Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE MÁSTER

Aprendizaje multi-etiqueta en entornos federados

Máster Universitario en Lógica,
Computación e Inteligencia Artificial

Realizado por
David Delgado Bejarano

Dirigido por
Dr. Jose María Moyano Murillo

Departamento de
Ciencias de la Computación e Inteligencia Artificial

Julio, 2024

Agradecimientos

Este trabajo pone fin a mi etapa como estudiante del Máster Universitario en Lógica, Computación e Inteligencia Artificial, el cual me ha proporcionado unos conocimientos nuevos para mí, a la vez que muy valiosos. Desde aquí mi agradecimiento a todos los profesores, compañeros y toda la comunidad que forma parte de una u otra manera de este Máster.

También agradecer a mi familia por todo el apoyo que me ha dado durante este año y a lo largo de mi vida académica.

Por último, quiero dar las gracias a mi tutor, José María, por los recursos, las explicaciones y el apoyo que me ha brindado durante todo este curso académico. Aprecio especialmente su plena disposición durante el desarrollo de este proyecto.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

El Aprendizaje Automático, *Machine Learning* (ML), ha emergido como una herramienta poderosa para extraer conocimientos y patrones complejos a partir de datos en una gran variedad de campos. Sin embargo, su implementación no está exenta de importantes desafíos, que pueden frenar el desarrollo y la implementación de sistemas de ML en aplicaciones del mundo real. Más concretamente, vamos a centrarnos en una serie de cuestiones relacionadas con la gestión de los datos, hablando de su propiedad y distribución, así como de cuestiones vinculadas a la privacidad y seguridad en su manejo.

En el marco del ML, especialmente en el caso del *Deep Learning* (DL), la obtención de grandes cantidades de datos, conocido como *big data*, es fundamental para alcanzar un rendimiento óptimo. Sin embargo, en numerosas aplicaciones, conseguir tales volúmenes de datos puede resultar difícil. En estos casos, se recurre al *small data*, donde los conjuntos de datos son pequeños o carecen de información completa, como valores o etiquetas, afectando negativamente al rendimiento de nuestro modelo.

Además, la creciente preocupación en la sociedad actual sobre la propiedad de datos plantea dudas sobre quién tiene derecho a utilizar datos para desarrollar tecnologías de inteligencia artificial. Así, como los datos son generados y poseídos por diversas organizaciones o empresas, la práctica tradicional de centralizar los datos en una ubicación para entrenar modelos de ML ya no es tan factible.

Igualmente, a medida que la IA se desarrolla, también lo hace la preocupación acerca del tratamiento de datos confidenciales y la protección de la privacidad[1]. Esto ha llevado a la publicación de nuevas leyes que regulan el uso y manejo de datos [1]. En este contexto, la recopilación de datos se vuelve aún más difícil, especialmente si son datos sensibles como los médicos o financieros, que por su propia naturaleza no se pueden compartir, estando sometidos a duras restricciones de circulación y son gestionados exclusivamente por sus propietarios.

Asimismo, el intercambio de grandes conjuntos de datos entre organizaciones no siempre tiene por qué resultar beneficioso. La transferencia de datos puede implicar en algunos casos la pérdida de control sobre ellos y además pueden

[1]The privacy paradox with AI. https://www.reuters.com/legal/legalindustry/privacy-paradox-with-ai-2023-10-31/

disminuir su valor, mientras que los beneficios derivados de un modelo pueden no distribuirse de manera justa entre todos los participantes involucrados [2].

En consecuencia, surgen desafíos en torno a la recopilación, uso y compartición de datos en el contexto del ML tradicional, así como de la manera que un modelo en este marco puede beneficiar a una u otra organización, creando incertidumbre sobre si la pérdida de privacidad que conlleva la compartición de datos se compensa con los beneficios que este modelo pueda aportar a una organización en particular.

El Aprendizaje Federado, *Federated Learning* (FL), nace como una solución para abordar los desafíos planteados anteriormente. La idea es descentralizar el proceso de entrenamiento, permitiendo así que los datos permanezcan en sus ubicaciones originales, evitando la necesidad de compartirlos para así reducir los riesgos asociados con la exposición de datos sensibles. Esta descentralización también facilita el cumplimiento normativo al permitir que las organizaciones mantengan el control sobre sus propios datos y cumplan con las regulaciones pertinentes. En conclusión, el FL ofrece una solución completa dentro de este marco de privacidad, promoviendo así la creación de nuevas técnicas de ML de manera segura y responsable.

Por otro lado, el segundo pilar principal de este trabajo es el Aprendizaje Multi-etiqueta, *Multilabel Learning* (MLL). Surge como una extensión natural del ML más primitivo, permitiendo que cada instancia de datos pueda estar asociada con múltiples etiquetas o clases en lugar de una sola. Este enfoque es necesario para tratar problemas complejos que no se ajustan a la estructura de clasificación tradicional y requieren ser tratados con MLL. Por ejemplo, en aplicaciones como la clasificación de imágenes, una fotografía puede contener múltiples objetos, cada uno de los cuales necesita ser etiquetado correctamente. También, en la clasificación de documentos, un documento puede tratar sobre varios temas diferentes al mismo tiempo. Estas situaciones plantearon la necesidad de desarrollar algoritmos y técnicas capaces de manejar la complejidad de los datos multi-etiqueta. En la Figura 1.1 se observa la diferencia entre clasificación tradicional y clasificación multi-etiqueta.

Figura 1.1: Izquierda: clasificación multiclase. Derecha: MLL. Fuente: [3].

En esta tesitura, se incidirá en la necesidad de desarrollar modelos capaces de abordar los problemas multi-etiqueta pero estando inmersos en entornos federados, ya que existen situaciones en las que hay datos delicados distribuidos en múltiples ubicaciones y, además, requieren ser tratados con MLL. Por ejemplo, en el sector sanitario, los informes médicos de los pacientes pueden contener

información sobre varias enfermedades o condiciones médicas, lo que requiere de múltiples etiquetas para una clasificación precisa. Así, aplicar FL y además utilizar MLL parece una buena idea para ser aplicada en determinados casos. Además, en este tipo de problemas es posible que cada parte posea etiquetas u objetivos de clasificación distintos, haciendo aún más complejo si cabe el proceso de aprendizaje.

En conclusión, pensamos que este trabajo es una buena oportunidad para combinar las ventajas de estos dos marcos de ML, y de ahí surge la idea del TFM: proponer un nuevo método que integre el FL con el MLL, una propuesta que, hasta donde llega nuestro conocimiento, no ha sido abordada previamente en la literatura académica.

2. Objetivos

Como se mencionó anteriormente, el objetivo fundamental de este trabajo es partir de las técnicas ya conocidas y publicadas y proponer un nuevo enfoque de MLL en entornos federados.

Este propósito se puede desglosar en varios subobjetivos que servirán de guía en nuestro trabajo:

- Estudio profundo del estado del arte en FL. Así, analizaremos las diferentes modalidades de este, como el FL vertical y horizontal, centrándonos en este último, en el que estudiaremos sus arquitecturas y los procesos seguidos en el entrenamiento de los distintos modelos, para de esta manera comprender mejor sus aplicaciones y limitaciones.
- Estudio de las técnicas existentes en MLL, presentando las investigaciones y métodos principales en ambos campos. Este análisis nos permitirá establecer bases sólidas para nuestro trabajo.
- Diseño e implementación de nuestra propuesta, la cual requerirá también de la elaboración de un método para simular particiones federadas trabajando con datos MLL donde cada cliente tenga distintas etiquetas, que será el caso de estudio.
- Llevaremos a cabo una experimentación para poner a prueba nuestra idea, utilizando diversos conjuntos de datos y evaluando el rendimiento de nuestro método en comparación con las técnicas convencionales de ML. Los resultados obtenidos nos darán una idea sobre la efectividad y la viabilidad de nuestro enfoque en casos prácticos.

3. Estado del arte

En el contexto actual de rápida evolución tecnológica, es fundamental abordar y comprender el estado del arte tanto en FL como en MLL y conocer las contribuciones previas relevantes al tema del trabajo. Así, a continuación incidiremos de forma más profunda en conceptos clave y presentaremos también aquellos que creemos interesante que se conozcan y de los cuales también hemos extraído ideas.

3.1. Federated Learning

La idea principal del aprendizaje federado es entrenar un modelo global donde cada localización, en las cuales residen los datos, contribuye a dicho modelo reentrenándolo con sus propios datos locales. En lugar de transferir datos de un sitio a otro, en el aprendizaje federado se transfiere información no sensible, por ejemplo los parámetros del modelo, de manera segura sin comprometer la información confidencial, evitando que otras partes puedan acceder a datos sensibles de los demás.

Este concepto nació en 2016 de la mano de Google [4], para coordinar millones de dispositivos móviles con un servidor central para hacer modelos conjuntos, pero sin compartir los datos locales de los propietarios. Rápidamente, este enfoque se extendió entre organizaciones y empresas, con el objetivo de construir un modelo conjunto de ML basado en los datos alojados en múltiples sitios.

El proceso de aprendizaje federado implica dos etapas: el entrenamiento del modelo, *ML training*, y la aplicación del modelo a nuevos datos *ML inference*. Cada parte contribuye con sus datos para entrenar el modelo, siempre sin que estos abandonen su ubicación original. Además, el modelo puede transferirse parcialmente de una parte a otra mediante una encriptación segura, garantizando así la confidencialidad de los datos. El modelo resultante es una buena aproximación del modelo ideal construido en una única parte con todos los datos.

Denotemos V_{SUM} y V_{FED} como la medida de rendimiento del modelo centralizado M_{SUM} y el modelo federado M_{FED}, respectivamente. Decimos que el

modelo de aprendizaje federado M_{FED} tiene una pérdida de rendimiento δ si:

$$|V_{SUM} - V_{FED}| < \delta \tag{3.1}$$

Lo que viene a decir que si usamos aprendizaje federado para hacer un modelo de aprendizaje automático, el rendimiento de este modelo en datos futuros es aproximadamente el mismo que con un modelo que se construye uniendo todas las fuentes de datos. La pérdida δ que pueda haber vale la pena teniendo en cuenta la seguridad y privacidad que se está ganando.

En aprendizaje federado hay diversas variaciones y tipos de estructuras. Por ejemplo, podemos tener un coordinador central (Figura 3.1) o no (Figura 3.2), haciendo el proceso de aprendizaje incluso más seguro al no haber terceras partes. Estas estructuras se analizan en mayor detalle en la sección 3.1.4.

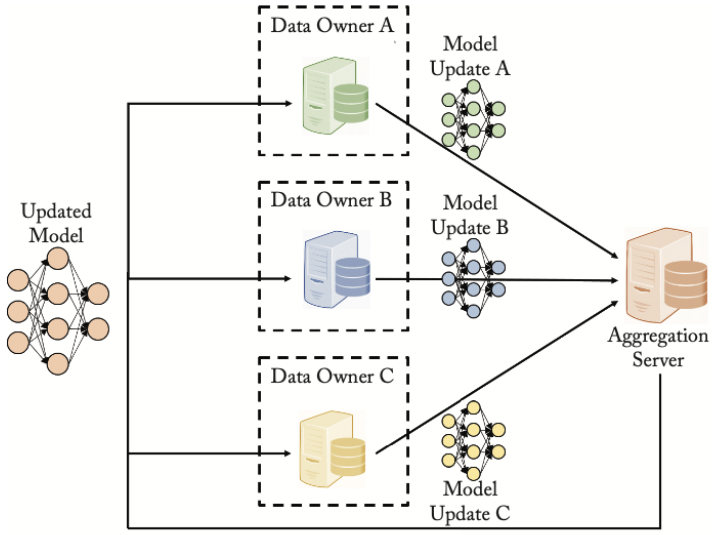


Figura 3.1: FL Cliente-Servidor. Fuente: [2].

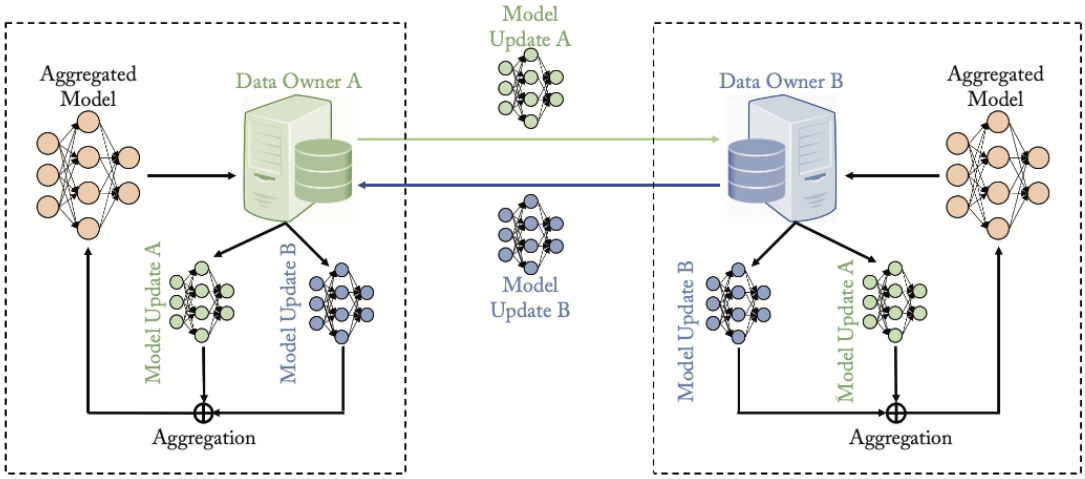


Figura 3.2: FL *Peer-to-Peer* (P2P). Fuente: [2].

3.1.1. Categorías de FL

Según la estructura propia de los datos y cómo se distribuyen entre las partes colaboradoras podemos diferenciar tres tipos de aprendizaje federado: Aprendizaje federado vertical, *Vertical Federated Learning* (VFL), aprendizaje federado horizontal, *Horizontal Federated Learning* (HFL) y aprendizaje federado por transferencia, *Federated Transfer Learning* (FTL).

Se habla de HFL en situaciones donde los conjuntos de datos de las partes comparten el espacio de atributos pero tienen distinto espacio de muestras, como se puede ver en la Figura 3.3. Un ejemplo típico es el de dos bancos de distintas ciudades: la inmensa mayoría de usuarios serán distintos, pero el espacio de atributos será similar, ya que es el mismo tipo de negocio.

Por otro lado, se habla de entornos VFL cuando los conjuntos de datos comparten el mismo espacio de muestras pero diferente espacio de atributos, tal como se observa en la Figura 3.4. Como ejemplo ilustrativo podemos pensar en un banco y una compañía de e-commerce de la misma ciudad: muchos usuarios pueden coincidir, pero al ser negocios muy diferentes, el espacio de atributos será distinto en los dos. Además, por lo general en este tipo de entornos, solo una de las partes interesadas (ya sea el banco o la compañía de e-commerce) posee la etiqueta de clase y por tanto será suyo el objetivo principal del clasificador.

Por último existe otro caso, el FTL, que se utiliza cuando apenas hay superposición ni en las muestras ni en los atributos, como se ilustra en la Figura 3.5. Este enfoque es menos frecuente y menos estudiado en la literatura, por tanto, al tampoco ser objetivo de este trabajo, no se analizará en mayor detalle.

