UNIVERSIDAD DE BUENOS AIRES

INFORME DE TRABAJO PROFESIONAL

DeltaML
Plataforma descentralizada de machine learning que preserva la privacidad del usuario

---

Autores:
Fabrizio GRAFFE 93158
Agustín ROJAS 91462

Tutor:
Dr. Mariano BEIRÓ

9 de noviembre de 2019

Índice general

1 Introducción

El siguiente trabajo se presenta en el marco del desarrollo del Trabajo Profesional para la carrera Ingeniería en Informática de los estudiantes Fabrizio Sebastian Graffe y Agustín Rojas.

El objetivo es aplicar los conocimientos adquiridos durante la carrera, por ésta razon se optó por desarrollar **"DeltaML: Plataforma descentralizada de machine learning que preserva la privacidad del usuario"**.

Machine learning (o "aprendizaje automático") es la rama del área de la inteligencia artificial centrada en el estudio y construcción de sistemas capaces de aprender de los datos, identificar patrones y realizar decisiones con mínima intervención humana.

Blockchain [1] o DLT (Descentralized Ledger Technology), por su lado, es una tecnología que consta de un registro contable distribuido en una red de nodos. Este registro almacena las transacciones que se realizan entre los nodos de la red y cada nodo tiene una copia completa. A su vez, para evitar que nodos maliciosos puedan cometer fraudes y/o alteraciones al registro, la tecnología cuenta con algoritmos de consenso y de prueba de trabajo realizado (Proof of Work). Las ventajas de la tecnología Blockchain por sobre bases de datos tradicionales son:

- **Mayor transparencia:** Todas las transacciones son públicas, por lo que cualquier participante de la red puede verlas.
- **Criptograficamente segura:** Debido a los algoritmos antes nombrados, para poder cometer fraude se necesitaría un poder de computo mayor al 51 % de la red (algo que no es posible hoy en día, al menos contra las redes de Bitcoin o Ethereum, ni por las mayores empresas de software en el mundo).
- **Irreversibilidad:** Una transacción registrada en la blockchain no puede ser alterada por ninguno de los participantes de la red (otra vez, se necesitaría un poder de computo mayor al 51 % de la red).

2 Descripción del problema

Tanto el área de Machine Learning como la tecnología Blockchain son elementos que están disrumpiendo la industria del software en la actualidad y tienen cada vez mas importancia en la vida diaria de las personas. En algunos casos su aplicación ha resultado en avances con un impacto positivo en la humanidad, pero también existen casos en los que se su impacto ha sido negativo.

Un ejemplo muy importante de mal uso de la tecnología es la red social Facebook, la cual ha tenido varios casos de violación de la privacidad de los datos de sus usuarios, venta e intercambio de éstos con otras compañías. Además del uso de técnicas de aprendizaje automático para generar adicción a las novedades en su plataforma y generación de cámaras de eco (echo chambers) por medio de filtrado de contenido [1], por nombrar algunos.

Así como Facebook incurrió en estas malas prácticas que debilitan y manipulan a los usuarios de su plataforma, un abundante número de empresas e incluso estados también lo hacen día a día.

La creciente necesidad de mantener la privacidad de nuestros datos, que día a día son mas valiosos, está provocando que diferentes gobiernos comiencen a plantear regulaciones mas estrictas en lo relativo al uso de los mismos. Esto, a su vez, está impulsando una gran cantidad de iniciativas en el mundo para cambiar el paradigma actual, donde las empresas son dueñas de los datos de las personas, a uno donde las personas sean dueñas de sus propios datos y puedan venderlos para un uso determinado y ningún otro. En el ámbito local se tiene el ejemplo de Wibson como una empresa que va en ese camino.

Teniendo en cuenta este contexto, se eligió la temática de este trabajo con la perspectiva de que en los años por venir se necesitarán maneras de entrenar modelos de Machine Learning que preserven la privacidad de las personas, es decir, sin que las empresas necesiten tener una copia de sus datos en sus bases de datos. Por lo cual el presente trabajo trata de contribuir en este paso hacia un futuro donde las personas sean dueñas de su información.

[1] para mostrar a los usuarios solo opiniones similares a la suya, provocando así un refuerzo de su propia visión, aprovechándose de sesgos propios de los humanos como el Sesgo de Confirmación o Confirmation Bias

3 Estado del arte

Actualmente existen varios proyectos en desarrollo y técnicas en investigación que intentan atacar la misma problemática de diferentes maneras.

- **Federated Learning: [2]** El aprendizaje federado es un método de aprendizaje automático en el que el objetivo es entrenar a un modelo centralizado de alta calidad con datos de entrenamiento distribuidos entre un gran número de clientes, cada uno con conexiones de red poco confiables y relativamente lentas. Los algoritmos de aprendizaje para esta configuración funcionan de la siguiente manera: en cada iteración, cada cliente calcula de forma independiente una actualización del modelo actual en función de sus datos locales y comunica esta actualización a un servidor central, donde las actualizaciones enviadas por los clientes se agregan para calcular una nueva versión global del modelo. Los clientes típicos en este entorno son los teléfonos móviles, y la eficiencia de la comunicación es de suma importancia. Actualmente, Google está investigando [3] [4] [5] [6] [7] esté método de aprendizaje automático y publicó varios papers al respecto, analizando mejoras y aplicaciones para dispositivos móviles (tales como predicción de palabras para GBoard).
- **OpenMined: [8]** Marketplace descentralizado de modelos de machine learning. Opera sobre la red de Ethereum. Utiliza Federated Learning para el entrenamiento de los modelos predictivos sobre Tensorflow y Pytorch ya que permiten entrenamiento desde clientes web. Modelos encriptados con homomorphic encryption [9] para una transmisión de datos segura. Smart contracts regulan el marketplace y pagan a quienes gastaron poder de computo entrenando un proporcional de acuerdo a cuanto aportaron a la mejora del modelo global. Actualmente en desarrollo.
- **DML: [10]** Muy similar a OpenMined.
- **NumerAI: [11]** Fondo de inversión que genera predicciones utilizando modelos predictivos crowd-sourced. Transforma y regulariza datos financieros en problemas de Machine Learning para ser resueltos por una red global de científicos de datos.
- **Augur: [12]** Es un protocolo de predicción de mercados destinado a ser propiedad de las personas que lo usan. Augur es un oráculo descentralizado y un protocolo peer to peer para la predicción de mercados. Augur es proyecto open-source. Es un conjunto de smart contracts escritos en Solidity que operan sobre Ethereum.
- **Golem Network: [13]** Primera supercomputadora descentralizada que crea un marktplace global de poder de computo. Golem conecta computadoras en una red peer-to-peer, permitiendo tanto a dueños de aplicaciones como a usuarios individuales (requestors") alquilar recursos de las maquinas de otros usuarios

("providers"). Los pagos entre requestors, providers y desarrolladores de aplicaciones se realizan por medio de un sistema de transacciones basado en la red de Ethereum. El sistema está orientado a competir con los proveedores de infraestructura o servicios cloud para aplicaciones que requieran gran capacidad de computo reduciendo el precio de dicha capacidad. Como consecuencia, aplicaciones complejas como la representación CGI, el cálculo científico y el aprendizaje automático se volverían más accesibles.

- **OceanProtocol: [14]** Ecosistema destinado a compartir activos y servicios. Los activos son datos y algoritmos. Los servicios son la integración, procesamiento, computación y almacenamiento. Ocean Protocol es un protocolo de intercambio de datos descentralizado, que permite que personas compartan y moneticen sus datos a la vez que garantiza control, auditabilidad, transparencia y conformidad a todos los actores involucrados.
- **Wibson: [15]** Mercado de datos descentralizado, basado en blockchain, que proporciona a las personas una forma segura y anónima de vender información privada validada en un entorno confiable. Opera sobre la red de Ethereum. Las personas pueden vender sus datos por medio de la aplicación móvil de Wibson.

4 Objetivos

El presente trabajo consta de los siguientes objetivos principales:

- Desarrollar un marketplace de modelos de ML que permita a usuarios dueños de estos modelos entrenarlos de manera descentralizada preservando, al mismo tiempo, la privacidad de los datos que serán utilizados para dicho entrenamiento.
- Desarrollar una forma de recompensar a los usuarios que provean datos y poder de computo para el entrenamiento de los modelos.
- Desarrollar un framework de Federated Learning de código libre para generar un aporte a la comunidad.
- Desarrollar un caso de uso de un modelo que requiera entrenamiento para hacer prueba del sistema desarrollado durante este trabajo.

5 Introducción teórica

5.1. Machine Learning

Machine Learning[16] o (Aprendizaje automático) es un subconjunto de la inteligencia artificial que se centra en el estudio y construcción de sistemas capaces de aprender de los datos, identificar patrones y realizar decisiones con mínima intervención humana.

Una definición más técnica podría ser:

"*Un programa aprende de la experiencia (**E**) con respecto a alguna tarea (**T**) y medida de desempeño (**P**), si su desempeño en dichas tareas mejora con la experiencia.*"

Ejemplo: un programa que aprende a jugar al ajedrez.

