Universidad ORT Uruguay
Facultad de Ingeniería

Implementación de las prácticas de MLOps para PATE

Entregado como requisito para la obtención del título de Master en Big Data

Javier Ramas – 107758

Antonia Rodríguez – 260101

Sebastián Zanotta – 252045

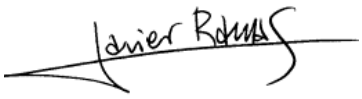
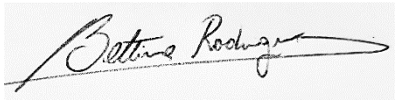
Tutor: Mikaela Pisani

2022

Declaración de autoría

Nosotros, Javier Ramas, Antonia Rodríguez y Sebastián Zanotta, declaramos que el trabajo que se presenta en esta obra es de nuestra autoría. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el Proyecto Final del Master en Big Data;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.


Javier Ramas | Antonia Rodríguez | Sebastián Zanotta

28/03/2022

Agradecimientos

Queremos agradecer a todas las personas que nos brindaron su apoyo durante el transcurso de la Maestría, familiares, amigos, docentes, compañeros de clase y tutores de este trabajo.

En particular, agradecemos a Sergio Yovine y Mikaela Pisani, quienes nos acompañaron a lo largo del proceso de creación de este trabajo, aportado sus conocimientos y desafiándonos para poder brindar nuestro mayor esfuerzo.

Abstract

El presente trabajo surge como una investigación con el fin de validar la aplicación de *MLOps* en el contexto de PATE. Para ello se ha llevado a cabo un estudio del estado del arte y se han analizado herramientas de *MLOps*, posibles escenarios de implementación junto a su gobernanza y herramientas de intercambio de información de manera segura entre los actores del sistema (*PySyft*).

Como resultado de este trabajo, se logró el análisis de cuatro herramientas de *MLOps* y la descripción detallada de una de ellas para el contexto definido. El trabajo permitió validar que todas las herramientas poseen pocas diferencias entre si y podrían ser utilizadas.

Se concluyó que *PySyft* en su versión 0.6. permite el intercambio de información de manera segura entre los actores del sistema.

En base a la experiencia obtenida, podemos asegurar que se podría implementar la herramienta de PATE con *PySyft* y utilizar *MLOps* para optimizar tiempos y recursos al momento de hacer un *deploy*.

Se plantea como línea para continuar el trabajo realizado, generar un entorno que contemple a todos los participantes incorporando PATE.

Palabras clave

***MLOps, PATE, PySyft, PyGrid, MLflow,* Gobernanza*, OpenMined, Federated Learning, Differential Privacy, Homomorphic Encryption, Machine Learning*.**

Índice

Índice de tablas

Índice de ilustraciones

GLOSARIO

Ensamble [1]: es un conjunto de modelos de *machine learning*. Cada modelo produce una predicción diferente. Las predicciones de los distintos modelos se combinan para obtener una única predicción. La ventaja que obtenemos al combinar modelos diferentes es que como cada modelo funciona de forma diferente, sus errores tienden a compensarse. Esto resulta en un mejor error de generalización. Hay varias formas de construir estos ensembles:

- votación por mayoría
- *bagging*
- *boosting*
- *stacking*

Arg_max [2]**:** En matemáticas, los argumentos de los máximos (abreviados *arg_max* o *argmax*) son los puntos, o elementos, del dominio de alguna función en la que se maximizan los valores de la función. A diferencia de los máximos globales, que se refieren a las salidas más grandes de una función, *argmax* se refiere a las entradas, o argumentos, en los que las salidas de la función son lo más grandes posible.

Ɛ (*Épsilon*) [3]: El parámetro de pérdida de privacidad, determina la cantidad de ruido a introducir. El *épsilon* se puede derivar de la distribución de probabilidad, conocida como Distribución de Laplace, que determina cuánta desviación hay en el cálculo si uno de los atributos de los datos se ha excluido del conjunto de datos. Cuanto menor sea el *Epsilon*, menor será la desviación en los cálculos en los que los datos de los usuarios debían eliminarse del conjunto de datos. O, los valores más altos de *Epsilon* representan resultados más precisos, menos privados y *Epsilon* más bajo proporciona resultados aleatorios altos que no permitirán a los atacantes aprender mucho en absoluto. Por lo tanto, el pequeño valor de *Epsilon* conducirá a una mayor preservación de los datos incluso si los resultados del cálculo son poco precisos. Sin embargo, aún no se ha determinado un valor óptimo de *Epsilon* que pueda garantizar / cumplir con el nivel necesario de protección y precisión de datos. Dependiendo de la compensación entre privacidad y precisión que los usuarios deben generar, la privacidad diferencial se puede adoptar a nivel mundial.

***Teachers*:** múltiples modelos entrenados con conjuntos de datos disjuntos, como registros de diferentes subconjuntos de usuarios. Debido a que se basan directamente en datos confidenciales, estos modelos no se publican, sino que se utilizan como "*Teachers*".

***Students*:** Modelos entrenados con datos privados sin etiquetar y que son guiados en su entrenamiento por el ensamble de *Teachers*.

***Trusted Curator*:** una entidad en la que los propietarios de los datos confían para almacenarlos y manipularlos.

***Aggregate label*:** Es el resultado del aprendizaje del Ensamble, en el cual se utilizan múltiples algoritmos para obtener un mejor rendimiento predictivo que el que se podría obtener solo con cualquiera de los algoritmos de aprendizajes constituyentes.

1. Introducción

En el contexto de Big Data uno de los desafíos más importantes está vinculado a la protección de los datos, lo que afecta la posibilidad de compartir información al momento de entrenar modelos de ML.

Differential Privacy (DP) [4] es un sistema para compartir públicamente información sobre un conjunto de datos, al describir los patrones de grupos que lo componen mientras se retiene información sobre las personas del *dataset*. La idea detrás de la privacidad diferencial es que, si el efecto de hacer una única sustitución arbitraria en la base de datos es lo suficientemente pequeño, el resultado de la consulta no se puede usar para inferir mucho sobre un solo individuo y, por lo tanto, proporciona privacidad.

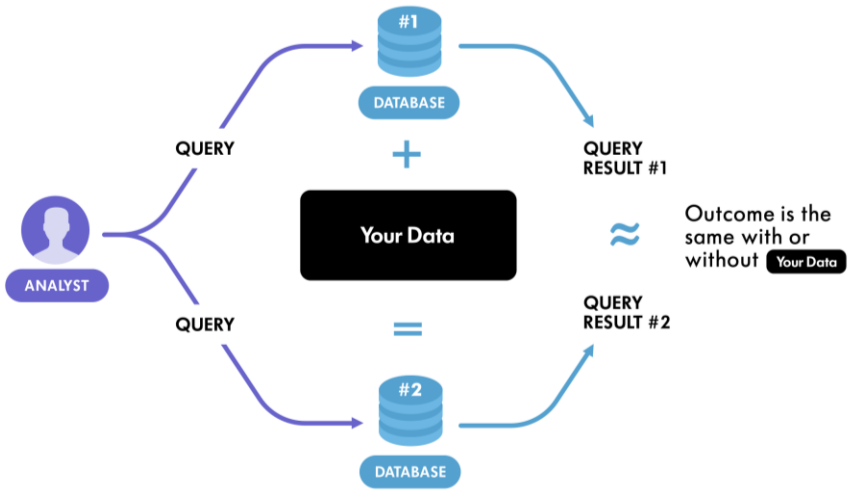


Figura 1 – Esquema sobre *Differential Privacy* – Fuente [4]

Dos de los enfoques con los que se viene trabajando para contrarrestar la pérdida de información al compartir datos son: “DP-SGD” y “PATE”.

Differentially-private Stochastic Gradient Descent (DP-SGD) – es un mecanismo de aprendizaje profundo. La fuga sólo ocurre cuando el modelo está entrenado. Una vez en producción, las partes internas del modelo no pueden ser atacados o analizados para obtener nueva información sobre las personas en el conjunto de datos de entrenamiento.

Private Aggregation of Teacher Ensembles (PATE) - es una técnica que permite el entrenamiento de modelos de aprendizaje automático de arquitectura arbitraria de manera

que las garantías de privacidad de los datos se puedan describir a través de *Differential Privacy*. La técnica propone entrenar múltiples modelos "*teachers*" en conjuntos disjuntos de datos privados confidenciales, y luego usar un conjunto de estos maestros para guiar el entrenamiento de un modelo de "*student*" con datos públicos sin etiquetar. En la formación del *student* los datos se envían a través de cada modelo del *teacher* para obtener una predicción de etiqueta y un ruido. La agregación de predicciones se utiliza como etiqueta de muestra de entrenamiento. [5]

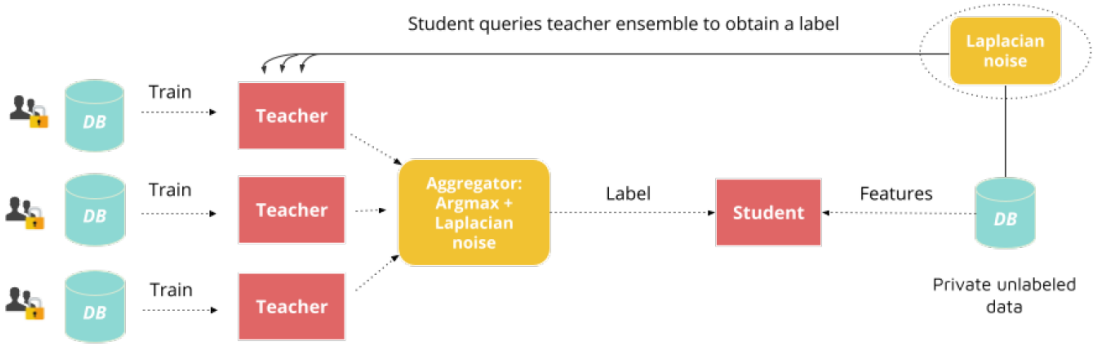


Figura 2 – Esquema sobre PATE – Fuente [5]

Un trabajo comparativo sobre la precisión de ambos modelos mostró que PATE obtuvo mejores resultados, manifestando una desviación estándar menor y menor desproporción en el impacto de la utilidad.[6]

Nuestro trabajo se va a centrar exclusivamente en el análisis de PATE de acuerdo a lo propuesto en el estudio de modelos de privacidad de datos [5] como técnica de privatización de los datos.

Existen incentivos para lograr la privacidad de los datos, como, por ejemplo:

- Mejor privacidad puede producir una mayor precisión
- Mejorar la calidad de los datos
- Aumentar la cantidad de los datos
- Impulsar el progreso científico
- Empoderar a los usuarios

Estos son algunos de los beneficios; para profundizar en este punto recomendamos ver el Anexo 1. También presentamos una definición de MLOps que es el otro componente que investigaremos a lo largo de este trabajo.

MLOps o ML Ops - es un conjunto de prácticas que tiene como objetivo implementar y mantener modelos de aprendizaje automático en producción de manera confiable y eficiente. La palabra es un compuesto de "*machine learning*" y la práctica de desarrollo continuo de DevOps (acrónimo inglés de *development y operations*). Los modelos de aprendizaje automático se prueban y desarrollan en sistemas experimentales aislados.[7]

1.1. Objetivo

El objetivo general de este trabajo es investigar la aplicación de MLOps en el contexto de PATE. Se espera lograr el entendimiento de la herramienta a nivel teórico y vincularlo con las prácticas que se desarrollan en MLOps.

Tomando como idea un caso concreto de aplicación, partimos de organizaciones médicas que están dispuestas a compartir datos de sus pacientes (vale la pena mencionar que esta información es totalmente confidencial), siempre y cuando, no exista pérdida de privacidad o la misma sea mínima.

Dentro de este contexto general la propuesta de implementar PATE, con la figura del *Trusted Curator*, permite aprovechar la privacidad diferencial como herramienta para garantizar la seguridad de los datos.

El objetivo específico es estudiar la factibilidad de la implementación MLOps en el contexto de PATE, superando los problemas que se presenten en la aplicación de las herramientas seleccionadas dentro del escenario elegido para dicho estudio.

Al momento de comenzar nuestra investigación nos encontramos con la contrariedad de que no existe un estudio previo en el cual basarnos y sabemos de antemano que un factor al que debemos prestar atención es la gobernanza del sistema.

Se espera identificar alternativas para la implementación de MLOps en el contexto de PATE y seleccionar la mejor herramienta. También se espera lograr la implementación de la herramienta seleccionada en un ejemplo concreto y generar comentarios que sirvan como insumos para futuros trabajos en la materia.

2. Gobernanza

Uno de los elementos más importantes para tener una buena gestión del entorno donde se aplicaría PATE para varias organizaciones que desean compartir los datos a efectos de entrenar modelos de manera segura es la gobernanza del ecosistema.

Cabe mencionar que no existe un contexto de aplicación concreto para tener como referencia al momento de definir algunos aspectos como los de Gobernanza y las funciones del Comité de Gobernanza. Hicimos algunas suposiciones/hipótesis, que deberán ser validadas o refutadas para cada caso de estudio particular.

2.1. Gobierno de datos (*Data Governance*)

Gobierno de Datos: es el conjunto de procesos, estructura organizacional y herramientas, que aseguran consistencia en la generación, administración, uso y evolución de los datos de una organización acordes a satisfacer objetivos de negocios, regulaciones y/o estándares del mercado [8].

El gobierno de datos formula políticas que son declaraciones escritas sobre cómo las personas deben actuar ante determinada situación. Se debe asegurar la privacidad de los datos para lo cual las organizaciones deben establecer políticas apropiadas asegurar el correcto uso de la información.

Sus objetivos son:

- Definir, aprobar y comunicar estrategias de datos, políticas, normas, procedimientos, arquitectura y métricas.
- Realizar un seguimiento y hacer cumplir el marco normativo asegurando la conformidad con las políticas de datos estándares, arquitectura y procedimientos.
- Fomentar, controlar, y supervisar la ejecución de los proyectos y servicios de gestión de datos.
- Gestionar y resolver los problemas relacionados con los datos corporativos.
- Entender y promover el valor de los activos de datos.

Tomando en cuenta la definición anterior, el gobierno de los datos, se hace necesario debido a varios aspectos, a continuación, mencionaremos los que consideramos más importantes:

- Los proyectos son dirigidos por no expertos.
- Las necesidades del negocio deben ser descubiertas.
- Se trabaja con tecnologías no necesariamente conocidas o con *expertise*.
- El caso de un negocio no se ha desarrollado.
- Las características de los datos no están claras.

Presentamos a continuación una figura que resume de manera gráfica lo hemos descripto anteriormente.

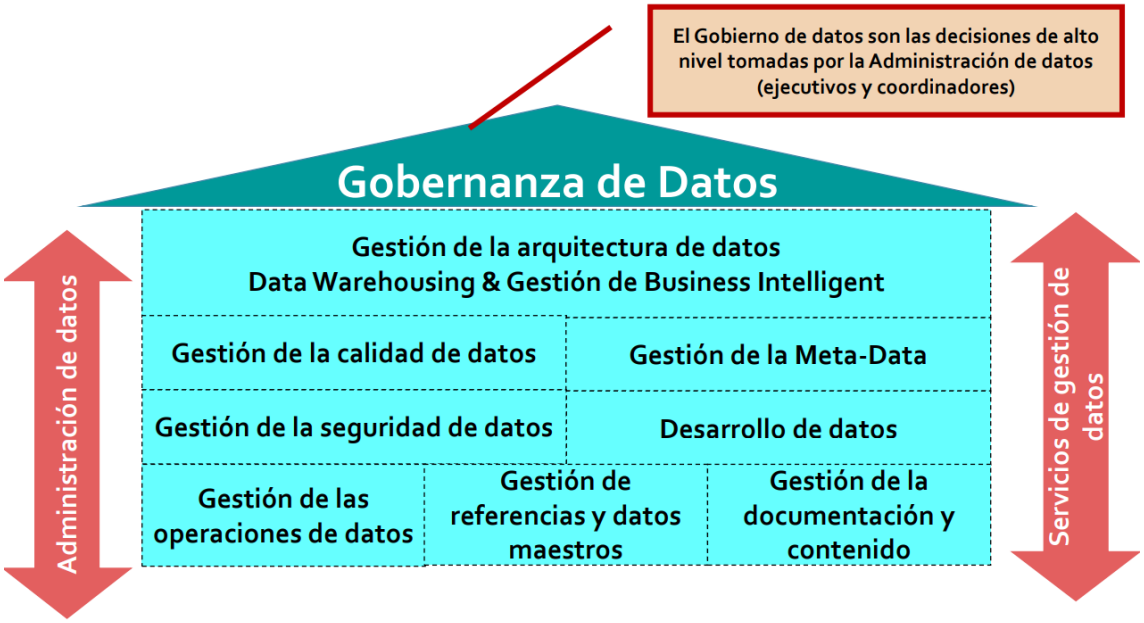


Figura 3 – Esquema Gobernanza de Datos

Hemos presentado los aspectos generales de la gobernanza de *Big Data* y a continuación presentaremos los aspectos más importantes para el caso de nuestro trabajo, en donde se hace necesario contar con una buena gobernanza debido a la dificultad de trabajar con múltiples organizaciones, que manejan datos heterogéneos.

Un aspecto fundamental es llegar a un acuerdo entre todos los participantes para definir que es un dato. Una vez alcanzado el consenso estos datos deben cumplir con ciertas características: seguridad, privacidad, confiabilidad, disponibilidad y performance.

2.2. Gestión de altas y bajas de organizaciones del ensamble

Otro aspecto sobre el cual se debe definir la gobernanza es la salida de organizaciones del ensamble, con la consecuente pérdida de datos y modelos de entrenamiento que ella aportaba. De la misma manera se debe considerar la forma en la cual permitir el ingreso de una nueva organización. Se recomienda la firma de un convenio con cada organización que estipule los plazos, condiciones y requerimientos para estas acciones.

Quizás sea bueno contar con un Comité de Gobernanza, integrado por algunos miembros de los equipos de las organizaciones participantes, que puedan rotar con cierta frecuencia establecida.

2.3. Gestión de los aportes al generar la votación del ensamble

Una vez que cada *teacher* ha definido sus etiquetas, las mismas son enviadas al *Trusted Curator* para que pueda generar la votación (*Argmax*, es la clase más votada). En esta situación el aporte de cada *teacher* es el mismo para el ensamble. Quizás sea recomendable generar un coeficiente de ponderación, tomando en cuenta la cantidad de datos que aporta para el entrenamiento de su modelo y la respuesta a la predicción de la consulta.

En teoría el modelo que fue entrenado con una mayor cantidad de datos, debería predecir mejor. Por lo tanto, su aporte relativo al momento de realizar la votación debería ser mayor.

Cuantificar la cantidad de datos por organización sería una tarea del Comité de Gobernanza.

La introducción de aportes requiere un análisis detallado del modelo teórico.

2.4. Mínimos de modelos por organización para que el ensamble funcione

A efectos de que el ensamble funcione, será necesario una cantidad mínima de modelos. En el contexto de nuestro país y pensando en la implementación práctica, nos animamos a predecir que se va a requerir más de un modelo por organización. La cantidad dependerá de cada caso de uso.

Esto implica un mayor requerimiento computacional para las organizaciones, que deben invertir tiempo y recursos en el entrenamiento de varios modelos. Se recomienda evaluar cuantos modelos pueden entrenar cada organización, en base a su capacidad y a la cantidad de datos disponibles. Esta sería otra función que debe asumir el Comité de Gobernanza.

2.5. Pérdida de privacidad aceptada

Uno de los requerimientos más importantes es asegurar la privacidad de los datos. Cada vez que ser responde una consulta por parte de los *teacher* sabemos que se produce la fuga de información, por lo tanto, es limitada la cantidad de respuestas que se pueden dar.

Existe una forma de medir está perdida de privacidad, mediante la variable *Épsilon*. Será función del Comité de Gobernanza, generar una métrica para controlar la pérdida de privacidad de cada organización. La gestión y el control serán responsabilidades del *Trusted Curator* cada vez que genere el ensamble. Una vez que se llegue al tope máximo de pérdida de privacidad aceptada, aclaramos que cada organización tiene un valor máximo diferente según los datos aportados al ensamble, el *Trusted Curator* deberá quitar a esa organización del ensamble.

Requiere un análisis teórico de pérdida de privacidad.

2.6. Otras funciones del Comité de Gobernanza:

- Monitorear y asegurar que las organizaciones están alineadas con las políticas, estándares, arquitecturas y procedimientos.
- Asegurar y monitorear el proyecto.

- Gestionar y resolver conflictos

Hasta aquí presentamos los aspectos generales que identificamos como relevantes para la gobernanza de un sistema de múltiples organizaciones que deseen implementar PATE. De todas maneras, pueden existir aspectos particulares para cada ámbito que desee llevar adelante la implementación.

Nuestro trabajo continuará con la descripción de posibles escenarios en los cuales se implemente PATE utilizando MLOps, con múltiples organizaciones.

3. Posibles escenarios de implementación de PATE utilizando MLOps

Vale aclarar que, en los siguientes escenarios propuestos, se plantea una extensión del concepto de PATE previamente definido.

Palabras claves:

***Teachers*:** múltiples modelos entrenados con conjuntos de datos disjuntos, como registros de diferentes subconjuntos de usuarios. Debido a que se basan directamente en datos confidenciales, estos modelos no se publican, sino que se utilizan como "*Teachers*".

***Students*:** Modelos entrenados con datos privados sin etiquetar y que son guiados en su entrenamiento por el ensamble de *Teachers*.

***Trusted Curator*:** una entidad en la que los propietarios de los datos confían para almacenarlos y manipularlos.

Se definen los siguientes elementos para todos los escenarios:

- N instituciones que están dispuestas a participar del ensamble (O_1, O_2 … O_N)
- Un *Trusted Curator* para los *Teachers*, quien hace la función de *Aggregate Labels* para el ensamble de las organizaciones (TC_T)
- Una institución que hace la consulta (O_S)
- Un *Trusted Curator* para los *Students*, quien hace la función de agregar ruido para las consultas del *Student* (TCs)
- Además, se llega a un acuerdo en la definición de X e Y como dato

3.1. Escenario 1

Las Organizaciones entrenan con sus propios datos a los *Teachers* y luego disponiblizan sus modelos al *Trusted Curator*.