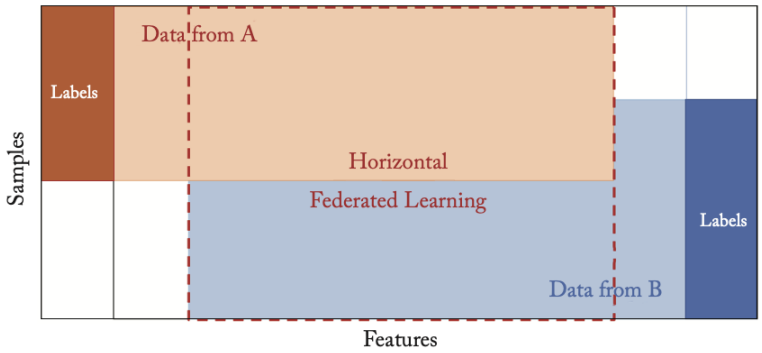


Figura 3.3: Horizontal Federated Learning. Fuente: [2].

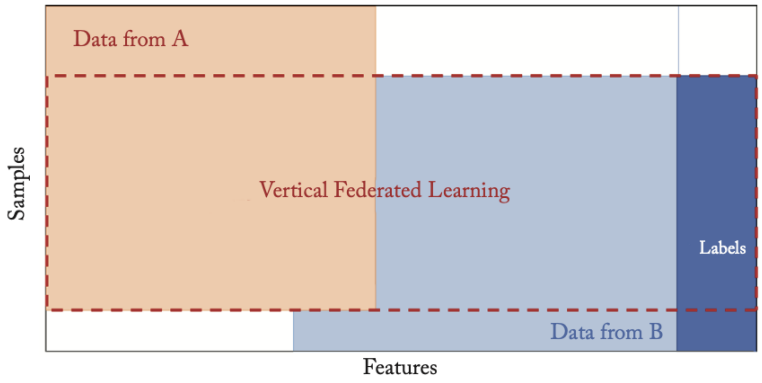


Figura 3.4: Vertical Federated Learning. Fuente: [2].

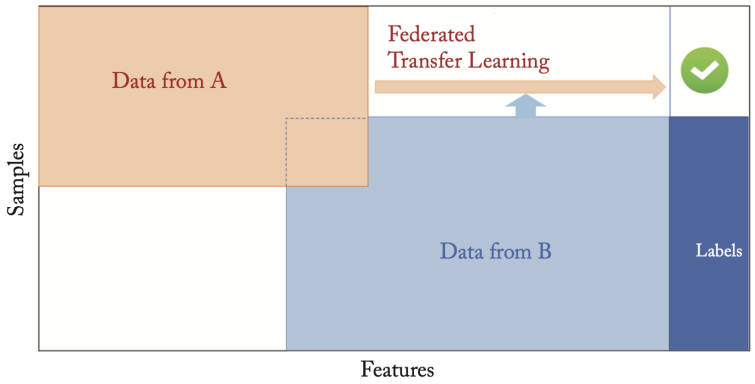


Figura 3.5: Transfer Federated Learning. Fuente: [2].

Las diferencias entre HFL y VFL resultan en tener que utilizar enfoques y métodos muy distintos. En HFL, cada parte suele entrenar un modelo local e intercambia sus parámetros actualizados con un servidor, el cual agrega las actualizaciones y envía el resultado de la agregación a cada parte, dando como resultado un modelo global compartido entre todas las partes. En cambio, en VFL, lo más común es que cada parte mantenga sus datos y su propio modelo, pero intercambie los resultados intermedios, dando como resultado un modelo local para cada parte, aunque puede obtenerse también un modelo global compartido, donde cada cliente pueda tener solo parte del modelo, y necesite de todas las partes de todos los clientes para hacer la predicción.

Con respecto a la privacidad y confidencialidad de los datos, en HFL se deben proteger los parámetros del modelo durante la comunicación entre las partes y el servidor centralizado, mientras que en VFL este problema se centra en salvaguardar los resultados intermedios, que pueden contener información sensible sobre las características de los datos. Estas ideas se resumen en la Tabla 3.1.

La necesidad del aprendizaje federado se ha incrementado fuertemente en la industria en los últimos años. Las empresas e instituciones que poseen solo pequeños y fragmentados conjuntos de datos buscan constantemente socios para desarrollar modelos, ampliando así el conjunto de datos y mejorando la precisión del modelo [2].

	HFL	VFL
Diferencias en los datos	Espacio de muestras	Espacio de atributos
¿Qué se intercambia?	Parámetros del modelo	Resultados intermedios
¿Qué no sale de cada parte?	Datos locales	Datos locales y el modelo
¿Qué obtiene cada parte?	Un modelo compartido	Un modelo local o parte de uno global

Tabla 3.1: Comparación de las principales diferencias entre HFL y VFL [5].

3.1.2. Privacidad Diferencial

Como hemos mencionado anteriormente, el aprendizaje federado surge en gran parte para subsanar problemas de privacidad y aumentar la seguridad en el tratamiento de los datos. En este marco es útil introducir el concepto de privacidad diferencial, que da una medida rigurosa sobre la pérdida de privacidad en un proceso de intercambio de información.

Se tiene privacidad diferencial si los resultados de un análisis o proceso no se ven influenciados de manera significativa por la presencia o ausencia de información de un individuo específico en el conjunto de datos. Es decir, si se elimina o se añade un registro, el resultado general del análisis completo no cambia de manera sustancial, protegiendo así la información a nivel individual. La idea principal de la privacidad diferencial es despistar a los clientes o partes que forman parte del proceso de aprendizaje federado si intentan consultar información individual de la base de datos. Por lo tanto, es crucial en entornos donde múltiples partes colaboran en un modelo de aprendizaje y hay falta de confianza entre las partes.

La idea anterior se puede cuantificar de la siguiente forma: un proceso de intercambio de información M guarda privacidad diferencial de parámetros (ϵ, δ) si dadas dos bases de datos D y D' que se diferencian solo en una entrada, y para todo $S \subseteq \text{Rango}(M)$,

$$\Pr[M(D) \in S] \leq e^{\epsilon} \cdot \Pr[M(D') \in S] + \delta \tag{3.2}$$

donde ϵ es el *"privacy budget"*, que mide cuánta privacidad se está dispuesto a sacrificar, y δ es la confianza, que controla cuánto estamos dispuestos a permitir que la desigualdad no sea cierta, un δ más bajo implica más dificultad garantizar la privacidad.

La cantidad

$$ln \frac{Pr[M(D) \in S]}{Pr[M(D') \in S]}$$

es la llamada *pérdida de privacidad*. Cuando $\delta = 0$, se logra la máxima privacidad diferencial [2].

Para garantizar la privacidad diferencial en el entrenamiento del modelo, se emplean métodos como la adición de ruido, que modifica los valores de los datos

durante el proceso de entrenamiento añadiendo valores aleatorios. Estas modificaciones no alterarán notablemente el resultado del análisis, pero sí que ocultan el dato real originario.

3.1.3. Horizontal Federated Learning

Una vez vistas las características generales del aprendizaje federado, estudiaremos ahora en profundidad el aprendizaje federado horizontal, el cual servirá como marco donde propondremos nuestro algoritmo.

Como se expuso en la sección 3.1.1, el HFL se aplica en casos en los que los conjuntos de datos de los diferentes clientes comparten el espacio de características pero difieren en el espacio de muestras (Figura 3.3). Para ilustrar esta idea, podemos considerar varios bancos en distintas ciudades: los usuarios de los bancos serán diferentes, sin embargo, los bancos compartirán el espacio de características debido a que el modelo de negocio es el mismo.

De manera rigurosa, esto se puede escribir como sigue:

$$\mathcal{X}_i = \mathcal{X}_j,\ \mathcal{Y}_i = \mathcal{Y}_j,\ I_i \neq I_j,\ \ \forall \mathcal{D}_i, \mathcal{D}_j, \quad i \neq j \tag{3.3}$$

donde el par de espacio de características y etiquetas de las dos partes, es decir, $(\mathcal{X}_i, \mathcal{Y}_i)$ y $(\mathcal{X}_j, \mathcal{Y}_j)$, se asumen iguales, mientras que los usuarios I_i e I_j se asumen diferentes; $\mathcal{D}_i$ y $\mathcal{D}_j$ denotan los conjuntos de datos de la i-ésima parte y la j-ésima parte [2].

También es importante considerar con qué tipo de datos estamos trabajando cuando hacemos aprendizaje federado. Usualmente, se asume que los datos utilizados en el entrenamiento de modelos de aprendizaje automático son IID, es decir, son independientes e idénticamente distribuidos. Sin embargo, en la mayoría de casos reales, los datos no cumplen con estas condiciones y son No-IID, siendo no independientes e idénticamente distribuidos. En la Figura 3.6 se puede observar un ejemplo de datos IID, donde cada cliente tiene datos de todas las clases (en este caso, de todos los dígitos), mientras que en el caso No-IID, cada cliente puede tener información de algunas de las clases pero no de todas. Por tanto, mientras que en el escenario IID cada cliente por sí solo podría crear un modelo capaz de predecir ejemplos de cualquiera de las clases, en el escenario No-IID no es así (el cliente de la izquierda nunca sería capaz de reconocer un 5), por lo que tiene aún más sentido la aplicación de técnicas de FL capaces de tomar ventaja de los datos e información almacenada por los distintos clientes. Además, el FL tiene herramientas suficientes para trabajar de forma natural con datos No-IID, como veremos más adelante.

Es preciso añadir que los datos No-IID no solo se presentan cuando cada cliente tiene ejemplos de diferentes clases. Si nos ceñimos al ejemplo de la Figura 3.6, puede ocurrir que cada persona tenga estilos de escritura muy diferentes, y por tanto, los "4" escritos por una persona específica difieran considerablemente de los "4" que tienen otros clientes, teniendo de nuevo datos No-IID.

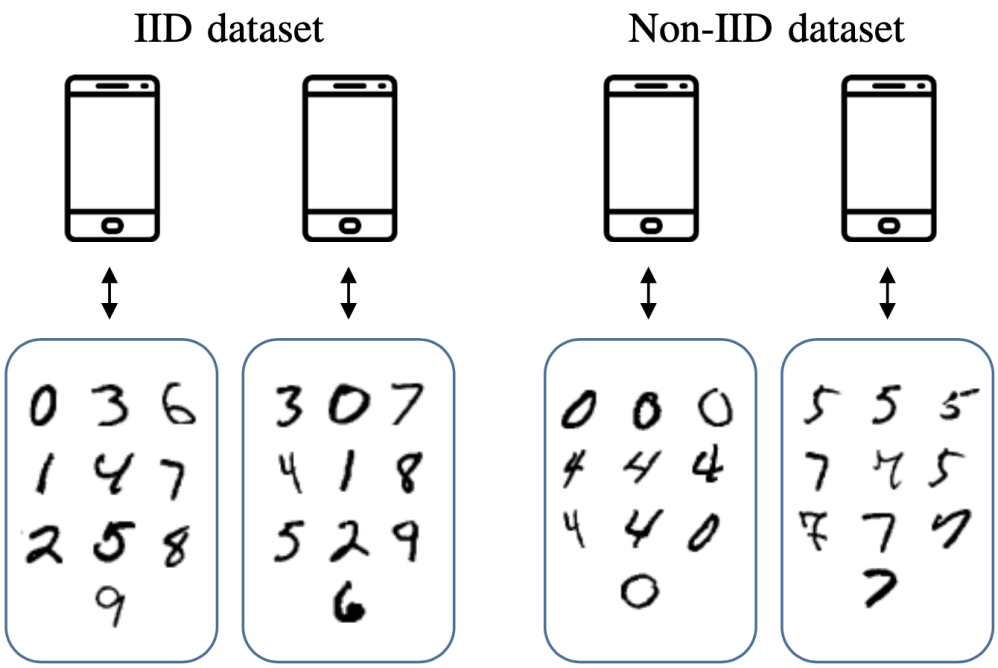


Figura 3.6: Datos IID vs Non-IID usando el dataset de MNIST. Datos generados por distintos usuarios tienden a ser distintos. Fuente: [6].

3.1.4. Arquitectura del HFL.

En esta sección vamos a ver cómo está construido un sistema de aprendizaje federado horizontal. Se presentarán dos construcciones distintas, la arquitectura cliente-servidor y la arquitectura *peer-to-peer*, la cual puede ser traducida como de igual a igual. Empezamos con la primera:

- **Cliente-Servidor**

  En este sistema, tenemos K partes que colaboran para entrenar un modelo de ML con ayuda de un servidor, también llamado coordinador o agregador. Por tanto, al haber una parte externa, es muy importante prevenir la filtración de información de los participantes hacia este. El proceso de entrenamiento de un sistema de estas características es el siguiente, también mostrado en la Figura 3.7:

  1. El servidor inicializa un modelo y lo envía a los clientes.
  2. Los participantes actualizan el modelo localmente y envían el resultado, la actualización del modelo, al servidor.
  3. El servidor agrega esas actualizaciones siguiendo un algoritmo determinado.
  4. El servidor envía los resultados de la agregación a las partes.
  5. Repetir los pasos 2-4 durante un número finito de iteraciones o hasta que se cumpla una condición de parada.

Es importante destacar que esta arquitectura es totalmente independiente del algoritmo de ML usado (redes neuronales, árboles de decisión, regresión logística, etc.). Finalmente, todos los participantes compartirán los mismos parámetros del modelo.

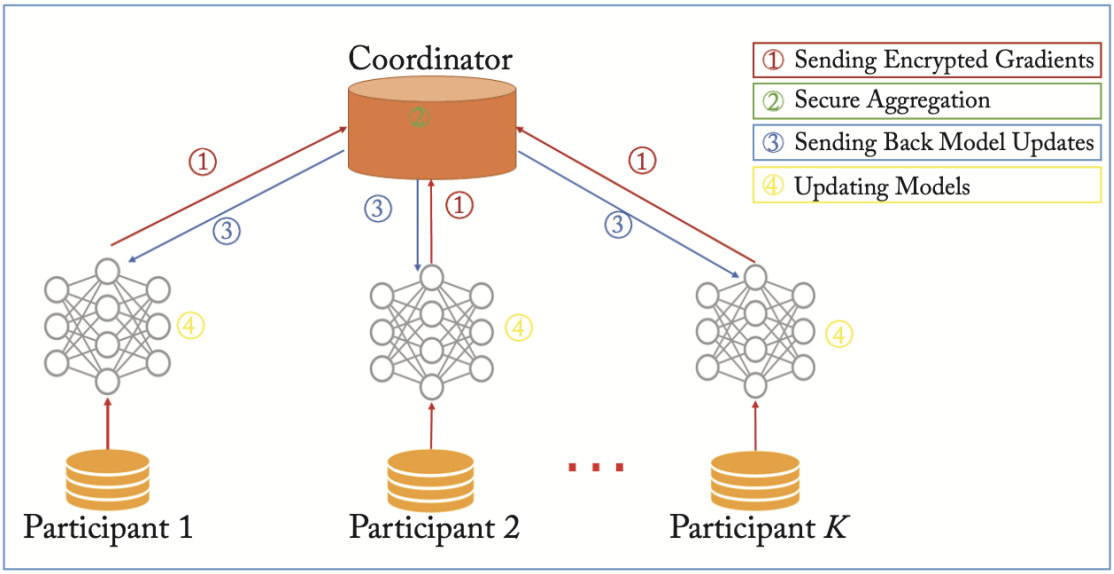


Figura 3.7: Arquitectura de un sistema HFL cliente-servidor. Fuente: [2].

