Componentes de software y documentación del caso de uso de infraestructuras cloud

MLEDGE - Aprendizaje automático en la nube y en el borde

(Cloud and Edge Machine Learning)

Diciembre de 2024

Información sobre el entregable

Nombre del documento:

M2 - Componentes de software y documentación del caso de uso de infraestructuras cloud

Versión actual: 1.0

Proyecto: MLEDGE - Aprendizaje automático en la nube y en el borde (Cloud and Edge Machine Learning)

Paquetes de trabajo:
- P2: Implementación de componentes básicos de MLEDGE
- P5 - Implementación del caso de uso de infraestructuras cloud

Tareas: El entregable es resultado del trabajo en los diversos componentes técnicos:
- A2.1: Uso eficiente de Federated Learning (FL) en nubes híbridas y protección contra ataques (incluidos ataques de envenenamiento e inferencia).
- A2.2: Protección de datos sensibles o confidenciales que sean intercambiados entre diferentes dominios administrativos en la nube, incluido su borde que es potencialmente menos seguro.
- A2.3: Equidad en términos de distribución de costos y ganancias cuando la computación en el borde se usa para entrenar de forma colaborativa modelos de aprendizaje automático.
- A2.4: Gestión de los desafíos de portabilidad de datos en el borde de la nube.
- A2.5: DevOps y desarrollo continuo para servicios de aprendizaje automático que se ejecutan en el borde de la nube (FLaaS).
- A5.2: Implementación y pruebas de componentes de infraestructuras cloud.

Entregable: E3.2 - Componentes software, informe y documentación preliminares.

Autores: Iker Ceballos Rodríguez (Acuratio), David Márquez Mínguez (Acuratio)

Revisores: Nikolaos Laoutaris (IMDEA) y Javad Dogani (IMDEA)

Historial de Versiones

Versión	Fecha	Resumen de modificaciones
Versión 1.0	31-12-2024	Versión final del documento

1. Introducción

En diciembre de 2022, el Ministerio de Asuntos Económicos y Transformación Digital del Gobierno de España adjudicó a IMDEA Networks el proyecto titulado "*MLEDGE - Aprendizaje automático en la nube y en el borde (Cloud and Edge Machine Learning)*" (REGAGE22e00052829516, en adelante el 'Proyecto' o MLEDGE). Esta iniciativa cuenta con financiación de la Unión Europea a través del Plan de Recuperación, Transformación y Resiliencia (European Union - NextGenerationEU/PRTR).

El objetivo principal del Proyecto es establecer un ecosistema robusto de servicios de aprendizaje federado (FL, por sus siglas en inglés) en el borde, que sean seguros y eficientes. Estos servicios están diseñados para facilitar el uso de datos personales y confidenciales, tanto de tipo B2B como de consumidores, en el entrenamiento de modelos de aprendizaje automático (ML), garantizando en todo momento la privacidad de los datos y de sus propietarios.

Los **objetivos generales del proyecto** se pueden resumir en los siguientes:

1. Facilitar la accesibilidad del aprendizaje federado en el borde mediante el desarrollo de una capa de software intermedio y componentes que simplifiquen la complejidad inherente al procesamiento y al intercambio de datos.
2. Abordar los desafíos técnicos asociados al aprendizaje federado en entornos de nube y borde, optimizando su implementación y rendimiento.
3. Validar la funcionalidad desarrollada a través de casos de uso representativos de problemas reales de la industria, demostrando el impacto práctico de estas tecnologías.
4. Difundir y explotar los resultados del Proyecto, involucrando a agentes externos clave y comunicando los hallazgos a un público más amplio y potencialmente interesado.

Uno de los objetivos fundamentales del proyecto consiste en diseñar, implementar y poner a disposición del público demostradores que trabajen con datos sensibles, tanto de carácter industrial como personal. Estos demostradores están orientados a alimentar modelos de aprendizaje automático aplicables a diversos sectores de la industria. En la etapa inicial del proyecto, se seleccionaron empresas clave para el desarrollo de la plataforma FLaaS (*Federated Learning as a Service*), el monitoreo de costes computacionales, y el diseño e implementación de casos de uso reales que se beneficien del aprendizaje distribuido en el borde de la nube.

La adjudicación del paquete de trabajo P5, enfocado en el diseño e implementación del caso de uso relacionado con las infraestructuras cloud, recayó en ACURATIO EUROPE SL con NIF B75217612.

El presente documento corresponde al entregable E5.2, titulado "*Componentes de software y documentación del caso de uso de infraestructuras cloud*". Su propósito principal es detallar el desarrollo e implementación de los elementos definidos en el entregable E5.1, describiendo el proceso seguido y presentando la solución preliminar alcanzada.

2. Definición del problema y objetivos

En este apartado se presenta un resumen de la definición del problema y los objetivos específicos del proyecto, tal como se detallaron en el entregable E5.1. Asimismo, se describen los casos de uso relevantes, con el objetivo de proporcionar un marco de referencia que facilite la comprensión de las tareas realizadas en este entregable y su integración en el contexto general del proyecto.

2.1. Objetivos

El objetivo global de este paquete de trabajo es el desarrollo y despliegue de una plataforma de FLaaS que facilite el diseño y ejecución de modelos federados incorporando novedosas técnicas de privacidad.

Idealmente esta plataforma de FLaaS permitirá tanto la simulación de casos de uso federados como la implementación de modelos reales en entornos de producción para los casos de uso de economía tradicional y digital. Los objetivos específicos se detallan a continuación:

1. Desplegar una capa de software distribuido FLaaS que permita a todos los adjudicatarios de MLEDGE implementar sus casos de uso.
2. Proporcionar la infraestructura para el despliegue de los entornos federados de los demostradores para MLEDGE.
3. Dar soporte técnico para el entrenamiento y explotación de los modelos federados para los casos de uso de MLEDGE.
4. Colaborar con la Fundación en la prueba, integración y demostración de módulos de FL seguro
    - Securing Federated Systems Against Poisoning Attacks
    - Leveraging Quadratic Voting in Federated Learning.

2.2. La solución

De cara a entrenar modelos en dispositivos en el extremo y poder procesar los datos en el propio dispositivo surge FL que permite entrenar modelos de forma distribuida sin mover los datos. Son los modelos los que se envían a dónde están los datos, lo que reporta numerosos beneficios como veremos a continuación:

1. Minimización del acceso al dato, los datos permanecen bajo el control del propietario.
2. En combinación con otras tecnologías habilitadoras de la privacidad, como Differential Privacy, protocolos de agregación segura o técnicas de k-anonimidad cumple las más estrictas garantías de privacidad y utilidad en sistemas productivos. Por ejemplo, Google tiene desplegados modelos predictivos en los billones de dispositivos móviles con sistema operativo Android.
3. Es una herramienta para entrenar modelos en el extremo (Training on the Edge), lo que permitirá, por ejemplo, procesar los terabytes de datos que producen los coches autónomos.
4. Es una herramienta colaborativa que permite, por ejemplo, a hospitales entrenar modelos sobre imágenes médicas, todo ello preservando la privacidad de los datos de los pacientes.

De cara a facilitar la adopción del FL de un creciente número de aplicaciones que usan modelos de Machine Learning en los dispositivos en el extremo de la red. MLEDGE busca el desarrollo de técnicas, librerías y componentes que permitan iniciar estos servicios más fácil y ágilmente.

Acuratio ha construido una plataforma de aprendizaje federado y dará soporte a los distintos casos de uso de MLEDGE. Para formar la infraestructura básica sobre la que se desarrollarán los casos de uso de Aprendizaje Federado para todos los paquetes de trabajo, necesitaremos cinco componentes principales:

1. **Capa de Ejecución:** En esta capa se ejecutarán los modelos sobre datos locales pertenecientes a cada participante.
2. **Capa de Comunicaciones:** Para poder conectar todos los elementos que componen una plataforma de aprendizaje federado, las comunicaciones son un elemento crítico.
3. **Capa de Coordinación:** Aunque el objetivo es generar una infraestructura lo más distribuida posible, es necesario un elemento común que permita coordinar a todos los participantes.
4. **Capa de Gobernanza:** La capa de gobernanza se implementará como una API accesible a todos los participantes.

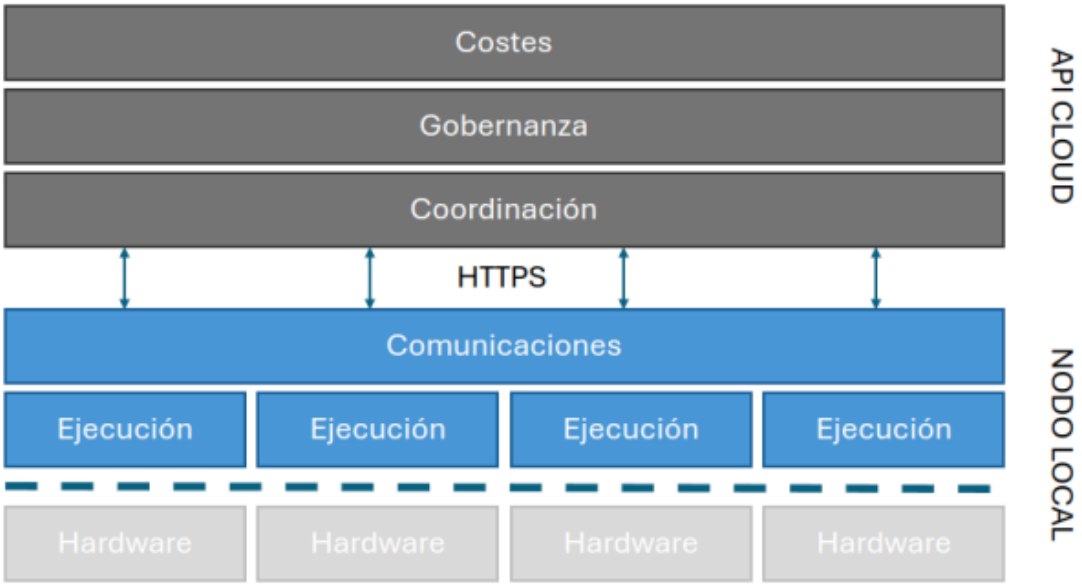
Adicionalmente, se ha implementado un caso de uso para la optimización de costes en el despliegue de proyectos de aprendizaje federado en infraestructuras cloud. Dicha implementación ha requerido añadir a la solución una Capa de Costes, una abstracción de la integración con las herramientas de facturación de la nube que permite obtener la información necesaria sobre la disponibilidad y el coste de los recursos necesarios para el aprendizaje federado en diferentes plataformas.

