UNIVERSIDAD ORT
Uruguay

**Facultad de Ingeniería**
***Bernard Wand-Polak***
Cuareim 1451
11100 Montevideo, Uruguay
Tel. 2902 1505

# Prototipo de “Application of Private Aggregation of Ensemble Models to Sensible Data” en la plataforma PySyft de OpenMined

Mikela Pisani, Sergio Yovine

UNIVERSIDAD ORT
Uruguay

**Facultad de Ingeniería**
***Bernard Wand-Polak***
Cuareim 1451
11100 Montevideo, Uruguay
Tel. 2902 1505

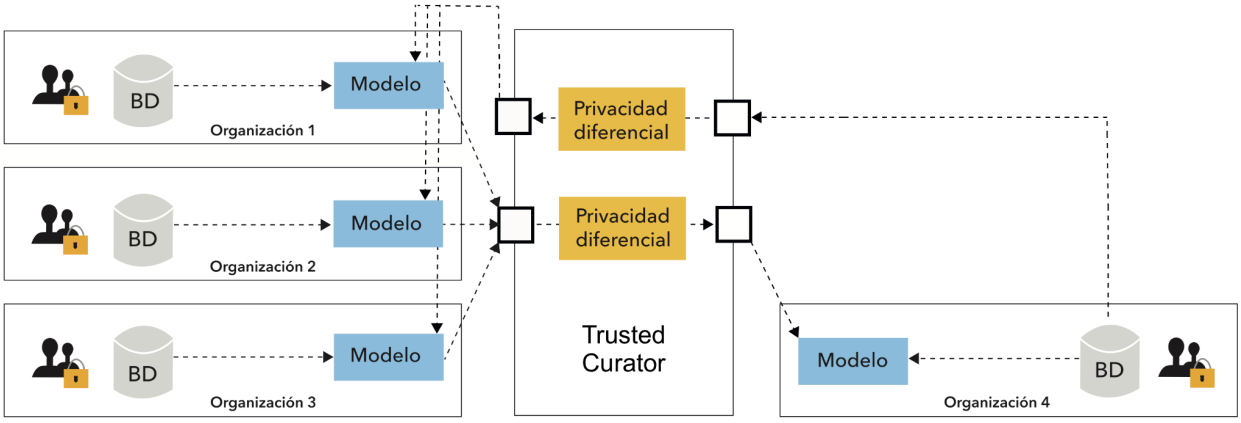
# Introducción

Private Aggregation of Teacher Ensembles (PATE) [1] es una técnica que permite entrenar modelos de aprendizaje automático con garantías de privacidad a través del uso de un mecanismo de privacidad diferencial [2]. La técnica PATE propone entrenar múltiples modelos de *'Teachers'* en conjuntos separados de datos privados confidenciales, y luego usar estos *teachers* para guiar el entrenamiento de un modelo de *'Student'* en datos públicos no etiquetados. Para esto último, el *student* envía un conjunto de datos a los *teachers* con el objetivo de obtener una predicción. Las predicciones de los *teachers* se agregan para crear un ensemble y devolver este resultado al *student,* previamente privatizado mediante un mecanismo de privacidad diferencial.

Claramente, por diseño, PATE provee privacidad solamente para los datos de entrenamiento de los *teachers,* pero no protege los datos del *student*, que se suponen de dominio público. Con el propósito de extender su uso en contextos en los cuales es necesario proteger datos privados del *student,* en [3] se propuso extender PATE mediante el agregado de un mecanismo de privacidad diferencial que actúa sobre los datos del *student* cuando estos son transmitidos a los *teachers*. Este esquema permite que diferentes organizaciones puedan compartir a la vez modelos y datos sin comprometer en ningún momento la privacidad de los datos utilizados en el entrenamiento y en las consultas a los modelos. Además, la validación experimental en dos casos de estudio arrojó resultados muy positivos en cuanto al trade-off entre privacidad y poder predictivo.

Tomando como punto de partida esta arquitectura conceptual, se analizaron diferentes alternativas de implementación, poniendo especial énfasis en los aspectos de gobernanza [4]. A los efectos de facilitar la implementación concreta del esquema seleccionado en la plataforma Syft [5], en este trabajo se propone la utilización de un único *trusted curator*, en vez de dos como sugerido en [3] y retomado en [4]. El rol del *trusted curator* es proveer los mecanismos de privacidad diferencial en ambos sentidos de la comunicación: *student-teachers* y *teachers-student*, que se realiza además a través en forma encriptada.

