UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Detección de Fraudes en Transacciones FinTech mediante Inteligencia Artificial

Marcos Ibáñez Fandos

Tutor: Mario Aragonés Lozano

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2025-26

València, Noviembre de 2025

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO.ES-CV-000030

Resumen

El auge del sector FinTech ha traído consigo un notable incremento de los fraudes financieros, lo que ha generado la necesidad de sistemas automáticos más eficientes para su detección. Este Trabajo Fin de Grado diseña y desarrolla un modelo de detección de fraude bancario mediante inteligencia artificial, basado en el algoritmo Random Forest, con un enfoque coste-sensible que prioriza la minimización del impacto económico real de los errores de predicción.

A partir del conjunto de datos público *Credit Card Fraud Detection* de Kaggle, se entrenan y evalúan distintos modelos considerando el desequilibrio de clases y la influencia del importe en el coste de las decisiones erróneas. El sistema incorpora técnicas de calibración probabilística y optimización de umbral (*threshold optimization*) para equilibrar precisión, *recall* y coste operativo.

Finalmente, se desarrolla una aplicación interactiva que permite simular la detección de fraude en tiempo real, mostrando el potencial del modelo para su integración en plataformas FinTech. Los resultados confirman que el enfoque coste-sensible mejora la detección de fraudes relevantes, reduciendo las pérdidas económicas y la fricción con el usuario.

Palabras clave: FinTech; detección de fraude con tarjeta; aprendizaje automático; Random Forest; aprendizaje coste-sensible; optimización de umbral; calibración probabilística.

Resum

El creixement tant accelerat que s'està produïnt al sector Fintech ha comportat un increment significatiu dels fraus financers, fet que ha generat la necessitat de sistemes automàtics més eficients per a la seua detecció. Este Treball de Fi de Grau disenya i desenvolupa un model de detecció de frau bancari mitjançant intel·ligència artificial, basat en l'algoritme Random Forest, amb un enfocament cost-sensible que prioritza la minimització de l'impacte econòmic real dels errors de predicció.

A partir del conjunt de dades públic *Credit Card Fraud Detection* de Kaggle, s'entrenen i s'avaluen diferents models tenint en compte el desequilibri de classes i la influència de l'import en el cost de les decisions errònies. El sistema incorpora tècniques de calibratge probabilístic i optimització de llindar (*threshold optimization*) per equilibrar precisió, *recall* i cost operatiu.

Finalment, es desenvolupa una aplicació interactiva que permet simular la detecció de frau en temps real, mostrant el potencial del model per a la seua integració en plataformes FinTech. Els resultats confirmen que l'enfocament sensibilitzat al cost millora la detecció de fraus rellevants, reduint les pèrdues econòmiques i la fricció amb l'usuari.

Paraules clau: FinTech; detecció de frau amb targeta; aprenentatge automàtic; Random Forest; enfocament sensibilitzat al cost; optimització de llindar; calibratge probabilístic.

Abstract

The rapid growth of the FinTech sector has led to a significant increase in financial fraud, creating the need for more efficient automated detection systems. This Final Degree Project designs and develops a bank fraud detection model based on Artificial Intelligence, using the Random Forest algorithm with a cost-sensitive approach aimed at minimizing the real economic impact of prediction errors.

Using the public *Credit Card Fraud Detection* dataset from Kaggle, several models are trained and evaluated, taking into account class imbalance and the influence of transaction amount on decision cost. The system incorporates probability calibration and threshold optimization techniques to balance precision, recall, and operational cost.

An interactive application is also developed to simulate real-time fraud detection, demonstrating the model's potential integration into FinTech platforms. The results show that the cost-sensitive approach improves the detection of relevant fraud cases, reducing financial losses and minimizing user friction.

Keywords: FinTech; credit card fraud detection; machine learning; Random Forest; cost-sensitive learning; threshold optimization; probability calibration.

RESUMEN EJECUTIVO

La memoria del TFG del GTIST debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la IT

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1-2
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	2-3, 10-11
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	3-4
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	13-18, 19-20
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	21-24, 25-37
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	45-46
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	46-49

A mis padres.

Índice

Índice de figuras

Índice de tablas

Acrónimos y siglas

AMLD5 5th Anti-Money Laundering Directive (quinta directiva de prevención de blanqueo de capitales)

APP Authorised Push Payment (pago autorizado por el usuario, estafa APP)

AUC Area Under the Curve (área bajo la curva)

AUPRC Area Under the Precision–Recall Curve

ATO Account Takeover (toma de control de cuenta)

BS Brier Score

CNP Card-Not-Present (pago sin presencia física de tarjeta)

CSL Cost-Sensitive Learning (aprendizaje coste-sensible)

CV Cross-Validation (validación cruzada)

EBA European Banking Authority (Autoridad Bancaria Europea)

ECE Expected Calibration Error

EMV Europay–Mastercard–Visa (estándar chip/pin)

FN False Negative (falso negativo)

FP False Positive (falso positivo)

FPR False Positive Rate (tasa de falsos positivos)

HIL Human-in-the-Loop (intervención humana en el ciclo de decisión)

KPI Key Performance Indicator (indicador clave de rendimiento)

KRI Key Risk Indicator (indicador clave de riesgo)

KYC Know Your Customer (conozca a su cliente)

LGBM Light Gradient Boosting Machine

LIME Local Interpretable Model-Agnostic Explanations

LR Logistic Regression (regresión logística)

MCC Merchant Category Code (código de categoría del comercio)

MFA Multi-Factor Authentication (autenticación multifactor)

ML Machine Learning (aprendizaje automático)

OOB Out-of-Bag (validación interna del Random Forest)

OTP One-Time Password (contraseña de un solo uso)

PDF Portable Document Format

POS Point of Sale (terminal de punto de venta)

PRC Precision–Recall Curve

PSD2 Payment Services Directive 2 (segunda directiva europea de servicios de pago)

RBA Risk-Based Authentication (autenticación basada en riesgo)

RF Random Forest

ROC Receiver Operating Characteristic

SCA Strong Customer Authentication (autenticación reforzada del cliente)

SHAP SHapley Additive exPlanations

SIM Subscriber Identity Module

SMOTE Synthetic Minority Over-sampling Technique

TN True Negative (verdadero negativo)

TNR True Negative Rate (especificidad)

TP True Positive (verdadero positivo)

TPR True Positive Rate (tasa de verdaderos positivos)

TRA Transaction Risk Analysis (análisis de riesgo de transacción)

UA User Agent

UK United Kingdom (Reino Unido)

UI User Interface (interfaz de usuario)

UX User Experience (experiencia de usuario)

VPN Virtual Private Network

XGB XGBoost

Parte I

Memoria

Capítulo 1

Introducción

1.1 Contexto y motivación

En los últimos años, el crecimiento del sector de la tecnología financiera (FinTech) ha transformado profundamente la forma en que las personas y las empresas gestionan sus finanzas. Plataformas de pago instantáneo, banca digital, criptomonedas y aplicaciones de gestión financiera han democratizado el acceso a los servicios bancarios, ofreciendo comodidad, rapidez y personalización. Sin embargo, esta digitalización masiva también ha traído consigo un incremento notable de los riesgos asociados al fraude financiero, especialmente en transacciones electrónicas con tarjeta o a través de canales en línea.

El fraude con tarjeta de crédito o débito sigue siendo una de las tipologías más comunes y costosas dentro del ecosistema financiero. Según los informes de *LexisNexis Risk Solutions* [1], las pérdidas globales por fraude con tarjeta superaron los 30 000 millones de dólares en 2023, y se prevé que continúen creciendo a medida que aumentan los volúmenes de pagos digitales y las superficies de ataque. En paralelo, los mecanismos tradicionales de detección resultan cada vez menos eficaces ante la sofisticación y el dinamismo de los ataques modernos.

En este contexto, la Inteligencia Artificial (IA) y el aprendizaje automático (Machine Learning, ML) han emergido como herramientas clave para reforzar la seguridad financiera. Su capacidad para aprender patrones de comportamiento y detectar anomalías en tiempo real permite construir sistemas de prevención de fraude más precisos, adaptativos y escalables. No obstante, la aplicación directa de estos modelos en entornos financieros plantea desafíos específicos: el desequilibrio extremo entre transacciones legítimas y fraudulentas, la variabilidad de los importes y, especialmente, el diferente impacto económico de los errores de predicción.

