evaluar el modelo final federado bajo las mismas condiciones que el modelo centralizado y facilita una comparación justa de ambos enfoques, que podrán ser testados sobre un espacio de prueba idéntico. En un entorno real, si se precisase de evaluación, sería necesario contar con un conjunto de test externo, independiente de los clientes, que no comprometiese la privacidad de los mismos. Durante toda la experimentación se ha asegurado que los conjuntos de datos usados en entrenamiento y en evaluación correspondan a las mismas muestras en ambos modelos para evitar sesgos en la evaluación.

3.2. Recopilación y preprocesado de datos

Esta sección detalla la obtención del conjunto de datos empleado para el desarrollo de las pruebas. Se expondrán brevemente los KPIs que han sido utilizados como variables predictoras y, posteriormente, el preprocesamiento que se ha aplicado sobre las variables.

3.2.1. Juego de datos sintético

En este proyecto se ha empleado un juego de datos de rendimiento sintético construido con un simulador dinámico de nivel de sistema que modela una red móvil LTE avanzada que ofrece 4 tipos de servicio: telefonía sobre IP (*Voice over IP*, VoIP), vídeo, transferencia de archivos (*File Transfer Protocol*, FTP) y navegación web. Las distintas funcionalidades se ofrecen además bajo diferentes configuraciones de carga y condiciones radio. Este enfoque permite obtener datos realistas y variados, representativos del comportamiento de una red real. Aunque el entorno de simulación corresponde técnicamente a una red LTE-A, el simulador ha sido configurado con parámetros y funcionalidades que permiten extender la validez de los experimentos a escenarios de nueva generación 5G [5].

El juego de datos generado está compuesto por 1259 muestras, donde cada una representa el comportamiento de una celda en el periodo de tiempo durante el que se promedia el rendimiento, conocido como periodo de reporte; en este trabajo ha sido de 15 minutos. A pesar de tratarse de un simulador, el periodo de reporte, en este caso, es idéntico al de redes móviles reales.

En la figura 3.3 puede observarse la distribución de los emplazamientos simulados. En total, el conjunto de datos abarca 108 celdas. El juego de datos incluye una hoja con la localización geográfica de cada celda, lo que permite agruparlas espacialmente por coordenadas en los 36 emplazamientos distintos que se observan en la imagen. Las muestras se consiguieron tras realizar 20 simulaciones, 10 para un ancho de banda de 5 MHz y otras 10 para 10 MHz. Cada simulación reproduce un escenario con una carga de tráfico distinta.

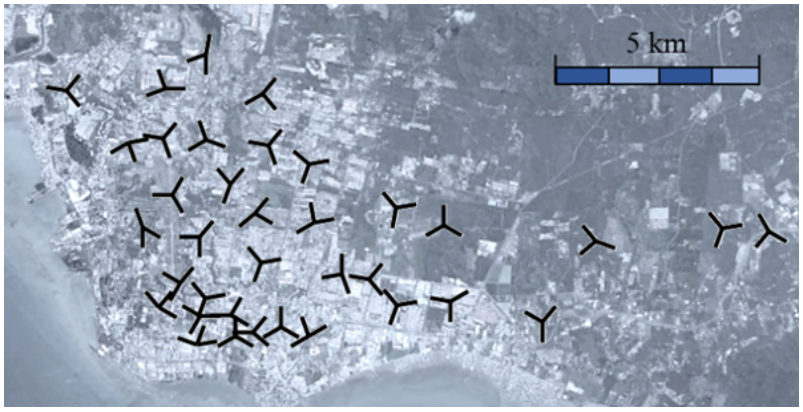

Figura 3.3: Ubicación de los emplazamientos en el escenario donde se extraen las estadísticas de rendimiento [5].

En las simulaciones con índices bajos, el tráfico generado es escaso, por lo que muchas estaciones base permanecen vacías, sin usuarios conectados ni tráfico activo. Esto impide generar muestras útiles para el modelo, ya que no hay información de rendimiento, dando lugar a muestras vacias en el conjunto de datos para algunas celdas. Como resultado, el número total de muestras varía entre simulaciones. Esta tendencia puede observarse en la figura 3.4, que enseña el número de muestras que tiene el juego de datos para cada índice de simulación. Se aprecia un incremento progresivo en el número de muestras a medida que aumenta el índice de simulación. A partir de la simulación 7, el número de muestras vuelve a disminuir. La figura 3.5 muestra la media del número de usuarios medios activos por simulación. De la imagen se deduce que en las simulaciones 6 y 7 se producen niveles de carga extremos, que probablemente generen un colapso en la red, reduciendo el número de celdas con tráfico y volviendo a aumentar el número de celdas vacias en la siguiente simulación.

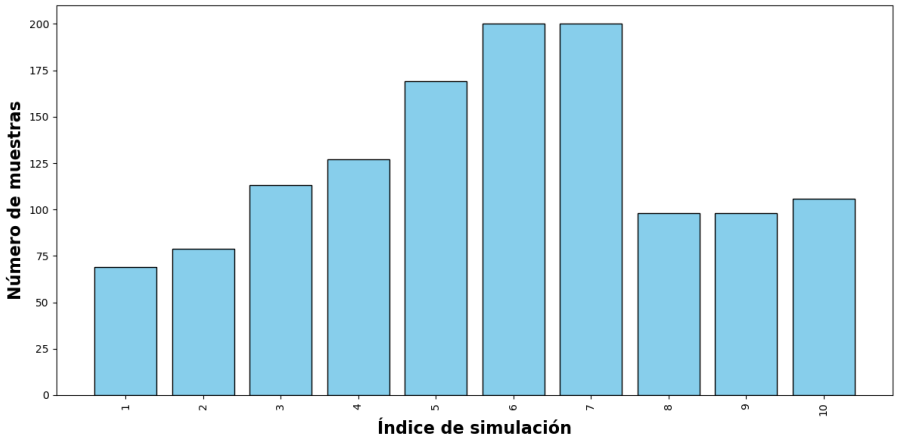

Figura 3.4: Histograma de muestras por simulación.

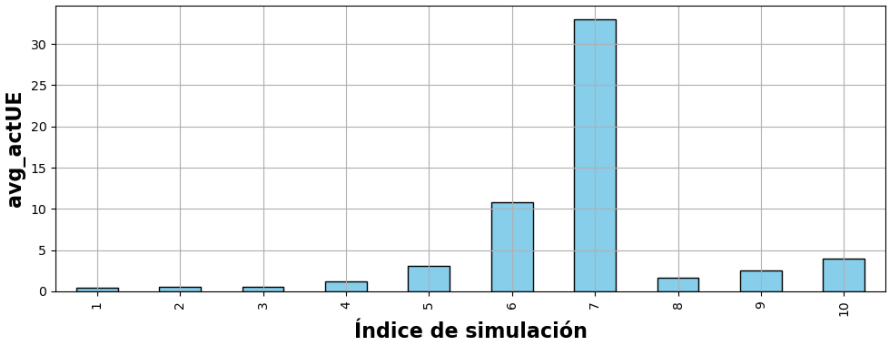

Figura 3.5: Número medio de usuarios activos.

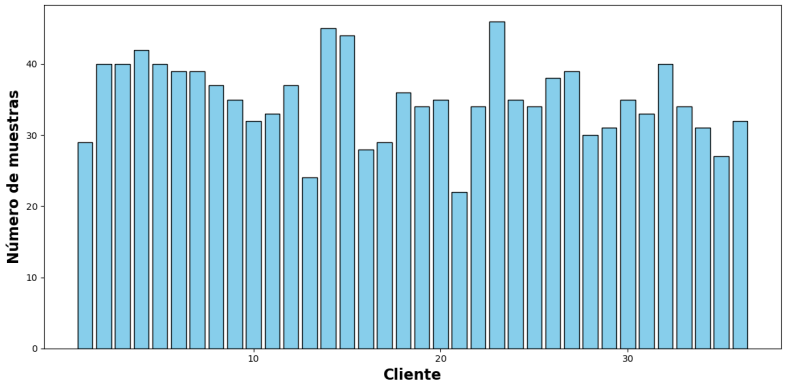

Figura 3.6: Histograma de muestras por cliente.

La figura 3.6 ilustra la cantidad de muestras de las que dispone cada uno de los 36 clientes, en forma de histograma. En el entorno federado cada nodo dispone de entre 22 y 46 muestras para su entrenamiento local, una cantidad reducida. Esta heterogeneidad se debe a la distribución desigual de la cantidad de muestras entre simulaciones, provocada por cargas de red dispares, como se explicó previamente.

Cada muestra incluye 10 variables predictoras que recogen KPIs relevantes para la estimación del *throughput*. Se han generado métricas estadísticas relacionadas con la gestión del rendimiento (*Performance Management*, PM), la gestión de la configuración (*Configuration Management*, CM) y trazas de conexión radio, siguiendo un enfoque similar al que se implementaría en un entorno real sobre datos extraídos del OSS. La tabla 3.1 presenta las características recogidas por el simulador. La variable objetivo es el *trhoughput* de enlace descendente (*Downlink*, DL) promedio por celda, medido como el volumen total de datos transmitidos en la capa de protocolo de convergencia de datos en paquetes (*Packet Data Convergence Protocol*, PDCP) por unidad de tiempo, expresado en kbps.

La variabilidad introducida por las distintas simulaciones, así como las diferencias estructurales entre celdas y emplazamientos, permiten conseguir un conjunto de datos para evaluar la robustez del modelo centralizado, como se demostró en [5]. El principal desafío radica en alcanzar un rendimiento competitivo en el escenario federado, con la limitación del tamaño reducido y la heterogeneidad de los 36 subconjuntos de datos segregados.

Tipo de indicador	Nombre	Descripción
Calidad de señal	`avg_CQI`	Media de la distribución de indicador de calidad del canal (*Channel Quality Indicator*, CQI) en DL en la celda
	`median_CQI`	Mediana de la distribución de indicador de CQI en DL en la celda
	`p5_CQI`	Percentil 5 de la distribución de CQI en DL en la celda
Tráfico	`avg_actUE`	Número medio de usuarios (*User Equipments*, UEs) activos por intervalo de tiempo de transmisión (*Transmission Time Interval*, TTI) en DL
	`PRButil_rat`	Ratio de utilización de bloques de recursos físicos (*Physical Resource Blocks*, PRB) en canal compartido de enlace descendente físico (*Physical Downlink Shared Channel*, PDSCH) en la celda
	`s_UE_rat` $\forall s \in \{$`VoIP, video, ftp, web`$\}$	Proporción de UEs que demandan cada servicio s en la celda
Configuración del canal	`cell_BW` [MHz]	Ancho de banda configurado para la celda

Tabla 3.1: Descripción de las principales características del juego de datos.

3.2.2. Preprocesamiento

Una vez definido y recopilado el conjunto de datos, se realiza un preprocesamiento de estos para adaptarlos a un formato válido para el entrenamiento de los modelos de aprendizaje automático. El proceso seguido ha sido idéntico en ambos escenarios, asegurando que la comparación entre ellos se lleve a cabo bajo condiciones equivalentes.

Escalado

En primer lugar, las variables han sido estandarizadas mediante la técnica de normalización Z-score. Consiste en reescalar los datos para que tengan media

cero y varianza unitaria, siguiendo la fórmula:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

donde x es el valor original de la característica (predictor o variable objetivo), μ la media de la característica, y σ su desviación estándar.

Esta técnica evita que las variables con mayor valor numérico o rangos dominen el proceso de aprendizaje. Sin este paso, en el juego de datos analizado el modelo podría sesgarse a favor de las métricas de CQI o ancho de banda, que presentan valores mucho mayores que el resto de indicadores. Además, la estandarización ayuda a la estabilidad del modelo y a técnicas como la selección de atributos relevantes para la reducción de dimensionalidad.

En el caso centralizado, se aplica el escalado sobre todo el subconjunto del juego de datos completo que será utilizado para entrenar el modelo, mientras que en el escenario federado se aplica el escalado de forma aislada a los datos de entrenamiento y validación de cada cliente, siguiendo la lógica de privacidad entre emplazamientos.

División del juego de datos

A continuación, se divide el juego de datos en tres subconjuntos: entrenamiento (64 %), validación (16 %) y test (20 %). Esta división se ha realizado de forma aleatoria pero reproducible, utilizando una semilla fija para garantizar la consistencia en los experimentos.

En el enfoque centralizado, la partición de los datos se aplica directamente sobre la totalidad del juego de datos. En el federado, cada nodo accede únicamente a los datos asociados a su emplazamiento y sobre este subconjunto realiza la división local con los mismos porcentajes. Para asegurar una comparación justa de los dos modelos finales, los índices de las muestras asignadas a cada función (entrenamiento, validación y test) han sido almacenados por nodo para forzar que cada muestra desempeñe el mismo papel tanto en el modelo centralizado como en el distribuido.

Gracias a este control se garantiza que los modelos centralizado y federado se evalúen exactamente sobre el mismo subconjunto de muestras, descartando posibles sesgos en la comparación derivados de diferencias en la selección de los datos de prueba.

3.3. Modelos

En el capítulo 1 se introdujeron los árboles de decisión como modelo de referencia. Además, se presentó XGBoost como la versión concreta de árboles que se ha empleado, resaltando su papel en el estado de la técnica en el campo del trabajo. En esta sección, se detallarán los fundamentos teóricos del modelo XGBoost, tanto para la versión centralizada como para su adaptación federada. También se expondrá la configuración necesaria para su funcionamiento en ambas modalidades, de forma general, sin entrar en los valores de configuración elegidos.