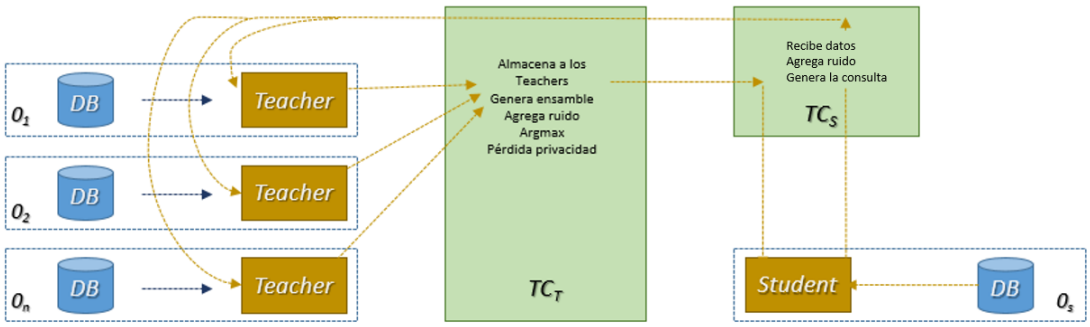


Figura 4 – Escenario 1

Secuencia:

- Cada institución entrena con sus datos a los modelos de manera independiente.
- Disponibilizan los modelos entrenados al TC_T, quien los almacena.
- La institución O_S entrena con sus datos al *Student*
- El TCs recibe los datos sin etiquetar, le agrega ruido y genera la consulta para los *Teachers* (que se encuentra en el TC_T)
- Cada *Teacher* del ensamble es evaluado con los datos que vienen del TCs y por ende generan sus propias etiquetas.
- El TC_T agrega ruido a lo anterior y calcula el resultado de la votación (resultado del ensamble) - *argmax* de la clase más votada. Luego evalúa la perdida de privacidad y devuelve el resultado al TCs

3.2. Escenario 2

Las Organizaciones reciben los datos sin etiquetar directamente, entrenan sus modelos luego envían la respuesta al *Trusted Curator* para aplicar DP y devolver la respuesta.

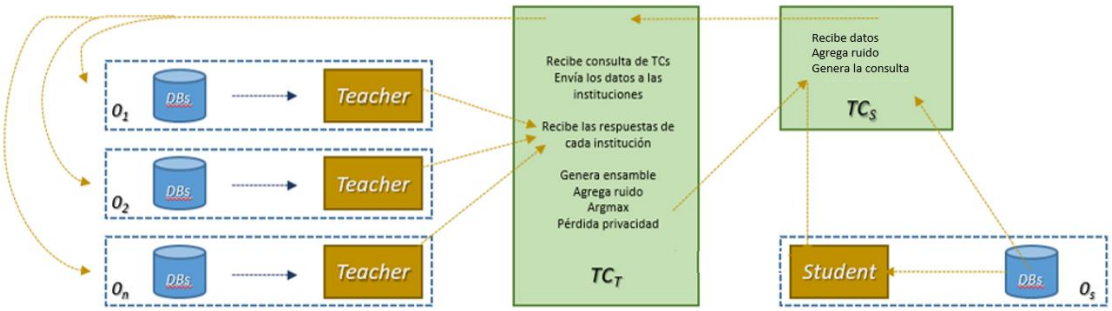


Figura 5 – Escenario 2

Secuencia:

- La institución O_S entrena con sus datos al *Student*
- El TCs recibe los datos sin etiquetar, le agrega ruido y genera la consulta para el TC_T
- El TC_T recibe la consulta y envía a las instituciones para que respondan
- Las instituciones reciben los datos y entrenan los modelos y envían la respuesta al TC_T
- El TC_T agrega ruido a lo anterior y calcula el resultado de la votación (resultado del ensamble) - *argmax* de la clase más votada. Luego evalúa la perdida de privacidad y devuelve el resultado al TCs

3.3. Escenario 3

Las organizaciones entregan todos los datos al *Trusted Curator* para que este pueda generar los modelos y responder las consultas, administrando la perdida de privacidad.

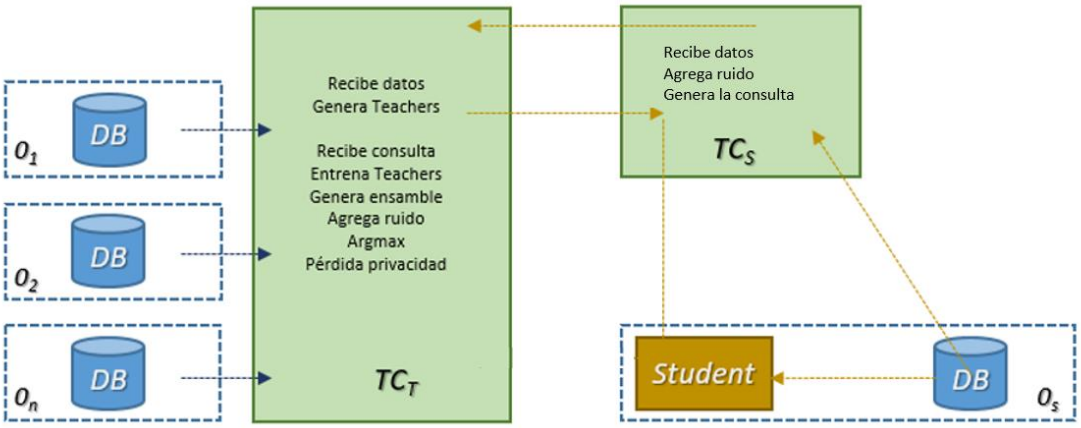


Figura 6 – Escenario 3

Secuencia:

- Cada institución disponibliza de manera segura sus datos al TC_T
- TC_T recibe los datos de cada institución y tiene la potestad de generar los modelos de *Teachers* a la espera de una consulta. Entrena los *teachers* y genera el ensamble
- La institución O_S entrena con sus datos al *Student*
- El TCs recibe los datos sin etiquetar, le agrega ruido y genera la consulta para el TC_T
- El TC_T al recibir la consulta, les agrega ruido y calcula el resultado de la votación (resultado del ensamble) - *argmax* de la clase más votada. Luego evalúa la perdida de privacidad y devuelve el resultado al TCs

3.4. Beneficios y problemas de cada escenario

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3
Beneficios	Las organizaciones entrenan sus propios modelos Existe un solo punto de contacto entre organizaciones y TC_T, es decir solo punto de fuga Menor demora al responder la consulta del *Student* porque los *Teachers* porque están todos centralizados en el TC_T	Como cada organización entrena los *teachers* y genera las predicciones, la capacidad computacional está distribuida El TC_T solo debe ejecutar la función *argmax* y devolver las consultas	El TC_T tiene un control absoluto de los datos y al contar con un gran *dataset* puede generar muchos modelos Menor demora al entrenar los *teachers* porque están todos centralizados en el TC_T Existe un solo punto de contacto entre organizaciones y TC_T, es decir solo punto de fuga Menor demora en dar respuesta en general
Problemas	El TC_T debe hacer todos los cálculos del ensamble, lo que requiere mucha capacidad computacional No existe una plataforma o herramienta concreta para disponibilizar los modelos de manera segura	Cada organización debe entrenar los *teachers* y etiquetar los datos y puede haber demoras No hay supervisión de los modelos que participan en el ensamble Existen más intercambios entre los actores del ecosistema lo que genera mayor cantidad de puntos de fuga	Los datos abandonan las organizaciones, es fundamental el control del TC_T para evitar perdida de privacidad El TC_T debe hacer todos los cálculos del ensamble, lo que requiere mucha capacidad computacional

Tabla 1 – beneficios y problemas de cada escenario

4. Estado del arte MLOps

4.1. Introducción

Hasta ahora hemos conseguido puntualizar los aspectos principales sobre la "Gobernanza" de un sistema que implementaría PATE con múltiples organizaciones. Además, logramos identificar y describir, tres escenarios posibles de funcionamiento, seleccionando el que entendemos es mejor para su implementación con las herramientas que se han investigado.

A continuación, analizaremos el estado del arte de MLOps con el objetivo de poder trasladar sus mejores prácticas a nuestro caso de estudio e intentar acelerar y automatizar los procesos de desarrollo.

4.2. DevOps y MLOps

Como indicamos en la introducción MLOps es una extensión de la metodología de DevOps.

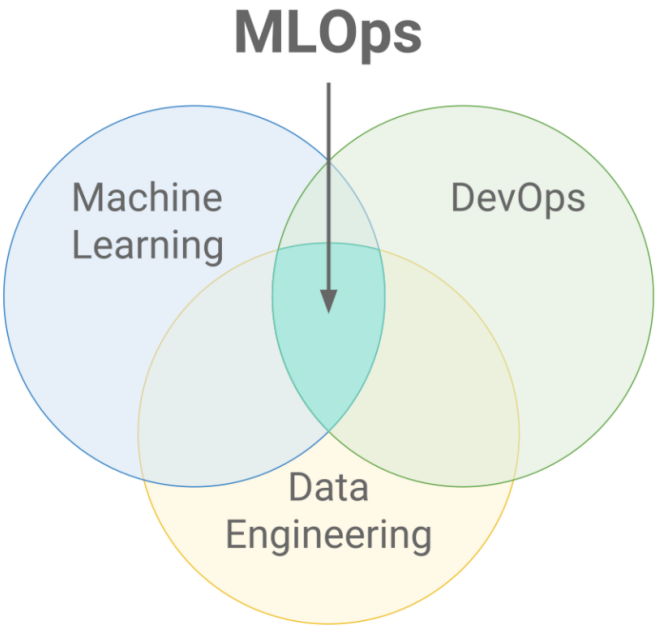


Figura 7 – MLOps – Fuente [7]

.

DevOps es un conjunto de prácticas que agrupan el desarrollo de *software* y las operaciones de TI. Su objetivo es hacer más rápido el ciclo de vida del desarrollo de software y proporcionar una entrega continua de alta calidad [9].

Algunas de estas prácticas ayudan a agilizar, automatizar y mejorar una fase específica. Otras abarcan varias fases y ayudan a los equipos a crear procesos homogéneos que favorezcan la productividad. A continuación, mencionaremos algunas de las principales prácticas: [10]

- Integración y entrega continuas (CI/CD)
- Control de versiones
- Desarrollo ágil de *software*
- Infraestructura como código
- Administración de configuración
- Supervisión continúa

MLOps debe garantizar que los modelos se puedan implementar reiteradamente y monitorear de forma continua. El proceso de MLOps permite lo siguiente:

- Implementar más modelos con mayor rapidez con procesos automatizados
- Acelerar el tiempo de creación de valor con una rápida entrega de modelos
- Optimizar la productividad a través de la colaboración y la reutilización de modelos
- Reducir el riesgo de perder tiempo y dinero en modelos que nunca se incorporan a la producción
- Monitorear y actualizar continuamente modelos a medida que los datos evolucionan con el tiempo

El proceso de MLOps incluye los siguientes pasos [11]:

Crear – esto implica la preparación de datos, la ingeniería de características, la creación de modelos y las pruebas.

Administrar – después de que se crean los modelos, a menudo se colocan en un repositorio de modelos que se puede auditar y cuenta con control de versiones para promover la reutilización en toda la organización.

Implementar – este paso implica exportar el modelo o el pipeline, implementarlo e integrarlo en los sistemas y las aplicaciones empresariales de producción.

Monitorear – se requiere un monitoreo continuo para garantizar un rendimiento óptimo. A medida que los datos evolucionan, se puede volver a entrenar el modelo o se puede reemplazar el existente por uno nuevo.

4.3. Alternativas para implementar MLOps

Existen varias plataformas que permiten la implementación de MLOps, a continuación, presentamos un listado con una breve descripción de cada una [12].

SageMaker – Amazon	*Amazon SageMaker* es el entorno integrado totalmente administrado de *Amazon* para el aprendizaje automático y el aprendizaje profundo. Incluye un entorno Studio que combina *Jupyter notebooks* con gestión y seguimiento de experimentos, un depurador de modelos, un "piloto automático" para usuarios sin conocimientos de aprendizaje automático, transformaciones por lotes, un monitor de modelos e implementación con inferencia elástica.
Azure – Microsoft	*Azure Machine Learning* es un entorno basado en la nube que se puede usar para entrenar, implementar, automatizar, administrar y rastrear modelos de aprendizaje automático. Admite la escritura de código *Python o R*, además de proporcionar un diseñador visual de arrastrar y soltar y una opción de *AutoML*. Puede crear, entrenar y realizar un seguimiento de modelos. *Azure Machine Learning* interactúa con herramientas populares de código abierto, como *PyTorch, TensorFlow, Scikit-learn, Git* y la plataforma MLflow para administrar el ciclo de vida del aprendizaje

	automático. También tiene su propio entorno MLOps de código abierto.
Cloud AI Platform - Google	*Google Cloud AI Platform* incluye una variedad de funciones que admiten la gestión del ciclo de vida del aprendizaje automático: un panel general, *AI Hub*, etiquetado de datos, notebooks, trabajos, orquestación del flujo de trabajo (actualmente en estado de prelanzamiento) y modelos. Los notebooks están integrados con *Google Colab*, donde puede ejecutarlos de forma gratuita. *AI Hub* incluye una serie de recursos públicos que incluyen canalizaciones de *Kubeflow*, notebooks, servicios, módulos de *TensorFlow*, imágenes de VM, modelos entrenados y guías técnicas. Los recursos de datos públicos están disponibles para imágenes, texto, audio, video y otros tipos de datos.
MLflow - Databricks	*MLflow* es una plataforma de gestión del ciclo de vida de aprendizaje automático de código abierto de *Databricks. MLflow* tiene tres componentes, que cubren el seguimiento, los proyectos y los modelos. Más adelante profundizaremos sobre estos aspectos

Tabla 2 – comparativa de plataformas

Al investigar las cuatro plataformas: *SageMaker* [13], *Azure* [14], *Cloud AI Plataform* [15] y *MLflow* [16], notamos que las mismas comparten características similares pero cabe mencionar que la única plataforma *open source* de las mencionadas es *MLflow*.

Tomando en cuenta que el objetivo de este trabajo no es analizar en profundidad ni evaluar cual la mejor, hemos decidido continuar el trabajo basándonos en MLflow por su condición de *open source*.

Quisiéramos hacer una mención especial sobre el desarrollo de algunos conceptos en etapa inicial de investigación, vinculados a Privacidad Computacional [17] y *Privacy Preserving Machine Learning* (PPML) [18] para lo cual recomendamos leer el Anexo 2.

A continuación, presentamos una descripción detallada de los componentes para la implementación de MLOps de *MLflow*:

4.3.1. MLflow Tracking

Es una API y una interfaz de usuario para registrar parámetros, versiones de código, métricas y artefactos al ejecutar su código de aprendizaje automático y para visualizar posteriormente los resultados. Se puede utilizar *MLflow Tracking* en cualquier entorno (por ejemplo, una secuencia de comandos independiente o un cuaderno) para registrar los resultados en archivos locales o en un servidor y luego comparar varias ejecuciones. Los equipos también pueden usarlo para comparar resultados de diferentes usuarios.

MLflow Tracking se organiza en torno a ejecuciones de algún código de ciencia de datos. En cada ejecución se registra la siguiente información:

- Versión del código: hash de confirmación de *Git* utilizado para la ejecución, si se ejecutó desde un proyecto *MLflow*.
- Hora de inicio y finalización: hora de inicio y finalización de la ejecución.
- Fuente: nombre del archivo para iniciar la ejecución, o el nombre del proyecto y el punto de entrada para la ejecución si se ejecuta desde un Proyecto *MLflow*.
- Parámetros: parámetros de entrada de valor clave de su elección. Tanto las claves como los valores son cadenas.
- Métricas: métricas de valor-clave, donde el valor es numérico.
- Artefactos: archivos de salida en cualquier formato.

Se puede registrar ejecuciones utilizando las API de *MLflow Python, R, Java y REST* desde cualquier lugar donde ejecute su código.

Podemos utilizar *MLflow Tracking* para registrar los parámetros y métricas, y posteriormente para guardar y visualizar los resultados. Se puede registrar ejecuciones utilizando varios programas y aplicaciones y desde cualquier lugar desde donde se ejecuten originalmente.

En nuestro trabajo esto permite tener disponibles los diferentes experimentos que hagan las instituciones, en las diferentes plataformas, y poder comparar los resultados (si es que son comparables).

Esta herramienta de *MLflow Tracking* nos permite:

- ❖ Organizar corridas en experimentos: *MLflow* le permite agrupar ejecuciones en experimentos, lo que puede ser útil para comparar ejecuciones destinadas a abordar una tarea en particular.
- ❖ Gestión de experimentos y ejecuciones con la API del servicio de seguimiento: *MLflow* proporciona una API de servicio de seguimiento más detallado para administrar experimentos y se ejecuta directamente, que está disponible a través del SDK del módulo cliente en el *MLflow.tracking*. Esto hace posible consultar datos sobre ejecuciones pasadas, registrar información adicional sobre ellas, crear experimentos, agregar etiquetas a una ejecución y más.
- ❖ IU de seguimiento: La IU de seguimiento le permite visualizar, buscar y comparar ejecuciones, así como descargar artefactos de ejecución o metadatos para su análisis en otras herramientas. Alternativamente, el *MLflow tracking* server sirve la misma interfaz de usuario y habilita el almacenamiento remoto de artefactos de ejecución.
- ❖ La interfaz de usuario contiene las siguientes características clave:
  - ✓ Listado y comparación de ejecuciones basadas en experimentos
  - ✓ Búsqueda de ejecuciones por parámetro o valor métrico
  - ✓ Visualización de métricas de ejecución
  - ✓ Descarga de los resultados de la ejecución
- ❖ La consulta se ejecuta mediante programación: Se puede acceder a todas las funciones de la IU de seguimiento mediante programación. Esto facilita la realización de varias tareas comunes:
  - ✓ Consultar y comparar ejecuciones utilizando cualquier herramienta de análisis de datos de su elección, por ejemplo, Pandas [22].
  - ✓ Determina el URI del artefacto de una ejecución para alimentar algunos de sus artefactos en una nueva ejecución al ejecutar un flujo de trabajo.
  - ✓ Carga artefactos de ejecuciones pasadas como Modelos de *MLflow*.
  - ✓ Ejecutar algoritmos de búsqueda de parámetros automatizados, donde consultar las métricas de varias ejecuciones para enviar nuevas.

- ❖ Servidores de seguimiento de *MLflow*: Un servidor de seguimiento de *MLflow* tiene dos componentes para el almacenamiento: una tienda de *backend* y una tienda de artefactos.
  - ✓ Tiendas *backend*: es donde *MLflow Tracking Server* almacena los metadatos de experimentos y ejecuciones, así como parámetros, métricas y etiquetas para las ejecuciones. *MLflow* admite dos tipos de almacenes de *backend*: almacén de archivos y almacén con respaldo de base de datos.
  - ✓ Tiendas de artefactos: es una ubicación adecuada para datos grandes (como un depósito S3 o un sistema de archivos NFS compartido) y es donde los clientes registran su salida de artefactos (por ejemplo, modelos). El cliente de *MLflow* almacena en caché la información de ubicación de artefactos por ejecución. Por lo tanto, no se recomienda alterar la ubicación del artefacto de una ejecución antes de que haya terminado.

Los servidores de *MLflow*, a través de las tiendas *backend* y las tiendas de artefactos nos pueden servir para almacenar todos los experimentos. Una funcionalidad muy útil e importante que nos ofrece es poder consultar datos de ejecuciones pasadas, crear experimentos nuevos, y agregar etiquetas a una ejecución ya realizada.

A continuación, presentaremos una tabla indicando la forma de utilizar *MLflow Tracking* para cada escenario:

Escenario 1	Estas tiendas nos permiten disponibilizar los modelos entrenados por cada una de las instituciones, para que sean utilizados por el TC_T. Esta funcionalidad de agregar etiquetas nos resulta muy útil tanto para que el *Trusted Curator* TC_T agregue etiquetas para el ensamble, así como también para que el TCs agregue las etiquetas necesarias para el *student*.
Escenario 2	En este escenario también nos resulta muy útil la funcionalidad de que los TC puedan agregar las etiquetas necesarias cuando corresponde, así como almacenar los datos y los resultados de los experimentos. Se pueden consultar y comparar las ejecuciones utilizando herramientas de análisis. Esto nos será útil para comparar los % de acierto de las predicciones.
Escenario 3	Las instituciones pueden disponibilizar de manera segura sus datos para que sean utilizados por el TC_T y pueda generar los modelos de los *Teachers*.

Tabla 3 – MLflow Tracking para cada escenario

4.3.2. MLflow Project

Son un formato estándar para empaquetar código de ciencia de datos reutilizable y reproducible, basado en convenciones. Cada proyecto es simplemente un directorio con código o un repositorio Git, y usa un archivo descriptor o simplemente una convención para especificar sus dependencias y cómo ejecutar el código. Cada proyecto puede especificar varias propiedades:

- Nombre: un nombre legible por humanos para el proyecto.
- Puntos de entrada: comandos que se pueden ejecutar dentro del proyecto e información sobre sus parámetros. La mayoría de los proyectos contienen al menos un punto de entrada al que desea que llamen otros usuarios. Algunos proyectos también pueden contener más de un punto de entrada.
- Entorno: el entorno de *software* que se debe utilizar para ejecutar los puntos de entrada del proyecto. Esto incluye todas las dependencias de la biblioteca requeridas por el código del proyecto.

4.3.3. MLflow Models

MLflow Model sirve para empaquetar modelos de aprendizaje automático en varios "*flavors*" y una variedad de herramientas para ayudarlo a implementarlos. Cada modelo se guarda como un directorio que contiene archivos arbitrarios y un archivo descriptor que enumera varios " *flavors*" en los que se puede usar el modelo.

4.3.4. MLflow Registry

Es una tienda de modelos centralizada, un conjunto de API y una interfaz de usuario para gestionar de forma colaborativa el ciclo de vida completo de un modelo de *MLflow*. Proporciona el linaje del modelo (qué experimento y ejecución de *MLflow* produjo el modelo), control de versiones del modelo, transiciones de etapa y anotaciones.

4.4. Síntesis

Como hemos visto hasta ahora la implementación de MLOps en una organización es un trabajo complejo y que implica el desarrollo de una cultura organizacional para obtener buenos resultados. Existen desafíos adicionales cuando se intenta lograr esa implementación en un contexto de múltiples organizaciones.

Según el paper "*MLOps Challenges in Multi-Organization Setup: Experiences from Two Real-World Cases*" [19] "…No es factible compartirlo todo", comprender las implicaciones de esa afirmación define los límites con los cuales se debe trabajar en el contexto de la inteligencia artificial (IA) y el aprendizaje automático (ML).

Para casos que involucran a diferentes entidades legales (empresas, organizaciones no gubernamentales, organizaciones sin fines de lucro etc.) su colaboración y los límites, para compartir los datos pueden ser más evidentes.

Al día de hoy existen varias limitantes que nos obligan a repensar el enfoque inicial de implementar las técnicas de MLOps en un entorno de PATE.

Es por eso que sugerimos que la implementación de MLOps se realice en la interna de cada una de las organizaciones que participan del ensamble. Nuestro objetivo de ahora en adelante, será buscar opciones para disponibilizar los resultados del entrenamiento de los *Teachers* de manera segura, para luego generar el ensamble, hacer la votación, privatizar los datos agregando ruido y devolver la respuesta al *Student*.

En el siguiente capítulo analizaremos la alternativa de implementar *Federeted Learning* la plataforma *OpenMined* para alcanzar el objetivo antes mencionado.

5. Concepto de *OpenMined y Fedegreted Learning* [20]

Es una comunidad de código abierto cuyo objetivo es hacer que el mundo preserve la privacidad al reducir la barrera de entrada a las tecnologías privadas de inteligencia artificial.

Hoy en día, los flujos de trabajo para la ciencia de datos se han diseñado con varios supuestos: los datos son centralizados por quienes los recopilan, los datos se consideran privados y cualquier uso externo de esos datos debe controlarse estrictamente o rechazarse por completo. Todos estos problemas van en contra de los mejores intereses de la humanidad. La ciencia de datos debe utilizarse para avanzar en el progreso científico y al mismo tiempo, priorizar la privacidad, la seguridad y la integridad de los datos.