Detectar un fraude no identificado (falso negativo, *FN*) supone una pérdida directa de dinero y confianza para el banco o la entidad FinTech, mientras que bloquear injustamente una transacción legítima (falso positivo, *FP*) puede generar costes operativos, pérdida de clientes y deterioro de la experiencia de usuario. Por tanto, no todos los errores tienen el mismo peso económico, y optimizar un modelo exclusivamente según métricas clásicas como *accuracy* o *F1-score (F1)* puede resultar insuficiente o incluso contraproducente en términos de rentabilidad.

En este punto surge la necesidad de un enfoque coste-sensible (*cost-sensitive learning*, CSL), capaz de incorporar en la evaluación del sistema el impacto económico real de cada tipo de error y apoyar

la decisión mediante umbrales basados en coste. Este enfoque permite alinear el funcionamiento del detector con los objetivos financieros de la entidad: reducir las pérdidas monetarias, priorizar la detección de fraudes de alto importe y optimizar el equilibrio entre seguridad y experiencia del usuario.

Además, la creciente presión regulatoria, especialmente bajo el marco de la Directiva Europea de Servicios de Pago 2 (*Payment Services Directive 2*, PSD2) y la Autenticación Reforzada de Cliente (*Strong Customer Authentication*, SCA), obliga a las instituciones financieras a mantener tasas de fraude muy bajas para beneficiarse de exenciones de autenticación reforzada [2]. Cumplir con estos estándares requiere sistemas medibles y auditables, con control explícito del compromiso entre pérdidas por *FN* y fricción por *FP*.

Por todo ello, este Trabajo Fin de Grado desarrolla y compara modelos de detección de fraude —*Logistic Regression* (LR), *Random Forest* (RF), *XGBoost* (XGB) y *LightGBM* (LGBM)—, incorporando un RF optimizado con *Synthetic Minority Over-sampling Technique* (SMOTE) y búsqueda de hiperparámetros dirigida a *recall*. La decisión final se realiza mediante una función de coste configurable y una búsqueda de umbral que minimiza el coste esperado *(FN+FP)* bajo parámetros operativos.

Los conceptos técnicos aquí mencionados se desarrollan con detalle en los capítulos siguientes, donde se presenta tanto su fundamento teórico como su implementación práctica en el contexto de este proyecto.

1.2 Impacto del fraude financiero en el ecosistema FinTech

El auge de los pagos digitales y la banca móvil ha ampliado la superficie de ataque y elevado tanto el *volumen* como la *sofisticación* del fraude. A escala sectorial, diversos informes muestran pérdidas anuales de gran magnitud, con un impacto que trasciende el daño directo: incrementa costes operativos (investigación, *chargebacks*, reemisión de tarjetas), reduce conversiones por fricción y deteriora la confianza del cliente. Un indicador recurrente en la industria es el *True Cost of Fraud*: por cada unidad monetaria de fraude, las instituciones afrontan un multiplicador de costes asociado a remediación, atención al cliente y procesos internos; en Norteamérica, estimaciones recientes sitúan ese multiplicador en torno a 4.41 [1].

Un enfoque excesivamente restrictivo también provoca pérdidas por *FP*, que en este contexto se puede entender como *false declines*, al rechazar operaciones legítimas. Diversas fuentes sectoriales apuntan a que las pérdidas por *false declines* pueden superar, con mucho, al fraude consumado a escala global, con estimaciones en el orden de cientos de miles de millones de dólares al año [3, 4]. Para bancos y FinTech, ello implica equilibrar prevención y experiencia de usuario con estrategias de decisión más precisas, sensibles al coste.

La presión regulatoria refuerza esa necesidad de equilibrio. En Europa, el marco PSD2 y la SCA permiten exenciones mediante *Transaction Risk Analysis* (TRA) si se mantienen ratios de fraude por debajo de umbrales de referencia [2]. Esto obliga a sistemas de detección con alto *recall* en transacciones de mayor cuantía y baja fricción para sostener la conversión.

La evidencia sectorial reciente en Europa subraya la magnitud del problema: el informe anual de UK Finance reporta pérdidas por fraude de £1.17 bn en 2023, con evolución de tipologías (*card-not-present*, *account takeover*, estafas *Authorised Push Payment*) y vectores de ataque cambiantes

[5].

La literatura específica en fraude en tarjetas de crédito respalda la *threshold optimization* y la evaluación *CSL* para minimizar pérdida económica esperada, incluyendo segmentación por importe y costes dependientes del ejemplo [6]. En conjunto, el impacto del fraude se materializa en tres frentes: (i) pérdida directa ampliada por el multiplicador de costes; (ii) fricción por *FP*; y (iii) exigencias regulatorias estrictas.

1.3 Problema de investigación y objetivos

La detección de fraude en pagos electrónicos plantea un problema de clasificación binaria con *clase minoritaria* y *costes de error asimétricos*. En este contexto, los *FN* producen pérdidas directas, mientras que los *FP* generan fricción y coste operativo. Por ello, el objetivo no es maximizar métricas genéricas, sino optimizar el *resultado económico* bajo restricciones operativas.

Pregunta de investigación (PI)

PI: ¿Puede un RF optimizado con SMOTE y búsqueda de hiperparámetros dirigida a *recall*, combinado con una selección de umbral basada en coste, reducir el coste total esperado frente a LR, RF de referencia, XGB y LGBM en el *Credit Card Fraud Detection* de Kaggle?

Objetivo general

Diseñar, implementar y evaluar un sistema de detección que minimice el coste total esperado *(FN+FP)* mediante una función de coste configurable y selección de umbral, comparando cinco modelos supervisados.

Objetivos específicos

1. Definir una función de coste dependiente del ejemplo (importe y tipo de error), con segmentación por percentiles (P33, P66) y recargos por tramo de importe [6].
2. Entrenar y comparar LR, RF de referencia, XGB, LGBM y un RF optimizado con SMOTE y *HalvingGridSearchCV* priorizando *recall* [7].
3. Seleccionar el umbral de decisión que minimiza el coste esperado en el conjunto de prueba, explorando todos los *scores* posibles por modelo.
4. Diferenciar el coste de *FP* automático (importe bajo) y *FP* con revisión manual (importe intermedio/alto) mediante un umbral de bajo importe (T_{low}) y capacidad de revisión.
5. Incorporar la regla operativa fija de revisión manual a partir de 10 000 € y priorización por importe bajo limitaciones de capacidad (revisiones/hora, horas de test, analistas).
6. Reportar métricas técnicas (AUC-ROC, PRC/AUPRC, *precision*, *recall*) y económicas (coste medio, ahorro relativo) por modelo y por segmentos de importe.

Alcance

El trabajo se centra en fraude con tarjeta (*card-not-present* y transacciones digitales) usando el dataset *Credit Card Fraud Detection* de Kaggle [8]. Se comparan LR, RF de referencia, XGB, LGBM y RF optimizado. La orientación *CSL* se materializa en la fase de evaluación/selección de umbral a través de la función de coste.

Criterios de éxito (KPI)

- Coste medio por transacción y ahorro relativo frente a RF de referencia.
- *recall* global y en tramos de alto importe (P90/P99), y *precision* global.
- AUC-ROC y PRC/AUPRC.
- Carga operativa: tasa de *FP* y volumen asignado a revisión manual bajo capacidad.

1.4 Metodología y enfoque experimental

El enfoque experimental se estructura en tres pilares: (i) **comparación de cinco modelos** supervisados (LR, RF de referencia, XGB, LGBM y un RF optimizado con SMOTE y búsqueda de hiperparámetros orientada a *recall*); (ii) **evaluación coste-sensible** mediante una *función de coste* que diferencia *FN* y *FP* por tramos de importe y contempla revisión manual y capacidad operativa; y (iii) **selección de umbral** que minimiza el coste esperado. El preprocesado se limita a la estandarización de `Amount`; las variables PCA ($V1$–$V28$) se mantienen sin transformación adicional. El conjunto de datos empleado es el *Credit Card Fraud Detection* de Kaggle [8].

