Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

2022-2023

Trabajo Fin de Grado

Desarrollo de sistemas de detección de fraude con técnicas de inteligencia artificial

---

Luis Docampo Mateos Pardo

Tutor

Ruben Majadas Sanz

Madrid

RESUMEN

Este trabajo de investigación se centra en la aplicación de técnicas de aprendizaje automático para la detección de fraudes en cuentas de Ethereum y transacciones de pago utilizando un conjunto de datos sintético de J.P. Morgan. El problema de detección de fraudes plantea desafíos significantes en la industria financiera, requiriendo la integración de inteligencia artificial, aprendizaje automático y análisis de datos. Este estudio tiene como objetivo abordar estos desafíos y desarrollar mecanismos efectivos de detección de fraudes. Existen dificultades intrínsecas en la detección de fraudes, como el cambio de concepto o el problema de los falsos positivos y falsos negativos, que han sido estudiados y considerados para el desarrollo del proyecto.

Para abordar este problema, se han examinado estudios previos sobre detección de fraudes, comparando los resultados obtenidos y las circunstancias en las que se lograron. Además, se han probado diferentes tipos de algoritmos de aprendizaje automático para la creación de los modelos finales, permitiendo comparar su rendimiento entre sí.

Además, se presenta el diseño de la implementación de un contrato inteligente que ejecuta el modelo creado para detectar cuentas fraudulentas de Ethereum.

ABSTRACT

This research work focuses on the application of machine learning techniques for fraud detection in Ethereum accounts and payment transactions using a synthetic J.P. Morgan data set. The problem of fraud detection poses significant challenges in the financial industry, requiring the integration of artificial intelligence, machine learning, and data analysis. This study aims to address these challenges and develop effective fraud detection mechanisms. There are intrinsic difficulties to fraud detection as Concept Shift or the problem with false positives and false negatives, these difficulties have been studied and considered for the development of the project.

To approach this problem, previous studies about fraud detection have been examined, comparing the results obtained and under which circumstances.

Moreover, different kinds of machine learning algorithms have been tested for the creation of the final models, being able to compare their performance with each other.

Furthermore, the design of the implementation of a smart contract that executes the model created to detect fraudulent Ethereum accounts is laid out.

Dedicatoria

Querría aprovechar este espacio para agradecer a mi familia por la las oportunidades que me han dado durante toda mi vida y darme la motivación para seguir adelante.

También agradecer a mis amigos por los buenos momentos y cualquier tipo de ayuda que me han dado.

Y por último agradecer a cualquier persona que haya dedicado su tiempo para que yo pueda aprender , especialmente a mis profesores tanto de la universidad como del colegio, y a mi tutor, Rubén Majadas Sanz, que con su ayuda he podido realizar este proyecto.

Índice

Índice de figuras

Índice de tablas

1. Introducción

In this initial section a description of the problem selected for the project and the motivation to solve it is going to be presented. En esta parte inicial del documento se presenta una descripción del problema y la motivación de trabajar en él para intentar resolverlo.

1.1. Descripción del problema

En la nueva era de la tecnología se presentan oportunidades para resolver problemas que se llevan mucho tiempo intentando resolver, el fraude bancario lleva siendo un problema desde las primeras transacciones y debido a la naturaleza de este, detectarlo puede ser no útil del todo ya que el daño puede que sea irresoluble, por ello es importante poder detectarlo de manera rápida y fiable para así poder anular cualquier transferencia de fondos para que no haya perjudicados. El término fraude es muy amplio, en este trabajo nos centramos en dos tipos. Uno es la detección de cuentas fraudulentas, específicamente las de Ethereum. Esto puede ayudar a un usuario que vaya a enviar dinero a una de estas cuentas consideradas como fraudulentas a no enviarlo para así no perder los fondos, ya que si por ejemplo estuviera comprando un producto o servicio esta persona no lo recibiría. Esto es comparable con detectar páginas webs que ofrecen servicios o productos que luego no dan, por lo que advertir a una persona antes de que envíe la transacción aporta tranquilidad al usuario y le ayuda a mantener su dinero, que según el caso puede ser muy importante para él. El segundo tipo de fraude que se intentará solucionar es la detección de transacciones fraudulentas. En el momento que se va a realizar una transacción tenemos información sobre la cuenta que envía y la cuenta que recibe los fondos. Aparte también se puede ir guardando información histórica de las actividades de cada

cuenta. Si este sistema creado puede clasificar la transacción como fraudulenta antes de que se envíen los fondos, se podría parar la transacción evitando la posible pérdida del dinero.

Según las características de las actividades fraudulentas, poder ayudar a detectarlo es un trabajo con especial relevancia, en este caso consideramos el fraude que puede suceder tanto en cuentas bancarias como en direcciones de Ethereum, una red de Blockchain. Aún siendo tecnologías distintas y la metodología para pasar fondos de una cuenta/dirección a otra sean distintas, la manera de detectar fraude es similar, ya que basicamente se fundamenta en pasar una cantidad de un activo financiero de una cuenta a otra.

1.2. Planteamiento del problema

La detección de fraude es un desafío constante en diversos campos, desde transacciones financieras hasta detección de fraudes en seguros y telecomunicaciones. En los últimos años, el uso de técnicas de aprendizaje automático ha demostrado ser muy efectivo en la detección temprana y precisa de actividades fraudulentas.

Los estafadores o defraudadores, es decir, las personas que cometen actividades fraudulentas, suelen estar alerta sobre los planes y métodos de cada industria. También hay que considerar que cualquier fraude puede provenir tanto externamente como de manera interna .El defraudador puede ser una tercera persona externa, o un conjunto de personas externas. Además, el defraudador puede cometer fraude ya sea en forma de un cliente potencial/existente (consumidor) o un proveedor existente.

Tanto el sector financiero como el bancario necesitan tener un sistema de transacciones seguro tanto para ellos como para sus clientes. Por esa razón

la innovación en este sector se puede comprobar con los sistemas que se han construido y tecnologías que implementan. Las interacciones diarias de personas con estos sistemas que les ayudan con sus tareas permiten la generación de una gran cantidad de datos, estos datos pueden ayudar a resolver otros problemas o mejorar soluciones existentes. Con la ayuda de tecnologías como aprendizaje automático o minado de datos se pueden encontrar patrones ocultos a primera vista.

La tecnología Blockchain trae nuevas oportunidades pero al mismo tiempo trae nuevos retos. Respecto al fraude en redes Blockchain uno de los problemas que conlleva es la anonimicidad de las direcciones y las transacciones, pero al mismo tiempo se pueden implementar smart contracts con el modelo generado por el desarrollador, pudiendo protegerse cada usuario de la manera que quiera y pueda.

1.3. Motivación

La principal motivación para llevar a cabo este trabajo es mi gran interés en el sector financiero y la ciencia de datos. En el sector de la banca y las finanzas, hay un gran énfasis en tecnologías como el aprendizaje automático y la inteligencia artificial. Sin embargo, la aplicación de estas tecnologías puede ser menos que óptima, o en ocasiones, las soluciones, aunque valiosas, pueden ser prohibitivamente complejas y costosas para un proyecto de esta naturaleza. En consecuencia, he decidido concentrar este trabajo en un área que creo que posee una utilidad significativa y me permite adquirir conocimientos que probablemente serán beneficiosos en el futuro.

Además, es importante reconocer que el fraude representa un desafío dentro del sector financiero, que necesita contramedidas innovadoras y eficientes. La

participación en la detección de fraudes contribuye a la seguridad de los sistemas financieros y protege a individuos y entidades de pérdidas financieras.

Adicionalmente, el rápido avance hacia la digitalización y el alcance mundial de los servicios financieros han creado un escenario bancario estrechamente vinculado, que es presa fácil de sofisticadas estrategias de fraude que no dejan de cambiar. Esto realmente resalta cuán importante es idear formas sólidas y flexibles para detectar fraudes. Además, un efecto positivo de este proyecto es que podría marcar una diferencia en la sociedad al hacer que los sistemas financieros sean más resistentes y capaces de mantenerse al día en un mundo que es más en línea y global que nunca.

Por otra parte, el dolor y la impotencia que alguien sufre cuando es estafado es razón suficiente para tratar de abordar este problema. No solo el dolor que alguien sufre, también la satisfacción de saber que has evitado que alguien o alguna empresa sea víctima de un fraude también es una gran motivación para tratar de detectar fraudes.

2. Estado de la cuestión

En este segundo apartado se procede a dar una explicación del área y las tecnologías usadas en el proyecto.

2.1. Área de la aplicación

En esta sección se explica el área a la que se aplica el proyecto. Principalmente el proyecto está basado en las áreas de las ciencias de la información conocidas como minado de datos (*Data mining*) y la inteligencia artificial, también se entra un poco en la tecnología blockchain debido a la relación con el trabajo.

2.1.1. Inteligencia Artificial

La inteligencia artificial se refiere al campo de estudio y desarrollo de sistemas informáticos capaces de realizar tareas que normalmente requerirían de la inteligencia humana. El objetivo principal de la inteligencia artificial es proporcionar a los ordenadores la capacidad de realizar tareas de manera autónoma y tomar decisiones basadas en el conocimiento que se les proporciona. Los avances en esta área son actualmente de especial relevancia, superando las expectativas de muchas personas ya que se han adelantado a lo esperado y el progreso de esta tecnología es exponencial. Una de las principales razones por el interés general en este área se debe a la multidisciplinaridad de sus aplicaciones, cualquier sector, ya sea en el ámbito profesional, académico o personal se verá afectado por el desarrollo de nuevas herramientas creadas con inteligencia artificial.

Para el desarrollo de la inteligencia artificial se necesita la aplicación de distintos campos de estudio de manera conjunta, como puede ser optimización

matemática, estadística, genética, biología, filosofía o neurociencia como ejemplos.

En la actualidad hay varios subcampos de la inteligencia artificial que se han desarrollado de manera notoria en los últimos años, algunos ejemplos son los siguientes:

- **Procesamiento del lenguaje natural (NLP)**: Con la introducción de los modelos grandes de lenguaje, o como se conocen comúmnmente en inglés, Large Languaje Models (LLM) se ha conseguido que sistemas comprendan texto y que también lo generen. Los modelos de generación de lenguaje, como GPT-3 (Generative Pre-trained Transformer 3), han demostrado la capacidad de generar texto coherente y de alta calidad [1]. Estos avances en NLP han impulsado aplicaciones como chatbots inteligentes, asistentes virtuales y análisis de sentimientos en redes sociales. También se ha avanzado en la tecnología text-to-image o image-to-text, que proporcionan a los ordenadores la capacidad de crear una imagen a través de una entrada de texto, o crear una descripción de una imagen cuando se le pasa como entrada un archivo de una imagen .

- **Detección de anomalías**: La detección de anomalías es uno de los temas candentes en inteligencia artificial, especialmente en dominios donde identificar lo que no encaja puede ser extremadamente valioso, como finanzas, atención médica o ciberseguridad. La esencia de la detección de anomalías es encontrar puntos de datos que son diferentes de lo habitual. Estos pueden ser desde un fraude con tarjeta de crédito hasta una irregularidad en los signos vitales de un paciente. Hay una gran variedad de técnicas que se han utilizado para esto, desde los métodos estadísticos tradicionales hasta algoritmos de aprendizaje automático más sofisticados.

Una de las principales herramientas para detectar anomalías es el uso de redes neuronales profundas [2].

2.1.2. Aprendizaje Automático

El aprendizaje automático, también conocido como *machine learning* en inglés, es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender y mejorar su rendimiento en una determinada tarea a través de la experiencia y sin ser programadas en que acciones deben tomar explícitamente. A diferencia de la inteligencia artificial, que busca dotar a las computadoras de habilidades y capacidades similares a las humanas, el aprendizaje automático se enfoca en desarrollar algoritmos que puedan analizar datos, extraer patrones, tomar decisiones o realizar predicciones basadas en esos patrones. En lugar de programar reglas específicas, el aprendizaje automático permite a las máquinas aprender a partir de los datos y adaptarse a medida que reciben más información, facilitando en algunos casos la resolución de problemas muy complejos que a lo mejor no podría ser realizado por una persona [3].

El estado del arte del aprendizaje automático ha experimentado avances significativos en los últimos años, impulsados por varios factores:

- **Avances en hardware y recursos computacionales**: El aumento en la potencia de computación y la disponibilidad de recursos como unidades de procesamiento gráfico (GPU) han facilitado el entrenamiento y la mejora en la ejecución de modelos de aprendizaje automático. Esto ha permitido realizar experimentos más complejos y procesar grandes cantidades de datos en tiempos más cortos. Gracias a servidores en la nube cualquier usuario con el capital necesario para alquilar dichos servidores

de empresas como Microsoft, Google o Amazon pueden usarlos para entrenar modelos con una gran cantidad de datos y coste computacional, que sería imposible para una persona o empresa mediana debido al coste de comprar las unidades de procesamiento.

- **Transformadores**: Los transformadores están basados en el enfoque a la atención que permiten que el modelo se centre en partes relevantes de la secuencia de entrada, otorgando mayor relevancia a las palabras importantes [4]. Este enfoque ha permitido la creación de herramientas como los modelos mencionados en el apartado anterior como GPT-3 o Midjourney .

Aparte de estos avances de estos últimos años, los algoritmos considerados para resolver el problema a resolver de este proyecto son variados. Las complejidades, beneficios de uso y capacidades de estos algoritmos son muy variadas, por lo que cada uno tiene una serie de ventajas y desventajas según la situación de uso. Los algoritmos son los siguientes:

- **Regresión Logística**: La regresión logística es un algoritmo de aprendizaje supervisado utilizado para la clasificación. A pesar de su nombre, en lugar de una técnica de regresión, es en realidad una técnica de clasificación probabilística, aplicada cuando la variable dependiente es categórica y binaria [5]. Sus principales ventajas residen en su interpretabilidad y eficiencia. Los coeficientes de la regresión logística pueden interpretarse en términos de probabilidades, lo cual resulta útil para entender la relevancia de diferentes características. Además, el algoritmo es relativamente rápido tanto en el entrenamiento como en la predicción, y no asume una relación lineal entre las características y la variable objetivo. Sin embargo, también presenta algunas limitaciones. Para problemas de clasificación multiclase,

es necesario emplear estrategias como “uno contra todos” o “uno contra uno”. Además, en casos en que las clases son perfectamente separables, la regresión logística puede sufrir de överfitting”, requiriendo entonces la aplicación de técnicas de regularización.

- **Random Forest**: Se trata de un algoritmo de aprendizaje supervisado que utiliza el método de *ensamble*, es decir, combina múltiples árboles de decisión para generar un modelo de predicción más robusto. Cada árbol se construye de manera independiente utilizando un subconjunto aleatorio de datos y características. La predicción final se realiza a través del promedio de las predicciones de todos los árboles en el caso de la regresión, o la clase más votada en el caso de la clasificación [6]. Este método es muy eficaz en la gestión de grandes conjuntos de datos con muchas características y tiene la capacidad de manejar tanto problemas de clasificación como de regresión. Sin embargo, a pesar de su robustez, Random Forest puede ser computacionalmente intensivo y, por lo tanto, más lento en comparación con otros algoritmos. Además, a veces puede ser difícil interpretar debido a su naturaleza de caja negra.

- **Redes Bayesianas**: Las redes bayesianas son un tipo de modelo gráfico probabilístico que utiliza el teorema de Bayes para realizar inferencias complejas en situaciones de incertidumbre. Son especialmente útiles cuando se trata de relaciones de causa y efecto, ya que pueden modelar eficazmente la dependencia condicional entre diferentes variables. Estas redes proporcionan una representación gráfica intuitiva que facilita el razonamiento y la interpretación, lo cual es una de sus grandes fortalezas [7]. Asimismo, su flexibilidad para manejar diferentes tipos de variables, ya sean discretas o continuas, y su capacidad para incorporar conocimientos previos en el modelo las hacen una herramienta poderosa en muchas aplicaciones. No

obstante, las redes bayesianas pueden ser computacionalmente exigentes, especialmente con un gran número de variables, y el proceso de aprendizaje de la estructura de la red a partir de los datos puede ser complejo.

- **Support Vector Machines (SVM)**: Las Support Vector Machines, o Máquinas de Vectores de Soporte, son un conjunto de métodos de aprendizaje supervisado utilizados para clasificación y regresión. En su versión básica para clasificación binaria, un SVM busca el hiperplano que maximiza el margen entre las dos clases en el espacio de características [8]. Cuando los datos no son linealmente separables, se aplica una técnica conocida como *trick del kernel*, que permite mapear los datos a un espacio de dimensiones superiores donde sí se pueden separar. Las SVM tienen una alta precisión y son efectivas en espacios con muchas dimensiones. También son robustas frente a overfitting, especialmente en casos donde el número de dimensiones es mayor que el número de muestras. Sin embargo, su desempeño puede verse afectado negativamente si el número de características es mucho mayor que el número de muestras y pueden ser poco eficientes en términos de memoria y tiempo de cálculo en conjuntos de datos grandes.

- **Árboles de Decisión**: Los árboles de decisión son un algoritmo de aprendizaje supervisado que se utiliza tanto para la clasificación como para la regresión. Se representan como estructuras de árbol, donde cada nodo interno denota una prueba en una característica, cada rama representa el resultado de una prueba, y cada nodo hoja es una etiqueta de clase [9]. Los árboles de decisión son intuitivos y fáciles de interpretar, ya que se asemejan a la forma en que los humanos toman decisiones. Adicionalmente, pueden manejar tanto datos categóricos como numéricos. No obstante, los árboles de decisión son propensos al sobreajuste, especialmente cuan-

do se tienen árboles muy profundos, y pueden ser sensibles a pequeñas variaciones en los datos, lo que puede resultar en árboles muy diferentes.

- **XGBoost**:También conocido como Extreme Gradient Boosting, es un algoritmo de aprendizaje supervisado que se basa en el principio de *gradient boosting*. A diferencia de otros métodos de boosting, XGBoost es conocido por su velocidad y rendimiento, ya que está diseñado para ser altamente eficiente, flexible y portátil. XGBoost construye un conjunto de árboles de decisión débiles de forma secuencial, en la que cada árbol se construye para corregir los errores cometidos por los árboles anteriores. El modelo resultante es una combinación ponderada de todos los árboles que puede manejar de manera efectiva tanto problemas de clasificación como de regresión [10]. Aunque XGBoost puede ser menos interpretable que un único árbol de decisión debido a su naturaleza de ensamble, ofrece ventajas significativas en términos de rendimiento predictivo y puede manejar automáticamente los valores perdidos y las variables categóricas.

2.1.3. Blockchain

La tecnología blockchain es una revolucionaria forma de transferencia de datos digitales con un nivel de encriptación muy alto. Esta tecnología fue propuesta inicialmente en el whitepaper de Bitcoin .

Bitcoin introdujo la idea de un sistema de transacciones digitales descentralizado que permite pagos de punto a punto sin la necesidad de una autoridad central que verifique estas transacciones [11]. Inicialmente esta primera implementación de la tecnología blockchain solo tenía como función las transacciones monetarias.

Una transacción en blockchain comienza cuando un usuario inicia una so-

licitud de transferencia de activos digitales. Esta transacción es transmitida a una red de nodos, cada uno de los cuales verifica la transacción para asegurarse de que es válida. Una vez verificada, la transacción se agrupa junto con otras transacciones verificadas en un bloque. Para agregar este bloque a la cadena de bloques existente, los nodos deben resolver un problema matemático en un proceso llamado "prueba de trabajo"(*Proof of Work,PoW*) o "prueba de participación"(*Proof of Stake,PoS*), dependiendo del protocolo blockchain específico. Una vez que el desafío se ha resuelto, el bloque se añade a la cadena de bloques.

Finalmente, la transacción es confirmada y se considera completa. Este proceso, desde la solicitud inicial hasta la confirmación final, asegura que las transacciones en blockchain sean seguras, transparentes e inmutables. Una característica muy importante de las transferencias en Blockchain es la anonimidad de los usuarios de la red.

2.1.4. Ethereum

Ethereum, al igual que Bitcoin, es una criptomoneda de código abierto basada en tecnología blockchain. Sin embargo, Ethereum trasciende la simple transferencia de valor entre pares, característica de Bitcoin, introduciendo funcionalidades notables como los contratos inteligentes y las aplicaciones descentralizadas *(dApps)* [12].

Los contratos inteligentes, o *smart contracts*, son programas que se ejecutan en la red blockchain de Ethereum, siguiendo la lógica codificada por sus creadores. Estos contratos se activan automáticamente cuando se cumplen ciertos requisitos predefinidos en el código, eliminando la necesidad de un intermediario. Los contratos inteligentes tienen una amplia gama de aplicaciones que van más allá del mero intercambio de dinero. Pueden ser utilizados para establecer

y verificar la propiedad de un bien, para facilitar procesos complejos como los derivados financieros o incluso para organizar y registrar votos en una elección, de forma segura y transparente [13].

Además, Ethereum permite la creación de aplicaciones descentralizadas, o dApps. Estas aplicaciones se construyen sobre la red Ethereum utilizando contratos inteligentes, permitiendo a los desarrolladores crear aplicaciones que no son controladas por una sola entidad, sino que se gestionan de forma colectiva. Esto significa que una vez que una *dApp* está en la red Ethereum, su operación ya no depende del desarrollador original. En cambio, está controlada por la lógica codificada en sus contratos inteligentes y operada por la red Ethereum.

Finalmente, Ethereum también introduce el concepto de "gas", que es una medida de la cantidad de trabajo necesaria para ejecutar diversas operaciones en la red, incluyendo las transacciones y los contratos inteligentes. Este mecanismo sirve para evitar el spam en la red y limitar el uso de recursos.

2.1.5. Minado de Datos *(Data Mining)*

La minería de datos, también conocida como *data mining*, es un campo enfocado en la extracción de información útil a partir de grandes conjuntos de datos. En las últimas décadas, ha habido un crecimiento significativo e importantes avances en esta área, impulsados en gran medida por el aumento en la disponibilidad y la escala de los datos.

Las técnicas de minería de datos abarcan desde los métodos clásicos de estadísticas, como el análisis de regresión y la prueba de hipótesis, hasta los métodos modernos de aprendizaje automático y la inteligencia artificial. Los algoritmos de clasificación (por ejemplo, árboles de decisión, redes neuronales, y máquinas de vectores de soporte), agrupamiento (por ejemplo, K-means, DBS-

CAN), y la detección de anomalías (por ejemplo, Isolation Forest, One-class SVM) son especialmente relevantes en la minería de datos.