- **Peer-to-peer**

  En esta arquitectura, la principal diferencia con la anterior es que no tenemos un servidor central. Aquí, cada cliente es responsable de entrenar el mismo modelo, usando solo sus datos locales, y una vez entrenado, envía sus actualizaciones a los demás participantes. Entonces, se realiza la agregación y se obtienen nuevos parámetros que actualizan el modelo global. Este proceso continúa hasta que el modelo converge o se llega al número máximo de iteraciones. Una representación gráfica de este tipo de arquitectura se puede consultar en la Figura 3.8.

  Con esta arquitectura puede surgir un problema de confianza y privacidad: ¿Quién envía primero sus pesos a la otra parte? ¿Cuál es la mejor forma de hacerlo? Para solucionarlo encontramos principalmente tres formas en las que se puede intercambiar la información del modelo:

  1. **Intercambio cíclico**: Los clientes se organizan como una cadena. El primero envía sus pesos al segundo, este actualiza los pesos recibidos usando pequeños conjuntos de datos de su propios dataset y envía los pesos actualizados al siguiente cliente de la cadena. Si existen K clientes, el último, el cliente K, enviaría sus parámetros al cliente 1.

  2. **Intercambio aleatorio**: El cliente k-ésimo selecciona un número i aleatoriamente de $\{1, ..., L\}\backslash\{k\}$, con igual probabilidad, y envía los pesos de su modelo a la parte i-ésima. Cuando i recibe estos pesos, los actualiza utilizando pequeños grupos de datos de entrenamiento de su propio conjunto de datos. Luego, el entrenador i selecciona un número j

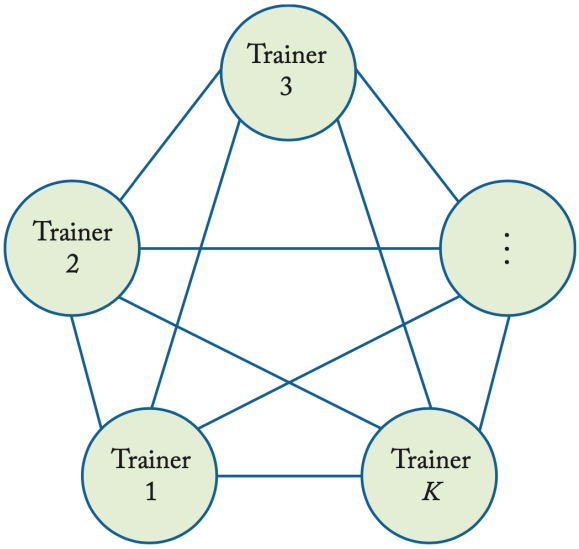


Figura 3.8: Arquitectura de un sistema HFL *peer-to-peer*. Fuente: [2].

de $\{1, ..., L\} \backslash \{i\}$ aleatoriamente y con igual probabilidad, y envía los pesos de su modelo al entrenador j.

3. **Intercambio paralelo**: Se asume un proceso paralelo donde, cuando un cliente actualiza, envía sus actualizaciones a todos los demás clientes. Por tanto, también debe estar receptivo constantemente de recibir actualizaciones del resto de clientes para agregarlas cuando considere necesario.

Una vez vistas las dos arquitecturas diferentes para hacer aprendizaje federado horizontal, vamos a ver ahora cómo se evalúa el modelo global. Esta tarea no es sencilla, pues cada participante puede probar fácilmente el rendimiento del modelo utilizando su conjunto de datos de prueba local, pero no es trivial obtener el rendimiento del modelo global sobre todos los participantes. Por simplicidad, vamos a detallar el proceso a seguir cuando tenemos un problema de clasificación binaria.

Para la primera arquitectura vista, la de Cliente-Servidor, en la cual nos vamos a centrar, podemos obtener el rendimiento del modelo global de la siguiente forma:

- **Paso 1**: El participante k evalúa el rendimiento del modelo utilizando su conjunto de datos local. Para nuestro problema de ejemplo, este paso genera resultados de la forma $(N^k_{TP}, N^k_{FP}, N^k_{TN}, N^k_{FN})$, donde $N^k_{TP}, N^k_{FP}, N^k_{TN}, N^k_{FN}$ denotan el número de verdaderos positivos, el número de falsos positivos, el número de verdaderos negativos y el número de falsos negativos respectivamente, para $k = 1, 2, \ldots, K$.
- **Paso 2**: El participante k envía los resultados de evaluación del modelo local $(N^k_{TP}, N^k_{FP}, N^k_{TN}, N^k_{FN})$ al coordinador, para $k = 1, 2, \ldots, K$.
- **Paso 3**: Después de recopilar los resultados de los K participantes, el coordinador puede calcular el rendimiento del modelo global, que en nuestro

caso de clasificación binaria, puede calcularse así:

$$\frac{\sum_{k=1}^{K} N_{TP}^{k}}{\sum_{k=1}^{K} (N_{TP}^{k} + N_{FN}^{k})}$$

- **Paso 4**: Finalmente, el coordinador envía el rendimiento del modelo global calculado a todos los participantes.

Esta forma que hemos visto es la principal y más extendida, aunque también sería posible evaluar el modelo sobre un conjunto de test propio del servidor, si es que se dispusiese de él, lo cual no siempre es así.

Para la segunda arquitectura, la *peer-to-peer*, calcular el rendimiento global será más complicado ya que no tenemos un servidor central que coordine los procesos. Lo que se suele hacer es llevarlo de alguna forma a la situación de Cliente-Servidor, y eso se consigue nombrando un cliente como un coordinador temporal, y entonces, ejecutar los pasos mostrados arriba.

3.1.5. Federated Averaging (FedAvg)

Una vez estudiadas de qué dos formas puede estar formado un sistema de HFL y habiendo explorado algunas de sus características, se profundizará sobre cómo sería un proceso completo de HFL usando la arquitectura cliente-servidor. Para ello, introducimos el algoritmo Federated Averaging [7], a partir de ahora, FedAvg.

Este algoritmo se emplea para el entrenamiento de modelos federados en sistemas HFL. Como su nombre indica, se basa en realizar una agregación de los parámetros del modelo. En concreto, su uso está restringido a modelos los cuales se puedan expresar como un conjunto de pesos, como por ejemplo las redes neuronales. Así, los árboles de decisión, entre otras arquitecturas, no encajarían en este concepto y no serían elegibles para hacer FedAvg [8].

Para el desarrollo de FedAvg, se tuvo en cuenta que se deben satisfacer diferentes propiedades inherentes a los sistemas federados, tales como:

- Los conjuntos de datos suelen ser no independientes ni idénticamente distribuidos (Non-IID).
- Generalmente hay un gran número de participantes, ya que se necesita un gran conjunto de datos.
- Habitualmente las conexiones entre los clientes y el servidor son lentas, debido a que estas conexiones dependen de la red de internet ordinaria. Por ejemplo, las subidas (de cliente a servidor) van a ser mucho más lentas que las descargas.

Una vez con estas características contempladas, vamos a introducir brevemente el funcionamiento de un sistema de FedAvg, para luego mostrar el algoritmo completo. El proceso sería el siguiente:

1. El servidor inicializa el modelo con unos parámetros aleatorios, o bien, con un modelo ya preentrenado. Una vez inicializado, lo envía a los clientes.
2. Cada cliente, o un porcentaje aleatorio de ellos (no es necesario que cada cliente participe en cada ronda), entrena el modelo con todos sus datos o parte de ellos durante un número prefijado de épocas de entrenamiento locales.
3. Los clientes envían la actualización de los pesos (o gradientes). Con los gradientes está asegurada la convergencia del modelo, mientras que con los pesos no. Aún así, el envío de pesos suele funcionar bien y su uso es predominante frente a los gradientes [2].
4. Una vez los pesos (o gradientes) se envían, el servidor los agrega calculando la media de los pesos y actualiza el modelo. Este proceso se repite hasta que converja o se llegue al numero de rondas máximas prefijadas.

Ya hecha la introducción, estamos en disposición de presentar el algoritmo completo (Algoritmo 1). Para una mayor facilidad de comprensión, dejamos indicada la notación utilizada en la Tabla 3.2.

Símbolo	Descripción
ρ	Porcentaje de clientes que participan en cada ronda global
S	Número de pasos de entrenamiento que cada cliente realiza sobre sus propios datos en cada ronda
M	Tamaño de la fracción de datos locales usados por cada cliente en cada ronda
$\mathbf{w}_t^k$	Conjunto de pesos del modelo local del cliente k en la ronda global t
$\bar{\mathbf{w}}_t$	Conjunto de pesos agregados por el servidor en la ronda global t
$\mathbf{w}_{b,i}^k$	Conjunto de pesos del modelo local del cliente k en la tanda b de la ronda local i
$\mathcal{C}_t$	Conjunto de clientes elegidos aleatoriamente de entre $(1, K\rho)$ para cada ronda global t
n_k	Número de instancias del cliente k
n	Número de instancias totales entre todos los clientes
B	número total de tandas, $B = n_k / M$
b	índice de la tanda de datos utilizados para cada ronda i de entrenamiento local, $b = 1, ..., B$

Tabla 3.2: Notaciones para FedAvg. [2]

Algoritmo 1 FedAvg [7]

1: El coordinador inicializa los pesos del modelo $\bar{\mathbf{w}}_0$, y envía el modelo inicial a todos los participantes.
2: **para** cada ronda global del modelo $t = 1, 2, \ldots$ **hacer**
3: El coordinador determina $\mathcal{C}_t$ aleatoriamente.
4: **para** cada participante $k \in \mathcal{C}_t$ **hacer (en paralelo)**
5: Actualiza los pesos del modelo local $\mathbf{w}^k_{(t+1)} \leftarrow \textit{ParticipantUpdate}(k, \bar{\mathbf{w}}_t)$
6: Envía los pesos del modelo actualizado $\mathbf{w}^k_{(t+1)}$ al coordinador.
7: **fin para**
8: El coordinador hace la media ponderada de los pesos recibidos: $\bar{\mathbf{w}}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}^k_{(t+1)}$.
9: El coordinador revisa si el modelo converge. Si lo hace, le dice a los clientes que paren.
10: El coordinador envía el modelo agregado $\bar{\mathbf{w}}_{t+1}$ a todos los participantes.
11: **fin para**

ParticipantUpdate $(k, \bar{\mathbf{w}}_\mathbf{t})$
Se ejecuta $\forall k = 1, 2, ..., K$

1: Obtiene el último modelo desde el servidor, fija $\mathbf{w}^k_{1,1} = \bar{\mathbf{w}}_\mathbf{t}$
2: **para** cada iteración local $i = 1, ..., S$ **hacer**:
3: *batches* $\leftarrow$ divide aleatoriamente el conjunto D_k en tandas de tamaño M.
4: Recupera los pesos locales de la última iteración: $\mathbf{w}^k_{1,i}$=$\mathbf{w}^k_{B,i-1}$
5: **para** $b = 1, ..., B = \frac{n_k}{M}$ **hacer**
6: Calcula el gradiente local de la tanda b, g^b_k
7: Actualiza los pesos localmente (*Backpropagation*): $\mathbf{w}^k_{b+1,i} \leftarrow \mathbf{w}^k_{b,i} - \eta g^b_k$
8: **fin para**
9: **fin para**
10: Se envían los pesos actualizados al servidor
11: Se establece los nuevos pesos locales $\mathbf{w}^k_{t+1} = \mathbf{w}^k_{B,S}$

Como se puede ver, este algoritmo que hemos presentado no envía gradientes, sino pesos. Cada cliente entrena su modelo, ejecuta retropropagación (u otro método de optimización) para actualizar sus pesos usando sus datos locales y envía de nuevo los pesos al servidor central. El servidor los agrega tal y como se indica en el Algoritmo 1 y los envía de nuevo a los clientes. Esto es lo que se llama *model averaging*. En contraposición, existe el *gradient averaging*, que como su nombre indica envía y agrega gradientes y no pesos. No profundizaremos más en este ultimo método ya que nosotros vamos a utilizar el primero aquí indicado, puesto que su comportamiento en redes neuronales profundas es muy bueno, tal y como se indica en varios estudios [9]. Nosotros presentaremos más adelante un algoritmo basado en redes neuronales, por tanto, la elección de implementar *model averaging* será la más adecuada.

3.2. Aprendizaje Multi-etiqueta

Una vez introducido el HFL de forma profunda y explorado el algoritmo principal utilizado en este entorno, el FedAvg, nos enfocamos ahora en el otro componente esencial de este trabajo: el Aprendizaje Multi-etiqueta, *Multilabel Learning* (MLL).

Dado un conjunto de datos de entrenamiento que consisten en un conjunto de características y una clase asociada, el objetivo de un problema de clasificación es obtener un modelo de ML que pueda asignar la clase adecuada a un patrón desconocido. Tradicionalmente, se asume que cada ejemplo tiene asociada una, y solo una clase. Sin embargo, cada vez hay más problemas de clasificación donde a una misma entrada se le puede asignar varias etiquetas. Esto es el MLL.

Resolver este tipo de problemas implica tratar con nuevas cuestiones que no aparecían en el ML tradicional, como el aumento de combinaciones a considerar en el espacio de salida, el coste computacional de realizar estos modelos, la relación entre etiquetas, etc. Como dijimos en la introducción, un ejemplo sencillo es la clasificación de imágenes. En una imagen pueden aparecer múltiples objetos. Por ejemplo, en una imagen de una playa podrían estar presentes etiquetas como océano, arena, persona, sol, etc.

Para esta revisión vamos a centrarnos en el artículo [10], que recoge las bases y características principales de este tipo de problemas. Además da una definición formal del problema, así como varios ámbitos donde se puede aplicar el MLL.

En una primera aproximación, podríamos decir que un problema de aprendizaje multi-etiqueta presenta las siguientes características:

- El conjunto de etiquetas está predefinido, tiene sentido semántico y es humanamente comprensible.
- Cada ejemplo de entrenamiento puede tener asociado varias etiquetas. También puede tener una o incluso ninguna, aunque este último caso no suele ser común.
- Las etiquetas pueden estar relacionadas, lo cual nos da un conocimiento extra que nos puede ser útil. Por ejemplo, si consideramos los conceptos de "fruta" y "manzana", el hecho de que algo sea identificado como una manzana implica necesariamente que también es una fruta. El hecho de que existan estas relaciones hace que el aprendizaje pueda ser más complejo, ya que lo ideal es ser capaz de modelarlas para tomar ventaja de ellas.
- Los datos pueden estar desbalanceados. Así, puede haber muchos ejemplos asociados con las etiquetas más comunes y haber muy pocos asociados con las etiquetas menos comunes, esto es lo que se llama el desbalanceo en las etiquetas.