Para alcanzar los objetivos seguiremos tres fases bien diferenciadas:

1. **Integración con APIs de facturación:** Se ha integrado la plataforma con las APIs de los principales proveedores cloud (GCP, AWS, Azure) para generar un dashboard centralizado en el que evaluar los costes de un proyecto federado.
2. **Identificación de recursos comunes:** Se han identificado el conjunto de recursos de computación más habitual en un proyecto federado.
3. **Generación de plantillas:** Se han generado plantillas que permitan lanzar proyectos federados en distintas nubes, de manera que la herramienta pueda servir para seleccionar la distribución optima de cargas para acceder a los recursos cloud con el proveedor más asequible.

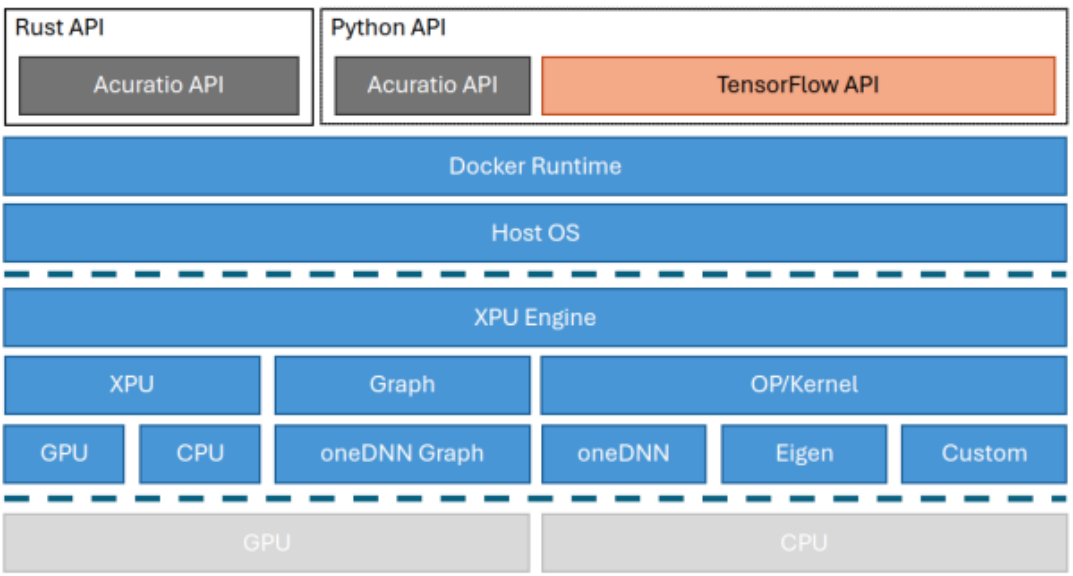
3. Plataforma FlaaS

En este apartado se presentan las distintas capas que forman parte de la plataforma FlaaS y se describe su implementación. En él se detallan los requisitos planteados en el entregable anterior y su pertinente justificación.



3.1. Implementación de capa de ejecución

En esta capa se llevan a cabo las ejecuciones de los modelos sobre los datos locales de cada participante. Constituye el componente de nivel más bajo, diseñado para instalarse y operar lo más próximo posible al lugar donde se almacenan los datos. En los demostradores de este proyecto, esta capa se despliega en servidores en la nube; sin embargo, debe ser portable y de fácil implementación, ya que en escenarios de uso en producción podría requerirse su despliegue en dispositivos edge.



Se ha seguido una filosofía de diseño modular basada en contenedores ya que ofrece la mayor flexibilidad a la hora de desplegar en entornos diversos. Las herramientas se han

desarrollado empleando principalmente dos lenguajes de programación: python y rust. Las interfaces que se han diseñado para ser empleadas directamente por el usuario se han programado en python por su flexibilidad y facilidad de uso, además esto nos ha permitido construir encima de herramientas como tensorflow muy extendidas en el mundo del aprendizaje automático. En cambio, los componentes más automatizados han sido construidos con rust por su eficiencia y seguridad.

Actualmente, se cuenta con una plataforma capaz de entrenar modelos de inteligencia artificial de forma local en servidores desplegados en la nube o localmente en hardware situado lo más cerca posible de las fuentes de datos. A continuación, se detallan los requisitos de esta capa y la justificación de su cumplimiento:

#	Requisito	Descripción	Tipo
RF1	Carga de datos	El sistema debe permitir la carga de datos desde múltiples fuentes diferentes, tales como sistemas de almacenamiento en la nube o a través de una interfaz con cifrado en el navegador	Obligatorio

La arquitectura desarrollada permite montar un directorio local del servidor como volumen en los contenedores de manera que cualquier archivo disponible en ese directorio sea automáticamente accesible para el software.

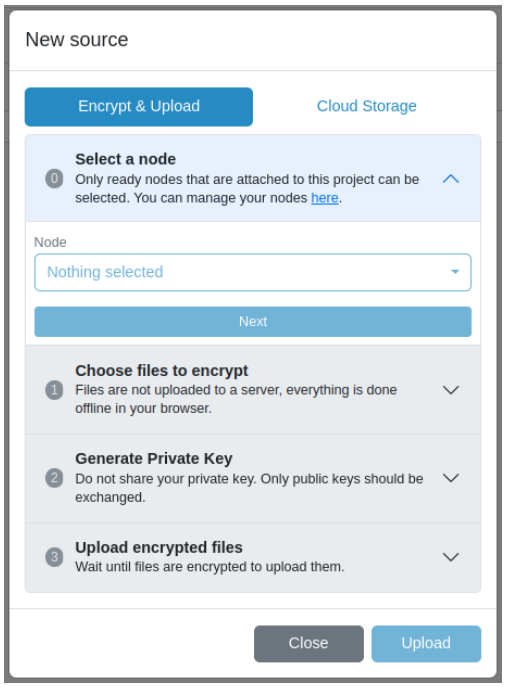


Ilustración 1: Interfaz para la carga segura de datos.

Finalmente se ha desarrollado una interfaz de carga de archivos que permite cifrarlos en el navegador con claves proporcionadas por el cliente y subirlos directamente al servidor protegidos de extremo a extremo. El diseño de esta interfaz se aprecia en la siguiente figura.

Adicionalmente se han desarrollado integraciones con las APIs de AWS S3 y GCP Storage para poder consumir, con las credenciales adecuadas, cualquier fichero almacenado en estos servicios.

RF2	Procesamiento de datos	El sistema debe permitir el tratamiento de datos con herramientas tradicionales como: pandas, numpy, sklearn...	Opcional

Se han incluido las librerías más comunes para el tratamiento de datos en el software incorporado por defecto en los nodos, además es posible personalizar estas instalaciones incluyendo librerías adicionales. Los módulos incluidos por defecto son los siguientes:

- eli5
- matplotlib
- numpy
- pandas
- pyarrow
- scikit-learn
- seaborn
- shap
- tensorflow

RF3	Visualización de datos	El sistema debe permitir la visualización de datos mediante una interfaz basada en *jupyter notebook* o similar.	Obligatorio

Se ha seleccionado JupyterLab por sus ventajas sobre jupyter notebook, las principales: la interfaz más flexible y avanzada que proporciona, la arquitectura modular que permite extender sus funcionalidades, la facilidad de uso y la comodidad para la visualización de datos.

RF4	Desarrollo de modelos	El sistema debe permitir entrenar modelos para desarrollar los distintos casos de uso.	Obligatorio

Se han realizado implementaciones de técnicas de aprendizaje federado horizontal y vertical incluyendo módulos de Split Neural Networks, XGBoost, Federated Multi-Task Learning y Federated Averaging. En la ilustración 2 se pueden observar los distintos tipos de modelos soportados actualmente por la plataforma.

Basándonos en nuestra experiencia hemos elegido estos algoritmos ya que son los más extendidos en la industria. Especialmente XGBoost es el algoritmo de referencia para datos tabulares mientras que el resto de los casos de uso suelen abordarse mediante redes neuronales. Esta implementación basada en Split Learning proviene de nuestra propia investigación con el MIT MediaLab. Finalmente, para casos de uso más sencillos se ha incorporado el soporte para Support Vector Machines con FMTL. De este modo la plataforma cuenta con todas las capacidades necesarias para que un científico de datos pueda desarrollar modelos federados en cualquier ámbito.

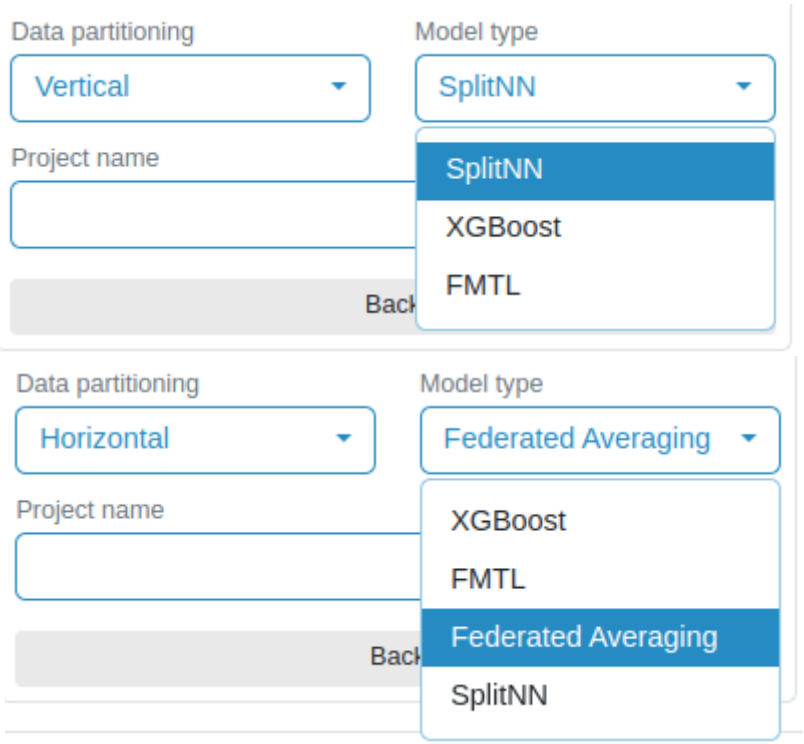


Ilustración 2: Tipos de modelos soportados por la plataforma.

