Blockchain para Mejorar la Reproducibilidad del Aprendizaje Automático

Abstract

La reproducibilidad de un experimento es un principio fundamental en la ciencia, que da lugar a validar y garantizar la integridad de resultados obtenidos. En la actualidad, los modelos de Machine Learning (ML) son ampliamente utilizados para obtener resultados y generar avances en una gran diversidad de áreas, tanto de la ciencia, como de la industria. Por lo tanto, la reproducibilidad de los resultados obtenidos con estos modelos es de gran relevancia para permitir la credibilidad de los mismos. Actualmente existen desafíos de la reproducibilidad en ML, ya que para poder llevar a cabo esta tarea puede ser necesario tener informacion relacionada al modelo entrenado, como el hardware subyacente, versiones de librerias, inicializaciones con elementos aleatorios, optimizaciones realizadas y documentación. En este trabajo se propone a la tecnología blockchain como una solución para favorecer la reproducibilidad de estos modelos. La tecnología blockchain se caracteriza por ser una base de datos transparente, trazable, altamente disponible e inmutable. Por lo tanto, se plantea utilizarla para mejorar la integridad y disponibilidad de las características de los procesos llevados a cabo en un experimento de ML. Se presenta como soluciones a tecnologías Blockchain basadas en la Ethereum Virtual Machine (EVM) o a la Blockchain Algorand, que dispone de la Algorand Virtual Machine (AVM).

Contents

1 Introducción

En los ámbitos de la ciencia y la industria, la reproducibilidad de experimentos permite aumentar la confianza y validar la consistencia de los resultados obtenidos, dando lugar a la posibilidad de generar avances y colaboraciones con mayor facilidad.

[Al momento de escribir tener en cuenta diferencias entre replicabilidad y reproducibilidad:

Defining Reproducibility and Replicability

Replicability is not Reproducibility: Nor is it Good Science]

En particular, en el área del aprendizaje automático, el progreso también depende de la reproducibilidad de los modelos y resultados con los que se ha tratado anteriormente. A pesar de la importancia de la reproducibilidad en ML, se enfrentan desafios debido a la complejidad que pueden presentar los entornos experimentales, como las dependencias de código y diferencias en versiones de librerías. Por otro lado, al momento de reproducir un modelo, los datos utilizados para entrenarlo pueden no estar disponibles o los preprocesamientos utilzados pueden no ser explicitados de forma suficiente. Además, a menudo los algoritmos de ML tienen componentes aleatorias que pueden influir en los resultados, por lo cual es relevante tener contabilidad de cómo estas son definidas.

También se resalta la importancia del código abierto y la disposición de los datos en forma pública para favorecer la reproducibilidad.

Expandir qué se habla en las referencias que están a continuación y cómo se alinean con la problemática mencionada.

- Improving Reproducibility in Machine Learning Research (2021)
- Reproducibility in Machine Learning-Driven Research (2023)
- Leakage and the reproducibility crisis in machinelearning-based science (2023)
- reforms: Reporting Standards for Machine Learning Based Science (2023)
- Traceability for Trustworthy AI: A Review of Models and Tools (2021)
- Reproducibility in PyTorch
- Advancing Research Reproducibility in Machine Learning through Blockchain Technology (2024)

2 Blockchain como herramienta para mejorar la reproducibilidad

[Agregar sección con una introducción más técnica de Blockchain. De todas formas mejor comenzar con las ideas mas importantes a tener en cuenta y luego enfocarse en introducciones para el lector.]

[Introducir hashes y mencionar los beneficios de utilizarlos para almacenar informacion.]

Blockchain es una tecnología que proporciona una base de datos que funciona como un registro inmutable, trazable y descentralizado. Puede ser utilizada para abordar la reproducibilidad de ML de la siguiente forma:

- **Registro inmutable:** Permite registrar cada etapa de un pipeline de ML, incluyendo configuraciones de hiperparámetros, preprocesamientos de datos y resultados. [Ver opción de incluir un hash del entorno de ejecucion para capturar versiones de software.]
- **Transparencia:** El registro queda asentado en la blockchain con su timestamp, permitiendo ser auditable y verificable, favoreciendo la replicabilidad de un modelo.
- **Integridad de los datos:** El almacenamiento de hashes de datos y modelos en la Blockchain permiten validar resultados obtenidos.

3 Información a trazar

[Además de almacenar hashes en blockchain, los datos y código deben estar disponibles a terceros.]

Para asegurar la reproducibilidad de modelos de ML, es fundamental trazar y registrar lo siguiente:

1. **Entorno**
    - Detalles del entorno de entrenamiento. Hardware y Software. Librerías y versiones.
2. **Datos**
    - Hash de datos utilizados para entrenamiento, validación y test.
    - Preprocesamiento de datos. Detalles de modificaciones y transformaciones al conjunto inicial de datos.
3. **Modelo**
    - Especificaciones de la arquitectura del modelo.
    - Hiperparámetros.
4. **Evaluación**
    - Metricas de evaluación del modelo. Entrenamiento, validación y test.
5. **Documentación**
    - Informes y documentación del experimento y resultados.

BlockchainTracer

BlockchainTracer es un paquete de Python diseñado para trazar información sensible y flujos de procesos en la blockchain.