Una vez vistas estas características, que nos dan una primera idea superficial del tema, vamos a cambiar el enfoque y vamos a dar una definición formal más rigurosa

y precisa, para así tener una base sólida en la que apoyarnos para desarrollar el resto del documento. Para hacerlo, primero introducimos la notación necesaria:

Sea $\mathcal{X}$ es un espacio de entrada de dimensión d de características numéricas o nominales; $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_q\}$ un espacio de salida de q etiquetas, con $q > 1$, donde cada subconjunto de $\mathcal{L}$ se llama conjunto de etiquetas; $(\mathbf{x}, Y)$, donde $\mathbf{x} = (x_1, ..., x_d) \in \mathcal{X}$, es una instancia de dimensión d que tiene un conjunto de etiquetas asociadas $Y \subseteq \mathcal{L}$. Estas asociaciones de etiquetas también pueden ser representadas como un vector binario de dimensión q: $\mathbf{y} = (y_1, y_2, ..., y_q) = \{0, 1\}^q$ donde cada elemento es 1 si tiene vinculada la etiqueta correspondiente o 0 de otro modo; $S = \{(\mathbf{x}_i, Y_i) | 1 \leq i \leq m\}$ es un conjunto de entrenamiento multi-etiqueta con m ejemplos.

Podemos dividir el MLL en dos problemas fundamentales: la Clasificación Multietiqueta, *Multilabel Classification* y el Ordenamiento de Etiquetas, *Label Ranking*:

1. ***Multilabel Classification (MLC)***: consiste en definir una función $h_{MLC} : \mathcal{X} \to 2^{\mathcal{L}}$. Esta función, dada una instancia de entrada, devolverá un conjunto de etiquetas asociadas (o relevantes) a la instancia de entrada, Y, y el complemento de este conjunto, $\bar{Y}$, el conjunto de etiquetas no asociadas. Así, obtenemos una bipartición del conjunto de etiquetas.

2. ***Label Ranking (LR)***: define una función $f : \mathcal{X} \times \mathcal{L} \to \mathbb{R}$ que devuelve un orden de las posibles etiquetas según la relevancia de estas para una instancia dada $\mathbf{x}$. Así, la etiqueta λ_1 se considera clasificada más alta que λ_1 si $f(\mathbf{x}, \lambda_1) > f(\mathbf{x}, \lambda_2)$. Por ejemplo, un ranking consistente sería poner las etiquetas del conjunto Y más arriba que las etiquetas pertenecientes a $\bar{Y}$.

A partir de esta definición formal, surge la necesidad de comprender las diferencias fundamentales entre la clasificación binaria, multiclase y multi-etiqueta. Estas diferencias se puede ver en la Figura 3.9.

		Single-label binary	Single-label multiclass	Multilabel output				
Example	Features	$Y \in \mathcal{L} = \{0, 1\}$	$Y \in \mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \}$	y_1	y_2	y_3	y_4	$Y \subseteq \mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$
1	$\overline{\mathbf{x}_1}$	1	λ_2	1	1	0	1	$\{\lambda_1, \lambda_2, \lambda_4\}$
2	$\overline{\mathbf{x}_2}$	0	λ_4	0	0	0	1	$\{\lambda_4\}$
3	$\overline{\mathbf{x}_3}$	0	λ_3	0	1	1	1	$\{\lambda_2, \lambda_3, \lambda_4\}$
4	$\overline{\mathbf{x}_4}$	1	λ_1	1	0	1	0	$\{\lambda_1, \lambda_3\}$
5	$\overline{\mathbf{x}_5}$	0	λ_3	0	1	1	0	$\{\lambda_2, \lambda_3\}$
				2	3	3	3	◁Count

Figura 3.9: Clasificación binaria, multiclase y multi-etiqueta. Notar que en multi-etiqueta tengo dos formas de asignarlas, de forma binaria o asociando la etiqueta directamente. Fuente: [10].

Como vemos, la diferencia principal entre ellos es la cantidad de clases que se asignan a cada instancia de datos. En el aprendizaje de etiqueta única (ya sea binario o multiclase), $|Y| = 1$. La clasificación multiclase puede ser vista como un caso particular de *Multilabel Classification* (MLC) donde $h_{MC} : \mathcal{X} \to \mathcal{L}$, mientras que en la clasificación binaria $h_B : \mathcal{X} \to \{0, 1\}$.

La clasificación binaria se utiliza mayormente en situaciones donde las

instancias pueden ser claramente separadas en dos categorías mutuamente excluyentes. Por otra parte, la clasificación multiclase extiende este concepto, permitiendo la asignación a una instancia de una de varias clases posibles. Aquí, las instancias pueden pertenecer a una de varias categorías mutuamente excluyentes. Por último, la clasificación multi-etiqueta es más compleja, donde las instancias pueden tener asociadas múltiples etiquetas simultáneamente, sin restricciones sobre el número de etiquetas que pueden ser asignadas a una instancia en particular. Esto permite una representación más rica de la información, pudiendo además establecer relaciones entre las etiquetas. Dependiendo del problema nos interesará usar una u otra.

En el estado del arte existen multitud de métodos capaces de resolver problemas de clasificación multi-etiqueta, como por ejemplo, [11] o [12]. Aun así, la investigación para resolver problemas de clasificación multi-etiqueta en entornos federados es muy escasa.

3.3. Clasificación multi-etiqueta en entornos federados

Esta sección viene a presentar las propuestas que puedan ser similares a la que más tarde se presentará en este trabajo y de las cuales se hayan extraído ideas o hayan servido de inspiración.

Como se ha dicho en la sección anterior, apenas hay investigación en MLL y FL de forma conjunta. Es más, considerando un escenario donde cada cliente posee subconjuntos de etiquetas distintos, hasta donde llega nuestro conocimiento, no existe nada. Por ejemplo, bajo este marco teórico, podríamos considerar el caso de unidades dentro de un mismo hospital que trabajen con un subconjunto compartido de pacientes, midiendo los mismos parámetros o información. Sin embargo, cada una de estas unidades tiene interés en predecir la probabilidad de desarrollo de enfermedades distintas. Así, en esta sección se comentará una propuesta que, pese a no resolver ese problema exactamente, si resuelve un problema parecido.

La publicación en cuestión es [13]. En ella, se trata de crear un modelo de aprendizaje federado teniendo en cuenta que no todos los clientes tienen por qué tener información de todas las etiquetas. Es importante recalcar que este enfoque no está basado en HFL, sino en VFL, pero que al contrario del enfoque tradicional de este, se asume que existen distintos clientes que poseen información de las clases, en lugar de solo uno. Aunque nosotros nos basaremos en HFL, creemos útil introducir este método ya que nos ha servido de inspiración en varias cuestiones.

Este método considera D poseedores de datos (entre los cuales se distribuyen los atributos) y K poseedores de etiquetas (entre los cuales se distribuyen las etiquetas). No se comparte ningún dato, solo la salida de la última capa de cada cliente (*cut layer*). Este es el enfoque más utilizado para hacer aprendizaje federado vertical, pero tiene un problema: solo se puede usar cuando tenemos un solo poseedor de etiquetas. Por tanto, en el paper se hace uso de un método de agregación para pasar a K dueños de etiquetas.

En este marco, en la publicación se propone que cada poseedor de datos tenga un modelo con una arquitectura única y todos los dueños de las etiquetas tengan la misma arquitectura entre ellos. Los dueños de los datos harían la propagación hacia adelante de la red hasta su *cut layer* y enviarían las salidas a los dueños de las etiquetas. Estos, a su vez, cuantifican la pérdida, realizan *backpropagation* y envían los gradientes correspondientes a los poseedores de los datos. Además, también enviarían sus pesos al servidor de agregación, el cual realiza la agregación y devuelve los pesos actualizados a los dueños de las etiquetas, los cuales se usan en la siguiente iteración del modelo.

Como se ha comentado anteriormente, aunque este enfoque está basado en VFL, encontramos ideas sobre cómo podríamos hacer nuestra propuesta. Por ejemplo, la distinción entre las capas de entrada e intermedias, así como la capa final de la red, en lugar de tratarlas como un todo, nos ha llevado a considerar una estrategia similar para implementar en nuestro propio modelo. Además, en esta publicación se considera que hay más de un propietario de las etiquetas, como así lo haremos nosotros también.

Sin embargo, debemos también incidir en las diferencias con lo que nosotros queremos hacer. Mientras que este paper se centra en problemas de clasificación multiclase, nosotros nos enfocaremos en problemas multi-etiqueta. Además, se distingue entre poseedores de datos y etiquetas, sin que ningún cliente posea ambos, lo que difiere de nuestro caso, ya que cada cliente poseerá datos y etiquetas. Es más, nosotros añadiremos el hecho de que cada cliente posea un subconjunto distinto de las etiquetas, siendo por tanto el objetivo de cada cliente distinto.

En conclusión, esta publicación nos ha servido de ayuda y nos ha proporcionado algunas ideas sobre cómo ejecutar nuestro objetivo, la integración de FL con MLL. Esta tarea no será fácil, pues como ya hemos dicho en anteriores ocasiones, no existe nada hecho para aplicar clasificación multi-etiqueta en entornos federados, y menos aún donde cada cliente tenga etiquetas diferentes.

4. Multilabel FedAvg

Mientras que FedAvg ha demostrado su eficacia en tareas de clasificación binaria o multiclase, su aplicación en el contexto de aprendizaje multi-etiqueta presenta otros problemas y además es un tema del que hay poca investigación hecha. En este escenario, donde las instancias pueden tener asociadas múltiples etiquetas simultáneamente, la agregación de los modelos requiere adaptaciones para conseguir un modelo global, objetivo del FL.

En este trabajo, exploramos la extensión de FedAvg al MLL, proponiendo nuevas estrategias para desarrollar un marco sólido y práctico, el cual llamaremos *Multilabel FedAvg* (MLFedAvg). Para ello hacemos diferentes suposiciones a partir de las cuáles vamos a partir:

- Cada cliente posee sus datos y un número fijo de etiquetas. Así, todos los clientes poseen etiquetas.
- El subconjunto de etiquetas que posee cada cliente es único. Una etiqueta solo puede pertenecer a un cliente. Por tanto, el objetivo de cada cliente es distinto, pudiendo complicar el aprendizaje global respecto al FedAvg clásico. Es más, el posible caso donde todos los clientes tuvieran todas las etiquetas (el cual es un problema más sencillo de resolver, ya que todos poseen información de todas ellas), sería simplemente una extensión sencilla de esta propuesta.
- El servidor central trabaja sobre un modelo global con una estructura determinada para todos los clientes. Sin embargo, dicho modelo alojado en el servidor no será capaz de predecir por sí solo, ya que la arquitectura de este no es suficiente para ello, como veremos a continuación.

Una vez hechas las primeras consideraciones, vamos a presentar nuestro modelo. Asumimos que tenemos K clientes. Cada cliente tiene un conjunto de datos, $\mathcal{D}_k$, formado por un vector de características $\mathbf{x}_k$, y un vector de etiquetas asociadas $\mathbf{y}_k$. Así, $\mathcal{D}_k = \{\mathbf{x}_{k,i}, \mathbf{y}_{k,j}\}$ con $i = 1, ..., N_k, j = 1, ..., M_k$, siendo N_k el número de instancias de los datos del cliente k, y M_k el número de etiquetas que posee el cliente k. Como hemos dicho, está configuración es común para todos los clientes, puesto que todos poseen etiquetas. El número de instancias N_k que pueda tener cada cliente sí es variable y en principio no tiene restricciones.

Por otro lado, tenemos un espacio de etiquetas $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_q\}$, con $q > 1$, donde cada subconjunto de $\mathcal{L}$ vendrá denominado por Y. Es más, como dijimos en

las consideraciones previas, consideraremos que el subconjunto de etiquetas que posee cada cliente es único. En consecuencia, vamos a determinar como Y_k el conjunto de etiquetas que posee el cliente k, y además, estamos en disposición de decir que $Y_k \cap Y_{k'} = \emptyset$, $\forall k, k' \in K$ con $k \neq k'$.

De manera resumida, los pasos que va a seguir nuestro modelo son:

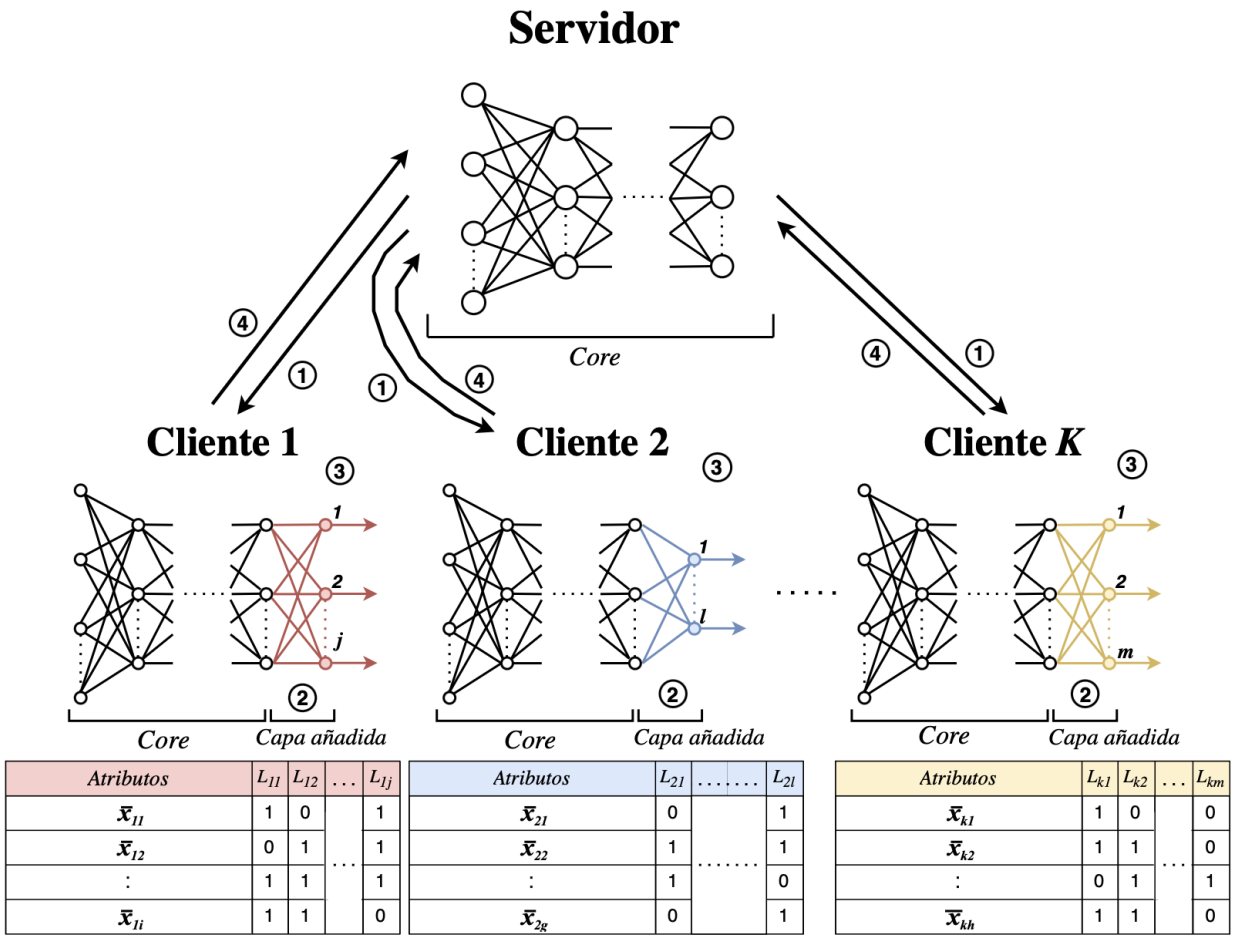
- **Paso 1.** El servidor inicializa el proceso y envía un modelo de red neuronal a los clientes, el modelo *core*.
- **Paso 2.** Cada cliente k añade una última capa al modelo, que servirá como capa de salida propia. Esta última capa tendrá tantos nodos como etiquetas posea el cliente. Los parámetros correspondientes a esta última capa nunca son compartidos, permanecen locales en cada cliente.
- **Paso 3.** Cada cliente k actualiza el modelo completo con sus datos locales.
- **Paso 4.** Los clientes envían al servidor los pesos correspondientes al modelo que este les ha enviado, es decir, los pesos de todas las capas menos de la última, los pesos del *core*.