RF5	Procesamiento seguro	El sistema debe permitir implementar técnicas de procesamiento seguro con sistemas de cifrado y diversas Privacy Enhancing Technologies.	Obligatorio

Todas las técnicas mencionadas en el criterio anterior se engloban dentro de las PETs, pero adicionalmente se han incorporado mecanismos de Privacidad Diferencial o agregación segura, entre otros, para mejorar las capacidades de privacidad de la plataforma.

RF6	Portabilidad	La infraestructura sobre la que se ejecutan las cargas de trabajo debe ser intercambiable, para ello el despliegue de la capa de ejecución se servirá de tecnologías open source de despliegue con contenedores (Docker).	Obligatorio

Docker utiliza contenedores para empaquetar aplicaciones junto con todas sus dependencias (bibliotecas, herramientas de sistema, configuraciones, etc.). Esto asegura que la aplicación se ejecute de manera idéntica sin importar dónde esté desplegada, ya sea en un entorno local, un servidor, o la nube.

De este modo la capa de ejecución es totalmente compatible con cualquier entorno, ya que Docker abstrae las diferencias entre sistemas operativos.

RF7	Versatilidad	El sistema debe permitir diferentes tipos de aprendizaje federado para servir a los casos de uso, incluyendo aprendizaje federado vertical y horizontal	Obligatorio

La justificación de cumplimiento de este criterio viene evidenciada en la respuesta al criterio RF4. En la ilustración 2 se pueden apreciar los distintos tipos de modelos soportados

actualmente por la plataforma. Además, el desarrollo modular de todos los componentes hace que la plataforma sea fácilmente extensible para soportar nuevos tipos de modelos federados.

Esta versatilidad se evidencia con el demostrador de implementación de mejoras de protocolos de Federated Learning, probando que es posible incluir mejoras en los protocolos implementados de forma sencilla.

RF8	Multicloud	El sistema debe ser una plataforma multicloud, que trabaje al menos con las principales plataformas (AWS, Azure, Google Cloud) y permitir su despliegue también en infraestructura *on-premise* del cliente o en el borde de la nube (cloud Edge).	Opcional

Docker asegura que las aplicaciones se desplieguen de manera predecible y consistente, independientemente del entorno (AWS, Azure, Google Cloud, on-premise). La integración con la capa de comunicaciones también asegura la conectividad independientemente del entorno en el que se despliegue el software.

RF9	Administración y gestión	La plataforma debe proporcionar herramientas administrativas y permisos especiales para la gestión de los modelos a ser entrenados.	Obligatorio

Se ha integrado una consola de administración que permite gestionar permisos a nivel de recurso para todos los usuarios de la plataforma. Para ello se ha diseñado un sistema de permisos basado en roles y se han creado permisos para todas las acciones de manera que se pueda controlar el acceso de forma granular.

Algunos ejemplos de acciones que tienen asociado un permiso específico son: arranque, parada y gestión de los nodos de computación, solicitudes de entrenamiento sobre la capa de ejecución o acceso y modificación de las fuentes de datos.

3.2. Implementación de capa de comunicación

La capa de comunicación es fundamental para integrar todos los componentes de una plataforma de aprendizaje federado, dado que las comunicaciones representan un aspecto crítico. Estas deben ser seguras y confiables, y, al mismo tiempo, permitir la conexión de nodos desplegados en redes muy heterogéneas. Algunos de estos nodos pueden enfrentar desafíos como firewalls complejos, conexiones a redes privadas o direcciones IP dinámicas. Por ello, disponer de una capa de comunicación que garantice una conexión robusta y fiable entre todos los dispositivos de la capa de ejecución resulta esencial.

Se realizó la selección del mejor conjunto de tecnologías, protocolos de comunicaciones y medidas de seguridad para garantizar un sistema de comunicaciones robusto y eficiente entre los elementos de la capa de ejecución. Los componentes encargados de entrenar los modelos deben comunicarse de forma segura para garantizar que, siguiendo los protocolos establecidos en los algoritmos de aprendizaje federado, los datos estén protegidos.

RF10	Transmisión segura	El sistema debe permitir la transmisión segura de datos con protocolos de cifrado como TLS.	Obligatorio

Para cumplir con este requisito se ha añadido a los nodos de computación la habilidad de conectarse directamente entre ellos, de manera que puedan entrenar modelos federados siguiendo protocolos predefinidos. Esta solución se ha basado en protocolos de código abierto, que permiten crear un conjunto de túneles cifrados extremadamente livianos entre dos puntos en una red. Estos túneles tienen que permitir la conexión directa entre todos los nodos en una red de malla, sin un concentrador que actúe de intermediario.

Para lograrlo se ha diseñado un protocolo capaz de establecer túneles de WireGuard entre dos clientes a través de NAT y/o firewalls corporativos. Esto permite establecer conexiones seguras punto a punto con una configuración mínima por parte de las empresas que quieran instalar los nodos. Además, se ha protegido todo el sistema con cifrado de extremo a extremo, asegurando que la clave privada nunca abandona el nodo. Esto es importante porque la clave privada es lo único que podría usarse para hacerse pasar por ese nodo. Como resultado, sólo ese nodo puede cifrar o descifrar paquetes dirigidos a él mismo.

RF11	Compatibilidad	La capa de comunicaciones debe permitir comunicarse con diferentes nubes.	Obligatorio

Al construir la solución sobre un protocolo robusto y de código abierto como WireGuard nos aseguramos de poder establecer comunicaciones en cualquier entorno, independientemente de la infraestructura de la red y las capas intermedias que puedan existir (NAT, Firewall, Transit Gateways, etc.).

También se ha diseñado la solución para requerir el menor número de configuraciones a nivel de firewall, y de manera que todas las conexiones sean siempre salientes para evitar tener que exponer puertos lo que podría dar lugar a instalaciones inseguras.

3.3. Implementación de capa de coordinación

Un elemento común es indispensable para coordinar a todos los participantes, y este rol lo desempeña la capa de coordinación. Esta consiste en una API desplegada en un dominio seguro y accesible para todos los componentes de la capa de ejecución. Su función principal es indicar a cada participante cuándo debe iniciar una computación y proporcionarle información sobre la identidad de los demás. En ningún caso esta capa gestionará datos en crudo ni los gradientes de los modelos, limitándose exclusivamente a la coordinación de los distintos elementos de la capa de ejecución. Estos elementos se conectarán directamente mediante una red de malla para intercambiar la información necesaria para entrenar los modelos.

Los nodos de computación siguen distintos protocolos de aprendizaje federado según el caso de uso y la técnica empleada, por lo que es esencial contar con un elemento común capaz de coordinar a todos los elementos en el borde de la nube para que el sistema funcione de forma conjunta.

Por lo tanto, se ha añadido a la plataforma el sistema de coordinación que permite a los nodos de computación entrenar algoritmos federados más complejos. Además, gracias a este nuevo sistema los protocolos federados serán parametrizables y personalizables.

RF12	Federación de modelos	El sistema debe permitir implementar protocolos federados para el entrenamiento de modelos preservando la privacidad del dato.	Obligatorio

Tal y como se ha descrito en la justificación de los requisitos RF4 y RF7 la plataforma soporta varios tipos de modelos horizontales y verticales. Además, su diseño modular permite implementar nuevos algoritmos federados de forma sencilla.

RF13	Analítica federada	El sistema debe permitir coordinar protocolos de analítica federada basada en Secure Multi-Party Computation y técnicas de cifrado homomórfico entre otros.	Obligatorio

Se han incorporado capacidades de analítica federada a la plataforma, incluyendo implementaciones de Private Set Intersection y Private Join and Compute basadas en Secure Multi-Party Computation.

Adicionalmente estas capacidades han sido incorporadas a la plataforma para poder ser utilizadas mediante interfaces no-code, lo que facilita su uso y disminuye las barreras de acceso para la adopción de estas tecnologías en nuevos casos de uso.

Finalmente, al basar los desarrollos en Python y librerías de código abierto como Numpy y Tensorflow, es trivial incorporar capacidades de cifrado homomórfico utilizando, por ejemplo, Concrete Numpy de Zama FHE.

RF14	Programación de tareas	El sistema debe permitir pre-autorizar y coordinar colas de experimentos para facilitar la experimentación y el desarrollo de nuevos sistemas federados.	Opcional

Se ha incorporado una interfaz de gestión de colas de tareas totalmente integrada con el sistema de permisos de manera que se puedan lanzar tareas sobre uno o más nodos, incluso si no se tiene acceso completo a todos los recursos.

Por ejemplo, en un proyecto federado yo podría entrenar un modelo con mis datos y los de otro participante sin necesidad de acceder al servidor de computación del otro participante. Haría esto generando un script y ejecutándolo de forma remota a través de la plataforma, gracias a la integración con el sistema de RBAC, el otro participante tendría control total para autorizar o denegar determinadas ejecuciones sobre sus datos.

RF15	Confidencialidad	La API de coordinación debe garantizar la seguridad y confidencialidad de la información sobre los participantes que almacene en su base de datos	Obligatorio

Nuevamente, gracias a la integración del sistema RBAC, se puede asegurar en todo momento que cada participante tiene control total sobre sus datos y quién tiene acceso a los mismos.

La gestión del acceso a los recursos de computación, que en definitiva son la puerta para acceder a los datos confidenciales de los participantes, está controlada también en todo momento a través de RBAC.

RF16	Aislamiento de recursos	La API de coordinación debe proporcionar la posibilidad de aislar los recursos de cada organización, de forma que una misma API sirva para múltiples proyectos federados.	Obligatorio

El diseño de la plataforma se ha basado en organizaciones, de manera que para cada proyecto se pueda lanzar una nueva organización totalmente aislada del resto. Esto permite coordinar a los equipos y controlar granularmente los permisos sin posibilidad de compartir ningún recurso con otros usuarios de la plataforma que no formen parte de la organización.

Esto es un elemento clave para la escalabilidad de la plataforma ya que permitirá soportar múltiples casos de uso simultáneamente.