- **E** = la experiencia de jugar muchas partidas de ajedrez.
- **T** = la tarea de jugar una partida de ajedrez.
- **P** = la probabilidad de que el programa gane la próxima partida.

El programa que aprende de la experiencia, en el ámbito del aprendizaje automático es llamado **modelo**.

5.1.1. Datasets

Contiene las muestras (experiencias) necesarias para el proceso de aprendizaje y evaluación del modelo. **Ejemplo:** un programa que aprende a jugar al ajedrez.

- **Set de entrenamiento:** muestras usadas para entrenar modelo.
- **Set de test:** muestras para evaluar cuán bien aprendió el modelo.

Si bien existen otros esquemas para separar el dataset y evaluar el modelo, se eligió éste al ser simple para no extender de más el alcance del trabajo.

5.1.2. Entrenamiento y predicción

Un modelo de aprendizaje automatico aprende de las experiencias o muestras dadas en forma de datos dentro de un **Dataset**. Dicho proceso se dá como se describe a continación:

Entrenamiento

- Se obtiene o genera el **dataset** (esto puede ser simple o puede contemplar la combinación de varias fuentes de datos).
- Se realiza un **pre-proceso de los datos** contenidos en el dataset, de manera que puedan ser comprendidos y usados por el modelo en las etapas siguientes. Como ejemplo, los datos categoricos se codifican en valores numericos enteros o en variables dummy dependiendo de si sus valores tienen algun orden.
- Se divide el dataset en set de entrenamiento y set de test.
- Se divide el set de entrenamiento en set de entrenamiento y set de validación.
- Se **entrena el modelo** usando el set de entrenamiento.
- Se **evalua el modelo** usando el set de validación.
- Si la métrica de evaluación del modelo muestra que el modelo no cumple con la calidad requerida, se alteran los hiperparámetros del modelo y se lo vuelve a entrenar.
- Una vez que el modelo cumple con la calidad requerida segun las evaluaciones hechas contra el set de validación, se evalua el modelo contra el set de test y la métrica de calidad del modelo aquí obtenida es la que expresa que tan bien ha aprendido durante el proceso.
- Finalmente se obtiene el **modelo entrenado**.

Predicción

- Se quiere realizar una prección a partir de **una nueva instancia de dato** que es del mismo tipo que los usados para entrenar y evaluar el modelo (tiene los mismos features o atributos).
- Se realiza un **pre-proceso del dato**, el mismo que se realizó para los datos de entrenamiento.
- Se procede a darle esta nueva instancia de dato como entrada al **modelo**. La salida es la predicción hecha por el modelo entrenado.

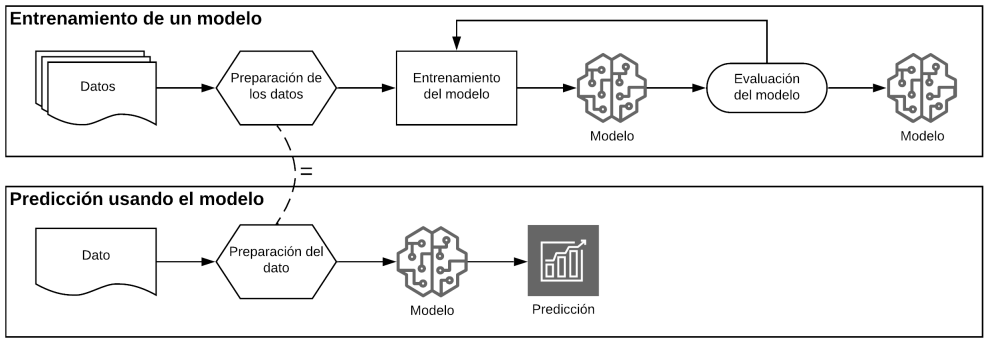

FIGURA 5.1: Entrenamiento, evaluación y uso de un modelo de aprendizaje automático.

5.1.3. Aprendizaje supervisado

En el campo del aprendizaje automático, a los algoritmos que requieren de un conjunto de entrenamiento con datos etiquetados se denominan supervisados. Analizando estos ejemplos de entrenamiento, el sistema infiere funciones que pueden ser usadas para aplicarlas a nueva información. Por el contrario, a los algoritmos que no son entrenados se los denomina no supervisados. Este grupo utiliza técnicas para encontrar patrones o grupos ocultos en los datos a analizar.

5.2. SMPC (Secure Multi-party Computation)

Una de las principales caracteristicas del presente trabajo es la posibilidad de realizar un cómputo entre varias partes de manera segura y privada, esto lleva el nombre de Secure Multi-party Computation.
Para ejemplificar, sean $P_1, ..., P_n$ diferentes partes dentro de un sistema distribuido, donde cada P_i es dueño de un input x_i y todas las partes están de acuerdo de usar una función f que toma n inputs. Su objetivo es computar $y = f(x_1, ..., x_n)$ cumpliendo, al mismo tiempo, las siguientes condiciones:

- Exactitud (Correctness): el valor correcto de y es computado.
- Privacidad (Privacy): y es la única nueva información que es publicada. Es decir, sus inputs siguen siendo secretos.

Existen muchos protocolos y métodos para construir un sistema que asegure SMPC. Algunos de estos son:

- Protocolos basados en Homomorphic Encryption
- Protocolos basados en zero proofs.

5.3. Cifrado Asimétrico

Es un método criptográfico que usa un par de claves para el envío de mensajes de manera segura. Ambas claves pertenecen a la persona que recibirá el dicho mensaje.

- **Clave pública:** que puede ser distribuida abiertamente
- **Clave privada:** que solo conoce el propietario.

Cualquiera puede cifrar un mensaje usando la clave pública, pero ese mensaje cifrado sólo puede descifrarse con la clave privada. Las claves se generan por medio de algoritmos basados en problemas matemáticos que no pueden ser resueltos por fuerza bruta en un tiempo razonable.

5.4. Differential Privacy

La privacidad diferencial [17] no es, en sí misma, una tecnología. Es una propiedad que describe algunos sistemas: una garantía matemática de que no se violará la privacidad uno o mas individuos si sus datos se utilizan para un determinado análisis. Un sistema que es diferencialmente privado permite realizar análisis sobre estos datos sensibles mientras los protege detrás de un velo de incertidumbre.
Otra definición [18] lo describe una promesa hecha por aquellos individuos o entes que recaban o curan datos, a los dueños originales de los mismos. Dicha promesa expresa que la privacidad del dueño original no será afectada de forma adversa al permitir que sus datos sean usados en algún estudio o análisis, no importa qué otros estudios, sets de datos o fuentes de información haya disponibles.

5.4.1. Local Differential Privacy

Con la privacidad local, no hay una parte confiable; cada persona es responsable de agregar ruido a sus propios datos antes de compartirlos. Cada persona es un curador a cargo de su propia base de datos privada. Por lo general, un sistema privado local involucra a una parte no confiable (el agregador) que recopila datos de un gran grupo de personas a la vez.

5.4.2. Global Differential Privacy

Los sistemas privados a globales son generalmente más precisos: todo el análisis se realiza en datos "limpios", y solo se necesita agregar una pequeña cantidad de ruido al final del proceso. Sin embargo, para que funcione la privacidad global, todos los involucrados deben confiar en el curador.
La privacidad local es un modelo más conservador y seguro. Bajo la privacidad local, cada dato individual es extremadamente ruidoso y no es muy útil por sí solo. Sin embargo, en números muy grandes, el ruido de los datos se puede filtrar, y los agregadores que recopilan suficientes datos privados locales pueden hacer análisis útiles sobre las tendencias en todo el conjunto de datos.