3.3.1. Construcción del modelo centralizado

En este apartado se profundiza en el modelo elegido, XGBoost, en su versión centralizada. En primer lugar, se presentan sus fundamentos teóricos. Posteriormente, el ajuste de hiperparámetros necesario y la selección de características para la reducción de dimensionalidad.

Modelo de referencia - XGBoost

El *Extreme Gradient Boosting*, conocido como XGBoost, ha sido el modelo utilizado para llevar a cabo los experimentos. Como se introdujo en la sección 1.3, la elección del modelo se fundamenta tanto en su rendimiento ampliamente contrastado en la literatura como en los resultados previos obtenidos en [5], donde demostró superar a los modelos de aprendizaje automático clásicos y competir con el aprendizaje profundo.

XGBoost es un modelo de aprendizaje supervisado que se basa en árboles de decisión y utiliza la técnica de ensamblaje conocida como *gradient boosting*. Su funcionamiento se visualiza en la figura 3.7. Este método entrena varios modelos débiles, en este caso árboles de decisión pequeños, de forma secuencial, corrigiendo los errores cometidos por los árboles anteriores. En cada iteración, el modelo calcula el gradiente de la función objetivo, y ajusta los nuevos árboles para reducir dicho valor de error. La predicción final es la combinación ponderada de la salida de todos los árboles generados.

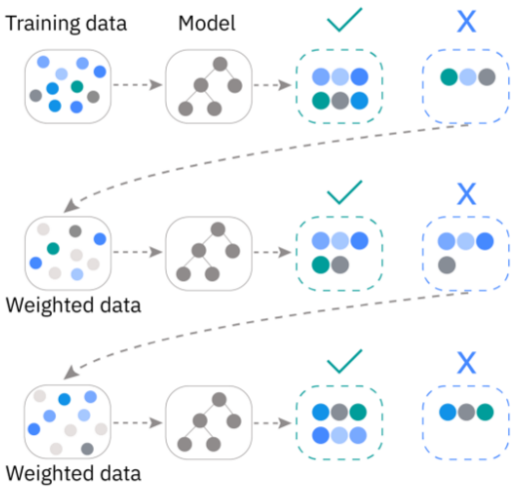

Figura 3.7: Arquitectura del modelo XGBoost [6].

Su implementación incorpora varias mejoras que lo diferencian de otros modelos de árboles de decisión [26]:

- **Regularización explícita** para evitar el sobreajuste. Incorpora regularización L1 y L2.
- **Aumento de la eficiencia en la búsqueda de divisiones *(splits)*:** Gracias a la incorporación del algoritmo W*eighted Quantile Sketch*, que reduce el número de valores umbrales a probar y a la optimización en paralelo, que permite que la búsqueda de *splits* se haga de forma paralela en las distintas ramas del árbol.
- **Submuestreo ajustable de columnas** para mejorar la generalización.
- ***Early stopping:*** Implementa criterios de parada basados en el rendimiento del modelo sobre el conjunto de validación cuando encuentra el punto óptimo sin necesidad de crear todos los árboles planeados. Para ello, supervisa si la deja función de pérdidas deja de mejorar durante un número determinado de iteraciones (árboles creados) consecutivas.

Una de las claves del rendimiento de XGBoost es la ya mencionada regularización explícita. Junto a la función de pérdidas, conforma una función objetivo eficaz a la hora de evitar el sobreajuste. Esta función objetivo se define como [27]:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3.2}$$

donde el primer término se refiere a las pérdidas y el segundo a la regularización.

$\Omega(f)$ se define como:

$$\Omega(f_m) = \gamma T_m + \frac{1}{2}\lambda \|w_m\|_2^2 + \alpha \|w_m\|_1 \tag{3.3}$$

Cada término de la función cumple una labor específica a la hora de controlar la complejidad del modelo:

- γT_m: penaliza el número de hojas T_m del árbol m, fomentando árboles más simples.
- $\frac{1}{2}\lambda \|w_m\|_2^2$: término de regularización L2, que penaliza valores grandes en los pesos de las hojas mediante la norma euclídea cuadrática, contribuyendo a suavizar el modelo.
- $\alpha \|w_m\|_1$: término de regularización L1, que también reduce los pesos, llegando a forzar que sean 0. Con una correcta configuración ayuda a podar nodos que no aportan información relevante.

En cada iteración del boosting se intentan corregir los errores introducidos por el último árbol. Esto se tiene en cuenta en la función de pérdidas del nuevo árbol, tomando la función objetivo la siguiente forma en la iteración t [26]:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \tag{3.4}$$

donde $\hat{y}_i^{(t-1)}$ representa la predicción acumulada hasta la iteración $t-1$, y $f_t(\mathbf{x}_i)$ es la nueva predicción generada por el árbol f_t. De esta manera, XGBoost, en lugar de generar un modelo complejo de una sola vez, mejora conforme crea más árboles. Este enfoque consigue aumentar la precisión global del modelo mientras se mantiene cada árbol individual relativamente simple, reduciendo el riesgo de sobreajuste en cada árbol.

Configuración del modelo

En este subapartado se explica como se configura la versión centralizada. En primer lugar se detallará el ajuste de hiperparámetros y, posteriormente, la reducción de dimensionalidad.

Para contrarrestar su tendencia al sobreajuste, XGBoost requiere de un fino **ajuste de sus hiperparámetros**. Esta tarea se ha llevado a cabo mediante la técnica de *grid search* [28]. Primero se establece un conjunto de hipeparámetros con un rango de posibles valores. Luego, se entrena el modelo con todas las posibles combinaciones de esos hiperparámetros, buscando la combinación que minimice la pérdida del modelo.

Se ha complementado con la técnica de validación cruzada (*Cross-Validation*, CV) con $k = 5$ particiones, ampliamente utilizada en el aprendizaje automático, que consiste en dividir el conjunto de datos de entrenamiento en k particiones *(folds)*, usar todas menos una para el entrenamiento y la restante para validación. Se entrena el modelo k veces alternando el subconjunto que se utiliza para validar y se considera el promedio de las k pérdidas para calificar el rendimiento de una combinación determinada de hiperparámetros, reduciendo el riesgo de sobreajuste.

A continuación, se enumeran los hiperparámetros que intervienen en el proceso de aprendizaje de *Extreme Gradient Boosting* [27]:

- **Número total de árboles a entrenar (`n_estimators`):** un valor demasiado bajo puede provocar que el modelo no aprenda bien los patrones, mientras que entrenar un número excesivo de árboles aumentará el coste computacional innecesariamente. Valores evaluados: 50, 100, 150 y 200.
- **Profundidad máxima de cada árbol (`max_depth`):** controla la complejidad del modelo; árboles más profundos capturan mejor el comportamiento de los datos, pero tienden a sobreajustarse. Valores evaluados: 5 y 10.
- **`learning_rate` (η):** pondera la contribución de cada nuevo árbol al modelo total. Valores evaluados: 0.01 y 0.1.
- **`reg_alpha` (α):** regularización L1 para reducir el número de nodos irrelevantes. Valores evaluados: 0.01, 0.1, 50 y 100.
- **`reg_lambda` (λ):** regularización L2 para suavizar los pesos. Valores evaluados: 0.01, 0.1, 50 y 100.
- **`gamma` (γ):** Controla la mejora necesaria para realizar una división y crear dos nuevos nodos. Ayuda a controlar la complejidad evitando particiones que no generan mejoras significativas. Valores evaluados: 0.01, 0.1, 20, 50 y 100.

Este proceso de búsqueda es fundamental para optimizar el rendimiento del modelo y asegurar su capacidad de generalización. Sin embargo, también supone un coste computacional considerable, ya que requiere entrenar el modelo

múltiples veces con distintas combinaciones de hiperparámetros. Por ello, la búsqueda se ejecuta el menor número posible de veces y, una vez encontrados los hiperparámetros óptimos, se fijan para las siguientes pruebas y ejecuciones.

Como cierre para este apartado de configuración del modelo centralizado, se detalla la **reducción de dimensionalidad** empleada.

Los modelos de aprendizaje supervisado pueden ver su rendimiento perjudicado si las variables de entrada tienen alta correlación entre sí o si no son relevantes para el comportamiento de la variable de salida. Además, la presencia de datos que podrían omitirse afecta negativamente a la eficiencia computacional del modelo y, en entornos reales, supone un aumento de costes, por ejemplo, en grandes bases de datos.

Como consecuencia, la reducción de dimensionalidad emerge como una herramienta crítica en la elaboración de modelos de aprendizaje automático. Siguiendo la línea de [5], esta tarea se ha realizado mediante selección de características, una técnica que permite eliminar las variables irrelevantes del juego de datos. Concretamente, se ha usado *Recursive Feature Elimination* (RFE) [22]. RFE empieza entrenando el modelo con todos los candidatos, es decir, el juego de datos completo, y va eliminando el menos relevante acorde a una función de pérdidas establecida previamente. El proceso se repite hasta que se han eliminado todos los candidatos, de forma que, en cada ronda, las características que no han sido eliminadas son más relevantes para la predicción de la variable objetivo que las que no han sido seleccionadas.

La selección de características, al igual que el *grid search*, ha sido combinada con *cross validation* (k=5). Por otro lado, la función de pérdidas que se ha considerado para RFE ha sido el error absoluto medio normalizado (*Mean Absolute Normalized Error*, MANE), una cifra de mérito sobre la que se profundizará en el apartado 3.4. Se tomará como válida la reducción de dimensionalidad (modelo simplificado) cuando suponga una pérdida de menos del 2 % de precisión con respecto a la versión sin selección de características (modelo completo).

3.3.2. Construcción del modelo federado

En este apartado se describe el proceso seguido para la elaboración del modelo federado basado en árboles de decisión del tipo XGBoost. Ya presentados los fundamentos teóricos de XGBoost, este apartado detalla su adaptación el enfoque distribuido, el ajuste de hiperparámetros necesario y el criterio elegido para la reducción de dimensionalidad.

Modelo de referencia XGBoost en su versión federada

El modelo federado se trata de un modelo global construido a partir de contribuciones de múltiples modelos individuales. Cada uno de los modelos individuales entrenados en los distintos clientes es un modelo basado en árboles de decisión del tipo XGBoost, cuyos fundamentos ya se han expuesto. Los siguientes párrafos profundizarán en la arquitectura del modelo federado utilizado y la técnica de agregación elegida.

El modelo federado utilizado en este proyecto sigue una estructura de tipo cliente-servidor. Cada cliente representa un emplazamiento de red con su propio proceso de entrenamiento local, mientras que el servidor central coordina el entrenamiento a nivel global, encargándose de la agregación de modelos y aplicando *early stopping*. Se ha optado por una estrategia de agregación del tipo *bagging*, una técnica de ensamblaje descrita previamente en la sección 2.1.3. Aunque XGBoost utilice *boosting* a nivel local, este ensamblaje secuencial queda limitado al número de árboles que se permitan entrenar en cada cliente. El *bagging* es utilizado para coordinar los árboles generados en cada nodo. En conjunto, puede considerarse que el modelo federado presenta un funcionamiento híbrido *boosting-bagging*.

En la estrategia elegida, durante cada ronda de entrenamiento, todos los clientes entrenan localmente un número reducido de árboles, normalmente uno o dos. Esto contrasta con el enfoque centralizado, donde el modelo se entrena con cientos de árboles, el federado alcanzará estas cantidades de árboles pero una vez realizadas varias rondas de entrenamiento. Esta elección pretende evitar el sobreajuste a conjuntos locales pequeños y busca una convergencia más gradual a lo largo de las rondas, favoreciendo la generalización del modelo global.

Al finalizar cada ronda, los árboles entrenados localmente se envían al servidor central, que no realiza un promedio de parámetros, como sí hacen otros métodos típicos de aprendizaje distribuido como el *Federated Averaging*, sino que concatena los árboles generados por todos los nodos. De este modo, el modelo va creciendo progresivamente en cada ronda.