Con la llegada de la conocida como la era de la información, la minería de datos se ha vuelto cada vez más crucial para una variedad de aplicaciones, incluyendo el análisis de negocios, la detección de fraudes, la bioinformática, y la medicina personalizada. Además, las técnicas de minería de datos también son fundamentales en la minería de texto y la minería web, donde los datos no estructurados (como el texto y las redes de hipertexto) son analizados para extraer información útil [14]. A pesar de los avances, la minería de datos sigue siendo un campo de investigación activo, con desafíos continuos en áreas como el manejo de datos de alta dimensionalidad, la integración de diferentes tipos y fuentes de datos, y la privacidad y la seguridad de los datos.

2.2. Datos sintéticos

Los datos sintéticos son datos generados artificialmente que imitan las características y estructuras de los datos reales. Su principal objetivo es servir como sustitutos de los datos reales en situaciones en las que estos últimos son difíciles de obtener, son insuficientes en cantidad, o están sujetos a restricciones de privacidad y confidencialidad. La generación de datos sintéticos implica la creación de un modelo que aprenda los patrones y distribuciones en los datos reales, para luego generar nuevos datos que mantengan dichas características. Aunque no representan eventos reales, los datos sintéticos pueden ser útiles en una amplia variedad de aplicaciones, como el entrenamiento y prueba de algoritmos de aprendizaje automático, la simulación de escenarios complejos y el análisis de privacidad. Sin embargo, la calidad y utilidad de los datos sintéticos dependen en gran medida de la precisión del modelo generativo utilizado[15].

2.2.1. Datos sintéticos en finanzas

En el campo de las finanzas existe un gran interés por la sintetización de datos para el posterior uso o estudio. Esto se debe a que en muchos casos estos datos con los que se trabajan presentan un nivel de confidencialidad muy alto debido a las leyes de protección de datos. En otros casos se debe a que no existen datos suficientes para la realización de las tareas para los que se quieran usar. Un problema que puede surgir aunque los datos hayan sido anonimizados puede ser que se desanonimicen, esto es factible ya que en algunos casos, como en [16], quienes desanonimizaron el conjunto de datos del premio Netflix utilizando datos de reseñas de Amazon.

En el caso de la detección de fraude es importante tener en cuenta que los datos serán imbalanceados, por ello en muchos casos los resultados al aplicar aprendizaje automático o técnicas de detección de fraude no arroja buenos resultados.

2.3. Sistemas de detección de fraude (FDS)

El campo de la detección de fraude, especialmente en lo que respecta al fraude bancario y de transacciones, ha visto un desarrollo considerable en las últimas décadas. Las innovaciones en tecnología de datos, inteligencia artificial y aprendizaje automático han llevado a avances significativos en la capacidad de identificar y prevenir el fraude. Una de las principales dificultades de la creación de FDS es el *Concept Shift*, que en la minería de datos, el término 'cambio de concepto' se refiere a la situación en la que el modelo o concepto fundamental cambia a medida que avanza el tiempo. Esto es particularmente relevante para los Sistemas de Detección de Fraudes (FDSs), que operan en entornos en constante cambio, donde los comportamientos de los usuarios legítimos y de

los defraudadores están en constante evolución. Tomando el sector de tarjetas de crédito como ejemplo, el comportamiento de un titular de tarjeta podría cambiar debido a numerosos factores externos. Es notable que las cantidades y frecuencias de transacciones a menudo están directamente vinculadas a los hábitos de gasto de una persona, que a su vez están influenciados por factores como ingresos, disponibilidad de recursos y cambios en el estilo de vida. Además, los defraudadores continuamente idean nuevas tácticas, y como tal, los sistemas de detección necesitan ser adaptables a estos tipos emergentes de fraude [17].

En este caso nos centramos en el fraude bancario y de transacciones, ya que existen más tipos de fraude, como el fraude fiscal o fraude a compañías de seguros que también se están viendo ayudados debido al progreso de las tecnologías.

2.3.1. Fraude Bancario

El fraude bancario incluye una variedad de actividades ilegales, como el fraude de tarjetas de crédito, el fraude de cheques, el fraude de préstamos y el lavado de dinero. Los bancos utilizan cada vez más tecnologías avanzadas para detectar y prevenir estas actividades.

Por ejemplo, los algoritmos de aprendizaje automático pueden ser entrenados para detectar patrones de transacciones sospechosas basándose en datos históricos de transacciones fraudulentas y no fraudulentas. Estos algoritmos pueden identificar patrones que pueden ser difíciles o imposibles de detectar para los humanos

Además, las técnicas de análisis de redes se utilizan cada vez más para detectar el fraude bancario. Estas técnicas pueden identificar grupos de cuentas o transacciones que muestran un comportamiento sospechoso, lo que puede indicar

una actividad fraudulenta.

2.3.2. Fraude de Transacciones

Comentar que el fraude en transacciones forma parte de lo que se denomina fraude bancario, este tipo de fraude es uno de los más estudiados. La detección de fraude en las transacciones, como el fraude de tarjetas de crédito y el fraude en línea, es otro área que ha visto un rápido desarrollo, principalmente debido a como ha crecido el sector del e-commerce y como se han empezado a usar sistemas de transacciones de dinero como *Bizum, Paypal....* Por lo que ahora mismo los bancos tienen muchos más datos y muchos más casos de fraude que estudiar.

Como en otros sectores, las técnicas de aprendizaje automático y de inteligencia artificial se utilizan ampliamente para detectar el fraude en las transacciones. Estos sistemas pueden ser entrenados para reconocer patrones de comportamiento fraudulentos y para adaptarse a medida que los estafadores cambian sus tácticas [18].

El creciente número de transacciones con tarjetas de crédito o pagos online ofrece más oportunidades para que los ladrones roben información de los clientes del banco o poseedores de las tarjetas y posteriormente cometan fraude con estos datos. Cuando los bancos pierden dinero debido al fraude con tarjetas de crédito u otros métodos de pago, los titulares de las cuentas bancarias pagan parte de esa pérdida a través de tasas de interés más altas, tarifas más elevadas y reducción de beneficios. Por lo tanto, es de interés tanto para los bancos como para los titulares de las tarjetas reducir el uso ilegítimo de las tarjetas de crédito u otros métodos mediante la detección temprana de fraudes. La mayoría de los estudios previos están hechos con datos de tarjetas de crédito y enfocados en este

sistema de pago principalmente [19]. Además, los métodos de biometría, como el reconocimiento facial o de huellas dactilares, se están utilizando cada vez más para verificar la identidad de una persona y prevenir el fraude de identidad, que es un método común para poder realizar transferencias o pagos sin que el titular de la cuenta sea consciente o este presente cuando se realiza la transacción.

En resumen, la detección de fraude ha avanzado significativamente con el desarrollo de la tecnología de datos, la inteligencia artificial y el aprendizaje automático. Sin embargo, a medida que los estafadores continúan adaptándose y utilizando nuevas tácticas, la detección de fraudes seguirá siendo un campo de investigación activo y en constante evolución.

2.3.3. Problema de los falsos positivos y falsos negativos

En el contexto de sistemas de detección de fraude, los falsos positivos y falsos negativos son problemas cruciales que tienen diferentes tipos de implicaciones. Los falsos positivos se refieren a cuando un sistema marca erróneamente una transacción legítima como fraudulenta. Esto puede ser inconveniente para los clientes, ya que pueden enfrentar bloqueos de cuentas o transacciones, y tener que pasar por procesos de verificación para resolver la situación. Por otro lado, los falsos negativos suceden cuando una transacción fraudulenta no es detectada y se permite que pase como legítima. Esto es particularmente peligroso, ya que puede resultar en pérdidas financieras significativas y daño a la reputación de la entidad involucrada.

En cierto sentido, algunas entidades pueden optar por ajustar sus sistemas de detección de fraude para ser más sensibles y tener una mayor tasa de falsos positivos. Esto se hace con la intención de asegurar que las transacciones fraudulentas no pasen desapercibidas. Aunque esto puede ser inconveniente para los

clientes que enfrentan falsas alarmas, la lógica detrás de esta estrategia es que es preferible correr el riesgo de identificar transacciones legítimas como fraudulentas antes que permitir que el fraude real pase desapercibido. Esta estrategia se puede considerar como una forma de ser extremadamente cautelosos, y puede ser justificada en contextos donde los riesgos y las consecuencias de permitir que el fraude ocurra son muy elevados.

Sin embargo, es importante reconocer que un enfoque excesivamente cauteloso que resulta en una alta tasa de falsos positivos también puede tener consecuencias negativas, incluida la pérdida de confianza y satisfacción del cliente. Por lo tanto, encontrar un equilibrio adecuado en la sensibilidad de los sistemas de detección de fraude es crucial, y puede requerir un monitoreo y ajuste continuo en función de la naturaleza de las transacciones y los riesgos involucrados.

2.4. Estado del arte

La escalada de actividades fraudulentas sofisticadas en la era moderna ha hecho necesaria la evolución de mecanismos de detección igualmente avanzados. La progresión histórica de la detección de fraudes ha experimentado un largo recorrido, abrazando gradualmente la utilización de avances tecnológicos. Los tipos más clásicos de métodos de clasificación estadística, como el análisis discriminante lineal y la discriminación logística, han demostrado ser lo suficientemente efectivos para algunas aplicaciones sin mucha complejidad. Pero otros tipos de técnicas más poderosas han sido estudiadas en los últimos años debido a la cantidad de datos y la complejidad del problema [19]. La inteligencia artificial, en particular, ha emergido como un instrumento potente en la batalla contra el fraude. El área de detección de fraudes es complicada y está llena de obstáculos; a menudo, los sistemas pueden fallar, tener baja precisión o

generar demasiadas alertas falsas. Esto dificulta que las empresas en línea o las compañías de sistemas de pagos aborden el fraude, lo que las lleva a enfrentar grandes pérdidas. Esto se debe a que los sistemas de detección de fraudes tienen que manejar varios desafíos [17]. La mayoría de las investigaciones realizadas en el área de detección de fraudes se centran en transacciones con tarjetas de crédito, la razón de esto es que antes de Internet no había tantas transacciones excepto las realizadas con tarjetas de crédito. Por esta razón, el enfoque de esta sección será en sistemas de detección de fraudes en transacciones con tarjeta de crédito y sistemas de detección de fraudes en transacciones.

En las siguientes secciones se presentarán las diferentes consideraciones de los múltiples estudios similares sobre detección de fraudes utilizando algoritmos de aprendizaje automático.

2.4.1. Datos usados en las investigaciones

No hay muchos conjuntos de datos para abordar este problema, por lo que existen diferentes consideraciones sobre los conjuntos de datos utilizados para diferentes estudios. Los datos de transacciones fraudulentas suelen tener una naturaleza desequilibrada, con instancias fraudulentas significativamente superadas en número por transacciones legítimas. La mayoría de los estudios que abordan este problema utilizan un conjunto de datos desequilibrado, en algunos casos solo el 0.1 % son casos de fraude y el resto transacciones legítimas [20]. Ciertas características dentro de los datos de transacciones de crédito suelen no divulgarse, pero idealmente, deberían incluir elementos como marcas de fecha y hora, detalles de la transacción actual (como monto, ubicación geográfica, código de industria del comerciante y código de validez), historial de transacciones, historial de pagos, e información adicional de la cuenta como la antigüedad de la misma.

La aplicación de datos sintéticos produjo resultados variados cuando se probaron con datos reales. Entre numerosos artículos discutidos en [20], tres utilizaron datos simulados, pero se descubrió que los datos de transacciones de crédito en estos casos carecían de realismo [21]. En algunos estudios, por ejemplo [22], se realiza un preprocesamiento de datos en el conjunto de datos. Se aplica una combinación de submuestreo y sobremuestreo al conjunto de datos significativamente desequilibrado para lograr dos distribuciones diferentes (10:90 y 34:66) para el análisis. Esto se logra mediante la adición y eliminación incremental de un punto de datos interpolado entre puntos de datos existentes hasta que se alcanza el umbral de sobreajuste.

En algunos otros estudios, han utilizado más de un conjunto de datos, como en [23], donde utilizaron un conjunto de datos europeo de transacciones con tarjetas de crédito, compuesto por solo 492 instancias de fraude de un total de 284.807 instancias, y está compuesto por 31 columnas. También han utilizado un conjunto de datos australiano que contiene 383 instancias normales y 307 instancias de fraude. El tercero es un conjunto de datos alemán que contiene 1000 instancias; de los cuales 700 son instancias normales y 300 son instancias de fraude. El conjunto de datos alemán también se utiliza en [24], en este caso también utilizan el conjunto de datos europeo mencionado anteriormente. En [25] utilizan los conjuntos de datos alemán y australiano, el aspecto beneficioso de usar estos conjuntos de datos es que son pequeños, pero probablemente no podrán ser utilizados en otros datos para pruebas.

En [26] han utilizado un conjunto de datos con más de 1 millón de muestras y más de 400 variables características, algunas financieras y otras no financieras.

2.4.2. Mediciones y evaluadores de rendimiento

La mayoría de las investigaciones sobre detección de fraudes que utilizan algoritmos supervisados, desde el año 2001, se han alejado de depender de métricas como la exactitud. En el dominio de la detección de fraudes, los gastos asociados con clasificaciones incorrectas, incluidos tanto falsos positivos como falsos negativos, no son los mismos.

Además, estos costos son impredecibles, pueden variar según el caso específico y están sujetos a cambios con el tiempo. Cabe destacar que, en esta área, la carga financiera de un falso negativo suele ser mayor que la de un falso positivo. Lamentablemente, algunos estudios, como los que se centran en el fraude de transacciones con tarjetas de crédito [21] y el fraude en la superposición, se han centrado únicamente en mejorar la precisión. Algunos estudios emplean análisis ROC (Receiver Operating Characteristic) para colocar la tasa de verdaderos positivos junto a la tasa de falsos positivos [20].

El AUC (Área bajo la curva de característica de funcionamiento del receptor) es una métrica valiosa para los sistemas de detección de fraudes porque captura la compensación entre identificar correctamente el fraude y evitar falsas alarmas, sin ser sensible a los desequilibrios de clases. Además, no depende de un umbral fijo, lo que permite flexibilidad en la toma de decisiones, y resume el rendimiento del modelo en un solo número, lo que facilita la comparación de diferentes modelos; esta métrica se utiliza en [23]. En otros casos, donde el conjunto de datos está más equilibrado, puede ser útil usar precisión, como en [22] [24], pero no es una práctica común. En el caso de [26], han utilizado tanto *Auc Roc Score* como *accuracy* para comparar resultados.

2.4.3. Algoritmos utilizados y resultados obtenidos

Los resultados obtenidos en algunos estudios pueden considerarse muy buenos, pero la mayoría de las veces estos modelos nunca llegan a entrar en producción, por lo que no se han utilizado en un entorno de la vida real.

En [26], como se mencionó anteriormente, han utilizado como métricas Roc Auc Score y Accuracy, y los algoritmos elegidos para sus experimentos fueron Regresión Logística, Máquina de Vectores de Soporte (*Support Vector Machine*) y Random Forest. Los resultados obtenidos para la métrica Auc Roc Score fueron 0.845, 0.916 y 0.925, respectivamente. Para la métrica de precisión obtuvieron 0.921, 0.952 y 0.968. Estos resultados son considerados muy satisfactorios, siendo Random Forest el mejor algoritmo para este caso. En el caso de [22], decidieron utilizar los siguientes algoritmos: *Naïve Bayes*, *k-Nearest Neighbor (KNN)* y *Regresión Logística*. Como se mencionó anteriormente, utilizaron diferentes distribuciones de datos; para la distribución (10:90) obtuvieron resultados similares con *Naïve Bayes* y *KNN*, 0.97 para precisión y 0.82 para sensibilidad, la única diferencia se puede ver en la métrica de precisión donde obtuvieron 1.0 con KNN y 0.05 para *Naïve Bayes*. Para la distribución (34:66) sus resultados fueron nuevamente similares para *Naïve Bayes* y *KNN*, lograron obtener una precisión similar, pero mejoraron significativamente la sensibilidad y la precisión hasta 0.95 y 0.97 respectivamente para *Naïve Bayes*, y hasta 0.93 y 1 para *KNN*. Para los experimentos de Regresión Logística, obtuvieron resultados que no son comparables, logrando 0.54 y 0.58 en precisión y sensibilidad.

En el estudio [24], comparan Máquinas de Vectores de Soporte y algoritmos de Aprendizaje Ensemble, siendo más precisos, Boosting. Como se mencionó anteriormente, se enfocan en la precisión, que no es una métrica típica para este tipo de problema. Los resultados que obtuvieron para el conjunto de datos

alemán fueron 0.7 para SVM y 0.72 para Boosting, sin ver muchas diferencias, lo cual tiene sentido considerando que el conjunto de datos no es grande. Para el otro conjunto de datos obtuvieron 0.49 para SVM y 0.86 para Boosting, esto probablemente se deba a que el conjunto de datos es más grande y Boosting podría ser una buena decisión en estos casos.

En este proyecto, se ha decidido utilizar dos conjuntos de datos; para el conjunto de datos de J.P. Morgan no ha sido posible encontrar trabajos relacionados, por lo que no es posible comparar los resultados obtenidos con ningún otro trabajo.

Para el de Ethereum, hay diferentes proyectos en Kaggle con los cuales se pueden comparar los resultados. Los algoritmos utilizados son en su mayoría los mismos que en otros proyectos, y los resultados son más o menos los mismos. Estos resultados serán discutidos en la parte de conclusión del documento.

3. Objetivos

El objetivo principal del trabajo es la realización de un análisis y creación de un modelo de detección de fraude para dos conjuntos de datos, uno de cuentas fraudulentas de Ethereum y otro con un conjunto de datos de transacciones de J.P. Morgan.

Para el sistema de detección de cuentas categorizadas como fraudulentas de Ethereum se pretende hacer un análisis previo y preprocesamiento de los datos para posteriormente aplicar algoritmos de aprendizaje automático que permitan crear modelos para detectar cuentas fraudulentas.

En el caso del conjunto de datos de J.P. Morgan, también se quiero hacer el análisis necesario y preprocesamiento de estos para crear un modelo con los

algoritmos de aprendizaje automático elegidos y crear un sistema de detección de transacciones fraudulentas.

Con estas tareas descritas anteriormente se pretende aprender técnicas de minado de datos y aprendizaje automático. Para esto se quiere usar Python y aprender sobre las librerías que se usan para el procesamiento de datos y creación de modelos de aprendizaje automático.

Para dejar claro los objetivos a continuación se procede a enumerarlos:

1. Estudiar el estado del arte de los sistemas de detección de fraude.
2. Realizar un análisis y preprocesamiento del conjunto de datos de cuentas fraudulentas de Ethereum.
3. Crear un modelo con algoritmos de aprendizaje automático con el conjunto de datos de cuentas fraudulentas de Ethereum.
4. Realizar un análisis y preprocesamiento del conjunto de datos de transacciones fraudulentas de J.P. Morgan.
5. Crear un modelo con algoritmos de aprendizaje automático con el conjunto de datos de transacciones fraudulentas de J.P. Morgan.
6. Estudiar los resultados obtenidos con ambos modelos.
7. Definir como sería la implementación del modelo en la red blockchain de Ethereum.

4. Desarrollo del proyecto

En este apartado se detalla el proceso de análisis de los datos y la creación de los modelos de detección de fraude.

4.1. Entorno Operacional

En este primer apartado se procede a detallar las características del ordenador usado para la realización del proyecto, que son los siguientes:

Características del sistema	
Procesador	Intel(R) Core(TM) i5-1035G4 CPU @ 1.10GHz 1.5 GHz
Memoria RAM	16,0 GB (15,7 GB usable)
Tipo de Sistema	Sistema operativo de 64 bits, procesador basado en x64 (Windows)

Tabla 1: Características del sistema

Se ha usado Python 3.9.7, y principalmente las librerias *pandas*, *numpy*, *sklearn*, *matplotlib*, *seaborn* y *XGBoost*.

4.2. Métricas de evaluación de los modelos

Como se menciona previamente, el problema a resolver en este caso se trata de detectar cuentas marcadas como fraudulentas. Por lo que es importante centrarse en métricas como:

- ***Recall* (Sensibilidad o Tasa de Verdaderos Positivos)**: Esta métrica indica la proporción de positivos reales (en este caso transacciones fraudulentas) que el modelo ha sido capaz de identificar correctamente.

$$\textbf{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{1}$$

- **Precisión**: La precisión se refiere a la proporción de positivos predichos

(transacciones fraudulentas, en este caso) que fueron realmente positivos.

$$\textbf{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{2}$$

- **F1-Score**: Esta es una métrica que combina recall y precision en un único número utilizando la media armónica. Un F1-score alto indicaría un buen equilibrio entre la detección de fraudes (recall) y la minimización de falsas alarmas (precision).

$$\textbf{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Se ha considerado importante recalcar que, debido a que es más importante detectar todas las cuentas marcadas como fraude, la métrica principal que se tendrá en cuenta será *F1-score*, esto se debe a que el coste de un falso negativo (fraude no detectado) puede ser mucho mayor que el coste de un falso positivo (cuenta legítima marcada como fraude). Claro, que nos centramos en la métrica *recall* si la precisión es relativamente aceptable, si *recall* de las cuentas fraudulentas es alto y la precisión no, significa que está poniendo casi todas las cuentas como fraudulentas, por lo que no sería fiable el modelo.

4.3. Análisis de los datos de Ethereum

En este apartado se procede a comentar el análisis y preprocesamiento de los datos previo a la creación del modelo de detección de cuentas fraudulentas de Ethereum. El desarrollo del análisis y los resultados se pueden encontrar en https://github.com/docampo99/Fraud_Detection_Ethereum.

4.3.1. Obtención de los datos

Este conjunto de datos ha sido obtenido de la página Kaggle, este según se indica en la descripción del conjunto ha sido obtenido usando las API de Ethereum Etherscan API y etherscamdb API, tambíen se han usado técnicas de Web Scraping.

4.3.2. Descripción de los atributos iniciales

En la siguiente tabla se muestran todas los nombres de los atributos y sus descripciones que componen inicialmente el conjunto de datos usado para la creación del modelo de detección de fraude de Ethereum.

- **Index**: el número de índice de una fila.
- **Address**: la dirección de la cuenta de Ethereum.
- **FLAG**: si la transacción es fraudulenta o no.
- **Avg_min_between_sent_tnx**: tiempo promedio entre transacciones enviadas para una cuenta en minutos.
- **Avg_min_between_received_tnx**: tiempo promedio entre transacciones recibidas para una cuenta en minutos.
- **Time_Diff_between_first_and_last_Mins**: diferencia de tiempo entre la primera y la última transacción.
- **Sent_tnx**: número total de transacciones normales enviadas.
- **Received_tnx**: número total de transacciones normales recibidas.
- **Number_of_Created_Contracts**: número total de transacciones de creación de contratos.