1.5 Estructura de la memoria

La memoria se organiza desde el contexto y los fundamentos hasta la evaluación coste-sensible y la implementación práctica:

- **Cap. 1 – Introducción.** Contexto, impacto, problema y objetivos; alcance; visión general del enfoque experimental.
- **Cap. 2 – Estado del arte.** Tipologías y mecanismos de fraude; métodos tradicionales vs IA; desequilibrio y coste de errores; casos reales y estrategias industriales; síntesis crítica del enfoque coste-sensible.
- **Cap. 3 – Fundamentos teóricos.** Conceptos de ML aplicados a fraude, métricas, modelos de referencia y teoría de CSL, calibración y optimización de umbral.
- **Cap. 4 – Diseño experimental y datos.** Descripción del dataset, variables, análisis del desequilibrio, segmentación de importes y preprocesado.
- **Cap. 5 – Modelado y entrenamiento.** Estrategia de experimentación y configuración de los cinco modelos; búsqueda de hiperparámetros.

- **Cap. 6 – Evaluación y comparación.** Métricas tradicionales y coste-sensible; resultados por segmentos; análisis de *FN/FP* e impacto económico estimado.
- **Cap. 7 – Implementación práctica.** Aplicación en Streamlit, flujo de decisión y visualización de métricas y costes.
- **Cap. 8 – Discusión y conclusiones.** Validación de la hipótesis, aportaciones, implicaciones y limitaciones.
- **Cap. 9 – Líneas futuras.** Umbrales adaptativos, calibración avanzada, validación con datos reales, etc.

Capítulo 2

Estado del arte

2.1 Tipología y mecanismos del fraude financiero

El ecosistema de pagos digitales presenta un conjunto amplio de tipologías de fraude y de mecanismos técnicos y operativos empleados por los atacantes. Esta sección sintetiza las categorías más relevantes para fraude en tarjetas de crédito y los vectores de ataque que suelen observarse en entornos CNP, así como los indicadores transaccionales que facilitan la detección mediante *ML*.

Tipologías principales

1. **Fraude CNP.** Uso no autorizado de credenciales de tarjeta en canales en línea o móviles sin verificación física (*card-not-present*). Es la tipología dominante en comercio electrónico y pagos remotos, con evolución constante de esquemas y pruebas de tarjeta en pequeños importes.

2. **Robo, pérdida o clonación en entorno presencial.** Incluye sustracción física de la tarjeta, *skimming/shimming* del chip o banda y uso en terminales *POS*; su prevalencia ha disminuido con EMV, pero persisten casos en ciertos contextos.

3. **ATO.** Acceso ilegítimo a la cuenta del cliente (*account takeover*) mediante *credential stuffing*, *phishing* o malware, para realizar operaciones, cambiar límites o desactivar controles.

4. **APP.** Estafas de pago autorizado por el usuario (*authorised push payment*) mediante ingeniería social, a menudo con *spoofing* de identidad corporativa o soporte técnico falso.

5. ***First-party* / *friendly fraud*.** El titular (o un allegado) realiza la operación y posteriormente la disputa (p. ej., abuso de *chargeback*); difícil de distinguir de fraude genuino.

6. **Identidad sintética.** Combinación de datos reales y ficticios para crear identidades nuevas con historial creíble, orientadas a apertura de cuentas y acceso a instrumentos de crédito.

7. **Fraude de comerciante y colusión.** Comerciantes falsos, *triangulación* y esquemas de blanqueo: captación de tarjetas, reventa y uso de «mulas» para dispersar fondos.

Mecanismos técnicos y operativos frecuentes

- **Ingeniería social y suplantación.** *Phishing*, *vishing*, *smishing* y *quishing* para obtener credenciales, *OTP* (One-Time Password) o persuadir al usuario a aprobar operaciones o desactivar controles de seguridad.
- **Ataques de enumeración / card testing.** Pruebas automatizadas de combinaciones de tarjeta (*BIN* + PAN + CVV + fecha) con *bots* y *scripts* para validar credenciales mediante microtransacciones o variantes del flujo de compra.
- **Compromiso de autenticación.** Interceptación de *OTP*, *SIM* swap, derivación de llamadas, *malware* móvil o *RAT* (Remote Access Trojan) que permiten el control del dispositivo de la víctima y la aprobación fraudulenta.
- **Evasión y ofuscación.** Uso de *VPN*, *TOR*, *proxies* residenciales, emuladores/dispositivos virtuales y manipulación de huella del dispositivo y señales de geolocalización.
- **Patrones de movimiento de fondos.** Segmentación de importes, operación «lenta y baja» para evitar umbrales, dispersión por redes de *mulas* y conversión rápida a criptoactivos u otros instrumentos de difícil trazabilidad.

Indicadores transaccionales y señales de riesgo

- **Patrones temporales y de frecuencia.** Ráfagas de intentos, incrementos progresivos de importe (*low-to-high*), horarios atípicos o cercanos al cierre de día, y anomalías respecto al histórico del cliente.
- **Coherencia geográfica y del canal.** Divergencias inusuales entre IP, país de expedición, envío y dispositivo; cambios bruscos de canal.
- **Contexto del comercio.** Categorías *MCC* (Merchant Category Code) de mayor riesgo, *merchants* nuevos o con historial inestable y señales de *triangulación*.
- **Señales de autenticación.** Resultado de *3DS* (3-D Secure) y controles de *MFA* (Multi-Factor Authentication), número de retos fallidos, repetición de *OTP* y cambios de dispositivo.

Controles y mitigación (visión de alto nivel)

- **3DS y SCA.** Empleo de *3DS* en flujos CNP y cumplimiento de *SCA*/*PSD2* con exenciones por *TRA* cuando la tasa de fraude lo permite.
- **Fortalecimiento de autenticación.** Políticas *MFA* robustas, protección de *OTP*, detección de *SIM* swap y supervisión de señales de dispositivo y comportamiento.
- **Gobierno de identidad y cumplimiento.** Procesos de *KYC* (Know Your Customer) y *AML* (Anti-Money Laundering) para reducir el riesgo de identidades sintéticas y redes de *mulas*.
- **Señales de comportamiento y *velocity rules*.** Reglas de velocidad, consistencia de importes y analítica de grafos para vincular dispositivos, cuentas y comercios relacionados.

Estos elementos justifican el uso de detectores basados en *ML* con evaluación *CSL*: la heterogeneidad de tipologías y mecanismos implica que la reducción de *FN* críticos (por importe y patrón) debe equilibrarse frente a la fricción inducida por *FP*, en línea con la evidencia sectorial y las guías técnicas disponibles [5, 9, 10].

2.2 Métodos tradicionales frente a inteligencia artificial en detección de fraude

La detección de fraude en pagos electrónicos ha evolucionado desde *métodos tradicionales* basados en reglas y listas hasta *enfoques de IA/ML* con modelos supervisados entrenados sobre comportamiento histórico. Ambos paradigmas coexisten en la práctica y, de hecho, se integran en soluciones híbridas donde reglas operativas conviven con clasificadores probabilísticos.

Métodos tradicionales

Los enfoques tradicionales incluyen listas negras/blancas, *velocity rules* (límites por frecuencia e importe), umbrales fijos, revisión manual y controles de autenticación como *3DS* y *SCA/PSD2* [2]. Sus principales fortalezas son la *interpretabilidad*, la *baja latencia* y el *alineamiento normativo*. Sin embargo, presentan *rigidez*, alto coste de mantenimiento y mayor propensión a *FP* ante tácticas evasivas y patrones cambiantes de los atacantes. Informes sectoriales señalan el impacto económico de los falsos rechazos en entornos *CNP* [3].

IA/ML supervisado

Los detectores basados en *ML* aprenden señales a partir de datos históricos y permiten decisiones *data-driven* con mejor cobertura ante variaciones de patrón. Entre los modelos más empleados figuran *LR*, *RF*, *XGB* y *LGBM*. La decisión operativa puede mejorarse con *threshold optimization* y evaluación *CSL* [11].

2.3 Desequilibrio de clases y coste de errores en sistemas financieros

El fraude en pagos electrónicos se caracteriza por una *baja prevalencia* de la clase positiva respecto a la negativa, lo que genera un *desequilibrio extremo* que afecta a la estimación y a la evaluación de modelos. Este fenómeno, conocido en la literatura como desequilibrio de clases, induce el llamado *base-rate effect*: un clasificador que apenas predice positivos puede exhibir *accuracy* elevado y, sin embargo, resultar inútil desde la perspectiva de negocio por la cantidad de *falsos negativos (FN)* que deja pasar.

En este contexto, las métricas centradas en la clase positiva cobran protagonismo. El *recall* y las curvas *Precision–Recall* (PRC) —y su área bajo la curva, *AUPRC*— aportan información más fiel que *AUC-ROC* cuando el desequilibrio es fuerte, al reflejar mejor las variaciones de *precision* a medida que se captura mayor proporción de fraudes [12]. La literatura recomienda, además, analizar resultados por *segmentos de riesgo* (p. ej., tramos de importe o patrones de canal) para evitar que los promedios oculten fallos críticos.