La predicción final del modelo federado con agregación *bagging* se obtiende promediando las salidas de todos los árboles acumulados al finalizar el entrenamiento. La figura 3.8 ilustra este proceso de predicción, donde $f_k(x)$ simboliza la decisión individual del árbol k para unas variables de entrada concretas. El entrenamiento se da por finalizado cuando se alcanzan el número máximo de rondas de entrenamiento establecidas como hiperparámetro o cuando se activa el criterio de parada marcado por el *early stopping*. Esta técnica de regularización es

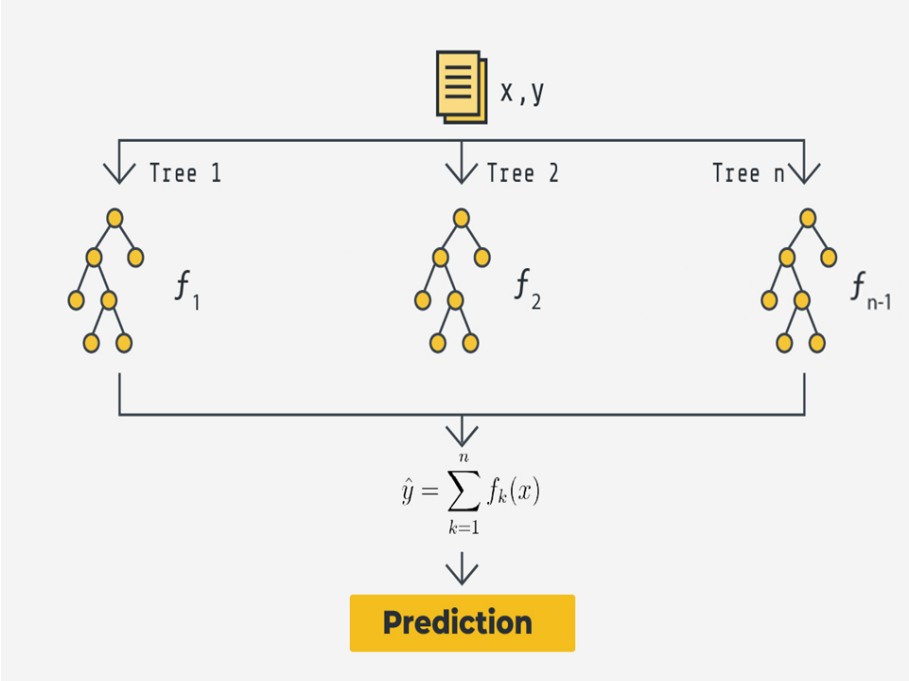

Figura 3.8: Predicción con estrategia de agregación tipo *bagging* [7].

gestionada por el servidor central, que, cuando detecta un número de rondas seguidas sin mejoras en la precisión, evaluada sobre la función de pérdidas en los conjuntos de validación, detiene el entrenamiento del modelo.

Configuración del modelo

Es fundamental afinar en el **ajuste de hiperparámetros** en el escenario federado del proyecto, ya que son la principal herramienta de la que se dispone durante la configuración del modelo para mitigar las limitaciones dadas por el reducido tamaño de los conjuntos de datos locales (pocas celdas por emplazamientos, pocos eventos de saturación de celda) y la heterogeneidad entre ellos (emplazamientos de características muy diferentes).

Esta elección se ha llevado a cabo con un enfoque práctico orientado a la mejora de la eficiencia. Dado que el escenario distribuido implica el entrenamiento en paralelo de 36 modelos por ronda, lo que incrementa considerablemente el coste computacional, se optó por definir manualmente un conjunto limitado de combinaciones de hiperparámetros, seleccionadas de forma intencionada para cubrir un rango amplio de valores relevantes. Esta búsqueda inicial permitió acotar las configuraciones más prometedoras. Posterioremente se terminó de pulir

la elección ajustando también manualmente las combinaciones de hiperparámetros que mejor rendimiento mostraban. Para garantizar la coherencia del modelo y reducir la complejidad, esta búsqueda se realizó de forma común para todos los nodos, aplicando los mismos valores de hiperparámetros para todos ellos.

Los hiperparámetros utilizados en el modelo federado han sido, en su mayoría, comunes a los del enfoque centralizado. El entrenamiento local de cada cliente sigue siendo basado en árboles de decisión XGBoost, por lo que parámetros como la profundidad máxima de los árboles o los coeficientes de regularización L1 y L2 tienen el mismo significado. No obstante, requerirán de ajustes específicos para alcanzar el rendimiento óptimo del modelo distribuido.

Los hiperparámetros particulares del modelo federado son los propios parámetros de control del ciclo distribuido:

- **Número de rondas federadas** `(num-server-rounds)`**:** marca el número máximo de iteraciones globales de entrenamiento y agregación.
- **Número de nodos participantes** `(num-supernodes)`**:** se fija en 36, coincidiendo con el número total de emplazamientos disponibles en el conjunto de datos. Queda fuera del proceso de búsqueda del *grid search*.
- **Rondas de paciencia para *early stopping*** `(pacience-rounds)`**:** se ha implementado un criterio de parada anticipada en el servidor, que detiene el entrenamiento si no se observa mejora en la métrica de evaluación tras un número determinado de rondas consecutivas.

Los valores específicos finalmente seleccionados para cada hiperparámetro se detallan en el capítulo 4, junto con los resultados obtenidos en los experimentos.

En cuanto a la **reducción de dimensionalidad**, en el modelo federado no se ha aplicado explícitamente el algoritmo de selección de características durante el proceso de entrenamiento. Al situarnos en un escenario de aprendizaje federado horizontal, todos los clientes comparten las variables de entrada, que además son particiones del conjunto de datos global utilizado en el entrenamiento centralizado. Es por ello que se ha optado por reutilizar directamente las características seleccionadas por RFE en el enfoque centralizado.

El objetivo en este caso, no es simplemente mejorar la eficiencia computacional sacrificando precisión, si no que se pretende mejorar la calidad de las estimaciones del modelo. Esto puede parecer contraintuitivo, sin embargo, la reducción de dimensionalidad es especialmente beneficiosa en el modelo federado de este proyecto ya que cada nodo trabaja con cantidades reducidas de muestras. Esto

puede conducir fácilmente al sobreajuste, memorizando patrones concretos de conjuntos pequeños de datos. Limitando el número de variables se reduce este riesgo a nivel local y se mejora la estabilidad del proceso.

El proceso de reducción de dimensionalidad será el siguiente: durante la fase experimental, se prueba eliminando las variables de entrada en el mismo orden que en el centralizado. Tras evaluar 10 modelos con dimensionalidad reducida, se escoge la cantidad de características que demuestra crear un modelo con menores pérdidas y con este modelo se continua con las pruebas.

3.4. Criterios de evaluación

Los criterios que se han seguido para evaluar la calidad predictiva de los modelos han sido principalmente las cifras de mérito y el análisis gráfico.

3.4.1. Cifras de mérito

Son indicadores que miden el rendimiento de un modelo de aprendizaje automático.

A continuación, se describen las cifras de mérito utilizadas para evaluar la precisión de los modelos en este proyecto [29]:

- **MAE (Mean Absolute Error):** la media de las diferencias entre el valor real y el pronosticado en cada predicción. Sin escalar, da un valor intuitivo del error promedio cometido en las mismas unidades que la variable de salida.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3.5}$$

- **RMSE (Root Mean Squared Error):** mide la raíz cuadrada del error cuadrático medio. Penaliza más los errores grandes que el MAE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.6}$$

- **R^2 (Coeficiente de determinación):** métrica que indica la proporción de la varianza de la variable de salida que es capaz de explicar el modelo, es decir, cómo de bien se ajusta un modelo a los datos. Si es 1 el modelo se

ha adaptado a la perfección a los datos, si es 0 el modelo es totalmente azaroso y si es negativo se tiene un modelo que predice peor que el azar.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.7}$$

- **MAPE (Mean Absolute Percentage Error):** expresa el error promedio absoluto en relación con el valor real del pronóstico, expresado en porcentaje:

$$MAPE\,[\%] = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{3.8}$$

  Adecuada cuando la MAE o el RMSE pueden ocultar información relativa a lo proporcinalmente grande que es el error en relación con el valor real. Es una métrica con limitaciones en escenarios donde la variable objetivo puede tomar valores cercanos a 0.

- **MANE (Mean Absolute Normalized Error):** es una versión de la MAE normalizada por el valor máximo teórico de la variable de salida.

$$MANE\,[\%] = \frac{1}{N_s}\sum_{i=1}^{N_s}\left(100 \cdot \left|\frac{\hat{y}(i) - y(i)}{TH_{\max}(k_i)}\right|\right) \tag{3.9}$$

  Siguiendo el criterio marcado por el estudio [5], el *throughput* máximo teórico se calcula considerando que todos los recursos del canal (PRBs) son asignados con la mejor calidad posible (CQI = 15). Con el esquema de modulación y codificación más alto implementado en el simulador, la tasa de datos máxima por PRB es 1 Mbps. Con esta información, el *throughput* máximo se calcula cómo el producto del número de PRBs por 1 Mbps. En el caso de LTE-A, el número de PRBs disponibles por canal depende del ancho de banda, siguiendo la correspondencia de la siguiente tabla 3.2:

Ancho de banda	**PRBs**
1.4 MHz	6
3 MHz	15
5 MHz	25
10 MHz	50
15 MHz	75
20 MHz	100

Tabla 3.2: Relación entre el ancho de banda del canal y el número de PRBs [8].

Esta métrica de evaluación relativa permite una comparación más justa entre celdas con distintas capacidades. Al normalizar por el valor máximo teórico, se evita que las celdas con mayor ancho de banda dominen injustamente el cálculo del promedio del error absoluto, lo que podría enmascarar deficiencias en la predicción en celdas con menor capacidad. Cada predicción es evaluada en relación con su propio límite superior teórico.

- **Trimmed MAPE y MANE:** Estas métricas permiten reducir el impacto de valores atípicos (*outliers*) que afecten desproporcionadamente al rendimiento. son versiones recortadas de las anteriores, donde se eliminan las muestras que originan los errores extremos.

3.4.2. Análisis gráfico

Una herramienta clave para evaluar el rendimiento de todo modelo de aprendizaje automático que realice estimaciones es la representación gráfica de los valores predichos frente a los valores reales. En este proyecto, se han utilizado principalmente grafos de dispersión, como el presentado en la figura 3.9, donde las estimaciones se reprensentan sobre el eje y, mientras que los valores reales se representan en el eje x. En la gráfica se incluye una línea diagonal de referencia que simboliza la precisión total. Cuanto más cercanos se encuentren los puntos a dicha línea, mayor será la precisión del modelo. Esta visualización es especialmente útil para identificar patrones de sesgo, variabilidad en las predicciones y la posible presencia de valores atípicos de manera intuitiva.

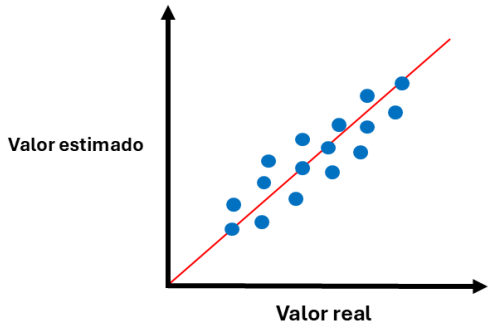

Figura 3.9: Grafo de dispersión genérico.

Otra técnica gráfica que se ha usado son las curvas de convergencia. Estas curvas constituyen una herramienta muy útil para monitorizar la adaptación del modelo a los datos durante la fase de entrenamiento, sin necesidad de recurrir al conjunto de datos reservado para el test. La figura 3.10 muestra un ejemplo de curvas de convergencia. Su funcionamiento e interpretación ya se ha descrito en detalle en el apartado 2.1.2 del capítulo 2. En este diseño, representarán la evolución de la función de pérdida a lo largo del número de iteraciones del modelo: por árbol en el caso centralizado y por ronda en el federado.

Una última representación gráfica útil en este trabajo es la curva de distribución acumulada (*Cumulative Distribution Function*, CDF) del error absoluto en las estimaciones. La figura 3.11 ilustra este tipo de gráficas. Esta curva permite analizar visualmente cómo se distribuyen los errores cometidos por el modelo. En concreto, representa la proporción acumulada de muestras cuyo error absoluto es menor o igual a un determinado valor. Además, permite identificar la presencia de errores extremos observando el comportamiento de la curva en los valores altos del eje de errores: si la curva se aplana y tarda en alcanzar el valor 1, indica que existe más porcentaje de muestras con errores elevados. Por el contrario, si la curva asciende rápido hacia el valor 1 con errores bajos, la mayoría de las predicciones son precisas, lo que indica un mejor rendimiento general del modelo.

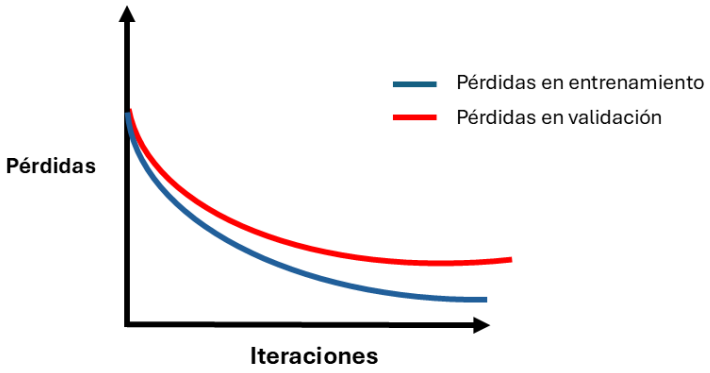

Figura 3.10: Curvas de convergencia genéricas.

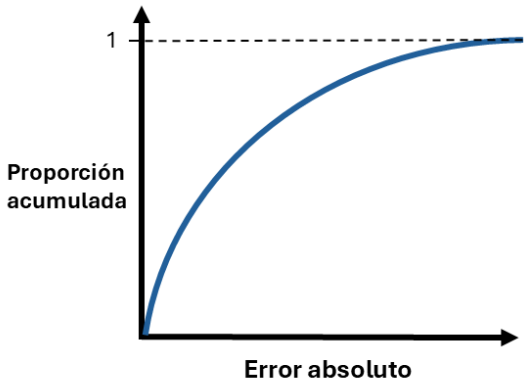

Figura 3.11: CDF de error absoluto.

3.5. Entorno de desarrollo

Finalmente, se presenta el entorno en el que han sido ejecutadas las simulaciones, tanto a nivel de *software* como de *hardware*.