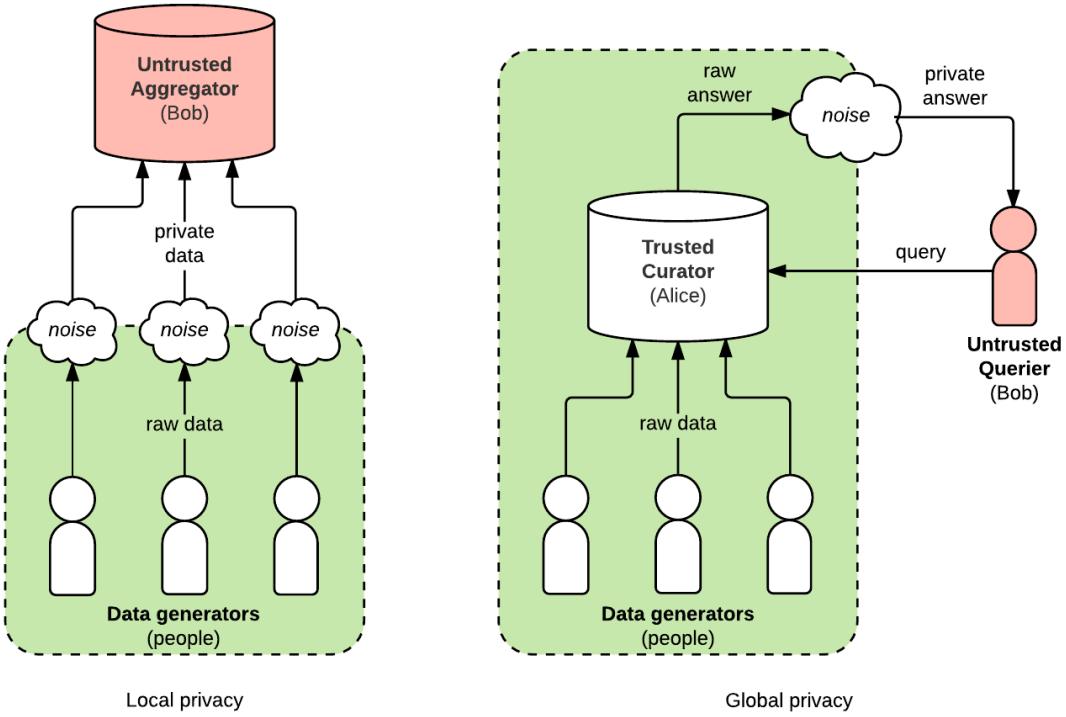

FIGURA 5.2: Local Differential Privacy vs. Global Differential Privacy

6 Fundamentos y herramientas

6.1. Modelo de Regresión Lineal [16]

Método estadístico que, en aprendizaje automático, se utiliza para predecir un valor objetivo (**y**) en base a variables independientes (los features o atributos, **X**).

$x_j^{(i)}$ = el valor del feature j en la i-ésima muestra de entrenamiento
$x^{(i)}$ = los features de la i-ésima muestra de entrenamiento
m = la cantidad de muestras de entrenamiento
n = el número de features

El modelo de regresión tiene la forma expresada en la ecuación 6.1

$$h_w(x) = w_o + w_1 x_1 + w_2 x_2 + w_3 x_3 + ... + w_n x_n \tag{6.1}$$

Función de costo

Expresa qué tanto error está cometiendo el modelo actualmente al predecir para cada una de las muestras del dataset.

Mean Squared Error (Error cuadrático medio)

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 \tag{6.2}$$

Gradient Descent

Algoritmo de optimización iterativo para hallar el mínimo de una función. Se utiliza, en el contexto del entrenamiento de un modelo de regresión lineal, para hallar los pesos **w** (parámetros del modelo de regresión lineal).

Repetir hasta que converja:

$$w_j = w_j - (\frac{\alpha}{m}\sum_{i=1}^{m}(h_w(x)^{(i)} - y^{(i)}) \cdot x_j^{(i)} paraj := 0...n \tag{6.3}$$

6.2. Homomorphic Encryption

[9] Método de encriptación asimétrico que permite realizar cálculos directamente sobre datos encriptados sin requerir acceso a la clave privada. El resultado de tal cálculo permanece en forma encriptada, y puede ser revelado posteriormente por el propietario de la clave.

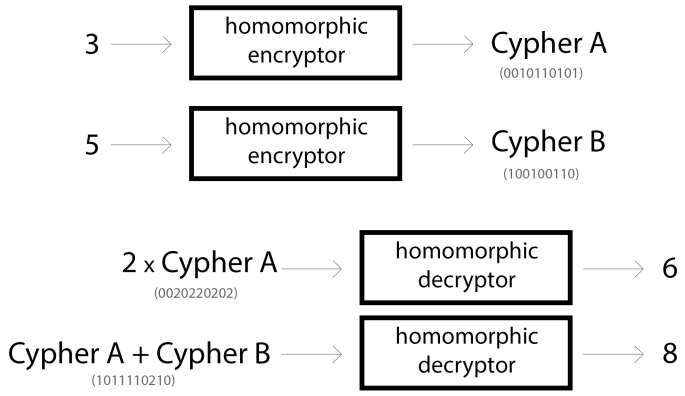

FIGURA 6.1: [19] Ejemplos de operaciones matemáticas realizadas sobre números encriptados por medio de Homomorphic Encryption.

Dentro de los cifrados con propiedades homomorficas, se pueden distinguir dos tipos [20] [21]: Fully Homomorphic Encryption y Partially Homomorphic Encryption.

Fully Homomophic Encryption

Un cifrado se considera totalmente homomórfico si exhibe tanto homomorfismo[1] aditivo como multiplicativo.

Partially Homomophic Encryption

Un cifrado es considerado parcialmente homomórfico si exhibe homomorfismo aditivo o multiplicativo, pero no ambos. Algunos ejemplos de estos sistemas son:

- **RSA:** homomorfismo multiplicativo.
- **ElGamal:** homomorfismo multiplicativo.
- **Paillier:** homomorfismo aditivo.

Paillier

El criptosistema utilizado en este trabajo es uno basado en Paillier [22]. Las propiedades homomorficas de un sistema de encriptación del tipo paillier son:

[1] **Homomorfismo en encriptación**: una operación realizada en un conjunto de datos cifrados tal que al desencriptar el resultado de la operación se obtiene el mismo resultado que si ésta se hubiera realizado sobre los datos sin cifrar.

- Números encriptados pueden ser encriptados por un escalar no encriptado.
- Números encriptados pueden ser sumados entre sí.
- Numeros encriptados pueden ser sumados a escalares no encriptados.

Se optó por éste cripto-sistema por la facilidad que brinda una de sus implementaciones (python-paillier) para la serialización y deserialización de números encriptados.

6.3. Federated Learning

El aprendizaje federado [2] [3] [4] [5] [6] [7] es un método de aprendizaje automático en el que el objetivo es entrenar a un modelo centralizado de alta calidad con datos de entrenamiento distribuidos entre un gran número de clientes, cada uno con conexiones de red poco confiables y relativamente lentas. Los algoritmos de aprendizaje para esta configuración funcionan de la siguiente manera: en cada iteración, cada cliente calcula de forma independiente una actualización del modelo actual en función de sus datos locales y comunica esta actualización a un servidor central, donde las actualizaciones enviadas por los clientes se agregan para calcular una nueva versión global del modelo.

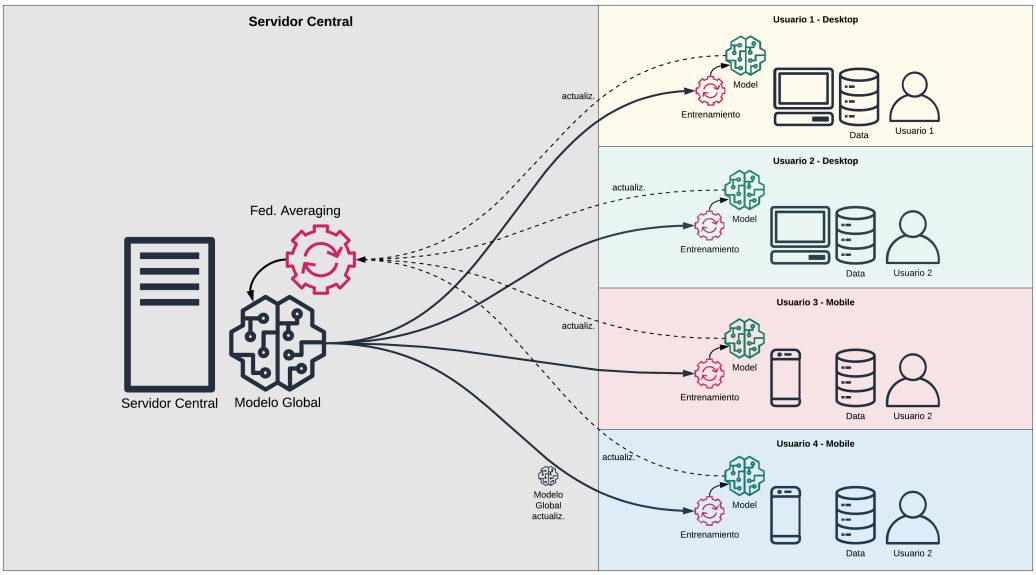

FIGURA 6.2: Esquema de entrenamiento de un modelo usando Federated Learning.

Caracteristicas

- Los clientes eligen si participan en el entrenamiento.
- El número de clientes disponible puede ser grande y variable.
- Incorpora preservación de la privacidad por medio de entrenamientos distribuidos.

- Los datos en general no están balanceados, ya que son específicos del cliente y auto-correlacionados.

6.4. Blockchain [1]

Blockchain es un registro contable abierto y distribuido que puede registrar transacciones entre dos partes eficientemente y de forma verificable y permanente.[23]

Este registro contable distribuido está tipicamente administrado por una red peer-to-peer que adhiere colectivamente a un protocolo para la comunicación entre nodos y validación de nuevas transacciones. Una Blockchain está compuesta por las siguientes partes:

Registro contable digital donde se anotan todas las transacciones agrupadas en bloques.
Las billeteras digitales o wallets, permiten a los usuarios realizar transacciones y manejar sus identidades digitales.
Los mineros, generan nuevos bloques (con las transacciones iniciadas por los usuarios de la red) resolviendo un problema matemático del protocolo de consenso. Reciben una recompensa por su trabajo.
Los nodos, almacenan una copia exacta del libro contable y validan nuevos bloques.