3.4. Implementación de capa de gobernanza

Esta capa se implementa como una API accesible para todos los participantes y emplea un sistema de cifrado asimétrico para funcionar como proveedor de identidades, controlando quién puede acceder a cada recurso. A través de esta capa se gestionan las interacciones entre los distintos participantes. No obstante, el acceso a la capa de ejecución de cada entidad, donde residen los datos sensibles, está protegido adicionalmente en un nivel inferior, dentro de la capa de comunicaciones. De este modo, incluso si la capa de gobernanza se viera comprometida y alguien obtuviera acceso a la capa de ejecución de otro participante, no podría acceder a sus datos gracias a los controles implementados en la capa de comunicaciones.

RF17	Acceso al dato	El sistema debe implementar un control de accesos que permita controlar quién accede a la interfaz de visualización de datos.	Obligatorio

Se ha diseñado un sistema de autenticación y autorización que permite controlar el acceso de los desarrolladores a los distintos elementos de la plataforma de aprendizaje federado. Este sistema exige asociar una identidad única a cada usuario a través de su correo electrónico, estos usuarios serán los que tengan acceso a los distintos recursos de la plataforma y todos sus accesos quedarán registrados en el sistema de logs.

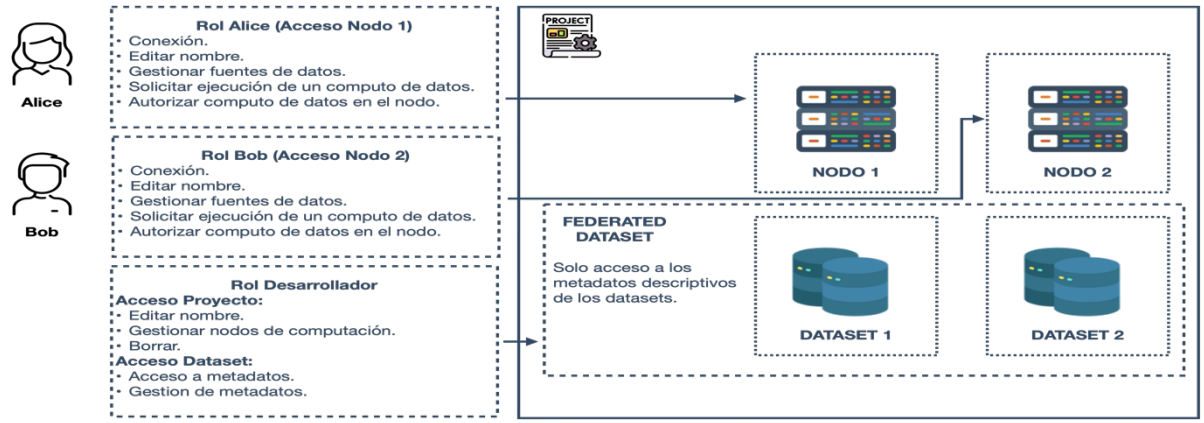


Ilustración 3: Ejemplo de configuración sistema RBAC.

En la figura anterior se muestra un ejemplo de configuración del control de accesos para un proyecto sencillo con dos usuarios, dos nodos y dos fuentes de datos. Esta arquitectura es modular y se puede extender para abarcar tantos recursos y usuarios como sea necesario.

RF18	Abstracción de datos	El sistema debe utilizar roles para agrupar los permisos relacionados.	Opcional

Para homogeneizar y unificar la gestión de los accesos, se ha diseñado un sistema de RBAC que permite agrupar los permisos en roles que pueden asignarse a uno o más usuarios. Por defecto todas las organizaciones tienen al menos 3 roles básicos con distintos niveles de acceso:

- Groups.FullAccess: Rol de administrador con acceso total a la organización.
- Groups.Developer: Rol de usuario que permite crear y gestionar recursos.
- Groups.Member: Rol invitado que permite visualizar recursos.

Adicionalmente los administradores podrán decidir crear y asociar nuevos roles en función de las necesidades de su organización.

RF19	Minimización de privilegios	A los usuarios se les deben asignar solo los roles que les dan el menor nivel de acceso necesario para desarrollar las funciones designadas.	Opcional
RF20	Separación de funciones	Ningún usuario debe tener control total sobre los elementos críticos de la capa de gobernanza.	Opcional

El sistema RBAC descrito anteriormente permite cumplir con estos dos criterios ya que proporciona flexibilidad absoluta al administrador de cada organización para aplicar estas metodologías de minimización de privilegios y separación de funciones.

3.5. Implementación de capa de costes

La capa de costes es una abstracción de la integración con las herramientas de facturación de la nube. Los principales proveedores de servicios en la nube proporcionan clientes en varios lenguajes de programación para realizar consultas a sus servicios de facturación y monitorizar todos los consumos. Mediante esta integración podremos obtener datos que nos permitan optimizar el uso de la infraestructura CloudEdge, ya sea mediante herramientas de gestión tradicionales o mediante el uso de novedosos algoritmos federados que permitan compartir datos de consumo en la nube entre distintas entidades.

RF21	Control de costes	El sistema debe permitir visualizar el coste total de los recursos consumidos por cada proyecto federado cuando estos recursos estén desplegados en la nube.	Obligatorio
RF22	Arbitraje	El sistema debe permitir la comparativa de costes entre proveedores para recomendar y poder seleccionar la opción más asequible para cada modelo.	Opcional

El sistema de plantillas desarrollado permite cumplir con estos dos objetivos ya que muestra un resumen de los costes para cada proyecto en cada uno de los tres proveedores cloud principales en función de los recursos necesarios para desarrollar el caso de uso. El diseño de la capa de costes se describe en más detalle en el punto 5, Demostradores.

RF23	Personalización	La plataforma debe permitir la integración con las credenciales de cada cliente para la interacción con las herramientas de gestión de costes de los proveedores cloud, de forma que la recomendación pueda ser personalizada.	Opcional

Se ha incorporado una interfaz para la gestión de credenciales de AWS, el único proveedor cloud que sigue permitiendo el acceso programático con claves programáticas.

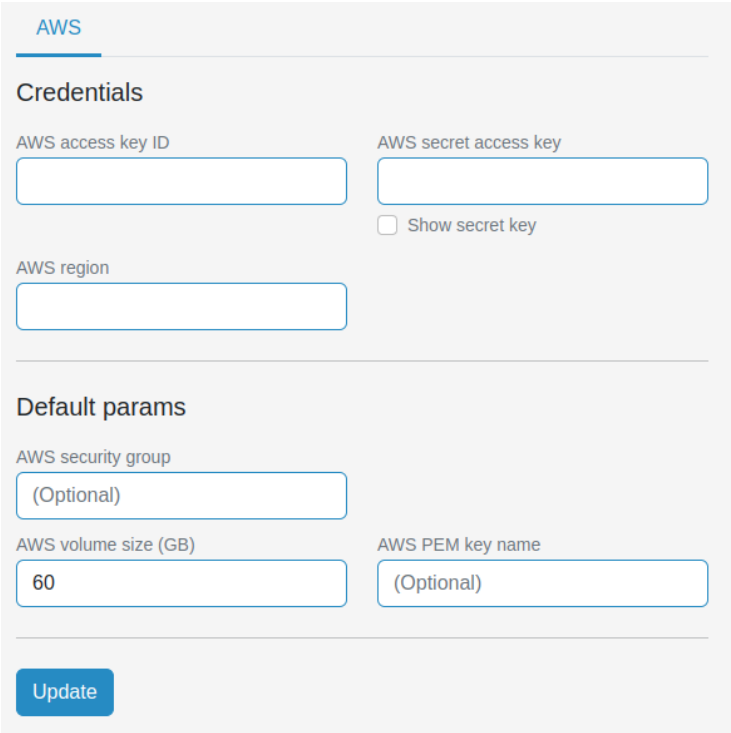


Ilustración 4: Interfaz para la configuración de credenciales en AWS.

En el caso de Google Cloud la integración debe hacerse a nivel de Federación de Identidades, permitiendo al usuario que accede a la plataforma adoptar un rol con los permisos necesarios para lanzar los recursos en la nube.

Por último, Azure requiere una integración más compleja ya que las credenciales de usuario han de tener siempre activa la autenticación multi-factor, lo que impide su uso programático. Para desplegar en esa infraestructura se propone tratarla como si fuera un servicio on-premise y valerse de la versatilidad de la plataforma para lanzar los nodos de computación en servidores de Azure sin necesidad de integrar los permisos.

4. Solución Preliminar

4.1. Federated Averaging

Uno de los objetivos del proyecto es dar soporte a todos los casos de uso de MLEDGE. Para cumplir este objetivo se ha desplegado la plataforma FlaaS descrita en el apartado anterior con nodos y recursos para todos los participantes en el proyecto. Esta plataforma tiene capacidades de aprendizaje federado y uno de los algoritmos principales implementados es el Federated Averaging (FedAvg).

El FedAvg es un algoritmo diseñado para entrenar modelos de aprendizaje automático de manera descentralizada. Permite que múltiples dispositivos o nodos colaboren en la construcción de un modelo global sin necesidad de compartir datos locales, preservando la privacidad y seguridad.