Pensamos que es una propuesta interesante dentro de una temática donde no hay nada hecho como es el aprendizaje federado con datos multi-etiqueta. Al enviar solo los pesos del modelo *core* guardamos la privacidad inherente a estos sistemas. Además, de esta forma los pesos de la última capa son totalmente únicos para cada cliente, adaptando el modelo a su conjunto específico de etiquetas.

Para ilustrar estos pasos de forma más intuitiva se presenta la Figura 4.1. En esta figura, los pasos son representados con números en círculos. Como dijimos anteriormente, cada cliente k tiene un conjunto de instancias $\bar{x}_{ki}$, con $i = 1, ..., N_k$, al cual se le asocian las etiquetas L_{km}, con $j = 1, ..., M_k$. Notar el cambio de notación, ya que dependiendo de a qué cliente le corresponde cada etiqueta, hemos pasado de representar el conjunto de etiquetas como $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_q\}$ a representarlo como:

$$\mathcal{L} = \{L_{11}, L_{12}, ..., L_{1j}, L_{21}, L_{22}, ..., L_{2l}, ..., L_{k1}, L_{k2}, ..., L_{km}\}$$

con $j = 1, ..., M_1 = |Y_1|$, $l = 1, ..., M_2 = |Y_2|$, y $m = 1, ..., M_k = |Y_k|$. Además, $M = \sum_{k=1}^{K} M_k$ es la suma del número de etiquetas de todos los clientes.



Atributos	L_{11}	L_{12}	...	L_{1j}
$\bar{x}_{11}$	1	0		1
$\bar{x}_{12}$	0	1	...	1
:	1	1		1
$\bar{x}_{1i}$	1	1		0

Atributos	L_{21}	.....	L_{2l}
$\bar{x}_{21}$	0		1
$\bar{x}_{22}$	1	.....	1
:	1		0
$\bar{x}_{2g}$	0		1

Atributos	L_{k1}	L_{k2}	...	L_{km}
$\bar{x}_{k1}$	1	0		0
$\bar{x}_{k2}$	1	1	...	0
:	0	1		1
$\bar{x}_{kh}$	1	1		0

Figura 4.1: Propuesta *Multilabel FedAvg* (MLFedAvg). Indicamos las etiquetas asociadas a cada ejemplo usando notación binaria.

Una vez ilustradas las ideas principales de nuestra propuesta, estamos en condiciones ahora de presentar rigurosamente el algoritmo completo MLFedAvg (Algoritmo 2), el cual implementaremos en el siguiente capítulo. La notación usada se puede consultar en la Tabla 4.1.

En el algoritmo se puede observar una función $f(\mathbf{w}_{d,t+1}^k)$ que define el tipo de agregación a realizar. En este trabajo se presentan tres tipos:

$$f(\mathbf{w}_{d,t+1}^k) = \begin{cases} \sum_{k=1}^K \mathbf{w}_{d,t+1}^k & \text{(Agregación 0)} \\ \sum_{k=1}^K \frac{n_k}{n} \mathbf{w}_{d,t+1}^k & \text{(Agregación 1)} \\ \sum_{k=1}^K \frac{n_k}{N} \frac{|Y_k|}{M} \frac{1}{\sum_{k=1}^K \left(\frac{n_k}{N} \times \frac{|Y_k|}{M} \right)} \mathbf{w}_{d,t+1}^k & \text{(Agregación 2)} \end{cases} \tag{4.1}$$

En el primer caso, se calcula una media simple sin considerar la proporción de datos que cada cliente tiene, por tanto, se considera igual la aportación de cada

parte. Respecto a la segunda forma, se está calculando una media ponderada donde los pesos están determinados por la proporción de datos que cada cliente aporta al total de datos. Así, se refleja la contribución de cada cliente al modelo en función del número de instancias que tiene. Por último, se tiene la agregación más compleja. En ella se tiene en cuenta tanto la cantidad de datos como el número de etiquetas de cada cliente. Como se puede ver en la ecuación 4.1, a más etiquetas, mayor es la contribución de ese cliente a la agregación. También se incluye una normalización para asegurar que la suma de los factores de ponderación sea 1.

Aunque en principio vamos a considerar las tres formas de agregación, en la Sección 5 se obtendrán resultados para cada una de estas agregaciones, y mediante un estudio de los mismos, se determinará cuál es la mejor para el problema que nos ocupa.

Símbolo	**Descripción**
ρ	Porcentaje de clientes que participan en cada ronda global
S	Número de pasos de entrenamiento que cada cliente realiza sobre sus propios datos en cada ronda
M	Tamaño de la fracción de datos locales usados por cada cliente en cada ronda
$\mathbf{w}^k_{out,t}$	Conjunto de pesos de la capa de salida del modelo local del cliente k en la ronda global t
$\mathbf{W}_t$	Conjunto de pesos agregados por el servidor en la ronda global t
$\mathbf{w}^k_t$	Conjunto de pesos del modelo local del cliente k en la ronda global t
$\mathbf{w}^k_{d,t}$	Conjunto de pesos de todas las capas intermedias (*core*) del modelo local del cliente k en la ronda global t
$\mathbf{w}^k_{b,i}$	Conjunto de pesos del modelo local del cliente k en la tanda b de la ronda local i
$\mathcal{C}_t$	Conjunto de clientes elegidos aleatoriamente de entre $(1, K\rho)$ para cada ronda global t
n_k	Número de instancias del cliente k
n	Número de instancias totales entre todos los clientes
B	Número total de tandas, $B = n_k / M$
b	Índice de la tanda de datos utilizados para cada ronda i de entrenamiento local, $b = 1, ..., B$
Y_k	Conjunto de etiquetas del cliente k
M	Suma del número de etiquetas de todos los clientes

Tabla 4.1: Notaciones para MLFedAvg.

Algoritmo 2 MLFedAvg

1: El coordinador inicializa los pesos del modelo *core* $\mathbf{W}_0$, y envía el modelo inicial a todos los participantes.
2: **para** cada ronda de actualización del modelo global $t = 1, 2, \ldots$ **hacer**
3: El coordinador determina $\mathcal{C}_t$ aleatoriamente.
4: **para** cada participante $k \in \mathcal{C}_t$ **hacer (en paralelo)**
5: Añade una capa de salida con $|Y_k| = M_k$ neuronas, cuyos pesos son $\mathbf{w}^k_{out,t}$
6: Obtiene los pesos de la red local completa $\mathbf{w}^k_t \leftarrow$ Concatena $\mathbf{W}_t$, $\mathbf{w}^k_{out,t}$
7: Actualiza los pesos del modelo local $\mathbf{w}^k_{(t+1)} \leftarrow$ *ParticipantUpdate*$(k, \mathbf{W}_t)$
8: Selecciona el conjunto de pesos de todas las capas menos de la última: $\mathbf{w}^k_{d,t+1} \leftarrow \{\mathbf{w}^k_{t+1} \setminus \mathbf{w}^k_{out,t+1}\}$
9: Envía los pesos $\mathbf{w}^k_{d,t+1}$ al coordinador.
10: **fin para**
11: El coordinador agrega los pesos recibidos: $\mathbf{W}_{t+1} \leftarrow f(\mathbf{w}^k_{d,t+1})$.
12: Revisa si el modelo converge. Si lo hace, le dice a los clientes que paren.
13: El coordinador envía el modelo agregado $\mathbf{W}_{t+1}$ a todos los participantes.
14: **fin para**

ParticipantUpdate $(k, \mathbf{W}_t)$
Se ejecuta $\forall k = 1, 2, ..., K$

1: **para** cada iteración local $i = 1, ..., S$ **hacer**:
2: *batches* $\leftarrow$ divide aleatoriamente el conjunto D_k en tandas de tamaño M.
3: Recupera los pesos locales de la última iteración: $\mathbf{w}^k_{1,i}$=$\mathbf{w}^k_{B,i-1}$
4: **para** $b = 1, ..., B = \dfrac{n_k}{M}$ **hacer**
5: Calcula el gradiente local de la tanda b, g^b_k
6: Actualiza los pesos localmente (*Backpropagation*): $\mathbf{w}^k_{b+1,i} \leftarrow \mathbf{w}^k_{b,i} - \eta g^b_k$
7: **fin para**
8: **fin para**
9: Se establece los nuevos pesos locales $\mathbf{w}^k_{t+1} = \mathbf{w}^k_{B,S}$

5. Experimentación

En esta sección se presentarán los diferentes experimentos y ensayos a las que se examinará la propuesta MLFedAvg. Primero, seleccionaremos los conjuntos de datos utilizados para esta evaluación. Posteriormente, evaluaremos tanto nuestro modelo federado como los modelos sin federar, es decir, modelos independientes para cada cliente. De este modo, se intentará probar la validez del método así como identificar sus puntos débiles de cara a trabajos futuros. Cabe decir que la comparación final de resultados solo será contra modelos sin federar pues no hay ningún trabajo en la literatura sobre aprendizaje federado multi-etiqueta.

5.1. Configuración de los experimentos

Al no haber trabajo previo, las diferentes evaluaciones y pruebas se realizarán únicamente sobre dos tipos de modelos:

1. Modelos que siguen MLFedAvg, donde los clientes colaboran de la forma descrita en la Sección 4.
2. Modelos independientes, donde cada cliente entrena su modelo con sus propios datos y no se comparte ningún tipo de información o parámetros.

La comparación entre estos dos enfoques permitirá evaluar la efectividad de MLFedAvg frente a los modelos no federados. Específicamente, se analizarán las métricas *Subset Accuracy* (SA) y F1-Macro. La elección de SA se debe a su amplio uso en la literatura. Sin embargo, es una métrica demasiado estricta, ya que requiere que todas las etiquetas de la predicción sean correctas para considerar un acierto. Por ello, consideramos oportuno estudiar otra métrica más, en este caso F1-Macro, que nos parece interesante ya que da el mismo peso a todas las etiquetas y por tanto las etiquetas minoritarias se consideran con igual peso que las mayoritarias.

El código de la implementación de MLFedAvg, así como funciones auxiliares y conjuntos de datos utilizados se puede encontrar en: https://github.com/daviddb11/MLFedAvg.

5.1.1. Conjuntos de datos

Para esta experimentación se han utilizado diez conjuntos de datos multi-etiqueta[1], cuyas características se pueden consultar en la Tabla 5.1. Estos conjuntos presentan una gran diferencia entre el número de ejemplos, así como características diversas y distintas distribuciones de etiquetas, lo que permite representar diferentes contextos y asegurar una evaluación justa y lo más real posible.

Dataset	Ejemplos	Atributos	Etiquetas	Cardinalidad	Densidad
scene [14]	2407	294	6	1.07	0.179
yeast [14]	2417	103	14	4.24	0.303
HumanPseAAC [15]	3106	440	14	1.19	0.085
20NG [15]	19300	1006	20	1.03	0.051
tmc2007_500 [14]	28600	500	22	2.22	0.101
slashdot [15]	3782	1079	22	1.18	0.054
YahooEducation [15]	12030	27530	33	1.46	0.044
mediamill [14]	43910	120	101	4.38	0.043
reutersK500 [15]	6000	500	103	1.46	0.014
Corel5k [14]	5000	499	374	3.52	0.009

Tabla 5.1: Características de los conjuntos de datos utilizados en la experimentación, donde se incluye el número de ejemplos, atributos, etiquetas, cardinalidad y densidad para cada dataset. La cardinalidad mide el número promedio de etiquetas asociadas a cada instancia y la densidad es la cardinalidad repartida entre el número de etiquetas.

5.1.2. Simulación de datos y etiquetas en el entorno federado

Estos conjuntos de datos necesitan ser repartidos entre los distintos clientes para simular un entorno de aprendizaje federado. Este reparto es un paso esencial, pues con él se va a simular el entorno de un sistema de aprendizaje federado, pues no disponemos de los recursos necesarios para acceder a una arquitectura federada real.

Así, por un lado repartiremos las etiquetas, y posteriormente, según las etiquetas que posea cada cliente, se repartirán los ejemplos entre los distintos clientes. Antes de este reparto, obtenemos un conjunto de test de cada conjunto de datos, que vamos a llamar conjunto de test global (CTG). Este conjunto se obtiene aleatoriamente, al que se le asigna un tamaño del 10 % del conjunto total. Al

[1]Extraídos de https://www.uco.es/kdis/mllresources/ y de la librería *scikit-multilearn*: http://scikit.ml/datasets.html

restante, el 90 % del conjunto de datos, le llamaremos conjunto de entrenamiento global (CEG).

Ahora se tratarán tanto las etiquetas como los ejemplos que se asignan a cada cliente. Los detalles del reparto de etiquetas se pueden consultar en la Tabla 5.2, donde se puede observar el reparto que se ha realizado entre los distintos clientes para cada conjunto de datos. Así se presentan diferentes configuraciones en términos de la cantidad de clientes y la distribución de etiquetas, intentando considerar una amplia cantidad de escenarios que reflejen tanto distribuciones balanceadas como no balanceadas.

Dataset	Etiquetas	Clientes	Balanceado	Reparto
scene [14]	6	2	✓	3, 3
yeast [14]	14	4	✓	4, 4, 3, 3
HumanPseAAC [15]	14	3	×	8, 4, 2
20NG [15]	20	5	✓	5, 4, 4, 4, 3
tmc2007_500 [14]	22	5	✓	5, 5, 4, 4, 4
slashdot [15]	22	4	×	10, 6, 4, 2
YahooEducation [15]	33	5	×	10, 8, 6, 5, 4
mediamill [14]	101	10	×	20, 15, 13, 10, 10, 10, 10, 5, 5, 3
reutersK500 [15]	103	8	✓	13, 13, 13, 13, 13, 13, 13, 12
Corel5k [14]	374	4	×	234, 80, 40, 20

Tabla 5.2: Reparto de etiquetas por conjunto de datos. Clientes indica la cantidad de participantes del modelo federado para cada conjunto de datos. Reparto expresa el número de etiquetas asignadas cada cliente. Balanceado señala si este reparto es equitativo o no.

