Facultad de Ciencias Económicas y Empresariales
ICADE

ANÁLISIS COMPARATIVO DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO PARA LA DETECCIÓN DE TRANSACCIONES ILÍCITAS EN BITCOIN

TRABAJO FIN DE GRADO

Autor: Rodrigo de Gracia de Dios
Director: Susana Josefa Gago Rodríguez

MADRID | Junio 2026

Contenido

Resumen

La detección de blanqueo de capitales en redes de criptomonedas representa un desafío cada vez mayor para los organismos reguladores tradicionales debido a su creciente uso y las características de este tipo de transacciones como la descentralización, anonimato y alcance global. En este contexto, el presente trabajo tiene como objetivo implementar y comparar siete modelos de aprendizaje automático para la detección de transacciones ilícitas en la red de Bitcoin, evaluando distintos enfoques y familias de algoritmos.

El estudio se realiza sobre el dataset Elliptic, que contiene más de 200.000 transacciones de Bitcoin etiquetadas de forma temporal y con clara asimetría entre sus clases (9,7% de las transacciones son ilícitas frente al 90,3% lícitas sobre el total de los datos etiquetados). Los modelos evaluados incluyen regresión logística, Random Forest, XGBoost, un perceptrón multicapa (MLP), Isolation Forest, KNN y LightGBM. A este último modelo se añadiran como variables extra un conjunto de representaciones vectoriales de los nodos de la red generadas mediante la librería node2vec. La comparación se realiza bajo tres configuraciones de variables distintas: sólo variables locales (LF: Local Features), todas las variables (AF: All Features) y todas las variables junto con las representaciones vectoriales anteriormente mencionadas (AF + NE: Node Embedding). La métrica principal de evaluación es el F1-score sobre la clase ilícita en línea con la literatura previa.

Los resultados muestran que LightGBM con todas las variables (AF) obtiene el mejor rendimiento (F1 de 0,818, precisión de 0,93), seguido muy de cerca por su variante con representaciones vectoriales de nodos (F1 de 0,817) y Random Forest en su versión simplificada (F1 de 0,815). La incorporación de estas representaciones vectoriales no aporta mejoras significativas al modelo pero tampoco perjudican el rendimiento, lo que sugiere que las variables agregadas del dataset ya capturan la información estructural de cada transacción. Todos los modelos sin excepción experimentan una caída considerable de rendimiento a partir del timestep 43, mostrando la debilidad de estos algoritmos ante cambios bruscos en la estructura de los datos.

Como conclusión, se recomienda Random Forest como primera opción para una aplicación real por su simplicidad y su excelente rendimiento incluso sin optimización de hiperparámetros, lo que aumentaría sus probabilidades de adopción por parte de organizaciones y analistas sin formación especializada en aprendizaje automático. Como segunda opción se proponen los modelos de gradient boosting (específicamente LightGBM), que ofrecen el mejor rendimiento del estudio a cambio de una mayor complejidad y la necesidad de un proceso de

optimización, resultando especialmente adecuados para entidades con mayor capacidad técnica y computacional.

Palabras clave

Bitcoin, Blanqueo de Capitales, Blockchain, Aprendizaje Automático, Criptomonedas.

Introducción

Contexto del problema

Bitcoin, creado por Satoshi Nakamoto en 2008, se planteó originalmente como un sistema de dinero electrónico que permitía a los usuarios realizar transacciones en línea sin la necesidad de un intermediario financiero que validara los movimientos (Nakamoto, 2008).

Su aportación principal es resolver el problema del doble gasto (la posibilidad de que una misma moneda digital pueda enviarse a varios destinatarios distintos) mediante una red entre iguales y un mecanismo basado en lo que Satoshi llama prueba de trabajo (proof-of-work en inglés), que permite validar las transacciones y otorgar incentivos a participar en la red (Nakamoto, 2008). En términos sencillos, la Blockchain funciona como un libro de registro compartido que agrupa transacciones en bloques públicos y los enlaza mediante criptografía. De este modo modificar el historial requiere rehacer el trabajo computacional asociado a esos bloques, lo que vuelve la manipulación de la red extremadamente costosa (Nakamoto, 2008).

Esta arquitectura está diseñada para operar sin un tercero que valide las operaciones, explicando por qué las criptomonedas han supuesto una innovación tan relevante en el contexto de los pagos digitales, desplazando parte del control que tradicionalmente ejercían los intermediarios financieros en la supervisión de transacciones monetarias.

Otra de las características principales de las criptomonedas y Bitcoin en particular, es que su diseño no exige en condiciones normales que los usuarios proporcionen datos personales para participar en la red. Para acceder a la red se utilizan pseudónimos (direcciones de Bitcoin), que son gestionados a través de una cartera (lo que se conoce como wallet). Cada dirección se genera a partir de un par de claves publica-privada y son estas claves las que permiten autorizar las

transacciones de criptomonedas e identificar la cuenta del usuario (Androulaki et al., 2013). Esta ausencia de vinculación explícita ha generado la idea de que las transacciones en Bitcoin son completamente anónimas, especialmente en comparación con los sistemas financieros tradicionales, donde los pagos están asociados a cuentas bancarias verificadas legalmente y que deben cumplir con las regulaciones establecidas como el KYC (Know Your Customer en inglés). Además, esta percepción ha sido reforzada por la asociación del uso de Bitcoin a los mercados ilegales dentro de la Dark Web, principalmente durante sus primeros años.

Sin embargo, la literatura académica ha demostrado que Bitcoin no es un sistema anónimo en sentido estricto. Aunque las direcciones no contienen información personal directamente, todas las transacciones están registradas de forma permanente y pública en la Blockchain, permitiendo observar los flujos de actividad entre direcciones a lo largo del tiempo. Esta característica permite aplicar técnicas que agrupan direcciones o analizan transacciones para poder predecir con alta probabilidad que un conjunto de movimientos pertenece a una misma entidad o usuario.

Igualmente, estas transacciones pueden relacionarse con comportamientos observables fuera de la red (como pagos exagerados), permitiendo deducir la identidad real de los usuarios a pesar de que las transacciones no incluyan sus datos de forma directa. En conclusión, Bitcoin sí que ofrece una gran capa de opacidad sobre la identidad del usuario, pero es falso decir que es totalmente anónimo, debido a que los movimientos en la red son totalmente transparentes.

Por su parte el blanqueo de capitales puede definirse en términos generales como el conjunto de prácticas orientadas a ocultar el origen ilegal de fondos para integrarlos en la economía como si fueran lícitos (Financial Action Task Force [FATF], s.f.). Históricamente, este proceso se ha estudiado en el contexto del sistema financiero convencional donde los intermediarios como bancos desempeñaban un papel fundamental en la aplicación de controles preventivos.

Sin embargo, la aparición de las criptomonedas introduce un cambio radical al permitir la transferencia de dinero a través de plataformas tecnológicas que no requieren la intermediación de instituciones financieras centralizadas y facilitan movimientos globales con prácticamente cero fricciones. Esta combinación de descentralización, anonimato, alcance global y rapidez en las transacciones ha aumentado las preocupaciones en torno al uso de criptomonedas para fines de blanqueo de capitales, entre otros muchos usos ilegales. Para dimensionar el problema, se estima que el volumen de transacciones ilícitas llego a los $158.000 millones en 2025, más que el doble del año anterior, impulsado en gran parte por el crecimiento generalizado del uso de criptomonedas (TRM Labs, 2026). Son

precisamente estas características, las que han motivado la creación de este trabajo, con el objetivo de determinar la efectividad de algoritmos de aprendizaje automático en este contexto.

Igualmente, y como respuesta a esta transformación del ecosistema financiero, los organismos internacionales encargados de establecer estándares para la prevención del blanqueo de capitales han ido creando progresivamente un marco de trabajo para luchar contra estas actividades dentro del ecosistema cripto. En este aspecto destaca especialmente el Grupo de Acción Financiera Internacional (FATF por sus siglas en ingles), organismo internacional creado en 1989 y considerado la autoridad global en materia AML (Anti Money Laundering; prevención de lavado de capitales en español).

En este contexto, el FATF introdujo el concepto de "activos virtuales", categoría que engloba bienes digitales que pueden utilizarse como métodos de pago o inversión; y "proveedores de servicios de activos virtuales" (VASP por sus siglas en inglés), con el objetivo de identificar los elementos principales del ecosistema cripto (FATF, 2019). De esta forma no solo se centra el foco en el activo subyacente (la criptomoneda) sino en todos los puntos de concentración que existen en la red como plataformas de intercambio o servicios de custodia, donde instituciones gubernamentales pueden solicitar acceso y colaboración, además de obligarles a aplicar las regulaciones necesarias.

Aun con este marco establecido, las características propias de las redes de criptomonedas plantean retos especialmente complejos para las técnicas tradicionales de AML. Muchos patrones de actividad ilícita no se explican solo por elementos aislados de una operación (cantidad, fecha, etc.), sino por la estructura y las relaciones entre transacciones y su evolución temporal. Por ello, un enfoque creciente en la investigación actual es tratar el problema como un análisis de redes donde se modela la actividad como un grafo (un esquema donde entidades y transacciones se representan en forma de nodos y enlaces) y se aplican técnicas de aprendizaje automático capaces de aprovechar esa información. En el presenta trabajo, a diferencia de la mayoría de estudios previos, no solo nos centraremos en el rendimiento neto de los modelos, sino que también tendremos en cuenta que tan fácil sería aplicar estos algoritmos en la práctica, asumiendo que no todas las organizaciones y empleados pueden tener los conocimientos o capacidades suficientes como para ejecutar los modelos más complejos, además de los requisitos regulatorios que puedan afectar a estas instituciones.