Se puede observar en la figura 6.3 una explicación del proceso que se dispara al realizarse una nueva transacción.

Caracteristicas

- **Pseudónimo**: ya que los usuarios se identifican por medio de direcciones que no pueden ser relacionadas directamente con ellos sin alguna información externa.
- **Irreversible e inmutable**: Los bloques que ya se hayan agregado a la cadena no pueden ser alterados ni borrados ya que cada uno tiene una referencia al anterior y todos los nodos validan constantemente la cadena y los nuevos bloques que se agregan a ella por medio de algoritmos de consenso.
- **Descentralizado**: Todos los participantes de la red tienen una copia de la cadena de bloques y no está controlada por una sola entidad por lo que no hay un punto único de fallo.
- **Consensuado**: Por medio de algoritmos de consenso que aprovechan la descentralización, se evitan fraudes y ataques.
- **Cronología:** Cada bloque tiene información del momento en el que fue creado y cual fue su bloque anterior.
- **Transparencia:** Todos los participantes de la red pueden ver el historial transacciones que se ejecutaron hasta el momento y los bloques que las contienen.

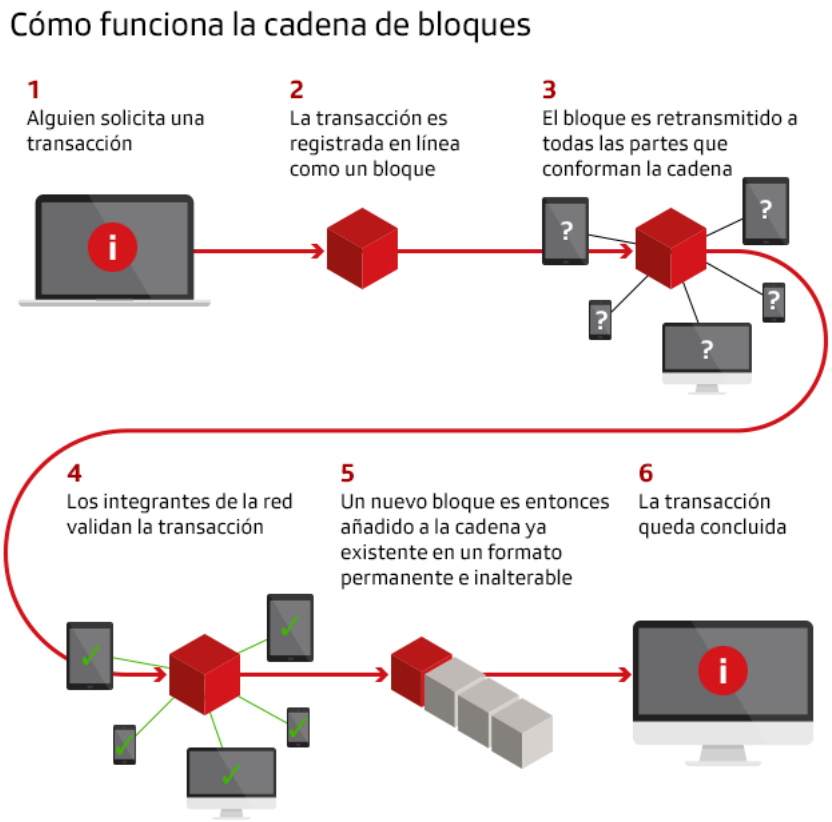

FIGURA 6.3: Cómo se procesa una nueva transacción en la Blockchain.
Fuente: Financial Times / PwC United States

6.4.1. Smart Contracts [24]

Aplicación informática que se programa para que ejecute acuerdos que hayan determinado dos o más partes de forma automática y sin necesidad de que un juez o una autoridad exija su cumplimiento. Queda almacenado en la Blockchain y por lo tanto cumple con las mismas características que se mencionaron antes. Un Smart Contract consta consta de las siguientes partes:

- **Address:** Una dirección donde se encuentra desplegado el contrato. Conociendo esta dirección podemos interactuar con él.
- **Código:** El contrato inteligente necesita que los acuerdos entre las partes esten escritos en forma de código. En general, se utiliza Solidity como lenguaje para programar estos contratos. Adicionalmente, al compilar el código del contrato, se genera el binario del contrato para ser desplegado y, lo que se llama ABI (Application Binary Interface), que no es más que la interfaz que deben conocer las aplicaciones que quieran comunicarse con dicho contrato.
- **Estado:** Los diferentes eventos o mensajes que le lleguen al contrato modificaran su estado interno (dependiendo de la lógica que se le haya programado).
- **Balance:** El contrato es capaz de almacenar dinero (en forma de Ether para la Blockchain Ethereum), y distribuirlo segun la lógica que se le haya programado.

7 Implementación

7.1. DeltaML: Interacción entre participantes

En la figura 7.1 se observa se observa un diagrama del sistema y las diferentes interacciones entre los participantes.

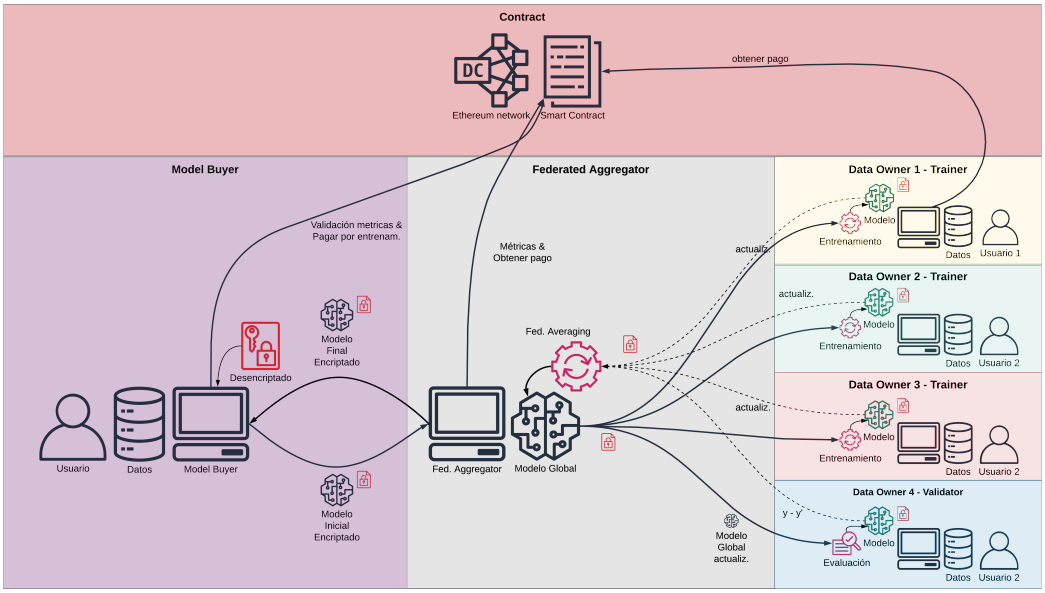

FIGURA 7.1: Interacciones de los módulos del sistema

1. El Model Buyer solicita a la plataforma el entrenamiento de un determinado modelo usando datos que cumplan ciertos requerimientos. El model buyer entonces envía un modelo encriptado sin entrenar al Federated Aggregator y inicializa una transacción de modelo en el smart contract.
2. Previamente, varios Data Owners se registraron al Federated Aggregator para estar disponibles para solicitudes de entrenamiento de modelos, así como también en el smart contract.
3. El Federated Aggregator les envía la solicitud de entrenamiento del modelo a cada uno de los Data Owners registrados así como también los requerimientos sobre los datos.
4. Cada Data Owner debe asegurarse de tener los datos requeridos, luego de lo cual, si así lo desean, aceptan la solicitud de participación en el entrenamiento.
5. El Federated Aggregator les asigna un rol de entrenador o validador a cada uno de los Data Owners, y les envía el modelo inicial encriptado.

6. Para el caso de un entrenamiento de un modelo de regresión lineal (el caso implemenentado en este trabajo), cada Data Owner con rol de entrenador realiza N pasos de Gradient Descent de manera local con sus datos privados, y envía el modelo actualizado encriptado, luego de ese proceso, al Federated Aggregator.
7. El Federated Aggregator realiza un promedio de todas las actualizaciones encriptadas, actualiza el modelo global que mantiene (también encriptado) y lo vuelve a enviar a los Data Owners para repetir el proceso.
8. El Federated Aggregator al finalizar el entrenamiento actualiza las métricas de calidad del modelo en el smart contract.
9. El model buyer recibe el modelo entrenado, y verifica las métricas de calidad en el smart contract usando el protocolo explicado en la figura 7.4.
10. El smart contract calcula las contribuciones de los participantes y realiza los pagos correspondientes por el entrenamiento y validacion del modelo.