Con *OpenMined* las personas y las organizaciones pueden alojar conjuntos de datos privados, lo que permite a los científicos de datos entrenar o consultar datos que "no pueden ver". Los propietarios de los datos mantienen el control total: los datos nunca se copian, mueven ni comparten.

La misión de la comunidad *OpenMined* es crear un ecosistema accesible de herramientas de privacidad y educación.

El análisis para preservar la privacidad comienza con el análisis de datos que no se pueden ver. Por lo tanto, comienza con la capacidad de ejecutar cálculos arbitrarios en datos que se encuentran dentro de una máquina a la que no se tiene acceso, también conocida como ejecución remota.

5.1. Federated Learning

Es un tipo de ejecución remota en el que los modelos se envían a máquinas de almacenamiento de datos remotas para la capacitación local. Esto elimina la necesidad de almacenar datos de entrenamiento confidenciales en un servidor central.

¿Qué se debe tener en cuenta al crear un sistema de *Federated Learning*?[21]

Empresas como *Google* y *Apple* han sido pioneras en *Federated Learning* como una forma de construir modelos de aprendizaje automático de mayor rendimiento en conjuntos de datos distribuidos sin comprometer la privacidad. Hoy, *Google* usa el aprendizaje federado para impulsar las predicciones del teclado en *Gboard* y *Apple* lo usa para mejorar la precisión de *Face ID y Siri*.

A continuación, presentaremos una guía que permitirá configurar un sistema de *Federated Learning* escalable:

- Paso 1: elegir el *framework* del modelo
- Paso 2: determinar el mecanismo de la red
- Paso 3: construir el servicio centralizado
- Paso 4: diseñar el sistema del cliente
- Paso 5: configurar el proceso de formación
- Paso 6: establecer el sistema de gestión de modelos
- Paso 7: abordar la privacidad y la seguridad

Antes de continuar explicaremos los conceptos básicos de *Federated Learning*, ver el Anexo 3, se plantea un ejemplo concreto en el Anexo 4.

A diferencia de las técnicas tradicionales de aprendizaje automático que requieren que los datos estén centralizados para el entrenamiento, *Federated Learning* es un método para entrenar modelos en conjuntos de datos distribuidos. Parte de un modelo de aprendizaje automático se entrenan donde se encuentran los datos (por ejemplo, estos podrían ser conjuntos de datos privados de dos o más empresas) y los parámetros del modelo se comparten entre los participantes para producir un modelo mejorado.

Ningún dato se mueve dentro del sistema, lo que significa que las organizaciones pueden colaborar sin comprometer la privacidad o la propiedad intelectual confidencial, mientras se evita el dolor y el gasto de transferir datos a través de medios tradicionales.

A continuación, se muestran algunos ejemplos de dónde se puede utilizar *Federated Learning*:

- Mejora de los modelos de procesamiento del lenguaje natural, para solución de procesos de automatización robótica usando información de múltiples empresas.
- Aumentar la precisión de los modelos de detección de fraude utilizando datos de empresas de tarjetas de crédito y bancos.
- Mejorar los modelos de atribución utilizando datos de anunciantes y editores.
- Mejorar los sistemas de personalización y recomendación utilizando datos de diferentes empresas de consumo.
- Mejorar los modelos de visión por computadora para el diagnóstico de la salud utilizando datos de varios hospitales.
- Evitar la necesidad de migrar bases de datos a una ubicación centralizada con fines de aprendizaje automático.

En este caso, la variable de destino y las entradas para la tarea de aprendizaje automático son las mismas en todos los conjuntos de datos, pero se necesitan más muestras para hacer un mejor modelo.

Paso 1: Elija el *framework* de su modelo

El primer paso es elegir donde se realizará la implementación del modelo. Es necesario elegir un *framework* que tenga algún soporte para *Federated Learning*. Los criterios de selección incluyen elementos como el dominio (imágenes, PNL o datos tabulares), la familiaridad del equipo con la tecnología y la compatibilidad del *framework* con la infraestructura existente.

PyTorch o *TensorFlow* son dos alternativas; estas bibliotecas proporcionan algunas facilidades para el aprendizaje federado, pero es necesario agregar componentes adicionales para obtener una solución completa. Los componentes clave se describen más adelante.

Paso 2: determinar el mecanismo de la red

A continuación, se debe determinar qué mecanismo de red utilizar. Este mecanismo es el formato de mensajería y el *framework* para pasar las instrucciones entre cada participante en la red de *Federated Learning*.

Se mencionan algunas opciones para elegir:

- *PySyft* con *PyTorch*: un marco centrado en *PyTorch* de *OpenMined* para permitir el aprendizaje federado
- *Flower*: un *framework* genérico que abstrae el flujo de mensajes para admitir múltiples *framework* de modelado.
- *Tensorflow Federated*: el enfoque de *Tensorflow* para distribuir las operaciones del modelo

Hemos decidido que la opción adecuada para nosotros es *PySyft*, más adelante profundizaremos sobre esta decisión. Esta decisión afectará el entorno en el cual se desea realizar la implementación, será necesario que todas las organizaciones que participen del ensamble en PATE, utilicen esta herramienta.

Paso 3: construya el servicio centralizado

Una vez que haya elegido su *framework* y el mecanismo de red, debe establecer un servicio centralizado para administrar a los participantes. Este servicio se encargará de coordinar la comunicación entre los participantes, así como de monitorear el avance de la formación. En nuestro caso este rol es desempeñado por el *Trusted Curator* de los *Teachers*.

Desde una perspectiva operativa, este servicio probablemente necesitará:

- Tener mecanismos de autenticación y autorización integrados, junto con la estructura de soporte para mantenerlo confiable; esto incluye asegurarse de que el servicio no tenga estado para ayudar en el equilibrio de carga, lo que significa que se debe elegir un mecanismo de almacenamiento para mantener el paso de información intermedia entre clientes.

- Implementarse y mantenerse para satisfacer la demanda del sistema de *Federated Learning*.
- Poder administrar las sesiones de *training*.

Hay consideraciones de diseño clave adicionales desde una perspectiva funcional, a tener en cuenta también. Por ejemplo:

- ¿Es necesario agregar autorización o aislamiento de servicio para separar diferentes redes de datos? Es decir, algunos grupos de participantes que pueden colaborar juntos, pero no entre grupos.
- ¿Puede un cliente activar una sesión de formación o tiene que administrarse de forma centralizada?
- ¿Cómo afectarán al entrenamiento los clientes que se desconecten y vuelvan a conectarse? Este problema se ve agravado por la cantidad de partes que participan en la capacitación.
- ¿Qué estadísticas se deben recopilar durante la sesión de entrenamiento y cómo se configurará el monitoreo para que el sistema pueda medir la calidad del modelo que se está entrenando?
- ¿Cómo se gestionará la diferencia de velocidades de procesamiento de las organizaciones?

Paso 4: diseñar el sistema del cliente

Ahora es el momento de considerar un posible diseño para el sistema cliente. Este sistema debe poder realizar operaciones de *training* del lado del cliente y coordinar los parámetros del modelo con el servicio central. El sistema cliente también necesitará obtener nuevos parámetros de otros clientes en la red para actualizar el modelo local.

Para diseñar y entregar una aplicación cliente, debe buscar responder las siguientes preguntas:

- ¿Cómo se autenticará el cliente y se comunicará con el servidor?
- ¿Cómo se manejará la recuperación de errores durante el proceso de entrenamiento del modelo?

Paso 5: configurar procesos de *training*

El sistema de *Federated Learning* necesita saber qué datos privados deben usarse de cada cliente para entrenar los modelos locales para una sesión en particular.

Esta información debe provenir de otro usuario o del servicio central. Por lo tanto, la metainformación sobre los datos disponibles debe gestionarse de alguna forma; esto normalmente lo hace el servicio central (un *Trusted Curator*).

Esto también requiere que los clientes registren esta metainformación sobre qué conjuntos de datos están disponibles para otros clientes. De manera similar, los metadatos de cada cliente deberán recuperarse del servicio central sobre qué conjuntos de datos deben usarse para una sesión de *training*. Como mencionamos todos estos datos se pueden obtener utilizando la herramienta *MLflow* antes descripta.

Paso 6: Establecer el sistema de gestión de modelos

El sistema de *Federated Learning* necesita administrar las métricas del modelo y el acceso para que los usuarios apropiados puedan usar el modelo entrenado.

Las siguientes preguntas se pueden utilizar para guiar cómo el sistema administrará los modelos:

- ¿Todos los participantes deberían poder acceder a una copia del modelo o solo algunos participantes tienen acceso a él?
- ¿Dónde se almacenará el modelo (normalmente lo hará el *Trusted Curator*)?

Paso 7: abordar la privacidad y la seguridad

El modelo final de *Federated Learning*, no soluciona el problema de la privacidad, es por eso que los trabajos previos investigaron la utilización de PATE como herramienta para resguardar la privacidad de los datos. Se toman los resultados obtenidos en la tesis [5] y el artículo [23] como válidos y que dan lugar a este trabajo.

La optimización del riesgo del modelo frente al rendimiento del modelo depende del caso de uso, por lo que es importante involucrar a las partes interesadas adecuadas al tomar esta decisión.

Conclusión: *Federated Learning* tiene un gran potencial para ser una herramienta clave para los científicos de datos, permitiéndoles entrenar mejores modelos en conjuntos de datos distribuidos sin comprometer la privacidad o la propiedad intelectual confidencial, en conjunto con PATE. Sin embargo, comprender dónde y cómo comenzar a implementar *Federated Learning* puede parecer una tarea abrumadora.

Avanzando con nuestro trabajo a continuación presentaremos un ejemplo de implementación de la herramienta *PySyft* en conjunto con *PyGrid* de *Openmined* que se han encontrado en la bibliografía.

5.2. Solución propuesta de PATE con *PyTorch* y *PySyft*

Luego de haber leído el material que nos fue proporcionado y el que pudimos encontrar, así como también de analizar las alternativas de implementación de MLOps en el contexto de PATE, hemos tomado la siguiente decisión.

De los tres escenarios descriptos anteriormente, el que vemos viable de implementar es el escenario 2. Recordamos que en este escenario las consultas ya privatizadas del *Student* son enviadas directamente a las organizaciones quienes entrenan sus modelos a la interna y luego disponibilizan el resultado al TC_T para que pueda generar el ensamble, votar, agregar ruido y devolver el resultado.

Esto se debe a que el flujo de información entre los actores, hace que este escenario sea más seguro para las organizaciones que participan del ensamble y que son las que pueden comprometer los datos en caso de fuga de información.

Lo anterior se logra debido a que, tanto los datos como los modelos de entrenamiento nunca abandonan las organizaciones. El TC_T solo debe acceder a las respuestas etiquetadas de los modelos de manera segura.

A continuación, presentaremos un ejemplo teórico de como funcionaria con las herramientas de *Pytorch* y *PySyft* para que se pueda comprender [24].

Para entender el contexto les presentamos las figuras que participaran del ensamble con la siguiente imagen.

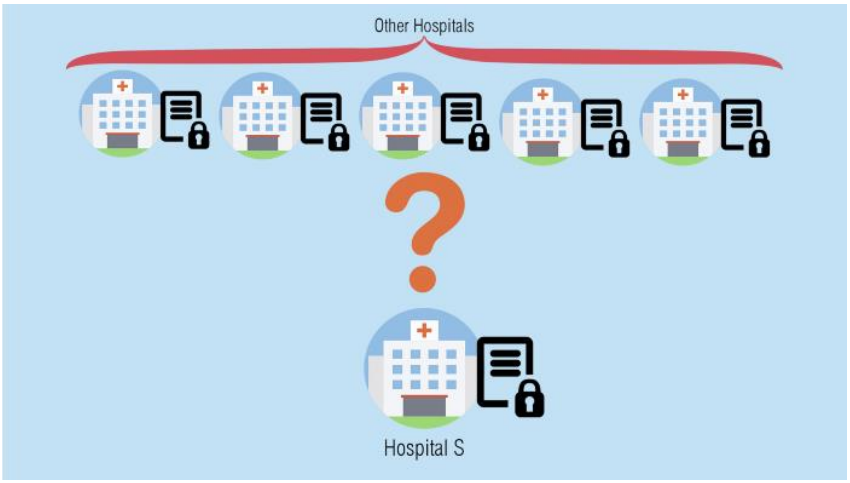


Figura 8 – Esquema PATE – Fuente [24]

Un hospital S tiene cierta cantidad de datos sin etiquetar sobre sus pacientes y quiere usar estos datos para entrenar a un clasificador. Para resolver esto, el hospital S considera usar datos etiquetados de otros hospitales para entrenar un algoritmo de aprendizaje y usarlos para etiquetar sus datos. Aunque los otros hospitales están dispuestos a ayudar, no están autorizados a compartir sus datos con terceros debido a problemas de privacidad.

Apliquemos el marco PATE. Consideraremos al Hospital S como nuestro *Student* y a los demás hospitales como nuestros *Teachers*. Se pide a los *Teachers* que entrenen sus algoritmos de aprendizaje utilizando sus conjuntos de datos privados.

1. Declare los trabajadores de cada hospital

La librería *Syft* utiliza la denominación “trabajadores” para identificar y conectarse con otros dispositivos. En nuestra situación, tenemos la máquina del *Student* (que consideraremos el trabajador local) y los dispositivos de los otros hospitales (que se llamarán *Teachers*). Esta herramienta no limita el número de *Teachers* que se pueden utilizar y se recomienda la mayor cantidad posible.

2. Definir y cargar los modelos para los *Teachers*

Una de las muchas ventajas del marco PATE es el hecho de que es independiente del modelo, lo que significa que no se limita a una familia particular de algoritmos de aprendizaje, por lo tanto, cada *Teacher* puede utilizar un modelo diferente.

Esta guía asume que todos los modelos de los *Teachers* ya han sido entrenados. Solo necesitamos cargarlos y enviarlos a sus máquinas. En un escenario de la vida real, el *Student* no tendría ningún papel en este proceso; en cambio, cada profesor estaría a cargo de instanciar su modelo.

3. Prepare los datos de los *Student*

Ahora que todos los *Teachers* están listos, carguemos el conjunto de datos sin etiquetar del *Student*.

En este punto, tenemos todos los requisitos para el marco PATE. Tenemos un conjunto de datos sin etiquetar y varios modelos previamente entrenados con datos privados e inconexos. Ahora solo necesitamos obtener las etiquetas de nuestro conjunto de datos agregando las predicciones de todos los *Teachers*.

4. Envíe los datos a cada *Teachers* para su análisis.

El *Student*, busca obtener predicciones de los *Teachers* con respecto a cada punto de datos en su conjunto de datos. Para preservar la privacidad desde el punto de vista de los *Teachers*, el *Student* no tiene acceso a sus modelos. En cambio, debe enviar sus datos a cada *Teachers* y esperar el análisis.

Aquí encontramos una pérdida de privacidad para el *Student*, y una solución es incorporar la figura del Trusted Curator (Secure Worker), en el diagrama presentado más abajo se representa con la línea de color naranja.

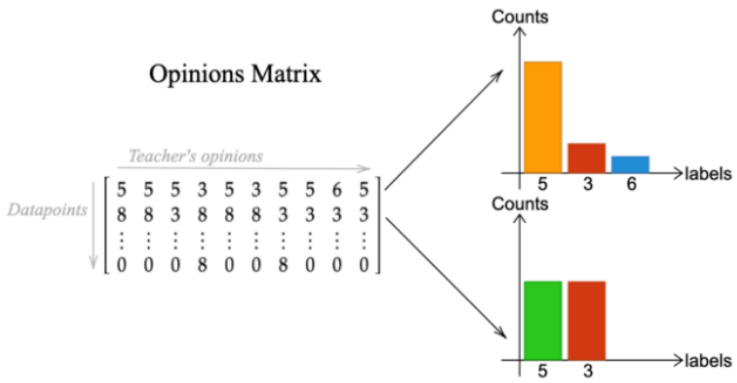
PATE requiere que todas las opiniones sean accesibles en un punto para generar las etiquetas más votadas, añadimos otro *Secure Worker* que sea el encargado del proceso de agregación para las respuestas de los *Teachers*.

Una vez finalizada esta etapa, *secure_worker Teachers* tendrá una matriz con forma (*data_size, num_teachers*). Esta matriz contiene todas las etiquetas predichas de todos los *Teachers* a lo largo de los puntos de datos en el conjunto de datos.

En la descripción del escenario 2, la figura de los *Secure Worker* la hemos nombrado *Trusted Curator*, uno para el *Student* y otro para los *Teachers*.

5. Agregue las Predicciones

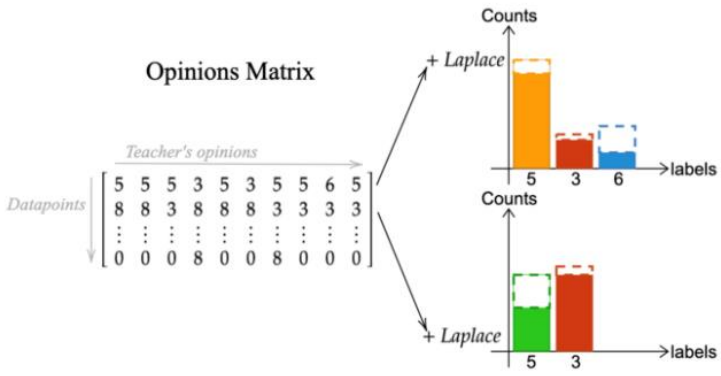
Consideremos cómo se ve la matriz de predicciones en este momento. Para cada fila de nuestro conjunto de datos, tenemos algunas posibles etiquetas que fueron asignadas por cada maestro.



An example of how the opinions matrix may look like. In some cases, there's a clear consensus, while others require further processing.

Figura 9 – Esquema de la matriz de opiniones de los *Teachers* – Fuente [24]

Al seleccionar la etiqueta más votada para cada punto de datos, obtenemos una clasificación generalizada de nuestros datos. Sin embargo, si hay casos en los que no hay una etiqueta obvia, corremos el riesgo de seleccionar una que se clasificó incorrectamente debido a un ajuste excesivo. Este es un problema de privacidad, y para mitigarlo, agregamos un ruido *laplaciano* cuidadosamente medido a todos los recuentos de votos. De esta manera, estamos agregando una negación plausible, al tiempo que retenemos una alta precisión.



Previous scenario but with added noise. A high correlation between teachers leads to higher accuracy, while a low consensus with added noise leads to stronger privacy guarantees.

Figura 10 – Esquema de la matriz de opiniones de los *Teachers* con ruido agregado – Fuente [24]

Se define una función, que llamaremos "El mecanismo ruidoso de *Argmax*". Esta función recibirá la matriz de opiniones y un valor para Seguridad diferencial (ε). De esta forma, se controla la cantidad de ruido que se agregará.

Recuerde que la matriz de opiniones está dentro de la máquina de *secure_worker*. Debido a esto, también tenemos que enviar el ruido que se agregará.

6. Obtención de las etiquetas resultantes

Ahora que la consulta de agregación está definida, podemos obtener los resultados y hacer una comparación con las etiquetas verdaderas.

Aquí está el diagrama completo de lo que se ha hecho hasta ahora.

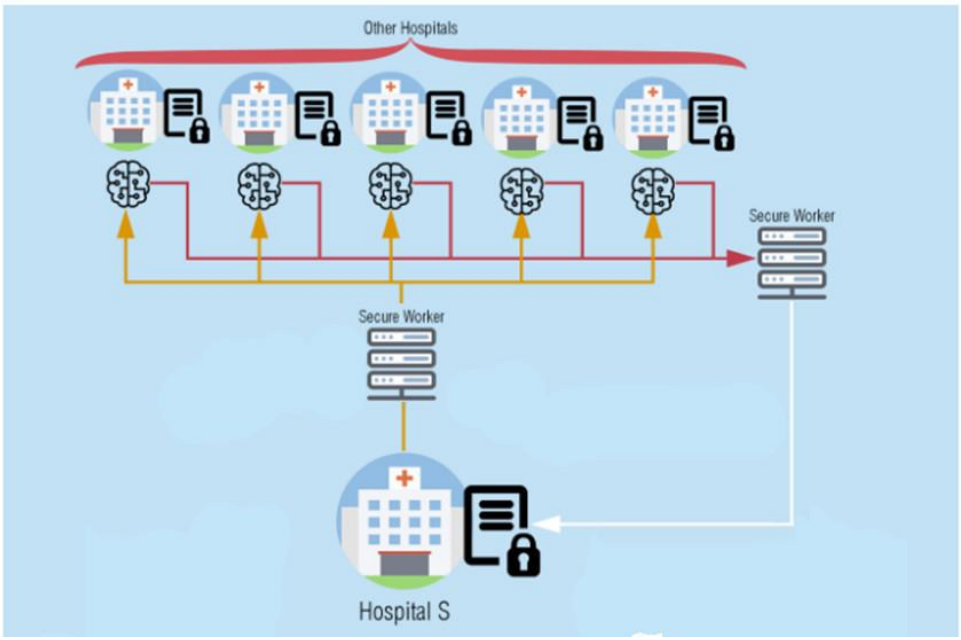


Figura 11 - Diagrama de implementación. Naranja datos del *Student* con ruido. Rojo predicciones de los *Teachers* y blanco, etiquetas del ensamble con ruido – Fuente [24]

En este punto, el *Student* puede entrenar su algoritmo de aprendizaje haciendo uso de este conjunto de datos, y solo se ha filtrado información por un valor menor o igual a *épsilon*.

Cuando se intentó replicar esta solución, no fue posible debido a que la versión de *PySyft* había quedado obsoleta. En el capítulo siguiente describiremos una solución propuesta por nosotros, utilizando la última versión de *PySyft*.

6. Implementación de *PyTorch* y *PySyft* para el escenario seleccionado

Recordamos cuales eran las condiciones del escenario seleccionado (escenario 2).

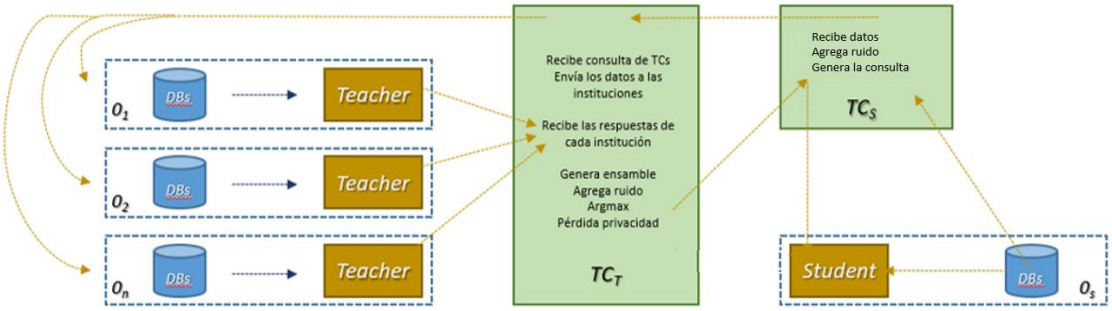


Figura 12 – Escenario 2

Al incorporar los conceptos de nodos para interactuar utilizando *PySyft*, el escenario cambia de la siguiente manera.