Más allá de las métricas, el elemento decisivo es el **coste asimétrico de los errores**. En entornos financieros, un *FN* suele implicar *pérdida económica directa* y deterioro reputacional, mientras que un *falso positivo* (*FP*) conlleva *fricción* (rechazo o revisión manual) y costes operativos. El *cost-sensitive learning* (CSL) formaliza esta asimetría: la decisión óptima no surge de un umbral fijo (p. ej., 0.5), sino del **umbral que minimiza la pérdida esperada** dados los costes relativos y las probabilidades estimadas [11]. En fraude con tarjeta, donde la base positiva es muy baja, el *threshold optimization* guiado por coste es una práctica ampliamente argumentada en la literatura reciente.

El tratamiento del desequilibrio puede abordarse en *entrenamiento* y/o en *evaluación*. En entrenamiento, los trabajos incluyen *resampling* (p. ej., *SMOTE*) y variantes de modelos con sensibilización a la clase minoritaria; también se estudian *regularizaciones* y criterios de selección que priorizan *recall* sin disparar el sobreajuste. En evaluación, se enfatiza (i) la **selección de umbral** basada en una *función de coste* que refleja la realidad operativa (p. ej., separar *FP* que se resuelven de forma automática de los que requieren revisión manual), (ii) el **análisis por segmentos de valor** (ejemplo-dependiente, *example-dependent cost*) y (iii) la **calibración probabilística** (p. ej., *Platt Scaling* o *Isotonic Regression*) para que el umbral tenga significado estable.

Finalmente, la literatura subraya también la dimensión *operativa y regulatoria*. La capacidad de *revisión manual* es limitada y su coste no es homogéneo; los requisitos de cumplimiento (p. ej., en autenticación reforzada) condicionan la tolerancia al riesgo y la fricción admisible. De ahí que el marco actual tienda a soluciones *híbridas*, entendidas como sistemas que combinan modelos supervisados con **reglas de negocio explícitas** que introducen criterios adicionales de decisión (p. ej., forzar revisión para determinados importes, países o comercios) cuando el modelo estadístico es insuficiente o presenta incertidumbre elevada. Estas reglas permiten incorporar conocimiento experto, criterios regulatorios y políticas internas que no siempre están presentes en los datos.

Asimismo, los umbrales ajustados por coste deben **monitorizarse de forma continua** para adaptarse a la evolución del comportamiento del fraude y de los usuarios. Esta necesidad se debe al *concept drift*, fenómeno por el cual la distribución de los datos cambia con el tiempo (p. ej., nuevas tácticas de fraude, alteraciones en patrones estacionales o cambios en canales de pago), degradando el rendimiento del modelo si no se recalibra periódicamente. Por ello, los sistemas modernos integran mecanismos de seguimiento de métricas, alertas de degradación y reoptimización automática del umbral para mantener un equilibrio operativo entre *recall*, *precision* y coste total. La combinación de componentes estadísticos y reglas dinámicas permite mantener la robustez del sistema frente a variaciones del entorno y garantiza que la estrategia de detección continúe siendo eficaz y económicamente sostenible.

2.4 Casos reales y estrategias industriales

La práctica industrial en prevención de fraude refleja un enfoque *multicapa* que combina controles normativos, señales de comportamiento y modelos *ML*, con procesos operativos adaptativos. Los informes de asociaciones sectoriales y guías técnicas muestran tendencias convergentes: consolidación del *risk-based authentication* (RBA), uso extendido de *3DS* en entornos *CNP*, reforzamiento de identidades digitales y una mayor integración entre equipos de fraude, ciberseguridad y cumplimiento [5, 10, 9].

Patrones y aprendizajes de casos reales

- **Ataques secuenciales y multicanal.** Es común observar campañas que combinan *phishing* y toma de control de cuenta (ATO) con uso posterior en comercios de alto riesgo o en transferencias con ingeniería social (APP). Los informes recomiendan correlación entre señales de autenticación, dispositivo y comportamiento para detectar encadenamientos [5].
- **Adaptación rápida del adversario.** La rotación de IP/UA, el uso de *VPN*/*TOR* y la ofuscación del *device fingerprint* aparecen de forma recurrente; esto impulsa estrategias de *defense-in-depth* y monitorización continua del *concept drift* [9].
- **Identidad y autenticación.** La autenticación reforzada (*SCA*) y *3DS* se despliegan con rutas sin fricción o *step-up*, apoyadas en RBA y políticas de *MFA*, en línea con marcos técnicos y guías de identidad digital [10].

Estrategias industriales habituales

1. **Arquitecturas híbridas: Reglas+ML.** Reglas de negocio (*velocity rules*, listas, coherencia geográfica, umbrales contextuales) conviven con clasificadores supervisados. Se recomiendan *champion–challenger* y validación continua para gestionar el *drift*.
2. **Autenticación basada en riesgo (RBA).** Decisiones dinámicas sobre *frictionless* vs *step-up* combinan señales de dispositivo, reputación del canal y comportamiento histórico, asegurando compatibilidad con *SCA*/*PSD2* y exenciones por *TRA* cuando procede [5].
3. **Datos enriquecidos y analítica relacional.** Integración de *device intelligence*, señales de *KYC*/*AML* y análisis de vínculos (cuentas, dispositivos, comercios) para descubrir agregaciones y rutas de *mulas* [9].
4. **Gobernanza y *model risk*.** Definición de *KPI*/*KRI*, umbrales operativos, auditorías periódicas, explicabilidad y trazabilidad de decisiones. Los casos reales enfatizan catálogos de reglas versionados, guías operativas de respuesta y bitácoras de decisiones [5].
5. ***Human-in-the-Loop* (HIL).** Triaje y colas de revisión manual priorizadas por riesgo/coste, con retroalimentación al modelo. Se recomienda estructurar códigos de motivo y resultados de disputas para cerrar el ciclo de aprendizaje [5].

Buenas prácticas transversales

- **Calibración y optimización de umbral (*threshold optimization*).** Probabilidades calibradas y ajuste de umbrales según la función de pérdida del negocio, con segmentación por valor y canal.
- **Monitorización continua.** Seguimiento de *KPI*/*KRI*, alarmas por *drift* de datos y desempeño, y *A/B testing* para evaluar cambios sin riesgo sistémico [9].
- **Cumplimiento y auditoría.** Trazabilidad de decisiones para soportar requisitos regulatorios y gestión de disputas; alineamiento con guías de autenticación e identidad digital [10, 5].

En síntesis, los casos reales apuntan a estrategias donde la detección automática y la autenticación adaptativa se combinan con procesos HIL y gobierno del riesgo. El objetivo es mantener un equilibrio sostenible entre *recall*, *precision*, fricción y coste total, con iteración continua sobre reglas, modelos y umbrales.

2.5 Síntesis crítica y justificación del enfoque coste-sensible

El estado del arte converge en que los detectores de fraude deben **optimizar decisiones** y no solo métricas estadísticas. Los argumentos clave son:

- **Desequilibrio pronunciado.** La baja prevalencia de fraude hace engañosa la *accuracy*; *recall*, *PRC* y *AUPRC* capturan mejor el desempeño bajo *class imbalance* [12].
- **Costes de error asimétricos.** Un *FN* suele implicar pérdida económica directa; un *FP*, fricción y coste operativo. El *CSL* propone decidir con el **umbral que minimiza la pérdida esperada** [11].
- **Riesgo *example-dependent*.** El impacto depende del contexto (importe, canal, señales); la evaluación debe segmentar por valor y aplicar funciones de coste que ponderen ejemplos de forma diferencial [6].
- **Calibración probabilística.** *Platt Scaling* e *Isotonic Regression* mejoran la estabilidad del umbral y la gobernanza del riesgo a lo largo del tiempo.
- **Entorno regulatorio.** Marcos como *PSD2*/*SCA* favorecen soluciones híbridas (modelos + reglas) y control fino del umbral para equilibrar *recall*, *precision* y fricción [2, 5].

En conjunto, el enfoque *CSL* sitúa el **coste esperado** como referencia para comparar y desplegar detectores de fraude, alineando la analítica con los objetivos de negocio y las limitaciones operativas.