El siguiente paso sería asignar los ejemplos de los que va a disponer cada cliente. La notación seguida en esta parte es la presentada en la Sección 4. Para ello, el conjunto de datos (ya sin CTG) se divide en K partes iguales. Cada una de estas partes se le asigna a cada cliente. Por tanto, el cliente k tendría un conjunto de ejemplos de $n_k = n/K$ filas.

Por otro lado, a cada cliente se le asigna un subconjunto de etiquetas, desechando las demás. El número de etiquetas que posee el cliente k ya viene prefijado por el reparto descrito en la Tabla 5.2. Sin embargo, la asignación concreta de etiquetas a cada cliente se hace de forma aleatoria. De ese mismo modo, tras repetir los experimentos con diferentes semillas, la asignación de etiquetas será distinta en cada caso.

Más detalladamente, el proceso seguido se puede describir de la siguiente manera:

1. El conjunto de datos se puede representar como dos matrices: la matriz de características y la matriz objetivo. La matriz de características tiene n

instancias, con V número de atributos. Se representaría como una tabla de datos de $n \times V$. Por otro lado, el conjunto de datos tendrá M etiquetas, por tanto la matriz objetivo tendrá dimensiones $n \times M$. Notar que en la matriz objetivo se está usando la representación binaria de etiquetas (Figura 3.9).

2. La matriz de características se divide aleatoriamente en K partes, asignando a cada cliente un subconjunto de tamaño $n_k \times V$. Como no puede ser de otra forma, las instancias de la matriz objetivo correspondientes a las instancias de la matriz de características también se asignan al cliente en cuestión. Por tanto, cada cliente tendrá una matriz objetivo de tamaño $n_k \times M$. En esta fase, cada cliente posee aún todas las M etiquetas del conjunto de datos, ya que todavía no se ha decidido cuáles etiquetas estarán presentes en los datos de cada cliente.

3. Para cada cliente, se seleccionan aleatoriamente un subconjunto de las M etiquetas disponibles (ver Tabla 5.2). Esto significa que cada cliente k tendrá una matriz objetivo de tamaño $n_k \times M_k$, donde M_k es el número de etiquetas asignadas al cliente k. De esta manera, cada cliente solo posee un número específico de etiquetas.

4. Por último, como solo son de interés las instancias que tengan al menos alguna etiqueta asignada, para cada cliente se eliminan los ejemplos que tengan todas las etiquetas con valor cero. Al realizar esta operación, estamos reduciendo el número de ejemplos que posee cada cliente. Así, cada cliente tendrá un conjunto de datos local (CL) único.

Para terminar con el tratamiento de datos y la asignación de los mismos a cada cliente, se añade un ruido gaussiano a CL. El objetivo es tener una simulación lo más cercana a la realidad posible de un entorno federado, y aunque en entornos de HFL las características son iguales, creemos que es indicado añadir este ruido ya que si no todos los clientes tendrían una distribución igual de los datos, lo que nos parece alejado de la realidad. Así, estaríamos introduciendo perturbaciones naturales en los datos de los distintos clientes. Cabe decir que por supuesto este ruido solo solo se añade a las características, no a las etiquetas.

Para implementarlo en los datos, primero generamos una lista de desviaciones estándar que tenga tantos elementos como número de clientes (Tabla 5.2):

$$L = (\sigma_1, \sigma_2, \ldots, \sigma_K) \tag{5.1}$$

Esta lista L se genera aleatoriamente con valores elegidos desde 0 a 0.5. Cada elemento de la misma determinará la varianza añadida a los datos de cada cliente. Así, el ruido añadido a los datos de cada cliente sería:

$$\begin{aligned} &\text{Cliente } 1 \sim \mathcal{N}(0, \sigma_1^2), \\ &\text{Cliente } 2 \sim \mathcal{N}(0, \sigma_2^2), \\ &\qquad \vdots \\ &\text{Cliente } K \sim \mathcal{N}(0, \sigma_K^2) \end{aligned}$$

Además, al añadir ruido gaussiano se estaría modelando datos No-IID, que como se estudió en la Sección 3, son el tipo de datos presentes en la mayoría de casos reales. Por tanto, se estaría añadiendo otra característica más para una representación realista del problema que estamos tratando.

Téngase en cuenta que todo este proceso se realiza para simular la partición de datos multi-etiqueta en entornos federados, y que podría haberse seguido cualquier otro. En cualquier caso, este proceso es independiente del algoritmo MLFedAvg propuesto en el trabajo, cuya operación será la misma sea cual sea la distribución de los datos en cada cliente.

Para entrenar nuestros modelos se necesita primero obtener los distintos conjuntos de prueba, validación y test para cada cliente, los cuales se obtienen a partir de CL. El conjunto de entrenamiento local (CEL) tendrá el 60 % de CL, mientras que los conjuntos de validación (CVL) y test (CTL) tendrán un 20 % cada uno de CL. Las divisiones realizadas sobre el dataset original se pueden consultar en la Figura 5.1.

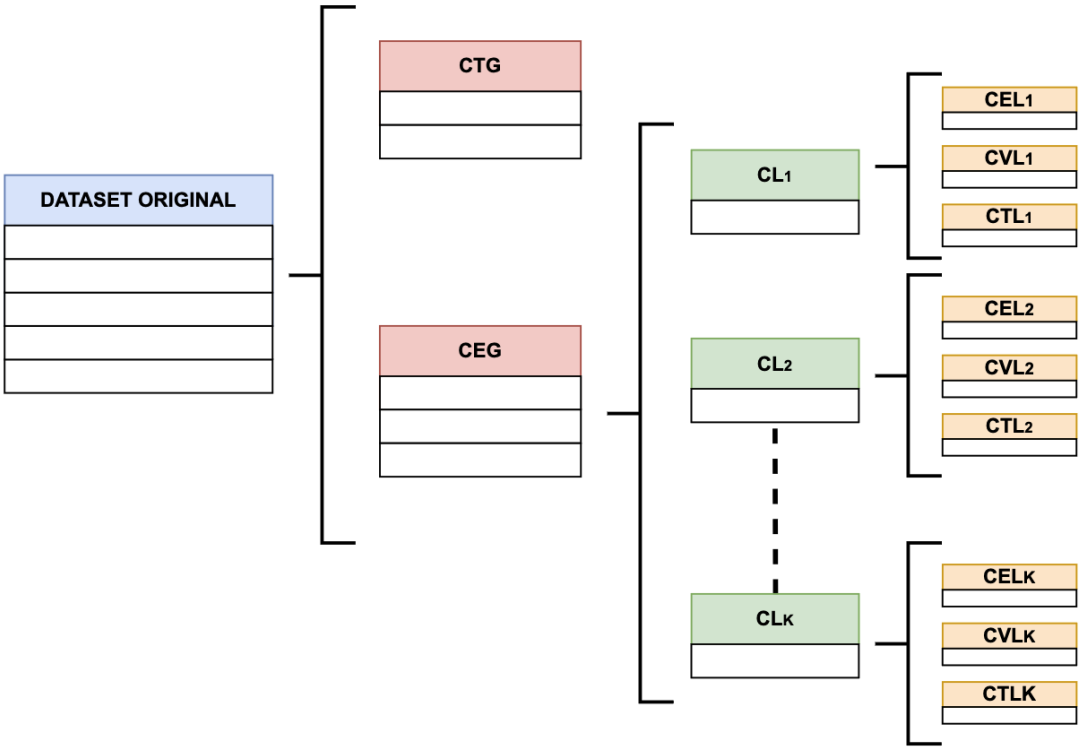


Figura 5.1: División de los datos. CTG: Conjunto de Test Global. CEG: Conjunto de Entrenamiento Global. CL_k: Conjunto Local del cliente k. CEL_k: Conjunto de Entrenamiento Global del cliente k. CVL_k: Conjunto de Validación Local del cliente k. CTL_k: Conjunto de Test Local del cliente k.

5.1.3. Entrenamiento del modelo federado

Para el entrenamiento de nuestro modelo, inicializamos el proceso federado con 1000 rondas, donde en cada ronda cada cliente ejecuta 1 época de

entrenamiento. Además, añadimos las siguientes condiciones de parada, que deben cumplirse simultáneamente para detener el entrenamiento:

1. La media de SA sobre CVL acumula dos rondas federadas consecutivas disminuyendo.
2. La media de la pérdida de validación sobre CVL acumula dos rondas federadas consecutivas aumentando.

Aquí el concepto de media hace referencia a lo siguiente: al final de cada ronda de entrenamiento federado, cada cliente calcula SA y pérdida de validación basadas en su rendimiento local sobre el CVL. La media aritmética de estas métricas se calcula tomando los valores individuales de todos los clientes y promediándolos.

Al añadir esa condición de parada, aunque en un principio consideramos un entrenamiento con 1000 rondas de aprendizaje federado, el entrenamiento puede detenerse antes de las 1000 rondas. Funciona como una clase de *early stopping* adaptado a aprendizaje federado. La ronda en la cual se detiene el federado, la cual llamaremos ronda parada (RP), se guarda para luego ser utilizada en el entrenamiento de los modelos sin federado. Por otro lado, la tasa de aprendizaje no es fija, sino que varía según el conjunto de datos, ya que depende del tamaño, la calidad, y la complejidad del conjunto de datos en cuestión.

Además, no prefijamos ningún tipo de agregación (ver Ecuación 4.1), sino que ejecutamos un entrenamiento completo con cada una. Después, se realizaran pruebas estadísticas para determinar qué agregación es la mejor.

5.1.4. Entrenamiento de modelos sin federado

Una vez entrenados los modelos en el entorno federado (de la forma expuesta en la Figura 4.1), pasamos a entrenar los modelos independientes de cada cliente. En esta fase no se comparte ningún tipo de información; cada cliente entrena su modelo con sus propios datos (CEL). Además, para posteriormente hacer una comparación justa de los resultados, se tienen en cuenta las siguientes consideraciones:

- Las épocas de entrenamiento se fijan en el valor obtenido al multiplicar las rondas federadas por el número de épocas en cada ronda. Por tanto, serían 1000 épocas.
- Se utiliza *early stopping*, que hace que el entrenamiento del modelo de cada cliente se detenga si el número de épocas donde la pérdida sobre CVL (*validation loss*) no mejora es igual a RP.
- La arquitectura de la red del modelo de cada cliente es la misma que en el entorno federado.

5.1.5. Métricas

Como se ha comentado anteriormente, en nuestro estudio se emplearán dos métricas para evaluar el rendimiento de los modelos: SA y F1-Macro. A continuación, se da una descripción detallada de estas métricas, así como de sus características.

SA es una métrica basada en ejemplos (*example based*). Las métricas basadas en ejemplos calculan la métrica para cada instancia de los datos y luego promedian su valor por el número de instancias. Este tipo de métricas otorga el mismo peso a todas las instancias, lo que significa que no consideran un posible desequilibrio de las etiquetas, que suele ser un problema común en la MLC. En el contexto de MLC, SA mide la proporción de instancias cuya predicción es completamente correcta, es decir, que todas las etiquetas han sido clasificadas correctamente. Por tanto, se puede catalogar como una métrica bastante estricta. Matemáticamente se puede escribir como:

$$SA = \frac{1}{m}\sum_{i=1}^{m}[\![(Y_i = \hat{Y}_i)]\!] \tag{5.2}$$

donde m es el conjunto total de instancias, Y_i es el conjunto de etiquetas asociada a la instancia i, $\hat{Y}_i$ es el conjunto de etiquetas predichas para la instancia i y $[\![(Y_i = \hat{Y}_i)]\!]$ devuelve 1 si se cumple la condición entre paréntesis y 0 en caso contrario [3].

Respecto a la otra métrica de evaluación utilizada, F1-Macro, es una métrica basada en etiquetas (*Label-based metrics*). Las métricas de evaluación basadas en etiquetas, a diferencia de las métricas basadas en ejemplos, se calculan en función de las etiquetas en lugar de las instancias. Es más, cualquier métrica de evaluación para clasificación binaria puede ser adaptada para escenarios de múltiples etiquetas, convirtiéndose en una métrica basada en etiquetas en problemas MLC. Es decir, se está extrapolando el uso de métricas de clasificación binaria a problemas MLC.

Además, las métricas basadas en etiquetas se pueden calcular siguiendo dos enfoques diferentes: promedio micro y promedio macro. Si se tiene una métrica de evaluación para clasificación binaria calculada en función de los verdaderos positivos (TP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (TN) de la matriz de confusión (Tabla 5.3), las métricas con promedio micro combinan las matrices de confusión de todas las etiquetas y luego calculan la métrica; las métricas con promedio macro calculan la métrica para cada etiqueta y luego promedian sus valores para todas las etiquetas. Como consecuencia, las métricas con promedio micro están más influenciadas por las etiquetas más frecuentes, mientras que las métricas con promedio macro dan la misma importancia a todas las etiquetas en su cálculo [3].

Aplicando estas consideraciones a la métrica F1-Macro, su cálculo se realiza combinando las métricas precisión y *recall*:

	Pred.Positiva	**Pred.Negativa**
Real Positiva	TP	FN
Real Negativa	FP	TN

Tabla 5.3: Matriz de Confusión. En problemas MLC se tiene una por cada etiqueta.

$$\text{F1-Macro} = 2 \times \frac{\text{P-Macro} \times \text{R-Macro}}{\text{P-Macro} + \text{R-Macro}} \tag{5.3}$$

donde P-Macro y R-Macro son las métricas de precisión y *recall* macro promediadas, definidas según las siguientes ecuaciones:

$$\text{P-Macro} = \frac{1}{q} \sum_{l=1}^{q} \frac{\text{TP}_l}{\text{TP}_l + \text{FP}_l} \tag{5.4}$$

$$\text{R-Macro} = \frac{1}{q} \sum_{l=1}^{q} \frac{\text{TP}_l}{\text{TP}_l + \text{FN}_l} \tag{5.5}$$

5.1.6. Test estadísticos para el análisis de resultados

En esta sección se presentan dos métodos estadísticos utilizados para interpretar y analizar los resultados obtenidos en la evaluación de los modelos. Estas herramientas son de gran ayuda, pues permiten extraer conclusiones y tomar decisiones de manera más fundamentada.

Para obtener los resultados y realizar dichos análisis, el experimento concreto realizado ha sido el siguiente:

- Se crea el modelo *core* como una red que tiene por entrada tantas neuronas como atributos tenga el conjunto de datos en cuestión, seguida por una capa de 100 neuronas con función de activación RELU, y por último otra capa de 20 neuronas con función de activación RELU. Luego, cada cliente añade una última capa con tantas neuronas como etiquetas posea (ver Figura 4.1) con función de activación sigmoide.
- Para cada conjunto de datos, se realizan varias ejecuciones con semillas aleatorias distintas, buscando así eliminar cualquier tipo de dependencia de las mismas. El número de semillas utilizadas para cada conjunto de datos se puede consultar en la Tabla 5.4. Nótese que el número de ejecuciones decae con los conjuntos de datos más complejos, debido a restricciones de tiempo y computacionales.
- En cada ejecución, el entrenamiento de los modelos en el entorno federado se realiza tres veces, una con cada agregación (Ecuación 4.1). Con la RP obtenida en cada agregación se pasa a entrenar los modelos sin federado. El número

concreto de ejecuciones y entrenamientos realizados con cada conjunto de datos se puede consultar en la Tabla 5.4.