En conclusión, la combinación entre la creciente importancia de las criptomonedas en el sistema financiero y las limitaciones de los enfoques tradicionales de AML, ponen de manifiesto la necesidad de desarrollar herramientas analíticas más avanzadas para tratar este nuevo ecosistema

tecnológico. Motivado por este contexto, el presente trabajo de fin de grado tiene como objetivo explorar y comparar diferentes técnicas de aprendizaje automático para la detección de actividades potencialmente ilícitas en redes de criptomonedas y especialmente en la red Bitcoin, para comprobar su capacidad de capturar la naturaleza y estructura de este tipo de tecnología.

Pregunta de investigación y objetivos

A partir del contexto expuesto en el apartado anterior, la investigación presentada en este trabajo gira en torno a la siguiente pregunta: hasta qué punto las técnicas de aprendizaje automático pueden contribuir a mejorar la detección y prevención de actividades ilegales en redes de criptomonedas, principalmente Bitcoin.

De forma más específica, esta pregunta se desglosa en tres objetivos. El primero consiste en entrenar un conjunto de modelos de aprendizaje automático aplicados a la red Bitcoin, cubriendo las principales familias de algoritmos: un modelo lineal (Regresión Logística), modelos de ensamble (Random Forest, XGBoost y LightGBM), un modelo no supervisado (Isolation Forest), una red neuronal (MLP) y un modelo basado en distancias (KNN). El segundo consiste en evaluar el rendimiento de los modelos mediante métricas adecuadas al contexto del problema, principalmente el F1-score sobre la clase ilícita. Como tercer y último objetivo, se realizará un análisis para determinar que modelo resultaría más adecuado para su aplicación en la práctica, teniendo en cuenta no solo los resultados obtenidos, sino también las características propias de cada modelo y su posible uso en un entorno real.

Estructura del trabajo

El proyecto se organiza en cinco bloques. En primer lugar, la introducción anteriormente presentada, contextualiza el problema, formula las preguntas de investigación y presenta los objetivos del trabajo. A continuación, el marco teórico abordará los fundamentos de la tecnología Blockchain, las técnicas de blanqueo de capitales en entornos digitales y los principios del uso de técnicas de aprendizaje automático aplicados a este ámbito. El tercer bloque describirá la metodología seguida durante el proyecto, incluyendo la descripción del dataset utilizado, el tratamiento de la asimetría entre clases, la configuración de variables de entrada, las métricas de evaluación y los modelos seleccionados. Finalmente, el cuarto bloque presentará los resultados obtenidos y el quinto discutirá las implicaciones prácticas de los modelos en un contexto de uso real.

Marco teórico

Fundamentos de la tecnología Blockchain y Bitcoin

Los principios de la tecnología

Podemos situar el origen de la tecnología Blockchain tal y como la conocemos en la actualidad en el documento publicado por Satoshi Nakamoto en 2008. Sin embargo, cabe mencionar que la tecnología no partía de cero, sino que se basaba en conceptos previos entre los que destaca el trabajo de Haber y Stornetta (1990), donde ya se identificaban conceptos que Satoshi usaría en su proyecto, como el encadenamiento de bloques mediante funciones hash (algoritmos que transforman datos en cadenas de longitud fija) para garantizar el orden de los documentos. El propósito establecido era ofrecer una versión descentralizada de dinero electrónico que permitiera enviar pagos directamente sin intermediario financiero. Para lograrlo, Bitcoin resolvía simultáneamente varios problemas técnicos que habían limitado intentos anteriormente, entre los que se incluyen la prevención del doble gasto sin un intermediario, el establecimiento de un orden cronológico verificable para las transacciones y la creación de incentivos económicos que mantuvieran la integridad de la red.

Además, el documento define una serie de elementos técnicos que serán la base de su estructura como una red entre iguales (peer to peer en inglés) con transacciones basadas en criptografía y unidas a una clave pública, bloques enlazados criptográficamente formando una cadena, un mecanismo basado en la llamada “prueba de trabajo” que evita que pueda modificarse la cadena de bloques; y un sistema de incentivos que otorga recompensas a los nodos que siguen las reglas establecidas. Otro de los conceptos clave introducidos es que, para añadir un nuevo bloque a la cadena, el nodo tiene que resolver un problema matemático extremadamente complejo, lo que requiere de mucha potencia de cómputo. En este caso, involucra obtener un valor que tras aplicar el algoritmo hash, el hash resultante empiece con un determinado número de ceros (Nakamoto, 2008). Ese esfuerzo es lo que se conoce como prueba de trabajo y es muy costoso en recursos del ordenador. Sin embargo, verificar que alguien resolvió el problema es instantáneo, lo que hace que falsificar el historial sea prácticamente imposible porque se tendría que volver a realizar todo el trabajo de computación realizado para todos los bloques de la cadena.

Estructura del grafo de transacciones de Bitcoin

Bitcoin opera sobre un modelo de salidas no gastadas (UTXO, por sus siglas en inglés). El funcionamiento del esquema UTXO se basa en que cada transacción

toma como punto de partida la salida de una operación anterior, consumiéndola y generando a su vez nuevas salidas que podrán ser utilizadas en operaciones más adelante (Motamed y Bahrak, 2019). Esta estructura se traduce en un grafo de transacciones donde los nodos representan las direcciones de las cuentas y los arcos representan los movimientos y operaciones entre ellas (Motamed y Bahrak, 2019).

El punto clave para el contexto de este trabajo es que, aunque las direcciones de Bitcoin son pseudoanónimas (no tienen datos directamente identificativos del usuario), todas las transacciones son públicas y permanentes en la Blockchain.

En este contexto, han surgido numerosos estudios que demuestran las vulnerabilidades de este falso anonimato de Bitcoin como el de Reid y Harrigan (2012), donde los autores mostraron que analizando el historial público de transacciones era posible asociar varios movimientos o cuentas a una misma entidad. Trabajos más recientes como el de Caringella et al. (2024) han mejorado estas técnicas combinando múltiples heurísticas de agrupación de direcciones (técnicas de clustering por su nombre en el ámbito de aprendizaje automático) para identificar con mayor precisión las entidades que operan detrás de las transacciones de Bitcoin.

Blanqueo de capitales en entornos digitales

La doctrina internacional para el blanqueo de capitales identifica tres fases que suelen ocurrir durante este proceso: colocación, estratificación e integración. Estas categorías, planteadas originalmente para los sistemas financieros tradicionales, son igualmente aplicables al ecosistema cripto, aunque los métodos se han adaptado a la nueva tecnología.

La fase de colocación consiste en la introducción de los fondos ilícitos en el sistema financiero (FATF, s.f.). En el ámbito cripto, esta fase puede incluir la conversión de efectivo a criptomonedas a través de cajeros automáticos especializados, operaciones en plataformas con controles de identificación deficientes o directamente cuando los fondos se generan y se mueven en mercados alternativos, como ocurre en los pagos ilegales dentro de la Dark Web.

La fase de estratificación busca alejar a los fondos de su origen criminal mediante operaciones diseñadas para dificultar el rastro de las transacciones (FATF, s.f.). En este contexto se utilizan técnicas como los mezcladores, servicios que agrupan depósitos de múltiples usuarios y los redistribuyen a otras direcciones rompiendo el vínculo original entre el emisor y el destinatario final a cambio de una comisión (por ejemplo, Coinomize); las monedas de privacidad como Zcash, que ocultan mediante diferentes tecnologías la información de las transacciones; o mover el

dinero entre diferentes Blockchains. El uso de estas tres técnicas sigue creciendo en la actualidad posicionándose como las herramientas principales de estratificación en el ecosistema cripto. Además la ejecución de estas técnicas a través de plataformas situadas en países con baja regulación para el AML sigue facilitando estos procesos al evitar controles de identificación que exigen los marcos regulatorios más estrictos (Europol, 2026).

Por último, la fase de integración finaliza el proceso volviendo a introducir los fondos en el sistema económico legal (FATF, s.f.). Esto puede realizarse mediante la conversión de criptomonedas a efectivo a través de plataformas cripto o la inversión en activos legales que acepten esta forma de pago.

De cara a coordinar y orientar a los participantes del ecosistema cripto y poder hacer frente a las fases anteriormente mencionadas, la FATF elaboró una lista de indicadores de alerta que pueden indicar posibles operaciones para blanquear dinero dentro de este tipo de redes. Entre los principales indicadores se encuentran múltiples transferencias de pequeña cantidad, realizar varias transacciones de alto valor en cortos periodos de tiempo, movimientos que involucran jurisdicciones de alto riesgo, uso reiterado de mezcladores o monedas de privacidad y operaciones cuya magnitud es claramente inconsistente con el perfil económico declarado por el usuario (FATF, 2020). Estos indicadores constituyen una base para los sistemas tradicionales de monitorización, pero su aplicación deja sin detectar esquemas más sofisticados, lo que buscamos resolver mediante los enfoques de aprendizaje automático planteados en este trabajo.

Aplicaciones del aprendizaje automático en la detección de blanqueo de capitales

Evolución histórica y metodológica

Una vez establecidas las características de Bitcoin y los retos que plantea el blanqueo de capitales en este contexto, esta sección mostrará cómo la literatura científica ha abordado el problema, especialmente mediante técnicas de aprendizaje automático.