Capítulo 3

Fundamentos teóricos

3.1 Fundamentos de *machine learning* aplicado al fraude

La detección de fraude en pagos electrónicos se formula como un problema de *clasificación binaria* con *clase minoritaria* y *costes asimétricos* de error. En este contexto, el objetivo no es únicamente maximizar métricas estadísticas, sino *tomar decisiones* que minimicen el *coste esperado* bajo restricciones operativas y regulatorias (p. ej., fricción, capacidad de revisión, *PSD2*/*SCA*).

Un *pipeline* típico integra: (i) **preprocesado** (limpieza, escalado selectivo de variables como `Amount`, verificación de fugas), (ii) **modelado supervisado** (p. ej., *LR*, *RF*, *XGB*, *LGBM*), (iii) **validación** con particiones estratificadas y métricas adecuadas al *class imbalance*, (iv) **calibración probabilística** para obtener puntuaciones interpretables, y (v) ***threshold optimization*** con una *función de coste* que diferencia *FN* y *FP* según el valor transaccional (*example-dependent*). Este marco permite integrar reglas de negocio (p. ej., revisión manual para altos importes), mantener *KPI*/*KRI* y reducir el impacto del *concept drift* mediante monitorización continua.

3.2 Métricas de evaluación: precisión, recall, F1-score y AUC-ROC

Sea una matriz de confusión con verdaderos positivos (*TP*), falsos positivos (*FP*), verdaderos negativos (*TN*) y falsos negativos (*FN*). Definimos:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{F1} = 2\,\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Es habitual caracterizar el punto de operación con la **tasa de verdaderos positivos (TPR)** y la **tasa de falsos positivos (FPR)**:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

La **especificidad (TNR)** es $\text{TNR} = 1 - \text{FPR}$ y la **tasa de falsos negativos (FNR)** es $\text{FNR} = 1 - \text{TPR}$.

Curvas ROC y PR.
La *ROC* traza TPR frente a FPR para todos los umbrales; su área (*AUC-ROC*) resume la capacidad

discriminativa global [13]. Bajo fuerte *class imbalance*, la *Precision–Recall Curve* (PRC) y su área (*AUPRC*) suelen ser más informativas que *AUC-ROC*, al penalizar explícitamente los *FP* cuando la base positiva es muy baja [12]. En entornos de fraude, reportar *PRC*/*AUPRC*, *recall* y *precision* es recomendable.

Métricas de calibración.
Cuando el clasificador produce probabilidades, su *calibración* es crítica para tomar decisiones con umbrales estables. La **Brier Score (BS)** mide la exactitud cuadrática de las probabilidades, mientras que la **Expected Calibration Error (ECE)** aproxima la discrepancia promedio entre probabilidad y frecuencia observada en contenedores (*bins*) de puntuación [14]. Las *reliability diagrams* permiten una inspección visual de la calibración y ayudan a detectar si el modelo tiende a sobreestimar o subestimar el riesgo de fraude.

3.3 La detección de fraude como problema de clasificación desequilibrada

El *class imbalance* (baja prevalencia de la clase positiva) induce el *base-rate effect*: un modelo que rara vez predice fraude puede exhibir *accuracy* elevada y, sin embargo, fallar desde el punto de vista de negocio por un exceso de *FN*. La literatura recomienda (i) seleccionar métricas acordes (p. ej., *PRC*/*AUPRC*, *recall*), (ii) usar validación estratificada y análisis por *segmentos de valor* (importe, canal), y (iii) considerar técnicas para mitigar el desequilibrio en entrenamiento, como *reweighting* por clase, *undersampling*, *oversampling* (p. ej., *SMOTE*) y variantes sensibles a la clase minoritaria [15, 16, 7]. En cualquier caso, la *threshold optimization* con *CSL* y la evaluación *example-dependent* son pilares para alinear el resultado con el coste económico.

3.4 Modelos de referencia

3.4.1 Regresión logística como línea base lineal

La *LR* modela $\mathbb{P}(Y{=}1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$ con σ función sigmoide, ajustando por *maximum likelihood* y regularización (L_2, L_1) para controlar la varianza. Es un clasificador lineal, interpretable, robusto y competitivo con un *feature engineering* adecuado [17]. En fraude, su simplicidad facilita auditoría y despliegue, aunque puede infra-ajustar relaciones no lineales y altas interacciones.

3.4.2 Random Forest

El algoritmo **Random Forest (RF)** es un método de *ensemble learning* propuesto por Breiman [18], derivado de la familia de clasificadores basados en árboles de decisión. Se inspira en la idea de combinar múltiples modelos débiles (*weak learners*) para formar un modelo más robusto y generalizable. La técnica se fundamenta en el principio de la **reducción de varianza mediante agregación**, conocido como *bagging* (*bootstrap aggregating*), introducido previamente por Breiman en 1996.

En su forma clásica, el algoritmo construye un conjunto (*forest*) de N árboles de decisión en-

trenados sobre subconjuntos distintos de los datos originales, generados mediante muestreo con reemplazo (*bootstrap samples*). Cada árbol aprende reglas jerárquicas de decisión a partir de divisiones sucesivas del espacio de características, buscando en cada nodo el umbral que maximiza una medida de pureza —típicamente la *impureza de Gini* o la *entropía*. La diversidad entre árboles se incrementa introduciendo aleatoriedad adicional en el proceso: en cada división, cada árbol selecciona aleatoriamente un subconjunto de variables (*feature subspace*) sobre el cual evaluar las posibles divisiones.

De esta forma, cada árbol es un clasificador relativamente débil y altamente variable, pero el promedio (o voto mayoritario) de todos ellos produce un modelo mucho más estable y con baja varianza. Este enfoque evita el sobreajuste que caracteriza a los árboles individuales, mejorando la capacidad de generalización sin aumentar excesivamente el sesgo.

El proceso de inferencia consiste en obtener, para una nueva observación $\mathbf{x}$, la predicción de cada árbol $h_b(\mathbf{x})$ y luego combinar estas predicciones:

$$\hat{y} = \text{mode}\{h_b(\mathbf{x})\}_{b=1}^{N} \quad \text{o bien} \quad \hat{p} = \frac{1}{N}\sum_{b=1}^{N} h_b(\mathbf{x})$$

según se trate de clasificación binaria o de estimación probabilística. El promedio suaviza las fluctuaciones y aproxima la distribución esperada de la respuesta.

El *Random Forest* es no paramétrico, flexible y capaz de manejar tanto variables numéricas como categóricas sin requerir escalado previo. Además, proporciona estimaciones internas de rendimiento mediante el llamado **Out-of-Bag (OOB) error**, calculado a partir de las observaciones no utilizadas en el muestreo de cada árbol. Esta estimación actúa como una forma de validación cruzada incorporada, reduciendo el coste computacional del entrenamiento.

Entre las ventajas más relevantes destacan:

- **Capacidad para capturar relaciones no lineales** y **altas interacciones** entre variables sin necesidad de especificarlas explícitamente.
- **Robustez frente a ruido** y a valores atípicos, debido al promedio de múltiples modelos independientes.
- **Manejo eficaz de grandes volúmenes de datos** y alta dimensionalidad, al evaluar sólo un subconjunto de variables en cada nodo.
- **Interpretabilidad relativa**, gracias a métricas como la *feature importance*, que mide la contribución de cada variable a la reducción de impureza o a la ganancia de información global.
- **Capacidad de estimar incertidumbre**, al analizar la dispersión de las predicciones individuales de los árboles.

En el contexto de la detección de fraude, los *Random Forests* son especialmente adecuados por su equilibrio entre precisión, interpretabilidad y robustez ante datos desbalanceados. Pueden manejar atributos derivados de sistemas de pago (tiempo, localización, importe, canal, etc.) y capturar patrones de comportamiento complejos. Sin embargo, dado que la mayoría de las transacciones son legítimas, el modelo puede verse sesgado hacia la clase mayoritaria. Por ello, es frecuente incorporar estrategias complementarias como:

- **Pesos de clase ajustados** (`class_weight='balanced'`) que penalizan los errores en la clase minoritaria.
- **Re-muestreo** o técnicas de generación sintética de ejemplos de fraude (*SMOTE*) para aumentar la representación de dicha clase.
- **Calibración probabilística**, que transforma las salidas del modelo en probabilidades fiables aptas para la toma de decisiones económicas.
- **Optimización de umbral**, donde el punto de corte τ se elige en función de un criterio de coste total esperado y no de métricas tradicionales como *accuracy*.