Dataset	Semillas	Entrenamientos totales
scene [14]	30	90
yeast [14]	30	90
mediamill [14]	30	90
tmc2007_500 [14]	30	90
Corel5k [14]	30	90
20NG [15]	8	24
HumanPseAAC [15]	5	15
slashdot [15]	5	15
YahooEducation [15]	5	15
reutersK500 [15]	5	15

Tabla 5.4: Ejecuciones por conjunto de datos. Semillas indica el número de semillas utilizadas, que serán las ejecuciones. Entrenamientos se refiere al número total de entrenamientos realizados, que son tres por cada ejecución (uno por cada agregación). La limitación en los recursos computacionales ha llevado a que ciertos conjuntos de datos se ejecuten un menor número de veces.

Una vez obtenidos los resultados, se utilizará el test de Friedman [16]. Este test es una prueba estadística no paramétrica, es decir, adecuada para datos los cuales no se sabe (o no se puede asumir) la distribución que siguen, como es el caso de nuestros resultados. A diferencia de las pruebas paramétricas, que requieren varias condiciones sobre los datos (como que sigan una distribución normal), las pruebas no paramétricas son más versátiles y tienen un rango de aplicación más amplio.

Si consideramos un conjunto de datos, donde los datos están divididos en grupos, el test de Friedman se utiliza para responder a la siguiente pregunta: ¿Existe diferencia entre los grupos? Por ejemplo, si los grupos son resultados de modelos utilizando aprendizaje federado frente a resultados de modelos sin aprendizaje federado, la pregunta sería: ¿hay una diferencia significativa entre usar aprendizaje federado y no usarlo?

Para responder esta pregunta, el test de Friedman funciona de la siguiente manera:

1. Se asume la hipótesis nula, es decir, no hay diferencia entre los grupos.
2. Se tiene un conjunto de datos de la forma $\{x_{ij}\}_{n\times k}$, esto es, una matriz con n filas y k columnas (grupos).
3. Para cada fila i ($i = 1, 2, \ldots, n$), asigna un valor a cada observación en función de los datos. Al mejor valor de la fila i le otorga un 1. Al segundo mejor valor, un 2. Así sucesivamente.

4. Se reemplazan los datos originales con una nueva matriz de rangos $\{r_{ij}\}_{n\times k}$, donde r_{ij} es el rango de x_{ij} dentro de la fila i.
5. Se calculan los valores promedio de rangos para cada columna j:

$$\bar{r}_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij}$$

6. Se calcula el resultado del test de Friedman como:

$$Q = \frac{12n}{k(k+1)}\sum_{j=1}^{k}\left(\bar{r}_j - \frac{k+1}{2}\right)^2$$

7. Se utiliza el valor crítico p tabulado de la distribución χ^2 ($p = P(\chi^2 \geq Q)$) que vendrá dado según los grupos considerados en los datos y ajustado por un parámetro α de confianza[2]. En nuestro caso, α=0.05.
8. Si el valor p es menor o igual que α, se rechaza la hipótesis nula y hay diferencia entre los grupos. Si el valor p es mayor que α, no se rechaza la hipótesis nula, es decir, no hay suficiente evidencia para concluir que hay diferencias significativas entre los tratamientos [17].

Cabe decir que, en el caso donde se rechaza la hipótesis nula, el resultado tendrá un grado de confianza de 1-α, es decir, en nuestro caso los resultados se obtendrán con un 95 % de confianza.

También, es importante recalcar que, en el caso que haya diferencia entre los grupos, esta puede ser entre uno y los demás, o entre varios. Esa cuestión no lo podemos concluir a partir de este test. Por ello, en casos donde el test de Friedman indica que hay diferencias significativas entre grupos (y estos son más de dos), se utiliza a continuación un algoritmo post-hoc, el test de Nemenyi.

El objetivo principal del test de Nemenyi es identificar específicamente qué pares de grupos son significativamente diferentes entre sí. En esta prueba, se calcula la diferencia entre los rangos promedio $\bar{r}_j$ de cada par de grupos. Si esta diferencia es mayor o igual a una diferencia crítica (DC), se puede afirmar que estos dos clasificadores son significativamente diferentes entre sí. La DC se calcula como:

$$DC = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{5.6}$$

donde q_α es un parámetro que se puede obtener de una tabla de valores críticos de la distribución *Studentized Range*[3] y N es el número de datos en total [17].

Así, utilizaremos estos métodos estadísticos para analizar nuestros resultados y determinar dos aspectos fundamentales:

[2] https://estdg.blogs.upv.es/files/2018/04/Tabla-Chi2_cola-derecha.pdf
[3] https://real-statistics.com/statistics-tables/studentized-range-q-table/

1. Qué agregación de las tres propuestas obtiene mejores resultados globales (ver Ecuación 4.1)
2. Para cada métrica, SA y F1-Macro, y cada una de ellas evaluada sobre CTG y CTL_k, determinar si es mejor usar nuestro método federado o simplemente usar modelos independientes sin ningún tipo de entorno federado.

5.2. Análisis de resultados

En primer lugar se van a presentar una serie de tablas resumen de los resultados obtenidos, a los que luego se les aplicará los test estadísticos anteriormente expuestos. Para mayor claridad y concisión, estas tablas muestran las medias de las ejecuciones realizadas para cada conjunto de datos. En cualquier caso, las tablas completas con todos los resultados se pueden encontrar en el repositorio de GitHub anteriormente enlazado.

5.2.1. Determinación de la mejor agregación

En las tablas 5.5-5.8 se muestran los resultados de nuestros experimentos sobre los diez conjuntos de datos, donde cada fila representa la media de los resultados obtenidos a partir de las ejecuciones del algoritmo con las diferentes semillas. Por ejemplo, si miramos la Tabla 5.4, Corel5k tendría 30 filas, una por semilla. El resultado expuesto en estas tablas es la media de esos resultados, que a su vez son la media de los resultados de cada cliente. La tabla compara los resultados de las tres agregaciones. Cada tabla corresponde a una métrica y un conjunto de test utilizado.

F1-MACRO TEST GLOBAL

Dataset	Agg 0	Agg 1	Agg 2
Corel5k	**0.0163**	0.0123	0.0146
20NG	**0.7768**	0.7530	0.7398
yeast	0.4355	**0.4377**	0.4362
reutersK500	0.0441	0.0295	**0.0594**
mediamill	0.1063	0.1038	**0.1081**
YahooEducation	0.3061	0.2934	**0.3116**
slashdot	0.2631	**0.2850**	0.2007
HumanPseAAC	0.1930	**0.2042**	0.2041
scene	0.7520	0.7504	**0.7524**
tmc2007_500	**0.6476**	0.6125	0.6213

Tabla 5.5: Valores obtenidos usando la métrica F1-MACRO y el conjunto de test CTG.

F1-MACRO TEST LOCAL

Dataset	Agg 0	Agg 1	Agg 2
Corel5k	**0.0133**	0.0096	0.0119
20NG	**0.7004**	0.6906	0.6675
yeast	0.4155	**0.4199**	0.4122
reutersK500	**0.0292**	0.0229	0.0270
mediamill	0.1054	0.1055	**0.1074**
YahooEducation	0.2325	0.2371	**0.2515**
slashdot	0.1752	**0.1924**	0.1153
HumanPseAAC	0.1922	0.2008	**0.2012**
scene	0.7845	0.7869	**0.7901**
tmc2007_500	**0.6224**	0.6060	0.6054

Tabla 5.6: Valores obtenidos usando la métrica F1-MACRO y el conjunto de test CTL.

SA TEST GLOBAL

Dataset	Agg 0	Agg 1	Agg 2
Corel5k	0.2170	0.2132	**0.2208**
20NG	**0.7913**	0.7747	0.7655
yeast	**0.5839**	0.5760	0.5785
reutersK500	0.2450	0.2421	**0.3292**
mediamill	0.6048	0.5976	**0.6059**
YahooEducation	**0.6653**	0.6599	0.6647
slashdot	0.3798	**0.4088**	0.3749
HumanPseAAC	0.6012	**0.6014**	0.5884
scene	0.7454	**0.7486**	0.7467
tmc2007_500	**0.7546**	0.7353	0.7416

Tabla 5.7: Valores obtenidos usando la métrica SA y el conjunto de test CTG.

SA TEST LOCAL

Dataset	Agg 0	Agg 1	Agg 2
Corel5k	**0.2167**	0.2063	0.2018
20NG	**0.7156**	0.7066	0.6835
yeast	**0.5889**	0.5823	0.5824
reutersK500	0.2801	0.2793	**0.3273**
mediamill	0.5488	0.5409	**0.5588**
YahooEducation	0.5968	0.5872	**0.5986**
slashdot	0.3979	**0.4248**	0.4037
HumanPseAAC	0.5762	0.5762	**0.5763**
scene	**0.7934**	0.7926	0.7928
tmc2007_500	**0.7535**	0.7416	0.7432

Tabla 5.8: Valores obtenidos usando la métrica SA y el conjunto de test CTL.

A simple vista no se aprecian demasiadas diferencias en los resultados presentados. Se ve como los mejores resultados para cada conjunto de datos se reparten de una manera aproximadamente equitativa entre las distintas

agregaciones. No parece haber una agregación que se destaque significativamente sobre las demás en todos los casos. Por lo tanto, sacar conclusiones o elegir la mejor agregación únicamente a partir de estas tablas sería precipitado.

Para tener una visión más clara y rigurosa, se va a ejecutar el test de Friedman. Cabe decir que, para una mayor precisión, el test de Friedman se ejecutará, para cada métrica y conjunto de test, sobre todos los resultados de todas las semillas, y no solamente sobre las medias que aparecen en las tablas.

RESULTADOS TEST DE FRIEDMAN (AGREGACIONES)

Métrica y conjunto de test	p-value	Diferencia entre agregaciones
F1-MACRO TEST GLOBAL	0.0047	✓
F1-MACRO TEST LOCAL	0.7112	×
SA TEST GLOBAL	0.0082	✓
SA TEST LOCAL	0.1998	×

Tabla 5.9: Resultados del Test de Friedman ejecutado sobre las dos métricas, cada una evaluada sobre CTG y CTL. Si p-value es menor que α=0.05 se rechaza la hipótesis y por tanto existe diferencia entre las agregaciones.

Analizando los resultados de la Tabla 5.9, se puede decir que si usamos la métrica F1-Macro evaluada sobre CTG, o si usamos la métrica SA sobre CTG, podemos determinar con un 95 % de confianza que hay diferencias entre las agregaciones, o dicho de otra forma, hay diferencias entre usar una u otra agregación. En los otros dos casos, vemos que no se rechaza la hipótesis nula, por tanto es indiferente la agregación que se use.

Como se dijo anteriormente, en el caso en que haya diferencia, no podemos establecer entre qué agregaciones son estas diferencias y mucho menos determinar una agregación como la mejor. Para ello, ejecutamos el test de Nemenyi sobre F1-Macro y CTG (Figura 5.2) y sobre SA y CTG (Figura 5.3).

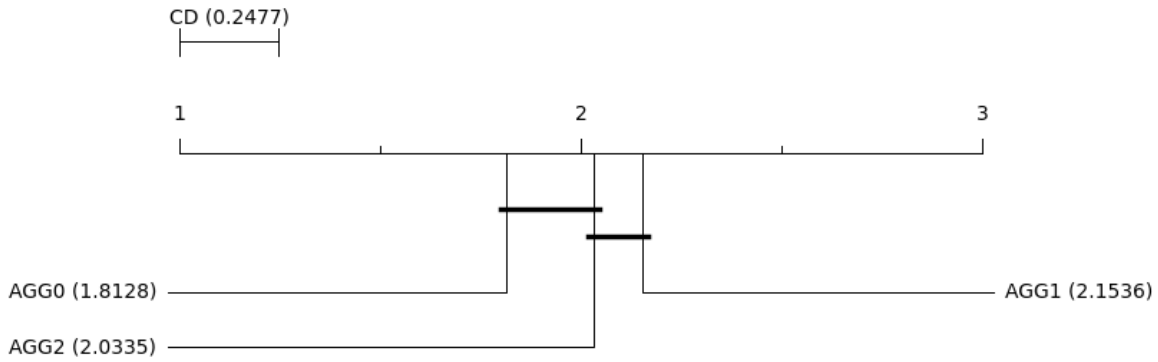


Figura 5.2: Resultados del test de Nemenyi sobre F1-Macro y CTG.

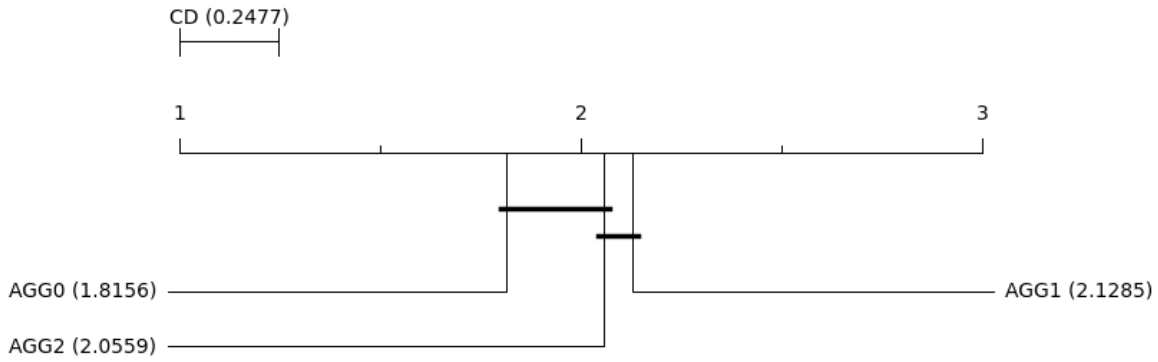


Figura 5.3: Resultados del test de Nemenyi sobre SA y CTG.

En la Figura 5.2 se puede observar que hay diferencias significativas entre la agregación 0 y 1, siendo mejor la 0. No obstante, entre la 0 y la 2 no hay grandes diferencias, por lo que no se puede determinar cuál sería mejor. Lo mismo ocurre en la Figura 5.3.

Por tanto, lo único que podemos concluir de estos test es que la agregación 1 sería la peor, mientras que elegir la 0 o la 2 sería indiferente. Aún así, para el siguiente apartado se debe elegir una para realizar la comparación entre el modelo federado y no federado. Como no habría diferencia entre la 0 y la 2, simplemente escogemos una cualquiera, por ejemplo la agregación 2.

5.2.2. Comparación modelo federado y no federado

Por último, realizaremos una comparación entre los resultados de los modelos utilizando aprendizaje federado y los resultados de los modelos sin utilizarlo. Como se mencionó anteriormente, los resultados seleccionados para la parte de los modelos federados corresponden a los obtenidos mediante la agregación 2.

Cabe mencionar que los resultados presentados a continuación son nuevamente tablas resumen. Estos no son los resultados completos, sino un promedio de ellos, al igual que en el apartado anterior. Los resultados completos pueden consultarse en el repositorio de GitHub.

Los resultados se pueden encontrar en las Tablas 5.10-5.13.