3.5.1. Software

El desarrollo de los experimentos se ha realizado en el lenguaje de programación **Python** en su versión 3.12.7, utilizando la plataforma de gestión de entornos y paquetes **Anaconda**, un potente ecosistema que facilita la reproducibilidad de los experimentos. El entorno de desarrollo elegido ha sido **Spyder**, donde se ha escrito y depurado el código, además de reutilizar repositorios públicos de *GitHub* para la implementación del caso federado.

La gran variedad de librerías específicas que ofrece ha consolidado a Python como uno de los lenguajes de programación más utilizados en el ámbito del aprendizaje automático. A continuación, se enumeran las utilizadas en este proyecto:

- **Scikit-learn** (`sklearn`): para preprocesamiento de datos, *grid search*, RFE

y validación cruzada.

- **Xgboost**: para la implementación del modelo de árboles de decisión.
- **Pandas** y **NumPy**: para el manejo y análisis de datos.
- **Matplotlib**: para la visualización de resultados.
- **Flower (FLWR)**: como *framework* de aprendizaje federado que permite simular múltiples clientes y la interacción de estos con un servidor central dentro de un entorno local.

3.5.2. Hardware

Todos los experimentos desarrollados en este trabajo han sido ejecutados desde un mismo ordenador portátil personal, asegurando así la consistencia entre los distintos experimentos y una base común para la comparación de los tiempos de ejecución. Esto puede ser importante para comparar la eficiencia computacional de los modelos, especialmente en un escenario como el del proyecto, donde el modelo federado implica entrenamiento de 36 nodos en paralelo.

- **Procesador (CPU):** 12th Gen Intel(R) Core(TM) i7-12650H) 2.3 GHz
- **Memoria RAM:** 16 GB
- **Sistema operativo:** Windows 11 Home, versión 24H2 (64 bits)
- **GPU:** NVIDIA GeForce RTX 3050 Laptop GPU

  Aunque el equipo cuenta con GPU dedicada, no se ha habilitado su uso durante los experimentos. Muchas de las librerías utilizadas no usan GPU en sus versiones básicas, y no ha sido necesario para la viabilidad del entrenamiento de los modelos, que se ha realizado exclusivamente en CPU.

Capítulo 4

Pruebas

Contenido

Sinopsis

El objetivo principal de este capítulo es comparar ambos modelos. Se pretende valorar la aplicabilidad del modelo federado en escenarios donde el aprendizaje centralizado no sea viable por limitaciones de privacidad.

Para ello, se presenta el desarrollo de las pruebas. En primer lugar, se introduce la metodología experimental seguida para ambos escenarios, empezando por una breve descripción de los experimentos. A continuación, se detallan los procesos de entrenamiento de los dos modelos. Finalmente, se cierra el capítulo exponiendo los resultados obtenidos sobre el conjunto de test y los tiempos de ejecución.

4.1. Metodología experimental

Esta sección explica el procedimiento experimental al completo. Se parte de una descripción general y, posteriormente, se detallan los procesos de entrenamiento.

4.1.1. Descripción de los experimentos

En este apartado se ofrece una breve descripción de los dos experimentos llevados a cabo, como introducción al desarrollo posterior. La tabla 4.1 contiene un pequeño resumen del experimento realizado, mostrando el conjunto de datos, metodos de evaluación y cifras de mérito seleccionadas para evaluar. La tabla no distingue entre ambos experimentos ya que se ha seguido una metodología idéntica en ambos en cuanto a la evaluación. Se quiere comparar el rendimiento de los modelos, por lo que se deben seguir procedimientos lo más homogéneos posibles.

Conjunto de datos	**Métodos de evaluación**	**Cifras de mérito**
Sintético (red LTE-A simulada)	Cifras de mérito, grafos de dispersión y CDF	MAE, RMSE, R^2, MANE y trimmed MAPE

Tabla 4.1: Resumen de los experimentos realizados.

Para la evaluación de los resultados se han tenido en cuenta la mayoría de las cifras de mérito introducidas en el capítulo 3: MAE, RMSE, R^2, MANE y *trimmed* MAPE. Esta elección permite obtener una imagen variada y representativa del rendimiento de los modelos. Se tienen métricas de error absoluto (MAE y RMSE), con capacidad explicativa de la adaptación del modelo a la variabilidad de los datos (R^2), una métrica relativa a la capacidad máxima de *throughput* de cada celda (MANE) y otra relativa al valor real de la predicción, a la vez que robusta frente a valores atípicos (*trimmed* MAPE). Además, se emplea un análisis gráfico que muestra la calidad de las estimaciones, como son los grafos de dispersión, y otro referente a la distribución de los errores cometidos, la CDF.

Por su parte, el entrenamiento de los modelos sigue una metodología muy similar en ambos casos: ajuste de hiperparámetros, reducción de dimensionalidad y construcción de los árboles. Además, durante la fase de entrenamiento se utilizan representaciones gráficas que permiten visualizar la estabilidad de

cada enfoque; las curvas de convergencia. El proceso de entrenamiento federado seguido, aunque en su base sea idéntico al centralizado, difiere en algunos aspectos clave como la incorporación mecanismos de agregación mediante ensamblaje tipo *bagging*. Por otro lado, debido a la complejidad del escenario distribuido, la reducción de dimensionalidad se basa en los resultados obtenidos previamente en el modelo centralizado y el ajuste de hiperparámetros se lleva a cabo mediante estrategias más simples: exploración manual sobre un conjunto de combinaciones seleccionado intencionadamente. Aunque este enfoque resulta menos riguroso, ha sido suficiente para evaluar el rendimiento del modelo y obtener conclusiones válidas. En los siguientes apartados se profundizará en ambos procesos de entrenamiento.

4.1.2. Entrenamiento del modelo centralizado

En primer lugar, se presentan los hiperparámetros escogidos, así como la selección de características final. Para finalizar el apartado, se muestra la convergencia durante el entrenamiento.

Ajuste de hiperparámetros

La tabla 4.2 recoge los hiperparámetros escogidos tras la búsqueda. Para ajustar los hiperparámetros del modelo XGBoost en su versión centralizada se empleó una búsqueda con *grid search* utilizando validación cruzada de tipo *K-Fold* con cinco particiones. La búsqueda se realizó sobre los rangos especificados en el capítulo 3, dando lugar a un espacio de 1600 combinaciones, lo que resultó en un total de 8000 evaluaciones del modelo.

Hiperparámetro	Valor óptimo
`n_estimators`	100
`max_depth`	5
`learning_rate`	0.1
`reg_alpha`	0.01
`reg_lambda`	0.1
`gamma`	0.01

Tabla 4.2: Valores de los hiperparámetros seleccionados mediante búsqueda en rejilla con validación cruzada.

La profundidad máxima, `max_depth`, de 5, reduce el riesgo de que el modelo

memorice los datos del conjunto de entrenamiento. Esta restricción implica un sacrificio a nivel de precisión por parte de cada árbol individual, que se compensa mediante el uso de un número suficiente de árboles en el conjunto, 100 en total. El resto de hiperparámetros adoptan valores moderados, lo que favorece una convergencia progresiva y estable durante el entrenamiento.

Estos valores fueron seleccionados tras observar un equilibrio adecuado entre la complejidad del modelo y su capacidad de generalización, evitando tanto el sobreajuste como el subajuste.

Reducción de dimensionalidad

En este apartado se muestra el comportamiento del modelo tras aplicarse selección de características para la reducción de dimensionalidad. Como se introdujo en la sección 3.3.1, la métrica considerada para evaluar el impacto de RFE ha sido la MANE, y se ha aceptado como válida una diferencia absoluta de menos del 2 % en la pérdida de precisión por descartar características no relevantes. La figura 4.1 ilustra la MANE en función del número de características seleccionadas durante el proceso de reducción de dimensionalidad.

Se observa cómo la curva se estabiliza a partir de 5 variables de entrada. Es entonces, además, cuando se cumple el criterio mencionado anteriormente. La tabla 4.3 recoge las 5 características seleccionadas por RFE en este punto y, por tanto, las más significativas para explicar el comportamiento del *throughput*.

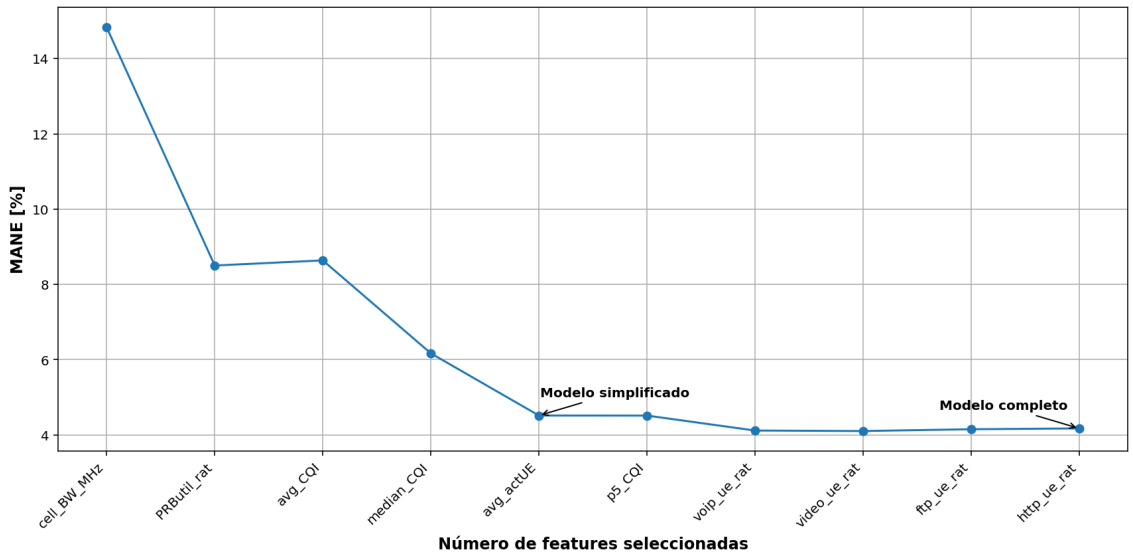

Figura 4.1: Evolución de la MANE respecto al número de características.

Características seleccionadas				
cell_BW_MHz	PRButil_rat	avg_CQI	median_CQI	avg_actUE

Tabla 4.3: Características seleccionadas mediante RFE cumpliendo la restricción del 2 % de degradación de MANE.

Para seguir las buenas prácticas en la elaboración de modelos de *Machine Learning*, se ha repetido la búsqueda de hiperparámetros con la nueva dimensionalidad encontrada por RFE. El nuevo *grid search* únicamente ha cambiado el valor de `gamma` de 0.01 a 0.1, lo cual es coherente. Al tener menor complejidad en el conjunto de variables de entrada, tiene sentido esperar mayor penalización a la creación de divisiones y nodos innecesarios.

Convergencia

Una vez finalizado el ajuste de hiperparámetros y la reducción de dimensionalidad mediante RFE, se entrena del modelo final con la configuración seleccionada. Durante este proceso, se ha empleado la funcionalidad de *early stopping*, con 10 rondas de paciencia. Por esto, el número de árboles utilizados por el modelo definitivo será menor a los 100 que se fijaron inicialmente. A continuación, se mostrarán las curvas de convergencia. Estas curvas permiten visualizar cómo se ha controlado el sobreajuste durante el entrenamiento.

A pesar de no requerir múltiples rondas, como ocurre en el modelo federado, se ha evaluado el comportamiento del modelo XGBoost a lo largo del número de árboles generados. La figura 4.2 muestra la evolución de las pérdidas conforme se añaden nuevos árboles al modelo en el escenario sin reducción de dimensionalidad. Por su parte, la figura 4.3 ilustra el comportamiento de ambas curvas cuando el modelo es entrenado con las 5 variables de entrada seleccionadas por RFE.

En ambos modelos se observa un descenso pronunciado de la pérdida en las primeras iteraciones, seguido de una estabilización del error. Como es de esperar, el modelo sin reducción de dimensionalidad necesita más rondas para converger completamente, el modelo necesita más información para captar la complejidad de la totalidad de las variables de entrada. Mientras que el modelo donde se aplicó RFE entrena 36 árboles, el modelo entrenado sobre el conjunto completo de datos entrena 88. Ninguno de los dos necesita llegar a los 100 árboles establecidos como hiperparámetro. En ambas gráficas se observa cómo las curvas de entrenamiento y validación no se despegan bruscamente en ningún

momento; aunque la separación entre las curvas es algo a tener en cuenta, el sobreajuste no se acentúa con el aumento de las iteraciones.

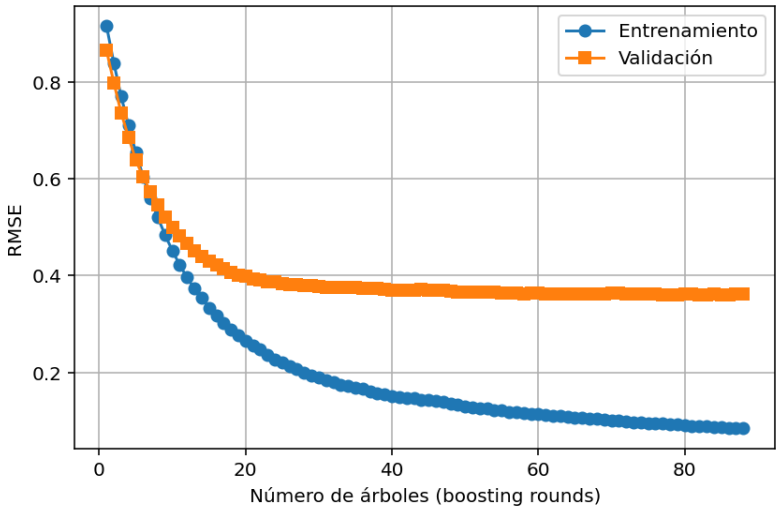

Figura 4.2: Curvas de convergencia del modelo centralizado entrenado con el juego de datos completo y *early stopping* con 10 rondas de paciencia.