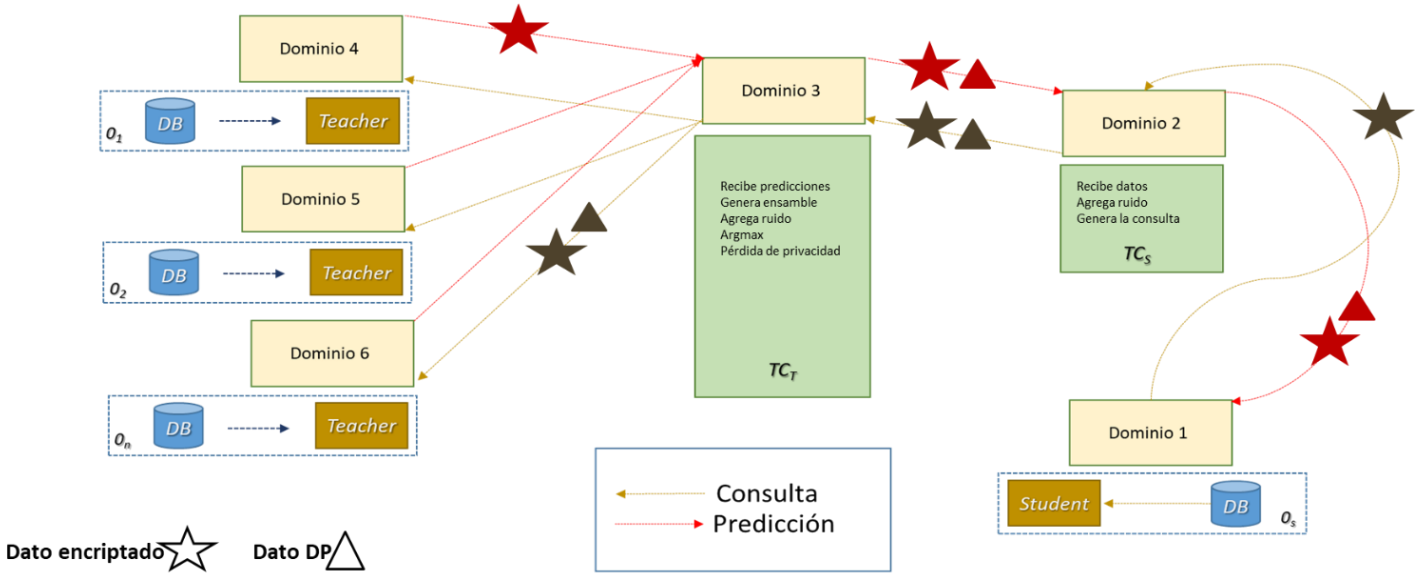


Figura 13 – Escenario 2 *PySyft*

A continuación, presentaremos un ejemplo simplificado de ese escenario, en el cual logramos la Implementación exitosa de *Pysyft* versión 0.6.

Cabe mencionar que para lograr dicha instalación se realizaron varios intentos fallidos debido a que el material disponible se encuentra basado en información desactualizada y código obsoleto. Presentamos en el Anexo 5 una descripción de ese recorrido, que incluye todas las alternativas testeadas sin éxito, con la intención de facilitar futuros trabajos de investigación.

Dividimos el proceso de intercambio de información en dos Etapas:

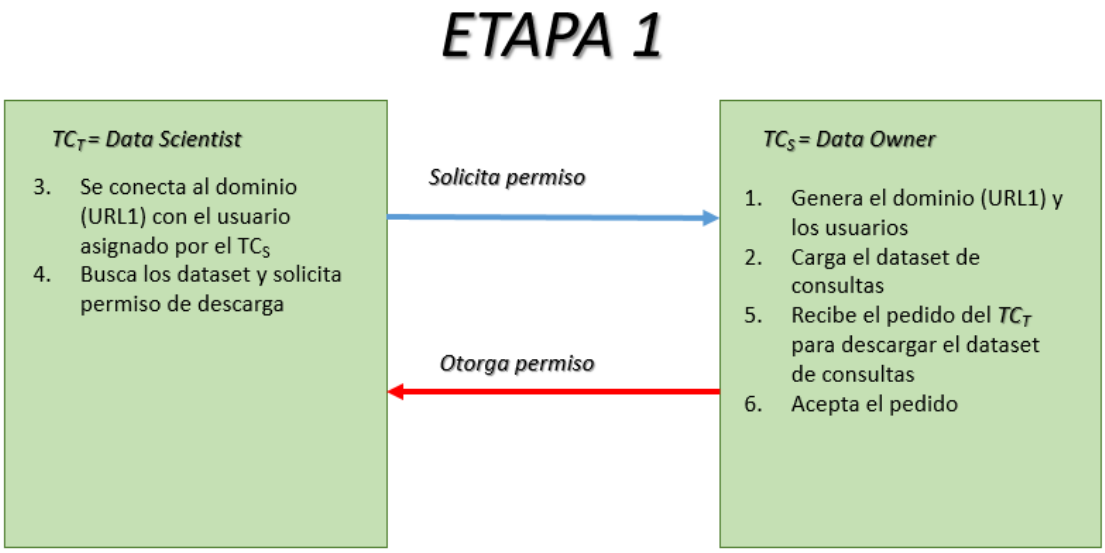


Figura 14 – Etapa 1

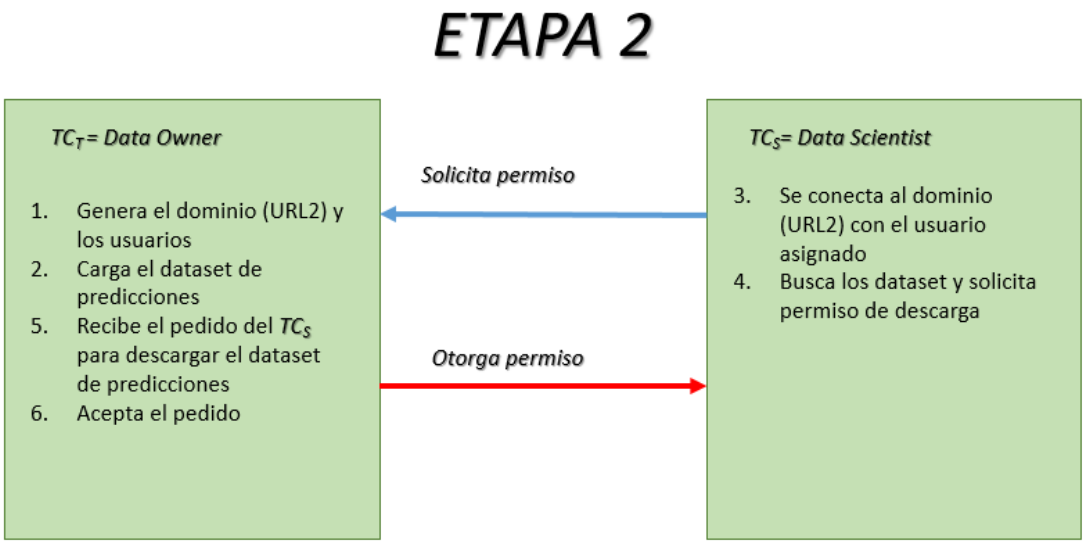


Figura 15 – Etapa 2

Como se ha podido observar es necesario contar con un dominio por cada organización para generar el intercambio de información. En nuestro caso se utilizó *Azure* pero se pueden utilizar otros, el beneficio de utilizar *Azure* en este caso, es que se cuenta con una instalación de un solo *click* para definir los dominios.

En cuanto a los roles, cuando la organización es quien lo genera el dominio, tiene el rol de *Data Owner* y se permite o no el acceso a los datos. En el caso de solo querer solicitar la información, se tiene el rol de *Data Scientist* y se debe hacer la solicitud y esperar la aprobación.

Otro de los puntos importantes es que los datos se manejan como tensores, se recomienda por comodidad, convertirlos a Panda *DataFrame* para poder entrenar los modelos.

En el Anexo 6 se presenta una explicación del código generado y se pueden encontrar los notebooks en el siguiente repositorio.

https://github.com/jramas79/Implementacion-Pysyft-version-0.6.-

Las bases y fundamentos de las interacciones y roles utilizando *PySyft* han quedado definidas y validadas. En un caso real será necesario incorporar más dominios (uno por cada participante), incorporar PATE agregando DP a los datos y generando el ensamble para la votación.

En próximas etapas, se debería implementar PATE. Dado que está herramienta ya se ha validado en otros proyectos, no fue el foco de este trabajo. Existieron algunas dificultades que demoraron dicha implementación.

Creemos que se pueden utilizar los conceptos de *MLOps* que han sido descriptos a lo largo de este trabajo, incorporando la herramienta *MLflow*, para poder hacer un seguimiento y generar un *deploy* de manera más rápida y eficiente.

7. Conclusiones

En base al objetivo de la tesis, investigamos distintas herramientas para la implementación de MLOps en el contexto de PATE. Habiendo realizado un análisis global.

Podemos asegurar que es viable un despliegue realista para implementar la herramienta de PATE utilizando *PySyft* y delegando la parte de MLOps a los participantes del sistema. Esto permitirá optimizar tiempos y recursos al momento de hacer un *deploy* y generar garantías de privacidad.

En base a las definiciones de abordaje del problema, la utilización de MLOps queda a cargo de las organizaciones que participan en el sistema. Esto se debe a que hicimos foco en el intercambio de datos de forma segura.

El objetivo específico de este trabajo era estudiar la factibilidad de la implementación MLOps en el contexto de PATE. Concluimos que esto es viable y para ellos se definieron varios criterios vinculados a la gobernanza del sistema y se describieron tres escenarios posibles de aplicación. En base al análisis de los beneficios y problemas de cada escenario, se eligió el que consideramos óptimo para desarrollar el estudio de *PySyft*.

Otra conclusión importante es que existe una herramienta de la cual realizamos una prueba de concepto, que permite el intercambio de información de manera segura entre los actores del sistema. Esa herramienta es *PySyft* en su versión 0.6.

Adicionalmente recomendamos mantener el contacto con los desarrolladores de *PySyft* (*OpenMined*) mediante la comunidad "*Slack*", porque la herramienta evoluciona más rápido que la actualización de la publicación de los contenidos y se discontinúan las versiones anteriores sin aviso.

8. Referencias Bibliográficas

[1] IArtificial.net, "Ensembles: voting, bagging, boosting, stacking", mayo 2019 [Online]. Available: https://www.iartificial.net/ensembles-voting-bagging-boosting-stacking/. Accesed on Nov. 05, 2021.

[2] Wikipedia, "Arg max", octubre 2022 [Online]. Available: https://en.wikipedia.org/wiki/Arg_max. Accesed on: Nov. 07, 2021.

[3] Ciberseguridad, "¿Qué es la privacidad diferencial y cómo protege tus datos?" [Online]. Available: https://ciberseguridad.com/guias/prevencion-proteccion/privacidad-diferencial/. Accesed on: Nov. 07, 2021.

[4] Wikipedia, "Differential privacy", octubre 2022 [Online]. Available: https://en.wikipedia.org/wiki/Differential_privacy. Accesed on: Nov. 17, 2021.

[5] Visca, Ramiro (2021) Estudio de modelos de privacidad de datos. Entregado como requisito para la obtención del título de Máster en Ingeniería (Por Investigación). Montevideo: ORT.

[6] Archit Uniyal, Rakshit Naidu, Sasikanth Kotti, Sahib Singh, Patrik Joslin Kenfack, Fatemehsadat Mireshghallah,Andrew Trask. DP-SGD vs PATE: Which Has Less Disparate Impact on Model Accuracy?. March 2022. arXiv:2106.12576

[7] Wikipedia, "MLOps", Setiembre 2022 [Online]. Available: https://en.wikipedia.org/wiki/MLOps. Accesed on: Oct. 12, 2021.

[8] Garbarino, Helena y Bianchi Alejandro (2019) Fundamentos de métodos analíticos para Big Data, Cátedra de Sistemas de Información. Montevideo: ORT.

[9] Wikipedia, "DevOps", Junio 2022 [Online]. Available: https://es.wikipedia.org/wiki/DevOps. Accesed on: Oct. 24, 2021.

[10] Microsoft-Azure, "¿Qué es DevOps?" [Online]. Available: https://azure.microsoft.com/es-es/overview/what-is-devops/#practices. Accesed on: Oct. 06, 2021.

[11] Alteryx, "Operaciones de Machine Learning (MLOps)" [Online]. Available: https://www.alteryx.com/es-419/glossary/mlops. Accesed on: Dic. 13, 2021.

[12] Infoworld, "10 MLops platforms to manage the machine learning lifecycle", Setiembre 2021 [Online]. Available: https://www.infoworld.com/article/3572442/10-mlops-platforms-to-manage-the-machine-learning-lifecycle.html. Accesed on: Feb. 10, 2022.

[13] Alonso, Fabricio y Esteves, Martín (2021) Investigación de Amazon SageMaker como plataforma de aprendizaje automático en la nube para moderación de opiniones en PedidosYa. Entregado como requisito para la obtención del título de Máster en Big Data. Montevideo: ORT.

[14] Google Cloud, "Productos de inteligencia artificial y aprendizaje automático" [Online]. Available: https://cloud.google.com/products/ai. Accesed on: Set. 22, 2021.

[15] Microsoft-Azure, "Aprender, conectar y explorar" [Online]. Available: https://azure.microsoft.com/es-es/. Accesed on: Set. 22, 2021.

[16] Mlflow, "An open source platform for the machine learning lifecycle" [Online]. Available: https://mlflow.org. Accesed on: Set. 22, 2021.

[17] Microsoft-Azure, "Azure confidential computing" [Online]. Available: https://azure.microsoft.com/en-us/solutions/confidential-compute/ Accesed on: Set. 08, 2021.

[18] Microsoft, "Privacy Preserving Machine Learning: Maintaining confidentiality and preserving trust" Noviembre 2021 [Online]. Available: https://www.microsoft.com/en-us/research/blog/privacy-preserving-machine-learning-maintaining-confidentiality-and-preserving-trust/. Accesed on: Set. 11, 2021.

[19] Granlund, T; Koponen, A; Stirbu, V; Myllyaho, L; Mikkonen, T. MLOps Challenges in Multi-Organization Setup: Experiences from Two Real-World Cases. arXiv:2103.08937v1 [cs.SE] 16 Mar 2021.

[20] OpenMined, "A world where every good question is answered" [Online]. Available: https://www.openmined.org/. Accesed on: Nov. 07, 2021.

[21] OpenMined, "Design a federated learning system in seven steps" Octubre 2021 [Online]. Available: https://blog.openmined.org/design-a-federated-learning-system-in-seven-steps/. Accesed on: Oct. 31, 2021.

[22] Pandas, "pandas" [Online]. Available: https://pandas.pydata.org/. Accesed on: Set. 14, 2021.

[23] Yovine, S.;Mayr, F.; Sosa, S.; Visca, R.; An Assessment of the Application of Private Aggregation of Ensemble Models to Sensible Data. Mach. Learn. Knowl. MDPI. September 13, 2021.

[24] Towards Data Science, "Making PATE Bidirectionally Private" Agosto 2019 [Online]. Available: https://towardsdatascience.com/making-pate-bidirectionally-private-6d060f039227. Accesed on: Agosto. 20, 2021.

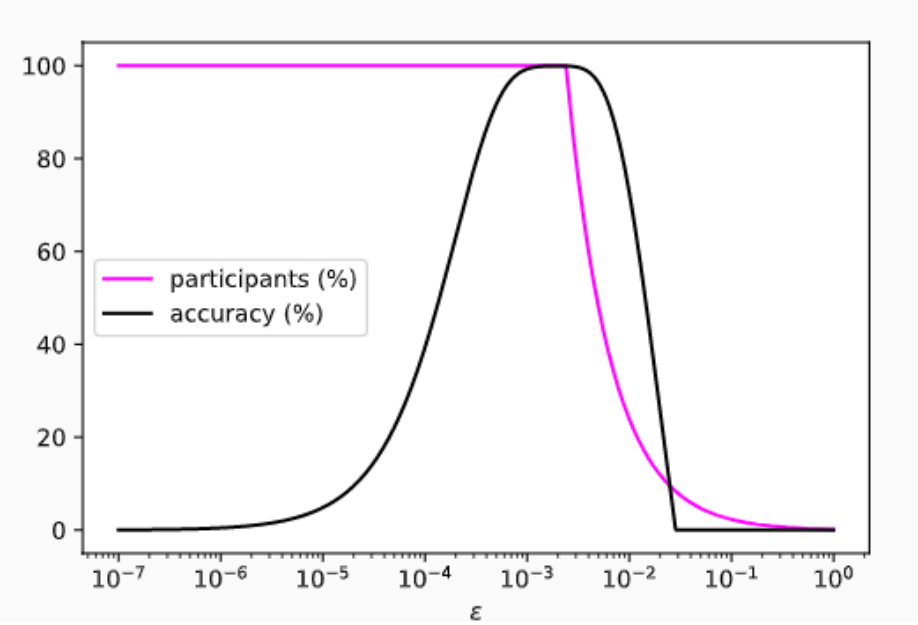
Anexo 1 – Incentivos internos para privacidad de los datos

(https://blog.openmined.org/internal-incentives-for-privacy/)

- ❖ Una mejor privacidad puede producir una mayor precisión:
  - ✓ En la privacidad diferencial, existe una compensación definida entre la privacidad y la precisión. La privacidad se mide mediante un parámetro llamado épsilon, la letra griega ε. Cuanto menor sea la épsilon, mayor será la garantía de privacidad. La privacidad se obtiene introduciendo ruido aleatorio. Las garantías más sólidas requieren más ruido. Por lo tanto, la compensación es inevitable: en igualdad de condiciones, una mayor privacidad implica una menor precisión. En este punto, uno podría llegar a la conclusión de que emplear tecnologías que preservan la privacidad siempre requiere sacrificar la precisión. Pero esta conclusión es falsa. Claro, en igualdad de condiciones, una mayor privacidad implica una menor precisión. Pero en el mundo real podría ser imposible cambiar las garantías de privacidad "en el vacío", sin cambiar nada más. Cambiar las garantías de privacidad puede afectar otros factores, que a su vez afectan la precisión.
- ❖ La privacidad puede mejorar la calidad de los datos:
  - ✓ La calidad de los datos afecta la precisión. Mejores datos producen una mayor precisión. Una técnica llamada respuesta aleatoria, basada en la misma intuición que la privacidad diferencial, puede ayudar a comprender cómo la privacidad puede mejorar la calidad de los datos.
  - ✓ Con respuesta aleatoria y privacidad diferencial, introducimos algo de ruido aleatorio en los datos. Pero conocemos las propiedades estadísticas de este ruido. Según la ley de los grandes números, si lanzamos una moneda normal muchas veces, saldrá cara aproximadamente el 50% de las veces. Entonces podemos corregir parcialmente este ruido. Sin privacidad todavía tendríamos algo de ruido en los datos, porque la gente respondería con menos honestidad. Pero no conoceríamos las propiedades estadísticas de ese ruido, por lo que no podríamos corregirlo. En ciertos casos,

introducir la privacidad como ruido controlado puede conducir a resultados más precisos.

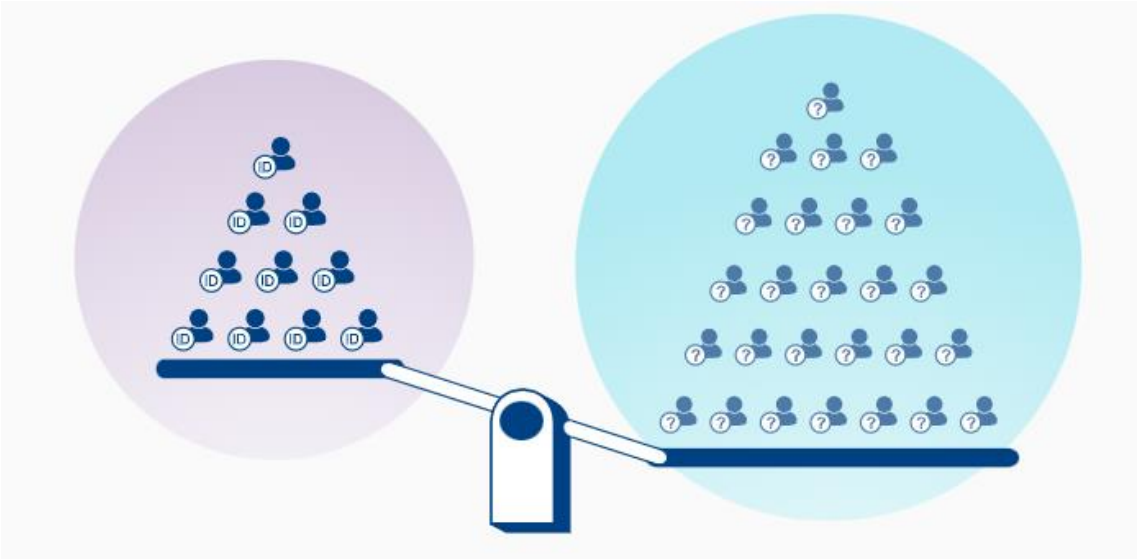
- La privacidad puede aumentar la cantidad de los datos:
  - ✓ La cantidad de datos es otro factor que afecta la precisión; por lo general, más datos conduce a una mayor precisión. ¿Hay situaciones en las que mejorar la privacidad nos pueda proporcionar más datos? Con seguridad. Se podría incentivar a los usuarios a utilizar un servicio, o proporcionar sus datos para análisis estadísticos, si se les ofrecieran buenas garantías de privacidad. Una vez más, esto se remonta al doble papel de la privacidad.
  - ✓ Como explicó el experto en privacidad diferencial Aaron Roth, podemos construir modelos económicos para examinar la relación entre privacidad y precisión. Con estos modelos podemos graficar la precisión en función de épsilon, el parámetro que indica la fuerza de las garantías de privacidad. Aquí tenemos un ejemplo de tales gráficos.



Analicemos lo que sucede en la trama, yendo de izquierda a derecha. En el extremo izquierdo de la parcela, ε es muy pequeño, lo que significa que estamos ofreciendo garantías de privacidad muy sólidas. Estamos introduciendo mucho ruido, por lo que la precisión es muy baja. Yendo hacia el centro de la parcela, ε se hace más grande; agregamos menos ruido, obteniendo mayor precisión.

Y ahora la parte más interesante. Yendo hacia la derecha, a medida que las garantías de privacidad se debilitan cada vez más, el número de participantes disminuye (la línea magenta en la trama). La gente valora la privacidad; al quitarlo, obtenemos menos gente. Con cada vez menos datos, la precisión también disminuye.

Si una mayor privacidad siempre implicaba una menor precisión, los valores más altos de precisión estarían en la parte correcta de la trama. En cambio, están en el medio. Incluso si solo le importa la precisión, emplear sólidas garantías de privacidad puede ser lo mejor.



En la imagen tenemos una escala que mide el valor informativo de dos escenarios. A la izquierda tenemos el escenario no privado: una pequeña cantidad de datos sin ruido. A la derecha, el escenario privado: una mayor cantidad de datos con algo de ruido de privacidad. En muchos casos podemos obtener más información en el escenario privado.