En este documento se presenta una prueba de concepto de implementación concreta de esta propuesta. El código se enfoca en proporcionar un marco de aplicación para el siguiente escenario de uso:




Las organizaciones O_1, O_2, … y O_n tienen datos privados para crear *teachers* T_1, T_2, … y T_n, que serán puestos a disposición para ser consultados a través de un ensemble construido por el *trusted curator*. La organización O_s, quien no posee datos etiquetados puede usar ese ensemble para etiquetarlos mediante consultas y luego entrenar con estos su propio modelo. El *trusted curator* recurre a la privacidad diferencial para no comprometer los datos privados de cada una de las organizaciones participantes.

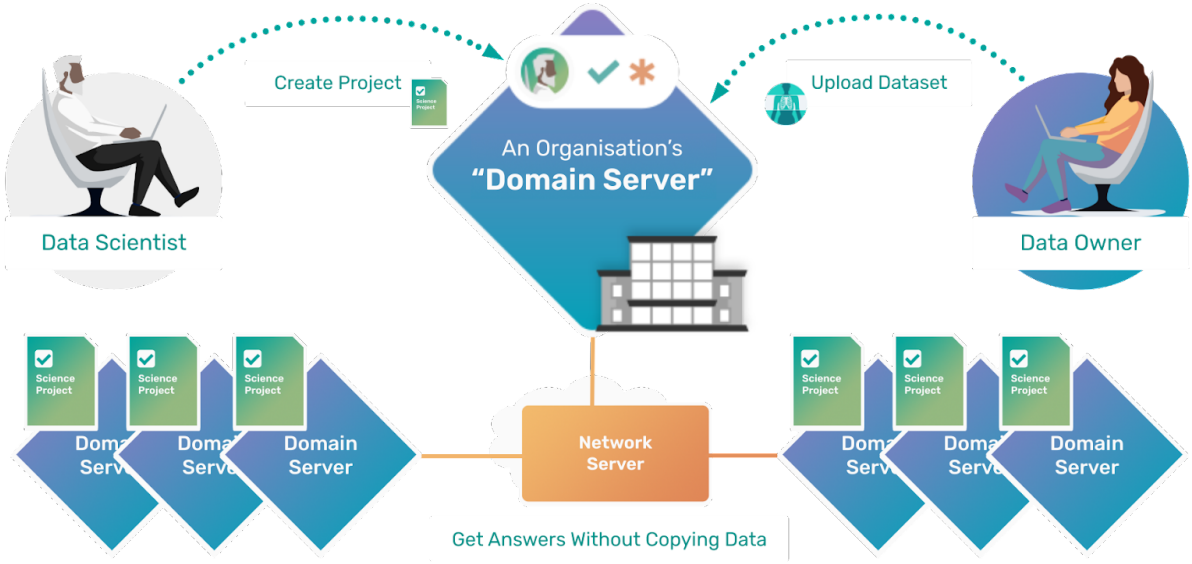
En este caso, tenemos los siguientes componentes:

- O_s inicia un ***Student*** para obtener las etiquetas correspondientes a un conjunto de datos.
- ***TrustedCurator*** crea un ensemble a partir de los resultados de los diferentes modelos de los ***Teachers.***
- ***Teacher_i***, i=[1… n], usa su modelo entrenado para hacer predicciones.

# Framework Syft

Para la implementación de este prototipo se utiliza Syft, un framework open source desarrollado por OpenMined, una comunidad open source. El framework pretende crear herramientas seguras para aplicar ciencia de datos protegiendo los datos. Utiliza técnicas como *federated learning* [6], privacidad diferencial y computación encriptada [7].

En Syft se dispone de un dominio, para compartir los datos, este dominio permite la interacción "remota".