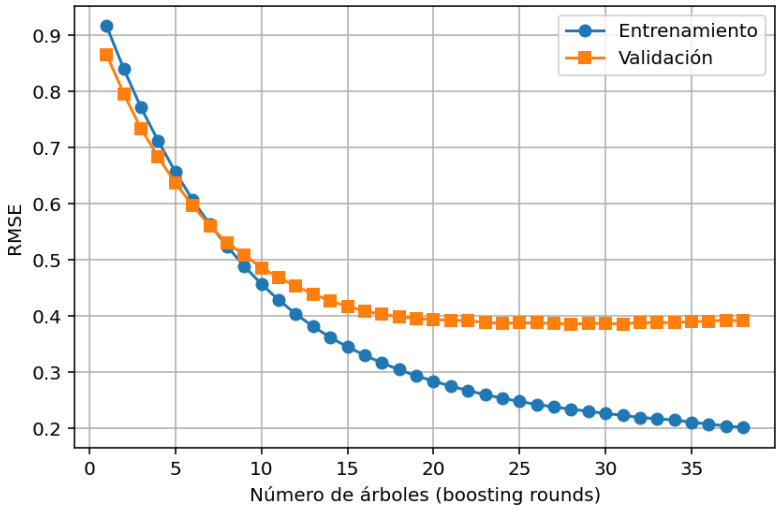

Figura 4.3: Curvas de convergencia del modelo centralizado aplicando reducción de dimensionalidad y *early stopping* con 10 rondas de paciencia.

4.1.3. Entrenamiento del modelo federado

Nuevamente, se comienza la sección presentando la elección de hiperparámetros y reducción de dimensionalidad utilizada. Posteriormente, se analiza la convergencia del entrenamiento.

Ajuste de hiperparámetros

La búsqueda de hiperparámetros en el entorno federado no se ha realizado mediante librerías que optimicen el proceso. Las librerías y clases utilizadas en la implementación del modelo centralizado no están directamente soportadas por el *framework* FLOWER utilizado para la simulación federada, al menos, no trivialmente. En su lugar, se ha llevado a cabo un proceso de ajuste manual, definiendo de antemano un conjunto de combinaciones de hiperparámetros manualmente. Este conjunto incluía un total de 25 configuraciones distintas, seleccionadas cuidadosamente para cubrir un rango amplio y representativo de valores, incluyendo tanto extremos como puntos intermedios del espacio de búsqueda (los rangos presentados en 3.3.1 ligeramente ampliados). Se entrenó el modelo 25 veces, una para cada combinación, y se escogió la combinación de hiperparámetros que generó el modelo con mejor rendimiento.

Aunque este procedimiento no constituye un *grid search* en sentido estricto, ha permitido identificar regiones que contengan valores cercanos a los que optimizan el rendimiento. A partir de esos resultados iniciales, se realizaron ajustes adicionales mediante pequeñas variaciones de los valores hasta encontrar una configuración con un rendimiento aceptable.

Hiperparámetro	**Valor**
`n_estimators`	2
`max_depth`	8
`learning_rate`	0.125
`reg_alpha`	0.1
`reg_lambda`	0.001
`gamma`	0.01
`num-server-rounds`	50
`num-supernodes`	36
`pacience-rounds`	10

Tabla 4.4: Valores de los hiperparámetros seleccionados en el modelo federado.

La tabla 4.4 recoge los valores fijados al finalizar el proceso de ajuste. Se observan algunas diferencias respecto al escenario centralizado, que son representativas del comportamiento de un modelo entrenado colaborativamente. Entre los nuevos hiperparámetros, destaca el número de árboles por cliente, `n_estimators`, fijado a 2. Como se introdujo en la sección 3.3.2, el número de árboles entrenados por cliente en cada ronda será reducido, con el objetivo de converger controladamente, favoreciendo la generalización. En cambio, la profundidad máxima de cada árbol, `max_depth`, aumenta de 5 a 8. Esta elección compensa la falta de aprendizaje que puede introducir una cantidad limitada de árboles por cliente y ronda, logrando un equilibrio que evita tanto sobreajuste como subajuste. Los hiperparámetros regularizadores y la tasa de aprendizaje no presentan cambios radicales, siguen adoptando valores moderados. Entre los regularizadores, resalta el parámetro regulador de L2 que reduce su penalización desde 0.1 hasta 0.001, un valor que no entraba en el rango de búsqueda del *grid search* centralizado.

En cuanto a los hiperparámetros propios del aprendizaje federado, referentes a la configuración del ciclo de entrenamiento, no han formado parte de la búsqueda, sino que han sido fijados manualmente. El número de clientes, `num-supernodes`, hace referencia a la cantidad de emplazamientos de red, y las rondas de entrenamiento y rondas de paciencia se fijaron tras observar la tendencia general de la convergencia del modelo en varias ejecuciones.

Reducción de dimensionalidad

De nuevo, el *framework* de aprendizaje federado utilizado no ofrece soporte directo para las clases que implementan RFE en el modelo centralizado. Por este motivo, y con el fin de mantener la coherencia experimental, se ha optado por reutilizar el orden de importancia de las variables obtenido previamente en el escenario centralizado. Esta decisión está justificada por el hecho de que los conjuntos de entrenamiento locales de cada cliente federado son particiones del mismo juego de datos utilizado en el modelo centralizado.

Durante el experimento, se han entrenado distintos modelos federados eliminando progresivamente las variables de menor relevancia, siguiendo el mismo orden de exclusión marcado por RFE en el modelo centralizado. En total, se han evaluado 10 modelos, desde el más simple (una sola variable) hasta el modelo completo con las 10 características. Esta estrategia ha permitido identificar la cantidad óptima de variables que maximiza el rendimiento del modelo en este entorno distribuido. La figura 4.4 muestra la MANE a lo largo del proceso descrito, resultando el modelo simplificado con 8 características el mejor.

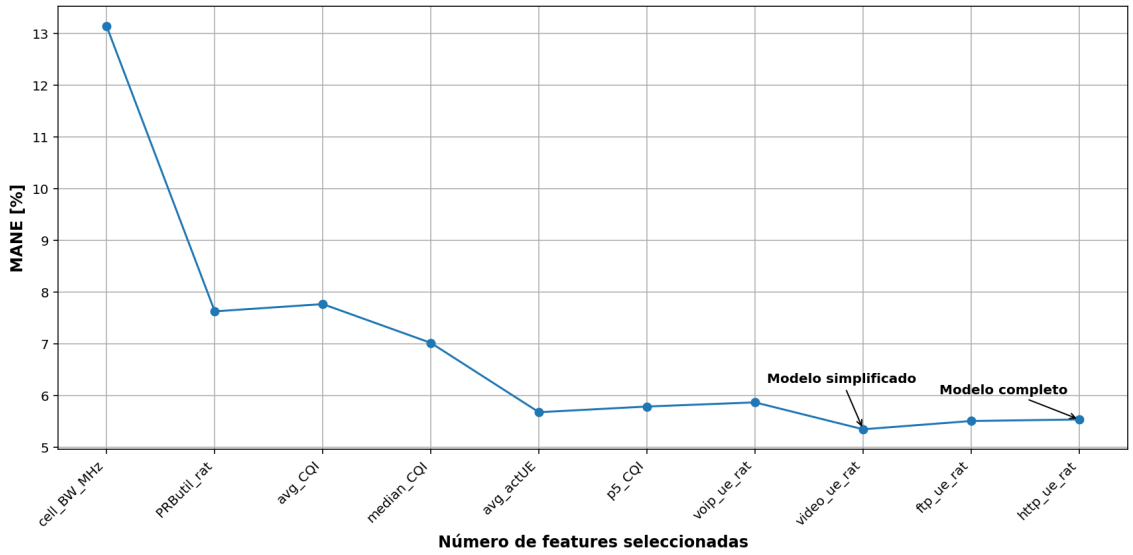

Figura 4.4: MANE en función de la reducción de dimensionalidad en el modelo federado.

Características seleccionadas			
`cell_BW_MHz`	`PRButil_rat`	`avg_CQI`	`median_CQI`
`avg_actUE`	`p5_CQI`	`voip_ue_rat`	`video_ue_rat`

Tabla 4.5: Características seleccionadas mediante RFE aplicadas al modelo federado.

La tabla 4.5 muestra las características elegidas para el modelo simplificado; se han seleccionado 8 variables. Respecto al caso centralizado, se han añadido 3 características: `p5_CQI`, `voip_ue_rat` y `video_ue_rat`. El objetivo de la reducción de dimensionalidad en el experimento del modelo federado no es simplemente mejorar la eficiencia computacional, sino que se pretende también mejorar la precisión del modelo. Por ello, las características seleccionadas en la tabla, son las que consiguen entrenar el modelo con mayor precisión. En la sección de resultados se mostrará la comparación entre el modelo federado completo y el modelo federado entrenado con 8 variables de entrada, para comprobar esta afirmación.

A partir de este punto, el resto del experimento en el entorno federado se ha desarrollado con esta configuración simplificada. El rendimiento alcanzado con esta configuración supera al obtenido al entrenar con la totalidad de variables de entrada disponibles, lo que refuerza la utilidad de la reducción de dimensiona-

lidad en entornos distribuidos. En estos escenarios, donde cada cliente cuenta con una cantidad de datos limitada, trabajar con menos variables contribuye a una mejor regularización del modelo, mitigando el riesgo de sobreajuste y favoreciendo la generalización.

convergencia

Ya expuesta la configuración del modelo, se procede con el entrenamiento final del mismo. De nuevo, se utiliza la técnica de *early stopping*, esta vez gestionada desde el servidor central, que recibe métricas de pérdidas ponderadas a partir de la RMSE escalada de cada cliente y, tras detectar 10 rondas sin mejoras en el rendimiento, detiene la ejecución, guardando el modelo de la ronda que menores pérdidas haya mostrado hasta el momento.

A diferencia del escenario centralizado, se mostrarán las pérdidas en función de la ronda, en vez de por cada árbol generado. Con la configuración elegida, se entrenan 72 árboles por ronda, 2 por cada cliente. La figura 4.5 ilustra el comportamiento del entrenamiento federado a lo largo de las rondas globales. Se observa como el modelo sigue una tendencia similar a la del escenario centralizado, extrapolándose a un número de árboles total mucho mayor, llegando a utilizar 2808 árboles en el momento que se activa el criterio de parada. En las primeras rondas se produce un descenso pronunciado de la pérdida, empezando la estabilización en la ronda 20. A partir de la ronda 30 el modelo alcanza su límite de aprendizaje, activándose el criterio de parada en la ronda 40, evitando que el modelo sobreajuste.

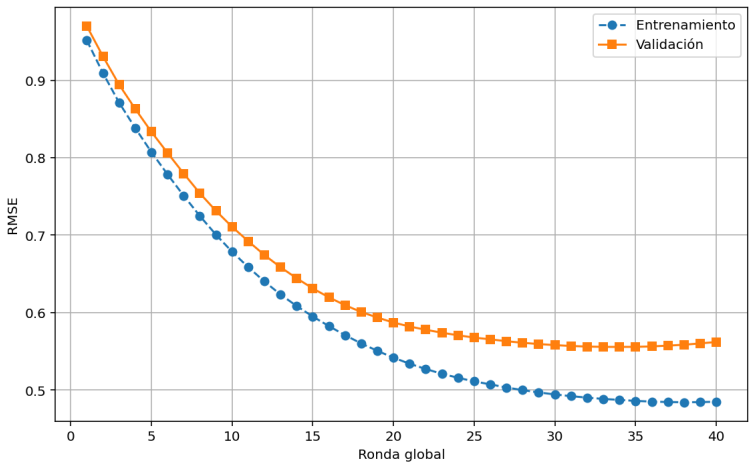

Figura 4.5: Curvas de convergencia del modelo federado entrenado con 8 características y *early stopping* con 10 rondas de paciencia.

4.2. Resultados

Esta sección recoge los resultados obtenidos tras el entrenamiento y evaluación de ambos modelos. Dado que el objetivo principal es comparar el rendimiento del modelo centralizado con el del federado, se ha optado por una presentación conjunta de los resultados. Para ello, la sección se divide en dos apartados: uno con los resultados numéricos y otro con los resultados gráficos. En los dos apartados se seguirá el siguiente enfoque: en primer lugar, exponer los resultados de los dos modelos individualmente y, a continuación, realizar una comparación directa de ambos. Al final de la sección, se añadirá un tercer apartado que agrupe los tiempos de ejecución de las distintas pruebas.

4.2.1. Evaluación numérica de la precisión

En este apartado, primero se presentarán los resultados del enfoque centralizado y, a continuación, los del federado. Finalmente, se recogen ambos en una tabla conjunta con intención comparativa.