- La privacidad puede impulsar el progreso científico: los datos son alimento para la IA:
  - ✓ Los mayores saltos en el progreso se han producido con nuevos conjuntos de datos, no con nuevos modelos. Por ejemplo, uno de los mayores avances para las redes neuronales ocurrió con ImageNet, un enorme conjunto de datos para el reconocimiento de objetos. En muchos campos de la investigación científica no existen grandes conjuntos de datos públicos. No existe ImageNet para el cáncer o la depresión.

- ✓ Las tecnologías que preservan la privacidad podrían resolver este dilema. Con aprendizaje federado, los datos nunca abandonan el lugar donde se producen. En lugar de enviar los datos al algoritmo, enviamos el algoritmo a los datos. Podemos obtener información a partir de enormes conjuntos de datos agregados, sin realmente agregarlos.

❖ La privacidad puede arreglar los mercados de datos, impulsando la competencia y la innovación:

- ✓ Los datos se pueden copiar muy fácilmente. Siempre que le da a alguien una copia de sus datos, no puede controlar lo que hacen con ellos. Las leyes de datos como GDPR le brindan cierta protección, pero son muy difíciles de hacer cumplir. Esto se conoce como problema de copia.-
- ✓ En la práctica, cuando vende sus datos, el comprador se convierte instantáneamente en su competidor y sus datos pierden valor muy rápidamente. Como resultado, los mercados de datos son extraños y disfuncionales. Una estrategia es vender datos lo más rápido posible; en los mercados financieros, las empresas pagan por tener sus servidores lo más cerca posible de la infraestructura comercial central. Otra estrategia es nunca vender datos a nadie, si son demasiado valiosos o privados.
- ✓ La privacidad podría revolucionar los mercados de datos. Con el aprendizaje federado no es necesario transferir datos para extraer valor de ellos. Puede vender el uso de sus datos sin enviar copias de ellos. Los compradores no se convertirían instantáneamente en sus competidores. Con más valor proveniente de los datos, más empresas podrían considerar la posibilidad de abrir sus algoritmos, facilitando el intercambio de conocimientos y las auditorías independientes.
- ✓ Las nuevas empresas de aprendizaje automático también se beneficiarían. En este momento se enfrentan al dilema del huevo o la gallina de los usuarios y los datos: no se pueden tener usuarios sin un producto, pero para crear un producto se necesitan los datos de los usuarios. Muchas empresas emergentes de aprendizaje automático comienzan con la consultoría, con la esperanza de que otra empresa invierta en ellas y les permita usar sus datos. En los nuevos mercados de datos, las nuevas

empresas podrían simplemente pagar a otras empresas para que usen sus datos.

- ✓ Surgirían modelos comerciales nuevos y repetibles. Los propietarios del mismo tipo de datos podrían formar redes de datos federadas, que necesitarían guardianes. Un ejemplo en la atención médica podría ser el guardián de la red de radiología: contactar a los hospitales, configurar la infraestructura para el uso remoto de sus datos de rayos X y los mecanismos para compensar dicho uso.

❖ La privacidad puede empoderar a los usuarios, lo que lleva a una IA más alineada

- ✓ En los nuevos mercados de datos que acabamos de describir, los primeros propietarios de datos serán las organizaciones que recopilan datos sobre sus clientes o usuarios. En algún momento, los propios interesados podrían convertirse en propietarios de los datos que producen. Las personas podrían ser propietarias de sus datos personales y recibir un pago siempre que permitan su uso. Esto proporcionaría un incentivo financiero contra el intercambio de copias de datos; Debido al problema de las copias, compartir copias podría significar perder ingresos futuros.
- ✓ Por otro lado, las empresas de tecnología necesitan datos de los usuarios. Con más control y agencia sobre sus datos, los usuarios obtendrán poder de negociación. Las empresas de tecnología tendrán un incentivo más fuerte para crear productos alineados con los intereses y valores de los usuarios.

Anexo 2 – Computación Confidencial

(https://www.microsoft.com/en-us/research/blog/privacy-preserving-machine-learning-maintaining-confidentiality-and-preserving-trust/)

La "computación confidencial" fue introducida con Azure mediante máquinas virtuales (VM) que se ofrecen en la nube y con soporte de contenedores confidenciales en Kubernetes para que se ejecuten cargas de trabajo sensibles dentro de entornos de ejecución de confianza (TEE). Microsoft también es miembro fundador del Confidential Computing Consortium (CCC), un grupo que reúne a fabricantes de hardware, proveedores de nube y proveedores de soluciones para trabajar conjuntamente en formas de mejorar y estandarizar la protección de datos en la industria de la tecnología.

Fundamentos informáticos confidenciales de Azure:

El estándar de confidencialidad de Microsoft, alinea y amplía el estándar establecido por la CCC para proporcionar una base integral para la informática confidencial. Azure cuenta con ofertas informáticas confidenciales que van más allá del aislamiento del hipervisor entre los inquilinos del cliente para ayudar a proteger los datos del cliente del acceso de los operadores de Microsoft. También cuenta con informática confidencial con enclaves seguros para ayudar, además, a evitar el acceso de los operadores del cliente.

Lo antes mencionado incluye:

- Raíz de confianza de hardware para garantizar que los datos estén protegidos y anclados en el silicio. La confianza se basa en el fabricante del hardware, por lo que ni siquiera los operadores de Microsoft pueden modificar las configuraciones del hardware.
- Certificación remota para que se verifique directamente la integridad del medio ambiente. Se puede verificar que tanto el hardware como el software en el que se ejecutan las cargas de trabajo son versiones aprobadas y protegidas antes de permitirles acceder a los datos.
- El lanzamiento confiable es el mecanismo que garantiza el arranque de las máquinas virtuales con software autorizado y que utiliza la certificación remota para que los clientes puedan verificar. Está disponible para todas las máquinas

virtuales, incluidas las confidenciales, y ofrece arranque seguro y TPM para agregar defensa contra rootkits, bootkits y firmware malicioso.

- ❖ Aislamiento y cifrado de la memoria para garantizar que los datos estén protegidos durante el procesamiento. Azure ofrece aislamiento de memoria por VM, contenedor o aplicación para satisfacer las diversas necesidades de los clientes, y cifrado basado en hardware para evitar la visualización no autorizada de datos, incluso con acceso físico en el centro de datos.
- ❖ Administración segura de claves para garantizar que las claves permanezcan encriptadas durante su ciclo de vida y se publiquen solo para el código autorizado.
- ❖ Los componentes anteriores juntos forman las bases de lo que Microsoft considera informática confidencial.

Azure ofrece el servicio de VM confidenciales, en la nube, donde no sólo el software está protegido, sino también el hardware. Para asegurar este cifrado y la seguridad requerida, las VMs deben estar disponibles en Azure Kubernetes Service (AKS) como nodo trabajador.

Las opciones de aislamiento y cifrado de memoria de Azure brindan protecciones más sólidas y completas para los datos de los clientes que cualquier otra nube.

Azure puede ofrecerle a PATE VMs seguras y cifradas, donde cada institución puede correr sus experimentos de forma segura y disponibilizar los resultados para futuras consultas. A través de la certificación remota los clientes pueden verificar ellos mismos que los ambientes son seguros. El hardware es confiable debido a que son sólo los propios fabricantes los que pueden hacer modificaciones. A través del lanzamiento confiable se asegura que no se utilicen softwares no autorizados, y se obtiene certificación remota. Con el aislamiento y cifrado de la memoria los datos están protegidos durante las corridas. Las claves están encriptadas durante todo el proceso, lo que nos da la tranquilidad de que no se puedan descifrar fácilmente.

Privacy Preserving Machine Learning – PPML con Azure

La iniciativa PPML se inició en asociación entre Microsoft Research y los equipos de productos de Microsoft con el objetivo de proteger la confidencialidad y privacidad de

los datos al entrenar modelos de lenguaje de gran capacidad. El objetivo de la iniciativa PPML es mejorar las técnicas existentes y desarrollar otras nuevas para proteger la información sensible que funciona tanto para las personas como para las empresas. Esto ayuda a garantizar que nuestro uso de datos proteja la privacidad de las personas y que los datos se utilicen de manera segura, evitando la filtración de información confidencial y privada.

Un enfoque holístico de PPML

- Comprender: Se intenta comprender el riesgo de fuga de datos del cliente y posibles ataques a la privacidad de una manera que ayude a determinar las propiedades de confidencialidad de las canalizaciones de ML. Además, es fundamental alinearse de manera proactiva con los responsables de la formulación de políticas sobre protección de datos.
- Medir: Una vez que se comprenden los riesgos para la privacidad y los requisitos que se deben cumplir, se definen métricas que pueden cuantificar los riesgos identificados y realizar un seguimiento del éxito para mitigarlos.
- Mitigar: La privacidad diferencial (DP) es una de las herramientas utilizadas para mitigar los riesgos. Después de aplicar las estrategias de mitigación se debe medir su éxito y utilizar los hallazgos para refinar el enfoque de PPML, en definitiva, mitigar la filtración de información privada.

Privacidad diferencial utilizando Azure

Microsoft fue pionera en la investigación de DP en 2006, y desde entonces DP se ha establecido como el estándar de privacidad de facto, con una gran cantidad de literatura académica y un número creciente de implementaciones a gran escala en la industria (por ejemplo, DP en telemetría de Windows o DP en Microsoft Viva Insights) y gobierno.

En escenarios de ML, DP funciona agregando pequeñas cantidades de ruido estadístico durante el entrenamiento, cuyo propósito es ocultar las contribuciones de las partes individuales cuyos datos se están utilizando. Cuando se emplea DP, una prueba matemática valida que el modelo de ML final aprende solo las tendencias generales en los datos sin adquirir información exclusiva de ninguna parte específica. Los cálculos

diferencialmente privados implican la noción de un presupuesto de privacidad, ϵ, que impone un límite superior estricto a la información que podría filtrarse del proceso. Esto garantiza que, independientemente de la información auxiliar que pueda poseer un adversario externo, su capacidad para aprender algo nuevo sobre cualquier parte individual cuyos datos se utilizaron en el entrenamiento del modelo es muy limitada.

En el artículo de Microsoft Research (https://www.microsoft.com/en-us/research/publication/numerical-composition-of-differential-privacy/) "Composición numérica de la privacidad diferencial", se relata el desarrollamos un nuevo contador de DP que brinda un resultado más preciso para el presupuesto de privacidad gastado al capacitar sobre los datos del usuario. Esto es particularmente importante cuando se imparte formación sobre datos empresariales, en los que normalmente hay muchos menos participantes en el conjunto de datos. Con el nuevo contador de DP, se espera poder entrenar modelos por más tiempo, logrando así una mayor utilidad mientras se utiliza el mismo presupuesto de privacidad.

La opción de utilizar un mejor contador de DP sería importante, tomando en cuenta que la cantidad de organizaciones y datos con los que se puede trabajar en el contexto de nuestro problema es limitada (pocas organizaciones y pocos datos). Debido a que esta herramienta no es de código abierto no fue posible hacer una evaluación de la misma, se recomienda profundizar en futuros trabajos.

Consideramos de gran utilidad las funcionalidades de DP, ya que, a través del agregado de ruido estadístico en el entrenamiento, se ocultan las contribuciones de las partes individuales. También a través de una prueba matemática se asegura que los modelos aprenden en cada corrida y no memorizan resultados anteriores. DP incluye la noción de presupuesto de privacidad, ϵ, que sería el límite de privacidad que se estaría dispuesto a perder, y se asegura que no exista ninguna corrida que sobrepase ese límite. Este concepto es muy importante para darle tranquilidad a las instituciones, que no perderán privacidad por encima del máximo estipulado por cada una. Además, el contador de DP podría indicar el número exacto de cuánta privacidad se ha perdido, aunque no fue probado.

Estas funcionalidades son de gran utilidad e importancia para cualquiera de los tres escenarios planteados por nuestra parte en el capítulo 5, ya que el agregado de ruido, la

seguridad requerida y el presupuesto de privacidad establecido junto el contador de DP, son los máximos requerimientos de seguridad que se pueden brindar para que las instituciones estén dispuestas a brindar sus datos con total tranquilidad. Vamos a centrar nuestra investigación de esta herramienta de Azure, de forma teórica, ya que está aún en desarrollo y tiene un costo asociado.

También vamos a tratar de continuar la investigación de posibles combinaciones de herramientas que garanticen la mayor seguridad y la obtención de los mejores resultados durante las corridas.

Anexo 3 - Aprendizaje Federado conceptos básicos

(https://blog.openmined.org/what-is-federated-learning/)
En este anexo de la serie introductoria sobre aprendizaje automático privado, presentaremos el aprendizaje federado (FL), explicando qué es FL, cuándo usarlo y cómo implementarlo con herramientas OpenMined. La información de este artículo será digerible para una amplia audiencia, pero sección por sección, profundizaremos más en las malas hierbas para comprender y usar el aprendizaje federado.

Para obtener más información sobre la serie, consulte el artículo de introducción o eche un vistazo a las otras publicaciones para obtener más información sobre las técnicas que pueden habilitar el aprendizaje automático para preservar la privacidad con las bibliotecas de OpenMined.

Introducción

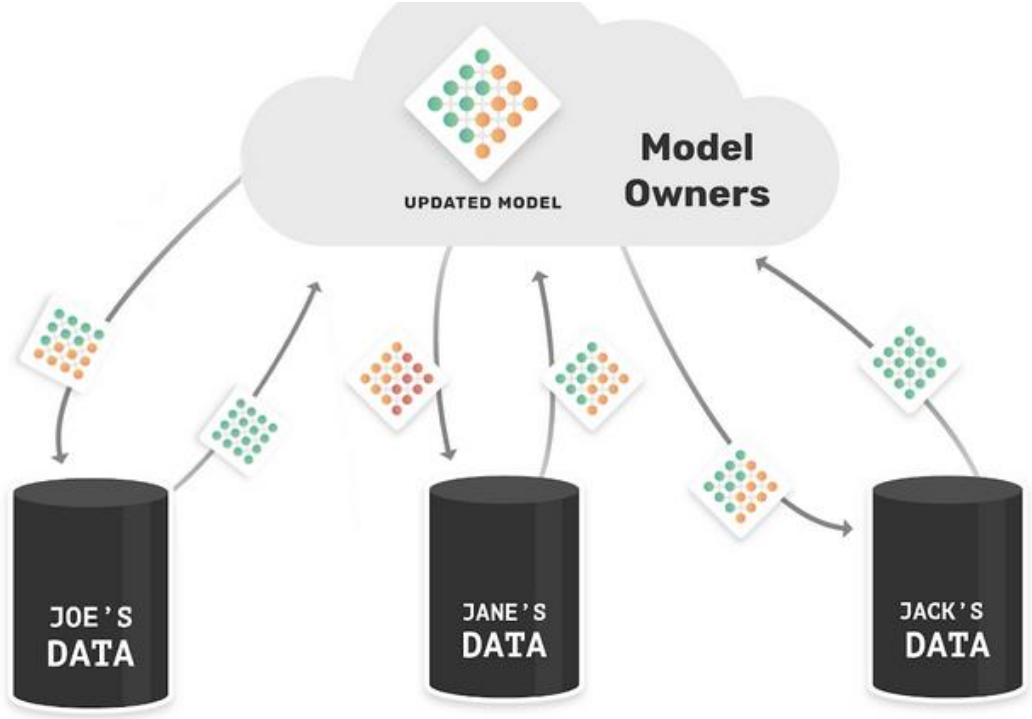
Propuesto inicialmente en 2015, el aprendizaje federado es una solución algorítmica que permite el entrenamiento de modelos ML al enviar copias de un modelo al lugar donde residen los datos y realizar el entrenamiento en el borde, eliminando así la necesidad de mover grandes cantidades de datos a un servidor central con fines de formación.

Los datos permanecen en sus dispositivos de origen, también conocidos como clientes, que reciben una copia del modelo global del servidor central. Esta copia del modelo global se entrena localmente con los datos de cada dispositivo. Los pesos del modelo se actualizan mediante entrenamiento local y luego la copia local se envía de vuelta al servidor central. Una vez que el servidor recibe el modelo actualizado, procede a agregar las actualizaciones, mejorando el modelo global sin revelar ninguno de los datos privados sobre los que fue entrenado.

Casos de uso

Una de las primeras aplicaciones de FL fue mejorar la recomendación de palabras en el teclado de Android de Google sin cargar los datos, es decir, el texto de un usuario, en la nube. Más recientemente, Apple ha detallado cómo emplea el aprendizaje federado para mejorar el reconocimiento de voz de Siri. Además, de manera intuitiva, mantener los

datos en su origen es valioso en cualquier aplicación que preserve la privacidad, especialmente cuando se aplica en el cuidado de la salud o en datos confidenciales en empresas y gobiernos.



Ventajas:

- Los investigadores pueden entrenar modelos utilizando datos privados y confidenciales sin tener que preocuparse por el manejo de los datos: los datos permanecen en el dispositivo y solo las actualizaciones de modelos aprendidas se transfieren entre el laboratorio y los propietarios de los datos.
- Cumple con las regulaciones de protección de datos como GDPR.

Desventajas:

- El costo de implementar el aprendizaje federado es más alto que recopilar la información y procesarla de manera centralizada, especialmente durante las primeras fases de I + D, cuando el método y el proceso de capacitación aún se están repitiendo.

- Requiere que los propietarios de datos realicen cálculos en el dispositivo que contiene los datos; para algunos dispositivos con capacidad de cálculo limitada, esto puede no ser posible o económico.
- La implementación de FL no es suficiente para garantizar la privacidad, ya que las actualizaciones del modelo pueden incluir rastros que se pueden usar para inferir información privada y sensible, por lo que es necesario combinarlo con otras técnicas.

Implementación

Para comenzar, usaremos el conjunto de datos MNIST clásico que sustituirá a los datos de nuestros clientes, PySyft proporcionará todos los componentes necesarios para hacer una demostración del aprendizaje federado y probarlo localmente en este conjunto de datos.

Si quieres imaginar una aplicación razonablemente cercana, podríamos concebir que los caracteres MNIST son parte de las firmas digitales de nuestros clientes, producidos al firmar documentos en su smartphone y nos gustaría utilizarlos para entrenar un modelo de reconocimiento de caracteres. En este escenario, nos gustaría brindar sólidas garantías de privacidad a nuestros usuarios al no cargar sus firmas en un servidor central.

En PySyft, los dispositivos de los clientes, es decir, las entidades que realizan el entrenamiento del modelo, se denominan trabajadores.

```
bob = sy.VirtualWorker(hook, id="bob")  # <-- define remote worker bob
alice = sy.VirtualWorker(hook, id="alice")  # <-- remote worker alice
```

En este punto, en un ejemplo simulado, tenemos que enviar los datos a los trabajadores mediante el cargador de datos federado de PySyft.

```
federated_train_loader = sy.FederatedDataLoader( # <-- this is now a FederatedDataLoader
    datasets.MNIST('../data', train=True, download=True,
                   transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ]))
    .federate((bob, alice)), # <-- NEW: we distribute the dataset across all the workers, it's now a Fe
    batch_size=batch_size, shuffle=True)
```

El cargador de datos federado, al igual que el cargador de datos de torch estándar en el que se basa, permite la carga diferida durante el entrenamiento y, por lo tanto, en la función de entrenamiento, debido a que los lotes de datos ahora se distribuyen entre Alice y Bob, debe enviar el modelo a la ubicación correcta para cada lote usando model.send (...). Luego, realiza todas las operaciones de forma remota con la misma sintaxis como si estuviera haciendo todo localmente en PyTorch. Cuando haya terminado, recuperará el modelo actualizado y la pérdida que se puede registrar usando el método .get ().

```
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(federated_train_loader): # <-- now it is a distributed d
        model.send(data.location) # <-- NEW: send the model to the right location
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        model.get() # <-- NEW: get the model back
        if batch_idx % args.log_interval == 0:
            loss = loss.get() # <-- NEW: get the loss back
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * args.batch_size, len(train_loader) * args.batch_size, #batch_idx * l
                100. * batch_idx / len(train_loader), loss.item()))
```

En este punto, el modelo ha sido entrenado con los datos de Alice y Bob por los respectivos trabajadores, y sus datos no han salido de sus dispositivos.

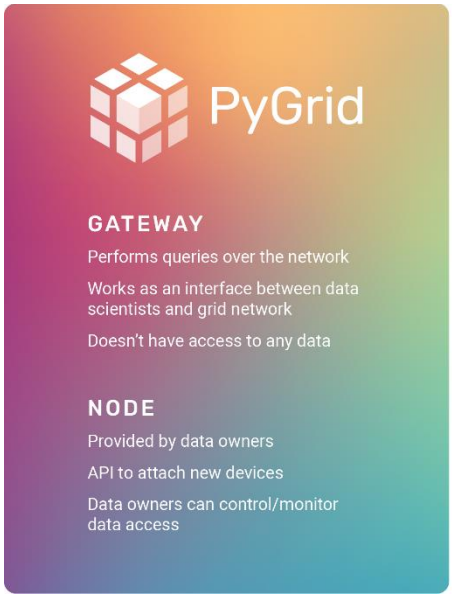
Sin embargo, el aprendizaje federado por sí solo no es suficiente para garantizar la privacidad porque, al utilizar las actualizaciones del modelo, un servidor "honesto pero curioso" podría reconstruir las muestras a partir de las cuales se calcularon las actualizaciones. Aquí es donde la Computación Segura MultiParty, el cifrado

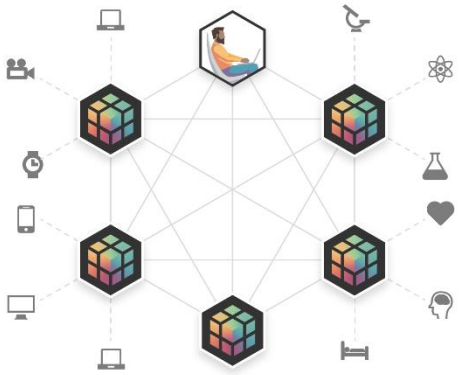
homomórfico y la privacidad diferencial llegan a brindar mayores garantías de seguridad a los propietarios de los datos. Exploraremos todos estos temas en esta serie.

PySyft + PyGrid para el aprendizaje federado a escala

Como mencionamos anteriormente, la visión más amplia de la tecnología va más allá de cualquier aplicación o servicio. Con la ayuda de datos de aprendizaje federados, los propietarios pueden mantener más fácilmente el control de sus datos que pueden usarse para entrenar modelos sin salir de los sistemas de los propietarios. Estas garantías, además de ser positivas para todos los usuarios de aplicaciones con uso intensivo de datos, tienen el potencial de poner a disposición conjuntos de datos completamente nuevos en sectores como la atención médica, donde, para seguir HIPPA o las disposiciones relacionadas con la salud de GDPR, la privacidad es la máxima prioridad.

Para contribuir a hacer realidad esta visión, OpenMined está trabajando en PyGrid, una plataforma peer-to-peer que utiliza el marco PySyft para el aprendizaje federado y la ciencia de datos.





Los propietarios de datos y los científicos de datos pueden conectarse en la plataforma, donde los propietarios de datos pueden sentirse seguros sabiendo que sus datos nunca saldrán de su nodo, y los científicos de datos pueden realizar su análisis sin infringir los derechos de privacidad de nadie.