El análisis de datos mediante el uso de modelos para la detección de fraude se remonta a finales del siglo XX, teniendo especial relevancia las técnicas de aprendizaje automático para identificar fraude en las tarjetas de crédito (Khan et al., 2025). Ngai et al. (2011), en una revisión de la literatura entre 1997 y 2008, documentaron como estas técnicas se fueron aplicando principalmente al ámbito financiero incluyendo el fraude corporativo, de seguros y de nuevo el de tarjetas de

crédito. En una revisión más reciente, Deprez et al. (2026) analizaron 97 artículos publicados en el contexto del aprendizaje automático en AML, documentando la aparición de modelos más complejos y un crecimiento a partir de 2018 de los modelos de redes neuronales de grafo. Igualmente, el estudio remarca las limitaciones de este tipo de modelos dependiendo del tipo de datos analizados, con especial atención al desequilibrio entre clases (Deprez et al., 2026), una de las características principales del dataset que utilizaremos en este trabajo.
En el contexto específico de las criptomonedas, uno de los momentos más relevantes en la literatura científica se produce en 2019 con la publicación del trabajo de Weber et al. (*Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics)* y la publicación del dataset Elliptic (dataset que utilizaremos para nuestro análisis), que permitieron crear una línea de investigación comparable entre distintos investigadores. El trabajo realizado por Weber et al. (2019) presenta a Random Forest como el mejor modelo, superando con un amplio margen al modelo GCN (representación de las redes neuronales de grafo). En trabajos posteriores se han refinado los enfoques, pero cabe recalcar el excelente desempeño obtenido por el Random Forest a pesar de su aparente simplicidad frente a modelos más complejos. Por ejemplo, Islam et al. (2025) compararon modelos de regresión logística, Random Forest y SVM (Maquina de Vectores de Soporte en español) sobre transacciones Bitcoin, destacando Random Forest como el modelo con mejor puntuación.

Igualmente, Ajagbe et al. (2025), usando datos proporcionados por IBM, realizaron una comparación entre XGBoost (Gradient Boosting), K-NN (K-Nearest Neighbors), Random Forest, Isolation Forest (modelo no supervisado) y SVM, posicionándose esta vez XGBoost como el mejor algoritmo. Además, para el contexto de este trabajo, Pettersson Ruiz y Angelis (2022) realizaron una comparación entre modelos usando el dataset Elliptic y complementándolo con entrevistas cualitativas a analistas de plataformas de criptomonedas, concluyendo que los árboles de decisión son los algoritmos con mayor potencial de adopción en el sector, a pesar de tener un rendimiento peor que otros modelos. El artículo además destaca la importancia del cumplimiento de las regulaciones a la hora de una posible adopción de estas técnicas, por lo que es fundamental poder explicar las causas por las que un modelo marca una transacción como ilícita (Pettersson Ruiz y Angelis, 2022).

Este punto, en ocasiones olvidado por la literatura científica, es clave para entender una posible adopción real de estas tecnologías. Poder explicar los principios que hay detrás de la toma de decisiones de estos algoritmos es un requisito fundamental a la hora de poner en práctica nuevas tecnologías en el ámbito de AML, especialmente para organismos regulados (FATF, 2021).

En su conjunto, la literatura coincide en señalar a los modelos basados en árboles como Random Forest y a los modelos de Gradient Boosting como algunos de los modelos más prometedores a pesar de su aparente simplicidad, pero es necesario tener en cuenta que los resultados dependen en gran medida del dataset utilizado, las particiones de datos empleadas y los algoritmos comparados. Es por ello que este proyecto propone trabajar con modelos de distintas familias de forma que puedan compararse dentro de un mismo dataset y división de datos.

Si bien los resultados presentados por Weber et al. (2019) para sus modelos basados en redes neuronales de grafos obtienen un rendimiento bajo en comparación con los modelos tradicionales, cabe recalcar que trabajos más recientes han mejorado los resultados de este planteamiento para el dataset Elliptic. Sin embargo, comparar resultados entre estudios puede resultar complicado debido al uso de diferentes metodologías, especialmente a lo que respecta a la partición de datos. Como se explicará en la siguiente sección, este trabajo realizará una división temporal de los datos siguiendo el planteamiento original sobre Elliptic, mientras que otros estudios han empleado particiones distintas (especialmente particiones que dividen los datos de forma aleatoria), que obtienen resultados considerablemente mejores, aunque la comparación directa puede resultar engañosa. Por ejemplo, Asiri y Somasundaram (2025) reportan un F1-score de 0,93 con un modelo básico de GCN y modelos más complejos como ChronoWave-GNN llegan a obtener un F1 de 0,98 (Lin et al., 2026). En este estudio la comparación se realizará con trabajos que utilicen una división temporal basada en los timesteps del dataset, por lo que trabajos que usen diferentes particiones no serán tenidos en cuenta. Aun así, para tener una referencia comparativa, nuestro mejor modelo (LightGBM con todas las variables) obtiene un F1 de 0,97 utilizando una partición aleatoria de 80/10/10 para entrenamiento, validación y test.

Por qué Bitcoin se representa como un grafo

Para comprender por qué los modelos basados en grafos pueden resultar de gran utilidad para este problema, primero se debe aclarar qué es un grafo y en qué se diferencia de una tabla convencional.

Un grafo es una estructura que representa un conjunto de entidades (nodos) y las conexiones o vínculos que existen entre ellas (arcos o aristas) (Zhou et al., 2020). Un ejemplo sencillo sería una red social, donde cada persona es un nodo y cada relación o amistad es un arco que conecta dos nodos. Lo que distingue a un grafo de una tabla son precisamente estas relaciones, ya que los nodos están conectados entre sí y esa conexión aporta información relevante (Figura 1).

En Bitcoin, la arquitectura del grafo está definida de forma estructural dentro del sistema UTXO como se ha explicado anteriormente. Cada dirección o identificador de la transacción puede representarse como un nodo y cada flujo monetario de una dirección a otra se representa como un arco (Motamed y Bahrak, 2019). El resultado es un grafo de transacciones a gran escala donde la posición de cada transacción en la red, sus vecinos y los patrones de conexión que la rodean contienen información que no está recogida en las características individuales de una transacción aislada.

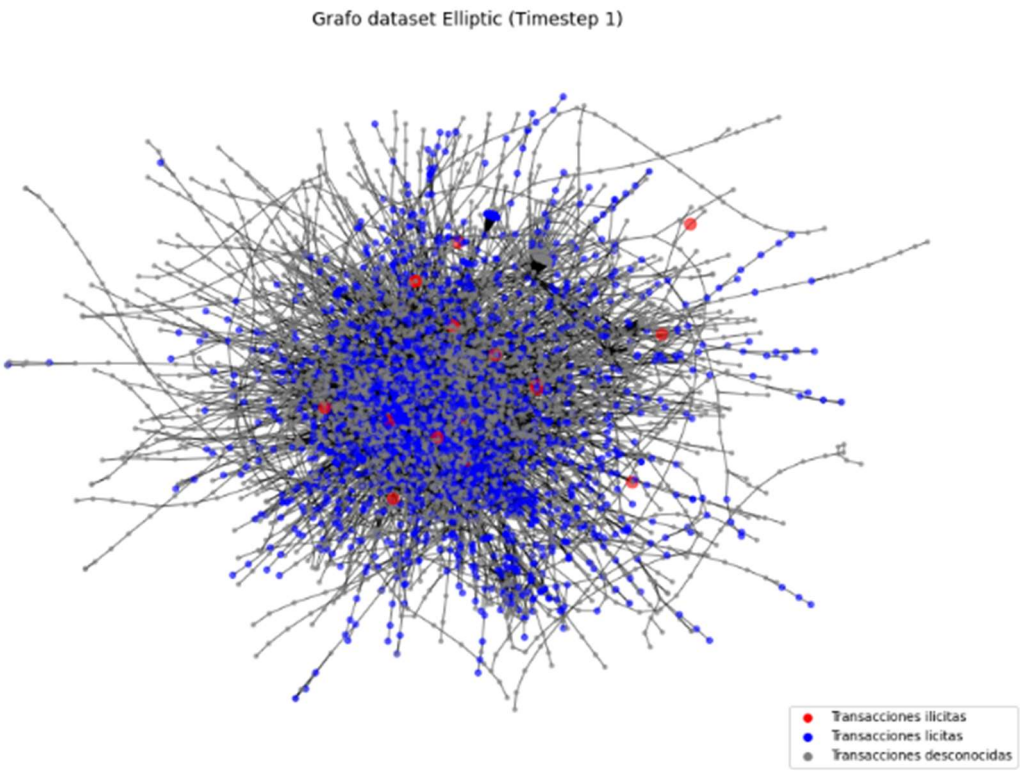

Figura 1: Representación gráfica de la estructura de grafo del dataset Elliptic

Esta premisa es el argumento que justifica el uso de modelos basados en grafos para la detección de blanqueo de capitales en Bitcoin. Sin embargo, su rendimiento en la práctica depende de la estructura del dataset y de una condición que no siempre se cumple y es que dicha información entre conexiones no esté ya disponible en otro formato. El dataset concreto sobre el que se trabaja en este estudio, el dataset Elliptic, presenta precisamente esta estructura, como se detallará más adelante. Elliptic no incluye únicamente las características propias de cada transacción, sino que incluye un segundo grupo de variables que resumen la información sobre las transacciones más próximas en la red. Esto ayuda a explicar por qué los modelos de grafo no obtienen la ventaja que cabría

esperar a partir de la teoría explicada en esta sección y porque no resultan tan atractivos para este dataset en particular.

Metodología

Una vez establecidos los fundamentos teóricos iniciales para la detección de blanqueo de capitales en la Blockchain, este capítulo describe el diseño y metodología experimental del trabajo. Se partirá del dataset Elliptic, presentando su estructura, propiedades y limitaciones; se justificarán las decisiones tomadas para tratar el desequilibrio entre clases y la configuración de las variables de entrada utilizadas en los experimentos. Además, se presentan los modelos seleccionados, así como los criterios que han guiado su elección y la forma en la que se ha planteado su implementación. El objetivo de este apartado, por tanto, es construir un marco comparativo que permita entender y evaluar de forma justa los diferentes modelos utilizados bajo un mismo conjunto de datos y unas mismas condiciones.