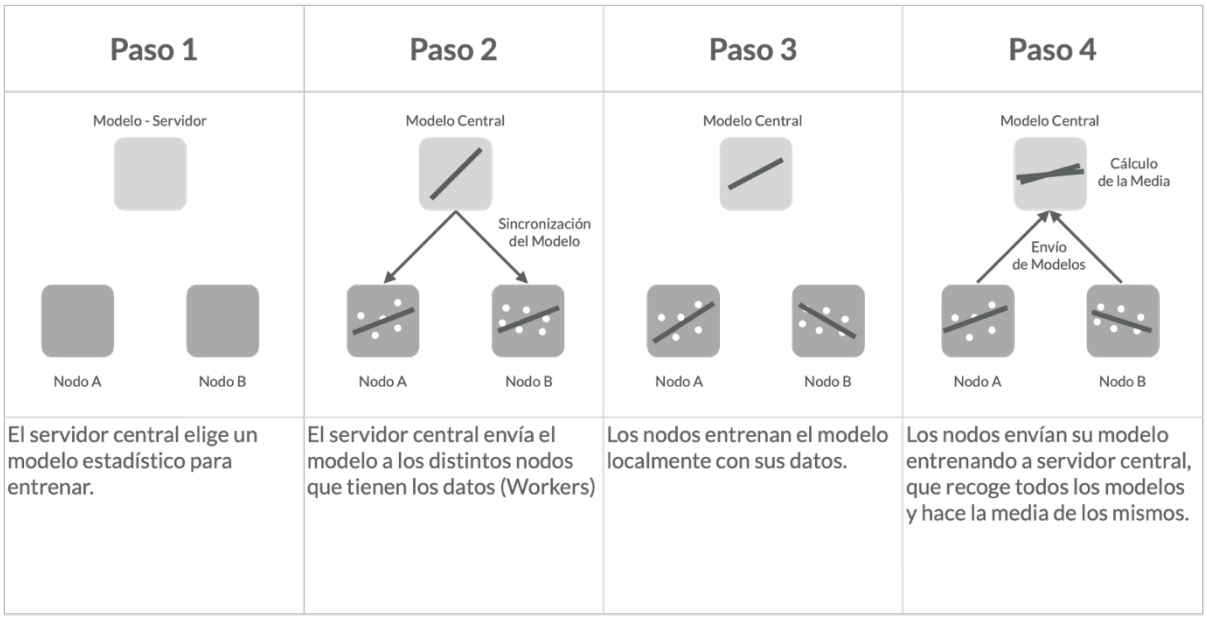


Ilustración 5: Proceso de entrenamiento de modelos federados.

El entrenamiento en cada nodo no difiere de lo que es el entrenamiento de una red neuronal local. Lo que lo diferencia es la agregación de las actualizaciones de los pesos en el servidor cada cierto número de pasos de entrenamiento a elección del usuario.

En primer lugar, se inicializan los pesos del modelo de manera aleatoria en todos los nodos participantes. Este paso garantiza que cada nodo comience con la misma configuración inicial del modelo, lo que es fundamental para asegurar la coherencia a lo largo de las iteraciones de entrenamiento.

Una vez inicializados los pesos, cada nodo seleccionado realiza la propagación hacia adelante utilizando los datos locales que tiene disponibles. Durante este proceso, el nodo calcula las activaciones de cada capa del modelo y genera una predicción basada en su conjunto de datos local. A continuación, se calcula el error correspondiente y se actualizan los pesos del modelo en el nodo. Este procedimiento de entrenamiento local se repite en cada nodo durante varias iteraciones hasta alcanzar el número de pasos definidos para el entrenamiento antes de realizar la agregación.

Tras completar el entrenamiento local, cada nodo envía las actualizaciones de sus pesos al servidor central. Estas actualizaciones pueden incluir los gradientes del error local o los pesos ajustados del modelo después de aplicar un algoritmo de optimización, como el descenso de gradiente estocástico. Este intercambio de información es esencial para combinar los avances realizados en cada nodo individual.

En el servidor central, se lleva a cabo un proceso de agregación de las actualizaciones de pesos recibidos de los nodos participantes. Este promedio ponderado considera el tamaño del conjunto de datos local de cada nodo, lo que permite que las contribuciones reflejen adecuadamente la importancia relativa de los datos disponibles en cada nodo. Este paso es crucial para construir un modelo global que represente de manera equilibrada los datos distribuidos.

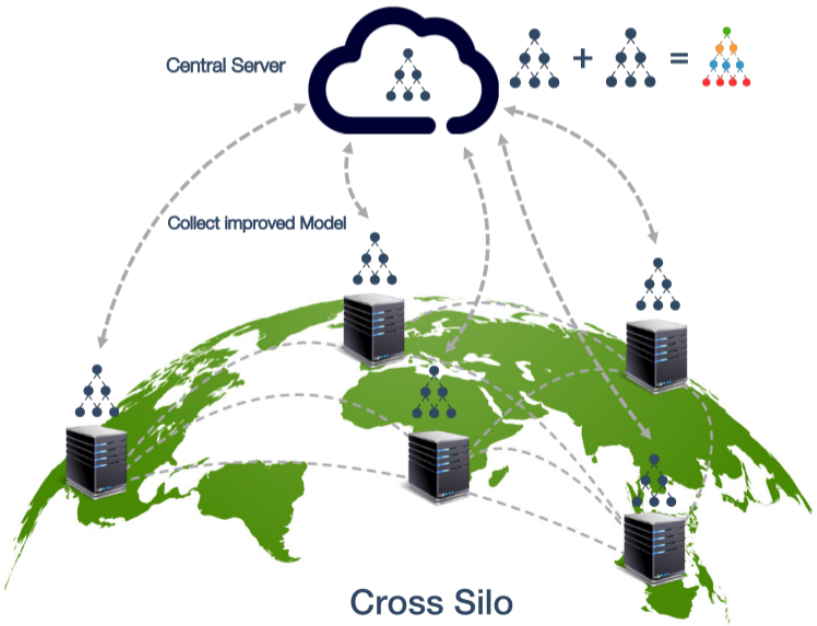


Ilustración 6: Ilustración entrenamiento federado.

Con el promedio ponderado calculado, el servidor central actualiza los pesos del modelo global. Los nuevos pesos son luego enviados de regreso a los nodos participantes, asegurando que todos inicien la siguiente iteración de entrenamiento con la misma configuración de pesos. Este proceso fomenta la convergencia del modelo global al integrar progresivamente las contribuciones de todos los nodos. En la figura anterior puede verse una representación del proceso de aprendizaje federado entre servidores distribuidos al rededor del mundo.

Los pasos descritos (entrenamiento local, envío de actualizaciones, agregación y redistribución de pesos) se repiten durante varias rondas de entrenamiento. Este enfoque iterativo permite que el modelo global se ajuste gradualmente a los datos distribuidos en los nodos participantes, mejorando su capacidad predictiva con cada iteración.

Finalmente, el proceso de entrenamiento se considera completo una vez que se han alcanzado un número predefinido de rondas o se cumple un criterio de convergencia específico. En este punto, el modelo global está listo para ser utilizado en la predicción de

nuevos datos, aprovechando el conocimiento obtenido a partir del entrenamiento en los nodos distribuidos.

4.2. Implementación de protocolos de seguridad

Una parte integral del proyecto es colaborar con la Fundación IMDEA en la integración y demostración de sus investigaciones el campo del Federated Learning seguro.

Como ya se ha mencionado en el entregable anterior, el aprendizaje federado (FL) permite a distintas partes entrenar en colaboración un modelo global sin revelar sus respectivos datos locales. Un paso crucial de FL, es la agregación de los modelos locales para entrenar el global. En este contexto, una de las principales debilidades de FL es su vulnerabilidad a los ataques de envenenamiento. Estos ataques, en los que participantes malintencionados manipulan las actualizaciones de los modelos, pueden comprometer su precisión y afectar negativamente a los usuarios legítimos. Este proceso comparte muchas similitudes con las elecciones puede interpretarse como una consecuencia del principio de una persona, un voto (en adelante 1p1v) principio en el que se basan la mayoría de las normas de agregación actuales. A continuación, se presentan dos protocolos de seguridad para hacer frente a este problema.

4.2.1. Leveraging Quadratic Voting in FL

Otro de las investigaciones de grupo de IMDEA Networks es el protocolo de seguridad desarrollado en “Leveraging Quadratic Voting in Federated Learning”. Acuratio está trabajando para implementarlo de manera efectiva en la plataforma de producción.

A continuación, describimos el artículo y las tareas realizadas hasta ahora para su implementación en la plataforma productiva, tal y como se describen en más detalle en la sección 5.2.

El artículo introduce FEDQV, un nuevo esquema de agregación para aprendizaje federado (FL) que se basa en el concepto de votación cuadrática (Quadratic Voting, QV). FEDQV propone un sistema más robusto al limitar la influencia de estas entidades y mejorar la seguridad del modelo sin comprometer la eficiencia.

Algorithm 1: FEDQV

Input : $\boldsymbol{w}^0 \leftarrow$ random initialisation; $B, \theta \leftarrow$ FEDQV parameters

Server :

for *Iteration* $t \leftarrow 1$ **to** $\frac{T}{E}$ **do**
  Broadcast $\boldsymbol{w}^{t-1}$ to randomly selected set of parties $\mathcal{S}^t$ ($|\mathcal{S}^t| = \mathcal{C} \geq 1$);
  Receive *the local updates* $(\boldsymbol{w}^t, s^t, |\mathcal{D}|)$ from selected parties ($i \in \mathcal{S}^t$) and compute the normalised $\bar{s}_i^t$;
  for $i \leftarrow 1$ **to** N **do in parallel**
    if $\bar{s}_i^t \leq \theta$ *or* $\bar{s}_i^t \geq 1 - \theta$ **then**
      Update $B_i \leftarrow \max(0, B_i + \ln \bar{s}_i^t - 1)$
    Credit voice $c_i^t \leftarrow$ Equation 1 and *Vote* $v_i^t \leftarrow$ Equation 2;
    Budget $B_i \leftarrow \max\left(0, B_i - (v_i^t)^2\right)$
  return $\boldsymbol{w}_n^t \leftarrow \sum_{i=1}^{N} \frac{v_i^t}{\sum_{i=1}^{M} v_i^t} \boldsymbol{w}_{i,n}^t$

Party :

for *Party* $i \in \mathcal{S}^t$ **do in parallel**
  Receive *the global update* $\boldsymbol{w}^{t-1}$ and conduct local training $\boldsymbol{w}_i^t \leftarrow \boldsymbol{w}_i^{t-1} - r_{t-1} \frac{\partial \ell_i(\boldsymbol{w}_i^{t-1}; \mathcal{D}_i)}{\partial \boldsymbol{w}}$;
  Calculate *the similarity score* $s_i^t \leftarrow \frac{\langle \boldsymbol{w}_i^t, \boldsymbol{w}^{t-1} \rangle}{\|\boldsymbol{w}_i^t\| \cdot \|\boldsymbol{w}^{t-1}\|}$ and send back $(\boldsymbol{w}_i^t, s_i^t, |\mathcal{D}_i|)$

Ilustración 7: Pseudo-código algoritmo FEDQV.

Para ello, FEDQV adapta los principios de la votación cuadrática al aprendizaje federado, distribuyendo votos a los participantes en función de su contribución y reputación. Este enfoque introduce varios mecanismos clave.