[Fuente: Padawan course]

Es importante comprender los 3 roles fundamentales en este framework:

1. **Data Subject:** persona que está representada en los datos, cuya información se quiere proteger
2. **Data Owner:** corresponde al dueño de los datos, quien sube la información al dominio
3. **Data Scientist:** quien desea acceder a los datos

El **Data Owner** sube los datos al dominio. El **Data Scientist** puede realizar operaciones sobre los datos sin la necesidad de manejar la información original: las operaciones se realizan sobre punteros a los datos y una vez que se desea obtener el resultado, éste se podrá obtener con o sin ruido, dependiendo de la situación. En el caso que se quiera obtener el resultado original, se deberá realizar un *request* para solicitar el acceso al **Data Owner** quien deberá aceptar la solicitud. En el caso que se desee obtener los datos con ruido, se debe utilizar el método *.publish(sigma)*, especificando el parámetro "sigma", correspondiente a la desviación estándar de la distribución utilizada por el mecanismo de privacidad diferencial para agregar ruido a la

respuesta, y luego utilizar el método *.get()* para obtener el valor perturbado. En este caso, no será necesario pedir permiso al **Data Owner**, pero de todas formas se realizan operaciones internas sobre los datos para calcular si el **Data Scientist** tiene un presupuesto de privacidad, o *privacy budget,* asignado para poder acceder a los datos.

Niveles de Privacidad Diferencial que provee Syft [8]:

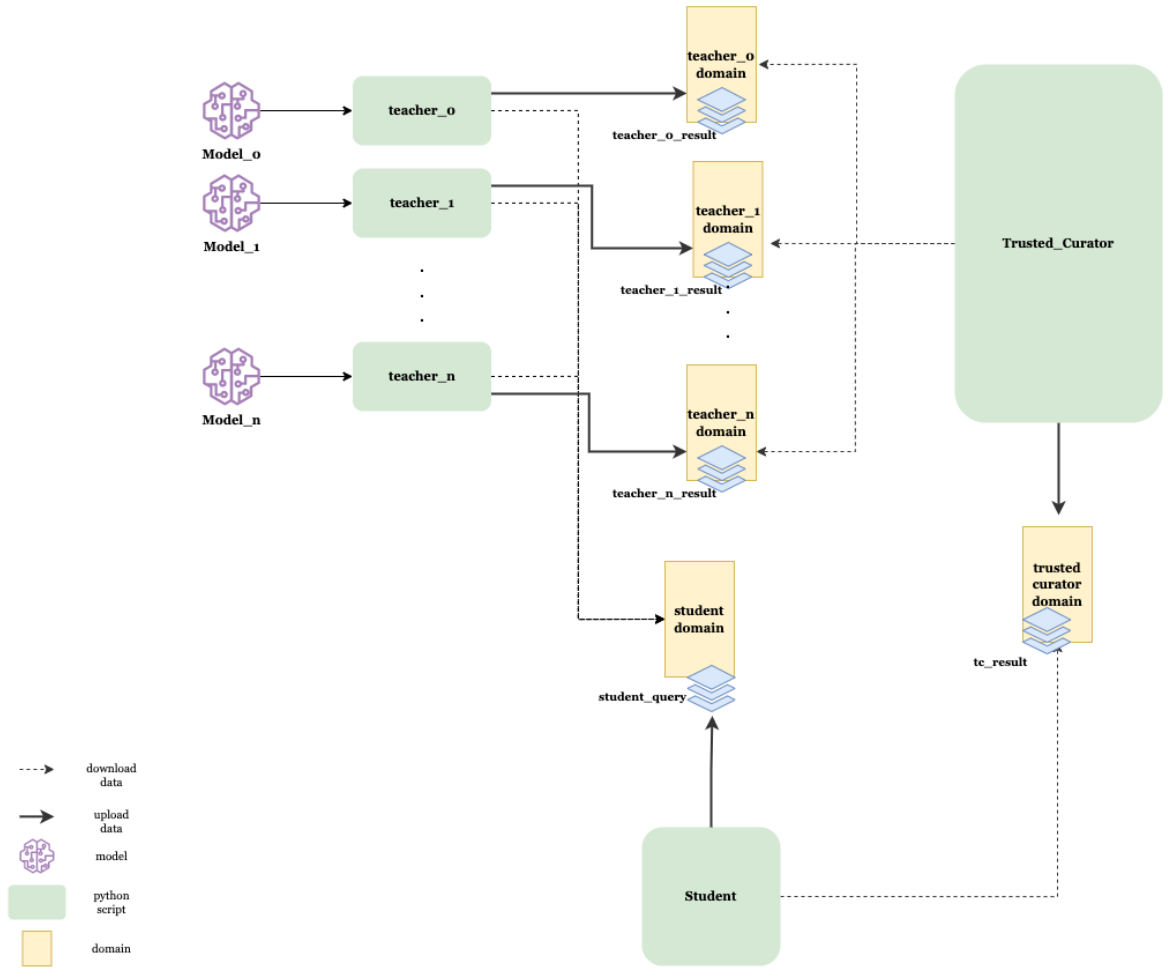
1. **Adversarial Differential Privacy:** el control del *privacy budget* se realiza a nivel del **Data Scientist** que desea acceder al dato.
2. **Individual Differential Privacy:** el resultado retornado no depende de un valor particular de un individuo (**Data Subject**). En este caso el *privacy budget* es rastreado a nivel individual de cada **Data Subject**.
3. **Automatic Differential Privacy:** Teniendo suficiente *privacy budget*, el **Data Scientist** puede obtener los resultados (con ruido), sin la necesidad de estar bloqueado esperando por el **Data Owner**.