- **Unique_Received_From_Addresses**: número total de direcciones únicas desde las que se recibieron transacciones.
- **Unique_Sent_To_Addresses20**: número total de direcciones únicas a las que se enviaron transacciones.
- **Min_Value_Received**: valor mínimo en Ether recibido.
- **Max_Value_Received**: valor máximo en Ether recibido.
- **Avg_Value_Received**: valor promedio en Ether recibido.
- **Min_Val_Sent**: valor mínimo en Ether enviado.
- **Max_Val_Sent**: valor máximo en Ether enviado.
- **Avg_Val_Sent**: valor promedio en Ether enviado.
- **Min_Value_Sent_To_Contract**: valor mínimo en Ether enviado a un contrato.
- **Max_Value_Sent_To_Contract**: valor máximo en Ether enviado a un contrato.
- **Avg_Value_Sent_To_Contract**: valor promedio en Ether enviado a contratos.
- **Total_Transactions_Including_Tnx_to_Create_Contract**: número total de transacciones.
- **Total_Ether_Sent**: total de Ether enviado a la dirección de la cuenta.
- **Total_Ether_Received**: total de Ether recibido por la dirección de la cuenta.
- **Total_Ether_Sent_Contracts**: total de Ether enviado a direcciones de contratos.

- **Total_Ether_Balance**: total de balance de Ether después de las transacciones realizadas.
- **Total_ERC20_Tnxs**: número total de transacciones de transferencia de tokens ERC20.
- **ERC20_Total_Ether_Received**: número total de transacciones recibidas de tokens ERC20 en Ether.
- **ERC20_Total_Ether_Sent**: número total de transacciones enviadas de tokens ERC20 en Ether.
- **ERC20_Total_Ether_Sent_Contract**: número total de transacciones de transferencia de tokens ERC20 a otros contratos en Ether.
- **ERC20_Uniq_Sent_Addr**: número de transacciones de tokens ERC20 enviadas a direcciones de cuenta únicas.
- **ERC20_Uniq_Rec_Addr**: número de transacciones de tokens ERC20 recibidas de direcciones únicas.
- **ERC20_Uniq_Rec_Contract_Addr**: número de transacciones de tokens ERC20 recibidas de direcciones de contrato únicas.
- **ERC20_Avg_Time_Between_Sent_Tnx**: tiempo promedio entre las transacciones de tokens ERC20 enviadas, en minutos.
- **ERC20_Avg_Time_Between_Rec_Tnx**: tiempo promedio entre las transacciones de tokens ERC20 recibidas, en minutos.
- **ERC20_Avg_Time_Between_Contract_Tnx**: tiempo promedio entre las transacciones de tokens ERC20 enviadas a contratos, en minutos.
- **ERC20_Min_Val_Rec**: valor mínimo en Ether recibido de transacciones de tokens ERC20 para la cuenta.

- **ERC20_Max_Val_Rec**: valor máximo en Ether recibido de transacciones de tokens ERC20 para la cuenta.
- **ERC20_Avg_Val_Rec**: valor promedio en Ether recibido de transacciones de tokens ERC20 para la cuenta.
- **ERC20_Min_Val_Sent**: valor mínimo en Ether enviado de transacciones de tokens ERC20 para la cuenta.
- **ERC20_Max_Val_Sent**: valor máximo en Ether enviado de transacciones de tokens ERC20 para la cuenta.
- **ERC20_Avg_Val_Sent**: valor promedio en Ether enviado de transacciones de tokens ERC20 para la cuenta.
- **ERC20_Uniq_Sent_Token_Name**: número de tokens ERC20 únicos transferidos.
- **ERC20_Uniq_Rec_Token_Name**: número de tokens ERC20 únicos recibidos.
- **ERC20_Most_Sent_Token_Type**: token más enviado para la cuenta a través de transacciones ERC20.
- **ERC20_Most_Rec_Token_Type**: token más recibido para la cuenta a través de transacciones ERC20.

4.3.3. Análisis inicial

El conjunto está formado por 9840 entradas y por 50 atributos, el primer paso es quitar la columna que marca el índice ya que no aporta nada de información relevante. El siguiente paso consiste en quitar de nuestro conjunto de datos las columnas que tienen como valor de su varianza igual a 0, ya que el modelo no

obtendrá ningún tipo de informacion de estas, la razón de que su varianza sea 0 es debido a que los valores de todas las filas de estas columnas es 0. Dichas columnas son las siguientes:

ERC20_avg_time_between_sent_tnx
ERC20_avg_time_between_rec_tnx
ERC20_avg_time_between_rec_2_tnx
ERC20_avg_time_between_contract_tnx
ERC20_min_val_sent_contract
ERC20_max_val_sent_contract
ERC20_avg_val_sent_contract

Tabla 2: Atributos eliminados del conjunto de datos

Una vez se han quitado las columnas sin varianza se inicia un análisis superficial de las columnas. Lo primero se observa el número de valores únicos de cada columna, donde por ejemplo en la columna FLAG se espera que solo haya 2 valores únicos ya que es un atributo binario. La mayoría de las columnas han de tener muchos valores únicos porque son valores númericos continuos, excepto las relacionadas con los contratos de Ethereum ya que estos si son valores discretos. Aún así se puede observar que en algunos atributos no hay muchos valores únicos, esto se estudia más en profundidad para ver si proporcionan información relevante al modelo que se pretende crear.

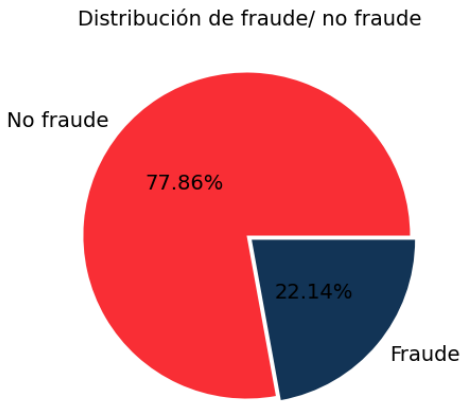


Figura 1: Distribución de cuentas categorizadas como fraudulentas.

La distribución de la columna FLAG, que indica si una cuenta tiene transacciones fraudulentas es la siguiente, 7662 entradas están marcadas como 0, que significa que no son fraudulentas, mientras que las 2179 filas restantes si que se consideran como fraudulentas. Para que el modelo no este sesgado se utilizan técnicas para tener igualar entradas que son fraudulentas y no fraudulentas. Se puede observar como la mayoría no son fraudulentas, esto es previsible y en la realidad se esperaría que fuera menor aún la proporción de casos considerados como fraude debido a la naturaleza de este problema.

Cabe mencionar que las columnas con datos relacionados con ERC20 suelen tener menos valores diferentes ya que la mayoría de las cuentas solo usan la red normal de Ethereum y no está secundaria. Lo que si se ha considerado ha sido en sumar los valores que determinan cuantas transacciones a distintas cuentas ya sean a través de la red de Ethereum o de ERC20 han sido realizadas por cada una de las cuentas, incluyendolas en 2 nuevas columnas, una de las enviadas y otra de las recibidas. Los nombres de las nuevas columnas son los siguientes.

- Total_unique_received

- Total_unique_sent

Después de esto se eliminan las filas duplicadas, pasando de 9841 a 9012.

4.3.4. Estudio de las correlaciones

Para saber cuanta información aporta cada atributo del conjunto de datos se hace un estudio de las correlaciones, así se podrán eliminar las columnas que tengan una correlación muy alta con otra, así se puede reducir el número de columnas con las que se va a trabajar facilitando la computación del modelo.

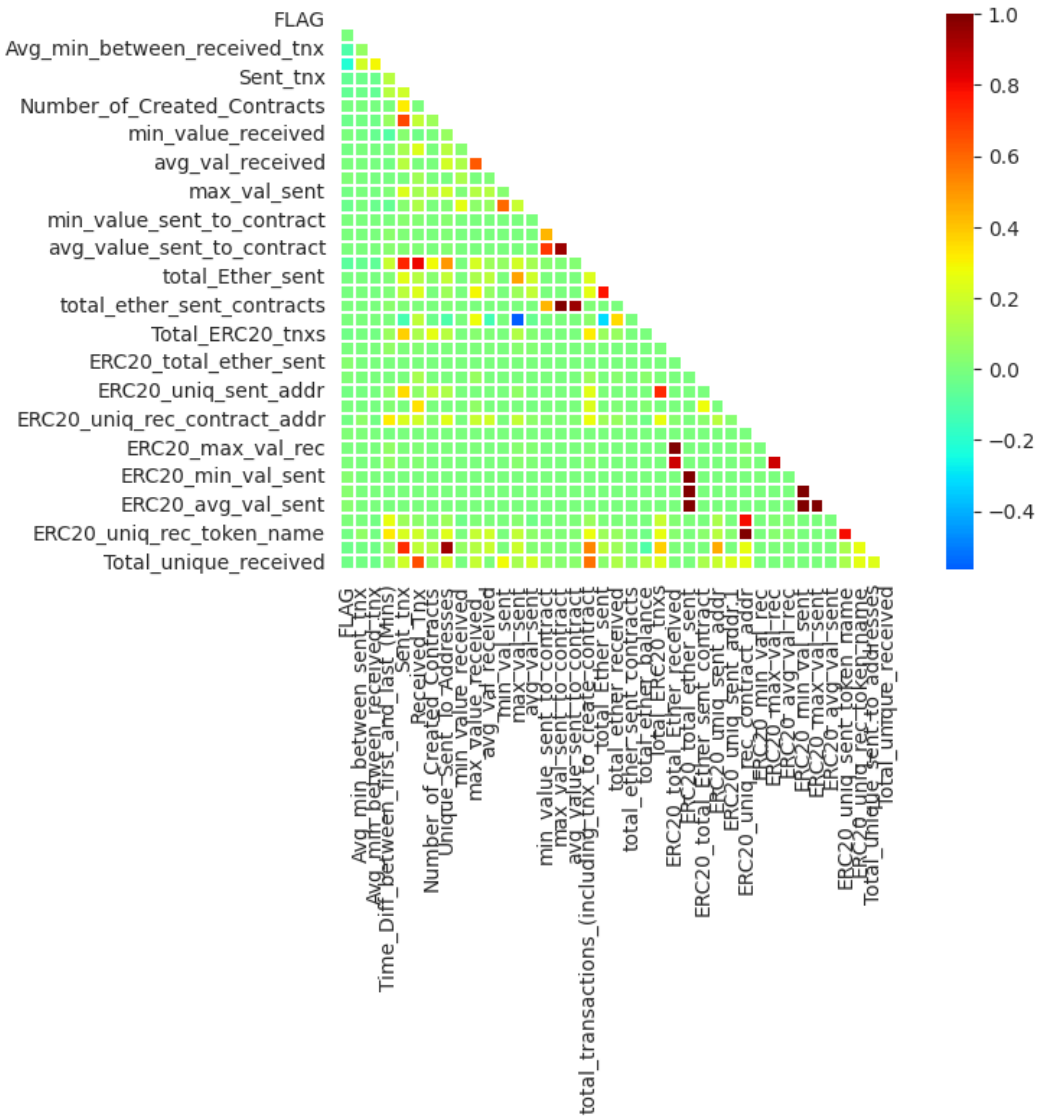


Figura 2: Correlaciones de los atributos iniciales

Los atributos con las correlaciones más altas son por lo general las métricas de ERC20, como la media, máximo o mínimo, ya sean de las transacciones realizadas por la cuenta en las que ha enviado o recibido. Esto también sucede con por ejemplo con algunas de valores de Ether, como avg_val_received y

max_value_received, pero la correlación de estos valores es de aproximadamente 0.6, a diferencia de los valores de ERC20 que están cerca de 1. También existe una correlación alta entre los atributos sobre los contratos, ya sea el valor enviado a los contratos o el número de contratos creados. Esto tiene sentido ya que naturalmente si una cuenta ha creado muchos contratos, pues habrá enviado más veces y más cantidad dinero a estos. Después de estudiar el gráfico previo, se decide que las siguientes columnas no aportan información relevante al modelo debido a la alta correlación con otras columnas , por lo que se quitan del conjunto de datos. Los 22 atributos que se retiran son los siguientes:

Total_transactions_(including_tnx_to_create_contract)
Total_ether_sent_contracts
Max_val_sent_to_contract
ERC20_avg_val_rec
ERC20_avg_val_rec
ERC20_max_val_rec
ERC20_min_val_rec
ERC20_uniq_rec_contract_addr
Max_val_sent
ERC20_avg_val_sent
ERC20_min_val_sent
Min_value_sent_to_contract
Avg_value_sent_to_contract
ERC20_uniq_sent_addr_1
ERC20_max_val_sent
Total_ERC20_tnxs
Total_ether_received
ERC20_uniq_sent_token_name
Min_value_received
Min_val_sent
Unique_Sent_To_Addresses
ERC20_uniq_sent_addr

Tabla 3: Columnas eliminadas por altas correlaciones

Con esto se quedan 18 columnas en el dataset, que presentan las siguientes correlaciones:

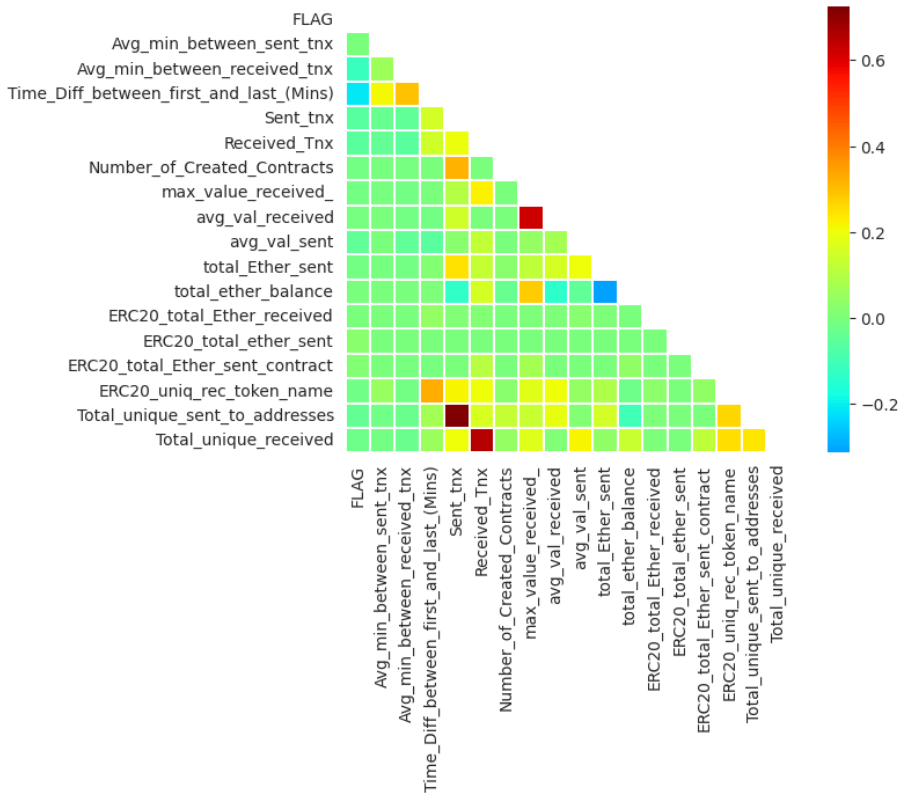


Figura 3: Correlaciones de las columnas finales.

Cabe destacar que aunque se observan tonalidades de rojo muy oscuro, aunque el valor máximo de las correlaciones es de 0,6 aproximadamente. Mientras que en la anterior gráfica se observaba que ese mismo color significaba casi un 100 por ciento de correlación entre las variables. Por lo que se decide mantener ambas columnas ya que una representa el numero total de cuentas diferentes a las que una cuenta ha realizado transacciones y la otra el total de tokens enviados, por lo que se han considerado que aún teniendo una correlación cercana alta la causalidad de estas no está relacionada.

4.3.5. Estudio distribuciones

En este apartado se realiza un estudio de la distribución de los valores de cada atributo. Este análisis aparte de ayudar a comprender la naturaleza de los datos proporciona la información necesaria para la posterior discretización de estos para poder usar posteriormente la técnica One-Hot Encoding. Las distribuciones son las siguientes:

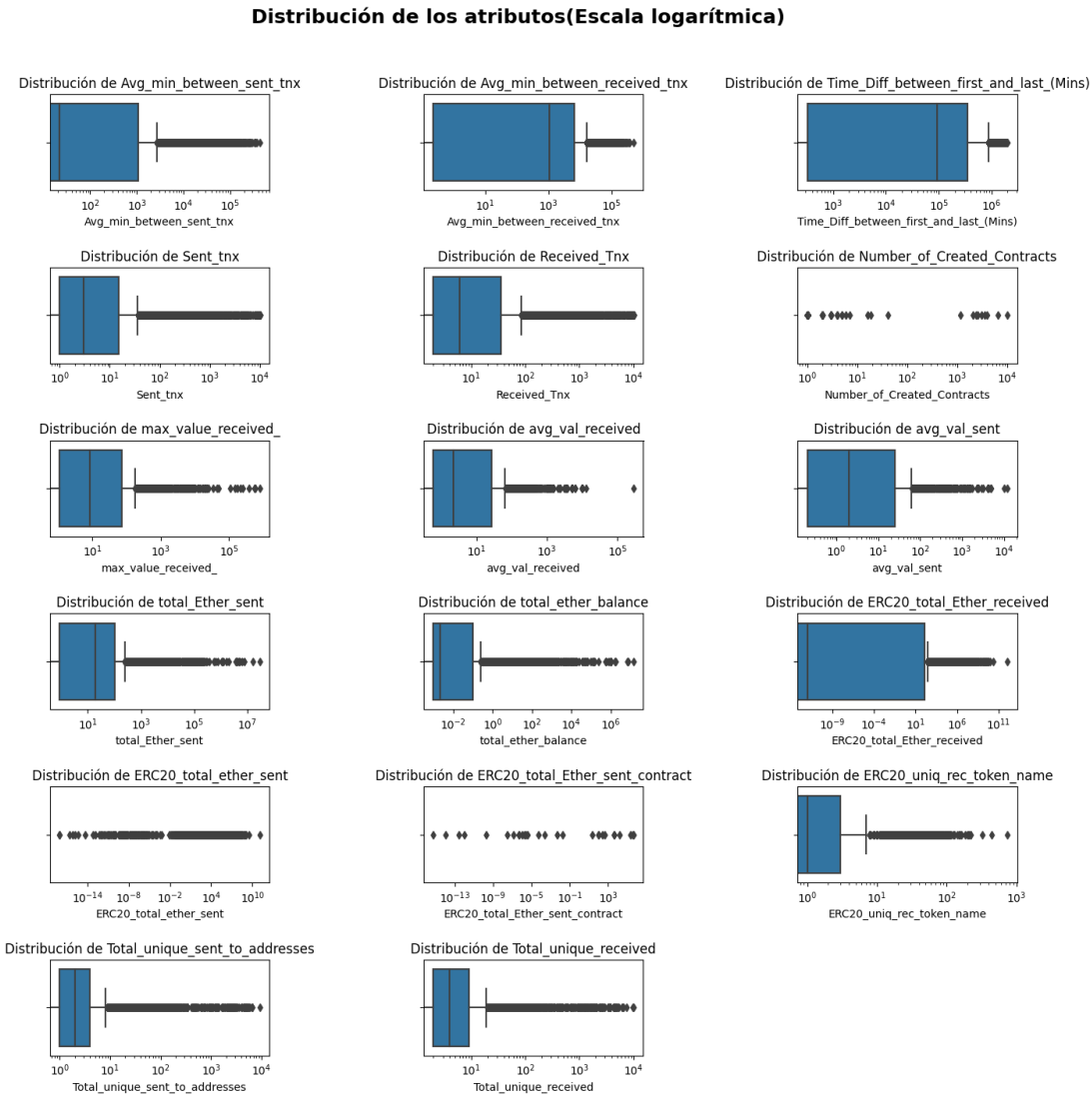


Figura 4: Distribución de los atributos

Como se puede observar estos diagramas de cajas dan a entender que las distribuciones de muchos atributos están totalmente sesgadas, por ellos se muestran con escala logarítmica ya que con escala normal no se aprecia la distribución de los datos. Aparte, también se observan valores atípicos, no se ha considerado

quitar estos valores ya que debido a que una cuenta fraudulenta se considera también un valor atípico. Por lo que no tendría sentido quitar estos valores ya que pueden proporcionar información relevante al modelo.

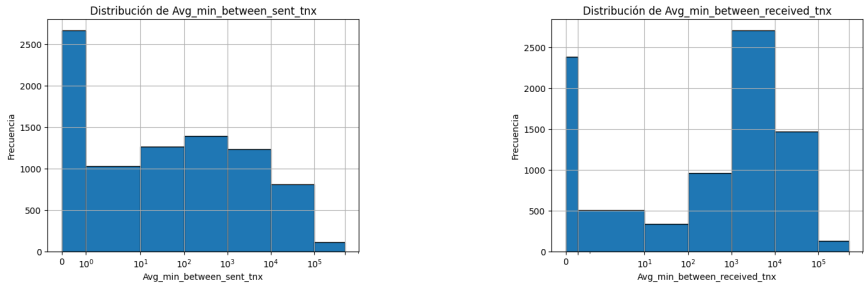


Figura 5: Histogramas de los atributos Avg_min between_sent_tnx, Avg_min between_received_tnx

En estas 2 primeras gráficas se puede observar mejor la distribución de los datos debido al uso de *bins* para poder entenderlos de manera más sencilla visual. Esto se ha hecho con todos las columnas. En los histogramas a continuación se ha hecho lo mismo. Los rangos marcados por los bordes negros marcan también los intervalos que se han considerado para la discretización de dichas columnas. También se han considerado las métricas estadísticas de las columnas para definir los intervalos. Con las dos primeras gráficas se pueden observar diferencias entre las medias de los tiempos entre transacciones enviadas o recibidas. En las transacciones enviadas, gran parte de las cuentas tienen valores pequeños mientras que en la media de las transacciones recibidas una considerable parte de estas están en un rango superior a 1000 minutos. Destacar que también hay muchos valores cerca del 0 pero entre el 0 y 1000 no se guarda la misma proporción que con las diferencias de tiempo de transacciones enviadas por dicha cuenta.