Por un lado, cada participante recibe un presupuesto limitado de votos, que puede usar en rondas de entrenamiento. A diferencia de los métodos tradicionales que otorgan peso proporcional al tamaño del conjunto de datos, FEDQV asigna votos de manera no lineal, lo que penaliza las contribuciones extremas y reduce la influencia de posibles atacantes.

Además, antes de cada ronda de agregación, los participantes calculan la similitud entre sus modelos locales y el modelo global anterior. Este puntaje se utiliza para determinar cuántos votos pueden emitir. Las actualizaciones que presentan similitudes anómalas son penalizadas para evitar manipulaciones.

Por otro lado, la votación cuadrática introduce un costo creciente para votos adicionales, lo que desalienta comportamientos extremos. Los votos de los participantes se usan para ponderar sus contribuciones durante la agregación, limitando el impacto de entidades maliciosas y priorizando las actualizaciones confiables. Finalmente, para mejorar aún más la robustez, FEDQV incluye un sistema de reputación que ajusta el presupuesto de votos de cada participante. Los usuarios con un historial confiable reciben más votos, mientras que aquellos con comportamientos sospechosos ven reducida su influencia en futuras rondas.

La eficacia de FEDQV fue evaluada a través de un análisis teórico y pruebas empíricas en varios conjuntos de datos populares, como MNIST, Fashion-MNIST, CIFAR10 y FEMNIST. Viendo que FEDQV logra una precisión comparable a FedAvg en escenarios sin ataques y muestra un rendimiento significativamente superior bajo ataques de envenenamiento. Reduce la tasa de éxito de estos ataques hasta cuatro veces en comparación con métodos tradicionales.

El uso de reglas cuadráticas y presupuestos adaptativos limita la influencia de participantes maliciosos, incluso cuando estos intentan aprovecharse de grandes volúmenes de datos locales. En escenarios experimentales, las contribuciones de estos participantes fueron efectivamente neutralizadas. Es compatible con sistemas de privacidad como el cifrado homomórfico y métodos robustos frente a ataques bizantinos, como Trimmed-Mean o Multi-Krum. Al combinarse con estas defensas, FEDQV mejora aún más la precisión y la resistencia del modelo global.v

	MNIST		Fashion-MNIST		CIFAR10		FEMNIST	
	FEDAVG	FEDQV	FEDAVG	FEDQV	FEDAVG	FEDQV	FEDAVG	FEDQV
Data Poison								
Labelflip	98.81±0.03	98.54±0.05	**86.70±0.02**	85.22±0.05	66.88±0.48	67.36±0.22	74.92±2.55	**78.42±0.65**
Gaussian	9.68±0.41	10.49±0.46	10.00±0.00	**27.38±17.38**	15.29±0.57	**19.76±3.66**	4.64±0.13	4.83±0.25
Backdoor								
ACC(%)	37.38±19.82	**98.30±0.15**	74.27±9.12	**78.40±3.95**	59.85±2.18	60.65±1.72	49.78±22.38	**75.20±3.96**
ASR(%)	68.49±22.00	**0.19±0.07**	14.58±12.53	**7.05±6.35**	18.20±5.27	**3.21±1.30**	30.88±7.52	**28.26±9.57**
Scaling								
ACC(%)	10.33±0.05	11.16±0.88	10.22±0.09	11.27±0.99	10.00±0.00	**28.55±18.55**	26.30±21.55	**64.80±1.38**
ASR(%)	99.94±0.06	98.96±1.04	99.74±0.10	98.21±1.45	100.00±0.00	67.66±32.34	0.47±0.08	0.56±0.06
Neurotoxin								
ACC(%)	81.17±15.39	**95.73±1.45**	70.00±7.85	**79.58±1.60**	22.40±7.16	**45.40±3.22**	47.29±18.07	**79.99±0.70**
ASR(%)	23.19±2.25	**18.11±1.67**	20.65±2.21	18.12±4.16	51.63±1.03	57.42±1.91	40.42±4.35	**9.00±1.29**
Model Poison								
Krum	10.57±0.39	**97.96±0.14**	10.00±0.00	**79.43±0.86**	10.00±0.00	**53.27±1.12**	5.20±0.22	**51.86±3.06**
Trim	10.04±0.16	**98.36±0.11**	10.00±0.00	**84.45±0.70**	10.00±0.00	**57.33±2.34**	5.09±0.33	**52.19±4.52**
Min-Max	35.00±25.38	**85.32±6.45**	10.00±0.00	**67.25±7.44**	10.00±0.00	**19.07±6.97**	56.37±13.67	**72.58±2.11**
Min-Sum	96.69±0.94	95.97±0.59	10.88±0.87	**83.93±0.81**	17.40±4.27	**43.94±3.56**	52.56±23.91	**72.36±1.61**
QV-Adaptive	**71.43±22.67**	56.94±23.95	35.92±4.60	**62.13±11.25**	10.00±0.00	11.14±1.14	22.08±18.72	**43.78±20.72**

Ilustración 8: Comparativa rendimiento modelo FEDQV vs FedAvg.

FEDQV representa una solución innovadora que fortalece el aprendizaje federado frente a ataques y manipulaciones. Su diseño inspirado en la votación cuadrática combina equidad y robustez, al mismo tiempo que permite integrar mecanismos avanzados de privacidad. Este enfoque no solo mejora la precisión del modelo global, sino que también establece un marco confiable y adaptable para aplicaciones de aprendizaje federado en entornos altamente vulnerables.

4.2.2. Securing Federated Systems Against Poisoning Attacks

En este apartado se presentará otro protocolo de seguridad en el que están colaborando IMDEA y Acuratio para su implementación de manera productiva. Este protocolo fue publicado en “Securing Federated Sensitive Topic Classification against Poison Attacks”.

La investigación se centra en la clasificación de URL con contenido sensible, como temas de salud o creencias políticas, en un marco respetuoso con la privacidad que cumple con regulaciones como el Reglamento General de Protección de Datos (GDPR).

Para reducir la vulnerabilidad a los ataques, los autores desarrollan un método de agregación robusto que combina la lógica subjetiva y la detección basada en residuos. Este enfoque introduce una puntuación de reputación dinámica para cada participante en el sistema, que se asigna en función de la calidad y la coherencia de las actualizaciones enviadas. La solución consta de tres pasos clave. En primer lugar, se utiliza un sólido análisis de regresión para identificar y limpiar las actualizaciones sospechosas enviadas por los nodos. Así se minimiza el impacto de los mensajes maliciosos. A continuación, se evalúan las actualizaciones y se les asignan métricas de confianza basadas en el rendimiento histórico de cada nodo. Las puntuaciones de reputación determinan el peso que se asigna a las futuras contribuciones de un nodo.

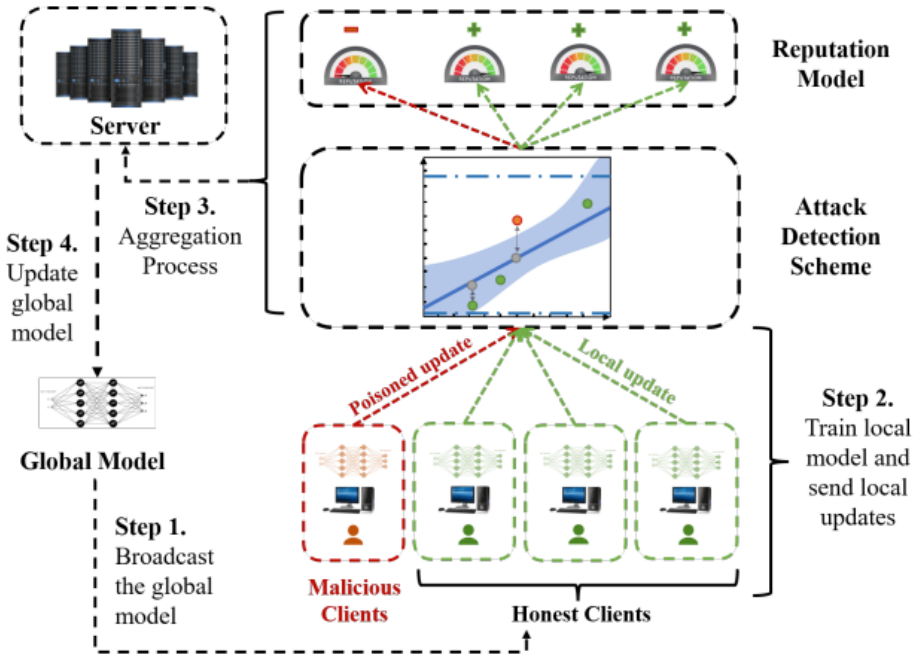


Ilustración 9: Diseño de experimentos FedSecure.

Los participantes con una reputación baja, que indica un posible comportamiento malicioso, ven reducida su influencia en el modelo global. Por último, el servidor central utiliza la

reputación para ponderar las actualizaciones del modelo local e integrar en el modelo global sólo las contribuciones dignas de confianza. Este enfoque garantiza que el modelo conserve su precisión y estabilidad incluso frente a los ataques.

La eficacia de la solución se validó mediante un enfoque combinado de simulaciones teóricas, pruebas empíricas y experimentos con usuarios reales. Por un lado, el sistema muestra una notable resistencia frente a ataques de envenenamiento, manteniendo una alta precisión en la clasificación de contenido sensible. Comparado con algoritmos tradicionales como FedAvg y defensas como FLTrust, el modelo propuesto presenta mejoras significativas en robustez y rapidez de convergencia. Por otro lado, a diferencia de los métodos centralizados, que dependen de conjuntos de datos estáticos, este esquema puede incorporar etiquetas emergentes, como las relacionadas con el COVID-19. Las pruebas revelaron que el modelo aprendió rápidamente estas nuevas categorías, superando los sistemas centralizados en escenarios dinámicos. Aparte, la tasa de éxito de los ataques se redujo considerablemente. Esto se atribuye a la combinación de detección temprana de anomalías y el uso de puntuaciones de reputación.

Representa un avance significativo en la protección de modelos de aprendizaje federado. Su capacidad para combinar robustez frente a ataques con un diseño flexible y adaptable lo convierte en una solución prometedora para aplicaciones en entornos de datos sensibles.