Aprovecha las propiedades inherentes de la blockchain —inmutabilidad, transparencia, disponibilidad y trazabilidad— para registrar y auditar pasos secuenciales en cualquier proceso. Es ideal para aplicaciones que requieren registros verificables de acciones o rastros de datos sensibles.

¿Qué hace?

Permite guardar pasos secuenciales de cualquier cosa. Casos de uso:

- Mejorar la reproducibilidad de modelos de Machine Learning (principal idea).
- Subir hashes de archivos grandes de datos.
- Rastrear donaciones de ONGs.
- Mejorar la trazabilidad de cadenas de suministro.
- Guardar información importante de estudios científicos.
- Prueba de autoría (resultados con dirección y timestamp).
- Cualquier texto.
- Lo que quieras.

Implementación

Una única clase Python multipropósito.

Roadmap

Etapa 1

1. **Leer bibliografía relacionada sobre reproducibilidad en ML.**
    - *A Survey of Data Provenance in e-Science*
    - *Ensuring Trustworthy Neural Network Training via Blockchain*
    - *Towards Enabling Trusted Artificial Intelligence via Blockchain*
    - *BlockFlow: Trust in Scientific Provenance Data*
    - *ProML: A Decentralised Platform for Provenance Management of ML Systems*
    - *Blockchain Based Provenance Sharing of Scientific Workflows*
    - *Improving Reproducibility in Machine Learning Research (2021)*
    - *Reproducibility in Machine Learning-Driven Research (2023)*
    - *Leakage and the Reproducibility Crisis in ML-based Science (2023)*

- *reforms: Reporting Standards for ML-based Science (2023)*
- *Traceability for Trustworthy AI: A Review of Models and Tools (2021)*
- *Reproducibility in PyTorch*
- *Advancing Research Reproducibility in ML through Blockchain Technology (2024)*
- *Promoting Distributed Trust in ML and Simulation via Blockchain*
- *Blockchain Analytics and Artificial Intelligence*
- *Automatically Tracking Metadata and Provenance of ML Experiments*
- *Reproducibility in ML-based Research: Overview, Barriers and Drivers (2024)*
- *Model Cards for Model Reporting*, *Datasheets, Nutrition Labels, Factsheets*
- *ML Reproducibility Tools and Best Practices*

2. **Especificar diferenciadores de este trabajo.**
    - Trazabilidad de modelos ML en blockchains EVM mediante una API Python.
    - Código abierto.
    - Adherencia a estándares de reproducibilidad de estudios previos.
    - Capacidad de trazar otros procesos, pero con foco en ML.
    - Trazado del entorno computacional donde se entrenó el modelo.
    - Uso de IPFS o Arweave para datos grandes (guardar solo el hash en blockchain).
3. **Afinar el caso de ML: ¿Qué se necesita para una buena reproducibilidad?**
    - Checklist de reproducibilidad de NeurIPS 2019.
    - Estructura JSON con cada configuración del pipeline ML (hardware, entorno, preprocesado, hiperparámetros, semillas, métricas, versiones de paquetes, etc.).
    - Info sheet del modelo (según el paper de la crisis de reproducibilidad).
    - ¿Entorno estandarizado? ¿Docker siempre es necesario?

- Checklist de *reforms*.
- Perfil de descripción mínima.
- Model Cards.
- MLFlow para logging de experimentos (permite comparar versiones y funciona con sklearn, XGBoost, etc.).

4. **Decidir qué trazar.** ¿Usar todos los estándares y eliminar repeticiones? ¿Elegir uno? ¿Centrarse en narrativas, parámetros u otro subconjunto? Pensar en conjuntos. Model Cards y AutoML ya están testeados, se pueden usar como base.
5. **Ofrecer a los usuarios lo necesario para reproducir modelos.**
6. **Asegurar facilidad de uso y funcionamiento.**
   - Código Python accesible para usuarios técnicos no expertos en blockchain.
   - Integración con blockchains EVM.
   - Seguridad del manejo de llaves privadas.
   - Testing.

Etapa 2

7. Resolver qué hacer con código y binarios.
8. Integración con IPFS o Arweave para archivos grandes.

Etapa 3

9. Frontend para escalabilidad (uso por personas no técnicas).
10. Smart contract para descentralizar la ejecución del código.
11. Extensión a otras blockchains públicas.
12. Soporte para blockchains privadas.
13. Opción para trazar datos con una nueva dirección (anonimato).
14. Soporte para más RPCs además de Infura.
15. Automatizar el completado de info sheet del modelo. ¿Usar un LLM? ¿Generar .tex?

4 comentarios

tener en cuenta requirements.txt para reproducibilidad

5 Apéndice

5.1 Plataformas Blockchain

Evaluamos dos posibilidades: blockchains basadas en EVM y la blockchain Algorand.

5.1.1 Ethereum Virtual Machine

- Ethereum y Polygon, algunos ejemplos de blockchains con EVM.
- Contratos inteligentes escritos en Solidity, el lenguaje más aceptado en blockchain.

5.1.2 Algorand

- Ofrece bajos fees de transacciones y rápida validación de las mismas.
- Permite volumen considerable de transacciones.
- Los contratos inteligentes se pueden escribir en Python.