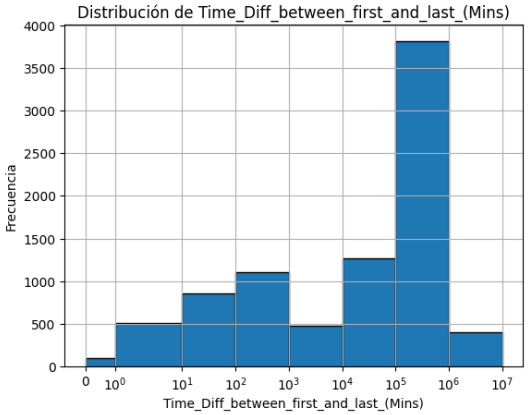
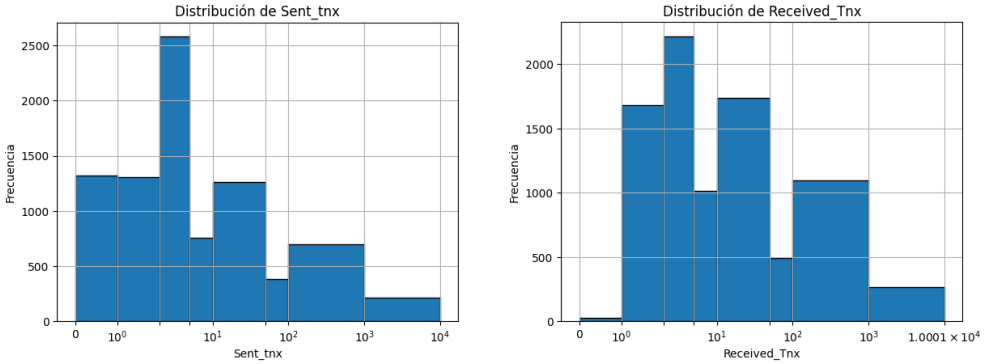


Figura 6: Histograma Time_Diff between_first_and_last_(Mins)

Como es previsible y como se observa en la grafica 6 la diferencia entre la primera y la última transacción tiene la gran mayoría de valores muy altos, del rango de 100.000 minutos.



(a) Histograma de Sent_tnx. (b) Histograma de Received_Tnx.

Figura 7: Histogramas de Sent_tnx y Received_Tnx.

En estas dos gráficas del número total de tokens de Ethereum que han recibido o enviado cada cuenta se pueden observar distribuciones muy parecidas. Cabe destacar que la mayoría de las cuentas tienen en estos atributos valores pequeños, y que ambos debido a outliers tienen una desviación típica respecto a

la media muy alta, también es posible observar como la mediana de los valores es notoriamente inferior a la media, indicando que la media está desproporcionada debido a los valores atípicos.

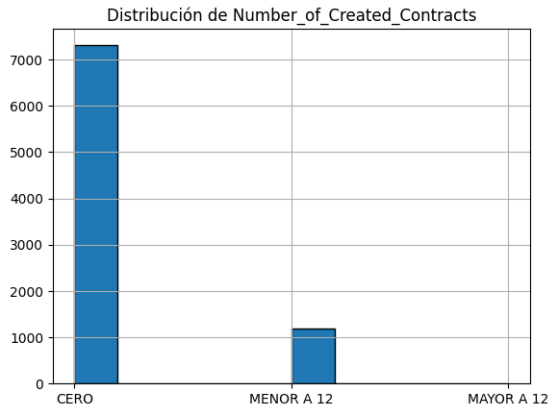


Figura 8: Histograma de Number_of_created contracts

Este atributo, como se observa en la Figura 10, tiene la mayoría de sus valores igual a 0. Esto se debe a que la mayoría de los usuarios no tienen ni los conocimientos ni la motivación, incluso ni los recursos para afrontar los costes de crear un contrato inteligente en la red de Ethereum.Aunque no se aprecie en la imagen, hay 12 cuentas con valores mayores a 12.

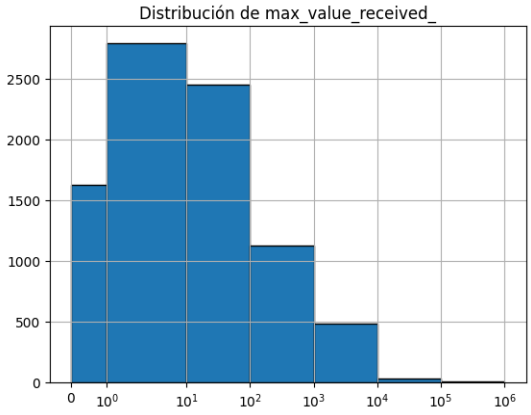
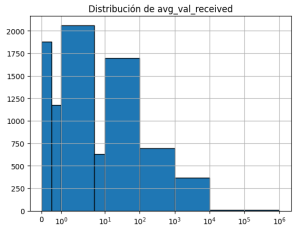
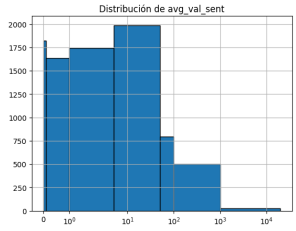


Figura 9: Histograma de Max_Value_received

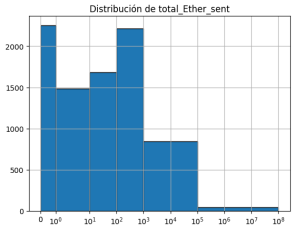
El máximo valor recibido por una cuenta tiene en general valores muy cercanos al 0, pero al mismo tiempo el valor máximo de todas las filas es 80.000 ETH. El razonamiento de este suceso principalmente se entiende considerando las fluctuaciones del precio de un ETH a lo largo de los últimos años, que ha pasado de unos 130 USD a un valor máximo de 4000 USD en 2 años. Estos valores tan altos comparados con la media se explican generalmente con operaciones realizadas cuando el valor de ETH era menor al actual o que haya sido realizado por grandes tenedores de capital en esta red, ya puedan ser, empresas de plataformas de comercio de criptomonedas, fondos de inversión, o inversores particulares que probablemente invirtieron cuando el valor de ETH era mínimo.



(a) Histograma de Avg_Value_received.



(b) Histograma de total_val_received.



(c) Histograma de total_Ether_sent.

Figura 10: Histogramas de Avg_Value_received, total_val_received y total_Ether_sent.

En estos tres atributos, representados en las figuras 10a, 10b y 10c, encontramos una distribución similar tanto entre ellas como a la descrita en el párrafo anterior. Por lo que hay que tener en cuenta la mediana en vez de en la media de los datos debido a que existen valores muy grandes que sesgan estos atributos, esto puede observarse también en los diagramas de cajas de estos atributos expuestos al principio del apartado en la figura 4.

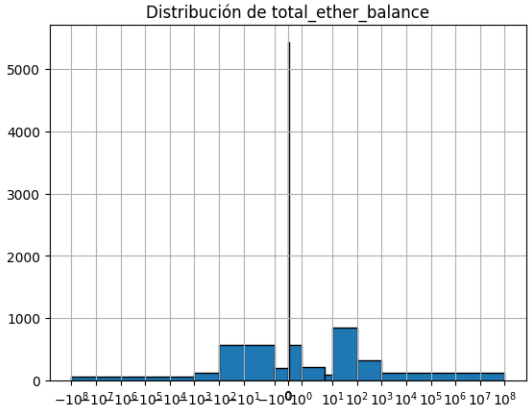


Figura 11: Histograma de total_ether_balance.

Como se espera, en un balance de una cuenta que no se usa para guardar los activos, la mayoría de valores en esta columna son casi 0. Esto se debe ya que

muchos de los tokens que son enviados o los que se invierten con dinero FIAT, luego son usados para comprar otro producto o servicio o son intercambiados de nuevo a la misma u otra moneda o criptomoneda. La gran mayoría de valores están en el intervalo de [0,0.1],exactamente 5383 entradas.

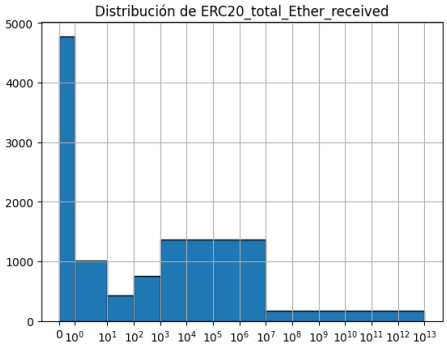


Figura 12: Histograma de ERC20_total_Ether_received.

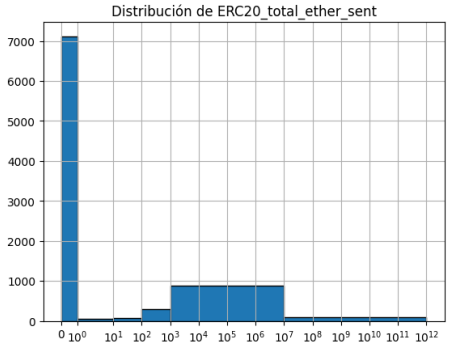


Figura 13: Histograma de ERC20_total_Ether_sent.

Estos dos atributos tienen distribuciones similares. En este caso la mayoría de valores están cerca del 0, al ser este token menos conocido y menos usado que los ETH pues se considera normal que la mayoría de las direcciones no hayan enviado o recibido dicho token. A diferencia de ETH, ERC tiene un valor mucho menor,cercano a 0.01 $, por lo que se puede observar que los valores distintos de 0 de ERC20 enviados y recibidos son mucho mayores comparados a los rangos de los tokens ETH, alcanzando valores de 10^{13} ERC20 .

Las columnas ERC20_total_Ether_sent contract y ERC20_uniq_rec_token_name se ha decidido que sean eliminadas ya que debido al ser la mayoría de sus valores igual a 0 y ya tener dos atributos iguales pero de los tokens de Ethereum no se ha considerado que vayan a aportar información al modelo, así se reduce el número de columnas con el que se trabaja.

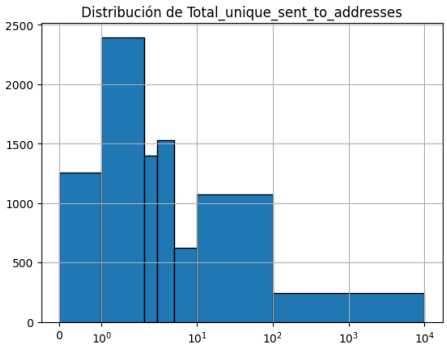


Figura 14: Histograma de Total_unique_sent_to_addresses.

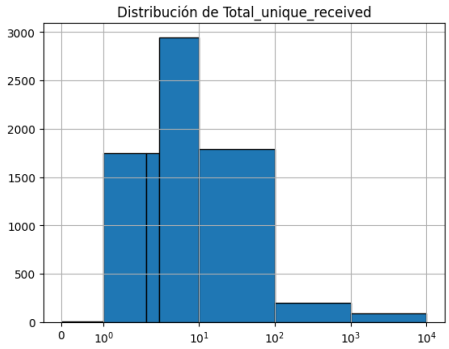


Figura 15: Histograma de Total_unique_received.

El primer atributo fue formado con la suma de las columnas iniciales ERC20 uniq sent addr y Unique Sent To Addresses. Ya que se considera que al enviar dinero a distintas cuentas da igual que tipo de token se este enviando, ya sea ERC20 o ETH. Lo mismo se hizo con el total de cuentas que han enviado dinero a dicha cuenta, sumando ambos valores de ETH y ERC20.

Como se puede observar en los histogramas, ambas tienen muchos valores cercanos cercanos a 0. La mayoría de las cuentas no mandan o reciben muchas transferencias a diferentes cuentas, pero como ha sucedido en las columnas anteriores, podemos encontrar valores mucho mayores en menor proporción.

4.3.6. Descripción datos finales de Ethereum

Después de discretizar todas las columnas para evitar que el modelo salga sesgado, considerando sobretodo la disparidad que podemos encontrar en los datos con los que se trabajan ya que muchos valores pueden ser 0 y luego encontrar valores muy altos respecto a la media y mediana. Los *bins* utilizados son los intervalos que se han usado para representar los atributos y sus distribuciones en el apartado anterior.

Una vez discretizados los datos, se usa la técnica One-Hot Encoding para poder representar los datos con valores numéricos. Esto hace que por cada posible valor de cada atributo se cree una columna, donde si una fila tiene dicho valor del atributo, habrá un 1 en esa columna, mientras que en el resto de los posibles valores de la columna inicial habrá un 0.

Con esto se obtiene finalmente un número total de columnas igual a 113. Ocupando 7.3 Megabytes y con 8518 entradas, donde no hay ningún valor NaN. A partir de este conjunto de datos se separa en los datos de entrenamiento del modelo y los datos de test, con una proporción del 80 % para los datos de entrenamiento y un 20 % para el test.

Después de separar los datos se usa otra para balancear las clases del conjunto de entrenamiento, conocida como SMOTE (Synthetic Minority Oversampling Technique) ,esta técnica estadística de sobremuestreo de minorías sintéticas tiene el objetivo de aumentar el número de casos de un conjunto de datos de forma equilibrada. El componente funciona cuando genera nuevas instancias a partir de casos minoritarios existentes que se proporcionan como entrada.

Una vez usadas estas dos técnicas y haber separado los datos, se tienen 11882 entradas en el conjunto de entrenamiento, formado por un 50 % de transacciones fraudulentas y la otra mitad con transacciones no fraudulentas.

4.4. Análisis de los resultados de Ethereum

En este apartado se presentan los resultados obtenidos en los experimentos para la creación del modelo.

4.4.1. Resultados de los experimentos

En la tabla 4 se muestran los resultados de los algoritmos usados.

Algoritmo	Clase	Precisión	Recall	F1-score	Media Macro Precisión	Media Macro Recall	Media Macro F1-Score
	0	0.99	0.94	0.97			
Logistic Regression	1	0.74	0.92	0.81	0.84	0.93	0.89
	0	0.98	0.94	0.96			
KNN (Neighbour=4)	1	0.69	0.88	0.78	0.84	0.91	0.87
	0	0.97	0.99	0.98			
Random Forest (Estimadores=100)	1	0.92	0.79	0.85	0.94	0.89	0.91
	0	0.98	0.92	0.94			
Naives Bayes (Bernouilli)	1	0.62	0.85	0.72	0.80	0.89	0.83
	0	0.98	0.89	0.93			
Arbol de decisión (Max Depth=4)	1	0.55	0.87	0.80	0.76	0.88	0.80
	0	0.98	0.98	0.98			
Support Vector Machine	1	0.88	0.86	0.87	0.93	0.92	0.92

Tabla 4: Resultados de los experimentos de los datos de Ethereum.

En general la mayoría de los resultados son relativamente buenos, pero el obtenido con SVM y Random Forest se acerca mucho a los obtenidos con XGboost. Aparte de estos algoritmos también se ha usado XGboost, este es un algoritmo de aprendizaje automático que utiliza el enfoque de 'gradient boosting' para producir un modelo predictivo a partir de combinaciones de modelos más simples. Este algoritmo optimiza tanto la precisión de la predicción como la eficiencia computacional, por ello se ha podido hacer la técnica Grid Search y probar combinaciones de distintos parámetros para encontrar el mejor resultado. Debido a que rapídamente se encontraron mejores soluciones que con los otros algoritmos, se probaron más opciones de las consideradas inicialmente.

XGboost Parámetros	Clase	Precisión	Recall	F1-Score	Media Precision	Media Macro Recall	Media Macro F1-Score
'learning_rate': 0.12, 'max_depth': 6, 'n_estimators': 100, 'subsample': 0.8, 'binary':logistic, eval_metric:'auc'	0	0.977	0.986	0.982			
	1	0.909	0.858	0.883	0.943	0.922	0.932
'learning_rate': 0.16, 'max_depth': 9, 'n_estimators': 100, 'subsample': 0.8, 'binary:logistic,eval_metric:'aucpr'	0	0.975	0.987	0.981			
	1	0.911	0.841	0.875	0.939	0.907	0.922
'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 1200, 'subsample': 0.5, 'binary:logistic',eval_metric:'aucpr'	0	0.975	0.989	0.982			
	1	0.924	0.845	0.883	0.950	0.917	0.932
'learning_rate': 0.05, 'max_depth': 6, 'n_estimators': 1000, 'subsample': 0.8, eval_metric'auc',binary:logistic	0	0.974	0.986	0.980			
	1	0.906	0.836	0.870	0.940	0.911	0.925
'learning_rate': 0.12, 'max_depth': 3, 'n_estimators': 300, 'subsample': 0.8, eval_metric:'aucpr', 'binary:hinge'	0	0.974	0.989	0.981			
	1	0.924	0.836	0.878	0.927	0.913	0.930
'learning_rate': 0.2, 'max_depth': 6, 'n_estimators': 300, 'subsample': 0.8, eval_metric:'auc','binary:1ogistic'	0	0.975	0.987	0.981			
	1	0.912	0.845	0.877	0.943	0.916	0.929
'learning_rate': 0.12, 'max_depth': 9, 'n_estimators': 100, 'subsample': 0.2, eval_metric'aucpr',binary:hinge	0	0.975	0.985	0.980			
	1	0.903	0.845	0.873	0.944	0.933	0.938

Tabla 5: Mejores resultados de los experimentos con XGBoost

El mejor resultado que se ha encontrado de todos los experimentos ha sido con XGboost, los parámetros elegidos han sido obtenidos en la búsqueda realizada. Antes de la interpretación del mejor modelo se va a estudiar de manera más profunda que criterio usan los modelos, se procede a interpretar el árbol decisión que se ha obtenido con un parámetro de máxima profundidad igual a 4, así se facilita la visualización de este.

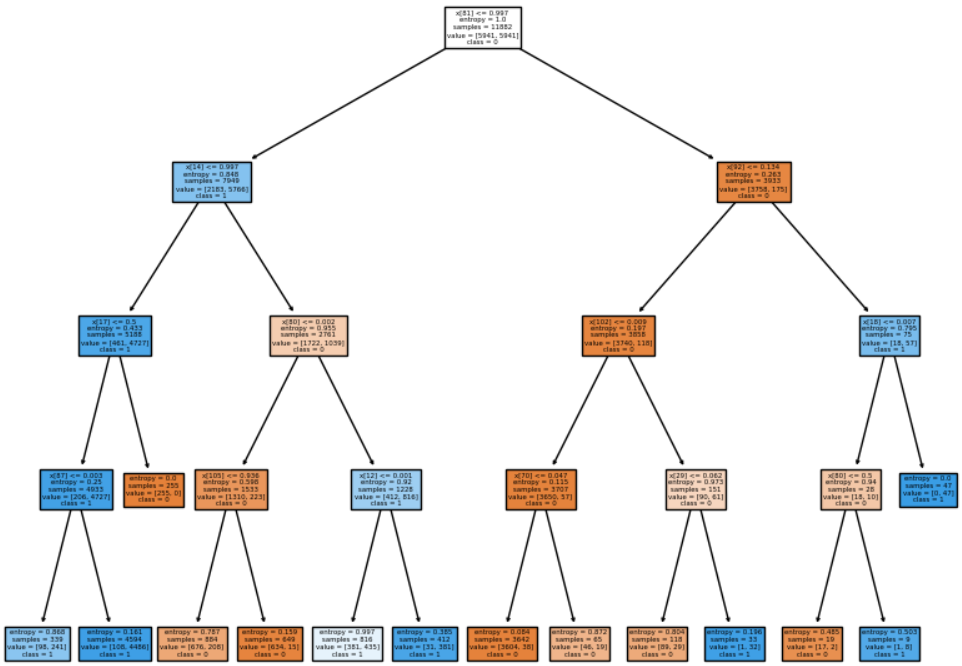


Figura 16: Árbol de decisión obtenido con profundidad igual a 4.

Característica	Importancia
ERC20_total_Ether_received_CERO	0.477607
Time_Diff_between_first_and_last_(Mins)_ALTISIMO	0.216049
Time_Diff_between_first_and_last_(Mins)_MUY ALTO	0.117393
total_ether_balance_UNO	0.070074

Tabla 6: Importancia de las características en el árbol de decisión.

Las características listadas en la tabla son las más influyentes en el resultado del

árbol de decisión. La primera característica, *ERC20_total_Ether_received_CERO*, representa la cantidad total de Ether recibido en transacciones ERC20 que es igual a cero. Esto sugiere que las cuentas que no han recibido Ether a través de transacciones ERC20 son significativamente diferentes en términos de la métrica objetivo. Las siguientes dos características están relacionadas con la diferencia de tiempo entre la primera y la última transacción, lo que indica que el comportamiento a lo largo del tiempo es un factor importante en la predicción.

Las características listadas en la tabla son las más influyentes en el resultado del árbol de decisión. La primera característica, ERC20_total_Ether_received_CERO, representa la cantidad total de Ether recibido en transacciones ERC20 que es igual a cero. Esto sugiere que las cuentas que no han recibido Ether a través de transacciones ERC20 son significativamente diferentes en términos de la métrica objetivo. Las siguientes dos características están relacionadas con la diferencia de tiempo entre la primera y la última transacción, lo que indica que el comportamiento a lo largo del tiempo es un factor importante en la predicción.

4.4.2. Interpretación del modelo final de cuentas de Ethereum

Los resultados encontrados con XGBoost pueden aportar mucho valor y se consideran como notoramiente satisfactorios, centrándonos en la sensibilidad del modelo de detectar el 84 % de las cuentas que son consideradas como fraudulentas, y que cuando determina que una cuenta es fraudulenta el 92.4 % de las veces lo es, se puede afirmar que este modelo realiza su función de manera correcta.