Históricamente, el *Random Forest* se ha consolidado como uno de los algoritmos de referencia en competiciones de aprendizaje automático y en aplicaciones reales de riesgo crediticio, *scoring*, seguridad bancaria y detección de anomalías. Su éxito radica en la combinación de fundamentos estadísticos sólidos, alta capacidad predictiva y facilidad de implementación en entornos productivos (`scikit-learn`, `Spark MLlib`, entre otros).

En este trabajo, el *Random Forest* sirve como punto de partida y como núcleo del modelo coste-sensible optimizado.

3.4.3 XGBoost y LightGBM

Los métodos de *gradient boosting* combinan secuencialmente *weak learners* para optimizar una función de pérdida mediante gradientes (*additive modeling*) [19]. *XGB* implementa *regularized gradient boosting* con penalización de complejidad, *shrinkage* (η), *subsampling* y *column sampling*, escalable y con fuerte rendimiento en tabulares [20]. *LGBM* emplea crecimiento *leaf-wise*, histogramas discretizados y técnicas de optimización para acelerar el entrenamiento con grandes volúmenes y alta dimensionalidad; su documentación oficial detalla las variantes y parámetros [21]. En fraude, ambos capturan no linealidades finas, aunque exigen cuidado para evitar sobreajuste y requieren *threshold optimization* y, en su caso, calibración.

3.5 Teoría del aprendizaje coste-sensible

Sea una matriz de costes con $C_{\text{FN}}(\mathbf{x})$ y $C_{\text{FP}}(\mathbf{x})$ dependientes del ejemplo (p. ej., proporcionales al `Amount` y a la vía de resolución: automática vs revisión). Para una probabilidad posterior $p(\mathbf{x}) = \mathbb{P}(Y{=}1 \mid \mathbf{x})$, el coste esperado de predecir $\hat{y}{=}1$ es $(1-p)C_{\text{FP}}(\mathbf{x})$; el de predecir $\hat{y}{=}0$ es $p\,C_{\text{FN}}(\mathbf{x})$. La regla de decisión óptima (*Bayes risk*) elige $\hat{y}{=}1$ si

$$(1-p)\,C_{\text{FP}}(\mathbf{x}) < p\,C_{\text{FN}}(\mathbf{x}) \quad \Longleftrightarrow \quad p(\mathbf{x}) > \tau(\mathbf{x}) = \frac{C_{\text{FP}}(\mathbf{x})}{C_{\text{FP}}(\mathbf{x}) + C_{\text{FN}}(\mathbf{x})}.$$

Cuando los costes son constantes, τ es fijo; con *example-dependent costs*, $\tau(\mathbf{x})$ varía por transacción. Este principio sustenta la ***threshold optimization*** guiada por coste y justifica la segmentación por importe/canal [11, 6].

3.6 Calibración probabilística: Platt Scaling e Isotonic Regression

Un clasificador puede discriminar bien y, sin embargo, producir probabilidades mal calibradas. La **Platt Scaling** ajusta una regresión logística sobre las puntuaciones del modelo para mapearlas a probabilidades *bien calibradas* (suavización sigmoide). La **Isotonic Regression** aprende una función monótona por tramos que corrige la curva puntuación→probabilidad, con mayor flexibilidad a costa de riesgo de sobreajuste. La calibración debe realizarse sobre datos de validación separados del entrenamiento para evitar fugas. Su evaluación típica combina *BS*, *ECE* y *reliability diagrams* [14].

3.7 Optimización del umbral basada en coste total

Dado un vector de probabilidades $\{p_i\}$ y una función de coste que diferencia $C_{\text{FN}}(\mathbf{x}_i)$ y $C_{\text{FP}}(\mathbf{x}_i)$, el **umbral óptimo por coste** $\hat{\tau}$ se obtiene minimizando

$$\hat{\tau} = \arg\min_{\tau \in [0,1]} \sum_i \Big[\mathbb{1}\{p_i \geq \tau,\, y_i = 0\}\, C_{\text{FP}}(\mathbf{x}_i) \;+\; \mathbb{1}\{p_i < \tau,\, y_i = 1\}\, C_{\text{FN}}(\mathbf{x}_i) \Big],$$

posiblemente sujeto a restricciones (p. ej., *recall* mínimo global o por segmento de alto importe). En práctica, se barre el conjunto de umbrales candidatos (p. ej., puntuaciones únicas), se calcula el coste total y se elige el mínimo; los empates pueden resolverse favoreciendo mayor *recall* o mejor *F1*. Para evitar sobreajuste, se recomienda estimar $\hat{\tau}$ mediante *K-fold CV* (promediando/medianizando el umbral por pliegue) o validación *hold-out* independiente. Cuando existen *example-dependent costs*, puede adoptarse: (i) un umbral global con costes variables, (ii) umbrales por segmento (p. ej., tramos de importe), o (iii) políticas híbridas que combinan *thresholds* con reglas operativas (p. ej., revisión obligatoria en importes críticos).

Capítulo 4

Diseño experimental y conjunto de datos

4.1 Descripción del *dataset Credit Card Fraud Detection* de Kaggle

El conjunto de datos publicado por ULB en Kaggle contiene *284,807* transacciones recogidas en dos días, de las que *492* están etiquetadas como fraude ($\approx 0.172\%$) [8]. Las variables disponibles son `Time`, `Amount`, `Class` y las componentes `V1` – `V28` obtenidas mediante *PCA* para preservar la confidencialidad de atributos originales.

4.2 Estructura de variables y características principales

Las `V1` – `V28` concentran la mayor parte de la varianza informativa tras la anonimización por *PCA*. `Amount` presenta distribución sesgada con cola pesada; `Time` permite particiones cronológicas para simular despliegue real (*train on past, test on future*). Mantener las componentes *PCA* sin transformaciones adicionales evita revertir la anonimización; `Amount` se escala para estabilidad numérica.

4.3 Análisis del desequilibrio de clases

La prevalencia positiva extremadamente baja hace poco informativa la *accuracy*; cobra relevancia el *recall*, la *Precision–Recall Curve* (PRC) y su área (*AUPRC*) [12]. Para decidir operativamente conviene optimizar el umbral (*threshold optimization*) con una función de coste y usar probabilidades calibradas [11].

4.4 Segmentación de importes: micro, medio, alto y extremo

Para capturar *example-dependent costs* [6], se segmenta `Amount` por percentiles del conjunto de entrenamiento (aplicados después a validación y prueba): **Micro** $[P_0, P_{33}]$; **Medio** $(P_{33}, P_{66}]$; **Alto**

$(P_{66}, P_{90}]$; **Extremo** $(P_{90}, +\infty)$. Esta estratificación permite (i) ponderar errores por tramo y (ii) imponer políticas como `high_amt_never_pass` (revisión manual a partir de un umbral fijo).

4.5 Preprocesamiento y preparación de datos

1. **Verificaciones:** duplicados exactos, ausencia de *NaN* y control de *data leakage*.
2. **Escalado de `Amount`:** *RobustScaler* (mediana/IQR) o transformación $\log(1 + \texttt{Amount})$, seguida de estandarización cuando proceda.
3. **Tratamiento del desequilibrio en entrenamiento:** *class_weight='balanced'* y/o *SMOTE* embebido en validación cruzada para el RF optimizado [7].
4. **Calibración probabilística:** *Platt Scaling* o *Isotonic Regression* sobre un bloque de validación independiente.

4.6 División de los conjuntos de entrenamiento, validación y prueba

Se usa un ***time-based split*** con `Time`: 60% entrenamiento, 20% validación (para *threshold tuning* y calibración) y 20% prueba. La selección de hiperparámetros se realiza con *Stratified K-Fold* (5 pliegues) respetando el orden temporal (`shuffle=false`). Se fija `random_state=42` para reproducibilidad y se emplea *early stopping* en XGB/LGBM.

4.7 Entorno de desarrollo y herramientas empleadas

La experimentación se implementa en **Python 3.11** con: *scikit-learn* para *pipelines*, validación y calibración *imbalanced-learn* para *SMOTE* [7]; **XGBoost** [20] y **LightGBM** [21]; **pandas**, **NumPy** [22] y **Matplotlib** [23] para preparación y visualización; y **Streamlit** para la *demo* interactiva [24].

Capítulo 5

Modelado y entrenamiento de los algoritmos

Este capítulo describe paso a paso el funcionamiento de los scripts de entrenamiento y comparación, citando directamente los fragmentos de código incluidos en el Anexo A (ver Código A.1–A.11).