7.2. Data Owner

El Data Owner es aquel que ejecuta en su computador el cliente web por medio del cual puede decidir en qué solicitudes de entrenamiento participar. Tiene en su poder uno o mas datasets.
En su rol de de Data Owner publica al Federated Aggregator las caracteristicas de los datasets que posee (cantidad de registros, columnas, tipo de los datos de cada columna, cantidad de nulos por columna, etc) y recompensa monetaria mínima que está dispuesto a percibir por el entrenamiento de cada dataset.

Capacidades

- Elige participar en el entrenamiento de un modelo.
- Aporta sus datos para, de manera local y sin poder ver el modelo.
- Entrenar el modelo solicitado por el Model Buyer.
- Calcular diferencia entre predicciones del modelo y lo esperado.

Limitaciones

No puede:

- Ver los parámetros del modelo en entrenamiento, trabaja con el modelo encriptado.
- Alterar el cálculo de las métricas de calidad o su contribución para ganar más dinero por el entrenamiento.

Roles

- **Trainer:** Usa sus datos privados y su poder de cómputo para entrenar modelos.
- **Validator:** Usa sus datos privados y su poder de cómputo para calcular diferencias entre lo esperado y las predicciones.

Los Data Owners se dividen en Trainers y Validators en proporciones de 70 % y 30 % respectivamente tal como se haría con un dataset para su entrenamiento y evaluación. Queda para futuros desarrollos hacer esta división emulando un K-Fold Cross-validation.

7.3. Model Buyer

El Model Buyer interactua con el sistema por medio del marketplace, publicando una solicitud de entrenamiento para un cierto modelo con ciertas caracteristicas. Al crear la solicitud envía un modelo inicial encriptado con Homomorphic Encryption al Federated Aggregator junto con las carácteristicas que debe cumplir el dataset para entrenar el modelo. Al mismo tiempo, inicializa en un smart contract desplegado en la red Ethereum, la transacción del correspondiente al entrenamiento del modelo y envía el monto que destina al proceso de mejorar el modelo inicial, las metricas de calidad del modelo inicial y los participantes involucrados.

Capacidades

- Solicita el entrenamiento de un modelo.
- Paga por el modelo entrenado.
- Desencripta el modelo una vez entrenado.
- Valida métricas de calidad del modelo.

Limitaciones

No puede:

- Ni ver ni robar los datos con los que se entrenan los modelos.
- Alterar las métricas de calidad del modelo.
- Realizar fraude en el pago por el modelo.

7.4. Federated Aggregator

Este participante del sistema verifica periodicamente la existencia de nuevas solicitudes de entrenamiento de modelos consultando al smart contract. De encontrar una nueva solicitud, obtiene del smart contract el hash que corresponde al directorio donde se almacenan los recursos referentes a esta solicitud en la base de datos, y comienza a verificar si hay nuevas mejoras publicadas por los Trainers. De existir

nuevas mejoras, el Federated Aggregator realiza un promedio de las mejoras al modelo y el modelo inicial, luego calcula la métrica de calidad del modelo nuevamente y la envia al smart contract.

Capacidades

- Asigna roles a los Data Owners que participan en un entrenamiento.
- Agrega de manera segura las actualiz. al modelo provenientes de cada Data Owner.
- Envía el modelo actualiz. encriptado a los Data Owners para que continúen el entrenamiento.
- Calcula las métricas de calidad del modelo.

Limitaciones

No puede:

- Robar el modelo en entrenamiento, está encriptado.
- Alterar el cálculo de las métricas de calidad ni las contribuciones de cada Data Owner (las valida el Model Buyer).

7.5. Contract

El smart contract es el encargado de regular el mercado, recibe solicitudes provenientes del Model Buyer y las vincula con los Data Owners que las acepten y al modelo en entrenamiento. Al mismo tiempo, permite al Federated Aggregator guardar métricas métricas de calidad del modelo y que sean validadas por el Model Buyer. Se define aquí tambien, la recompensa mínima que recibirán todos los participantes del entrenamiento (monto fijo) y el monto variable de acuerdo a la contribución a la mejora de la calidad del modelo.
El carácter descentralizado del smart contract permite que los cálculos y pagos se realicen sin necesitar de una tercera parte en la que confiar para esta parte del proceso.

Capacidades

- Restringe el acceso a sus funcionalidades según el rol del participante.
- Calcula contribuciones de cada Data Owner.
- Paga a cada participante por su trabajo.
- Devuelve dinero al Model Buyer si tuviera que hacerlo.
- Valida métricas de calidad del modelo.

Limitaciones

No puede:

- Robar el modelo en entrenamiento (no tiene visión del mismo).
- Alterar el cálculo de las métricas de calidad.
- Alterar el cálculo de las contribuciones.

7.6. Protocolos

Obtención de recursos para entrenamiento

Este protocolo se encarga de la registración de los Data Owners disponibles al servicio del Federated Aggregator, de la inicialización de la solicitud de entrenamiento del modelo, y la aceptación o declinación de dicha solicitud por parte de cada Data Owner.

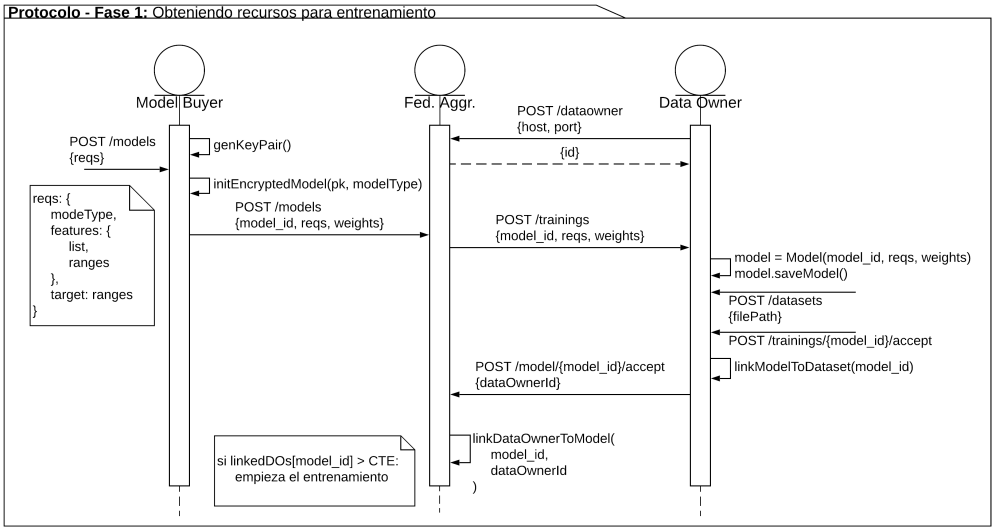

FIGURA 7.2: Protocolo de inicialización del proceso de entrenamiento.

Entrenamiento del modelo encriptado

El protocolo de entrenamiento encriptado, basado en la técnica de Federated Learning, define como se dará éste entrenamiento de manera distribuida, con el modelo encriptado en todo momento y, con cada Data Owner brindando sus datos locales y procesamiento.

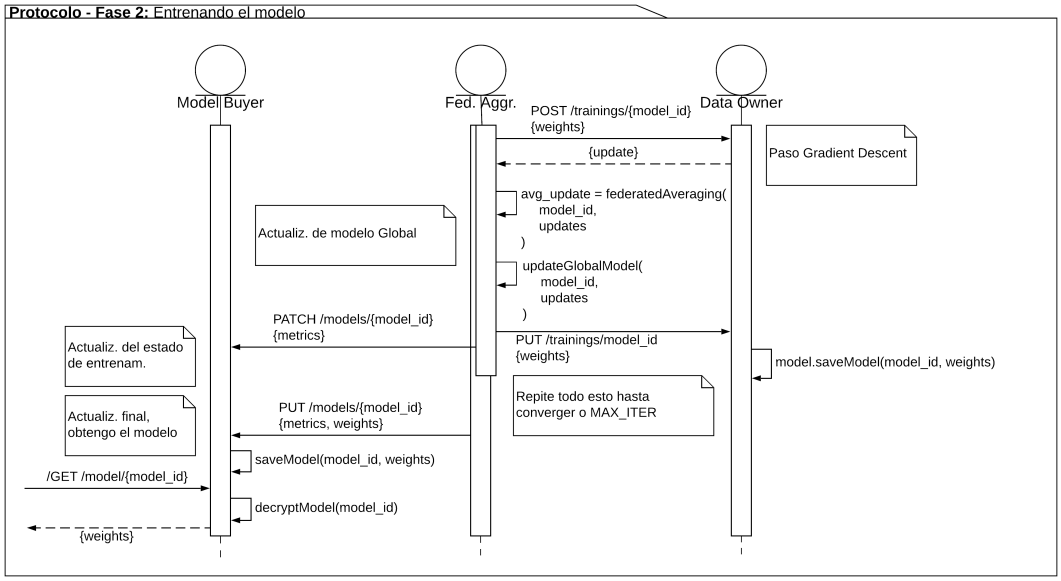

FIGURA 7.3: Protocolo de entrenamiento de un modelo de manera distribuida y segura.