En el inicio del planteamiento del problema se consideraba de antemano que el algoritmo detectaría algunas cuentas de más que no eran fraudulentas como que si lo eran, ya que siguiendo el dicho popular, *es mejor prevenir que curar* a la hora de realizar una transferencia a una cuenta desconocida. Pero finalmente

no se considera que el modelo determine de manera errónea muchas cuentas que no son fraudulentas como tal. la interpretación de estos resultados es la siguiente:

- Precisión para cuentas no fraudulentas: 97.5 %. Indica que el 98.1 % de las cuentas que el modelo predijo como no fraudulentas eran efectivamente no fraudulentas.
- Recall (Sensibilidad) para cuentas no fraudulentas: 98.9 %. Indica que el modelo identificó correctamente el 98.9 % de todas las cuentas no fraudulentas reales.
- Puntuación F1 para cuentas no fraudulentas: 98.2 %. Esta es una medida de la precisión del modelo en la predicción de las cuentas no fraudulentas que considera tanto la precisión como la sensibilidad.
- Precisión para cuentas fraudulentas: 92.4 %. El 92.4 % de las cuentas que el modelo predijo como fraudulentas eran efectivamente fraudulentas.
- Recall (Sensibilidad) para cuentas fraudulentas: 84.5 %. Esto indica que el modelo identificó correctamente el 84.5 % de todas las cuentas fraudulentas reales.
- F1-Score para cuentas fraudulentas: 88.3 %. Esta es una medida de la precisión del modelo en la predicción de las cuentas fraudulentas que considera tanto la precisión como la sensibilidad.
- Exactitud: 96.9 %. Esta es una medida de la proporción total de predicciones correctas (tanto fraudulentas como no fraudulentas) realizadas por el modelo. Este dato no aparece en la tabla.
- Precisión media macro: 95.0 %. Este es el promedio de la precisión para cuentas no fraudulentas y fraudulentas.

- *Recall* (Sensibilidad) media macro: 91.7 %. Este es el promedio de la sensibilidad para cuentas no fraudulentas y fraudulentas.
- F1-Score media macro: 93.2 %. Este es el promedio de la puntuación F1 para cuentas no fraudulentas y fraudulentas.

4.5. Análisis de los datos de J.P. Morgan

En esta segunda parte del proyecto los datos provienen como se menciona previamente en el documento del banco estadounidense J.P. Morgan. Estos datos sintéticos representan transacciones desde una perspectiva centrada en el sujeto con el objetivo de identificar transacciones fraudulentas [27]. Estos datos contienen una gran variedad de tipos de transacciones que representan actividades normales, así como actividades anormales/fraudulentas que se introducen con probabilidades predefinidas. Los datos se generaron mediante la ejecución de un simulador de planificación-ejecución de IA y traduciendo las trazas de planificación de salida a formato tabular. Los parámetros del modelo de generación de datos incluyen el número de clientes, la duración del tiempo y las probabilidades de fraude [15].

Debido a la confidencialidad de los datos el desarrollo en Python del proyecto no esta público.

4.5.1. Descripción de los atributos iniciales

Inicialmente el conjunto de datos está formado por 13 atributos y por 1.498.178 filas. Los atributos iniciales son los siguientes:

Atributo	Descripción
Time_step	Fecha y hora de la transferencia. Con formato YYYY-MM-DD HH:MM.
Transaction_Id	Identificador de la transferencia. Formado por el tipo de transferencia y un número identificador.
Sender_Id	Identificador del remitente de la transferencia. Formado por el tipo de cliente y un número identificador.
Sender_Account	Identificador de la cuenta del remitente de la transferencia.
Sender_Country	País de la cuenta remitente.
Sender_Sector	Sector del remitente.
Sender_lob	Indica si pertenece a la línea de negocio Credit and Consumer Banking.
Bene_Id	Identificador de la cuenta del destinatario de la transferencia. Formado por el tipo de cliente y un número identificador.
Bene_Account	Identificador de la cuenta del destinatario de la transferencia.
Bene_Country	País de la cuenta destinataria.
USD_amount	Valor de la transferencia realizada en Dólares estadounidenses.
Label	Indica si la transferencia esta categorizada como fraude o no. Atributo a predecir.
Transaction_Type	Tipo de transferencia realizada.

Tabla 7: Nombre y descripción de los atributos iniciales del conjunto de datos de transferencias.

4.5.2. Análisis inicial

El primer paso del análisis es estudiar la distribución de transferencias categorizadas como fraude.

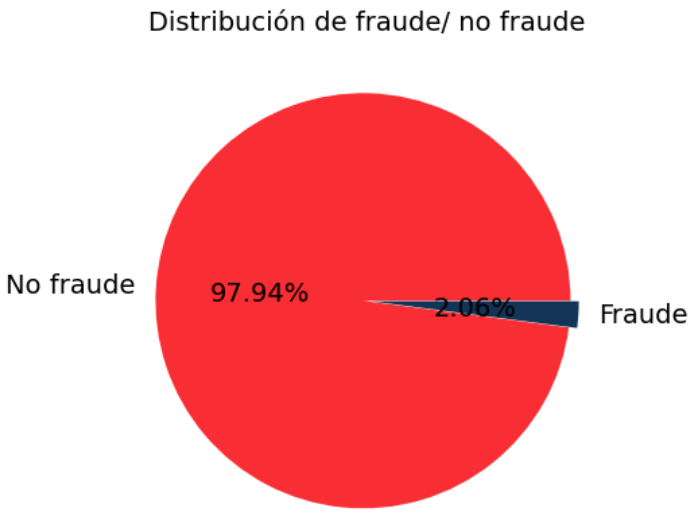


Figura 17: Proporción de las transferencias categorizadas como fraudulentas.

Como se puede observar y era de esperar la proporción de transferencias fraudulentas es muy inferior a las que que no lo son. Ya se sabe que el conjunto de datos está considerablemente imbalanceado. En un principio se consideró eliminar las filas sin valores en alguna columna, pero después de analizar los datos se consideró que esto podía afectar al entrenamiento del modelo debido a que cuando por ejemplo se realiza una retirada de efectivo *(WITHDRAWAL)*, pues no existen valores del beneficiario, solo del remitente. Al igual sucede con pagos en metálico o con un cheque, donde solo aparecen los valores del beneficiario.

La mayoría de las columnas no aportan ningún tipo de información relevante al modelo, excepto atributos como USD_Amount, Sender_lob o Transaction_Type. Aunque no aporten información el resto de columnas son útiles ya que usando técnicas de *ingeniería de características* se pueden obtener nuevos atributos de los iniciales que si que aportan información al modelo.

Un ejemplo de como los datos están estructurados y sus formatos es el si-

guiente:

Time_step	Transaction_Id	Sender_Id	Sender_Account	Sender_Country	Sender_Sector	Sender_Iob	Bene_Id	Bene_Account	Bene_Country	USD_amount	Label	Transaction_Type
2022-03-15 10:24:00	QUICK-PAYMENT-10116	JPMC-CLIENT-10098	ACCOUNT-10109	USA	15287	CCB	CLIENT-10100	ACCOUNT-10106	CANADA	622.78	0	QUICK-PAYMENT
2022-03-15 10:24:00	PAY-CHECK-9832	JPMC-CLIENT-9812	ACCOUNT-9825	USA	38145	CCB	JPMC-CLIENT-9814	ACCOUNT-9824	USA	989.09	0	PAY-CHECK
2022-03-16 23:34:00	MOVE-FUNDS-2520	CLIENT-2510	ACCOUNT-2511	DOMINICA	4628	CCB	JPMC-CLIENT-2492	ACCOUNT-2502	USA	353.59	1	MOVE-FUNDS

Tabla 8: Ejemplo de los valores del conjunto de datos de transferencias.

El atributo Transaction_Id está formado por el valor que existe en el atributo Transaction_Type y un número identificativo. Por ello está columna se retira de la entrada del algoritmo de entrenamiento para obtener el modelo final centrándonos solo en Transaction_Type. También el atributo Time_Step no aporta información al modelo como tal, pero si que se usa para sacar nuevas columnas que si que lo hacen.

4.5.3. Estudio de los atributos

En este apartado se presentan los las distribuciones y cualquier tipo de información relevante que se ha obtenido de los atributos inciales. En el siguiente histograma se puede observar la distribución de los 4 posibles tipos de transferencias:

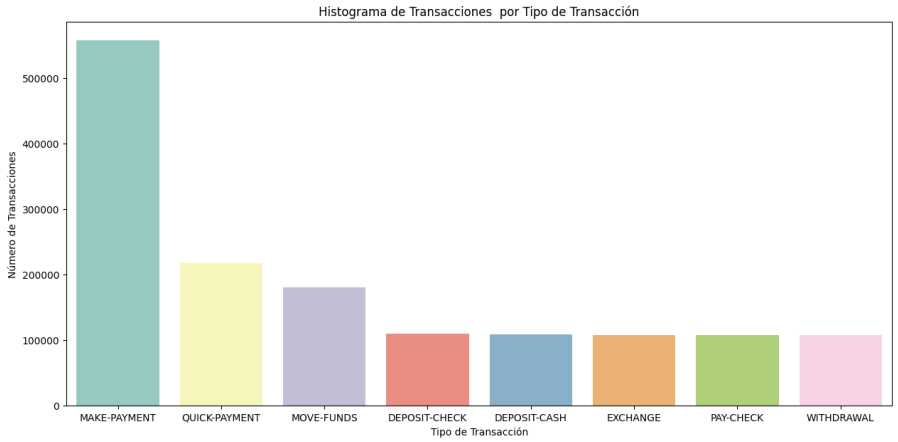


Figura 18: Distribución de los tipos de transferencias.

Aquí se muestran las proporciones de los tipos de transferenciaS:

Tipo de Transferencia	**Porcentaje (%)**
MAKE-PAYMENT	37.22
QUICK-PAYMENT	14.52
MOVE-FUNDS	12.02
DEPOSIT-CHECK	7.30
DEPOSIT-CASH	7.30
EXCHANGE	7.22
PAY-CHECK	7.19
WITHDRAWAL	7.19

Tabla 9: Porcentaje de los tipos de transferencias.

Como se observa en la tabla 9, la mayoría de las transacciones consisten en pagos normales, luego pagos rápidos, movimientos de fondos y el menor número de transferencias es el de las nóminas. La razón de esto se entiendo como que el propietario de una cuenta recibirá una vez al mes una nómina pero realizará muchos tipos de movimientos y pagos durante ese mismo mes. En el siguiente histograma podemos ver el número de transacciones fraudulentas teniendo en cuenta el tipo de transacción:

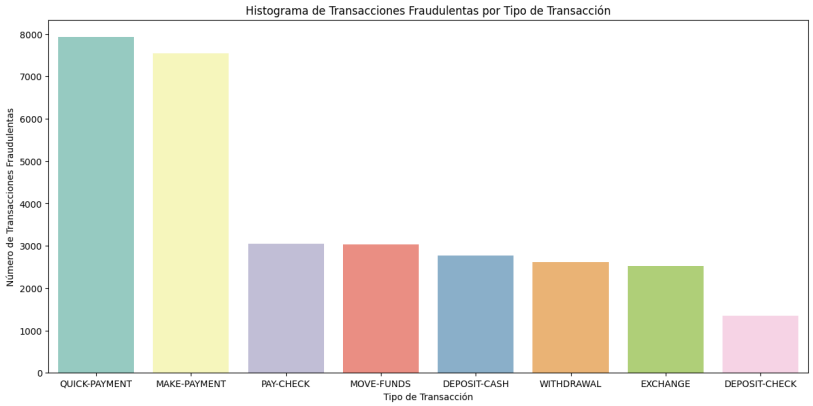


Figura 19: Distribución de las transferencias categorizadas como fraudulentas.

A continuación, en la tabla 10 se muestran los porcentajes de transacciones fraudulentas según el tipo de transacción:

Tipo de Transferencia	**Porcentaje (%)**
QUICK-PAYMENT	25.74
MAKE-PAYMENT	24.47
PAY-CHECK	9.88
MOVE-FUNDS	9.86
DEPOSIT-CASH	8.99
WITHDRAWAL	8.46
EXCHANGE	8.19
DEPOSIT-CHECK	4.37

Tabla 10: Porcentaje de los tipos de transferencias fraudulentas.

Los valores de QUICK-PAYMENT y MAKE-PAYMENT tienen mayores tran-

sacciones fraudulentas asociadas, lo más notorio de esto es la diferencia entre QUICK-PAYMENT respecto al número de transacciones totales de este tipo, siendo el 3,67 % de las transacciones de este tipo consideradas como fraudulentas. La razón de esto se entiende debido a que al ser *PAY-CHECK* (*Nómina*) un tipo de transferencia por lo general recurrente para pagar a un trabajador pues es menos frecuente que haya algún tipo de actividad fraudulenta asociada. Al igual que MOVE-FUNDS, que consiste en un movimiento de fondos que no es un pago rápido y corriente entre dos cuentas si no un movimiento con mayor relevancia para el propietario de los fondos. En el caso de *DEPOSIT-CHECK*, podemos ver como la proporción de transacciones de este tipo es menor a la proporción de transacciones fraudulentas ya que por lo general ya no hay muchos fraudes con cheques.

Respecto a los países de origen y destino de las transferencias se confirma que la mayoría de las transferencias provienen o están dirigidas a cuentas en Estados Unidos, se considera normal ya que se trata de un banco de este país. Debido a esta razón y también porque posteriormente se quiere usar la técnica One-Hot Encoding se han juntado la mayoría de los países en continentes excepto EE.UU, Alemania y Cánada ya que son los países con más transferencias y pueden aportar información. Con esto se consigue que no se creen para cada país una columna nueva, facilitando la computación del modelo. En las siguientes gráficas se presenta como quedan las distribuciones de los primeros 5 valores tanto para los remitentes como los beneficiarios de las transferencias.

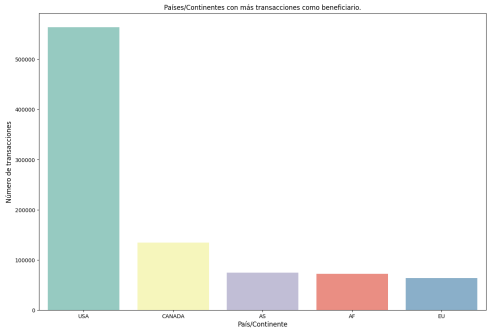


Figura 20: Distribución de los países/continentes con transferencias como beneficiarios.

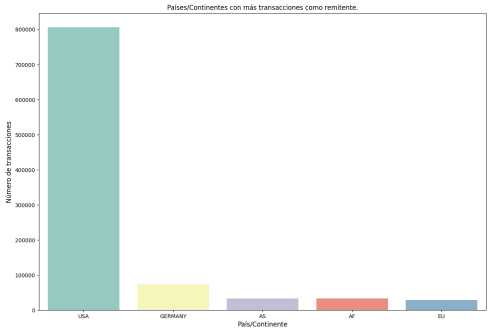


Figura 21: Distribución de los países/continentes con transferencias como remitentes.

De las columnas Sender_Id y Bene_Id se han quitado los valores identificativos para poder obtener información relevante del tipo de cuenta del remitente y del beneficiario. Una vez hecho esto se pueden observar los tipos de cuentas y las distribuciones de las transferencias según estas:

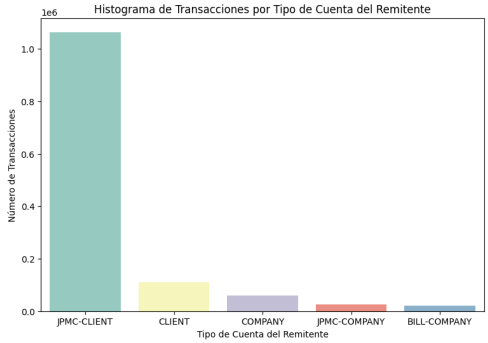


Figura 22: Distribución de los tipos de clientes remitentes.

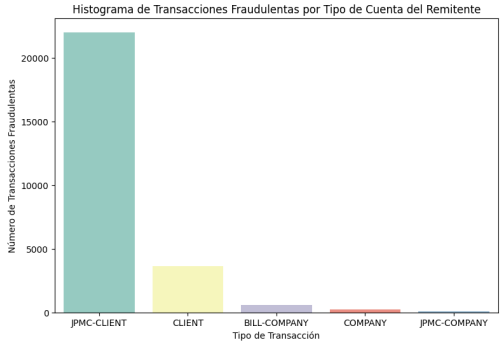


Figura 23: Distribución de los tipos de clientes beneficiarios

En la Figura 14 se comprueba como la proporción es similar entre las transacciones fraudulentas y las totales, aumentando un poco la proporción de los remitentes que no son clientes de J.P. Morgan.

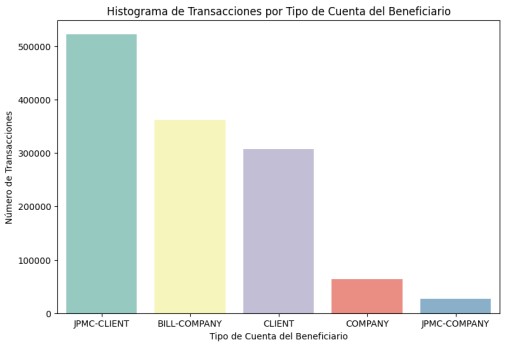


Figura 24: Distribución de los tipos de clientes remitentes.

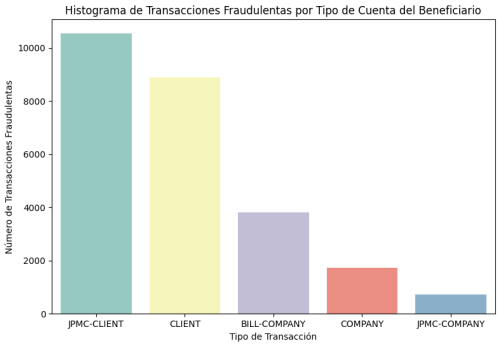


Figura 25: Distribución de los tipos de clientes beneficiarios

En la Figura 16 el tipo de cuenta beneficiario *CLIENT* en las transacciones donde se ha categorizado como fraudulenta es mayor la proporción respecto al total de las transferencias, donde hay más del tipo *BILL-COMPANY*.

Claramente se observa una diferencia entre las Figuras 13 y 15 muy grande entre los remitentes y beneficiarios en el tipo de cuenta *BILL-COMPANY*, el sentido a esta diferencia se entiende debido a que una compañía que recibe facturas recibirá muchísimas más transferencias a su cuenta en vez de realizarlas, y habrá muchos tipos de clientes que realizarán transacciones a este tipo de empresas.

Respecto a las cantidades de dinero enviadas en cada transacción, En promedio, las transferencias son de aproximadamente 513.48 $. Sin embargo, los valores varían bastante, ya que la desviación estándar es de alrededor de 539.81 $, lo que indica una dispersión significativa en las cantidades transferidas.

El valor mínimo de las transferencias es extremadamente bajo, solo 0.01 $, mientras que el valor máximo es bastante alto, siendo casi 20.000 $. Esto muestra que hay una amplia gama de valores en las transferencias.

La mediana, o el valor del medio cuando todos están ordenados, es de cerca

de 404.17 $, lo que significa que la mitad de las transferencias son por menos de esta cantidad y la otra mitad son por más.

Respecto a la columna Sender_lob, 1279268 entradas tienen valor igual a CCB, que determina que el remitente forma parte de la línea de negocio de *Consumer and Credit Banking*. Esta línea se ha hecho binaria para simplificar el trabajo posterior de One-Hot Encoding y no tener dos columnas.

4.5.4. Ingeniería de características

De las columnas iniciales se ha hecho un procesamiento de estas para sacar nuevos atributos para intentar darle conociminento al algoritmo y que el modelo final arroje los mejores resultados posibles.

Las nuevas columnas que se han obtenido son las siguientes:

- **Bene_ClienteJP**: Esta columna es una variable binaria que indica si el beneficiario (Bene) de la transacción es un cliente del banco J.P. Morgan Chase. Toma el valor 1 si el Bene_Id comienza con "JPMC", lo que indica que es un cliente del banco, y 0 en caso contrario.

- **Sen_ClienteJP**: Similar a Bene_ClienteJP, (Sender) de la transacción es un cliente del banco. Toma el valor 1 si el Sender_Id comienza con "JPMC", lo que indica que es un cliente del banco, y 0 en caso contrario.

- **National_Transaction**: Esta columna es una variable binaria que indica si la transacción es nacional, es decir, si el país del remitente (Sender_Country) es igual al país del beneficiario (Bene_Country). Toma el valor 1 si ambos países son iguales, indicando una transacción nacional, y 0 en caso contrario. Se ha creado antes de pasar los países a continentes para guardar más información en menos columnas.

- **Time_Diff_Last_Trans_Sen**: Esta columna representa la diferencia de tiempo en segundos entre la transacción actual y la transacción anterior realizada por la misma cuenta de remitente (Sender_Account). Esto se logra agrupando el DataFrame por Sender_Account y luego calculando la diferencia entre el valor actual de Time_step y su valor anterior dentro de cada grupo. Si no hay una transacción anterior (es decir, es la primera transacción de esa cuenta de remitente), se rellena con 0.

- **Time_Diff_Last_Trans_Ben**: Representa la diferencia de tiempo en segundos entre la transacción actual y la última transacción en la que la misma cuenta fue beneficiaria ('Bene_Account'). Similar a 'Time_Diff_Last_Trans_Sen' pero enfocándose en las cuentas beneficiarias.

- **Average_Diff_Time_Rec**: Representa el promedio acumulativo de la diferencia de tiempo entre transacciones consecutivas en las que cada cuenta fue beneficiaria, excluyendo la primera transacción de cada cuenta. Es útil para comprender el comportamiento de las cuentas beneficiarias a lo largo del tiempo en términos de frecuencia de transacciones recibidas.

- **Average_Diff_Time_Sen**:Representa el promedio acumulativo de la diferencia de tiempo entre transacciones consecutivas en las que cada cuenta fue el remitente, excluyendo la primera transacción de cada cuenta. Es útil para comprender el comportamiento de las cuentas que envían dinero a lo largo del tiempo en términos de frecuencia de transacciones recibidas.

- **average_USD_sent**: Representa el promedio acumulativo del monto en dólares estadounidenses (USD) enviado por cada cuenta de remitente hasta la transacción actual, excluyendo la transacción actual en el promedio.

- **average_USD_rec**: Similar a *average_USD_sent*, pero calcula el promedio acumulativo de los montos en USD recibidos por cada cuenta beneficiaria

(*Bene_Account*).