4.3. MLEDGE-Acuratio security Add-On

La plataforma **Acuratio** está evolucionando para ofrecer un nuevo **Security Add-On** que incorporará las técnicas avanzadas de seguridad desarrolladas en colaboración con **IMDEA**. Este complemento de seguridad estará diseñado para proteger la plataforma frente a amenazas avanzadas, como los ataques de envenenamiento y otras vulnerabilidades específicas en entornos distribuidos. El Security Add-On integrará algoritmos robustos que analizan y mitigan los riesgos en tiempo real, proporcionando una capa adicional de protección en escenarios críticos donde la integridad de los datos y la resiliencia del sistema son esenciales.

Este add-on será parte de una versión especial de la distribución de Acuratio, denominada **MLEDGE-Acuratio**, que combinará las innovaciones de IMDEA con las capacidades avanzadas de análisis de costos desarrolladas durante el proyecto. Esta integración garantizará no solo un sistema más seguro, sino también una plataforma optimizada para decisiones informadas y sostenibles, ofreciendo una solución de alto valor añadido para los clientes que operan en entornos complejos y distribuidos.

5. Demostradores

5.1. Caso de uso de optimización de costes cloud

Para poder monitorizar los costes asociados con un proyecto de aprendizaje federado, se están desarrollado interfaces en la que se puedan consultar de forma unificada los costes asociados a los recursos lanzados en los principales proveedores de servicios en la nube (GCP, AWS y Azure). Esto, en combinación con la plataforma FLaaS que permite lanzar y gestionar recursos en dichas nubes y coordinar un entrenamiento federado permite optimizar los costes asociados a estos proyectos.

Diseñar una interfaz que resuma los costos de una infraestructura en AWS, GCP y Azure requiere claridad y una presentación bien organizada. A continuación, se describen los componentes principales de esta interfaz:

Entrada de Parámetros de Infraestructura

Una sección donde el usuario puede definir los recursos necesarios:

- vCPUs: Entrada numérica.
- RAM: Entrada numérica en GB.
- GPUs: Selector desplegable con modelos específicos (e.g., NVIDIA A100, T4).
- Almacenamiento SSD: Entrada numérica en GB.

Tabla Comparativa de Costos

Una tabla clara con columnas para cada proveedor con el coste mensual estimado.

Filas adicionales, opcionales:

- Costo por vCPUs.
- Costo por GB de RAM.
- Costo por GPU (si aplica).
- Costo por almacenamiento SSD.

También podrían incluirse detalles adicionales como descuentos aplicados o notas sobre limitaciones regionales o disponibilidad.

Gráficos Comparativos Opcionales:

Podrían incluirse gráficos de barras para representar el coste total por proveedor o para dividir el coste en categorías (vCPUs, RAM, GPU, SSD).

Para recoger los datos necesarios de costes y características de infraestructura en AWS, GCP y Azure, se pueden utilizar sus respectivas APIs. Estas APIs ofrecen acceso a información sobre precios, características de servicios y configuraciones específicas.

AWS proporciona varias APIs para consultar los costos y características de sus servicios:

AWS Pricing API:

Proporciona información detallada sobre precios de servicios como EC2, EBS, GPUs, etc.

Endpoint relevante: *GetProducts* para consultar precios y características como precio por hora/mes, tipos de instancia, opciones de almacenamiento. Permite filtrar por atributos como región, tipo de instancia, y sistema operativo.

AWS Compute Optimizer API:

Puede ser útil para sugerencias de costos optimizados.

GCP ofrece la Cloud Billing Catalog API para acceder a precios de sus servicios:

Cloud Billing Catalog API:

Permite recuperar información de precios para servicios como Compute Engine, discos SSD, y GPUs.

Endpoint relevante: *services.skus.list* para obtener la lista de SKU (Stock Keeping Units) y precio unitario por hora/mes, categorías como vCPUs, RAM, discos, y GPUs. Permite filtrar por tipo de servicio (e.g., compute.googleapis.com), región y configuraciones específicas.

Azure cuenta con la Retail Prices API para acceder a los precios de sus servicios:

Azure Retail Prices API:

Proporciona precios públicos de máquinas virtuales, almacenamiento y otros recursos.

Endpoint relevante: *prices* con filtros como región, tipo de servicio, y categoría de producto, permite recabar prrecio por hora/mes, SKU, características del recurso.

A continuación, se describe el proceso a seguir para integrar estas APIs en la plataforma:

1. Autenticación: Configurar credenciales para cada API.
2. Consulta y Filtrado: Usar parámetros de filtro específicos para recuperar solo los datos necesarios (región, tipo de recurso, etc.).
3. Normalización de Datos: Las APIs de los proveedores pueden devolver datos en formatos diferentes; es necesario unificar las unidades y estructuras para comparar.
4. Almacenamiento en Caché: Guardar respuestas de API en una base de datos o memoria caché para reducir llamadas frecuentes y mejorar tiempos de respuesta.
5. Actualización Periódica: Configurar cron jobs o triggers para actualizar la información cada cierto tiempo, asegurando precios actualizados.

En la siguiente figura se muestra la interfaz de configuración del conjunto de recursos en la nube necesarios para resolver un caso de uso federado. Las opciones disponibles incluyen: el número de nodos, la cantidad de CPUs por instancia, la cantidad de memoria RAM por nodo y el tamaño del disco para cada servidor.

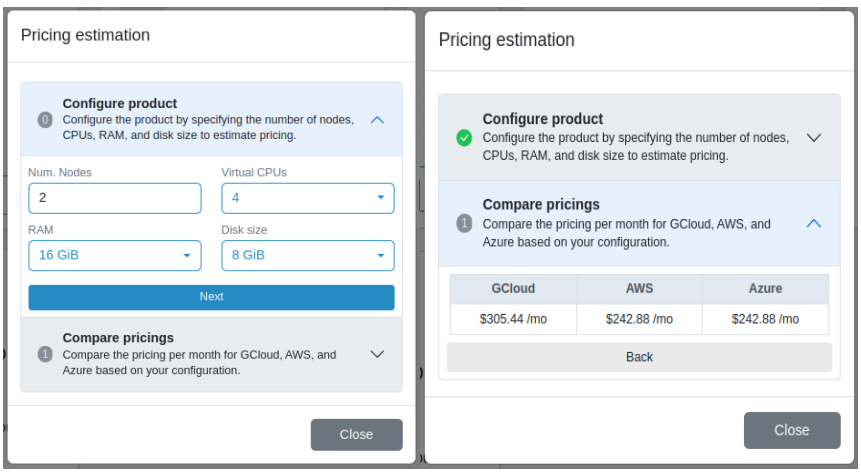


Ilustración 10: Interfaz estimación costes de proyectos federados.

Una vez configurados todos estos aspectos, la plataforma muestra una estimación del coste mensual de la configuración elegida en cada uno de los tres principales proveedores de servicios en la nube (GCP, AWS y Azure). En la figura anterior se muestra la interfaz que resume los costes estimados para cada uno de los proveedores.

Una interfaz como la de este demostrador, que compara los costos de infraestructura en múltiples nubes puede ser una herramienta estratégica para reducir los costes en la nube de una organización.

En primer lugar, podría ayudar a seleccionar el proveedor más económico permitiendo comparar los costos de recursos específicos (vCPUs, RAM, SSD, GPUs) en diferentes proveedores y regiones, destacando la opción más barata. Muchas veces, el costo de los servicios varía significativamente entre regiones dentro del mismo proveedor. La interfaz podría también facilitar la selección de la región más económica para las necesidades de la organización.

También podría ayudar con el descubrimiento de las opciones más rentables recomendando instancias económicas como instancias spot (AWS), preemptible (GCP) o spot VMs (Azure) que suelen ser mucho más baratas para cargas de trabajo tolerantes a fallos. En muchos casos existen diferencias significativas entre costes por uso puntual (por hora) y contratos a largo plazo (reservas o ahorros por compromiso).

En tercer lugar, uno de los elementos esenciales de esta interfaz es la simulación de costes. Esta característica ofrece una vista previa del costo total para configuraciones específicas antes de desplegarlas, lo que evita sorpresas en las facturas. Además, permite probar diferentes configuraciones (más RAM o menor almacenamiento) y comparar sus implicaciones en el costo. Incluso puede ayudar a prever el impacto de aumentos en el uso o cambios en las necesidades de la infraestructura.

En línea con la filosofía de la federación de datos, esta interfaz de comparación de costes también da soporta a la toma de decisiones multinube, ayudando a seleccionar diferentes proveedores para cada componente de la infraestructura (almacenamiento en GCP, cómputo en AWS), aprovechando los puntos fuertes de cada uno. Además evita la dependencia de un solo proveedor.

Finalmente, también puede ayudar con la identificación de ineficiencias y la implementación de políticas de ahorro mediante la integración con herramientas de despliegue automatizado.

En resumen, una interfaz de comparación de costes no solo facilita identificar opciones más económicas, sino que también fomenta una cultura de optimización continua. Esto puede traducirse en un ahorro significativo a corto y largo plazo, especialmente en organizaciones con infraestructuras en crecimiento o distribuidas en múltiples nubes como es el caso de los escenarios de federación de datos.

5.2. Caso de uso de mejora de protocolos de FL

Tal y como se ha descrito anteriormente FedQV es un mecanismo de agregación basado en votación cuadrática (QV) para el Aprendizaje Federado (FL). Este enfoque mejora FedAvg al abordar sus vulnerabilidades frente a ataques de envenenamiento. Al abordar las debilidades inherentes de FedAvg y mantener la compatibilidad con los sistemas existentes, FedQV ofrece un marco de agregación más seguro y confiable para el Aprendizaje Federado.

```python
class FederatedQV(federatedmodel.FederatedModel):
    """FederatedQV class for federated model averaging.
    This class extends the `FederatedModel` class and implements the Federated
    Quadratic Voting (FederatedQV) algorithm for model averaging in a federated
    learning setting.
    Attributes:
        budgets (dict): A dictionary mapping participant UUIDs to their respective
        budgets.
        theta (float): The threshold value for voice credit calculation.
    """
```

Ilustración 11: Implementación de la clase FEDQV en la plataforma.