Estructura y propiedades del dataset Elliptic

El dataset Elliptic fue publicado en 2019 como parte de la colaboración entre la empresa de análisis Blockchain Elliptic y el MIT-IBM Watson AI Lab, dentro del trabajo anteriormente mencionado de Weber et al. (2019). Su publicación supuso un punto de inflexión para la investigación en este campo, ya que en el momento de su aparición era el dataset de transacciones de criptomonedas más grande disponible públicamente (Weber et al., 2019), lo que permitió que trabajos posteriores pudieran comparar sus resultados sobre una base común.

El grafo que conforma el dataset contiene 203.769 transacciones representadas como nodos y 234.355 conexiones o aristas, donde cada una indica un flujo de pago en la red. Cabe mencionar que este dataset es simplemente una muestra del grafo total de Bitcoin, lo que significa que algunos nodos pueden haber perdido parte de sus conexiones originales, lo que puede romper patrones que en la red completa serían visibles (Alarab et al., 2020). El dataset se divide en tres ficheros. El primero, elliptic_txs_features.csv, contiene las 166 variables anónimas de cada transacción junto a su identificador. El segundo, elliptic_txs_classes.csv, recoge la etiqueta de cada transacción:1 (ilícita), 2 (lícita) o “unknown” para las no etiquetadas. El tercero, elliptic_txs_edgelist.csv, incluye parejas de identificadores que indican el flujo de cada transacción.

	0	1	2	3	4	5	6	7	8	9	...
0	230425980	1	-0.171469	-0.184668	-1.201369	-0.121970	-0.043875	-0.113002	-0.061584	-0.162097	...
1	5530458	1	-0.171484	-0.184668	-1.201369	-0.121970	-0.043875	-0.113002	-0.061584	-0.162112	...
2	232022460	1	-0.172107	-0.184668	-1.201369	-0.121970	-0.043875	-0.113002	-0.061584	-0.162749	...
3	232438397	1	0.163054	1.963790	-0.646376	12.409294	-0.063725	9.782742	12.414558	-0.163645	...
4	230460314	1	1.011523	-0.081127	-1.201369	1.153668	0.333276	1.312656	-0.061584	-0.163523	...
...	...	...	...	...	...	...	...	...	...	...	...
199995	4605578	48	-0.171209	-0.112272	1.018602	-0.121970	-0.063725	-0.113002	-0.061584	-0.161829	...
199996	3335089	48	0.039316	-0.123131	1.018602	-0.121970	-0.043875	-0.113002	-0.061584	0.053570	...

Figura 2: fichero elliptic_txs_features.csv

	txId	class
0	230425980	unknown
1	5530458	unknown
2	232022460	unknown
3	232438397	2
4	230460314	unknown

Figura 3: fichero elliptic_txs_classes.csv

	txId1	txId2
0	230425980	5530458
1	232022460	232438397
2	230460314	230459870
3	230333930	230595899
4	232013274	232029206

Figura 4: fichero elliptic_txs_edgelist.csv

De las 203.769 transacciones totales, solo el 23% dispone de etiqueta: 4.545 son ilícitas (2% del total), 42.019 son lícitas (21%) y el resto de transacciones se representan sin etiquetar (77%). Este proceso de etiquetado se basó en patrones de comportamiento y no en confirmaciones oficiales, siguiendo una heurística donde las entidades que reutilizan direcciones y agrupan varios pagos en una sola transacción, algo típico de plataformas, tienen más probabilidades de ser lícitas. Por el contrario, las que operan con un menor número de entradas para reducir su exposición, tienen más probabilidades de ser ilícitas (Weber et al., 2019). Esta forma de etiquetar puede introducir un cierto margen de error, ya que no todas las clasificaciones pueden considerarse correctas con total fiabilidad.

Cada transacción está descrita por 166 variables numéricas y anónimas (a excepción del Timestep) (Figura 2), es decir, sin nombres que expliquen qué mide cada una, lo que hace que cualquier análisis sobre qué variables son más relevantes para detectar el fraude pierda utilidad en la práctica. Igualmente se dividen en dos grupos con naturalezas muy distintas. El primer grupo tiene 94 variables que recogen información propia de cada transacción, como el momento

en el que ocurrió, el número de pagos recibidos y enviados, las comisiones pagadas a la red o el volumen total transferido (Weber et al., 2019). Todo esto puede saberse mirando una transacción de forma aislada, sin necesidad de saber nada sobre las transacciones que la rodean. El segundo grupo tiene 72 variables que resumen las conexiones a su alrededor, concretamente, el valor máximo, mínimo, desviación típica y los coeficientes de correlación de la información de las transacciones directamente conectadas (Weber et al., 2019). Esto significa que estas 72 variables ya contienen información sobre el entorno de cada transacción, lo que permite que modelos tradicionales puedan tener información de la red sin tener que recurrir a técnicas complejas.

Esta es la implicación más relevante para el contexto de este proyecto, debido a que, al analizar las 166 variables en su conjunto, un modelo como Random Forest ya está recibiendo información sobre el entorno de cada transacción sin necesidad de analizar el grafo directamente, reduciendo la ventaja que en teoría deberían tener los modelos de grafo. Weber et al. (2019) observaron exactamente este efecto en su estudio, donde el modelo de Random Forest con las 166 variables obtuvo un F1 de 0,79 sobre las transacciones ilícitas, mientras que la GCN en su versión base solo llegó a 0,63, mostrando uno de los peores resultados de todos los modelos evaluados. La diferencia no se debe a que Random Forest sea un modelo mejor en términos absolutos, sino a que las 72 variables externas ya le estaban proporcionando información sobre los nodos vecinos. Posteriormente se explicará en detalle las tres formas de combinar las variables que se usan en este trabajo, precisamente para poder separar ambos efectos y que la comparación entre modelos tenga más sentido.

Otra de las características principales del dataset es su estructura temporal. Elliptic divide las transacciones en periodos temporales o “Timesteps”, equivalentes a dos semanas, que indican el momento en que la transacción fue realizada. La peculiaridad del dataset es que a partir del timestep 43, el número de transacciones ilícitas cae de forma brusca. Weber et al. (2019) lo asociaron al cierre de un importante mercado ilegal durante ese periodo. El efecto es visible en todos los modelos sin excepción y sus predicciones se vuelven considerablemente menos fiables a partir de este punto. Es por esta estructura temporal que caracteriza no solo a las transacciones de Bitcoin, sino a cualquier transacción financiera, que en este trabajo se utiliza una partición temporal que reserva los timesteps 1 a 30 para entrenamiento, 31 a 34 para validación durante el proceso de ajuste de hiperparámetros y los timesteps 35 a 49 para test. Además, al tener el dataset variables que incluyen información de transacciones vecinas, hacer una partición aleatoria podría dar información de los datos utilizados para el test. Por ejemplo, si dos transacciones directamente conectadas

se incluyeran en particiones distintas (entrenamiento y test), los datos de entrenamiento contarían con información que no deberían de poder observar.

Como conclusión, a pesar de su utilidad como referencia común en la literatura, el dataset Elliptic tiene varias limitaciones que conviene tener en cuenta para la implementación y análisis de los modelos. Primero, al ser una muestra del grafo total de Bitcoin, algunas transacciones pueden haber perdido conexiones con otras que en la red real sí existirían, lo que puede hacer que ciertos patrones no sean detectables. Segundo, el sistema de etiquetado basado en el comportamiento de las transacciones introduce un margen de error que no es posible cuantificar con exactitud. Tercero, el hecho de que el 77% de las transacciones no tenga etiqueta limita considerablemente el rendimiento de modelos de aprendizaje supervisado. Por último, al ser las variables anónimas, no es posible saber directamente qué está midiendo cada una, lo que dificulta interpretar por qué los modelos toman unas decisiones u otras.

Tratamiento de la asimetría entre clases

Uno de los principales problemas del dataset Elliptic es que hay muchas más transacciones lícitas que ilícitas. Solo el 9.7% de las transacciones etiquetadas son ilícitas frente al 90.3% que son lícitas. Esta distribución condiciona enormemente a los modelos, debido a que si el algoritmo simplemente etiqueta toda transacción como lícita acertaría el 90% de las veces, sin detectar ninguna transacción ilegal. Por este motivo, no tiene sentido medir el rendimiento con la tasa de acierto y en su lugar se usará el F1-score sobre la clase ilícita, que penaliza directamente los errores en las transacciones que más nos importa detectar.

La literatura ha tratado este problema de formas distintas. Weber et al. (2019) optaron por dar más peso a las transacciones ilícitas durante el entrenamiento de la GCN, utilizando un peso de 0,3-0,7 para las clases lícita e ilícita respectivamente. Vassallo et al. (2021) probaron varias técnicas de muestreo y vieron que NCL-SMOTE, una combinación que primero limpia posibles outliers y después genera transacciones ilícitas sintéticas para equilibrar las clases, mejoraba la sensibilidad ligeramente respecto para la mayoría de modelos. Aun así, Alarab et al. (2020) advirtieron de que aplicar este tipo de técnicas en Elliptic puede conllevar problemas, debido a que las 72 variables agregadas se calculan a partir de los vecinos del grafo y generar ejemplos sobre ellas no tiene sentido, ya que los datos sintéticos resultantes no corresponderían a ninguna transacción real. Igualmente, De Andrade (2023) probó a eliminar parte de los datos lícitos para equilibrar las clases, obteniendo signos de sobreajuste en todos los modelos evaluados.

Teniendo esto en cuenta, en este trabajo se ajustará el peso de cada clase durante el entrenamiento, sin aplicar ninguna técnica de muestreo siguiendo el enfoque original de Weber et al. (2019). No se aplicarán técnicas de muestreo SMOTE ni NCL-SMOTE para evitar el problema que señala Alarab et al. (2020) con las variables agregadas.