Hoy en día, este tipo de interacción podría llevar de semanas a meses en los sectores que trabajan con datos confidenciales, pero con PyGrid todo podría estar a solo unas pocas líneas de código. Para obtener más información sobre PyGrid, aquí hay una inmersión más profunda en la plataforma y los casos de uso que permite.

Anexo 4 – Aprendizaje Federado ejemplo

(https://blog.openmined.org/what-is-pygrid-demo/)
Como líder del equipo PyGrid, Ionésio Lima Da Costa Junior está construyendo una plataforma peer-to-peer para ciencia de datos privados y aprendizaje federado. Este artículo es un resumen de la presentación AMA de Ionesio en febrero de 2020. Ionesio es un investigador de IA cuyo trabajo cuenta con el apoyo del programa de subvenciones OpenMined / RAAIS y encargado por la Universidad de Oxford y el Equipo de tareas de privacidad de las Naciones Unidas.

¿Qué pasaría si pudiera entrenar con todos los datos del mundo, sin que esos datos salieran del dispositivo y manteniendo la privacidad de esos datos?

PyGrid es una plataforma peer-to-peer para ciencia de datos privados y aprendizaje federado. Con PyGrid, los propietarios de datos pueden proporcionar, monitorear y administrar el acceso a sus propios clústeres de datos privados. Los datos no salen del servidor del propietario de los datos.

Los científicos de datos pueden usar PyGrid para realizar análisis estadísticos privados en el conjunto de datos privados, o incluso realizar aprendizaje federado en múltiples conjuntos de datos de instituciones.

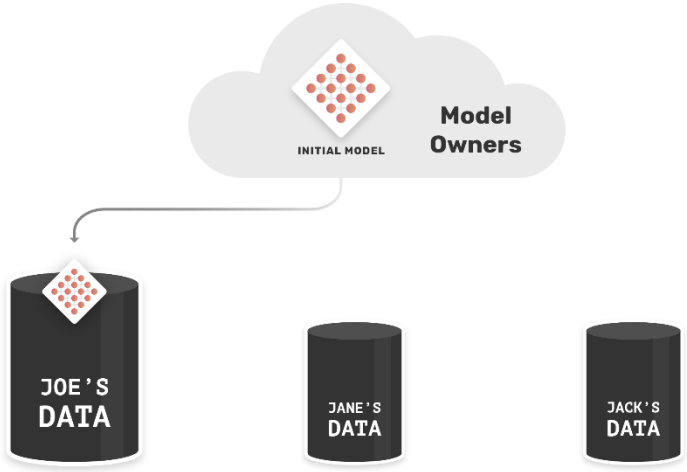
Esta publicación de blog cubrirá:

- Conceptos básicos necesarios para comprender PyGrid, como el aprendizaje federado y la computación segura multipartita
- La biblioteca PySyft y la plataforma PyGrid: PySyft es una biblioteca privada de aprendizaje automático que se implementa utilizando la plataforma PyGrid.
- Varios ejemplos prácticos de análisis de preservación de la privacidad utilizando PyGrid: estos ejemplos nos ayudarán a comprender la arquitectura de PyGrid y descubrir cómo se puede aplicar a problemas reales
- Hoja de ruta de OpenMined para el desarrollo de PyGrid en 2020

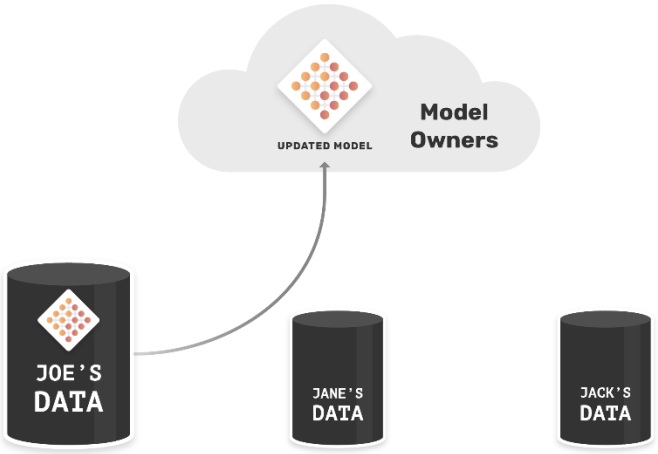
Aprendizaje federado

El primer concepto que debemos comprender es el aprendizaje federado. El aprendizaje federado es una técnica que permite que los modelos de IA aprendan sin que los usuarios tengan que ceder sus datos. ¿Como funciona esto?

El primer paso es crear un modelo inicial. Luego, el científico de datos enviaría este modelo al propietario del conjunto de datos, en este caso, el dispositivo de Joe.

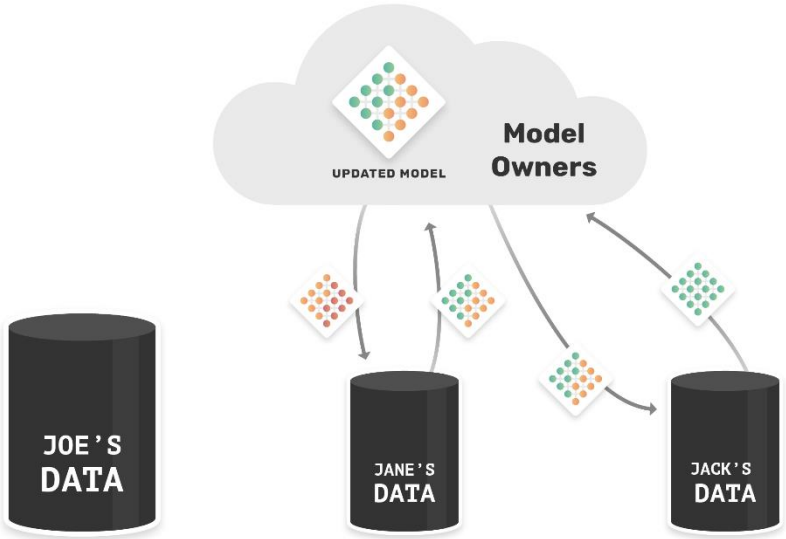


Ahora, Joe puede actualizar este modelo entrenándolo en su conjunto de datos. Después del entrenamiento, el modelo actualizado volverá a la empresa de IA.



Ahora, AI Incorporated envía el modelo de AI actualizado a otro dispositivo, en este caso el dispositivo de Jane.

Jane's actualizará este modelo entrenándolo en su conjunto de datos. Una vez que el modelo se entrena con los datos de Jane, Jane envía los pesos actualizados al científico de datos.



Ahora el modelo ha aprendido de los datos de Joe y Jane. Podemos repetir este proceso en muchos nodos e incluso podemos entrenar modelos simultáneamente en varios nodos y promediarlos, lo que permite mejoras más rápidas en el modelo.

Los principales beneficios del aprendizaje federado son:

- Los datos de entrenamiento permanecen en los dispositivos del usuario (o, por ejemplo, en los servidores del hospital).
- Esto aumenta la privacidad de los datos sensibles.
- Reduce la responsabilidad legal de los propietarios de modelos (científicos de datos, empresas)
- Reduce el ancho de banda de la red involucrado en la carga de grandes conjuntos de datos

Es fácil pensar en posibles casos de uso de esta tecnología. Por ejemplo,

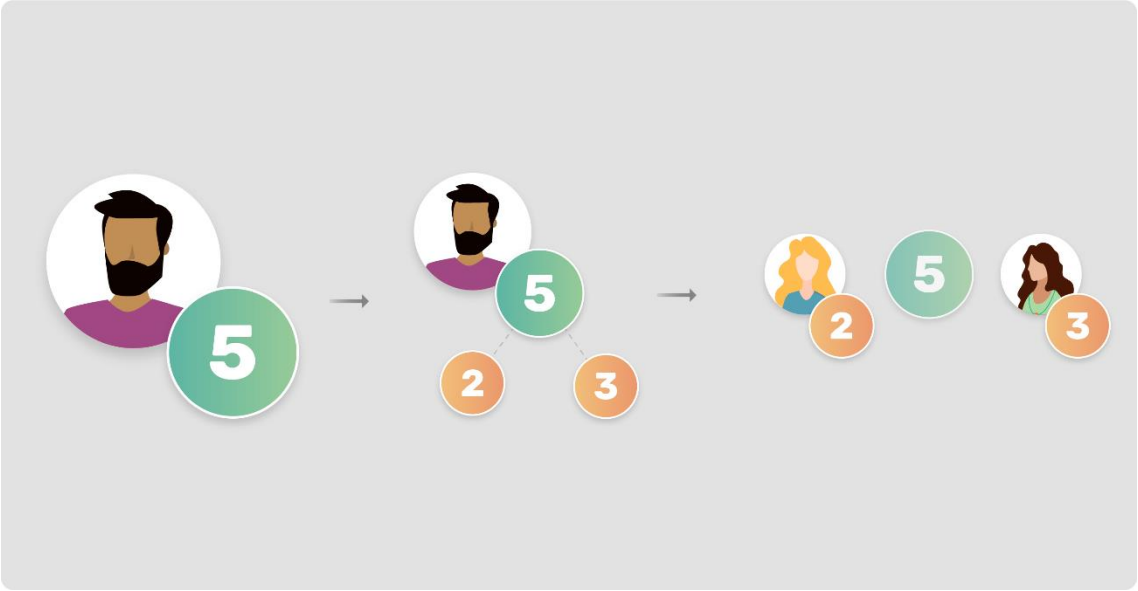
- Una formación científica sobre datos de varios hospitales.
- Una aplicación de capacitación para teléfonos inteligentes sobre datos de varios teléfonos

- ❖ Una empresa que se inclina por predecir datos cuando sus máquinas necesitan mantenimiento mediante el entrenamiento de modelos a través de datos de muchos sensores.

Computación segura multipartita

Secure Multi-Party Computation (SMPC) es una forma diferente de cifrar datos y compartirlos con diferentes dispositivos. La principal ventaja es que, a diferencia de la criptografía tradicional, SMPC nos permite realizar operaciones lógicas y aritméticas utilizando datos cifrados.

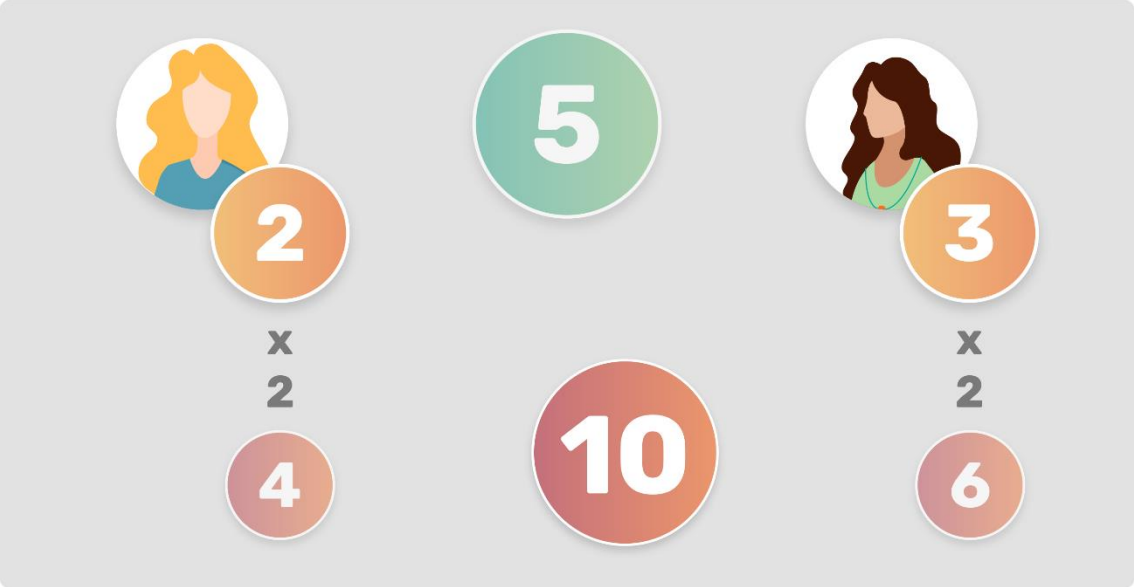
¿Cómo se pueden realizar las matemáticas con la computación multipartita? A continuación, se muestra un ejemplo (muy) simplificado de cómo funciona esto:



En este ejemplo, tenemos a Andrew con su número, en este caso es el propietario del número 5, sus datos personales. Andrew puede anonimizar sus datos descomponiendo su número en 2 (o más) números diferentes. En este caso, descompone el número 5 en 2 y 3. De esa manera, puede compartir sus datos anonimizados con sus amigos Marianne y Bob.

En este caso, ninguno de ellos conoce realmente el valor real de los datos de Andrew. Tienen solo una parte. Cualquiera de ellos puede realizar cualquier tipo de operación sin el consentimiento de todos ellos. Pero, aunque estos números estén encriptados entre

ellos, aún podremos realizar cálculos. De esa manera, podemos utilizar valores cifrados para calcular los datos del usuario sin mostrar ningún tipo de información confidencial.



Con estos conceptos entendidos, ahora podemos explicar PySyft y PyGrid.

PySyft es una biblioteca de Python para el aprendizaje profundo seguro y privado. PySyft tiene como objetivo proporcionar herramientas de preservación de la privacidad dentro de los principales marcos de aprendizaje profundo como PyTorch y TensorFlow. De esa manera, los científicos de datos pueden usar estos marcos para administrar cualquier tipo

de datos sensibles aplicando conceptos de preservación de la privacidad, sin tener que ser expertos en privacidad y ellos mismos.

PyGrid tiene como objetivo ser una plataforma de igual a igual que utiliza el marco PySyft para el aprendizaje federado y la ciencia de datos.

La arquitectura se compone de dos componentes: Gateways y Nodes. El componente Gateway funciona como un DNS, enrutando los nodos que proporcionan los conjuntos de datos deseados.

Los nodos son proporcionados por los propietarios de los datos: son grupos de datos privados que serán administrados y monitoreados por sus propietarios de datos. Los datos no salen del servidor del propietario de los datos.

Los científicos de datos pueden usar PyGrid para realizar análisis estadísticos privados en ese conjunto de datos, o incluso realizar un aprendizaje federado en varios conjuntos de datos de la institución.

A continuación, explicamos cómo se puede realizar cada uno de esos casos de uso.

CASO DE USO 1: ANÁLISIS ESTADÍSTICO PRIVADO

Exploremos dos flujos de trabajo:

• El propietario de los datos que quiere publicar sus datos sensibles en su nodo. (En este caso, una sala de pediatría de un hospital).

• El científico de datos que quiere encontrar un conjunto de datos específico en la red de cuadrícula para calcular algún análisis estadístico.

El propietario de los datos

Paso 1: Importar PySyft y las dependencias

El primer paso como propietario de datos es importar nuestras dependencias.

En este caso, importaremos syft y reemplazaremos los módulos de Torch estándar usando syft hook.

```
In [1]: import syft as sy
        from syft.workers.node_client import NodeClient
        import torch as th
        hook = sy.TorchHook(th)
```

Paso 2: conecta tu nodo

El siguiente paso es conectarse con su propio nodo. Es importante tener en cuenta que la aplicación de nodo se implementó en algún entorno y es necesario conocer su dirección previamente. En este caso, nos conectaremos con el nodo del hospital.

```
In [2]: hospital_datacluster = NodeClient(hook, "ws://localhost:3000")
        hospital_datacluster
Out[2]: Federated Worker < id: hospital-datacluster >
```

Paso 3: Prepare los datos como tensores y agregue una breve descripción

Ahora, debemos preparar nuestro conjunto de datos para que se publique en el nodo del hospital. Para proporcionar una comprensión clara sobre los datos que queremos publicar, debemos agregar una breve descripción que explique el significado y la estructura de los datos. En este caso de uso, queremos publicar los registros de nacimiento mensuales del hospital.

```
In [3]: data_description = """Description:
                            This dataset represents the birth records for the month of February.
                            The data is arranged in the following format:

                            Columns:
                                Gender: 0-Male, 1-Female
                                Weight (Kilograms): Float value
                                Height (centimeters): Float value

                            Shape: (5 x 3)"""

        monthly_birth_records = th.tensor([[ 1, 3.5, 47.3],
                                           [ 0, 3.7, 48.1],
                                           [ 0, 3.9, 50.0],
                                           [ 1, 4.1, 52.3],
                                           [ 0, 4.1, 49.7]])
```

Paso 4: definir las reglas de acceso y los permisos

Después de eso, necesitamos definir reglas para controlar el acceso a los datos. En este caso, permitimos que algunos usuarios (Bob, Ana y Alice) tengan acceso total a los valores reales de estos datos.

```
In [4]: private_dataset = monthly_birth_records.private_tensor(allowed_users=("Bob",
                                                                              "Ana",
                                                                              "Alice"))
```

Paso 5: agregue etiquetas y rótulos para ayudar a los científicos de datos a encontrar su conjunto de datos

Para que nuestros datos sean accesibles desde consultas, también necesitamos agregar etiquetas para identificarlos y etiquetarlos.

En este ejemplo, agregamos dos etiquetas: #Febrero para identificar el mes y # registros de nacimiento para identificar el significado de los datos.

```
In [5]: private_dataset = private_dataset.tag("#February",
                                                "#birth-records").describe(data_description)
        private_dataset
Out[5]: (Wrapper)>PrivateTensor>tensor([[ 1.0000,  3.5000, 47.3000],
                [ 0.0000,  3.7000, 48.1000],
                [ 0.0000,  3.9000, 50.0000],
                [ 1.0000,  4.1000, 52.3000],
                [ 0.0000,  4.1000, 49.7000]])
```

Paso 6: ¡Publica! Ya terminaste.

Ahora, los datos están listos para ser publicados. Es importante tener en cuenta que debe poder publicar conjuntos de datos privados en este nodo. En este ejemplo, usamos la credencial de Bob para publicar estos datos en el nodo.

```
In [6]: data_pointer = private_dataset.send(hospital_datacluster,user="Bob")
        data_pointer
Out[6]: (Wrapper)>[PointerTensor | me:82280449211 -> hospital-datacluster:71870130091]
                Tags: #February #birth-records
                Shape: torch.Size([5, 3])
                Description: Description:...
```

Como propietario de los datos, ¡eso es todo lo que tenemos que hacer!

El científico de datos

Paso 1: Importar PySyft y las dependencias

Como científico de datos, también necesitamos importar la biblioteca syft y reemplazar los módulos de antorcha usando syft hook.

```
In [1]: import syft as sy
        from syft.grid.public_grid import PublicGridNetwork
        import torch as th
        hook = sy.TorchHook(th)
```

Paso 2: Conéctese a la plataforma Grid

```
In [2]: grid = PublicGridNetwork(hook, "http://localhost:5000")
```

A diferencia de los propietarios de los datos, no sabemos dónde están los nodos y los conjuntos de datos, por lo que primero debemos conectarnos con GridNetwork. La dirección de la red de la red será la dirección del componente de la puerta de enlace.

Paso 3: ¿Qué datos estás buscando? Busca en la red.

```
In [3]: results = grid.search("#February", "#birth-records")
        results
Out[3]: {'hospital-datacluster': [(Wrapper)>[PointerTensor | me:59911064729 -> hospital-datacluste
        r:38625356599]
                Tags: #birth-records #February
                Shape: torch.Size([5, 3])
                Description: Description:...]}
```

Después de conectarnos con la red de cuadrícula, podemos buscar las etiquetas del conjunto de datos deseadas. Quizás esté buscando radiografías de neumonía o registros de nacimiento del hospital. En este ejemplo, usamos las mismas etiquetas que publicamos antes. La red de cuadrícula devolverá un diccionario que contiene los identificadores de los nodos como claves y punteros de datos como valores.

Paso 4: crea una referencia a ese puntero de datos

A continuación, definimos una referencia directa al puntero de datos del hospital.

```
In [4]: feb_records = results['hospital-datacluster'][0]
```

Paso 5: comprender y explorar los datos

Comprender los datos con los que está trabajando es fundamental para cualquier científico de datos. A continuación, podemos explorar los indicadores de datos para comprender su significado y cómo están organizados.

```
In [5]: print(feb_records.description)

        Description:
                                    This dataset represents the birth records for the month of Februar
        y.
                                    The data is arranged in the following format:

                                    Columns:
                                        Gender: 0-Male, 1-Female
                                        Weight (Kilograms): Float value
                                        Height (centimeters): Float value

                                    Shape: (5 x 3)
```

Espera, ¿qué pasa si intento copiar esos datos?

Si intentamos recuperar los valores reales del puntero de datos sin estar permitido, se generará una excepción. Así es como PyGrid puede mantener los datos en manos del propietario y habilitarlos con el control para permitir o denegar el acceso a las muestras de datos.

```
In [6]: feb_records.get()

        ---------------------------------------------------------------------------
        GetNotPermittedError                      Traceback (most recent call last)
        <ipython-input-6-201bdd0cc151> in <module>
        ----> 1 feb_records.get()
```

Paso 6: realiza tus cálculos

Incluso sin copiar los datos, todavía podemos realizar cálculos remotos sobre estos datos. En este ejemplo, queremos calcular el promedio del peso y la altura de los bebés. Para hacer esto, necesitamos sumar los valores de las columnas de forma remota.

```
In [7]: def sum_column(dataset, column):
            sum_result = dataset[0][column].copy()
            for i in range(1,dataset.shape[0]):
                sum_result += dataset[i][column]
            return sum_result
```

Ahora, podemos calcular la suma del peso de forma remota. Generará otro tensor remoto.

```
In [8]: weight_sum = sum_column(feb_records, 1)
        weight_sum

Out[8]: (Wrapper)>[PointerTensor | me:55164810828 -> hospital-datacluster:61829325017]
```

Podemos hacer lo mismo con la columna de altura.

```
In [9]: height_sum = sum_column(feb_records, 2)
        height_sum

Out[9]: (Wrapper)>[PointerTensor | me:57021990033 -> hospital-datacluster:3864430273]
```

Ahora, solo necesitamos recuperar el valor agregado usando nuestras credenciales y dividir el valor entre 5, que es el tamaño del conjunto de datos.

```
In [10]: avg_weight = weight_sum.get(user="Bob") / 5
         avg_weight

Out[10]: (Wrapper)>PrivateTensor>tensor(3.8600)
```

Podemos hacer lo mismo para obtener el promedio de altura. De esa manera, podemos calcular el peso y la altura promedio de los bebés que nacieron en este mes sin tener acceso a datos confidenciales.