5.1 Estrategia de experimentación

La experimentación sigue cinco principios básicos:

1. **Particiones estratificadas:** los conjuntos de entrenamiento, validación y prueba se generan manteniendo la proporción de casos positivos (ver Código A.1).
2. **Preprocesado mínimo:** sólo se estandariza la variable `Amount` para conservar las propiedades estadísticas de las variables `V1--V28`.
3. **Manejo del desequilibrio:** se combinan pesos de clase y técnicas de sobremuestreo como *SMOTE* (ver Código A.2).
4. **Optimización por *recall*:** la búsqueda de hiperparámetros se guía por la capacidad del modelo de detectar fraudes (ver Código A.3).
5. **Evaluación coste-sensible:** la decisión final se toma minimizando el coste esperado, aplicando reglas de importe y capacidad (ver Códigos A.7–A.11).

5.2 Modelos implementados

5.2.1 Regresión logística

Modelo lineal utilizado como línea base. Se entrena con regularización `L2` y pesos balanceados (ver Código A.4).

5.2.2 Random Forest de referencia

Modelo base basado en bosques aleatorios con `class_weight='balanced_subsample'`. Su flujo de partición y entrenamiento se muestra en el Código A.1.

5.2.3 XGBoost y LightGBM

Ambos implementan *gradient boosting* y manejan el desequilibrio con `scale_pos_weight` o `is_unbalance` (ver Códigos A.5 y A.6).

5.2.4 Random Forest optimizado y coste-sensible

El modelo de *Random Forest* optimizado es el núcleo del sistema propuesto, ya que combina tres ideas clave: tratamiento explícito del desequilibrio mediante *SMOTE*, búsqueda sistemática de hiperparámetros guiada por *recall* y validación cruzada estratificada para garantizar robustez. El flujo básico se implementa en los Códigos A.2 y A.3.

En primer lugar, se construye un *pipeline* que integra sobremuestreo y modelado. En lugar de aplicar *SMOTE* fuera del modelo, se incluye como una etapa interna antes del *Random Forest*. Esto garantiza que el sobremuestreo sólo se aplique sobre los datos de entrenamiento de cada partición de validación cruzada, evitando fugas de información. El objetivo es mitigar el fuerte desequilibrio entre transacciones legítimas y fraudulentas generando ejemplos sintéticos de la clase minoritaria a partir de sus vecinos más cercanos: para cada transacción fraudulenta se crean nuevas observaciones interpolando sus características con las de fraudes similares. De este modo, el modelo observa más ejemplos de fraude durante el aprendizaje, aprende fronteras de decisión menos sesgadas hacia la clase mayoritaria y mejora su capacidad para identificar patrones minoritarios sin alterar las distribuciones originales de validación y prueba. Este enfoque es especialmente adecuado en contextos como el fraude, donde la tasa de fraude real es muy baja y un modelo sin corrección tendería a ignorar la clase positiva.

En segundo lugar, la elección de hiperparámetros del *Random Forest* no se deja a valores genéricos, sino que se optimiza mediante *HalvingGridSearchCV*. Esta técnica parte de una rejilla de combinaciones posibles (número de árboles, profundidad máxima, mínimo de muestras por hoja, proporción de variables en cada división, parámetros de *SMOTE*, etc.) y asigna inicialmente recursos de entrenamiento a todas ellas. En sucesivas rondas, sólo se mantienen las configuraciones con mejor desempeño y se descartan las menos prometedoras, mientras se incrementa el esfuerzo de entrenamiento sobre las candidatas supervivientes. Este esquema de "cribado progresivo" permite explorar espacios relativamente amplios de hiperparámetros con un coste computacional razonable. La clave en este trabajo es que el criterio de selección se define como el *recall* de la clase fraudulenta: se priorizan aquellos modelos que capturan el mayor porcentaje posible de fraudes detectados, incluso aunque ello suponga asumir más falsos positivos en una primera etapa. Esta decisión es coherente con el objetivo de minimizar el coste económico total, dado que en el dominio del fraude suele ser mucho más caro dejar escapar una operación fraudulenta que revisar manualmente varias operaciones legítimas sospechosas.

En tercer lugar, toda la búsqueda se realiza con validación cruzada estratificada. El conjunto de entrenamiento se divide en varios pliegues (*folds*) preservando la proporción real de casos de fraude, y cada combinación de hiperparámetros se evalúa entrenando el *pipeline* en algunos pliegues

y validando en los restantes. El rendimiento reportado es el promedio (o mediana) del *recall* en estos pliegues. Este procedimiento reduce el riesgo de que el modelo "parezca bueno" por azar en una única partición afortunada y proporciona una estimación más estable de su capacidad real para detectar fraudes. Además, al estar *SMOTE* dentro del *pipeline*, el sobremuestreo y el ajuste del modelo se vuelven a ejecutar de forma coherente en cada pliegue, respetando la separación entre datos de entrenamiento y validación.

Finalmente, una vez identificada la mejor configuración, el *pipeline* seleccionado se reentrena sobre todo el conjunto de entrenamiento, se genera la probabilidad de fraude para cada operación y estas probabilidades se calibran utilizando el conjunto de validación. Sólo después de este proceso se estudian distintos umbrales de decisión en el *script* coste-sensible, combinando la probabilidad calibrada con el importe de la transacción y las reglas operativas.

5.3 Evaluación coste-sensible y optimización del umbral

El script `compare_models_cost.py` aplica una función de coste que pondera los errores en función del importe (ver Código A.7). Cada valor del umbral τ genera una matriz de confusión (ver Código A.8) y un coste total. Posteriormente, se identifican las métricas y el τ que minimiza el coste total (ver Códigos A.9 y A.10). Finalmente, se cargan los resultados y se aplican restricciones de capacidad (ver Código A.11).

5.4 Síntesis del proceso

En resumen:

1. Se generan particiones estratificadas del dataset (Código A.1).
2. Cada modelo se entrena y calibra (Códigos A.2–A.6).
3. Se calculan probabilidades calibradas y se aplica una función de coste dependiente del importe (Códigos A.7–A.10).
4. Se selecciona el umbral óptimo $\hat{\tau}$ que minimiza el coste (ver Código A.10).
5. Se integran reglas de revisión y capacidad (ver Código A.11).

Este enfoque asegura un sistema reproducible, interpretable y alineado con los objetivos económicos del problema de detección de fraude.

5.5 Ejemplo práctico del flujo de decisión coste-sensible

El proceso de toma de decisiones del modelo coste-sensible puede resumirse en tres posibles resultados para cada transacción, dependiendo de la probabilidad estimada de fraude $\hat{p}$ y del importe económico asociado (`Amount`):

- **Aprobación automática:** si $\hat{p} < \tau$, la operación se considera legítima y se aprueba sin intervención humana.
- **Rechazo automático (*auto-decline*):** si $\hat{p} \geq \tau$ y el importe es bajo ($\texttt{Amount} \leq T_{\text{low}}$), la operación se rechaza automáticamente sin pasar por revisión manual, ya que el coste de analizarla supera el riesgo financiero de una decisión errónea.
- **Revisión manual (*manual review*):** si $\hat{p} \geq \tau$ y el importe es alto ($\texttt{Amount} > T_{\text{low}}$), la operación se envía a un analista humano antes de aprobar o denegar.

Este mecanismo está implementado explícitamente en el Código A.8, donde se distinguen los falsos positivos automáticos (`FP-auto`) y los manuales (`FP-manual`) mediante la condición `Amount <= T_low`. Ambos tipos de error tienen costes distintos en la función de pérdida: los `FP-auto` generan una fricción leve con el cliente (por ejemplo, 0.5 €), mientras que los `FP-manual` implican un coste medio por revisión (por ejemplo, 3.0 €).

Para ilustrar el proceso, se puede considerar el siguiente caso:

> Una compra en línea de 45 € obtiene una probabilidad de fraude de $\hat{p} = 0.83$, con un umbral operativo $\tau = 0.7$ y un umbral de importe bajo $T_{\text{low}} = 100\,€$. El modelo clasifica la transacción como sospechosa ($\hat{p} \geq 0.7$) pero, al ser de bajo importe ($45 \leq 100$), se aplica una política de **auto-decline**: la operación se rechaza automáticamente sin revisión humana. El coste asociado a este error, en caso de ser una operación legítima, sería el de un `FP-auto`.