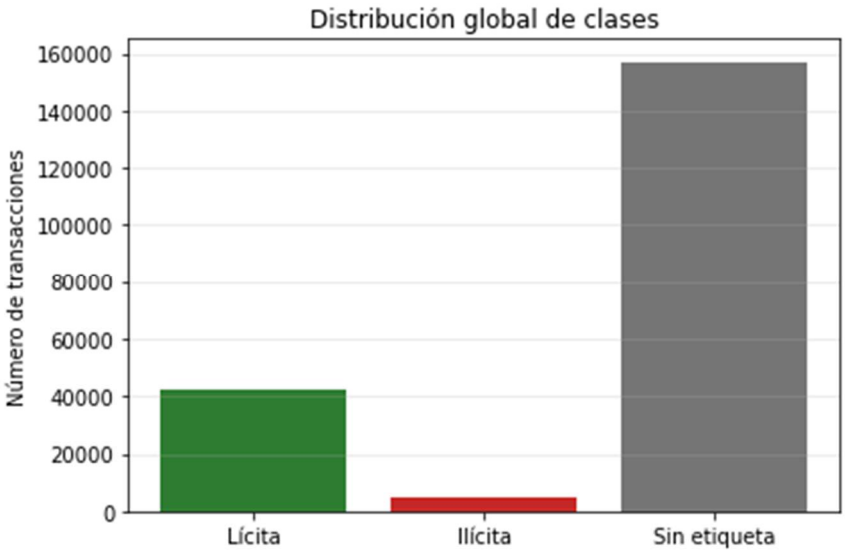

Figura 5: distribución de clases en el dataset Elliptic

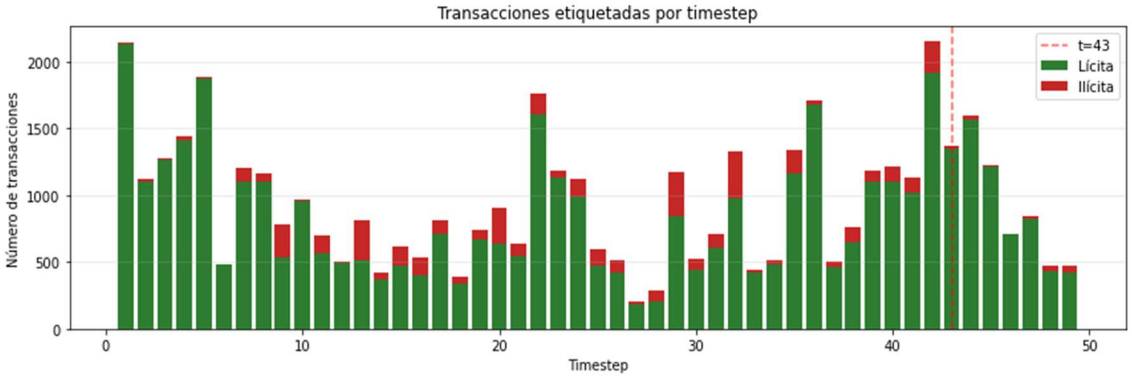

Figura 6: distribución de clases en el dataset Elliptic a lo largo del tiempo

Configuración de las clases y variables de entrada

Además de cómo tratar el desequilibrio entre clases, otra decisión clave en el diseño de los experimentos es qué variables utilizar para entrenar a los modelos. Weber et al. (2019) definieron tres configuraciones posibles sobre el dataset Elliptic, que se usarán como base para el estudio de este trabajo.

La primera configuración (LF) usa únicamente las 94 variables locales de cada transacción, la segunda (AF); incluye las 166 variables completas, incluyendo las 72 variables agregadas; y la tercera (AF+NE), incorpora además variables adicionales que representan la posición de cada transacción en el grafo de la red. Para este trabajo, solo se tendrá en cuenta esta última configuración para el modelo LightGBM, ya que Weber et al. (2019) y Vassallo et al. (2021) mostraron que añadir representaciones vectoriales de nodos (NE) no produce mejoras significativas de forma consistente, por lo que no justifican su complejidad adicional.

La distinción entre LF y AF es relevante porque las 72 variables agregadas de AF ya recogen información del entorno inmediato de cada transacción, lo que significa que un modelo entrenado con todas las variables tiene acceso a parte de la estructura del grafo sin necesidad de procesarla directamente, permitiendo analizar si esta información produce mejoras en el rendimiento del modelo o simplemente aporta ruido a los datos originales.

Métricas de evaluación

La elección de las métricas de evaluación es una decisión fundamental, debido a que el desequilibrio entre clases hace que algunas métricas habituales pierdan utilidad práctica. Como ya se ha justificado, la tasa de acierto no es de utilidad en este contexto, por este motivo la métrica principal del trabajo será el F1-score sobre la clase ilícita, en línea con la literatura científica previa.

El F1-score se define como la media armónica entre precisión y sensibilidad (De Andrade, 2023), penalizando simultáneamente los falsos positivos y los falsos negativos. La precisión mide qué proporción de las transacciones marcadas como ilícitas por el modelo realmente lo son, mientras que la sensibilidad indica qué proporción de las transacciones realmente ilícitas el modelo ha detectado correctamente. Ambas métricas se reportarán también de forma individual para tener una visión más completa de los resultados.

$$F1 = \frac{2 \times Precisión \times Sensibilidad}{Precisión + Sensibilidad}$$

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$

$$Sensibilidad = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

Junto al F1-score, se incluirá el ROC-AUC, que evalúa la capacidad del modelo para distinguir entre clases sin depender del umbral de decisión escogido. Finalmente, se reportará el F1-score por timestep, siguiendo el trabajo inicial de Weber et al. (2019), para observar el efecto posterior al timestep 43 y comparar la capacidad de cada modelo para mantener su rendimiento ante un cambio brusco en la distribución de los datos.

Modelos seleccionados

En el presente trabajo se han seleccionado siete modelos de distintas familias, con la idea de comparar enfoques con lógicas muy diferentes y entender cuál encaja mejor con la naturaleza del dataset.

Como punto de partida se incluye una regresión logística, que actuará como modelo de referencia que habrá que superar para justificar una mayor complejidad. Posteriormente se evaluarán dos modelos de ensamble, Random Forest y XGBoost, principales candidatos a obtener el mejor rendimiento. Random Forest fue el modelo con mejor F1 en el trabajo inicial de Weber et al. (2019), con 0,79 y XGBoost ha obtenido resultados similares o superado al Random Forest en algunas configuraciones para el estudio de Vassallo et al. (2021) tras un proceso de tuning bayesiano. También se incluye un perceptrón multicapa (MLP) como representante de las redes neuronales y como representante de los métodos no supervisados se utilizará Isolation Forest, replicando los experimentos de Lorenz et al. (2020).

En el mencionado estudio se concluye que los modelos no supervisados obtienen resultados por debajo de lo esperado para el dataset Elliptic, por lo que se incluirá en este trabajo para confirmar que este enfoque no es apto para el contexto presentado. Se añade también un clasificador por vecinos más cercanos (KNN), que sirve como referencia adicional basado en distancias y finalmente, se incluye un modelo LightGBM al que se añadirán representaciones vectoriales de los nodos (NE) generados mediante la librería node2vec. LightGBM es un algoritmo de gradient boosting que construye árboles de decisión de forma secuencial, destacando por su eficiencia frente a otras implementaciones similares como XGBoost (Ke et al., 2017). A este modelo se le añadirán estas nuevas características, con el objetivo de evaluar si añadir esta información sobre la estructura del grafo aporta ventaja real frente a usar únicamente las variables originales. Estas representaciones resumen la posición estructural de cada

transacción dentro del grafo, de forma que las transacciones con entornos similares en la red obtengan representaciones parecidas (Grover y Leskovec, 2016).

Implementación y configuración

Todo el desarrollo se realizará en Python, utilizando las librerías estándar para cada familia de modelos. Para el preprocesamiento se cargarán los tres ficheros originales del dataset, se codificarán las etiquetas de cada clase, las dos configuraciones de variables (LF y AF) y se dividirá el conjunto de datos en entrenamiento, validación y test siguiendo la partición temporal descrita anteriormente. Adicionalmente, se creará una configuración extra para los modelos que cuenten con optimización de hiperparámetros, utilizando los parámetros estándar de la guía de uso sin optimizar, de tal forma que se pueda comparar el rendimiento de ambas versiones y comprobar si esa complejidad extra es necesaria.

La regresión logística, el Random Forest, el KNN, MLP e Isolation Forest se implementarán con la librería scikit-learn; XGBoost con la librería oficial Xgboost y LightGBM con la librería Lightgbm; siguiendo en todo momento las guías de uso oficiales para cada modelo. Las representaciones vectoriales de los nodos del modelo LightGBM se generarán con el modelo node2vec propuesto por Grover y Leskovec (2016), siguiendo la guía de uso para Pyhton presentada en el repositorio de Github creado por Elior Cohen (Cohen, 2024).

La optimización de hiperparámetros se realizará mediante el algoritmo Tree-Structured Parzen Estimator (TPE) implementado con la librería optuna, replicando el procedimiento de Vassallo et al. (2021); con 30 iteraciones de búsqueda por modelo, buscando maximizar el F1-score. La evaluación durante el ajuste de hiperparámetros se realizará siempre sobre el conjunto de validación (timesteps 31–34), que permanecerá aislado del entrenamiento. Una vez identificada la mejor configuración tras 30 iteraciones de búsqueda, cada modelo se reentrenará sobre el conjunto combinado de entrenamiento y validación (timesteps 1–34) y se evaluará definitivamente sobre el conjunto de test (timesteps 35–49). Los modelos sin búsqueda de hiperparámetros (MLP e Isolation Forest) se entrenan directamente sobre ese mismo conjunto. En ambos casos, la evaluación final se realiza sobre el conjunto de test (timesteps 35–49), aislado durante todo el proceso de entrenamiento.