Se ha integrado un módulo de agregación en la plataforma Federada capaz de sustituir la agregación tradicional por la votación cuadrática. De este modo se consigue el objetivo de facilitar el uso de este mecanismo de agregación segura al desarrollar una interfaz sencilla que permite implementar FedQV con una sola línea de código tal y como se puede observar en la siguiente figura.

Para implementar FedQV se ha heredado de la clase FederatedModel que implementa FedAvg en la plataforma FLaaS. De esta forma partimos de una clase base que ya tiene implementado todo el proceso de coordinación entre nodos, tanto al inicio del entrenamiento como en las sucesivas fases de agregación.

A la clase base se le han añadido dos nuevos atributos:

- **Budgets:** Es un diccionario cuyas claves son los uuid de los nodos (identificadores únicos con los que están registrados en la base de datos) que participan en el entrenamiento como clientes y los valores son el budget que le queda a cada uno de esos nodos.
- **Theta:** Es un valor que se usa para, una vez escalados el valor de las *cosine similarities* calculadas por los nodos a entre 0 y 1, descartar las actualizaciones de los pesos de aquellos con *cosine similarity* menor que theta o mayores que 1-theta (uno menos theta).

De la clase original se han modificado los métodos que realizan la operación de agregación tanto en los clientes como en el servidor, así como los métodos que gestionan el envío de las actualizaciones en los clientes y de recepción en el servidor.

La función de agregación en los clientes se ha modificado para que calcule el *cosine similarity* entre sus pesos actuales y los últimos pesos globales tras la última ronda de agregación. Su función de envío de actualizaciones de los pesos se ha modificado para enviar ese *cosine similarity* al servidor junto con sus actualizaciones.

```
sim_score = (
   smp.cosine_similarity(
   array_ops.reshape(flattened_new_weights, [1, -1]),
   array_ops.reshape(flattened_old_weights, [1, -1]),
)
.flatten()
.tolist()[-1]
)
```

En el servidor la función de recepción de actualizaciones de los pesos de los clientes se ha modificado para recibir el *cosine similarity* de cada cliente y guardarlo como valor en un diccionario usando como clave el uuid del cliente remitente.

Una vez recibidas las actualizaciones de los pesos y el *cosine similarity* de los clientes, el servidor realiza un escalado de esta última métrica para convertir el valor más pequeño en un 0, el mayor en un 1 y el resto a su correspondiente valor escalado. Después convierte en 0 todos aquellos valores que estén por debajo del valor theta o por encima del valor 1-theta y aplica la función *-ln(valor + 1)* a los valores restantes. Calculando de esta manera el valor del voto que cada cliente tendrá en esta ronda de entrenamiento.

```
scaler = MinMaxScaler()
normalized_sim_scores = scaler.fit_transform(
    np.array(sim_scores).reshape(-1, 1)
)

voice_credit = []
for i in range(len(normalized_sim_scores)):
    if normalized_sim_scores[i][0] <= self.theta:
        voice_credit.append(0)
    elif normalized_sim_scores[i][0] >= 1 - self.theta:
        voice_credit.append(0)
    else:
        voice_credit.append(
            -math.log(normalized_sim_scores[i][0]) + 1
        )
```

A continuación, para todos aquellos clientes cuyo voto no sea ya 0, comprueba que en el diccionario de *budgets* les sigue quedando presupuesto suficiente para votar. Si el presupuesto ya era 0, convierte ese voto en 0. Si el presupuesto es mayor que cero, pero menor que el valor del voto, usa el valor del presupuesto como voto y pone el presupuesto a 0. Si no se da ninguno de estos dos casos, le resta al presupuesto existente el valor de este voto.

Los votos se escalan multiplicando cada uno por el tamaño del *dataset* de cada cliente y se comprueba que al menos uno de los votos sea mayor que cero antes de proceder a agregar las actualizaciones de los pesos.

Para agregar las actualizaciones de los pesos, se multiplica la actualización de cada cliente por la raíz cuadrada de su voto dividida por la suma de las raíces cuadradas los votos de todos los clientes.

Por último, se suman todas las actualizaciones modificadas por el paso anterior y se envían de vuelta a los clientes, que ya pueden continuar con el proceso normal de entrenamiento.

Las principales mejoras esperadas por este desarrollo son:

Robustez frente a ataques de envenenamiento

La dependencia de FedAvg en promedios ponderados lo hace vulnerable a partes maliciosas con grandes conjuntos de datos, que pueden enviar actualizaciones manipuladas. FedQV contrarresta esto utilizando un costo cuadrático para votar, lo

que limita la influencia de actualizaciones maliciosas al penalizar comportamientos extremos en la votación.

Las partes envían sus pesos de agregación basándose en la similitud entre sus actualizaciones locales y el modelo global. Los comportamientos sospechosos, como puntuaciones de similitud anormalmente altas o bajas, son penalizados, reduciendo el impacto de contribuciones adversas.

Presupuestos de votación adaptativos

FedQV integra un mecanismo basado en reputación para asignar presupuestos de votación desiguales, otorgando más influencia a las partes de confianza y reduciendo el impacto de las partes maliciosas. Esto fortalece aún más el modelo frente a ataques de envenenamiento.

Mecanismo de veracidad

FedQV emplea una estrategia de agregación veraz, incentivando a las partes a enviar actualizaciones honestas mediante sanciones por informes falsos. Las reglas de votación enmascaradas ocultan la influencia exacta de cada voto, desalentando manipulaciones estratégicas.

Integración sencilla

FedQV puede complementar los mecanismos existentes resistentes a ataques bizantinos y preservadores de privacidad, mejorando la resiliencia frente a ataques de envenenamiento y privacidad sin requerir cambios significativos en la arquitectura subyacente de FL.

Mejoras en el rendimiento

El análisis teórico confirma que la tasa de convergencia de FedQV es comparable a la de FedAvg en condiciones normales y adversas. Los resultados experimentales demuestran una mayor precisión y robustez en comparación con FedAvg en diversos escenarios de ataque.

Durante estos meses también se ha trabajado en la implementación de la investigación Securing Federated Systems Against Poisoning Attacks del paper de IMDEA. Se han identificado los pasos necesarios para finalizar su desarrollo antes del siguiente hito del proyecto.

Este método mejora un modelo federado al implementar un enfoque robusto basado en la combinación de lógica subjetiva y detección de ataques basada en residuos para abordar problemas como ataques de envenenamiento. Este método no solo mejora la seguridad y la precisión del aprendizaje federado, sino que también proporciona un marco más eficiente para manejar datos distribuidos en entornos reales.

Las principales mejoras aportadas por este desarrollo son:

Protección contra ataques de envenenamiento

El sistema identifica actualizaciones sospechosas enviadas por clientes maliciosos y las rectifica o reduce su impacto en el modelo global. Esto asegura que el modelo federado sea más resistente frente a clientes que intentan comprometer la precisión general.

Puntuación de reputación

El método introduce un modelo de reputación que evalúa el comportamiento histórico de los clientes. Esto permite asignar pesos de agregación basados en la confiabilidad de los clientes, priorizando actualizaciones de clientes confiables y reduciendo la influencia de clientes sospechosos.

Velocidad de convergencia mejorada

En comparación con otros métodos de agregación, el enfoque propuesto logra tasas de convergencia más rápidas (1.6× a 2.4× más rápido) mientras mantiene o supera la precisión en escenarios con y sin ataques.

Adaptación a datos dinámicos

Este sistema permite la actualización continua del modelo a medida que los usuarios etiquetan nuevos contenidos sensibles localmente. Así, el modelo puede incorporar rápidamente etiquetas nuevas, como las relacionadas con la pandemia de COVID-19, logrando una mayor relevancia y actualidad.

Análisis teórico y validación experimental

El documento valida teóricamente que el enfoque propuesto converge bajo ciertas condiciones estándar y muestra que mantiene una precisión alta y una baja tasa de éxito de ataques en comparación con otros métodos.

6. Conclusión y siguientes pasos

En este entregable se presenta el diseño definitivo y la implementación de la plataforma FLaaS para el proyecto MLEDGE, detallando cómo los distintos componentes de la plataforma cumplen con los requisitos establecidos en la propuesta de diseño del entregable anterior. Asimismo, se demuestra que estos componentes pueden servir como base para el desarrollo de los casos de uso restantes de MLEDGE.

El alcance de este entregable abarca el cumplimiento de los objetivos de varias actividades clave dentro del proyecto, entre las cuales se destacan:

A2.1: Uso eficiente de *Federated Learning* (FL) en nubes híbridas, con énfasis en la protección contra ataques, incluidos los de envenenamiento e inferencia.

A2.2: Protección de datos sensibles o confidenciales que son intercambiados entre diferentes dominios administrativos en la nube, considerando también el borde de la nube, que es potencialmente más vulnerable.

A2.3: Garantizar la equidad en la distribución de costos y beneficios cuando la computación en el borde se emplea para el entrenamiento colaborativo de modelos de aprendizaje automático.

A2.4: Gestión de los desafíos asociados con la portabilidad de datos en el borde de la nube.

A2.5: Implementación de prácticas de DevOps y desarrollo continuo para servicios de aprendizaje automático en el borde de la nube (FLaaS).

A5.2: Implementación y pruebas de los componentes de infraestructura en la nube.

Además, se ha completado el diseño preliminar de la capa de costos y la interfaz para el despliegue de infraestructura federada en múltiples nubes. Esta interfaz, como se ha demostrado, permitirá evaluar el coste de los proyectos federados y seleccionar el proveedor más económico, optimizando así la estructura de costos de la plataforma.

Dentro del marco del proyecto, también se ha brindado soporte a los casos de uso relacionados con la economía tradicional y digital. Esto incluyó el despliegue de infraestructura en la nube mediante la plataforma, así como el diseño de mecanismos para la carga, gobernanza y procesamiento preliminar de datos en ambos casos de uso. Como parte de este apoyo, se ha proporcionado soporte técnico en el desarrollo y despliegue de modelos tanto centralizados como federados.

Finalmente, se ha implementado una de las dos técnicas de privacidad desarrolladas por MLEDGE (FedQV), y se han sentado las bases para la implementación de la segunda técnica durante el desarrollo del último entregable del proyecto.

En definitiva, con este entregable se completa la implementación de los componentes clave de la infraestructura de MLEDGE sentando las bases para el éxito final del proyecto y permitiendo el desarrollo de todos sus casos de uso.