Calculo de métricas seguro

El protocolo de calculo de métricas seguro fue creado debido a la incapacidad de los Data Owners de calcular la métrica de calidad del modelo por ellos mismos. Ésta limitación se da debido a que la biblitoca de Homomorphic Encryption utilizada no permite realizar productos entre números encriptados (y se necesita elevar un número encriptado al cuadrado para el calculo del MSE). Por lo que se ideó un protocolo mediante el cual, se puede calcular el MSE sin que sea el Model Buyer quien lo haga (ya que podría alterar el calculo para obtener un beneficio), pero a la vez, dandole la posibilidad de validar dicho calculo. El protocolo hace uso de una mecanica utilizada en Global Differential Privacy y en algunos esquemas de SMPC, la adición de ruido.ª la información que se envía a una parte en la que no se confía plenamente.

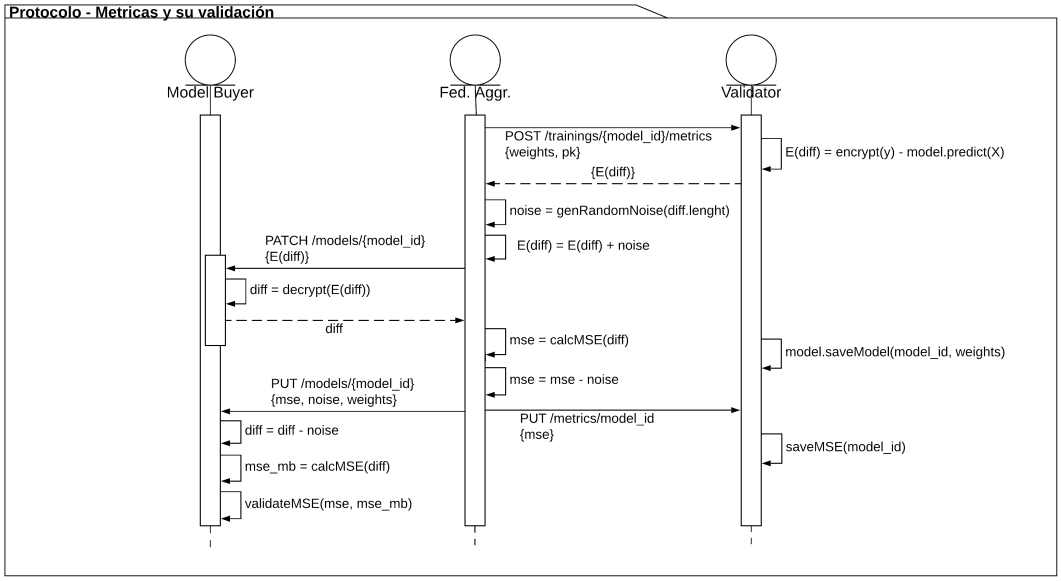

FIGURA 7.4: Esquema de calculo y validación de métricas seguro.

7.7. Imágenes de la plataforma DeltaML

La figura 7.5 muestra la pantalla de Inicio de sesión que tanto el Data Owner como el Model Buyer ven antes de ingresar a la plataforma ingresando sus credenciales.

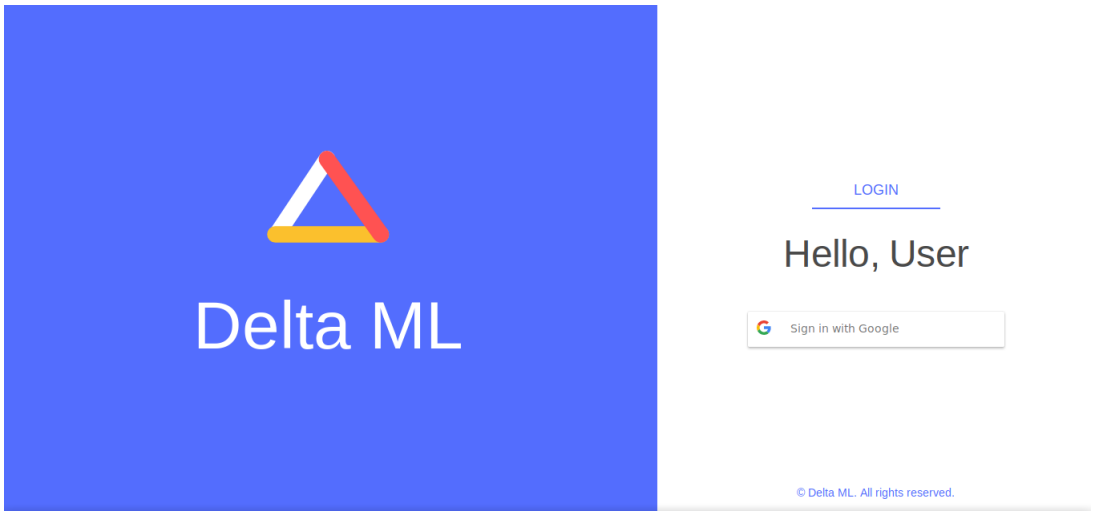

FIGURA 7.5: Pantalla de Login de la plataforma.

La figura 7.6 muestra la pantalla donde, el Data Owner, puede ver todos los pedidos o solicitudes de entrenamiento de modelos que le han llegado. Estos pueden estar en los siguientes estados:

- **WAITING:** El Data Owner no tiene aún el dataset requerido para aceptar el entrenamiento.
- **READY:** El Data Owner tiene el dataset requerido pero no ha aceptado el entrenamiento aún.

- **INITIATED:** El Data Owner ha aceptado participar en el entrenamiento, pero el proceso no ha comenzado debido a que todavía no se ha alcanzado el número mínimo requerido de Data Owners que acepten la solicitud.
- **IN PROGRESS:** El entrenamiento del modelo ha comenzado.
- **FINISHED:** El entrenamiento del modelo finalizó.

El Model Buyer tiene una pantalla similar pero para las solicitudes que realizó.

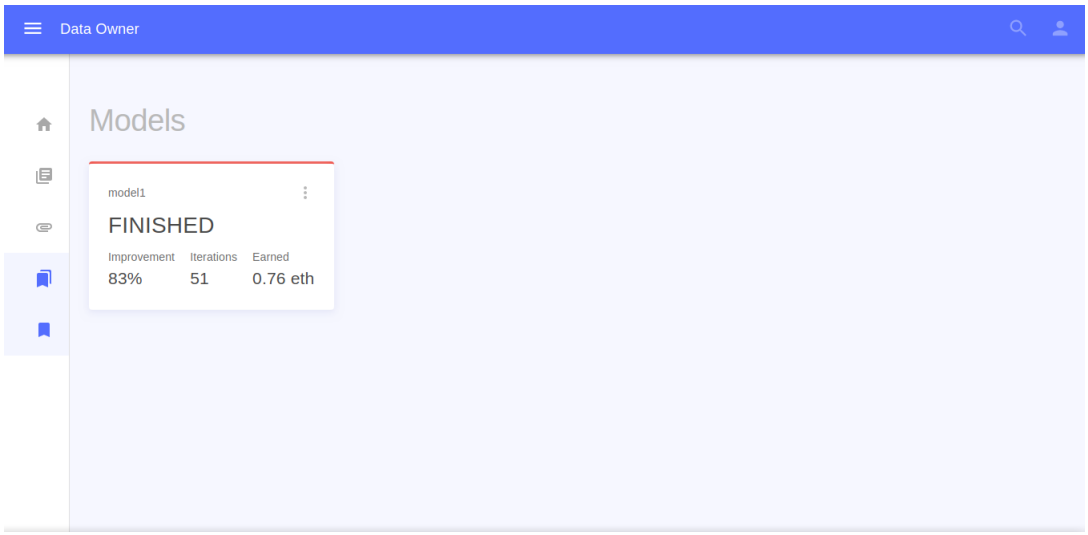

FIGURA 7.6: Pantalla de lista entrenamientos solicitados al Data Owner.

La figura 7.7 muestra la pantalla de detalle de progreso de un modelo en entrenamiento en el Model Buyer. En esta pantalla se puede observar el estado actual del modelo (los listados anteriormente), cuánto ha mejorado el modelo hasta ahora (expresado en porcentaje y a través de la métrica de calidad MSE) y cuánto se está paagando por el proceso hasta ahora. Además se observa la progresión de la mejora del modelo en forma de grafico, donde la curva descendente es el MSE (Mean Squared Error) decreciendo hasta el fin del entrenamiento. Y finalmente, se muestran las contribuciones que cada Data Owner entrenador realizaron al entrenamiento (expresado en porcentaje) y cuánto se le paga a cada uno por dicho trabajo.