Syft está implementado en python y provee una forma fácil de levantar los diferentes componentes de syft (backend, frontend, base datos, colas de mensajes, el dominio, entre otros) en contenedores Docker.

# Implementación

Para la implementación se utiliza un dominio de syft en Docker *containers*, se crean los diferentes **Data Scientist** que se utilizarán para consumir los datos. En principio, se utilizaría un dominio y **Data Owner** diferentes para cada uno de los componentes. Sin embardo, para simplicidad de la infraestructura requerida en el prototipo de prueba de concepto se utilizó el mismo dominio y el mismo **Data Owner**, pero es posible configurar para cada componente los dominios y usuarios a utilizar.

En el repositorio https://hdl.handle.net/20.500.12381/2372 se encuentra el código referente al prototipo.




**TrustedCurator.** Este componente realiza operaciones sobre el conjunto de resultados de los Teachers. Obtiene los punteros a los resultados de cada Teacher y construye el ensemble realizando operaciones sobre los punteros. En el caso de clasificación, realiza una votación y para el caso de predicción realiza un promedio.

**Votación (*Voting*).** Para esto, el resultado de los *Teachers* en el caso de tratarse de una clasificación, es un dataframe que contiene las siguientes columnas: "*result_0, result_1, .. ,*

*result_k",* siendo *k* la cantidad de clases. De esta forma, **TrustedCurator** puede contar para cada clase cuántos *Teachers* la "votaron"; todo esto sobre los punteros. Luego, que se tiene la suma para cada clase, se obtiene el resultado (utilizando el método *publish*) y se realiza un *argmax*, para saber cual es la clase más votada. Esto nos permite saber el resultado final, sin saber los resultados particulares de los *Teachers*, dado que la operación de suma de votos se hacen sobre los punteros.

**Promedio (*Average*)**. En el caso de una regresión, los *Teachers* retornan un dataframe con la columna "*result"* , con los valores resultado para cada una de las filas. Se realiza una suma de todos los resultados de los *Teachers* sobre los punteros, y luego se obtiene el resultado (utilizando el método *publish*). Al resultado se lo divide por la cantidad de *Teachers* para calcular el promedio. De esta manera, también se logra construir el ensemble sin acceder directamente a los datos resultado de los *Teachers.*

**Privacidad diferencial.** Cada vez que se utiliza el método *publish* para bajar información que está disponible en el dominio, el framework le agrega ruido mediante la aplicación de un mecanismo de privacidad diferencial. Esto quiere decir que los datos que se obtienen no son los datos originales, sino perturbaciones aleatorias de estos, garantizando así su privacidad. Si se desea tener acceso a los datos reales, es necesario realizar una solicitud expresa y esperar que **Data Owner** la apruebe.

# Instalación

1. Crear ambiente de trabajo:

```
python3.10 -m venv env
source env/bin/activate
```

2. Instalar dependencias:

```
git pull git@gitlab.com:ai-research-ort/pate-syft.git
cd pate-syft
pip install -r requirements.txt
git pull https://github.com/OpenMined/PySyft.git
git checkout dev
git pull
```

**Facultad de Ingeniería**
***Bernard Wand-Polak***
Cuareim 1451
11100 Montevideo, Uruguay
Tel. 2902 1505

```
cd packages/hagrid/
pip install -e .
```

3. Utilizar hagrid para levantar los contenedores Docker de syft:
   ```
   hagrid launch --dev --tail=false
   ```

# Ejecución

Para cada uno de los pasos siguientes es necesario estar dentro del ambiente de trabajo:

```
source env/bin/activate
cd src/
```

Para ejecutar cada uno de los componentes, se requiere realizarlo en consolas separadas. Si el dominio es remoto, podrían incluso ejecutarse desde diferentes computadoras o contenedores.