En la regresión logística se ajusta el parámetro C, que controla el grado de regularización del modelo, partiendo de la configuración por defecto de la librería scikit-learn con valores entre 0.1 y 10. El desequilibrio entre clases se gestiona

activando la opción class_weight='balanced', que ajusta automáticamente el peso de cada clase de forma inversamente proporcional a su frecuencia en los datos.

El Random Forest se iniciará con la configuración base de Weber et al. (2019) (50 árboles y 50 variables consideradas en cada división) y se modificarán mediante TPE los parámetros de profundidad máxima de cada árbol y el número mínimo de muestras por hoja. Al igual que en la regresión logística, el desequilibrio entre clases se compensa mediante class_weight='balanced'.

Para XGBoost se optimizarán 6 hiperparámetros mediante TPE incluyendo la tasa de aprendizaje, profundidad máxima y pérdida mínima para realizar una división. Scale_pos_weight, se usará para ajustar los pesos del modelo y se fijará como el ratio entre transacciones licitas e ilícitas (8.11 para los datos de entrenamiento)

El MLP mantiene la arquitectura por defecto de la librería scikit-learn con una capa oculta de 100 neuronas, función de activación relu, optimizador Adam con learning rate 0.0001 y 200 épocas de entrenamiento.

Para el modelo KNN los hiperparámetros optimizados incluyen el número de vecinos, el peso (uniforme o por distancia) y el tipo de distancia empleada (manhattan o euclidiana).

Finalmente, el modelo LightGBM combinará las características originales (166 variables) con 32 dimensiones adicionales formadas por las representaciones de los nodos generadas con node2vec sobre el grafo completo de transacciones, creando un espacio de entrada de 198 variables por transacción. Para generar el grafo sobre el que se aplicará la librería node2vec, llamaremos a la función from_pandas_edgelist, de la librería NetworkX, utilizando el archivo elliptic_txs_edgelist.csv para crearlo. Igualmente, se entrenará y optimizará con los parámetros por defecto de LightGBM, incluyendo tasa de aprendizaje, número de hojas o profundidad máxima.

Resultados

Tras describir la metodología y modelos seleccionados para este trabajo, esta sección presenta los resultados obtenidos con el fin de responder a la pregunta de investigación planteada al inicio del proyecto: hasta qué punto las técnicas de aprendizaje automático pueden contribuir a la detección de actividades ilícitas en Bitcoin.

La Figura 7 presenta el rendimiento de los catorce modelos evaluados sobre el conjunto de test (timesteps 35–49), ordenados por F1-score sobre la clase ilícita para su versión con optimización de hiperparámetros. Los resultados para el F1-

score de la izquierda reflejan el rendimiento obtenido con las configuraciones estándar sin optimización de hiperparámetros, mientras que las cifras de la derecha muestran los resultados después de utilizar esta búsqueda. Además, la Figura 8 complementa esta información con la evolución del F1 por cada timestep. Los resultados se sitúan en un rango esperable según la literatura, siendo el mejor F1 alcanzado (0,818) superior a los valores reportados por Weber et al. (2019) y similares a los presentados por Vassallo et al. (2021), que se situaban en torno a 0,79-0,83. Esta mejora se observa de forma consistente en los modelos de gradient boosting, especialmente tras el proceso de optimización bayesiana de hiperparámetros mediante TPE; y sorprendentemente, en la versión de Random Forest sin optimización.

El mejor modelo del estudio ha sido LightGBM con todas las características (AF), que supera ligeramente a su variante con node embeddings (LGBM+NE) y a XGBoost. Siguiendo los resultados de la literatura previa sobre Elliptic, se puede observar que los cuatro mejores modelos incluyen algoritmos de gradient boosting (LightGBM 0,818 – XGBoost 0,811) y Random Forest (0,815). Igualmente, LightGBM alcanza una precisión de 0,93, lo que significa que el 93% de las transacciones que clasifica como ilícitas lo son realmente y solo el 7% serían falsas alarmas. Este es el valor más alto del estudio, lo que lo hace especialmente útil en un escenario real, donde cada alerta del sistema puede conllevar una revisión manual costosa. XGBoost, por su parte, confirma la solidez mostrada en la literatura previa (F1 de 0,811 con AF), mientras que Random Forest se sitúa como el tercer mejor modelo con un F1 de 0,815 para su variante con todas las variables y con los parámetros estándar de la guía de scikit learn sin optimizar.

Cabe recalcar que Random Forest es el único modelo cuyo mejor resultado se obtiene sobre la configuración estándar, además de presentar la mayor diferencia absoluta de rendimiento en términos de F1-score (0,815 vs. 0,733). La explicación más probable de este fenómeno es que la elección de hiperparámetros para el modelo con optimización fuera subóptima. Esto se debe a que en esta configuración existen dos parámetros que no pasan por el proceso de optimización (n_estimators y max_features), siguiendo el ejemplo de Weber et al. (2019) donde aplica el valor 50 para ambas características. De este modo, esta restricción es la causa más probable del limitado rendimiento del modelo y explica que sea el único algoritmo que mejora sus resultados en su versión simplificada. Igualmente, los resultados muestran de forma clara que el proceso de optimización mejora el rendimiento de los modelos, sin tener en cuenta está única excepción.

Otro resultado especialmente relevante es el rendimiento de los modelos de gradient boosting frente a los enfoques basados en redes de grafo. Mientras que

Weber et al. (2019) reportaron un F1 de 0,63 para su GCN (0,72 para su variante con mejor resultado), en el presente estudio todas las versiones de LightGBM y XGBoost obtienen un resultado superior sobre el mismo dataset. Esto refuerza la conclusión de que los enfoques tradicionales no solo pueden igualar a los modelos basados en análisis de grafos, sino que superan claramente a estos algoritmos a la hora de tratar la estructura del dataset Elliptic.

Igualmente, cabe recalcar el papel de los node embeddings generados con node2vec. La comparación entre LightGBM (AF) y LightGBM+NE muestra que añadir estas 32 dimensiones adicionales del grafo no aporta mejora real (F1 de 0,818 frente a 0,817), pero tampoco empeora el rendimiento ni introduce ruido perjudicial al modelo. Este resultado muestra que una vez que el modelo ya dispone de las 72 variables agregadas que resumen el entorno de cada transacción, la información adicional capturada por los embeddings no añade valor añadido de forma considerable a los algoritmos. En la práctica esto implica que en datasets donde ya existan variables que codifiquen la relación de los nodos, el coste computacional de generar estas representaciones no se justifica por la escasa ganancia obtenida.

En el tramo intermedio aparece el MLP con variables locales (F1 de 0,686) y con todas las variables (F1 de 0,546), ofreciendo un rendimiento dentro del rango esperado según el estudio de Weber et al. (2019), entre 0,62 y 0,69. El KNN, alcanza un F1 aceptable (0,695 con LF), lo que sugiere que las transacciones comparten características lo suficientemente similares entre sí como para ser identificadas por su proximidad, superando igualmente a la GCN presentada por Weber et al. (2019).

En el extremo inferior, la regresión logística cumple su rol de referencia con un F1 bajo (0,326) pero una sensibilidad muy alta (0,859). El Isolation Forest, con F1 prácticamente nulo, confirma lo anticipado por Lorenz et al. (2020), mostrando como los modelos no supervisados no son adecuados para el contexto del dataset Elliptic.

Otro patrón interesante del estudio es que los tres mejores modelos (LightGBM, XGBoost y Random Forest), obtienen los mejores resultados en su versión con todas las características (AF). Este comportamiento muestra que las 72 variables agregadas sí aportan información relevante para la detección de transacciones ilegales y no introducen ruido perjudicial en estos algoritmos.

Finalmente, el análisis del F1 por timestep (Figura 8) muestra uno de los resultados más importantes del estudio. A partir del timestep 43, todos los modelos sin excepción sufren una caída drástica del rendimiento, pasando de F1 entre 0,7 y 0,95 antes del cierre a valores inferiores a 0,2 después. Este comportamiento es prácticamente idéntico en todos los modelos, lo que confirma

que la limitación está ligada al cambio de distribución de los datos y no es atribuible a la arquitectura propia de los algoritmos.