Modelo centralizado

Una vez entrenados los modelos, se mide la precisión de estos con y sin reducción de dimensionalidad a través de las cifras de mérito. La tabla 4.6 presenta los resultados obtenidos de dichas métricas al final del entrenamiento para los modelos en ambos escenarios: completo (sin RFE) y simplificado (con RFE).

Cifra de mérito	**Sin RFE**	**Con RFE**
MAE [kbps]	1607.42	1796.35
RMSE [kbps]	2453.93	2650.42
R^2	0.899	0.883
MANE [%]	3.83	4.28
Trimmed MAPE (10 %) [%]	7.52	8.50

Tabla 4.6: Comparativa de métricas de evaluación del modelo centralizado con y sin reducción de dimensionalidad.

Como se puede observar en la tabla, los resultados obtenidos por el modelo

entrenado sin selección de características presentan un rendimiento ligeramente superior, mejorando al modelo con RFE en todas las métricas. Sin embargo, son diferencias modestas y más que asumibles, teniendo en cuenta los beneficios de la reducción de dimensionalidad en cuanto a eficiencia computacional y costes de infraestructura.

Las métricas de error absoluto (MAE y RMSE) obtenidas oscilan entre 1600 y 2700 kbps, como se observa en la tabla. Estos valores reflejan el error medio en las predicciones, considerando todas las muestras del conjunto de test (con celdas en distintos periodos). Cabe destacar que el conjunto de test cuenta con celdas con hasta 43000 kbps, por lo que estos valores de error absolutos pueden considerarse aceptables. Por otro lado, el coeficiente de determinación (R^2) es elevado en ambos casos, cerca de 0.9. Esto indica que los modelos son capaces de captar la varianza de los datos. Finalmente, en cuanto a las cifras de mérito relativas, todas presentan porcentajes de error menores al 10 %.

Modelo federado

Una vez finalizado el proceso de entrenamiento, se almacena el modelo federado de la ronda que mejor rendimiento ofrece en base a la función de pérdidas de validación y se procede a su evaluación. La tabla 4.7 recoge los resultados obtenidos por el modelo federado completo (10 características) y por el modelo federado simplicado (8 características). Las métricas muestran un rendimiento adecuado del modelo federado, aunque inferior al obtenido por el enfoque centralizado. Además, queda demostrada la afirmación del apartado 4.1.3, que asumía que el rendimiento con reducción de dimensionalidad superaba al del modelo completo en el escenario distribuido. Aunque es una diferencia humilde, el modelo simplificado mejora ligeramente al completo en todas las métricas.

Cifra de mérito	**Modelo completo**	**Modelo simplificado**
MAE [kbps]	2211.59	2177.40
RMSE [kbps]	2898.73	2874.20
R^2	0.8531	0.8619
MANE [%]	5.54	5.36
Trimmed MAPE (10 %) [%]	12.38	11.76

Tabla 4.7: Comparativa de métricas de evaluación del modelo federado con y sin reducción de dimensionalidad.

Comparación directa de la precisión de ambos modelos

Con el fin de facilitar la comparación, la tabla 4.8 recoge los mejores resultados obtenidos por cada modelo, a su vez que la diferencia entre ambos. El rendimiento del modelo federado se evalua sobre el mismo conjunto de test que se utilizó para medir el enfoque centralizado. Se ha asegurado que las muestras involucradas en entrenamiento y evaluación sean las mismas en los dos escenarios, garantizando así una comparación justa de la precisión de ambos. A continuación, se expondrá un análisis de los resultados del modelo federado siguiendo un enfoque comparativo con los del modelo centralizado.

En cuanto a las métricas de error absoluto, la MAE aumenta en 569.98 kbps su error promedio, un aumento mayor que el de 420.27 kbps que sufre la RMSE. Esto sugiere que, aunque el federado falla más de media, no comete errores demasiado extremos, ya que la presencia de estos dispararía la RMSE. El coeficiente de determinación solo disminuye 0.0371, lo que indica que la versión distribuida sigue siendo capaz de captar la varianza de la variable objetivo. Por otro lado, las dos métricas relativas se mantienen en niveles de precisión aceptables, con un incremento del error de 1.53 % la MANE y de 4.24 % la *trimmed* MAPE.

Cifra de mérito	**Centralizado (completo)**	**Federado (simplificado)**	**Diferencia**
MAE [kbps]	1607.42	2177.40	+569.98
RMSE [kbps]	2453.93	2874.20	+420.27
R^2	0.899	0.8619	–0.0371
MANE [%]	3.83	5.36	+1.53
Trimmed MAPE (10 %) [%]	7.52	11.76	+4.24

Tabla 4.8: Comparativa de métricas de evaluación entre los mejores rendimientos del modelo centralizado y federado.

4.2.2. Evaluación gráfica de la precisión

Este apartado tiene como objetivo, una vez presentados los resultados numéricos, complementarlos con herramientas visuales que permitan valorar la calidad de las estimaciones realizadas por los modelos. Para ello, se emplearán grafos de dispersión, tanto en escala lineal como en escala logarítmica: la primera per-

mite interpretar directamente el error absoluto en las unidades del *throughput* (kbps), mientras que la logarítmica permite observar los errores porcentualmente. En este último caso, la distancia respecto a la linea diagonal representa errores relativos:

$$\log(\hat{y}) - \log(y) = \log\left(\frac{\hat{y}}{y}\right)$$

Se evita así dar protagonismo visualmente a los errores de celdas con mayor *throughput* solo porque el valor absoluto sea elevado. Además, permite descomprimir los rangos y visualizar mejor el comportamiento en valores pequeños.

Se presentarán los grafos de dispersión de los distintos modelos. En estas gráficas, cada punto representa una celda en un periodo de tiempo. Como cierre a este apartado, se incluirán gráficas de dispersión superpuestas de ambos modelos, análisis de la CDF del error cometido por los dos modelos y, finalmente, un un nuevo gráfico de dispersión, esta vez de los errores cometidos por el modelo centralizado respecto a los cometidos por el federado, permitiendo observar si las celdas-periodo con más error son las mismas en ambos modelos.

Modelo centralizado

La figura 4.6 muestra los grafos de dispersión en ambas escalas para el modelo centralizado sin reducción de dimensionalidad. Por su parte, la figura 4.7 presenta las mismas gráficas pero del modelo entrenado tras aplicar la reducción de dimensionalidad.

Todas las gráficas están expuestas en la misma página para facilitar la comparación. En primer lugar, se analizan los grafos en escala lineal de los dos modelos. Del análisis visual de ambas gráficas, puede concluirse que los modelos están siendo capaces de captar la tendencia general de los datos, consiguiendo alinear razonablemente los puntos alrededor de la línea diagonal $y = x$.

No obstante, se observa mayor dispersión en la zona derecha de los grafos, algo esperable al tratarse de escala lineal. En este tipo de representación, los errores absolutos en celdas de mayor *throughput* serán visualmente más destacados, aunque en términos relativos sean idénticos a otros errores que ocurran en celdas con menor *throughput*, donde el impacto visual es menor.

Comparando ambas gráficas, el modelo entrenado con 5 variables de entrada presenta ligeramente mayor dispersión vertical, como era de esperar. La selección de características implica un pequeño sacrificio de precisión a cambio de eficiencia.

A continuación, se analizan las gráficas de dispersión en escala logarítmica.

Esta representación permite visualizar los errores en términos relativos, es decir, en proporción al valor real del *throughput*. En ambos modelos la mayor dispersión se concentra en la zona izquierda de la gráfica, correspondiente a muestras con valores inferiores a 10.000 kbps. Esto indica que los errores porcentuales más elevados se producen en celdas con menor *throughput*. Sin embargo, no implica una tendencia preocupante en cuanto a capacidad descriptiva para valores bajos de *throughput*, sino que es un resultado lógico: valores absolutos menores acumularán errores relativos más grandes, aunque pueda parecer contraintuitivo si solo se analizan los gráficos en escala lineal, donde los errores absolutos en valores altos dominan visualmente los grafos.

Modelo federado

La figura 4.8 muestra los gráficos de dispersión en ambas escalas para el modelo federado con mejor rendimiento, es decir, el simplificado con 8 características. En la escala logarítmica se aprecia cómo el modelo federado también capta los patrones de los datos razonablemente. De nuevo, los errores relativos son mayores en las celdas con menores *throughput* absolutos. Por su parte, la escala lineal permite observar cómo el modelo federado sigue la tendencia general de los datos, con buena alineación de los puntos en torno a la línea diagonal. Sin embargo, presenta una mayor dispersión en valores altos de *throughput*, un comportamiento común en escalas lineales, y coherente con los resultados del modelo centralizado. Además, se observa cómo el modelo federado subestima de forma sistemática el *throughput* para las muestras con valores superiores a 30000 kbps. Este comportamiento puede explicarse por la escasa cantidad de muestras con valores tan altos de *throughput*.

La situación que se acaba de describir solo se da en celdas con muy poca interferencia recibida en el DL, donde la eficiencia espectral es muy alta, y se pueden utilizar los valores más altos del esquema de modulación y codificación (cercanos a 1 Mbps por PRB). En celdas con estas características, solo hay servicios que llenan los PRB, cómo FTP y vídeo, sin presencia de usuarios de voz (que no llenan los PRBs que se les asignan). Estas condiciones son poco representativas en el subconjunto de datos de entrenamiento; solo un 7 % de las muestras de entrenamiento presentan un throughput mayor a 30 Mbps, lo que confirma que el modelo ha tenido poca exposición a este tipo de celdas-periodo durante el entrenamiento. La escasez de este tipo de celdas, se acentúa en el escenario federado, donde cada modelo individual dispone de menos muestras. Como resultado, el modelo tiende a aprender del tipo de celdas más numeroso, con menor valor de *throughput*, subestimando los valores elevados de este.

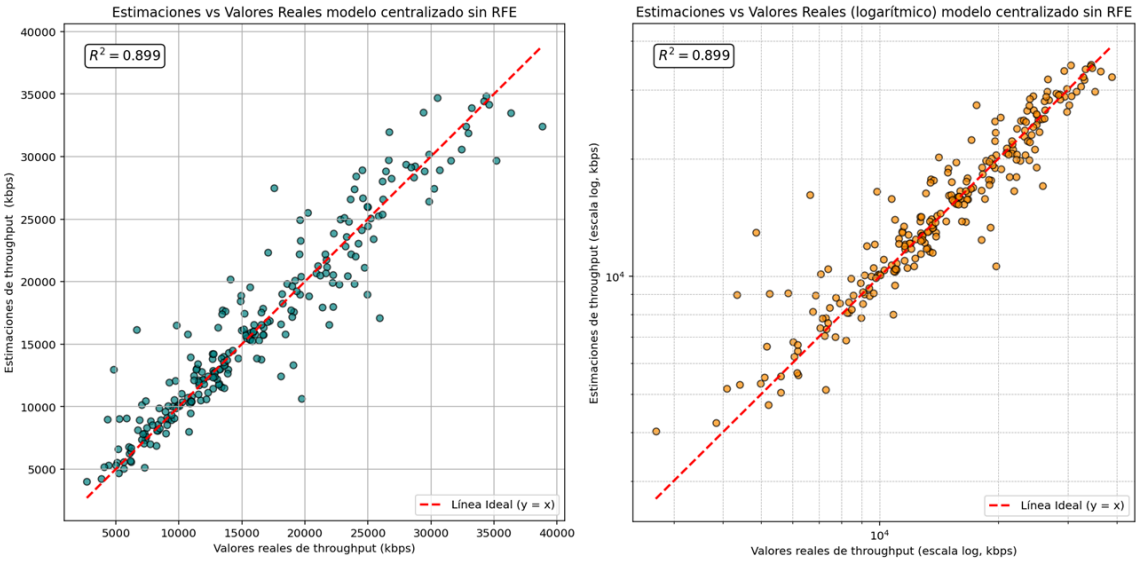

Figura 4.6: Grafos de dispersión lineal y logarítmico del modelo centralizado sin RFE.

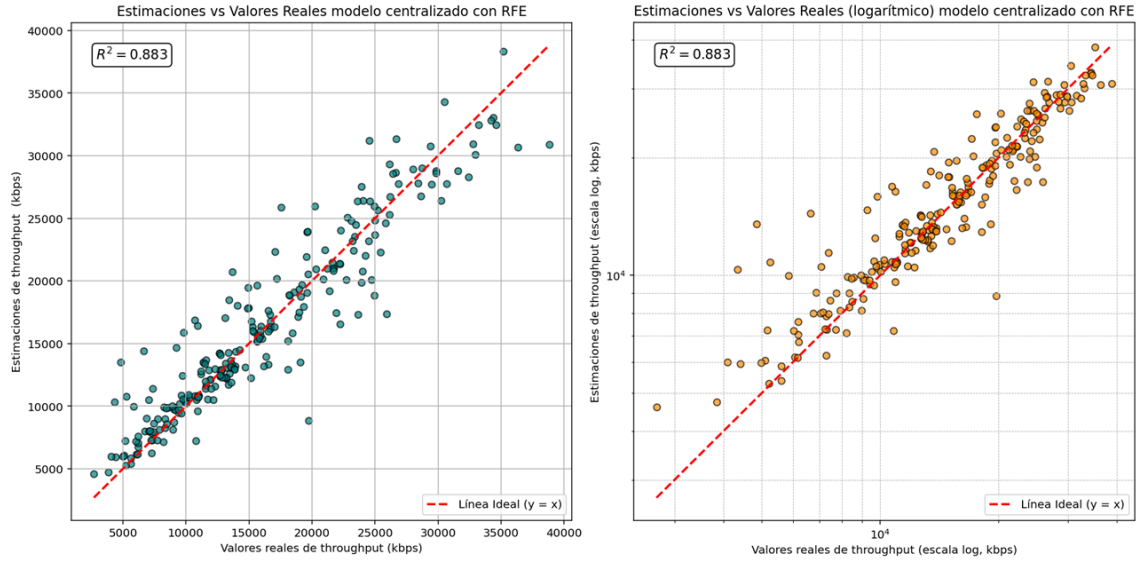

Figura 4.7: Grafos de dispersión lineal y logarítmico del modelo centralizado con RFE.