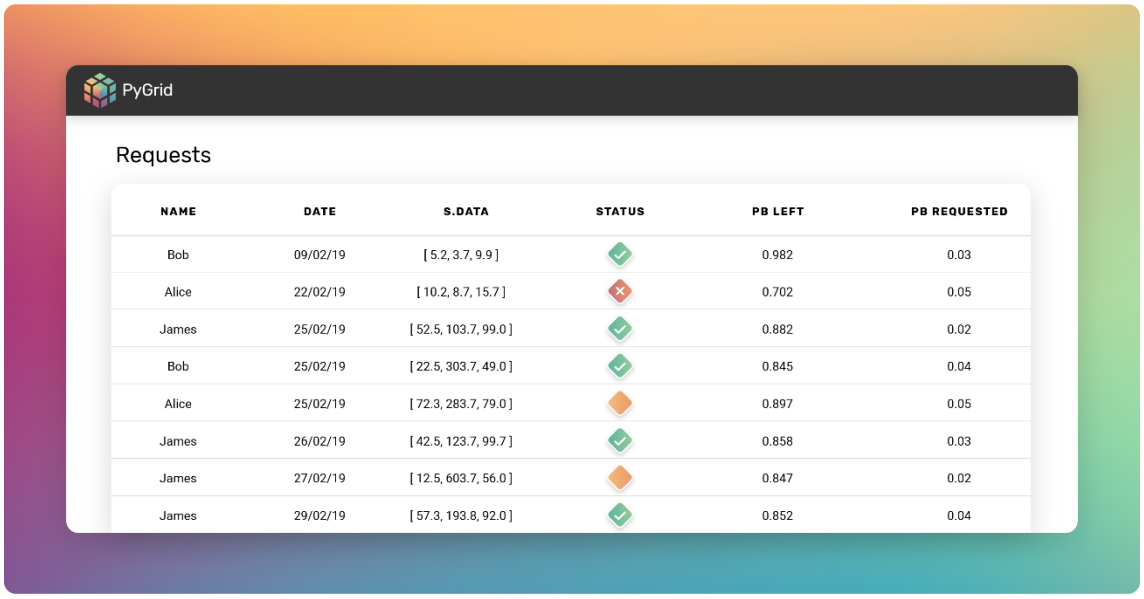
```
In [11]: avg_height = height_sum.get(user="Bob") / 5
         avg_height

Out[11]: (Wrapper)>PrivateTensor>tensor(49.4800)
```

¡Hecho! Conocemos la altura y el peso promedio de los bebés nacidos en febrero sin tener que mover el conjunto de datos a nuestro propio servidor, y nunca necesitamos recibir información privada sobre los bebés individuales.

¿Cómo podemos gestionar el acceso a los datos?

En un futuro próximo, proporcionaremos una interfaz sencilla para autenticar y administrar las reglas de tensores. Como administrador de su propio nodo de red, podrá administrar las cuentas del nodo. Como propietario de los datos, puede identificar y controlar quién puede acceder a su nodo.

El administrador de la red estará facultado para permitir o denegar el acceso evaluando las solicitudes utilizando diferentes técnicas.



CASO DE USO 2: APRENDIZAJE FEDERADO ENTRE SILOS

¿Cómo podemos usar la arquitectura PyGrid para realizar un aprendizaje federado en instituciones o dispositivos? En este caso de uso, entrenaremos un modelo MNIST con un enfoque de aprendizaje federado. El proceso para que el propietario de los datos llene los nodos con muestras de conjuntos de datos es el mismo que en el caso de uso 1, por lo que pasaremos directamente al flujo de trabajo del científico de datos.

El científico de datos

Paso 1: Importar PySyft y las dependencias

In [1]:
```
import syft as sy
from syft.grid.public_grid import PublicGridNetwork
import torch as th
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
```

Paso 2: Defina nuestra arquitectura modelo

Ahora, necesitamos definir nuestra arquitectura modelo y todos los elementos necesarios para realizar un proceso de aprendizaje automático en redes neuronales.

In [2]:
```
hook = sy.TorchHook(th)
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5, 1)
        self.conv2 = nn.Conv2d(20, 50, 5, 1)
        self.fc1 = nn.Linear(4*4*50, 500)
        self.fc2 = nn.Linear(500, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2, 2)
        x = x.view(-1, 4*4*50)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)


device = th.device("cuda:0" if th.cuda.is_available() else "cpu")

if(th.cuda.is_available()):
    th.set_default_tensor_type(th.cuda.FloatTensor)

model = Net()
model.to(device)
optimizer = optim.SGD(model.parameters(), lr=0.01)
criterion = nn.CrossEntropyLoss()
```

Paso 3: conéctese a la plataforma Grid

Como hicimos antes, necesitamos conectarnos con la grid de la red para realizar consultas sobre los nodos.

In [3]:
```
GRID_ADDRESS = 'localhost'
GRID_PORT = '5000'

my_grid = PublicGridNetwork(hook,"http://" + GRID_ADDRESS + ":" + GRID_PORT)
```

Paso 4: busque su conjunto de datos deseado

En este ejemplo, buscamos conjuntos de datos MNIST y sus etiquetas.

```
In [4]: data = my_grid.search("#X", "#mnist", "#dataset")
        target = my_grid.search("#Y", "#mnist", "#dataset")
```

Como podemos ver aquí, nuestra red de cuadrícula tiene algunos nodos que albergan conjuntos de datos MNIST.

```
In [5]: data
```

```
Out[5]: {'Bob': [(Wrapper)>[PointerTensor | me:11578666584 -> Bob:58492988924]
                Tags: #mnist #dataset #X
                Shape: torch.Size([2500, 1, 28, 28])
                Description: The input datapoints to the MNIST dataset....],
         'ai-company': [(Wrapper)>[PointerTensor | me:37423176826 -> ai-company:55368203755]
                Tags: #mnist #dataset #X
                Shape: torch.Size([2500, 1, 28, 28])
                Description: The input datapoints to the MNIST dataset....],
         'university': [(Wrapper)>[PointerTensor | me:56299237390 -> university:49614354750]
                Tags: #mnist #dataset #X
                Shape: torch.Size([2500, 1, 28, 28])
                Description: The input datapoints to the MNIST dataset....],
         'ai-research-lab': [(Wrapper)>[PointerTensor | me:72069214981 -> ai-research-lab:79465898
        021]
                Tags: #mnist #dataset #X
                Shape: torch.Size([2500, 1, 28, 28])
                Description: The input datapoints to the MNIST dataset....]}
```

```
In [6]: target
```

```
Out[6]: {'Bob': [(Wrapper)>[PointerTensor | me:59982559387 -> Bob:64966265784]
                Tags: #Y #mnist #dataset
                Shape: torch.Size([2500])
                Description: The input labels to the MNIST dataset....],
         'ai-company': [(Wrapper)>[PointerTensor | me:73137935503 -> ai-company:51459620941]
                Tags: #Y #mnist #dataset
                Shape: torch.Size([2500])
                Description: The input labels to the MNIST dataset....],
         'university': [(Wrapper)>[PointerTensor | me:78688528504 -> university:51297790986]
                Tags: #Y #mnist #dataset
                Shape: torch.Size([2500])
                Description: The input labels to the MNIST dataset....],
         'ai-research-lab': [(Wrapper)>[PointerTensor | me:76790395234 -> ai-research-lab:94143777
        988]
                Tags: #Y #mnist #dataset
                Shape: torch.Size([2500])
                Description: The input labels to the MNIST dataset....]}
```

Paso 5: crea una referencia a ese puntero de datos

Ahora, solo necesitamos obtener una referencia directa para manejar estos punteros.

```
In [7]: data = list(data.values())
        target = list(target.values())
```

Esta función nos ayudará a descubrir cómo funciona nuestro algoritmo de aprendizaje federado.

```
In [8]: def epoch_total_size(data):
            total = 0
            for i in range(len(data)):
                for j in range(len(data[i])):
                    total += data[i][j].shape[0]

            return total
```

Paso 6: ¡Entrena tu modelo!

En esta función de entrenamiento, estamos iterando sobre los punteros de datos para encontrar al trabajador respectivo y entrenando los modelos de forma remota. Aquí tenemos un par de instrucciones:

• El primero es model.send (trabajador): esta función enviará una copia de nuestro modelo global al trabajador actual para que lo capacite con los datos del trabajador.

• El segundo es model.get (): después de entrenar con los datos locales, necesitamos recuperar el modelo local para actualizar nuestro modelo global.

De esa manera, el proceso de aprendizaje federado pasará por todos los nodos que albergan el conjunto de datos MNIST.

In [9]:
```
N_EPOCS = 4

def train(epoch):
    model.train()
    epoch_total = epoch_total_size(data)
    current_epoch_size = 0
    for i in range(len(data)):
        for j in range(len(data[i])):
            current_epoch_size += len(data[i][j])
            worker = data[i][j].location
            model.send(worker)
            optimizer.zero_grad()
            pred = model(data[i][j])
            loss = criterion(pred, target[i][j])
            loss.backward()
            optimizer.step()
            model.get()
            loss = loss.get()
            print('Train Epoch: {} | With {} data |: [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                      epoch, worker.id, current_epoch_size, epoch_total,
                            100. *  current_epoch_size / epoch_total, loss.item()))

for epoch in range(N_EPOCS):
    train(epoch)
```

```
Train Epoch: 0 | With Bob data |: [2500/10000 (25%)]    Loss: 2.311662
Train Epoch: 0 | With ai-company data |: [5000/10000 (50%)]     Loss: 2.307105
Train Epoch: 0 | With university data |: [7500/10000 (75%)]     Loss: 2.307899
Train Epoch: 0 | With ai-research-lab data |: [10000/10000 (100%)]      Loss: 2.305987
Train Epoch: 1 | With Bob data |: [2500/10000 (25%)]    Loss: 2.303039
Train Epoch: 1 | With ai-company data |: [5000/10000 (50%)]     Loss: 2.299403
Train Epoch: 1 | With university data |: [7500/10000 (75%)]     Loss: 2.299507
Train Epoch: 1 | With ai-research-lab data |: [10000/10000 (100%)]      Loss: 2.297732
Train Epoch: 2 | With Bob data |: [2500/10000 (25%)]    Loss: 2.294667
Train Epoch: 2 | With ai-company data |: [5000/10000 (50%)]     Loss: 2.291863
Train Epoch: 2 | With university data |: [7500/10000 (75%)]     Loss: 2.291327
Train Epoch: 2 | With ai-research-lab data |: [10000/10000 (100%)]      Loss: 2.289621
Train Epoch: 3 | With Bob data |: [2500/10000 (25%)]    Loss: 2.286435
Train Epoch: 3 | With ai-company data |: [5000/10000 (50%)]     Loss: 2.284387
Train Epoch: 3 | With university data |: [7500/10000 (75%)]     Loss: 2.283218
Train Epoch: 3 | With ai-research-lab data |: [10000/10000 (100%)]      Loss: 2.281530
```

CASO DE USO 3: MLAAS CIFRADO

¿Cómo podemos alojar modelos y realizar inferencias de forma segura y privada?

La solución de PyGrid para este problema es utilizar una estructura de datos denominada plan y protocolos de Computación Multipartita (MPC). Un plan es una estructura de datos para definir y serializar un conjunto de instrucciones que se ejecutarán de forma remota. Usando planos, podemos definir una estructura de modelo que se ejecutará en un dispositivo remoto. De esa manera, un plan que utiliza punteros MPC remotos distribuidos en diferentes máquinas puede realizar inferencias de modelos de forma segura.

Paso 1: Importar PySyft y las dependencias

Como anteriormente, el primer paso es importar nuestras dependencias. Es importante tener en cuenta que debemos configurar hook.local_worker.is_client_worker en False. Esto permitirá que la biblioteca syft almacene los metadatos del plan en su estructura.

```
In [1]: import syft as sy
        from syft.grid.public_grid import PublicGridNetwork
        import pickle

        import torch as th
        import torch.nn as nn
        import torch.optim as optim
        import torch.nn.functional as F

        hook = sy.TorchHook(th)
        sy.hook.local_worker.is_client_worker = False
```

Paso 2: conéctese a la plataforma Grid

Ahora, como se vio anteriormente, debemos conectarnos a la plataforma de la red.

```
In [2]: public_grid = PublicGridNetwork(hook, "http://localhost:5000")
```

Paso 3: definir el modelo

Aquí estamos definiendo nuestro modelo. Es importante señalar la necesidad de ampliar los planes en la definición del modelo. Para esta explicación, usaremos un modelo lineal con una sola capa que tiene los pesos y los valores de sesgo establecidos de antemano. De esa forma, podemos predecir y comprender los resultados.

```
In [3]: # Build Model
        class Net(sy.Plan):
            def __init__(self):
                super(Net, self).__init__(id="encrypted-model")
                self.fc1 = th.tensor([2.0, 4.0])
                self.bias = th.tensor([100.0])

            def forward(self, x):
                x = self.fc1.matmul(x) # x * [2.0 , 4.0]
                x = x + self.bias # x + 100
                return x
```

Paso 4: definir los datos de entrada

Nuestros datos de entrada serán este tensor de 1 dimensión. Así, ya podemos visualizar el resultado final.

```
In [4]: input_value = th.tensor([1.0, 2])
```

Paso 5: inicializar el modelo

Ahora, necesitamos inicializar el modelo. A modo de comparación, también iniciaremos un modelo descifrado.

```
In [5]: model = Net()
        decrypted_model = Net()
        model.build(th.tensor([5.0, 3.0]))
```

Paso 6: sirva el modelo en la red de cuadrícula

Finalmente, podemos servir este modelo en el grid de la red. Es importante tener en cuenta la bandera MPC. Esta bandera permitirá que la biblioteca syft divida los parámetros del plan en diferentes dispositivos a través de la red utilizando protocolos MPC.

Aquí, podemos ver dónde está nuestro plan y sus porciones de parámetros. El grupo de datos de la empresa aloja la estructura de nuestro plan, el grupo de datos del hospital y el grupo de datos públicos alojan los valores de los parámetros de MPC. Y finalmente, el grupo de datos de la universidad es nuestro proveedor de cifrado que nos permite realizar multiplicaciones de MPC.

```
In [7]: public_grid.query_model_hosts(model.id, mpc=True)
Out[7]: (Federated Worker < id: company-datacluster >,
         [Federated Worker < id: hospital-datacluster >,
          Federated Worker < id: public-datacluster >],
         Federated Worker < id: university-datacluster >)
```

Paso 7: Devuelve los resultados

```
In [8]: result = public_grid.run_remote_inference(model.id, input_value, mpc=True)
        result
Out[8]: tensor([110.])
```

Aquí tenemos muchas cosas sucediendo al mismo tiempo. Así que profundicemos:

1: esta función descargará la estructura del plan del grupo de datos de la empresa y recuperará sus punteros remotos.

2 - El segundo paso es dividir input_values en valores MPC y compartirlo con los mismos dispositivos que albergan los punteros MPC del plan.

3 - Como todos los punteros MPC están distribuidos entre los dispositivos, podemos ejecutar el plan.

4 - Finalmente, podemos agregar los resultados de MPC que devolverán el resultado real.

Comparación con el modelo descifrado

A modo de comparación, si ejecutamos el modelo descifrado, obtendremos el mismo resultado.

```
In [9]: expected_result = decrypted_model(input_value)
        expected_result

Out[9]: tensor([110.])
```

El futuro de PyGrid tiene 4 objetivos principales:

Red heterogénea (syft.js, swift.js, trabajadores móviles): Primero, es crear una forma estándar de enviar y recibir mensajes desde diferentes plataformas y aplicaciones. Hoy, la plataforma PyGrid es una plataforma basada en servidor, lo que significa que necesita configurar y proporcionar una infraestructura a los nodos. Tenemos la intención de extender las funciones de PyGrid a los dispositivos móviles.

Presupuesto de privacidad: Desarrollar el presupuesto de Privacidad para evaluar y controlar el nivel de anonimización de los datos.

Seguimiento de privacidad diferencial automático: Esto permite el seguimiento automático de un presupuesto de privacidad para las entidades en un conjunto de datos a lo largo del tiempo. De esa manera, podemos ofrecer garantías formales sobre la cantidad de información que se filtra cada vez que se publica un activo privado (como un modelo de inteligencia artificial). Se prefiere que esta infraestructura sea lo más automatizada posible, pero también se permiten las aplicaciones (UI) que permiten la investigación de activos digitales para información privada por parte de humanos (y probablemente sean esenciales para los primeros usuarios).

Cola de solicitud de datos: Crearemos la cola de solicitud de datos, que permite a los propietarios de los datos evaluar las solicitudes de datos controlando el acceso a los datos.

Anexo 5 – Recorrido para lograr la instalación exitosa de PyGrid y Pysyft

Lo primero que intentamos fue recrear el ejemplo que figuraba en el siguiente link:

https://towardsdatascience.com/making-pate-bidirectionally-private-6d060f039227

Ese artículo tenía asociado el siguiente GITHUB:

https://github.com/aristizabal95/Making-PATE-Bidirectionally-Private

En base a la información de ese repositorio generamos un notebook para probar la interacción de los Trusted Curators y las organizaciones, pero no fue posible, porque el código no funcionaba de manera correcta. Lo que notamos es que estaba desactualizado porque trabajaba en la versión 0.2.6. la información indicaba que ya se encontraba disponible la versión 0.5.0.

En conjunto con los tutores se buscó un ejemplo con la última versión 0.5.0. Adjuntamos el notebook encontrado, pero que tampoco funcionaba.

https://colab.research.google.com/github/agungsantoso/private-ai/blob/master/Section%203%20-%20Securing%20Federated%20Learning.ipynb

Pensando que la versión 0.5.0 era la última y encontramos el siguiente video tutorial:

OpenMined PyGrid 0.5 Demo

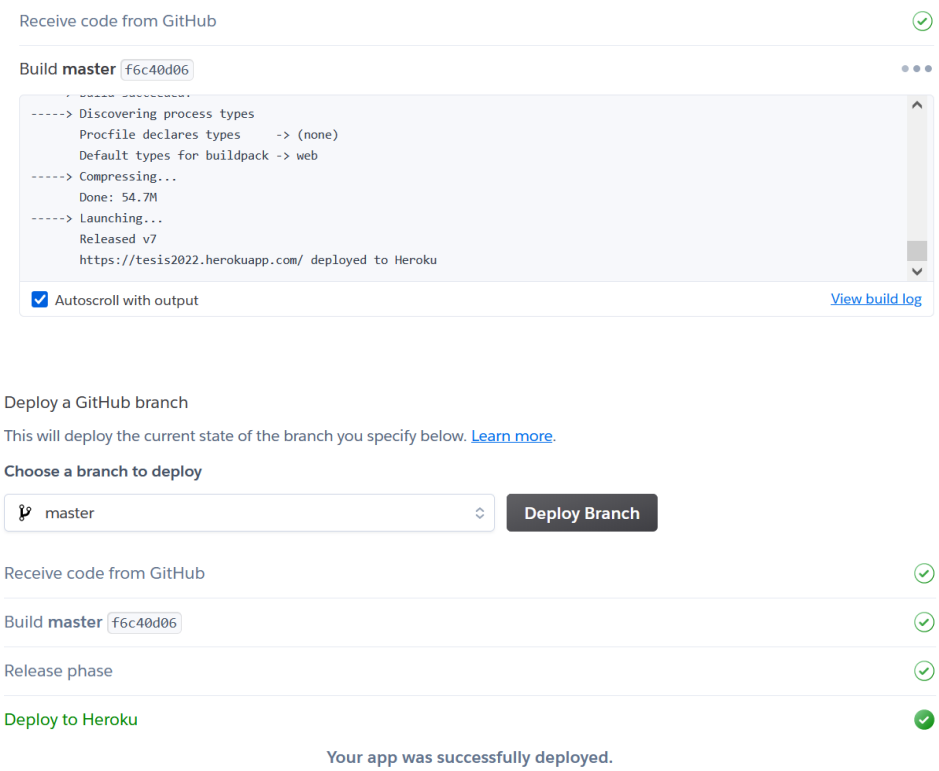
En ese video se había introducido el concepto de nodos y dominios para ejecutar las relaciones usando Pysfyt. Luego, en el blog de OpenMined se encontró el link a un repositorio de GITHUB el cual había sido actualizado y contaba con una versión posterior, 0.6.0.

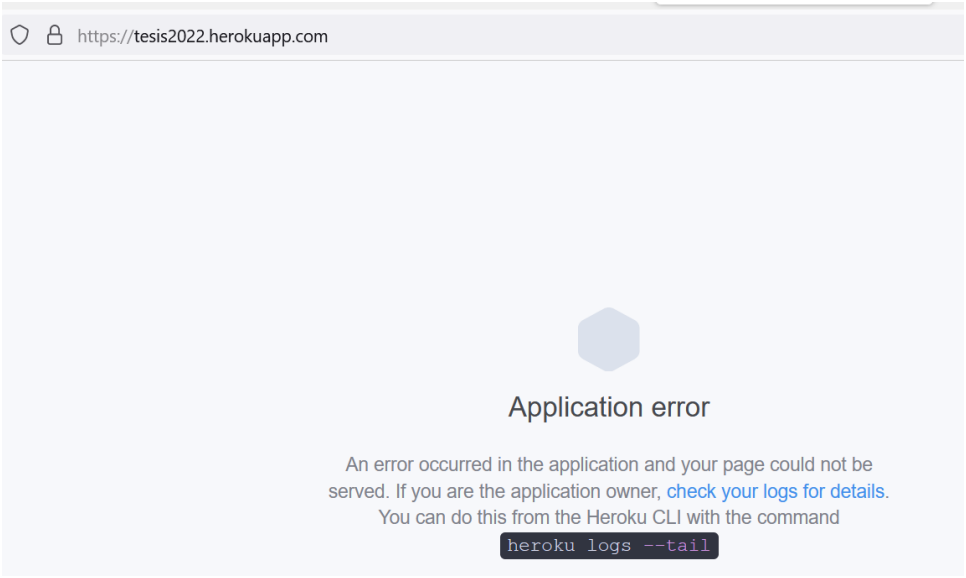
https://github.com/OpenMined/PySyft/tree/0.6.0

Lo primero que intentamos fue instalar PySyft utilizando Docker Hub con el siguiente código, sin éxito:

```
docker run -d -e PORT=5000 -e DATABASE_URL="sqlite:///nodedatabase.db" -p
6000:5000 --platform linux/amd64 openmined/grid-domain
docker run -d -e PORT=7000 -e DATABASE_URL="sqlite:///nodedatabase.db" -p
8000:7000 --platform linux/amd64 openmined/grid-network
docker run -d -e PORT=7000 -e DATABASE_URL="sqlite:///nodedatabase.db" -p
8080:8080 --platform linux/amd64 openmined/grid-worker
Create virtualenv and install syft
       python3 -m venv venv
       source venv/bin/activate
       venv/bin/python3 -m pip install --upgrade pip
       sudo pip install syft
       sudo pip install jupyterlab
```

Otra de las alternativas era instalar Pygrid Admin primero se intentó utilizando Heroku, en este caso logramos hacer el deploy con éxito, pero al ejecutar la app nos dio el siguiente error:





También intentaremos cargarlo de manera local, pero sin resultados positivo. La segunda opción fue instalarlo con el comando:

```
yarn install
yarn dev
```

En ese caso debíamos ingresar a http://localhost:3000 pero no logramos instalarlo.