- **Max_USD_Sent**: Para cada cuenta de envío (*Sender_Account*), esta columna almacena el monto máximo en USD que ha sido enviado por esa cuenta en cualquier transacción hasta el momento.
- **Max_USD_Rec**: Similar a *Max_USD_Sent*, pero para cuentas beneficiarias (*Bene_Account*). Almacena el monto máximo en USD que ha sido recibido por cada cuenta beneficiaria en cualquier transacción hasta el momento.
- **Min_USD_Sent**: Para cada cuenta de envío (*Sender_Account*), esta columna almacena el monto mínimo en USD que ha sido enviado por esa cuenta en cualquier transacción hasta el momento.
- **Min_USD_Rec**: Similar a *Min_USD_Sent*, pero para cuentas beneficiarias (*Bene_Account*). Almacena el monto mínimo en USD que ha sido recibido por cada cuenta beneficiaria en cualquier transacción hasta el momento.
- **Time_step:** Esta columna es convertida a formato de fecha y hora para permitir cálculos temporales. Además, se le añade un pequeño valor aleatorio en milisegundos para asegurar la unicidad de cada registro temporal.
- **Sender_Transactions_Last_1_Days:** Número de transacciones enviadas por la cuenta remitente (*Sender_Account*) en los últimos 1 día. Ayuda a entender la frecuencia de transacciones en un corto plazo.
- **Sender_Transactions_Last_7_Days:** Similar a la columna anterior, pero calcula el número de transacciones enviadas por la cuenta remitente en los últimos 7 días. Permite evaluar la actividad de la cuenta en una semana.
- **Sender_Transactions_Last_30_Days:** Similar a las dos columnas anteriores, pero calcula el número de transacciones enviadas por la cuenta

remitente en los últimos 30 días. Ofrece una perspectiva de la actividad de la cuenta en un mes.

- **Bene_Transactions_Last_1_Days:** Número de transacciones recibidas por la cuenta beneficiaria (*Bene_Account*) en los últimos 1 día. Similar a *Sender_Transactions_Last_1_Days* pero para la cuenta beneficiaria.

- **Bene_Transactions_Last_7_Days:** Similar a Sender_Transactions_Last_7_Days, pero calcula el número de transacciones recibidas por la cuenta beneficiaria en los últimos 7 días.

- **Bene_Transactions_Last_30_Days:** Similar a *Sender_Transactions_Last_30_Days*, pero calcula el número de transacciones recibidas por la cuenta beneficiaria en los últimos 30 días.

La figura abajo muestra una serie de boxplots, que son utilizados para visualizar y comparar la distribución de distintas características numéricas en nuestro conjunto de datos de transacciones. Cada boxplot representa una variable específica, como la cantidad en dólares de las transacciones o el tiempo transcurrido entre ellas. Es importante destacar que estamos utilizando una escala logarítmica en el eje Y para permitir una mejor visualización de los datos, que pueden variar en órdenes de magnitud.

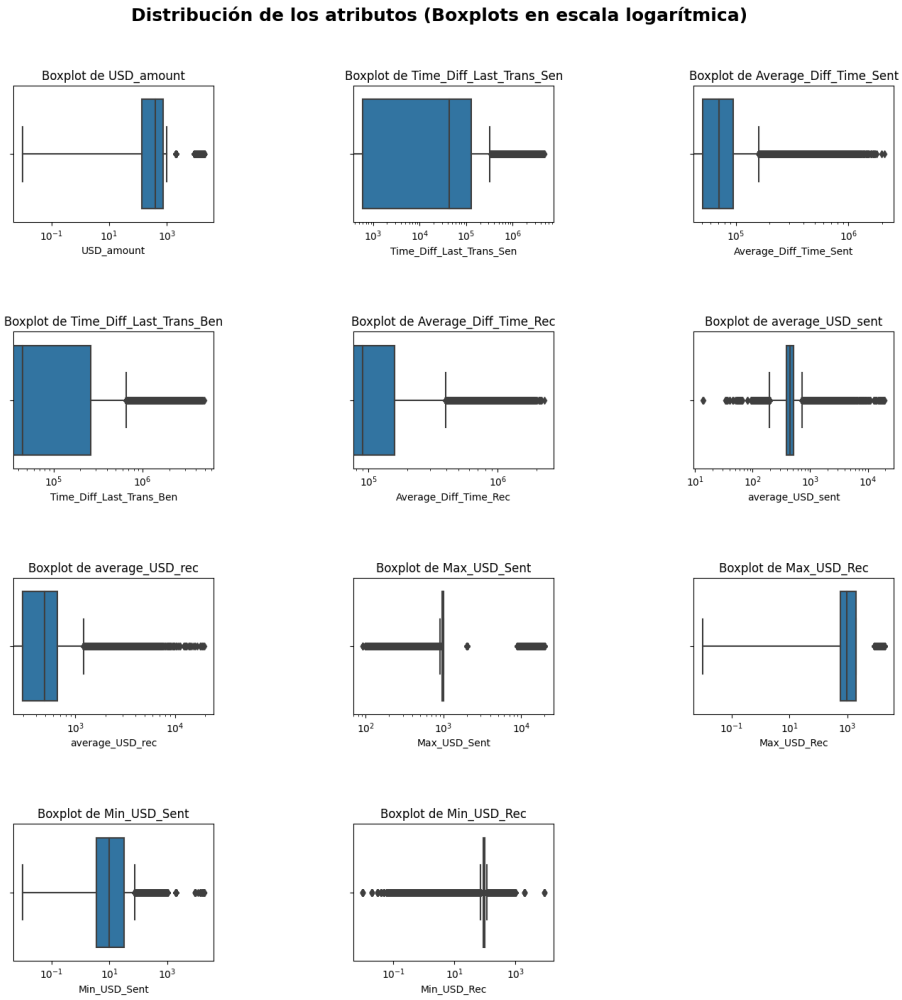


Figura 26: Distribuciones de las columnas

A continuación se muestran las correlaciones entre las columnas:

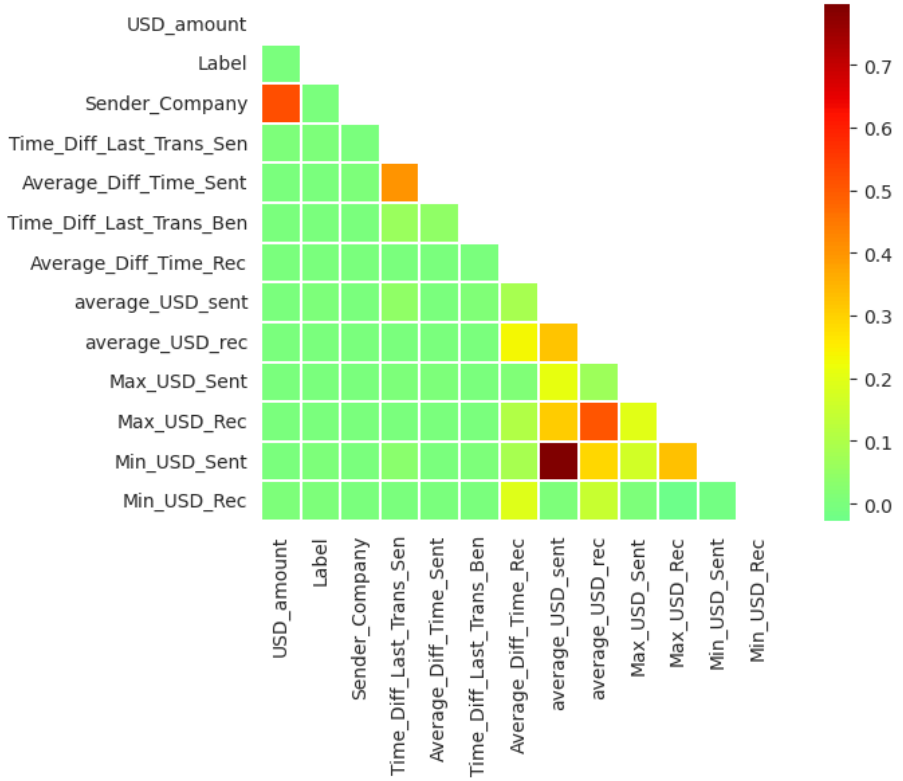


Figura 27: Distribuciones de las columnas

Como se puede observar la columna *Min_USD_Sent* tiene una alta correlación con la columna *average_USD_Sent*. Se ha decidido quitarse debido a esta razón, para poder reducir el número de columnas y facilitar el entrenamiento del modelo. También se ha decidido quitar *Bene_ClienJP* y *Sen_ClienJP* que está última aporta la misma información que *Sen_lob*. Se retira porque la información que aportan estos atributos se encuentran en *Sender_Id* y *Bene_Id*, y una vez se use One-Hot Encoding está información estará en las columnas que genere.

4.6. Análisis de los resultados

Para este apartado se van a usar las mismas métricas usadas para la evaluación del modelo creado con los datos de las cuentas de Ethereum, explicadas en el apartado *Métricas de evaluación de los modelos*. Los datos finales para la creación del modelo han sido discretizados y luego se han convertido en valores binarios con One-Hot Encoding están compuestos por 1.498.154 filas y 162 columnas, ocupando 1.8 Gigabytes de memoria. Para reducir la memoria ocupada se pasan a variables booleanas, ya que al estar en valores binarios es posible, resultando en 231 Megabytes.

Una vez discretizados se usa la técnica de sobre muestreo, con esta se consigue tener las datos balanceados. También se probó con la ténica *SMOTE* pero los resultados eran peores en general, no se ha querido usar también por el hecho de que crear datos sintéticos de otros que ya son sintéticos a lo mejor no es la mejor opción.

4.6.1. Resultados de los experimentos

Los resultados obtenidos son los siguientes

Algoritmo	Clase	Precisión	Recall	F1-score	Media Macro Precisión	Media Macro Recall	Media Macro F1-Score
	0	0.99	0.59	0.74			
Logistic Regression	1	0.04	0.73	0.07	0.51	0.66	0.40
	0	0.98	0.89	0.93			
KNN (Neighbour=4)	1	0.05	0.27	0.08	0.52	0.58	0.51
	0	0.98	0.93	0.95			
Random Forest (Estimadores=50)	1	0.06	0.20	0.09	0.52	0.57	0.52
	0	1.00	0.21	0.34			
Naives Bayes (Bernouilli)	1	0.03	0.98	0.05	0.51	0.59	0.20
	0	1.00	0.24	0.39			
Arbol de decisión (Max Depth=5)	1	0.03	1.00	0.05	0.51	0.62	0.22

Tabla 11: Resultados de los experimentos de los datos de J.P. Morgan.

Como se puede observar claramente, los resultados obtenidos con estos algoritmos no son satisfactorios. Hay que destacar que si la métrica a utilizar fuera la exactitud los resultados serían buenos en algunos algoritmos, por ejemplo en KNN y Random Forest, ya que estos si que han aprendido a clasificar como no fraudulentas las transacciones que no lo son. Pero principalmente sería debido a la estructura del set de datos de test, que tiene la misma proporción que el conjunto de datos original, un 2,3 % de transacciones fraudulentas.

Los algoritmos aprenden de manera fiable cuando una transacción no es fraudulenta, esto se puede comprobar observando la métrica de precisión de la clase 0. Uno de los problemas encontrados, es que observando la métrica recall para la clase 1 por ejemplo con los algoritmos de Naives Bayes o el árbol de decisión, encontramos valores muy altos, 0.98 y 1 respectivamente. El significado de esto es que detectan correctamente el 98 % y 100 % respectivamente de las transferencias que son fraude. Esto parece bueno, pero lo que está haciendo el modelo es marcar como fraude a la gran mayoría de las transacciones, por eso

hay un 0.03 de precisión en ambos. Así que la estrategia del modelo se basa en marcar una gran mayoría de las transacciones como fraudulentas cuando realmente solo una pequeña parte lo son. Por eso cuando el modelo marca una transacción como fraudulenta lo hace bien, y por esta misma razón el recall es más bajo para la clase 0, porque al resto de las transacciones no fraudulentas las marca como fraudulentas.

El árbol de decisión obtenido es el siguiente, la relevancia de los atributos se muestran en la tabla 12, destacar que solo se muestran los 6 atributos con más peso.

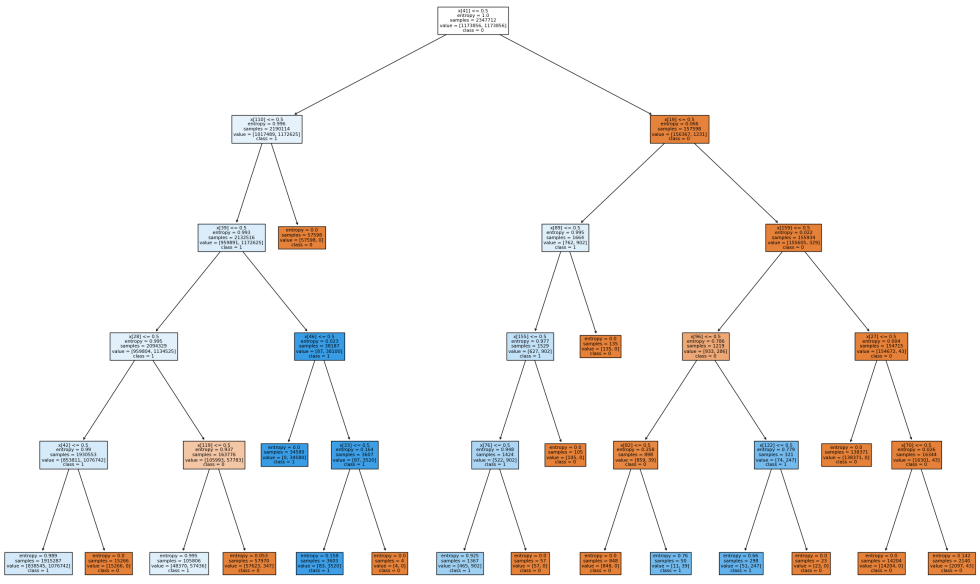


Figura 28: Árbol de decisión de obtenido con profundidad igual a 5.

Característica	Importancia
USD_amount_NORMAL	0.452094
Max_USD_Sent_MUCHISIMO	0.189405
Max_USD_Rec_POCO	0.131362
USD_amount_MUY ALTA	0.094679
Bene_Country_CANADA	0.053835
USD_amount_POCO	0.052683

Tabla 12: Importancia de las características en el árbol de decisión.

Las características en la Tabla 12 son las más importantes para el árbol de decisión. La característica *USD_amount_NORMAL* sugiere que la cantidad de dinero en una transacción es fundamental. *Max_USD_Sent_MUCHISIMO* y *Max_USD_Rec_POCO* indican que los montos enviados y recibidos son también significativos. *Bene_Country_CANADA* implica que el país destino de las transacciones es relevante. Estos atributos sugieren que el modelo da especial atención a los montos involucrados y a los destinos geográficos en sus decisiones. Es interesante observar como no incluye muchos de los atributos creados para aportar información al modelo, sobre todo los relacionados con el tiempo de las transferencias, pudiendo significar que el enfoque a la hora de incluirlos ha sido erróneo.

Como se puede observar en la tabla 14, los resultados obtenidos con XGBoost tampoco han sido mejores que los mostrados en la tabla 11.

Algoritmo	Clase	Precisión	Recall	F1-score	Media Macro Precisión	Media Macro Recall	Media Macro F1-Score
'learning_rate': 0.12, 'max_depth': 3, 'n_estimators': 400, 'subsample': 0.5, 'binary:logistic' , 'eval_metric'='auc'	0	0.99	0.62	0.76	0.51	0.68	0.41
	1	0.04	0.75	0.07			
'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 700, 'subsample': 0.2, 'binary:logistic' , 'eval_metric'='aucpr'	0	0.98	0.72	0.83	0.51	0.66	0.46
	1	0.04	0.61	0.08			
'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 700, 'subsample': 0.2, 'binary:logistic' , 'eval_metric'='aucpr'	0	0.98	0.88	0.92	0.51	0.60	0.51
	1	0.05	0.31	0.09			
'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 800, 'subsample': 0.5, 'binary:logistic' , 'eval_metric'='aucpr'	0	0.98	0.89	0.93	0.51	0.59	0.51
	1	0.05	0.28	0.09			
'learning_rate': 0.25, 'max_depth': 9, 'n_estimators': 800, 'subsample': 0.2, 'binary:hinge' , 'eval_metric'='aucpr'	0	0.98	0.71	0.82	0.51	0.65	0.45
	1	0.04	0.61	0.07			
'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 800, 'subsample': 0.5, 'binary:hinge' , 'eval_metric'='aucpr'	0	0.98	0.71	0.82	0.51	0.65	0.45
	1	0.04	0.61	0.07			

Tabla 13: Resultados de los experimentos de XGBoost de los datos de J.P. Morgan.

El tiempo de entrenamiento de estos algoritmos ha sido mucho mayor comparado con el tiempo de entrenamiento del modelo de Ethereum. Por está razón no se ha incluido un resultado de Support Vector Machines ya que era demasiado tiempo el entrenamiento. Esta razón también ha hecho que el *Grid Search* para la optimización de parámetros de XGBoost haya sido mucho más complicado y costoso, por lo que no se han podido probar todas las configuraciones que

se pretendían. Cabe destacar que también se ha probado a usar el método de validación cruzada, pero como no se encontraban resultados mejores y el tiempo de entrenamiento era mayor se ha decidido no incluirse.

Por último, para intentar mejorar los resultados, se ha probado usar métodos de ensamble, exactamente un clasificador por votación. El tiempo de entranamiento era mayor que la suma de los entrenamientos de cada algoritmo por separado, y al tampoco tener resultados medio aceptables no se esperaban mejorías significativas. Se ha querido incluir XGBoost con KNN pero el tiempo de entrenamiento era demasiado grande. En verde se muestra el mejor resultado obtenido de todos los experimentos con este conjunto de datos.

Algoritmos		Clase	Precisión	Recall	F1-score	Media Macro Precisión	Media Macro Recall	Media Macro F1-Score
Modelo 1	XGBoost	0	0.98	0.94	0.96	0.52	0.57	0.53
	Random Forest	1	0.06	0.19	0.10			
Modelo 2	XGBoost	0	0.98	0.91	0.95	0.52	0.60	0.53
	Logistic Regression	1	0.06	0.28	0.11			

Tabla 14: Resultados de los experimentos con ensamble de los datos de J.P. Morgan.

4.6.2. Interpretación del modelo de detección de fraude en transacciones bancarias

El modelo de ensamble, compuesto por Logistic Regression y XGBoost utilizando un Voting Classifier, ha mostrado resultados no muy satisfactorios en la detección de transacciones bancarias fraudulentas, es similar al compuesto en vez de Logistic Regression con Random Forest, pero con mejor average recall y mejores resultados en la clase 1, es decir, las transacciones fraudulentas. A continuación se presentan los detalles de su rendimiento.

- **Precisión para transacciones no fraudulentas: 98 %**. Esto indica que el 98 % de las transacciones que el modelo predijo como no fraudulentas eran efectivamente no fraudulentas. Es esencial mantener una alta precisión en este aspecto para evitar la detección de falsos positivos y no alertar a los clientes innecesariamente.

- **Recall (Sensibilidad) para transacciones no fraudulentas: 91 %**. Esto indica que el modelo identificó correctamente el 91 % de todas las transacciones no fraudulentas reales. Aunque es un buen valor, un recall más elevado sería ideal para minimizar los falsos positivos.

- **Puntuación F1 para transacciones no fraudulentas: 95 %**. Esta es una medida de la precisión del modelo en la predicción de transacciones no fraudulentas que considera tanto la precisión como la sensibilidad.

- **Precisión para transacciones fraudulentas: 6 %**. Esto sugiere que solo el 6 % de las transacciones que el modelo predijo como fraudulentas eran realmente fraudulentas. Esta precisión es bastante baja, lo que podría generar un número considerable de falsos positivos.

- **Recall (Sensibilidad) para transacciones fraudulentas: 28 %**. Esto indica que el modelo identificó correctamente solo el 28 % de todas las transacciones fraudulentas reales. Es un valor bajo, lo que significa que hay un alto número de transacciones fraudulentas que el modelo no está detectando.

- **Puntuación F1 para transacciones fraudulentas: 11 %**. Con una puntuación F1 tan baja, el modelo necesita mejoras significativas en la detección de transacciones fraudulentas para alcanzar un equilibrio adecuado entre precisión y recall.

- **Exactitud: 90 %**. Esta medida indica la proporción total de predicciones correctas (tanto fraudulentas como no fraudulentas) realizadas por el modelo. Aunque la exactitud parece alta, es importante tener en cuenta que esto está influenciado por el desequilibrio de clases en el conjunto de datos.
- **Precisión media macro: 52 %**. Este es el promedio de la precisión para transacciones no fraudulentas y fraudulentas.
- **Recall (Sensibilidad) media macro: 60 %**. Este es el promedio de la sensibilidad para transacciones no fraudulentas y fraudulentas.
- **Puntuación F1 media macro: 53 %**. Este es el promedio de la puntuación F1 para transacciones no fraudulentas y fraudulentas.

Dado que la precisión y el recall para las transacciones fraudulentas son bajos, sería recomendable trabajar en la mejora del modelo, posiblemente a través de técnicas de re-muestreo, explorando otros algoritmos o con otros atributos sacados del conjunto de datos inicial, para mejorar su capacidad de detectar transacciones fraudulentas de manera efectiva.

5. Diseño de la implementación modelo Ethereum

Para implementar un modelo de aprendizaje automático para la detección de cuentas fraudulentas en Ethereum requiere la interacción entre la red de blockchain y un servidor externo donde se ejecute el modelo. En esta sección, describimos los componentes clave y el flujo general de datos en este sistema.

5.1. Interacción con el modelo de aprendizaje automático

Dado que ejecutar un modelo de aprendizaje automático directamente en la blockchain sería impráctico y costoso en términos de gas, el modelo debe residir en un servidor externo. La comunicación entre el smart contract en Ethereum y el modelo se manejará mediante un oráculo.

5.2. Recopilación de datos de entrada

El smart contract necesita recopilar varios datos de entrada. Estos datos pueden ser recopilados de eventos y transacciones en la blockchain. Es importante optimizar la recopilación de datos para minimizar los costos de gas. Además de recopilar datos directamente de la blockchain, también se pueden utilizar APIs para obtener información adicional o histórica. Exactamente las usadas para obtener los datos usados para generar el modelo. Estas APIs pueden ser particularmente útiles para recopilar datos históricos que pueden no ser prácticos de obtener directamente a través de la blockchain. Por ejemplo, se usaría la API de Etherscan para recopilar un historial de transacciones de una cuenta y después calcular los atributos finales que necesita el modelo.

Es importante tener en cuenta que al utilizar APIs externas para recopilar datos, la interacción entre estas APIs y el smart contract también deberá ser manejada por un oráculo, similar a cómo se comunica el smart contract con el modelo de aprendizaje automático.

5.3. Uso de un oráculo

Para la comunicación entre el smart contract y el modelo de aprendizaje automático, se utiliza un oráculo, que basicamente es un puente de datos entre

blockchains y fuentes de datos externas. El smart contract envía una solicitud al oráculo con los datos de entrada. Luego, el oráculo pasa estos datos al modelo y, finalmente, envía la respuesta del modelo (si la cuenta es fraudulenta o no) de vuelta al smart contract.