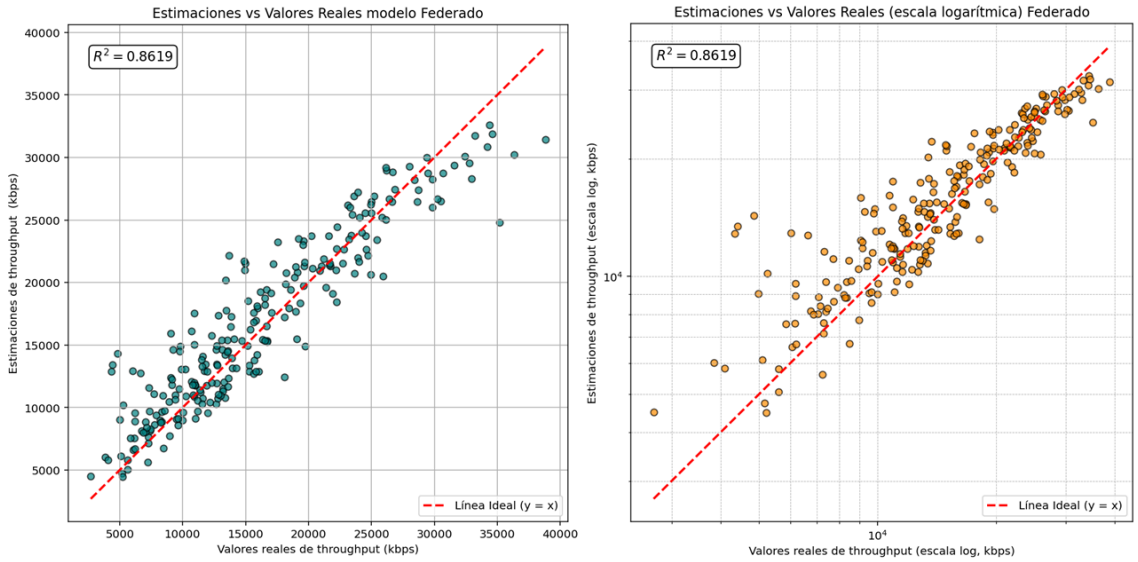

Figura 4.8: Grafos de dispersión lineal y logarítmico del modelo federado con 8 características.

Comparación gráfica directa de la precisión de ambos modelos

Para mejorar la comparativa de los modelos, y visualizar de forma más directa las diferencias entre ambos enfoques, se incluirán a continuación gráficos con ambas soluciones superpuestas. Las superposiciones en escala lineal y logarítmica se presentan en la figura 4.9.

La gráfica superpuesta lineal muestra un rendimiento aceptable por parte de los dos modelos. Ambos siguen distribuciones similares, ajustándose al comportamiento de la línea diagonal. Por otro lado, la superposición en escala logarítmica confirma lo observado hasta el momento: buen comportamiento de ambos modelos, captando la tendencia general de los datos. Nuevamente, con mayor dispersión en el federado debido a sus limitaciones sistémicas.

A continuación, se presentarán dos gráficas que pretenden analizar la naturaleza de los errores cometidos en ambos modelos. En primer lugar, se ha añadido la función de distribución acumulada (CDF) del error absoluto cometido en los dos escenarios, mostrada en la figura 4.10. Esta gráfica muestra la distribución de los errores a lo largo del conjunto de datos. Este análisis permite comprobar si alguno de los modelos tiende a cometer más errores extremos que el otro. En este caso, se observa que la curva correspondiente al modelo federado (línea

naranja) asciende de forma más lenta y alcanza el valor máximo (probabilidad acumulada de 1) con errores absolutos mayores que el modelo centralizado (línea azul). Esto indica que, además de presentar un error medio más alto, el modelo federado contiene una mayor cantidad de errores elevados en su distribución. El modelo centralizado no solo es más preciso en general, sino también más robusto frente a situaciones atípicas, cometiendo menos errores extremos.

Finalmente, se analiza si las celdas que presentan los errores más grandes son las mismas en ambos experimentos. Para ello, se introduce la figura 4.11, una gráfica de dispersión que muestra los errores de cada punto del modelo centralizado frente a los errores del modelo federado. Estos errores se han calculado con signo, no en valor absoluto, para reflejar si se tratan de subestimaciones o sobreestimaciones. De la gráfica se deduce que, cuando uno de los modelos comete un error elevado, el otro tiende a cometer un error de magnitud similar, ya que se observa cierta alineación respecto a la diagonal. Esta tendencia se ha cuantificado mediante el cálculo de la correlación de Pearson entre ambos vectores de error, realizado conjuntamente con la generación de la figura, y obteniendo un valor de 0,628. Esta cifra indica una correlación positiva moderada, lo que sugiere que ambos modelos tienden a fallar en las mismas muestras, aunque con diferencias en la magnitud y el signo del error.

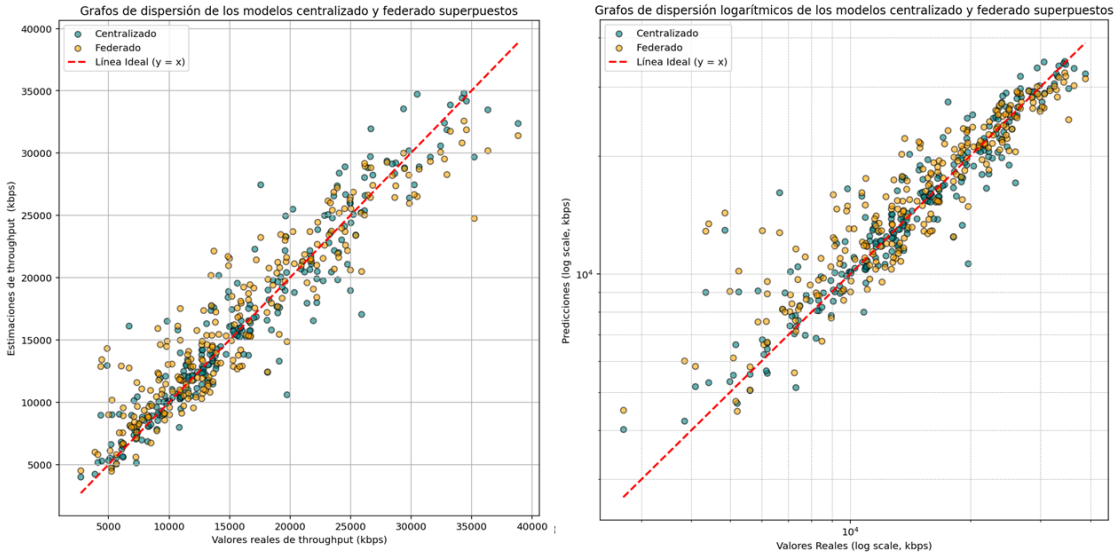

Figura 4.9: Gráfos de dispersión de ambos modelos superpuestos.

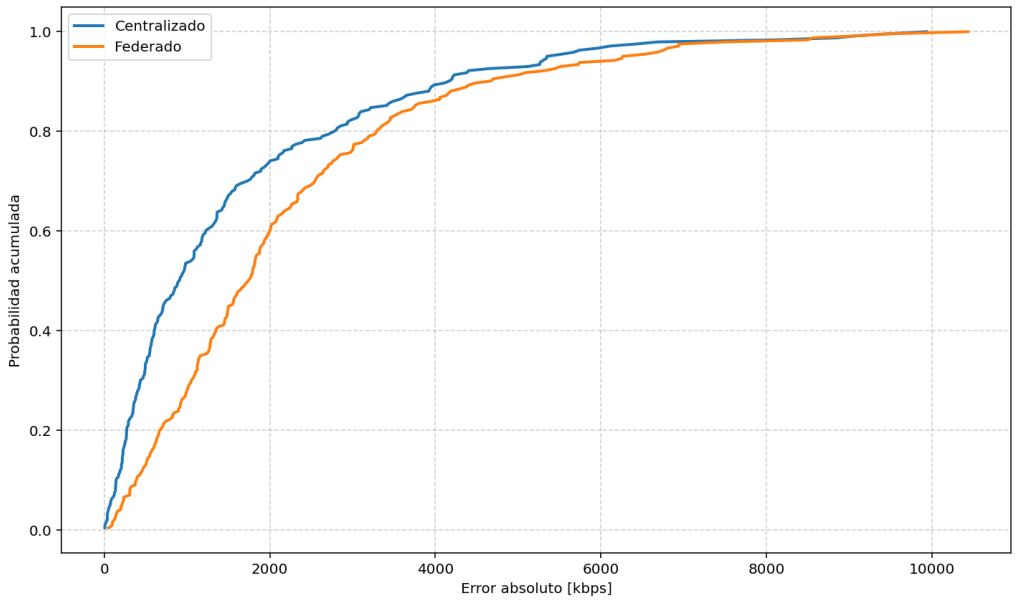

Figura 4.10: CDF del error absoluto: modelo centralizado vs modelo federado.

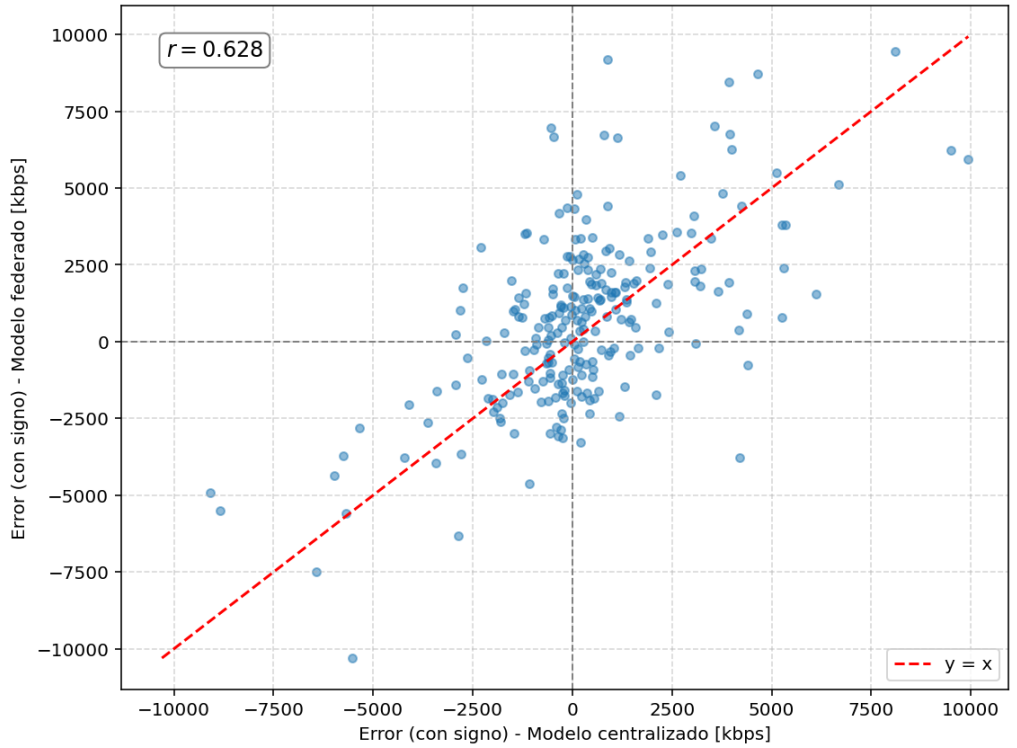

Figura 4.11: Dispersión de los errores: modelo centralizado vs modelo federado.

4.2.3. Tiempos de ejecución

Tras haber descrito las pruebas realizadas, así como los resultados obtenidos en ellas, se pretende analizar la eficiencia computacional de los distintos procesos involucrados. Para ello, en este apartado se exponen los tiempos de ejecución que han tomado las distintas simulaciones en ambos experimentos. La tabla 4.9 muestra los tiempos del modelo centralizado. Por su parte, la tabla 4.10, los del federado. En líneas generales, se observa una diferencia considerable a favor del modelo centralizado en términos de eficiencia computacional, presentando tiempos de ejecución mucho menores en todas las pruebas.

Para el ajuste de hiperparámetros en el modelo centralizado, se ejecutó una búsqueda en rejilla que abarcaba un amplio rango de combinaciones, y aún así, gracias al uso de librerías que optimizaron el proceso, se consiguió un tiempo de ejecución de 40.42 segundos. La misma situación se dio para el algoritmo RFE en la reducción de dimensionalidad, tardando únicamente 9 segundos en ejecutarse. En el modelo centralizado, destacan las ejecuciones en las que se realizaron los entrenamientos finales de los modelos con y sin reducción de dimensionalidad. Presentan tiempos de ejecución (una vez fijadas las configuraciones definitivas) prácticamente imperceptibles, gracias a la potente clase de Python que implementa XGBoost.