El siguiente intento fue instalar PyGrid en Azure utilizando la información del siguiente repositorio:

https://github.com/OpenMined/PyGrid/blob/dev/deployment.md

En ese caso intentamos generar las credenciales usando terrafrom:

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/service_principal_client_secret#password

Instalamos también el cli de azure, pero sin éxito:

1. cli de azure - https://docs.microsoft.com/en-us/cli/azure/install-azure-cli windows?tabs=azure-cli
2. Terraform
3. Cli pygrid – pip install pygrid-cli
4. Obtener los datos subscription id , client id - https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/service_principal_client_secret#password

Obtuvimos el siguiente error:

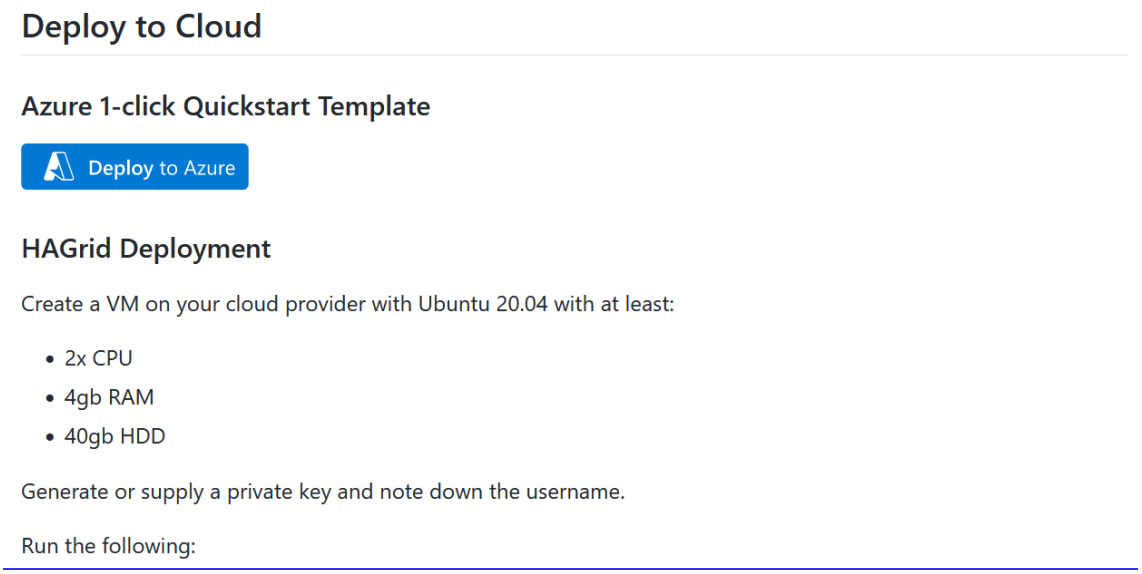
```
Failed to query available provider packages Could not retrieve the list of available versions for provider hashicorp/random: provider registry.terraform.io/hashicorp/random was not found in any of the search locations
```

Luego buscamos en el marketplace de azure, encontrando un dominio llamada openmind grid – lo instalamos, pero no supimos como loguearnos ni usarlo.

En paralelo intentamos instalar el servidor con vagrant y con VirtualBox (máquina virtual) siguiente la información que se encontraba en el siguiente GITHUB.

https://github.com/OpenMined/PySyft

Habiendo agotado las posibles de instalaciones disponibles en la literatura, la alternativa fue escribir directamente a los desarrolladores de OpenMined, quienes respondieron de manera rápida y nos indicaron que la alternativa vigente era instalar PySyft utilizando Azure 1-click Quickstart Template.



En este caso tuvimos éxito y se generaron los ejemplos de interacción entre los Trusted Curators en un ejemplo simplificado del escenario 2.

Anexo 6 – Explicación del código generado para las relaciones de las Etapas 1 y 2

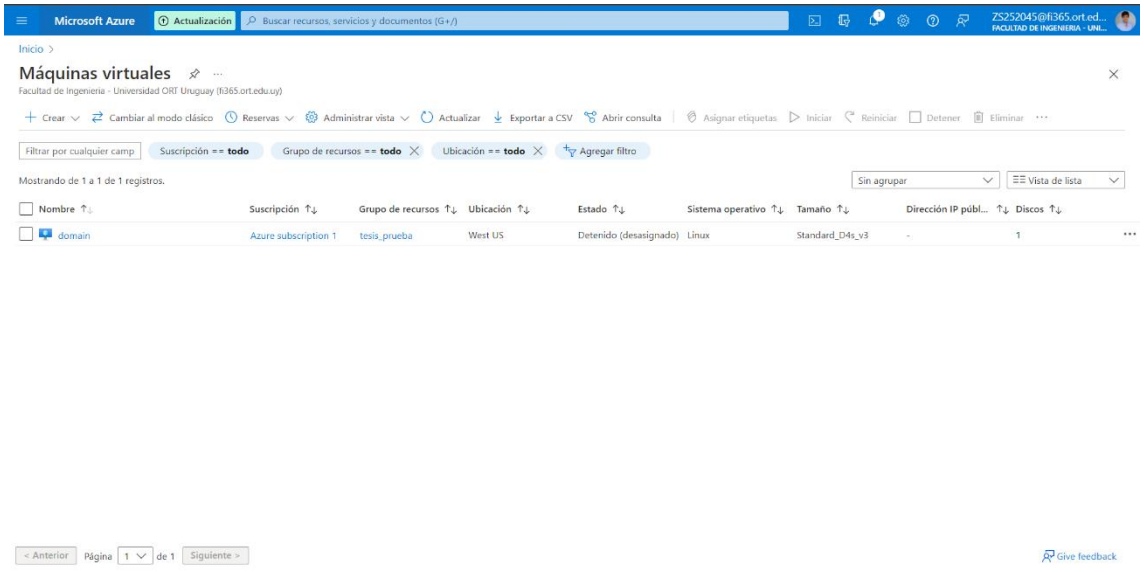
En este caso presentaremos las relaciones de los TC_S y TC_T utilizando PySyft y supondremos que el TC_T posee el modelo para entrenar.

A futuro sugerimos probar la herramienta en la situación donde las organizaciones entrenan el modelo con sus datos y solo reciben las consultas.

Al inicio el Data Owner (TC_S) genera en Azure un dominio utilizando la información disponible en el siguiente repositorio de GitHub

https://github.com/OpenMined/PySyft/tree/0.6.0

Existe una Instalación predefinida con un solo click, esta fue la que se utilizó, presentamos la información del dominio en la siguiente imagen.



En la **<u>Etapa 1</u>** antes descripta, el rol de Data Owner (TC_S) comienza con la instalación de Syft

```
!pip install syft==0.6.0
```

Se conecta al dominio URL1.

```
DOMAIN_URL1 ="http://40.118.207.120:80"
```

gin to the domain as admin and create an user with the role Data Scientist

```
domain = sy.login(
    url=DOMAIN_URL1,
    email="info@openmined.org",
    password="changethis"
    )
```

Una vez configurado el dominio, el Data Owner (TCs) debe generar los usuarios para los Data Scientist. Esto implica correo y contraseña.

```
domain.users.create(name='TCS',email='szanottag@gmail.com',role='Data Scientist',  password='passwordtest')
domain.users.create(name='TCT',email='betirod@hotmail.com',role='Data Scientist',  password='passwordtest1')
```

List the users in order to check that it has been correctly created

```
domain.users
```

	id	email	name	budget	verify_key	role	added_by	website	institution	daa_pdf	cre
0	1	info@openmined.org	Jane Doe	5.55	81708d92bedf4ad7d79455abc8607cbf59be146a3d8f85...	Owner	None	None	None	NaN	202 02:25:42
1	34	mikaelapisani@gmail.com	test	0.00	8cf43ee33553b7b870872240061068cfa696ea87ca1d5a...	Data Scientist	Jane Doe			1.0	202 21:36:41
2	35	jramas@gmail.com	test2	0.00	0b441a39e7e8ce192eaf3bbe499415a921e54d90332917...	Data Scientist	Jane Doe			2.0	202 21:39:43
3	36	szanottag@gmail.com	TCS	0.00	efb85c04adb2761180386c5413ef16824985361b99e6fc...	Data Scientist	Jane Doe			3.0	202 23:39:39
4	37	betirod@hotmail.com	TCT	0.00	5ddf5c68aea32691f6bb49c827b3a9066120a6395919c8...	Data Scientist	Jane Doe			4.0	202 23:39:40

Carga el dataset con las preguntas para el modelo.

```
df = pd.read_csv('/content/drive/MyDrive/df_consultas.csv', index_col=0 )
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
537	0	57	60	0	0	21.7	0.735	67
538	0	127	80	37	210	36.3	0.804	23
539	3	129	92	49	155	36.4	0.968	32
540	8	100	74	40	215	39.4	0.661	43
541	3	128	72	25	190	32.4	0.549	27
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

231 rows × 8 columns

Y lo transforma en un tensor.

```
data_tensors_consultas = sy.Tensor(df.values.astype(np.int32))
```

Agrega información de identificación sobre los datos del tensor.

```
domain.load_dataset(
    assets={"data": private_data_tensors_consultas},
    name="diabetes_consultas",
    description="Diabetes_Consultas"
)
```

Revisa los pedidos de descarga de la información y autoriza si cree conveniente

```
domain.requests
```

	Name	Email	Role	Request Type	Status	Reason	Request ID	Requested Object's ID	Requested Object's tags	Rec
0	TCS	szanottag@gmail.com	Data Scientist	DATA	pending	Para salvar la tesis	<UID: 4a5f513439154d91a1093651db2ea51d>	<UID: 0188a01ab220455f93c75b1eb82ce0c6>	[#data]	

prove the request by the id

```
domain.requests[0].approve()
```

```
domain.requests
```

	Name	Email	Role	Request Type	Status	Reason	Request ID	Requested Object's ID	Requested Object's tags	Re
0	TCS	szanottag@gmail.com	Data Scientist	DATA	accepted	Para salvar la tesis	<UID: 4a5f513439154d91a1093651db2ea51d>	<UID: 0188a01ab220455f93c75b1eb82ce0c6>	[#data]	

Esto se puede ver en el notebook 1.DataOwner(Trusted Curator Student).ipynb

Por otro lado el rol del el rol de Data Scientist (TC_T) comienza con la instalación de Syft.

```
!pip install syft==0.6.0
```

Luego entrena el modelo con los datos propios.

```
df1 = pd.read_csv('/content/drive/MyDrive/df_entrenar_modelo.csv', index_col=0 )
df1
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
532	1	86	66	52	65	41.3	0.917	29	0
533	6	91	0	0	0	29.8	0.501	31	0
534	1	77	56	30	56	33.3	1.251	24	0
535	4	132	0	0	0	32.9	0.302	23	1
536	0	105	90	0	0	29.6	0.197	46	0

537 rows × 9 columns

```
x = df1.iloc[:,0:8] # Features
```

```
x
```

...

```
y = df1["Outcome"] # Lables
```

```
y
```

...

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(x.values)
```

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0, n_jobs=-1).fit(X_scaled , y.values)
```

Con los datos proporcionados por el Data Owner (TC_S) se conecta al dominio URL1 y explora los datasets disponibles.

```
# Logging into the domain as DS

DOMAIN_URL1 ="http://40.118.207.120:80"
domain = sy.login(
    url=DOMAIN_URL1,
    email="szanottag@gmail.com",
    password="passwordtest",
    port=8081
)
```

```
Connecting to http://40.118.207.120:80... done!          Logging into node... done!
```

the Dataset in order to check that it has been correctly created

```
domain.datasets
```

Idx	Name	Description	Assets	Id
[0]	diabetes_consultas	Diabetes_Consultas	["data"] -> Tensor	91ef9c35-f706-4ee9-a404-41fe11a4a768

Solicita permiso para descargar el tensor y lo guarda en una variable que luego convierte a Panda DataFrame.

```
ptr.request(reason="Para salvar la tesis")
```

s check the request

```
domain.requests
```

...

h the request accepted by the Data Owner, the Data Scientist has the possibilty to download the information

```
data_tensor = ptr.get()
```

s check the information downloaded

```
data_tensor
```

...

```
data_tensor.child.child
```

```
values = list(map(lambda x: x.child[0], data_tensor.child.child))
```

```
values
```

...

ls check the DataFrame

```
df2 = pd.DataFrame(values)
df2
```

	0	1	2	3	4	5	6	7
0	0	57	60	0	0	21	0	67
1	0	127	80	37	210	36	0	23
2	3	129	92	49	155	36	0	32
3	8	100	74	40	215	39	0	43
4	3	128	72	25	190	32	0	27
...	...	...	...	...	...	...	...	...
226	10	101	76	48	180	32	0	63
227	2	122	70	27	0	36	0	27
228	5	121	72	23	112	26	0	30
229	1	126	60	0	0	30	0	47
230	1	93	70	31	0	30	0	23

231 rows × 8 columns

Utiliza el modelo entrenado para generar una predicción.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df2.values)
```

```
from sklearn.linear_model import LogisticRegression

prediccion = clf.predict(X_scaled)
prediccion
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0])
```

Almacena y exporta el resultado, dando fin a la Etapa 1. Esta información se puede ver en el notebook DataScientist (Trusted Curtator Teacher).ipynb

```
df_prediccion = pd.DataFrame(prediccion, columns=["prediccion"])
```

```
df_prediccion
```

	prediccion
0	0
1	0
2	0
3	0
4	0
...	...
226	0
227	0
228	0
229	0
230	0

231 rows × 1 columns

```
df_prediccion.to_csv("df_prediccion.csv")
df_prediccion.to_csv("/content/drive/MyDrive/df_prediccion.csv")
```

La **Etapa 2** inicia de manera similar pero ahora los roles están invertidos. Quien genera el dominio es el TC_T como Data Owner. Se conectar al URL2.

```
DOMAIN_URL2 = "http://ort.westus.cloudapp.azure.com:80"
```

jin to the domain as admin and create an user with the role Data Scientist

```
domain = sy.login(
    url=DOMAIN_URL2,
    email="info@openmined.org",
    password="changethis"
)
```

Una vez configurado el dominio, el Data Owner (TC_T) debe generar los usuarios para los Data Scientist. Esto implica correo y contraseña.

```
domain.users
```

	id	email	name	budget	verify_key	role	added_by	website	institution	daa_pdf	
0	1	info@openmined.org	Jane Doe	5.55	81708d92bedf4ad7d79455abc8607cbf59be146a3d8f85...	Owner	None	None	None	NaN	02:2
1	34	mikaelapisani@gmail.com	test	0.00	8cf43ee33553b7b870872240061068cfa696ea87ca1d5a...	Data Scientist	Jane Doe			1.0	21:3
2	35	jramas@gmail.com	test2	0.00	0b441a39e7e8ce192eaf3bbe499415a921e54d90332917...	Data Scientist	Jane Doe			2.0	21:3
3	36	szanottag@gmail.com	TCS	0.00	efb85c04adb2761180386c5413ef16824985361b99e6fc...	Data Scientist	Jane Doe			3.0	23:3
4	37	betirod@hotmail.com	TCT	0.00	5ddf5c68aea32691f6bb49c827b3a9066120a6395919c8...	Data Scientist	Jane Doe			4.0	23:3
5	38	t@gmail.com	testuser2	0.00	1ff88ee8317b998dce5c2518dca2d63439a6f2dc5cf5d1...	Data Scientist	Jane Doe			5.0	01:0
6	39	test@gmail.com	test	0.00	9a015d1fddcd3ee60b4c563c553c68c978716bb0094215...	Data Scientist	Jane Doe			6.0	12:2

Carga el DataFrame con las predicciones generado en la **Etapa 1.**

```
df = pd.read_csv('/content/drive/MyDrive/df_prediccion.csv', index_col=0 )
df
```

	prediccion
0	0
1	0
2	0
3	0
4	0
...	...
226	0
227	0
228	0
229	0
230	0

231 rows × 1 columns

Lo convierte en tensor, agrega información adicional al dataset para ser identificado y aguarda el pedido del Data Scientist (TC_S)

```
data_tensors_prediccion = sy.Tensor(df.values.astype(np.int32))
```

```
domain.load_dataset(
    assets={"data": private_data_tensors_prediccion},
    name="diabetes_prediccion",
    description="Diabetes_Prediccion"
)
```

```
domain.requests
```

15	TCS	szanottag@gmail.com	Data Scientist	DATA	pending		<UID: 94b828b74cf84d95abbb8057812348e3>	<UID: 7d22bc4f4dcc4077b471413c7179be1e>	[#da __len
16	TCS	szanottag@gmail.com	Data Scientist	DATA	pending		<UID: 53570ea229ab4da7a881304a34e0978a>	<UID: 6cb343792031431e9d058c4b35c31774>	[#da __len
17	TCS	szanottag@gmail.com	Data Scientist	DATA	accepted	Para salvar la tesis	<UID: 26db0ee8ede54fc6b92cb5a84f6d4af6>	<UID: 1db5f9542ee343f2b551be0f495fdf72>	[#da
18	TCS	szanottag@gmail.com	Data Scientist	DATA	pending		<UID: e31989112c9f4a64afe6627061698369>	<UID: b4bf2625663f4186bb4f6e6ac8219b77>	[#da __len
19	Jane Doe	info@openmined.org	Owner	DATA	pending		<UID: a2aaa353bc724c97b1c471bde0933693>	<UID: bb79da3c976c4401a44891cca3add769>	[#da __len
20	Jane Doe	info@openmined.org	Owner	DATA	accepted	para ver la prediccion	<UID: 427113add5b0460f90ed8956f91db6c9>	<UID: a34aabb541944940b332cea0df1d0426>	[#da
21	test2	jramas@gmail.com	Data Scientist	DATA	pending	para ver la prediccion	<UID: f72e6230604444b29a70ca7e42cbf227>	<UID: 4242a88d43ba43199bd1b646c308a291>	[#da

Una vez que el Data Scientist solicita la autorización para descargar, la acepta y finaliza su rol.

Esto se puede ver en el notebook DataOwner(Trusted Curtator Teacher).ipynb.

```
domain.requests[21].approve()
```

```
domain.requests
```

15	TCS	szanottag@gmail.com	Data Scientist	DATA	pending		<UID: 94b828b74cf84d95abbb8057812348e3>	<UID: 7d22bc4f4dcc4077b471413c7179be1e>	[#da __len
16	TCS	szanottag@gmail.com	Data Scientist	DATA	pending		<UID: 53570ea229ab4da7a881304a34e0978a>	<UID: 6cb343792031431e9d058c4b35c31774>	[#da __len
17	TCS	szanottag@gmail.com	Data Scientist	DATA	accepted	Para salvar la tesis	<UID: 26db0ee8ede54fc6b92cb5a84f6d4af6>	<UID: 1db5f9542ee343f2b551be0f495fdf72>	[#da
18	TCS	szanottag@gmail.com	Data Scientist	DATA	pending		<UID: e31989112c9f4a64afe6627061698369>	<UID: b4bf2625663f4186bb4f6e6ac8219b77>	[#da __len
19	Jane Doe	info@openmined.org	Owner	DATA	pending		<UID: a2aaa353bc724c97b1c471bde0933693>	<UID: bb79da3c976c4401a44891cca3add769>	[#da __len
20	Jane Doe	info@openmined.org	Owner	DATA	accepted	para ver la prediccion	<UID: 427113add5b0460f90ed8956f91db6c9>	<UID: a34aabb541944940b332cea0df1d0426>	[#da
21	test2	jramas@gmail.com	Data Scientist	DATA	accepted	para ver la prediccion	<UID: f72e6230604444b29a70ca7e42cbf227>	<UID: 4242a88d43ba43199bd1b646c308a291>	[#da

Veamos ahora lo que sucede con el Data Scientist (TC_S). En primera instancia se conecta al segundo dominio URL2, con los datos proporcionados por el Data Owner (TC_T).

```
DOMAIN_URL2 ="http://ort.westus.cloudapp.azure.com:80"
domain = sy.login(
    url=DOMAIN_URL2,
    email="jramas@gmail.com",
    password="orttest2"
)
```

Busca los datasets disponibles y una vez identificado el que le interesa, solicita permiso para descargar el tensor, indicando cual es el motivo de su solicitud.

```
domain.datasets
```

Idx	Name	Description	Assets	Id
[0]	diabetes	Diabetes	["data"] -> Tensor	7e4bbe12-5c78-48cd-84b8-e12afcdea299
[1]	diabetes	Diabetes	["data"] -> Tensor	d4b8f7e9-f271-4958-bf20-17288625b05b
[2]	diabetes_train	Diabetes_Train	["data"] -> Tensor	36458f8c-d23e-49de-85db-975e73c06486
[3]	diabetes_test	Diabetes_Test	["data"] -> Tensor	15e21f33-b655-4345-8647-3362e77b6bb6
[4]	diabetes_train	Diabetes_Train	["data"] -> Tensor	b8c06d05-61fe-422d-9fb2-515ba6c84206
[5]	diabetes_test	Diabetes_Test	["data"] -> Tensor	33aba72f-ae41-4adc-ae8d-0a1d0dfa51c9
[6]	diabetes_prediccion	Diabetes_Prediccion	["data"] -> Tensor	736c587c-b9b4-4a11-8ca9-ab41acdc4b7a
[7]	diabetes_prediccion	Diabetes_Prediccion	["data"] -> Tensor	f353636a-0321-443e-bef4-446620223fea

```
ptr.request(reason="para ver la prediccion")
```

; check the request

```
domain.requests
```

2	test2	jramas@gmail.com	Data Scientist	DATA	pending		<UID: d7eae0de20aa4e029fd4b6bc34eacc34>	<UID: 5fa97396e18641568ba83f9e021819f1>	[#data, __len__]
3	test2	jramas@gmail.com	Data Scientist	DATA	pending		<UID: 57482cd372174a6cba95406ba3085c7e>	<UID: 022aa6def8734953938906b352c8c40d>	[#data, __len__]
4	test2	jramas@gmail.com	Data Scientist	DATA	accepted	Para salvar la tesis	<UID: d6910261c49b48179b15234795caed3d>	<UID: ed661f6653674ced9463a85aea01c9c5>	[#data]
5	test2	jramas@gmail.com	Data Scientist	DATA	pending		<UID: 3d429fca4da6446a9f141e0f8977a5d2>	<UID: e0a5843a6b9a47f3bd14cc69a4883cfe>	[#data, __len__]
6	test2	jramas@gmail.com	Data Scientist	DATA	pending		<UID: 0187881026ce46b28926c3244ab4c3c8>	<UID: 20874413bff24ef1a9be84e39025fa96>	[#data, __len__]
7	test2	jramas@gmail.com	Data Scientist	DATA	pending		<UID: db990a5d0e874f16a5071db82cf4d40f>	<UID: 8804271b4c484708a2d2642a6d642241>	[#data, __len__]
8	test2	jramas@gmail.com	Data Scientist	DATA	accepted	para ver la prediccion	<UID: f72e6230604444b29a70ca7e42cbf227>	<UID: 4242a88d43ba43199bd1b646c308a291>	[#data]

Una vez aceptada su solicitud descarga el tensor y lo convierte en un Panda DataFrame.

```
values = list(map(lambda x: x.child[0], data_tensor.child.child))
```

```
values
```

s check the DataFrame

```
df_prediccion = pd.DataFrame(values, columns=["prediccion"])
df_prediccion
```

	prediccion
0	0
1	0
2	0
3	0
4	0
...	...
226	0
227	0
228	0
229	0
230	0

231 rows × 1 columns

Lo almacena en un archivo .CSV para poder utilizarlo luego. Con esto finaliza el ejemplo y se demuestra la posibilidad de utilización de la herramienta. Esto se puede ver en el notebook DataScientist (Trusted Curator Student).ipynb

```
df_prediccion.to_csv("df_prediccion_student.csv")
df_prediccion.to_csv("/content/drive/MyDrive/df_prediccion_student.csv")
```