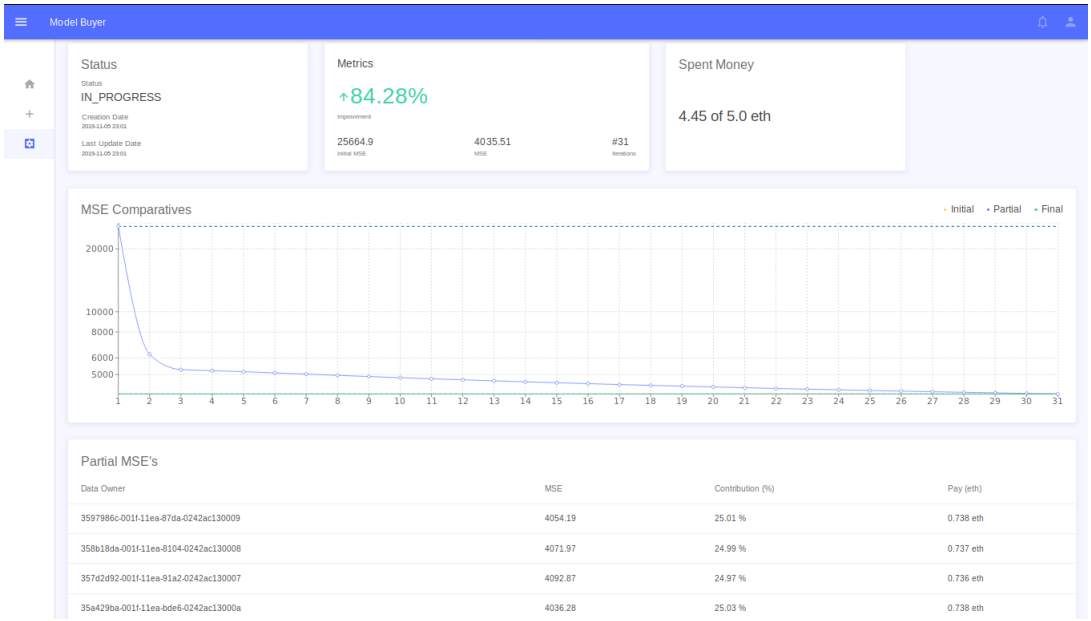

FIGURA 7.7: Pantalla de detalle del progreso del entrenamiento de un modelo en el ModelBuyer.

8 Resultados

En el presente trabajo se desarrolló una plataforma capaz de entrenar modelos de manera descentralizada manteniendo de privacidad de los datos de los participantes involucrados. A su vez, se procuró que los modelos en entrenamiento no pudieran ser robados y ni tampoco las métricas alteradas. Todo esto se logró usando Homomorphic Encryption para generar un equema SMPC (Secure Multy-Party Computation), y Global Differential Privacy para el protocolo de calculo de métricas seguro.

Adicionalmente, se comparó la calidad del modelo entrenado usando el esquema implementado en el presente trabajo (con **5 Data Owners**) y usando una regresión lineal tradicional. Para realizar ésta comparación se utilizaron los mismos datasets e hiperparametros, los cuales son:

- **Dataset:** Diabetes[25]
- **Cantidad de iteraciones máxima:** 100
- **Paso (Gradient Descent):** 1.5
- **MSE Inicial (Modelo sin entrenar):** $\approx$ 27000

Y los resultados obtenidos fueron:

MSE (Mean Squared Error)	
DeltaML	**Regresión Lineal local**
3311.84	3167.81

Por lo que, además de lo logrado en materia de privacidad y seguridad, el entrenamiento de los modelos por medio del esquema implementado continuó arrojando resultados similares a los de un entrenamiento tradicional con un dataset local.

9 Trabajo futuro

Dada la longitud del alcance del presente trabajo, algunas tareas fueron relegadas al no ser esenciales al objetivo planteado. Se detallan a continuación, algunos de los puntos que contemplamos como trabajo futuro para seguir profundizando sobre la temática de machine learning seguro y privado, o para continuar con el desarrollo de DeltaML son:

Refactorizar el esquema para extraer un framework de Federated Leaning seguro y extensible tal que:

- El método de comunicación entre los participantes sea configurable (local, http, etc)
- El mecanismo de SMPC sea extensible y se pueda cambiar por otro de manera simple.
- Permita agregar nuevos tipos de modelo (regresión logística, redes neuronales, etc).
- Pueda ser usado con varias bibliotecas de Machine Learning (como Scikit-learn, Spark ML o incluso Tensorflow).
- Investigar otros métodos de SMPC y Homomorphic Encryption, y comparar como impacta uno u otro en la performance durante el entrenamiento de un modelo.
- Migrar Data Owners a aplicaciones mobile.
- Experimentar con otros frameworks (nuevos) de Federated Learning privado, tales como: PySift de OpenMined, TF-Encrypted de DropoutLabs, CrypTFlow de Microsoft.

10 Conclusiones finales

En este trabajo se propuso un esquema de entrenamiento de modelos de aprendizaje automático que no requieran el acceso físico a los datos por parte de quien desee entrenar dicho modelo, permitiendo preservar la privacidad de datos que quizas sean sensibles para los usuarios. A partir de esta propuesta se implementó DeltaML, plataforma que cumple con lo mencionado y, además, evita la apropiación del modelo en entrenamiento por cualquier otro participante de la misma. Como parte de esta solución se requería de una forma de pago a los distintos participantes que hubieran contribuido al entrenamiento de un modelo, sin embargo, usar un medio de pago que requiriera confiar en terceros, que centralizara esta dependencia o hiciera fácil algun tipo de fraude iba en contra del espiritú del trabajo. Por lo que se decidió optar por la plataforma Ethereum que por medio de sus smart contract y el carácter descentralizado de su tecnología base brindaba todas las medidas de seguridad que un trabajo de estas caracteristicas requería.

En lo referente a los modelos de aprendizaje automático disponibles para entrenar por medio de DeltaML, por el momento está limitado a modelos de Regresión Lineal. Se optó por esta restricción para no extender el alcance del trabajo más allá de lo estipulado para un Trabajo Profesional de Ingeniería. Sin embargo, el proyecto es facilmente extendible para modelos de clasificación mediante Regresión Logística. El único punto a tener en cuenta en ese caso es que se deberá usar una aproximación de la función Sigmoide (si se optará por ésta) para la Regresión Logística, ya que la librería de Homomorphic Encryption usada en el proyecto no permite operaciones no lineales entre números encriptados.

Con respecto a la performance, existe una diferencia considerable entre los tiempos entrenando un modelo usando DeltaML y usando un técnica tradicional. La plataforma desarrollada en el presente trabajo requiere de alrededor de 15 minutos para entrenar el mismo modelo que de otra manera se logra entrenar en cuestión de segundos. Esta diferencia tan grande se debe a que todos los cálculos se dan sobre datos que se encuentran encriptados. Una mejora propuesta para disminuir esta baja de performance, es cambiar el mecanismo de SMPC actual (que usa Homomorphic Encryption) a uno como SPDZ [26] [27] [28] [29].

Finalmente, la calidad de los modelos entrenados con la plataforma se mantiene en comparación a la de un modelo entrenado con una Regresión Lineal con un dataset local. Por lo que al usar la plataforma se gana en protección de la privacidad y no se ve afectado el resultado final.

11 Referencias

[1] Satoshi Nakamoto. «Bitcoin: A Peer-to-Peer Electronic Cash System». En: (). URL: https://bitcoin.org/bitcoin.pdf.

[2] Jakub Konečný y col. «Federated Optimization: Distributed Machine Learning for On-Device Intelligence». En: (2016). URL: https://ai.google/research/pubs/pub45630.

[3] Keith Bonawitz y col. «Practical Secure Aggregation for Federated Learning on User-Held Data». En: (2016). URL: https://ai.google/research/pubs/pub45808.

[4] Jakub Konečný y col. «Federated Learning: Strategies for Improving Communication Efficiency». En: (2016). URL: https://ai.google/research/pubs/pub45648.

[5] Andrew Hard y col. «Federated Learning for mobile keyboard prediction». En: (2018). URL: https://ai.google/research/pubs/pub47586.

[6] Sebastian Caldas y col. «Expanding the Reach of Federated Learning by Reducing Client Resource Requirements». En: (2018). URL: https://ai.google/research/pubs/pub47774.

[7] Timothy Yang y col. «Applied Federated Learning: Improving Google Keyboard Query Suggestions». En: (2018). URL: https://ai.google/research/pubs/pub47655.

[8] *OpenMined*. Markeplace de modelos de deep learning. URL: https://www.openmined.org/.

[9] Ciara Moore y col. «Practical homomorphic encryption: A survey». En: (2014). URL: https://ieeexplore.ieee.org/abstract/document/6865753.

[10] *DML*. Marketplace de modelos de machine learning. URL: https://decentralizedml.com/.

[11] *NumerAI*. Fondo de inversion que genera predicción con modelos de ML crowdsourced. URL: https://numer.ai/.