Por otro lado, las pruebas realizadas para la elaboración del modelo federado fueron costosas computacionalmente. El proceso colaborativo en el que participan 36 clientes distintos, a la vez que interaccionan con un servidor de agregación, es demandante. El entrenamiento de cada modelo federado tarda alrededor de 850 segundos (cerca de 15 minutos). En este trabajo, además, la ausencia de librerías que optimizasen el ajuste de hiperparámetros y reducción de dimensionalidad ha acentuado la ineficiencia computacional. Estas dos técnicas se han implementado mediante la elaboración de códigos que permitían ejecutar de forma iterativa los entrenamientos necesarios para cada una. Por ejemplo, para el ajuste de hiperparámetros, se realizaron 25 ejecuciones sucesivas del modelo federado, cada una con una combinación distinta de valores, proceso que duró un total de 21671.97 segundos (cerca de 6 horas).

De este apartado, se concluye que, para aprovechar por completo las ventajas que ofrecen los entornos federados en la elaboración de modelos de aprendizaje automático, es necesario seguir mejorando su eficiencia computacional. En este trabajo, se ha optado por procesos más rudimentarios para el modelo federado por cuestión de simplicidad. Sin embargo, en otro tipo de contextos resultará fundamental incorporar técnicas más específicas y optimizadas que permitan reducir los elevados costes de computación asociados al entrenamiento distribuido.

Proceso	**Tiempo de ejecución (s)**
Ajuste de hiperparámetros (*Grid Search* con CV)	40.42
Reducción de dimensionalidad (RFE con CV)	9.05
Entrenamiento final (modelo completo)	0.09
Entrenamiento final (modelo simplificado)	0.05

Tabla 4.9: Tiempos de ejecución del modelo centralizado.

Proceso	**Tiempo de ejecución (s)**
Ajuste de hiperparámetros	21671.97
Reducción de dimensionalidad	8123.78
Entrenamiento final (modelo completo)	854.00
Entrenamiento final (modelo simplificado)	844.97

Tabla 4.10: Tiempos de ejecución del modelo federado.

Capítulo 5

Conclusiones y líneas futuras

Sinopsis

En este capítulo de cierre se expondrán las conclusiones finales del proyecto, haciendo hincapié en los logros y limitaciones encontradas durante la elaboración del mismo. Además, se presentarán posibles líneas de trabajo futuro en el ámbito en el que se enmarca este estudio.

5.1. Conclusiones

Este Trabajo de Fin de Grado ha abordado la implementación de un sistema de aprendizaje supervisado distribuido empleando métodos de aprendizaje federado, aplicado a la predicción de *throughput* a nivel de celda en redes móviles de última generación. El principal objetivo del proyecto ha sido el estudio de la viabilidad del enfoque de aprendizaje federado en entornos realistas del ámbito de las telecomunicaciones, donde la privacidad de la información es un aspecto primordial y, a su vez, una limitación para elaborar modelos de aprendizaje automático que requieren centralizar los datos. Para ello, se ha seguido una línea comparativa de los rendimientos del modelo generado colaborativamente con el del modelo centralizado, ambos entrenados sobre el mismo conjunto de datos y utilizando árboles de decisión XGBoost como regresores.

A lo largo del desarrollo del proyecto, se han encontrado una serie de limitaciones prácticas. Las dos principales radican en la naturaleza y distribución de los datos a lo largo de los distintos clientes involucrados en el proceso de entrenamiento federado. Una se deriva del limitado volumen de datos del que dispone

cada nodo de entrenamiento en el escenario distribuido. La escasez de muestras implica un riesgo considerable para la capacidad de generalización de cada uno de los modelos locales que serán utilizados para la agregación y generación del modelo global. A diferencia del modelo centralizado, donde la variedad y abundancia de datos permite construir un modelo robusto que camufle la presencia de patrones anómalos, el rendimiento del modelo federado depende en gran medida de la calidad y cantidad de las muestras de datos de las que disponga cada cliente.

Una segunda limitación ha sido la heterogeneidad de los datos entre clientes. Además de reducida, la cantidad de los datos es dispar entre clientes debido a las diferencias en el número y la naturaleza de las muestras por simulación introducidas por las distintas condiciones de carga de red. En redes móviles, es común que los distintos emplazamientos de red ofrezcan servicios diferentes o estén diseñados para operar en condiciones muy variadas, como puede ser la presencia de diferentes obstáculos físicos o densidades de tráfico distintas. Esto se ve reflejado en el simulador empleado para la obtención de los datos, dando lugar a distribuciones de datos non-IID entre clientes, uno de los principales retos del aprendizaje federado que puede dar lugar a la agregación de modelos con patrones contradictorios entre sí por parte de distintos nodos del sistema.

A pesar de estas dificultades, los resultados obtenidos muestran que se ha conseguido una precisión aceptable en el modelo federado, con una degradación moderada de las prestaciones que ofrecía el enfoque centralizado. Tanto las métricas numéricas como el análisis gráfico han demostrado que, pese a las limitaciones de esta variante del aprendizaje automático, acentuadas por la escasez de datos disponibles, la construcción de modelos de aprendizaje federado entre distintos emplazamientos de red es más que viable, aún más en escenarios reales con medios mucho menos limitados. En el estudio, se han conseguido errores porcentuales en las estimaciones por debajo del 5.5 % y tiempos de ejecución inferiores a 15 minutos (una vez fijadas las configuraciones de hiperparámetros y reducción de dimensionalidad), tiempos asumibles para tareas de planificación de red que han de ejecutarse con una periodicidad, en muchos casos, mensual.

En la era de los datos, el aprovechamiento de los mismos puede dar lugar a herramientas potentísimas que mejoren la eficiencia, automatización y calidad de los servicios de telecomunicaciones. El aprendizaje federado introduce nuevos escenarios en los que las operadoras móviles pueden explotar la gran cantidad de información de la que disponen sin comprometer la privacidad de los clientes. Este enfoque no solo facilita el aprovechamiento de los datos de forma segura entre usuarios o emplazamientos de una compañía, sino que, introduce

la posibilidad de construir modelos colaborativos a gran escala entre distintas operadoras. Esto supondría un desafío a nivel de coordinación, las operadoras tendrían que coordinar las ejecuciones de sus modelos locales para que se realicen de forma síncrona, sin embargo, enfoques como el anterior abrirían la puerta a la creación de nuevas soluciones que permitan mejorar las prestaciones de las redes móviles a niveles inexplorados.

5.2. Líneas futuras

En base a las conclusiones anteriores, se presentan a continuación una serie de mejoras y extensiones de este estudio:

1. **Agrupamiento de emplazamientos por tipo de entorno:** la escasa cantidad de muestras por emplazamiento podría afrontarse realizando un agrupamiento (*clustering*) de los emplazamientos basado en características comunes, como el tipo de emplazamiento (rural, urbano, residencial...). Si las limitaciones de privacidad lo permitiesen, esto ayudaría a disponer de más muestras para entrenamiento y validación, mejorando la calidad del modelo.

2. **Ajuste de hiperparámetros personalizado por cliente:** en este trabajo, todos los nodos federados han utilizado los mismos hiperparámetros para XGBoost. Sin embargo, dada la distribución irregular de los datos entre celdas, la personalización del proceso de ajuste de hiperparámetros puede resultar beneficiosa para el rendimiento global, siempre teniendo en cuenta el riesgo de sobreajuste que podría introducir esta mejora.

3. **Selección de características distribuida:** se ha reutilizado la misma selección de características empleada en el modelo centralizado. Incorporar técnicas de selección de características locales permitiría identificar los atributos más relevantes en cada celda para el modelo federado, mejorando la eficiencia y posiblemente también la precisión de las estimaciones.

4. **Mecanismos de ponderación en la agregación:** la estrategia de *bagging* empleada no distingue entre árboles aportados por distintos clientes. Explorar la opción de añadir pesos a los distintos árboles, basándose en criterios como el tamaño de la muestra de datos de la que dispone el cliente o métricas de error durante la validación interna podrían mejorar notablemente el rendimiento del modelo final.

5. **Extensión a redes reales y nuevas tecnologías radio:** otra posible línea futura consistiría en repetir los experimentos con datos procedentes de redes reales y tecnologías más recientes, como 5G en bandas milimétricas. Este experimento solo será viable cuando dichas redes estén suficientemente maduras, de modo que existan situaciones de alta demanda donde se agote la capacidad máxima del sistema. Estas condiciones son necesarias para entrenar y evaluar un modelo como el del estudio que necesita aprender también de situaciones extremas en un entorno de red.

Bibliografía

[1] Corning Optical Communications, “Demystifying 5g,” 2024. [Online]. Available: https://www.corning.com/optical-communications/cala/es/home/Resources/5g-is-here-and-it-is-made-of-glass/demystifying-5g.html

[2] AWS, “Model fit: Underfitting vs. overfitting,” 2024. [Online]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

[3] A. Ananthaswamy, *Why Machines Learn: The Elegant Math Behind Modern AI*. Dutton, 2024.

[4] C. Gijón, M. Toril, S. Luna-Ramírez, J. L. Bejarano-Luque, and M. L. Marí-Altozano, “Estimating pole capacity from radio network performance statistics by supervised learning,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2090–2101, 2020.

[5] C. Gijón, M. Toril, and S. Luna-Ramírez, “Data-driven estimation of throughput performance in sliced radio access networks via supervised learning,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1008–1023, 2022.

[6] IBM, “What is xgboost?” 2024. [Online]. Available: https://www.ibm.com/think/topics/xgboost

[7] Y. Gao, “Federated xgboost with bagging aggregation,” 2023. [Online]. Available: https://flower.ai/blog/2023-11-29-federated-xgboost-with-bagging-aggregation/

[8] Keysight Technologies, “Lte overview - keysight technologies,” 2024. [Online]. Available: https://helpfiles.keysight.com/csg/89600B/Webhelp/Subsystems/lte/content/lte_overview.htm

[9] W. Xiang, K. Zheng, and X. Shen, *5G mobile communications*. Springer, 2016.

[10] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[11] Y. Chen, K. Wu, and Q. Zhang, "From qos to qoe: A tutorial on video quality assessment," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1126–1165, 2014.

[12] J. Lee, F. Solat, T. Y. Kim, and H. V. Poor, "Federated learning-empowered mobile network management for 5g and beyond networks: From access to core," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 3, pp. 2176–2212, 2024.

[13] C. Baena, S. Fortes, E. Baena, and R. Barco, "Estimation of video streaming kqis for radio access negotiation in network slicing scenarios," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1304–1307, 2020.

[14] S. Khan, O. L. A. López, and A. B. Khattak, "Mutual exploration for missing data imputation, qos parameter selection, and qos prediction in 5g networks using a novel skewness driven distribution imputation algorithm, pearson correlation, and xgboost," *Computer Networks and Communications*, pp. 374–392, 2024.

[15] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, and V. Gopalakrishnan, "On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 11–17, 2020.

[16] D. Minovski, N. Ögren, K. Mitra, and C. Åhlund, "Throughput prediction using machine learning in lte and 5g networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1825–1840, 2021.

[17] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, "Decision tree regression for soft classification of remote sensing data," *Remote Sensing of Environment*, vol. 97, no. 3, pp. 322–336, 2005.

[18] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, pp. 139–157, 2000.

[19] M. V. Luzón, N. Rodríguez-Barroso, A. Argente-Garrido, D. Jiménez-López, J. M. Moyano, J. Del Ser, W. Ding, and F. Herrera, "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, no. 4, pp. 824–850, 2024.

[20] Y. Gao, "Federated xgboost: Flower is all you need," 2024. [Online]. Available: https://flower.ai/blog/2024-02-14-federated-xgboost-with-flower/

[21] G. Zhu, J. Zan, Y. Yang, and X. Qi, "A supervised learning based qos assurance architecture for 5g networks," *IEEE Access*, vol. 7, pp. 43 598–43 606, 2019.

[22] Q. Chen, Z. Meng, X. Liu, Q. Jin, and R. Su, "Decision variants for the automatic determination of optimal feature subset in rf-rfe," *Genes*, vol. 9, no. 6, p. 301, 2018.

[23] L. Ciampiconi, A. Elwood, M. Leonardi, A. Mohamed, and A. Rozza, "A survey and taxonomy of loss functions in machine learning," *arXiv preprint arXiv:2301.05579*, 2023.

[24] M. Schmidt, G. Fung, and R. Rosales, "Optimization methods for l1-regularization," *University of British Columbia, Technical Report TR-2009-19*, 2009.

[25] T. Hastie, "Ridge regularization: An essential concept in data science," *Technometrics*, vol. 62, no. 4, pp. 426–433, 2020.

[26] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[27] D. Nielsen, "Tree boosting with xgboost-why does xgboost win every machine learning competition?" Master's thesis, NTNU, 2016.

[28] B. Sumathi *et al.*, "Grid search tuning of hyperparameters in random forest classifier for customer feedback sentiment prediction," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020.

[29] A. Botchkarev, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology," *arXiv preprint arXiv:1809.03006*, 2018.