1. Crear usuarios necesarios y crear los modelos:
   ```
   cd src
   python setup.py
   ```

2. Iniciar **Student**: Para iniciar **Student** hay 3 scripts (según sea un problema de clasificación binaria o multiclase o regresión). Es necesario cambiar estos scripts para cargar otros datos.

   ```
   python start_student_binary_classifier.py
   python start_student_multiclass_classifier.py
   python start_student_regression.py
   ```

3. Iniciar **TrustedCurator**: para iniciar **TrustedCurator** hay 2 scripts (según sea un problema de clasificación o regresión). Es necesario pasar como parámetro cuantos teachers se utilizarán, en este caso 2. Y la cantidad de clases en el caso del clasificador, por defecto está predefinido en 2.

```
python start_trusted_curator_classifier.py 2 3
python start_trusted_curator_regression.py 2
```

4. Iniciar **Teacher 0**: para iniciar el **Teacher** hay 2 scripts (según sea un problema de clasificación o regresión). Es necesario especificar el identificador del teacher. Y la cantidad de clases en el caso del clasificador, por defecto está predefinido en 2.

   ```
   python start_teacher_classifier.py 0 3
   python start_teacher_regression.py 0
   ```

5. Iniciar **Teacher 1**

   ```
   python start_teacher_classifier.py 1 3
   python start_teacher_regression.py 1
   ```

# Observaciones

Este prototipo funciona para cuando el Student desea etiquetar un conjunto de datos. En el caso que desee etiquetar solamente un dato (un dataframe con una sola fila), este prototipo no es útil porque el framework garantiza privacidad sobre los **Data Subject**, en el caso de una sola fila, no podría garantizarse esta propiedad.

# Limitaciones

Syft tiene una gran curva de aprendizaje y es una herramienta que aún está en proceso de ser construida, por lo que es importante estar cerca de la comunidad para poder comprender su funcionamiento, relevar dudas y resolver errores. Es una comunidad muy activa y están avanzando rápidamente en cubrir varias funcionalidades, por lo que es prometedor este camino, pero aún no está pronto para ser utilizado en producción.

# Mejoras futuras

Se identifican los siguientes puntos para explorar en el futuro y mejorar la arquitectura propuesta:

1. Utilizar el adecuado seteo para las variables *sigma* y *privacy budget* para asegurar el correcto funcionamiento y garantizar que no hay pérdida de privacidad.
2. Los componentes **TrustedCurator** y **Teacher** deben ser modificados para esperar más de una petición y de esta forma poder soportar más de un **Student**.
3. El componente **Teacher** podría registrarse con **TrustedCurator** para disponibilizar su modelo, y en lugar de asignar el identificador mediante un parámetro sería asignado por **TrustedCurator**.
4. Mejorar logging y configuración de los componentes.
5. Crear containers Docker para cada uno de los componentes.

# Bibliografía

1. Papernot, N.; Abadi, M.; Erlingsson, U.; Goodfellow, I.; Talwar, K. Semi-supervised knowledge transfer for deep learning from private training data. arXiv 2016, arXiv:1610.05755. https://arxiv.org/pdf/1610.05755.pdf
2. C. Dwork, A. Roth. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, Vol. 9, Nos. 3–4 (2014) 211–407, 2014. https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf
3. 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. Extr. 2021, 3, 788-801. https://doi.org/10.3390/make3040039
4. J. Ramas, A. Rodríguez, S. Zanotta. Implementación de las prácticas de MLOps para PATE. Trabajo Final de Carrera, 2022. https://hdl.handle.net/20.500.12381/2362
5. OpenMined: Syft. https://github.com/OpenMined/PySyft.
6. R. Shokri, V. Shmatikov. Privacy-preserving deep learning. ACM SIGSAC Conference on Computer and Communications Security, CCS ’15, 2015.

7. Yi, X., Paulet, R., Bertino, E. (2014). Homomorphic Encryption. In: Homomorphic Encryption and Applications. SpringerBriefs in Computer Science. Springer, Cham.
8. Clase sobre Differential Privacy https://github.com/OpenMined/PySyft/blob/dev/notebooks/padawan/06-differential-privacy.ipynb