5.4. Descripción del Smart Contract

El smart contract en la blockchain de Ethereum cumple un rol vital en la implementación del modelo de detección de cuentas fraudulentas. A continuación, se detallan sus funciones:

- **Recopilación de datos**: El smart contract recopila información sobre cuentas y transacciones en Ethereum. Los datos relevantes incluyen elementos como el tiempo entre transacciones, el número de transacciones realizadas y más, que son necesarios como entrada para el modelo de detección de fraudes.
- **Comunicación con el modelo externo**: Dado que el smart contract no puede ejecutar modelos de aprendizaje automático, utiliza un oráculo para interactuar con el modelo externo. El oráculo envía datos al modelo y recibe la predicción sobre si una cuenta puede ser categorizada como fraudulenta o no.
- **Interpretación de los resultados del modelo**: El smart contract procesa la información recibida del modelo a través del oráculo. Si el modelo identifica una cuenta como potencialmente fraudulenta, el smart contract registra esta información en la blockchain.
- **Administración de recursos**: Es importante que el smart contract esté optimizado en términos de consumo de gas para garantizar que las opera-

ciones sean económicamente viables en la blockchain de Ethereum.

El smart contract, por lo tanto, sirve como un componente crucial que facilita la recopilación de datos, la interacción con el modelo de detección de fraudes y el registro de información en la blockchain.

5.5. Calidad y Mantenimiento del Sistema

Es fundamental realizar pruebas exhaustivas y asegurar la seguridad del smart contract, especialmente cuando se trata de etiquetar cuentas como fraudulentas que pueden tener consecuencias significativas.

El modelo de aprendizaje automático puede requerir actualizaciones con el tiempo. Es importante tener una estrategia para manejar estas actualizaciones, como asegurarse de que el oráculo este funcionando de manera correcta.

5.6. Casos de uso

En el desarrollo de software, los casos de uso son una herramienta ampliamente utilizada para describir las interacciones y funcionalidades de un sistema desde la perspectiva de los usuarios o actores involucrados. Proporcionan una forma efectiva de capturar los requisitos del sistema y ofrecen una comprensión clara de cómo los usuarios interactúan con él.

En las siguiente tablas se presentan los casos de uso identificados para el sistema, donde se detallan los diferentes elementos que nos ayudan a comprender y documentar cada caso de uso en detalle.

Caso de Uso CU-01: Recopilación de datos de la blockchain	
Código	CU-01
Actores	Smart Contract
Precondiciones	El Smart Contract está desplegado en la blockchain de Ethereum.
Postcondiciones	Los datos son recopilados y listos para ser enviados al modelo de aprendizaje automático.
Escenario Principal	1. El Smart Contract se activa. 2. Recopila datos de transacciones. 3. Almacena los datos temporalmente.

Tabla 15: Caso de uso CU-01: Recopilación de datos de la blockchain

Caso de Uso CU-02: Recopilación de datos mediante APIs externas	
Código	CU-02
Actores	Smart Contract, Oráculo
Precondiciones	El Smart Contract ha identificado la necesidad de datos adicionales o históricos.
Postcondiciones	Los datos son recopilados y almacenados temporalmente junto con los datos de la blockchain.
Escenario Principal	1. El Smart Contract identifica la necesidad de datos adicionales o históricos. 2. Un oráculo facilita la comunicación entre el Smart Contract y las APIs externas, como Etherscan. 3. Los datos son recopilados a través de la API y pasados al Smart Contract por el oráculo. 4. Los datos se almacenan temporalmente junto con los datos de la blockchain.

Tabla 16: Caso de uso CU-02: Recopilación de datos mediante APIs externas

Caso de Uso CU-03: Comunicación con el modelo de aprendizaje automático	
Código	CU-03
Actores	Smart Contract, Oráculo
Precondiciones	El Smart Contract ha recopilado los datos. El modelo de aprendizaje automático está en el servidor externo.
Postcondiciones	El Smart Contract recibe la respuesta del modelo a través del oráculo.
Escenario Principal	1. El Smart Contract envía los datos al oráculo. 2. El oráculo envía los datos al modelo. 3. El modelo procesa los datos y genera una respuesta. 4. El oráculo devuelve la respuesta al Smart Contract.

Tabla 17: Caso de uso CU-03: Comunicación con el modelo de aprendizaje automático

[Caso de uso CU-04]Caso de Uso CU-04: Registro de cuentas fraudulentas en la blockchain	
Código	CU-04
Actores	Smart Contract y Usuario
Precondiciones	El Smart Contract ha recibido la predicción del modelo.
Postcondiciones	La información de las cuentas fraudulentas es registrada en la blockchain.
Escenario Principal	1. El Smart Contract recibe la predicción del modelo. 2. Si la predicción indica que una cuenta es fraudulenta, el Smart Contract registra esta información en la blockchain y notifica al usuario.

Tabla 18: Caso de uso CU-04: Registro de cuentas fraudulentas en la blockchain

Caso de Uso CU-05: Actualización del modelo de aprendizaje automático	
Código	CU-05
Actores	Administrador del sistema
Precondiciones	El modelo de aprendizaje automático está desplegado en un servidor externo.
Postcondiciones	El modelo de aprendizaje automático está actualizado.
Escenario Principal	1. El administrador identifica la necesidad de actualizar el modelo. 2. El administrador realiza ajustes o entrena un nuevo modelo. 3. Se verifica que el oráculo esté funcionando correctamente. 4. El modelo actualizado se implementa en el servidor externo.

Tabla 19: Caso de uso CU-05: Actualización del modelo de aprendizaje automático

Caso de Uso CU-06: Análisis de una dirección de Ethereum proporcionada por el usuario	
Código	CU-06
Actores	Usuario, Smart Contract
Precondiciones	El usuario tiene una dirección de Ethereum válida que desea analizar. El sistema está en funcionamiento y puede acceder a la blockchain de Ethereum.
Postcondiciones	El sistema proporciona el resultado del análisis de la dirección proporcionada.
Escenario Principal	1. El usuario inicia el sistema e ingresa una dirección de Ethereum. 2. El sistema valida la dirección, recopila datos mediante smart contracts y/o APIs, y los envía al modelo de aprendizaje automático. 3. El modelo analiza los datos y el sistema muestra los resultados al usuario.

Tabla 20: Caso de uso CU-06: Análisis de una dirección de Ethereum proporcionada por el usuario

5.7. Requisitos

Los requisitos son especificaciones que el sistema debe cumplir para satisfacer las necesidades de los usuarios. Se dividen en requisitos funcionales, que describen las funcionalidades que el sistema debe proporcionar, y requisitos no

funcionales, que establecen criterios para evaluar el funcionamiento del sistema.

5.7.1. Requisitos Funcionales (RF)

A continuación se muestran los requisitos funcionales derivados de los casos uso:

Requisito Funcional RF-01			
Código:	RF-01	Fuente:	CU-01, CU-02
Nombre:	Recopilación de datos de la blockchain		
Descripción:	El sistema debe ser capaz de recopilar datos de la blockchain.		
Necesidad:	Esencial para el análisis de datos	Prioridad:	Alta

Tabla 21: Requisito Funcional (RF-01)

Requisito Funcional RF-02			
Código	RF-02	Fuente	CU-02
Nombre	Comunicación con APIs externas		
Descripción	El sistema debe comunicarse con APIs para recopilar datos.		
Necesidad:	Obtener datos adicionales para el análisis	Prioridad:	Media

Tabla 22: Requisito Funcional (RF-02)

Requisito Funcional RF-03			
Código	RF-03	Fuente	CU-03
Nombre	Envío de datos a modelo de aprendizaje automático		
Descripción	El sistema debe enviar los datos al modelo de aprendizaje automático para el análisis.		
Necesidad:	Realizar análisis de los datos recopilados	Prioridad:	Alta

Tabla 23: Requisito Funcional (RF-03)

Requisito Funcional RF-06			
Código:	RF-06	Fuente:	CU-01, CU-02, CU-03
Nombre:	Análisis de dirección de Ethereum proporcionada por el usuario		
Descripción:	El sistema debe permitir al usuario ingresar una dirección de Ethereum, validarla, recopilar datos relacionados, analizarlos mediante un modelo de aprendizaje automático y mostrar los resultados.		
Necesidad:	Proporcionar análisis sobre direcciones de Ethereum a los usuarios	Prioridad:	Alta

Tabla 26: Requisito funcional (RF-06)

Requisito Funcional RF-04			
Código:	RF-04	Fuente:	CU-04
Nombre:	Registro de cuentas fraudulentas		
Descripción:	Registrar la información de cuentas fraudulentas en blockchain.		
Necesidad:	Almacenar información de cuentas fraudulentas	Prioridad:	Alta

Tabla 24: Requisito Funcional (RF-04)

Requisito Funcional RF-05			
Código:	RF-05	Fuente:	CU-05, CU-06
Nombre:	Detección de cuentas fraudulentas		
Descripción:	El modelo de aprendizaje automático debe de detectar cuentas fraudulentas.		
Necesidad:	Identificar cuentas fraudulentas para su registro	Prioridad:	Alta

Tabla 25: Requisito Funcional (RF-05)

5.7.2. Requisitos No Funcionales (RNF)

A continuación se muestran los requisitos funcionales derivados de los casos uso:

Requisito No Funcional RNF-01			
Código:	RNF-01	Fuente:	CU-03, CU-04
Nombre:	Estabilidad del sistema		
Descripción:	El sistema debe ser estable y resistente a fallos.		
Necesidad:	Evitar pérdida de datos y tiempo de inactividad	Prioridad:	Alta

Tabla 27: Requisito No Funcional (RNF-01)

Requisito No Funcional RNF-02			
Código:	RNF-02	Fuente:	CU-01, CU-02
Nombre:	Escalabilidad del sistema		
Descripción:	El sistema debe manejar un creciente volumen de solicitudes sin degradar su rendimiento significativa.		
Necesidad:	Acomodar el crecimiento y demanda del sistema	Prioridad:	Alta

Tabla 28: Requisito No Funcional (RNF-02)

Requisito No Funcional RNF-03			
Código:	RNF-03	Fuente	CU-03
Nombre:	Tiempo de respuesta		
Descripción:	El sistema debe procesar las solicitudes en un tiempo razonable.		
Necesidad:	Asegurar una experiencia de usuario adecuada	Prioridad:	Media

Tabla 29: Requisito No Funcional (RNF-03)

6. Gestión del Proyecto

Este proyecto se centró en el desarrollo de sistemas de detección de fraude mediante aprendizaje automático, específicamente enfocándose en la detección de cuentas fraudulentas de Ethereum y detección de transacciones fraudulentas. Los pasos clave en la gestión de este proyecto fueron:

1. **Definición de objetivos**: Establecimiento claro de los objetivos del proyecto, centrándose en la detección de fraudes con aprendizaje automático.
2. **Revisión bibliográfica y estado del arte**: Revisión de literatura existente y estado del arte en sistemas de detección de fraude.
3. **Adquisición de datos**: Identificación y recolección de conjuntos de datos relevantes de transacciones y cuentas Ethereum para su uso en el entrenamiento y validación de los modelos de aprendizaje automático.
4. **Preprocesamiento de datos**: Limpieza y preprocesamiento de los datos para garantizar que estén en un formato adecuado para el entrenamiento de modelos.
5. **Selección de características**: Identificación y selección de características relevantes que podrían ser indicadores de comportamiento fraudulento.
6. **Desarrollo del modelo para cuentas Ethereum**: Desarrollo de un modelo de aprendizaje automático para la detección de cuentas fraudulentas de Ethereum. Esto incluye la selección del algoritmo, el entrenamiento del modelo y la validación del mismo.
7. **Evaluación del modelo para cuentas Ethereum**: Evaluación de la eficacia del modelo desarrollado utilizando métricas adecuadas como precisión, recall y F1-score.

8. **Desarrollo del modelo para transacciones fraudulentas**: Tras finalizar el modelo de cuentas, se inició el desarrollo de un modelo para la detección de transacciones fraudulentas, siguiendo un proceso similar al del modelo de cuentas Ethereum.

9. **Evaluación del modelo para transacciones fraudulentas**: Evaluación de la eficacia del segundo modelo desarrollado utilizando métricas similares.

10. **Comparación y análisis**: Comparación de los resultados de ambos modelos y análisis de sus diferencias y similitudes.

11. **Documentación y reporte**: Documentación detallada del proceso, metodología, resultados y conclusiones del proyecto.

Cada uno de estos pasos fue crítico para garantizar que el proyecto se completara de manera efectiva y completa.

6.1. Marco Regulador

Es fundamental reconocer y cumplir con las regulaciones legales y normativas al trabajar con datos financieros personales, especialmente cuando se trata de datos provenientes de entidades bancarias como JPMorgan. Este proyecto ha tenido en cuenta las siguientes consideraciones regulatorias:

1. **Anonimización de datos**: Dado que este proyecto involucra el uso de datos financieros personales, es necesaro asegurar que todos los datos utilizados estén anonimizados para proteger la privacidad de los individuos. En la Unión Europea, el Reglamento General de Protección de Datos establece directrices estrictas sobre cómo se deben manejar los datos personales.

De acuerdo con el Artículo 5 del GDPR, los datos personales deben ser procesados de manera que garantice la protección de los derechos y libertades de las personas, incluido el aseguramiento de que los datos estén adecuadamente anonimizados o pseudonimizados [28].

2. **Confidencialidad y posesión de los datos**: Los datos son confidenciales y deben ser tratados con medidas similares a las que se aplicarían a cualquier información confidencial. Es importante reconocer que el título y la plena propiedad de los datos están reservados para J.P. Morgan y permanecerán bajo su posesión. Además, todos los derechos de propiedad y derechos de propiedad intelectual de cualquier índole son y seguirán siendo propiedad de J.P. Morgan.

3. **No afiliación con JPMorgan**: Es importante aclarar que si bien los datos utilizados en este proyecto pertenecen a JPMorgan, cualquier resultado, análisis, o conclusión obtenida no tiene ninguna afiliación con JPMorgan y no refleja necesariamente las perspectivas o posiciones de la entidad. Los datos han sido utilizados únicamente con fines académicos y de investigación.

6.2. Planificación y presupuesto del proyecto

En este apartado se muestra la planificación del proyecto.

Descripción	Duración Estimada (Semanas)	Duración Real(Semanas)
Definición del problema	1	1
Obtención de los datos	1	2
Análisis estudios previos	2	2
Análisis datos de Ethereum	6	6
Desarrollo modelo Ethereum	3	2
Análisis de datos J.P. Morgan	8	10
Desarrollo modelo J.P. Morgan	4	6
Estudio resultados	1	1
Documentación	10	11
Presentación	2	2
Total	38	43

Tabla 30: Planificación del proyecto en semanas. (Estimada vs Real)

En la tabla 30 se muestra en la siguiente tabla la estimación inicial. Como se puede comprobar se tardó mas de lo esperado, sobre todo con el análisis y el desarrollo del modelo de J.P. Morgan.

A continuación se muestra el presupuesto del proyecto, considerando también el uso de mejores ordenadores que el usado.

6.2.1. Presupuesto para recursos humamos

Los recursos humanos constan de las personas necesarias para la realización de este proyecto. En este proyecto se ha considerado que se necesita un científico de datos y un jefe de proyecto como supervisor para que se desarrolle de manera correcta. En la tabla 31 se muestra el presupuesto para recursos humanos:

Puesto	Horas estimadas	Coste (€/hora)	Coste Total (€)	Coste SS(€)
Desarrollador	1.368	15	20.520	26.676
Jefe de proyecto	152	35	5.320	6.916
Total	1.520		25.840	33.592

Tabla 31: Estimación del coste de los recursos humanos.

Categoría	Valor(€)
Presupuesto para recursos humamos	33.592
Presupuesto de los recursos de hardware	713,9
Presupuesto Total	34.305,9

Tabla 33: Presupuesto total del proyecto.

A estos salarios hay que añadirles un 30 % para cubrir los gastos de las contribuciones a la seguridad social (SS), como se muestra en la última columna.

6.2.2. Recursos de hardware y software

Para el desarrollo del proyecto hay que incluir los costes de los ordenadores o software en caso de cuesten dinero.

Puesto	Coste	Vida Útil	Uso	Coste Proyecto(€)
Ordenador	1999	28 (Meses)	10	713.9
Total	152	35	5320	713.9

Tabla 32: Presupuesto de los recursos de hardware

Como se muestra en la tabla 32 no hay necesidad de ningún tipo de software que cueste dinero ya que todo el software usado es *Open Source*. Pero cabe destacar que se precisa de un buen ordenador con gran capacidad de procesamiento para agilizar el proceso de entrenamiento.

6.2.3. Presupuesto final

En la tabla 35 se muestra el presupuesto total estimado para la realización del proyecto.

6.2.4. Presupuesto Real

Dado que las horas estimadas no fueron como las reales, se procede a ajustar el presupuesto total añadiendo las horas de más.

Puesto	Horas estimadas	Coste (€/hora)	Coste Total (€)	Coste SS(€)
Desarrollador	1.568	15	23.520	30.576
Jefe de proyecto	152	35	5.320	6.916
Total	1.720		28.840	37.492

Tabla 34: Estimación del coste de los recursos humanos.

Con los datos de la tabla 34 se proceden a pasar los costes del proyecto con los tiempos reales.

Categoría	Valor(€)
Presupuesto para recursos humamos	37.492
Presupuesto de los recursos de hardware	713,9
Presupuesto Total	38.205,9

Tabla 35: Presupuesto total del proyecto.

6.3. Entorno Socieconómico

El fraude en transacciones financieras y el fraude en las cuentas de Ethereum son problemas graves que afectan tanto a individuos como a organizaciones a nivel global. La aplicación de sistemas de detección de fraude basados en aprendizaje automático puede tener un impacto socioeconómico significativo en varios frentes.

En una encuesta de la consultora PricewaterhouseCoopers (PwC)[29] sobre el fraude, según las empresas entrevistadas, el 46 % de estas sufrió fraude de alguna manera. De las empresas entrevistadas que facturan mas de 10 mil millones

de dólares americanos el 52 % de estas sufrieron algún tipo de fraude. Como se observa hay mucho valor para las empresas para resolver este problema.

El impacto económico es el siguiente:

- **Reducción de pérdidas económicas:** Al detectar de forma eficaz y precisa el fraude, las instituciones financieras y los usuarios de Ethereum podrían evitar pérdidas económicas significativas. Esto podría generar ahorros directos que podrían invertirse en otras áreas productivas de la economía.
- **Aumento de la confianza**: Un sistema efectivo de detección de fraude podría aumentar la confianza del público en las transacciones digitales y en la criptomoneda, lo que podría resultar en un mayor uso y adopción de estas tecnologías, beneficiando al desarrollo económico.

Mientras que el impacto social es el siguiente:

- **Protección al usuario**: Los individuos que se benefician de un sistema de detección de fraude efectivo pueden evitar el estrés y la angustia asociados con ser víctimas de fraude.
- **Igualdad de oportunidades financieras**: Al reducir el fraude, se puede hacer que los servicios financieros y las criptomonedas sean más seguros y accesibles para un grupo demográfico más amplio, contribuyendo a la inclusión y equidad financiera.

7. Conclusiones

In this last part conclusions of the work done and about the results obtained are presented. En esta última sección se exponen las conclusiones del trabajo

realizado y de los resultados obtenidos.

7.1. Conclusiones generales

El desarrollo de este proyecto ha sido un desafío notable y puede considerarse un éxito. Un objetivo primordial era adquirir conocimiento y habilidades en técnicas de ciencia de datos tales como minería de datos y aprendizaje automático. Me siento confiado al afirmar que mi conocimiento y competencia en estas áreas han mejorado considerablemente.

Aunque la curva de aprendizaje fue pronunciada, la experiencia adquirida a través de este proceso ha demostrado ser de gran valor. Haber podido investigar el estado del arte en sistemas de detección de fraudes me permitió apreciar la diversidad y complejidad del dominio. Además, la experiencia en análisis de datos, preprocesamiento y construcción de modelos tanto para cuentas de Ethereum como para transacciones de J.P. Morgan, me ha proporcionado conocimientos prácticos sobre los desafíos y complejidades de la detección de fraudes.

Sin embargo, el proyecto no estuvo exento de desafíos. Los resultados obtenidos fueron algo mixtos, particularmente con el conjunto de datos de J.P. Morgan. Este, presentó varias dificultades imprevistas. Los tiempos de computación de los modelos fueron significativamente más largos de lo anticipado. Esto podría atribuirse al gran volumen o complejidad de los datos, o posiblemente a la selección de algoritmos que quizás no hayan sido los más adecuados para este conjunto de datos en particular. Además, las métricas de rendimiento del modelo no fueron tan altas como se podría haber esperado. Esto podría indicar la necesidad de una mayor optimización o experimentación con diferentes técnicas de modelado.

También es importante reconocer que los datos del mundo real, aunque en

este caso sean sintéticos, a menudo son desordenados e impredecibles, y a veces no cumplen con las expectativas.

Además, el ejercicio de comparar y analizar los resultados de los modelos desarrollados para Ethereum y J.P. Morgan fue una experiencia de aprendizaje crítica. Puso de relieve la importancia de comprender el contexto y de enfoques personalizados cuando se trabaja con diferentes tipos de datos y dominios.

Dicho todo esto, el sentimiento de orgullo y satisfacción está bien fundamentado. El proyecto no trataba solo de lograr la perfección en los resultados, sino también sobre el proceso de aprendizaje. Fue una oportunidad para desarrollar pensamiento crítico, habilidades de resolución de problemas y un entendimiento práctico de las técnicas de ciencia de datos. Estas son competencias que son inmensamente valiosas para mí y servirán como una sólida base para proyectos futuros.

Al reflexionar sobre el desarrollo de este proyecto, recuerdo la importancia del aprendizaje continuo. El campo de la ciencia de datos y la detección de fraudes está en constante cambio, y mantenerse actualizado con las últimas investigaciones es crítico. Las perspectivas y la experiencia que he obtenido de este proyecto sin duda serán útiles para una innovación más profunda en el futuro.

En conclusión, este proyecto representa un hito en mi trayectoria de aprendizaje. Los desafíos enfrentados y el conocimiento adquirido son muy importantes para mi crecimiento como científico de datos.

7.2. Conclusiones para los datos de Ethereum dataset

En este caso, los resultados obtenidos con el modelo generado con XGBoost pueden considerarse relativamente buenos. Todavía hay margen de mejora ya

que probablemente haya algo que se pueda hacer para mejorar el modelo, por ejemplo, utilizando técnicas de ensamble. Pero debido a que el enfoque principal del problema era el conjunto de datos de J.P. Morgan, el tiempo para el proyecto no se dedicó a tratar de mejorar los resultados obtenidos con los datos de Ethereum tanto como podría haber sido.