[12] *Augur*. Protocolo de predicción de mercados en la Blockchain. URL: https://www.augur.net/.

[13] *Golem Network*. Supercomputadora descentralizada. URL: https://golem.network/.

[14] *OceanProtocol*. Ecosistema destinado a compartir datos como activos y ML como servicios de manera descentralizada. URL: https://oceanprotocol.com/.

[15] *Wibson*. Marketplace de datos descentralizado. URL: https://wibson.org/.

[16] Andrew Ng. *Curso de Aprendizaje Automático*. URL: https://www.coursera.org/learn/machine-learning.

[17] *Understanding differential privacy and why it matters for digital rights*. URL: https://www.accessnow.org/understanding-differential-privacy-matters-digital-rights/.

[18] Cynthia Dwork y Aaron Roth. «The Algorithmic Foundationsof Differential Privacy». En: (2014). URL: https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf.

[19] *Building Safe A.I.* Un Tutorial para Deep Learning encriptado, por Andrew Trask. URL: https://iamtrask.github.io/2017/03/17/safe-ai/.

[20] Liam Morris. «Analysis of Partially and Fully Homomorphic Encryption». En: (2013). URL: http://gauss.ececs.uc.edu/Courses/c6056/pdf/homo-outline.pdf.

[21] Hidayet Aksu Abbas Acar y A. Selcuk Uluagac. «A Survey on Homomorphic Encryption Schemes: Theory and Implementation». En: (2017). URL: https://arxiv.org/pdf/1704.03578.pdf.

[22] *Paillier crypto system*. Sistema de encriptación que es parcialmente homomórfico. URL: https://en.wikipedia.org/wiki/Paillier_cryptosystem.

[23] Karim R. Iansiti Marco; Lakhani. *The Truth About Blockchain*. URL: https://hbr.org/2017/01/the-truth-about-blockchain.

[24] «An Overview of Smart Contract: Architecture, Applications, and Future Trends». En: (2018). URL: https://ieeexplore.ieee.org/abstract/document/8500488.

[25] *Diabetes Dataset*. Dataset usado para evaluar proceso de entrenamiento privado de DeltaML. URL: http://scikit-learn.org/stable/datasets/index.html#diabetes-dataset.

[26] Nigel Smart Ivan Damgard Valerio Pastro y Sarah Zakarias. «Multiparty Computation from Somewhat Homomorphic Encryption». En: (2011). URL: https://eprint.iacr.org/2011/535.pdf.

[27] «Practical Covertly Secure MPC for Dishonest Majority – or: Breaking theSPDZ Limits». En: (2012). URL: https://eprint.iacr.org/2012/642.pdf.

[28] Morten Dahl. *The SPDZ Protocol, Part 1: Secure Computation using Precomputed Triples*. URL: https://mortendahl.github.io/2017/09/03/the-spdz-protocol-part1.

[29] Valerio Pastro Valerie Chen y Mariana Raykova. «Secure Computation for Machine Learning With SPDZ». En: (2019). URL: https://arxiv.org/pdf/1901.00329.pdf.

[30] *python-paillier*. Biblioteca de encriptación homomórfica basada en Paillier. URL: https://python-paillier.readthedocs.io/en/develop.

[31] Buterin Vitalik. *Ethereum: A secure descentralized generalized transaction ledger*. URL: https://www.ethereum.org.

[32] *Solidity*. Lenguaje de programación de smart contracts. URL: https://solidity.readthedocs.io/en/v0.5.3/.

[33] *Truffle*. Framework de desarrollo de smart contracts. URL: https://truffleframework.com/truffle.

[34] *Ganache*. Testnet local para Ethereum Blockchain. URL: https://truffleframework.com/ganache.

[35] *Python*. Lenguaje de programación interpretado. URL: https://www.python.org/.

[36] *Flask*. Framework para aplicaciones web Python. URL: https://www.fullstackpython.com/flask.html.

[37] *React*. Biblioteca de javascript para desarrollo de clientes web. URL: https://reactjs.org/.

[38] *Docker*. Entorno de virtualizacion. URL: https://www.docker.com/.

[39] *Docker Compose*. Orquestador de contenedores Docker. URL: https://docs.docker.com/compose/.

12 Glosario

- **Machine Learning**: (en castellano Aprendizaje Automático) es la rama del área de la inteligencia artificial centrada en el estudio y construcción de sistemas capaces de aprender de los datos, identificar patrones y realizar decisiones con mínima intervención humana.
- **Deep Learning**: (o Aprendizaje Profundo) es un subconjunto del área de Machine Learning que intenta generar modelos que constan de arquitecturas compuestas de transformaciones no lineales múltiples. Se utilizan, en general, para resolver problemas de gran complejidad computacional por medio de aproximaciones logradas a través del entrenamiento de un modelo con las caracteristicas antes nombradas a partir de una cantidad enorme de datos.
- **Blockchain**: o DLT (Descentralized Ledger Technology), es una tecnología que consta de un registro contable distribuido en una red de nodos. Las características principales de Blockchain son: inmutabilidad de la historia de las transacciones realizadas, resistencia a fraudes utilizando algoritmos de consenso entre los nodos de la red, transparencia (todos los nodos pueden ver el historial de transacciones), resistente a ataques (mediante la descentralización de los datos y el poder de computo combinado de todos los nodos participantes), y pseudo-anonimidad de los participantes de la red (se identifican con una clave que lleva el nombre de wallet, pero a priori no hay forma de relacionar dicha wallet con su dueño).
- **Smart Contract**: aplicaciones que se ejecutan sobre la red de nodos de una blockchain exactamente como fueron programadas sin ninguna posibilidad de downtime, censura, fraude o interferencia de una tercera parte.
- **Differential Privacy**: es una técnica estadística que trata de maximizar la exactitud de la información que se desea obtener de una base de datos, mientras se minimiza el impacto en la privacidad de las personas cuyos datos se encuentran en dicha base.
- **Multi-party computation**: es una rama de la criptografía, la cual tiene por objetivo la creación de métodos para que diferentes partes puedan en conjunto realizar el computo de una función sobre sus datos a la vez que mantienen la privacidad de ellos.
- **Homomorphic Encryption**: es un método de encriptación que difiere de los métodos típicos en que permite realizar cálculos directamente en datos encriptados sin requerir acceso a la clave. El resultado de tal cálculo permanece en forma encriptada, y puede ser revelado posteriormente por el propietario de la clave.

13 Tenologías utilizadas

Homomorphic Encryption

Python-paillier [30]

Biblioteca de python para Encriptación Homomórfica Parcial (Partially Homomorphic Encryption) badada en el sistema llamado Paillier [22]. Se optó por esta biblioteca por las facilidades para serialización y deserialización de números encriptados que brinda.

Blockchain

Ethereum network [31]

Plataforma descentralizada que ejecuta contratos inteligentes (smart contracts): aplicaciones que se ejecutan exactamente como fueron programadas sin ninguna posibilidad de downtime, censura, fraude o interferencia de una tercera parte.

Solidity [32]

Lenguaje de alto nivel, orientado a objetos para implementar smart contracts. Los smart contracts son programas que gobiernan el comportamiento de cuentas dentro del ecosistema Ethereum.

Truffle [33]

Framework de desarrollo de smart contracts para blockchains que utilizan la Máquina Virtual Ethereum (EVM).

Ganache [34]

Blockchain personal para el desarrollo en Ethereum (testnet local) que se puede utilizar para implementar contratos, desarrollar sus aplicaciones y ejecutar pruebas.

API

Python [35]

Lenguaje de alto nivel y tipado dinámico con soporte de paradigma de objetos que se utilizó para el desarrollo de los módulos de Federated Learning y Encriptación. Se eligió este lenguaje por su uso simple, por las herramientas desarrolladas para las áreas de Machine Learning y análisis de datos, y su gran adopción en el mundo del desarrollo y ciencias de datos, lo que reduce, también, dificultades en la resolución de problemas que puedan surgir durante el proyecto.

Flask [36]

Flask es un framework minimalista escrito en Python que te permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código.

UI

React [37]

Biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

General

Docker [38]

Automatiza el despliegue de aplicaciones dentro de contenedores, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Docker Compose [39]

Es una herramienta para definir y ejecutar aplicaciones que consisten en varios contendores de Docker.

14 Código fuente

- **Model Buyer API:** https://github.com/DeltaML/model-buyer
- **Model Buyer UI:** https://github.com/DeltaML/model-buyer-ui
- **Data Owner API:** https://github.com/DeltaML/data-owner
- **Data Owner UI:** https://github.com/DeltaML/data-owner
- **Federated Aggregator:** https://github.com/DeltaML/federated-aggregator
- **Contract:** https://github.com/DeltaML/contract
- **Libreria commons:** https://github.com/DeltaML/commons
- **docker-compose:** https://github.com/DeltaML/docker-compose
- **Scripts:** https://github.com/DeltaML/testing