Modelo	F1 ilícita	Precisión	Sensibilidad	ROC-AUC
LGBM (AF)	0,786 - 0,818	0,926	0,732	0,935
LGBM+NE (AF)	0,789 - 0,817	0,929	0,729	0,924
XGBoost (AF)	0,795 - 0,811	0,921	0,724	0,934
LGBM (LF)	0,748 - 0,782	0,864	0,714	0,914
XGBoost (LF)	0,766 - 0,776	0,852	0,712	0,897
Random Forest (LF)	0,794 - 0,744	0,750	0,739	0,898
Random Forest (AF)	0,815 - 0,733	0,734	0,731	0,919
KNN (LF)	0,587 - 0,695	0,671	0,722	0,909
MLP (LF)	0,686	0,780	0,612	0,903
MLP (AF)	0,546	0.770	0.423	0.896
KNN (AF)	0,569 - 0,597	0,543	0,664	0,861
Regresión Logística (AF)	0,294 - 0,326	0.201	0.859	0.887
Regresión Logística (LF)	0,221 - 0,243	0.140	0.898	0.885
Isolation Forest (AF)	0,0007	0.0006	0.0009	0.159

Figura 7: resultados obtenidos por modelo y configuración

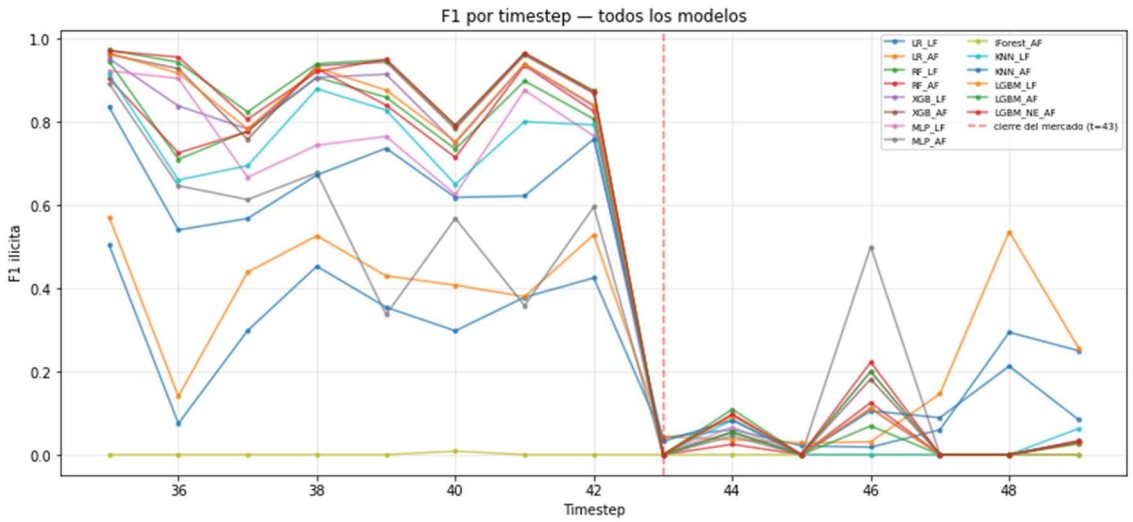

Figura 8: F1-score por timestep para todos los modelos evaluados

Implicaciones prácticas y aplicabilidad

A partir de los resultados obtenidos en la sección anterior, es posible extraer varias conclusiones sobre la viabilidad de estos modelos en un contexto real de supervisión financiera, ya sea por parte de organismos reguladores como la FATF o las propias plataformas de criptomonedas obligadas a implementar controles de prevención del blanqueo de capitales.

El primer factor que condiciona cualquier decisión de un modelo es la escala. El dataset Elliptic, con sus 203.769 transacciones, representa una fracción mínima de la actividad real de la red Bitcoin, que acumula históricamente más de 1.300 millones de transacciones (Blockchain.com, s.f.). El propio Weber et al. (2019) ya señalaba esta limitación al indicar en el momento de la publicación de su estudio que la red completa contaba con aproximadamente 438 millones de nodos y 1.100 millones de aristas. El tiempo de cómputo requerido para entrenar los modelos más complejos sobre el dataset de Elliptic ya resulta considerable en este contexto reducido; por lo que extrapolarlo a la red completa plantea un problema de escalabilidad que muchos organismos difícilmente podrían resolver sin infraestructura especializada.

En este contexto, los modelos de gradient boosting (LightGBM y XGBoost) y Random Forest se presentan como los candidatos más adecuados para una aplicación práctica real. LightGBM ofrece el mejor rendimiento del estudio (F1 de 0,818), combinándolo con tiempos de entrenamiento relativamente eficientes. Random Forest, por su parte, ofrece un resultado prácticamente idéntico en

términos de rendimiento (0,815), pero ofrece ventajas difíciles de sustituir para su uso común. La principal ventaja de Random Forest es su mayor simplicidad y facilidad para entender la importancia de las distintas variables usadas por el modelo, lo que permite explicar de forma más objetiva por qué una transacción ha sido marcada como ilícita. En el caso del dataset Elliptic las variables son anónimas, por lo que este análisis no tiene tanto sentido, pero una plataforma con datos y variables propias podría beneficiarse en gran medida de esta característica del modelo. De esta forma, analistas humanos sin formación previa en métodos de aprendizaje automático podrían entender qué variables son más importantes a la hora de analizar una posible transacción ilegal y dedicarle más recursos a su solución.

Igualmente, Random Forest obtiene este resultado sin aplicar una optimización de hiperparámetros, utilizando solo los valores por defecto de scikit-learn. Esto lo convierte en la práctica en la mejor opción con diferencia en términos de rendimiento frente a complejidad o coste computacional. Para organismos o instituciones que no dispongan de la capacidad de cómputo necesaria para realizar una búsqueda de hiperparámetros, utilizar Random Forest con su configuración estándar ofrece un resultado que iguala a los mejores modelos sin requerir una inversión técnica significativa. Esto lo convierte en una opción especialmente atractiva para organizaciones más pequeñas, democratizando el acceso a este tipo de herramientas dentro del ámbito de AML.

Otro punto a resaltar es el comportamiento de todos los modelos tras el timestep 43. La bajada general del rendimiento muestra la necesidad de tener especial precaución con el uso de algoritmos de aprendizaje automático en este contexto. Esta caída demuestra que estos algoritmos podrían ser ineficientes ante métodos nuevos de blanqueo de capitales o cambios en el comportamiento de los participantes de la red. En un entorno real, donde las técnicas de blanqueo evolucionan continuamente, esto implica la necesidad de reentrenamiento periódico y supervisión humana constante, lo que favorece de nuevo a los modelos más simples e interpretables cuya lógica puede ser validada por expertos en el área.

Conclusión

En definitiva, los resultados de este trabajo apuntan a que la detección automática de blanqueo de capitales en Bitcoin es técnicamente viable con herramientas ampliamente disponibles y que la elección del modelo para un despliegue real debería priorizar la interpretabilidad y la escalabilidad frente a una mejora de rendimiento marginal de los resultados obtenidos (en este caso el F1-

score). Añadir complejidad cuando la ganancia de rendimiento no es lo suficientemente alta no parece justificado en un área donde la toma de decisiones está sujeta a requisitos regulatorios.

En base a las conclusiones anteriores, se recomienda el uso de Random Forest como primera opción por su simplicidad, interpretabilidad y excelente rendimiento incluso sin optimización de hiperparámetros. Como segunda opción se proponen los algoritmos de gradient boosting (especialmente LightGBM), que ofrecen el mejor rendimiento del estudio a costa de una mayor complejidad y la necesidad de un proceso de optimización, resultando especialmente adecuados para entidades con mayor capacidad técnica y computacional.

Bibliografía

Ajagbe, S. A., Majola, S., & Mudali, P. (2025). Comparative analysis of machine learning algorithms for money laundering detection. *Discover Artificial Intelligence*, *5*(1), 144.

Alarab, I., Prakoonwit, S., & Nacer, M. I. (2020, June). Comparative analysis using supervised learning methods for anti-money laundering in bitcoin. In *Proceedings of the 2020 5th international conference on machine learning technologies* (pp. 11-17).

Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., & Capkun, S. (2013, April). Evaluating user privacy in bitcoin. In *International conference on financial cryptography and data security* (pp. 34-51). Berlin, Heidelberg: Springer Berlin Heidelberg.

Asiri, A., & Somasundaram, K. (2025). Graph convolution network for fraud detection in bitcoin transactions. *Scientific Reports*, *15*(1), 11076.

Blockchain.com. (s.f.). *Total number of transactions*. Recuperado el 10 de junio de 2026, de https://www.blockchain.com/explorer/charts/n-transactions-total

Caringella, M., Violante, F., De Lucci, F., Galantucci, S., & Costantini, M. (2024). Bach: A tool for analyzing blockchain transactions using address clustering heuristics. *Information*, *15*(10), 589.

Cohen, E. (2024). *node2vec* (versión 0.5.0) [Software]. GitHub. https://github.com/eliorc/node2vec

De Andrade, V. M. (2023). Money Laundering and Bitcoins: Detecting Fraudulent Transactions Using Machine Learning.

Deprez, B., Vanderschueren, T., Baesens, B., Verdonck, T., & Verbeke, W. (2026). Network analytics for anti-money laundering—a systematic literature review and experimental evaluation. *INFORMS Journal on Data Science*, *5*(2), 119-154.

Europol. (2026). *Internet Organised Crime Threat Assessment (IOCTA) 2026: The evolving threat landscape — How encryption, proxies and artificial intelligence are expanding cybercrime*. Publications Office of the European Union, Luxembourg. https://www.europol.europa.eu/publication-events/main-reports/iocta-2026-evolving-threat-landscape#downloads

Financial Action Task Force. (s.f.). *Frequently asked questions*. Recuperado el 9 de junio de 2026, de https://www.fatf-gafi.org/en/pages/frequently-asked-questions.html

Financial Action Task Force. (2019). *Guidance for a risk-based approach to virtual assets and virtual asset service providers*, FATF, Paris, https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-rba-virtual-assets.html

Financial Action Task Force. (2020). *Money Laundering and Terrorist Financing Red Flag Indicators Associated with Virtual Assets*, FATF, Paris, France, https://www.fatf-gafi.org/en/publications/Methodsandtrends/Virtual-assets-red-flag-indicators.html

Financial Action Task Force. (2021). *Opportunities and challenges of new technologies for AML/CFT*, FATF, Paris, France https://www.fatf-gafi.org/content/dam/fatf-gafi/guidance/Opportunities-Challenges-of-New-Technologies-for-AML-CFT.pdf.coredownload.inline.pdf

Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864).

Haber, S., & Stornetta, W. S. (1990, August). How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography* (pp. 437-455). Berlin, Heidelberg: Springer Berlin Heidelberg.