F1-MACRO TEST GLOBAL		
Dataset	FED (Agg 2)	NO FED
Corel5k	0.0146	**0.0546**
20NG	0.7398	**0.7567**
yeast	0.4362	**0.4771**
reutersK500	0.0594	**0.1113**
mediamill	0.1081	**0.1502**
YahooEducation	**0.3116**	0.2796
slashdot	0.2007	**0.3657**
HumanPseAAC	0.2041	**0.3695**
scene	**0.7524**	0.7389
tmc2007_500	0.6213	**0.6614**

Tabla 5.10: Valores obtenidos usando la métrica F1-MACRO y el conjunto de test CTG.

F1-MACRO TEST LOCAL		
Dataset	FED (Agg 2)	NO FED
Corel5k	0.0119	**0.0214**
20NG	0.6675	**0.7080**
yeast	0.4122	**0.4701**
reutersK500	0.0270	**0.0308**
mediamill	0.1074	**0.1471**
YahooEducation	**0.2515**	0.2512
slashdot	0.1153	**0.2583**
HumanPseAAC	0.2012	**0.3365**
scene	**0.7901**	0.7515
tmc2007_500	0.6054	**0.6607**

Tabla 5.11: Valores obtenidos usando la métrica F1-MACRO y el conjunto de test CTL.

SA TEST GLOBAL		
Dataset	FED (Agg 2)	NO FED
Corel5k	**0.2208**	0.1683
20NG	0.7655	**0.7753**
yeast	**0.5785**	0.5542
reutersK500	**0.3292**	0.3176
mediamill	0.6059	**0.6102**
YahooEducation	**0.6647**	0.6101
slashdot	0.3749	**0.5263**
HumanPseAAC	0.5884	**0.6248**
scene	**0.7467**	0.7377
tmc2007_500	0.7416	**0.7444**

Tabla 5.12: Valores obtenidos usando la métrica SA y el conjunto de test CTG.

SA TEST LOCAL		
Dataset	FED (Agg 2)	NO FED
Corel5k	**0.2018**	0.1984
20NG	0.6835	**0.7360**
yeast	**0.5824**	0.5533
reutersK500	**0.3273**	0.2892
mediamill	**0.5974**	0.5839
YahooEducation	0.6606	**0.6619**
slashdot	0.4037	**0.5746**
HumanPseAAC	0.5672	**0.6490**
scene	**0.7928**	0.7703
tmc2007_500	0.7432	**0.7532**

Tabla 5.13: Valores obtenidos usando la métrica SA y el conjunto de test CTL.

Observando los resultados, parece claro que el modelo no federado obtiene mejores resultados usando la métrica F1-Macro evaluado tanto sobre el CTG como en el CTL. En ambas tablas (5.10 y 5.11) vemos como el modelo no federado

obtiene mejor rendimiento en 8 de los 10 conjuntos de datos, por lo tanto sugiere que nuestra propuesta es peor en este caso de evaluación. Aún así, se podrá confirmar esta hipótesis utilizando el test de Friedman.

Por otro lado, los resultados parecen más parejos usando la métrica SA (Tablas 5.12 y 5.13). Es decir, el modelo federado se comporta mejor al predecir conjuntos de etiquetas enteros. Vemos como en cada tabla hay cinco conjuntos de datos que obtienen mejor rendimiento con el federado, y cinco que lo obtienen con el no federado. Con estos resultados, es difícil sacar alguna conclusión. Por ello, se ejecutará de nuevo el test de Friedman para las dos métricas y los dos conjuntos de test, y como en el apartado anterior, también se hará sobre los resultados completos, no sobre las medias (Tabla 5.14).

RESULTADOS TEST DE FRIEDMAN (FED vs NO FED)

Métrica y conjunto de test	p-value	Diferencia entre modelos
F1-MACRO TEST GLOBAL	$7.654 \cdot 10^{-11}$	✓
F1-MACRO TEST LOCAL	$3.654 \cdot 10^{-11}$	✓
SA TEST GLOBAL	0.0205	✓
SA TEST LOCAL	0.1785	×

Tabla 5.14: Resultados del Test de Friedman ejecutado sobre las dos métricas, cada una evaluada sobre CTG y CTL. Si p-value es menor que α=0.05 se rechaza la hipótesis y por tanto existe diferencia entre los algoritmos.

En la Tabla 5.14, se ve como en tres de los cuatro casos podemos afirmar con un 95 % de confianza que hay diferencia entre usar aprendizaje federado o no. El único caso donde sería indiferente sería utilizando SA evaluada sobre el CTG.

Cabe decir que, en este apartado, al ser solo dos grupos los que estoy evaluando, evidentemente se sabría entre qué grupos hay diferencias, pero solo con esto no es posible afirmar qué modelo es el mejor de los dos. Por esto, se recurre de nuevo al test de Nemenyi y se ejecuta sobre los tres casos con donde existen diferencias entre usar federado o no.

Como se puede observar en las Figuras 5.4-5.6, los dos algoritmos no están unidos por ninguna línea, con lo cual se puede afirmar que existe una diferencia significativa entre ellos. Además, se puede notar que los resultados son variados: en dos ocasiones es más favorable no usar nuestro modelo, mientras que en una ocasión sí lo es. En las figuras 5.4 y 5.5 se ve como no usar federado sería mejor que usarlo, mientras que en la figura 5.6 nuestra propuesta muestra un mejor rendimiento. Es decir, nuestro modelo es más efectivo al predecir todas las etiquetas asociadas a las instancias (mejor rendimiento con SA), pero no es tan competitivo como el modelo no federado en términos de F1-Macro para predecir etiquetas individuales.

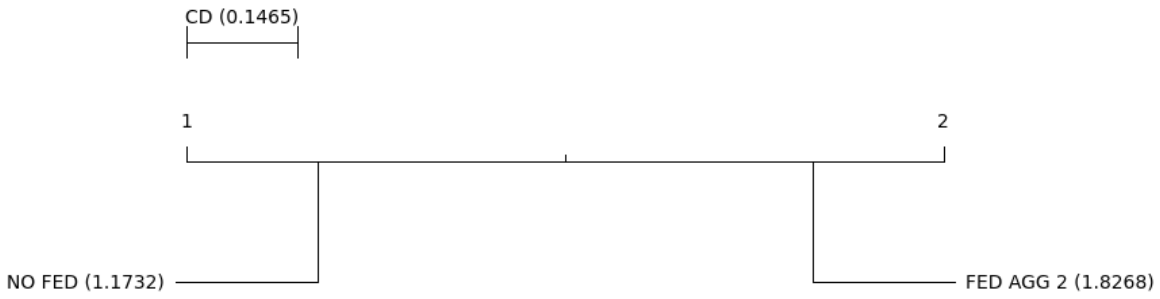


Figura 5.4: Resultados del test de Nemenyi sobre F1-Macro y CTG.

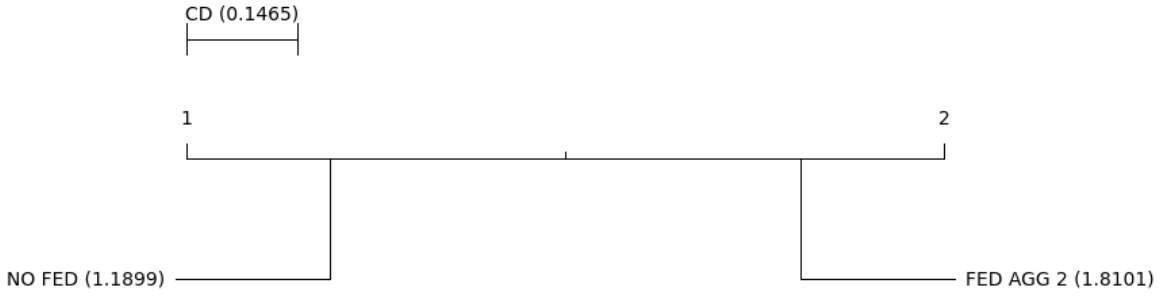


Figura 5.5: Resultados del test de Nemenyi sobre F1-Macro y CTL.

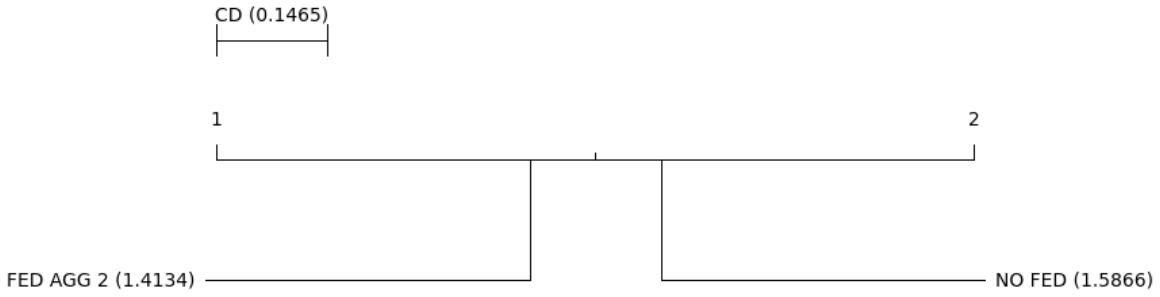


Figura 5.6: Resultados del test de Nemenyi sobre SA y CTG.

6. Conclusiones y trabajo futuro

Se presentan ahora las conclusiones más relevantes extraídas de la realización de este trabajo, así como las posibles direcciones para futuras investigaciones en el campo del aprendizaje federado multi-etiqueta.

6.1. Conclusiones

En primer lugar, se ha planteado el marco en el cual se desarrolla este trabajo, el aprendizaje federado multi-etiqueta. Así, se han presentado las dos principales vertientes del aprendizaje federado, horizontal y vertical, y se ha proporcionado una introducción formal al concepto del aprendizaje multi-etiqueta.

Seguidamente, se han presentado las técnicas existentes en el estado del arte que pudieran estar relacionadas con nuestro trabajo. Así, nos sirvió de ayuda [13], que aunque se base en VFL (nuestra propuesta está basada en HFL), tiene similitudes con nuestra propuesta, como la distinción entre capas de una red o la posibilidad de tener múltiples poseedores de etiquetas, caso no contemplado en el aprendizaje federado horizontal clásico. Además, se ha revisado FedAvg [7], un enfoque desarrollado para problemas multiclase y binarios, el cual ha sido extendido por nuestra propuesta al ámbito del aprendizaje multi-etiqueta.

Seguidamente, presentamos nuestra propuesta, MLFedAvg, que consideramos innovadora, ya que aborda por primera vez problemas multi-etiqueta en un entorno federado. Además, contempla el caso en el que todos los clientes tienen etiquetas diferentes, con lo cual objetivos distintos, un escenario que, hasta donde llega nuestro conocimiento, no ha sido tratado anteriormente.

Una vez desarrollada teóricamente nuestra propuesta, la ponemos a prueba realizando diversos experimentos, los cuales en total han supuesto más de 500 entrenamientos (Tabla 5.4), desafiando así los límites computacionales de los que se disponía, ya que un entrenamiento de este tipo solía demorarse más de una hora, incluso en algunos casos dos o tres.

Al obtener los resultados y analizarlos, determinamos que las agregaciones que mejores resultados ofrecen son la 2 y la 0 (ver Ecuación 4.1). Estas agregaciones, aportaciones propias de este trabajo, superan el rendimiento obtenido mediante el

uso de la agregación 1. Esto es especialmente destacable, pues la agregación 1 es la implementada en FedAvg [7] y es ampliamente superada por nuestras propuestas de agregación.

Para finalizar, al comparar nuestra propuesta con modelos no federados, encontramos que, al realizar la evaluación de rendimientos con la métrica F1-Macro, nuestra propuesta es superada en dos casos de estudio. Sin embargo, sí muestra un mejor rendimiento en la métrica *Subset Accuracy*, lo que abre la puerta a futuras investigaciones y mejoras.

6.2. Trabajo futuro

En primer lugar, pensamos que es una propuesta bien fundamentada y con mucho potencial, con lo que sería interesante seguir trabajando para mejorarla y que valga la pena usar aprendizaje federado por encima de los modelos locales en este tipo de problemas multi-etiqueta donde todas las partes tienen etiquetas distintas.

Como propuestas para seguir investigando en en esta línea, pensamos que sería bueno utilizar un entorno federado real. Así, se podría usar un servidor real externo que realice la agregación y se podrían obtener resultados más cercanos a la realidad. Además, sería interesante contar con datos que tengan características federadas reales (por ejemplo, que provengan de distintas organizaciones o entidades del mismo sector). Incorporar estos elementos permitiría evitar una simulación completa desde cero y podría tener un impacto significativo en los resultados de futuras investigaciones.

Finalmente, otro enfoque interesante sería aplicar este mismo paradigma, o uno similar, a redes convolucionales para la clasificación de imágenes. Dado que los filtros convolucionales son efectivos para detectar bordes, formas, y otras características visuales, esta etapa de preprocesado podría ser entrenada de manera común entre todos los clientes, aunque luego cada uno tenga distintas etiquetas de clasificación. Esta metodología sería análoga a nuestra propuesta, en la que se comparte una parte de la red, que serviría para detectar bordes y formas, pero cada cliente tiene una parte no compartida, la cual realizaría la clasificación.

Bibliografía

[1] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council.

[2] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. *Federated Learning*. Morgan & Claypool Publishers, 2019.

[3] Jose M. Moyano. *Multi-Label Classification Models for Heterogeneous Data: an Ensemble-based Approach*. PhD thesis, Universidad de Córdoba (España) y Virginia Commonwealth University (EE.UU.), 2020.

[4] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[5] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2024.

[6] Henrik Hellström, José Mairton Barros da Silva Júnior, Viktoria Fodor, and Carlo Fischione. Wireless for machine learning. 09 2020.

[7] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[8] Jose M. Moyano. Aprendizaje federado: conceptos, estado del arte, y retos en modelos no-deep learning. https://www.cs.us.es/investigacion/seminario-de-inteligencia-artificial, 2023. Seminario de Inteligencia Artificial de la Universidad de Sevilla. Último acceso: 20/03/2024.

[9] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[10] Eva Gibaja and Sebastián Ventura. A tutorial on multilabel learning. *ACM Comput. Surv.*, 47(3), apr 2015.

[11] Jose M. Moyano, Eva L. Gibaja, Krzysztof J. Cios, and Sebastián Ventura. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Information Fusion*, 44:33–45, 2018.

[12] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5177–5186, 2019.

[13] Vaikkunth Mugunthan, Pawan Goyal, and Lalana Kagal. Multi-vfl: A vertical federated learning system for multiple data and label owners. 2021.

[14] Piotr Szymanski and Tomasz Kajdanowicz. A scikit-based python environment for performing multi-label classification. *CoRR*, abs/1702.01460, 2017.

[15] Jose M. Moyano. Multi-label classification datasets compilation. https://www.uco.es/kdis/mllresources/. Último acceso: 2024-07-01.

[16] Milton Friedman. A Comparison of Alternative Tests of Significance for the Problem of *m* Rankings. *The Annals of Mathematical Statistics*, 11(1):86 – 92, 1940.

[17] KDIS Lab. Statds. https://github.com/kdis-lab/StaTDS/tree/main, 2023. Versión principal.