Dicho esto, se puede creer que este conjunto de datos no es muy realista porque no se ha actualizado en 2 años, y las condiciones en la cadena de bloques de Ethereum han cambiado, no solo el precio de ETH y las estadísticas de las cuentas. Entonces, probablemente si se fuera a usar el modelo, uno debería tener cuidado al confiar en esta clasificación de cuentas fraudulentas, ya que probablemente los resultados no sean tan confiables como uno espera.

Además, se pueden verificar en Kaggle diferentes resultados de diferentes proyectos obtenidos con este conjunto de datos. Algunos no deben tenerse en cuenta, ya que ni siquiera explican o muestran qué métrica están usando. Por ejemplo, un caso de estos trabajos obtiene mejores resultados con XGBoost, pero el trabajo previo de preprocesamiento de datos consiste en eliminar columnas altamente correlacionadas y no normaliza ni discretiza los datos.

El trabajo en Kaggle considerado más relevante debido al preprocesamiento de los datos ha obtenido mejores resultados con XGBoost, pero es interesante ver cómo con otros algoritmos los resultados obtenidos son peores en otros proyectos, por ejemplo, con la Regresión Logística obtuvimos un valor para el f1-score de 0.89 y con el trabajo con el que nos comparamos un 0.85. Pero con XGBoost obtuvimos un 0.942 y ellos un 0.97, después de probar su configuración exacta obtuvimos un 0.92 para el f1-score [30]. Una posible razón que se piensa para esto es la discretización de los datos, porque ellos no discretizan los datos. La discretización implica agrupar variables continuas en categorías discretas. Esto a veces puede llevar a la pérdida de información, ya que las variaciones

sutiles dentro de los grupos ya no se capturan. XGBoost, al ser un modelo más complejo, podría verse más afectado por esta pérdida de información que la Regresión Logística.

7.3. Conclusiones para los datos de J.P. Morgan

Los resultados obtenidos con este conjunto de datos no son tan buenos como se esperaba. En primer lugar, ha sido difícil trabajar con este conjunto de datos, principalmente debido a su tamaño y al poder de cómputo necesario para obtener los diferentes modelos. Por ejemplo, para la ejecución del algoritmo *KNN*, se necesitaron 8 horas y 38 minutos, todo para obtener resultados insatisfactorios. En algunos otros estudios, como se menciona en la parte de Estudios Previos, han utilizado técnicas de conjunto, esta es una técnica de alto consumo de cómputo y aunque se ha intentado, los resultados no han mejorado significativamente. Quizás con otras configuraciones o diferentes técnicas de conjunto, los resultados podrían haber mejorado. También se consideró el uso de redes neuronales, más aún, redes neuronales profundas, pero no era una opción factible en términos de cómputo. Esto se puede ver en [23], donde utilizan conjunto con diferentes modelos, incluso redes neuronales convolucionales, lo cual no se pudo considerar debido al tiempo de cómputo que habría llevado en este caso y con el ordenador disponible para el desarrollo del proyecto.

En segundo lugar, se cree firmemente que los datos no eran lo suficientemente completos como para poder proporcionar información significativa al algoritmo. Como se afirma en algunos otros estudios, los datos de transacciones también necesitan datos históricos de la cuenta. Las principales razones son las siguientes:

- **Comprensión contextual:** Los datos históricos de la cuenta proporcionan contexto para las transacciones. Por ejemplo, si una cuenta general-

mente realiza pequeños pagos y de repente hace una compra muy grande, esta desviación del comportamiento histórico podría indicar una actividad fraudulenta. Sin datos históricos, es difícil determinar qué constituye el comportamiento "normal"para una cuenta. O cuál es la cantidad de dinero en la cuenta, para comparar con la transacción.

- **Tendencias temporales:** Los patrones de fraude pueden evolucionar con el tiempo. Al combinar datos de transacciones con datos históricos de la cuenta, el modelo puede discernir tendencias temporales que podrían ser indicativas de un comportamiento regular o de actividades fraudulentas.
- **Agrupación de cuentas:** Similar a los sistemas de recomendación, los datos históricos se pueden usar para agrupar cuentas similares en función de su historial de transacciones. Al comprender el comportamiento de un grupo de cuentas similares, el modelo puede hacer predicciones más precisas para los individuos dentro de ese grupo.

Dicho esto, todavía hay margen de mejora para este conjunto de datos y quizás otro enfoque al problema podría ser mejor que el utilizado en este caso.

7.4. Trabajos futuros

Se necesita mucha más investigación para el desarrollo de sistemas de detección de fraudes. Para Ethereum y blockchain en general, hay una gran ventana de oportunidades, ya que al ser capaz de obtener información sobre direcciones y transacciones hay una ventaja para seguir trabajando en este problema. Para los sistemas de pagos de bancos también hay mucho trabajo por hacer, pero como se menciona en el Estado del Arte, probablemente hayan desarrollado sistemas internos con sus datos, los cuales no están disponibles para el público por razones legales y financieras. Además, una posibilidad para mejorar estos sistemas

es usar modelos preentrenados y personalizarlos para cada banco o sistema de pago. Amazon ha construido una API para esta tarea, pero para el desarrollo de este proyecto no se podría haber utilizado porque no tengo el permiso de J.P. Morgan Chase Bank para usarlo en el sistema de otra compañía.

Anexo

Introduction

In the new era of technology, the need to detect and prevent fraud has become increasingly important. Fraud in banking transactions has been a longstanding issue, and its detection is crucial to mitigate financial losses and protect individuals and entities. This work focuses on two types of fraud: the detection of fraudulent accounts in the Ethereum network and the detection of fraudulent transactions. By identifying fraudulent accounts and transactions before any funds are transferred, we can prevent financial losses and ensure the security of transactions.

Fraud detection is a constant challenge in various industries, including finance, insurance, and telecommunications. Machine learning techniques have proven to be effective in detecting fraudulent activities. However, fraudsters continually adapt and devise new strategies to evade detection systems, making fraud detection a complex and evolving problem. To address this challenge, innovative approaches and technologies are needed to stay ahead of fraudsters and protect financial systems.

Motivation

The motivation for this study is rooted in a deep interest in financial systems coupled with the potential of data science. By utilizing machine learning and artificial intelligence in fraud detection, there is an opportunity to bolster the security within financial systems. The digitalization and global integration of financial services have created opportunities for complex fraud schemes. Deve-

loping adaptable and robust fraud detection mechanisms is vital to keep ahead and secure transactions. Additionally, tackling fraud is not just about preventing financial loss; it is about building a trustworthy financial environment, which has far-reaching implications on a personal and societal level.

Objectives

To clearly outline the objectives, the following list is provided:

1. Study the state-of-the-art in fraud detection systems.
2. Perform an analysis and preprocessing of the Ethereum fraudulent account dataset.
3. Create a model using machine learning algorithms with the Ethereum fraudulent account dataset.
4. Conduct an analysis and preprocessing of the J.P. Morgan fraudulent transaction dataset.
5. Build a model using machine learning algorithms with the J.P. Morgan fraudulent transaction dataset.
6. Examine the results obtained from both models.
7. Define the implementation approach for the model in the Ethereum blockchain network.

Project Development

The development of the analysis of the data and the results of the model obtained with both data sets is presented in this section.

Model Evaluation Metrics

To address the problem of detecting fraudulent accounts, it is important to focus on specific metrics:

- Recall (Sensitivity or True Positive Rate): This metric measures the proportion of actual fraudulent transactions that the model correctly identifies.
- Precision: Precision quantifies the proportion of predicted fraudulent transactions that are actually true positives.
- F1-Score: The F1-Score combines recall and precision into a single metric using the harmonic mean. It provides an overall measure of the model's ability to detect fraud while minimizing false alarms.

The choice of the primary metric depends on the relative importance of detecting all fraudulent accounts. In this case, the F1-Score is considered essential as the cost of missing fraudulent transactions can be significantly higher than falsely identifying legitimate accounts as fraud. However, if precision is relatively acceptable, focus can be placed on recall. If the recall of fraudulent accounts is high while precision is low, it suggests that the model is labeling a large number of legitimate accounts as fraudulent, making it unreliable.

Ethereum account data analysis

The dataset used in this project was obtained from Kaggle, utilizing Ethereum Etherscan API, etherscamdb API, and web scraping techniques. It consists of 9,840 entries and 50 attributes. Initially, the index column, which provides no relevant information, was removed. Following that, columns with zero variance

were eliminated, as they do not contribute any useful information to the model.

The distribution of the FLAG column, which indicates whether an account has fraudulent transactions, is as follows: 7,662 entries are marked as 0 (non-fraudulent), while the remaining 2,179 rows are considered fraudulent. To address bias in the model, techniques were employed to balance the fraudulent and non-fraudulent entries. It is worth noting that the majority of entries are non-fraudulent, which aligns with the nature of the problem.

Columns related to ERC20 data typically have fewer distinct values, as most accounts primarily use the Ethereum main network rather than the secondary ERC20 network. However, the total number of transactions sent and received by each account, whether through Ethereum or ERC20, were summed and added as two new columns: Total unique received and Total unique sent.

After removing duplicate rows, the dataset was reduced from 9,841 to 9,012 entries. Attributes with higher correlations are often ERC20 metrics, such as mean, maximum, or minimum values of transactions sent or received. Some Ether-related attributes, such as avg val received and max value received, also exhibit correlations, although around 0.6 compared to the near-perfect correlations of ERC20 values. Additionally, a strong correlation exists between attributes related to contracts, including the value sent to contracts and the number of contracts created. After this and after studying the distributions of the attributes, they have been discretized, for the model training it has been used the SMOTE technique to balance the classes.

Results obtained with the Ethereum data set

The results obtained are the following ones:

Algorithm	Class	Precision	Recall	F1-score	Macro Average Precisión	Macro Average Recall	Macro Average F1-Score
	0	0.99	0.94	0.97			
Logistic Regression	1	0.74	0.92	0.81	0.84	0.93	0.89
	0	0.98	0.94	0.96			
KNN (Neighbour=4)	1	0.69	0.88	0.78	0.84	0.91	0.87
	0	0.97	0.99	0.98			
Random Forest	1	0.92	0.79	0.85	0.94	0.89	0.91
	0	0.98	0.92	0.94			
Naives Bayes (Bernouilli)	1	0.62	0.85	0.72	0.80	0.89	0.83
	0	0.98	0.89	0.93			
Decision tree (Max Depth=4)	1	0.55	0.87	0.80	0.76	0.88	0.80
	0	0.98	0.98	0.98			
Support Vector Machine	1	0.88	0.86	0.87	0.93	0.92	0.92
	0	0.97	0.99	0.98			
XGBoost	1	0.92	0.84	0.88	0.95	0.92	0.93

Tabla 36: Results obtained with the Ethereum data set.

The best result has been obtained with XGBoost, parameter optimization have been done to find the best configuration. The results obtained with XGBoost are highly satisfactory, with the model demonstrating effective performance in detecting fraudulent accounts. It achieves a sensitivity of 84.5 % in identifying fraudulent accounts, and when classifying an account as fraudulent, it is accurate 95 % of the time. The precision for non-fraudulent accounts is 97.5 %, while the precision for fraudulent accounts is 92.4 %. These metrics indicate the model's ability to correctly classify both types of accounts. Overall, the model exhibits an f1-score of 93 %.

Transactional data analysis

These data contain a wide variety of transaction types representing normal activities as well as abnormal/fraudulent activities introduced with predefined probabilities. The data was generated by executing an AI planning-execution simulator and translating the output planning traces into tabular format. Initially, the dataset consists of 13 attributes and 1,498,178 rows. The dataset is considerably imbalanced, with 97.94 % of transactions labeled as not fraud and 2.06 % labeled as fraud. The majority of transactions consist of normal payments, followed by fast payments, fund movements, and the least frequent type being salary transfers. This distribution can be attributed to individuals receiving monthly salaries but conducting various types of movements and payments throughout the month. Regarding the countries of origin and destination for transfers, it is confirmed that the majority of transfers originate from or are directed to accounts in the United States, which is expected as it is a U.S.-based bank. For the purpose of later using the One-Hot Encoding technique, most countries have been grouped by continents, except for the United States, Germany, and Canada, as they have the highest number of transfers and can provide valuable information. This approach avoids creating a new column for each country, making the model computation more efficient.

Feature Engineering

The feauteres obtained from the originals are the following ones:

- **Bene ClienteJP:** Binary variable indicating if the beneficiary is a client of J.P. Morgan Chase.
- **Sen ClienteJP:** Binary variable indicating if the sender is a client of J.P.

Morgan Chase.

- **National Transaction:** Binary variable indicating if the transaction is domestic.
- **Time Diff Last Trans Sen:** Time difference between current and last transaction by the same sender account.
- **Time Diff Last Trans Ben:** Time difference between current and last transaction with the same beneficiary account.
- **Average Diff Time Rec:** Cumulative average time difference between consecutive transactions received by each account.
- **Average Diff Time Sen:** Cumulative average time difference between consecutive transactions sent by each account.
- **Average USD Sent:** Cumulative average amount in USD sent by each sender account.
- **Average USD Rec:** Cumulative average amount in USD received by each beneficiary account.
- **Max USD Sent:** Maximum amount in USD sent by each sender account.
- **Max USD Rec:** Maximum amount in USD received by each beneficiary account.
- **Min USD Sent:** Minimum amount in USD sent by each sender account.
- **Min USD Rec:** Minimum amount in USD received by each beneficiary account.
- **Time step:** Column converted to date and time format.

- **Sender Transactions Last 1/7/30 Days:** Number of transactions sent by the sender account in the last 1, 7, and 30 days, respectively.
- **Bene Transactions Last 1/7/30 Days:** Number of transactions received by the beneficiary account in the last 1, 7, and 30 days, respectively.

Results obtained with the transactional J.P. Morgan data set

The results obtained are the following ones:

Algorithm	Class	Precision	Recall	F1-score	Macro Average Precision	Macro Average Recall	Macro Average F1-Score
Logistic Regression	0	0.99	0.59	0.74	0.51	0.66	0.40
	1	0.04	0.73	0.07			
KNN (Neighbour=4)	0	0.98	0.89	0.93	0.52	0.58	0.51
	1	0.05	0.27	0.08			
Random Forest (Estimators=50)	0	0.98	0.93	0.95	0.52	0.57	0.52
	1	0.06	0.20	0.09			
Naives Bayes (Bernouilli)	0	1.00	0.21	0.34	0.51	0.59	0.20
	1	0.03	0.98	0.05			
Decision Tree (Max Depth=5)	0	1.00	0.24	0.39	0.51	0.62	0.22
	1	0.03	1.00	0.05			
XGBoost	0	0.98	0.88	0.92	0.51	0.60	0.51
	1	0.03	0.31	0.09			
XGBoost & Logistic Regression	0	0.98	0.94	0.96	0.52	0.60	0.53
	1	0.06	0.19	0.10			

Tabla 37: Results of J.P Morgan data set experiments

The ensemble model, consisting of Logistic Regression and XGBoost using a Voting Classifier, showed mixed results in detecting fraudulent bank transactions. While it achieved high precision for non-fraudulent transactions (98 %),

the recall for fraudulent transactions was low (28 %). The F1-Score for fraudulent transactions was also low (11 %), indicating the need for significant improvements in detecting fraud. Although the overall accuracy was 90 %, the model's performance was influenced by the class imbalance in the dataset. To enhance the model's effectiveness, further improvements such as resampling techniques, exploring different algorithms, or incorporating additional attributes should be considered.

Implementation of the Ethereum model in a smart contract

To detect fraudulent accounts in Ethereum, an off-chain machine learning model is employed. The model resides on an external server, which is crucial as executing it directly on the blockchain would be impractical and costly. A smart contract on Ethereum gathers necessary data, which can be obtained from blockchain events and transactions, and potentially enriched with historical or additional information through APIs. An oracle serves as a bridge for communication between the smart contract and the external machine learning model. The smart contract sends a request with input data to the oracle, which then passes the data to the model. The model processes the data and sends back its prediction (whether an account is fraudulent or not) through the oracle to the smart contract.

The smart contract plays a vital role; it not only collects data but also communicates with the external model via the oracle. Once the smart contract receives the prediction, it processes the information and, if the account is detected as potentially fraudulent, records this information on the blockchain. It is

essential to have the smart contract optimized regarding resource consumption to ensure economic viability on the Ethereum blockchain. Besides, extensive testing and ensuring the security of the smart contract are paramount, especially considering the serious implications of labeling accounts as fraudulent. Lastly, the machine learning model may require updates and maintenance over time. It is crucial to have strategies for managing these updates and ensuring that the oracle functions correctly and that the system's integrity is maintained.

Conclusions

This project has been a fulfilling learning experience in data science techniques, such as data mining and machine learning. The steep learning curve has offered invaluable insights into the complexity of fraud detection. However, the endeavor faced challenges, particularly with computational times and mixed results in the J.P. Morgan dataset. Real-world data can be messy, necessitating adaptability and revised strategies. Comparing models between Ethereum and J.P. Morgan highlighted the importance of contextual understanding and custom approaches. This project was not only about achieving results but also about honing critical thinking and problem-solving skills in data science, which are essential for future projects. The experience has underscored the importance of continuous learning in this ever-evolving field, and the insights gained will be invaluable for future innovation. Overall, this project marks a significant milestone in my development as a data scientist.

Conclusions for the Ethereum model

In summary, the XGBoost model gave good results, though there's room for improvement, possibly through ensemble techniques. However, due to the focus on the J.P. Morgan dataset, this project did not allocate much time to fine-tune the Ethereum model. The Ethereum dataset may not be very realistic as it's outdated by two years, and the Ethereum blockchain has since evolved. Caution is advised if using the model for classification as its reliability is questionable. Comparing with results from Kaggle, it's evident that preprocessing plays a significant role. One notable Kaggle work had better XGBoost performance, possibly due to not discretizing the data, which can cause information loss affecting complex models like XGBoost more than simpler ones like Logistic Regression.

Conclusions for the J.P. Morgan model

The results obtained using this dataset did not meet expectations. One of the key issues was the large size of the dataset, which made it computationally demanding to work with. For instance, running the KNN algorithm took a substantial amount of time (8 hours and 38 minutes), and even then, the results were unsatisfactory. Other methods such as ensemble techniques and neural networks were considered, but they either did not offer a significant improvement or were too resource-intensive to be practical with the computing power available.

Additionally, it is believed that the dataset was not comprehensive enough. Specifically, it lacked historical account data, which is often essential for effective fraud detection. The absence of historical data deprived the model of important contextual information. For example, knowing an account's typical transaction patterns would help identify deviations that might indicate fraud.

The historical data is also important for recognizing how fraud patterns change over time (temporal trends), and for grouping similar accounts together to better understand typical behaviors (account grouping).

Given these challenges, there is still room for improvement. It's possible that an alternative approach to the problem or incorporating more comprehensive data could yield better results. The combination of transaction data with historical account data might enable the model to discern more complex patterns and behaviors, which is critical for effective fraud detection.

Referencias

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[2] M. García, José y Martínez, "El estado de la detección de anomalías utilizando aprendizaje profundo," *Revista de Investigación en Inteligencia Artificial*, vol. 47, pp. 93–121, 2023.

[3] T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach.* Tioga Publishing Company, 1983.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[5] D. W. H. Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Wiley, 2013.

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[9] L. Breiman, J. H. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, August 2016, pp. 785–794.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008.

[12] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," https://ethereum.org/en/whitepaper/, 2014.

[13] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 1–1, 01 2016.

[14] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115002336

[15] S. Assefa. (2020, June) Generating synthetic data in finance: Opportunities, challenges and pitfalls. Available at SSRN. [Online]. Available: https://ssrn.com/abstract=3634235

[16] M. Archie, S. Gershon, A. Katcoff, and A. Zeng, "Who ' s watching ? de-anonymization of netflix reviews using amazon reviews," 2018.

[17] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *Journal of Network and Computer Applications*, vol. 68, pp. 90–113, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516300571

[18] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5916–5923, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413003072

[19] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 17, no. 3, pp. 235–249, 2002. [Online]. Available: http://www.jstor.org/stable/3182781

[20] S.-H. Li, D. C. Yen, W.-H. Lu, and C. Wang, "Identifying the signs of fraudulent accounts using data mining techniques," *Computers in Human Behavior*, vol. 28, no. 3, pp. 1002–1013, may 2012. [Online]. Available: https://doi.org/10.1016/2Fj.chb.2012.01.002

[21] R.-C. Chen, M.-L. Chiu, Y.-L. Huang, and L.-T. Chen, "Detecting credit card fraud by using questionnaire-responded transaction model based on support vector machines," 08 2004, pp. 800–806.

[22] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," pp. 1–9, 2017.

[23] P. Raghavan and N. E. Gayar, "Fraud detection using machine learning and deep learning," in *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2019, pp. 334–339.

[24] S. Lei, M. Xinming, X. Lei, and H. Xiaohong, "Financial data mining based on support vector machines and ensemble learning," in *2010 International Conference on Intelligent Computation Technology and Automation*, vol. 2, 2010, pp. 313–314.

[25] N. K. Gyamfi and J.-D. Abdulai, "Bank fraud detection using support vector machine," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 37–41.

[26] W. Deng, Z. Huang, J. Zhang, and J. Xu, "A data mining based system for transaction fraud detection," in *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 2021, pp. 542–545.

[27] D. Borrajo and M. Veloso, "Domain-independent generation and classification of behavior traces," *CoRR*, vol. abs/2011.02918, 2020. [Online]. Available: https://arxiv.org/abs/2011.02918

[28] P. E. y del Consejo. (2016) Reglamento (ue) 2016/679 del parlamento europeo y del consejo de 27 de abril de 2016. Accedido el: [11/06/2023]. [Online]. Available: https://eur-lex.europa.eu/legal-content/ES/ALL/?uri=celex:32016R0679

[29] PwC, "Global economic crime and fraud survey 2022," Tech. Rep., 2022. [Online]. Available: https://www.pwc.com/gx/en/services/forensics/economic-crime-survey.html

[30] C. Chiticariu, "Fraud detection: Ethereum transactions," 2023, kaggle (https://www.kaggle.com/code/chiticariucristian/fraud-detection-ethereum-transactions). [Online]. Available: https://www.kaggle.com/code/chiticariucristian/fraud-detection-ethereum-transactions