Islam, M. Z., Islam, M. S., Reza, S. A., Bhowmik, P. K., Bishnu, K. K., Rahman, M. S., ... & Pant, L. (2025). Machine learning-based detection and analysis of suspicious activities in Bitcoin wallet transactions in the USA. *arXiv preprint arXiv:2504.03092*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.

Khan, T. A., Tulsi, J., Alam, M., Kadir, K., Ali, K. M., & Mazliham, M. S. (2025). Analysis and visualization of fraud detection patterns through data mining and classification using MLP and hybrid deep learning model. *Egyptian Informatics Journal*, *32*, 100829.

Lin, Z., Luo, Q., Wu, D., Shen, J., Li, L., Nong, X., & Qin, Z. (2026). Detecting illicit transactions in bitcoin: a wavelet-temporal graph transformer approach for anti-money laundering. *Scientific Reports*, *16*(1), 1548.

Lorenz, J., Silva, M. I., Aparício, D., Ascensão, J. T., & Bizarro, P. (2020, October). Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity. In *Proceedings of the first ACM international conference on AI in finance* (pp. 1-8).

Motamed, A. P., & Bahrak, B. (2019). Quantitative analysis of cryptocurrencies transaction graph. *Applied Network Science*, *4*(1), 131.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Ngai, E. W., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, *50*(3), 559 569.

Pettersson Ruiz, E., & Angelis, J. (2022). Combating money laundering with machine learning–applicability of supervised-learning algorithms at cryptocurrency exchanges. *Journal of Money Laundering Control*, *25*(4), 766-778.

Reid, F., & Harrigan, M. (2012). An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks* (pp. 197-223). New York, NY: Springer New York.

TRM Labs. (2026). *2026 crypto crime report*. https://www.trmlabs.com/reports-and-whitepapers/2026-crypto-crime-report

Vassallo, D., Vella, V., & Ellul, J. (2021). Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies. *SN Computer Science*, *2*(3), 143.

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, *1*, 57-81.

Declaración de Uso de Herramientas de Inteligencia Artificial Generativa en Trabajos Fin de Grado

Por la presente, yo, Rodrigo de Gracia, estudiante de Administración y Dirección de Empresas y Análisis de Negocio de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado "Análisis comparativo de técnicas de aprendizaje automático para la detección de transacciones ilícitas en Bitcoin", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación:

1. **Crítico:** Para encontrar contra-argumentos a una tesis específica que pretendo defender.
2. **Referencias:** Usado conjuntamente con otras herramientas, como Science, para identificar referencias preliminares que luego he contrastado y validado.
3. **Metodólogo:** Para descubrir métodos aplicables a problemas específicos de investigación.
4. **Interpretador de código:** Para realizar análisis de datos preliminares.
5. **Corrector de estilo literario y de lenguaje:** Para mejorar la calidad lingüística y estilística del texto.
6. **Sintetizador y divulgador de libros complicados:** Para resumir y comprender literatura compleja.
7. **Generador de problemas de ejemplo:** Para ilustrar conceptos y técnicas.
8. **Revisor:** Para recibir sugerencias sobre cómo mejorar y perfeccionar el trabajo con diferentes niveles de exigencia.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las

referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: 10/06/2026

Firma: Rodrigo de Gracia de Dios

Anexo

Código

Regresión logística

Función de optimización para obtener los mejores hiperparámetros

```
def objetivo(trial):
  c = trial.suggest_float('C', 0.1, 10)
  modelo = LogisticRegression(C=c, class_weight='balanced',
                solver='lbfgs', max_iter=500,
                random_state=semilla)
  modelo.fit(x_ent, y_entrenamiento)
  predicciones = modelo.predict(x_val)
  return f1_score(y_validacion, predicciones, pos_label=1, zero_division=0)
estudio = optuna.create_study(direction='maximize',
                sampler=optuna.samplers.TPESampler(seed=semilla))
estudio.optimize(objetivo, n_trials=30)
```

Configuración estandar sin optimización

```
modelo_lf = LogisticRegression(C=1, class_weight='balanced', max_iter=100,
random_state=semilla)
modelo_lf.fit(x_ent_val_lf, y_ent_val)
predicciones_lf = modelo_lf.predict(x_test_lf)
```

```
f1_lf = f1_score(y_test, predicciones_lf, pos_label=1)
```

Random Forest

Función de optimización para obtener los mejores hiperparámetros

```
def objectivo(trial):
  modelo = RandomForestClassifier(
    n_estimators=50,
    max_features=50,
    max_depth=trial.suggest_int('max_depth', 3, 30),
    min_samples_leaf=trial.suggest_int('min_samples_leaf', 1, 20),
    class_weight='balanced',
    n_jobs=-1,
    random_state=semilla,
)
  modelo.fit(x_ent, y_ent)
  y_pred = modelo.predict(x_val)
  return f1_score(y_val, y_pred, pos_label=1, zero_division=0)
optimizacion = optuna.create_study(direction='maximize',
                sampler=optuna.samplers.TPESampler(seed=semilla))
optimizacion.optimize(objectivo, n_trials=30)
```

Configuración estandar sin optimización

```
modelo_lf = RandomForestClassifier(
  n_estimators=100,
  max_features="sqrt",
  max_depth=None,
  min_samples_leaf=1,
  class_weight='balanced',
  n_jobs=-1,
```

```
        random_state=semilla,
    )
```

XGBoost

Función de optimización para obtener los mejores hiperparámetros

```
def objetivo(trial):
  modelo = XGBClassifier(
    learning_rate=trial.suggest_float('learning_rate', 0, 1),
    min_split_loss=trial.suggest_float('min_split_loss', 0, 5),
    max_depth=trial.suggest_int('max_depth', 3, 10),
    min_child_weight=trial.suggest_int('min_child_weight', 1, 10),
    max_delta_step=trial.suggest_int('max_delta_step', 0, 10),
    subsample=trial.suggest_float('subsample', 0.5, 1.0),
    scale_pos_weight=ratio,
    eval_metric='logloss',
    random_state=semilla,
    n_jobs=-1,
  )
  modelo.fit(x_ent, y_ent)
  predicciones = modelo.predict(x_val)
  return f1_score(y_val, predicciones, pos_label=1, zero_division=0)
optimizacion = optuna.create_study(direction='maximize',
                sampler=optuna.samplers.TPESampler(seed=semilla))
optimizacion.optimize(objetivo, n_trials=30)
```

Configuración estandar sin optimización

```
    modelo_af = XGBClassifier(
      learning_rate=0.3,
      min_split_loss=0,
      max_depth=6,
```

```
    min_child_weight=1,
    max_delta_step=0,
    subsample=1,
    scale_pos_weight=ratio,
    eval_metric='logloss',
    random_state=semilla,
    n_jobs=-1,
  )
```

MLP

Configuración estandar sin optimización

```
  modelo_lf = MLPClassifier(
    hidden_layer_sizes=(100),
    activation='relu',
    solver='adam',
    learning_rate_init=0.0001,
    max_iter=200,
    random_state=semilla,
  )
```

Isolation Forest

Configuración estandar sin optimización

```
modelo_af = IsolationForest(
  n_estimators=100,
  max_samples='auto',
  contamination='auto',
  random_state=semilla,
  n_jobs=-1,
)
```

KNN

Función de optimización para obtener los mejores hiperparámetros

```
def objetivo(trial):
  modelo = KNeighborsClassifier(
    n_neighbors=trial.suggest_int('n_neighbors', 1, 200),
    weights=trial.suggest_categorical('weights', ['uniform', 'distance']),
    p=trial.suggest_int('p', 1, 2),
    n_jobs=-1
  )
  modelo.fit(x_ent, y_entrenamiento)
  return f1_score(y_validacion, modelo.predict(x_val), pos_label=1,
zero_division=0)


estudio = optuna.create_study(direction='maximize',
                  sampler=optuna.samplers.TPESampler(seed=semilla))
estudio.optimize(objetivo, n_trials=30)
```

Configuración estandar sin optimización

```
modelo_af = KNeighborsClassifier(n_neighbors=5, weights='uniform', p=2, n_jobs=-1)
modelo_af.fit(x_ent_val_af, y_ent_val)
pred_af = modelo_af.predict(x_test_af)
f1_af = f1_score(y_test, pred_af, pos_label=1)
```

LightGBM

Creación de Node Embeddings con la librería node2vec

```
 aristas = pd.read_csv('elliptic_txs_edgelist.csv')
graph = nx.from_pandas_edgelist(aristas, source='txId1', target='txId2')
```

```
node2vec = Node2Vec(
  graph,
  dimensions=32,
  walk_length=5,
  num_walks=5,
  workers=4,
  temp_folder='./datos_finales',
)
model = node2vec.fit(window=10, min_count=1, batch_words=4)
model.wv.save_word2vec_format(EMBEDDING_FILENAME)
model.save(EMBEDDING_MODEL_FILENAME)
```

Función de optimización para obtener los mejores hiperparámetros

```
def objetivo(trial):
  modelo = LGBMClassifier(
    learning_rate=trial.suggest_float('learning_rate', 0.01, 0.3),
    num_leaves=trial.suggest_int('num_leaves', 15, 100),
    max_depth=trial.suggest_int('max_depth', 3, 10),
    n_estimators=trial.suggest_int('n_estimators', 50, 300),
    is_unbalance=True,
    random_state=semilla,
    n_jobs=-1,
  )
  modelo.fit(x_ent, y_ent)
  predicciones = modelo.predict(x_val)
  return f1_score(y_val, predicciones, pos_label=1, zero_division=0)
```

Configuración estandar sin optimización

```
modelo_lf = LGBMClassifier(
  learning_rate=0.1,
  num_leaves=31,
```

```
    max_depth=-1,
    n_estimators=100,
    is_unbalance=True,
    random_state=semilla,
    n_jobs=-1,
)
```