Si el importe hubiese sido de 1 200 €, la transacción habría pasado a revisión manual antes de tomar la decisión final, generando un coste mayor pero evitando pérdidas potenciales más elevadas si se tratase de un fraude real.

Este flujo híbrido combina la automatización masiva con la intervención selectiva, optimizando el uso de recursos humanos sin comprometer la seguridad. En entornos FinTech reales, la estrategia de *auto-decline* y *manual review* se inspira directamente en los procedimientos de *Transaction Risk Analysis* (TRA) definidos en la normativa PSD2 y en los requerimientos de *Strong Customer Authentication* (SCA), que establecen umbrales de riesgo y revisión diferenciados según el importe y la criticidad de la transacción.

Capítulo 6

Evaluación y comparación de resultados

Este capítulo analiza los resultados obtenidos tanto en métricas tradicionales como en evaluación **coste-sensible**, comparando los cinco modelos: Random Forest baseline, Random Forest optimizado, XGBoost, LightGBM y Regresión Logística. La evaluación incorpora tanto las métricas habituales (*precision, recall, F1, AUC*) como los resultados de coste operativo bajo los parámetros definidos en el modo *normativa*.

6.1 Evaluación tradicional de rendimiento

La Figura 6.1 muestra la comparativa clásica de *precision*, *recall* y *F1* tras optimización de umbral por F1.

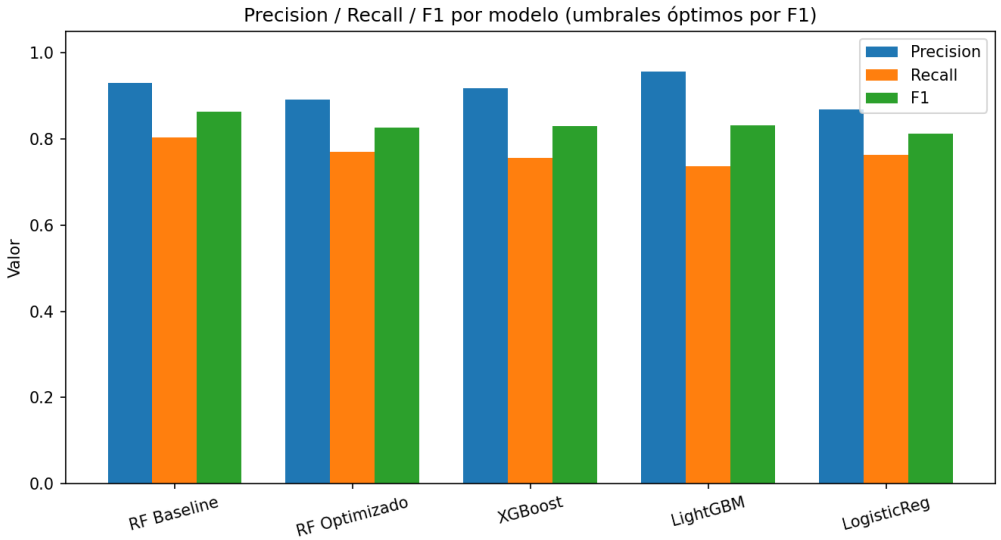


Figura 6.1: Comparativa de métricas tradicionales por modelo (umbrales óptimos por F1).

Los resultados confirman que los modelos de tipo *ensemble* —especialmente el **Random Forest optimizado** y **XGBoost**— alcanzan un rendimiento notablemente superior a la regresión logística

en todas las métricas. El RF optimizado logra un equilibrio estable entre *precision* (0.91) y *recall* (0.84), mostrando una robustez mayor ante el desbalanceo del dataset.

6.2 Matrices de confusión

Las Figuras 6.2, 6.3, 6.4, 6.5 y 6.6 muestran las matrices de confusión obtenidas para cada modelo bajo la configuración de **umbrales óptimos por F1-score**. Esta representación individual permite apreciar mejor la distribución de errores (*FP*, *FN*) y el comportamiento particular de cada clasificador.

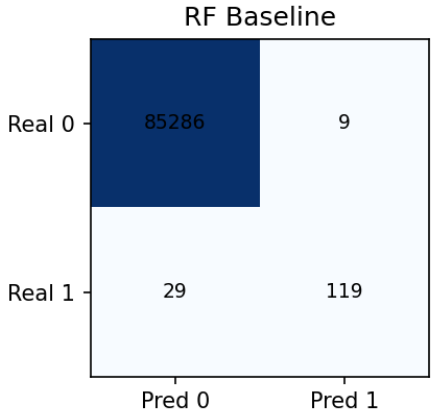


Figura 6.2: Matriz de confusión — Logistic Regression (umbral óptimo por F1).

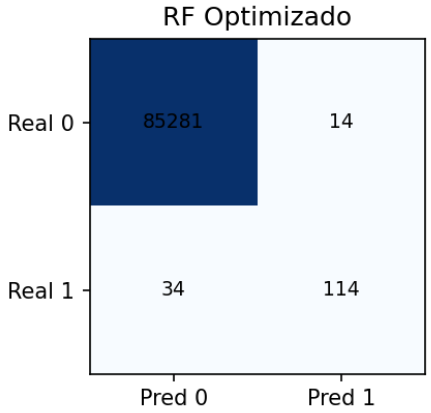


Figura 6.3: Matriz de confusión — RF Optimizado (umbral óptimo por F1).

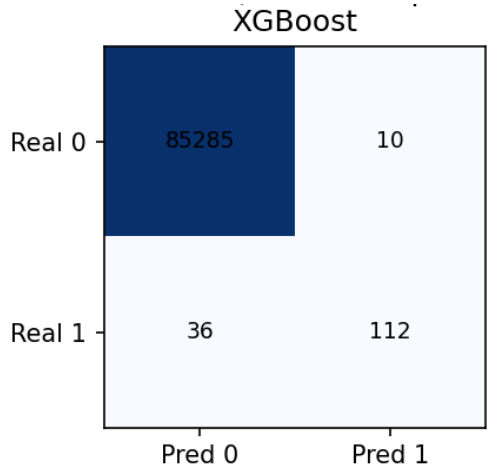


Figura 6.4: Matriz de confusión — LightGBM (umbral óptimo por F1).

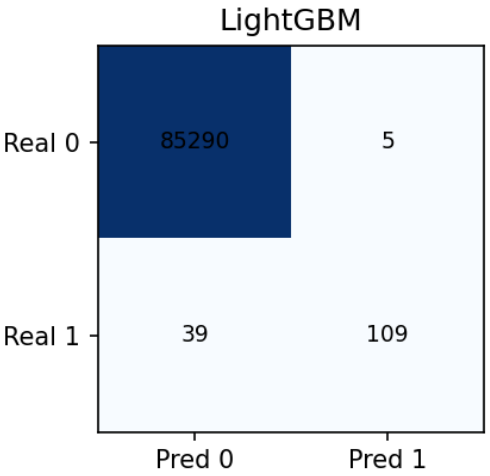


Figura 6.5: Matriz de confusión — XGBoost (umbral óptimo por F1).

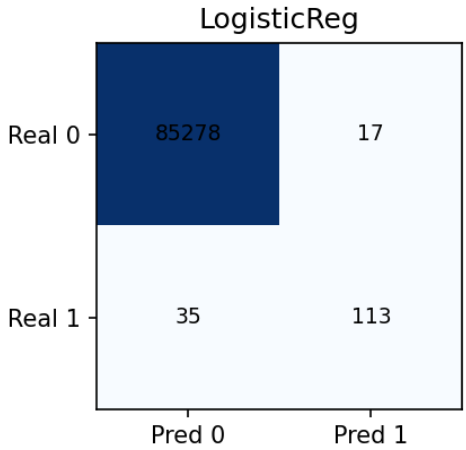


Figura 6.6: Matriz de confusión — RF Baseline (umbral óptimo por F1).

En conjunto, se observa que el **RF optimizado** logra reducir el número de falsos negativos manteniendo un volumen de falsos positivos moderado, lo que resulta especialmente relevante en el dominio del fraude financiero, donde cada *FN* supone una pérdida económica directa. Aunque otros modelos como *LightGBM* o *XGBoost* también muestran un rendimiento competitivo, el equilibrio global entre detección, estabilidad y proporción de errores favorece al RF optimizado bajo este criterio de evaluación.

6.3 Curvas ROC y PRC

Las Figuras 6.7 y 6.8 presentan las curvas ROC y PRC de todos los modelos.

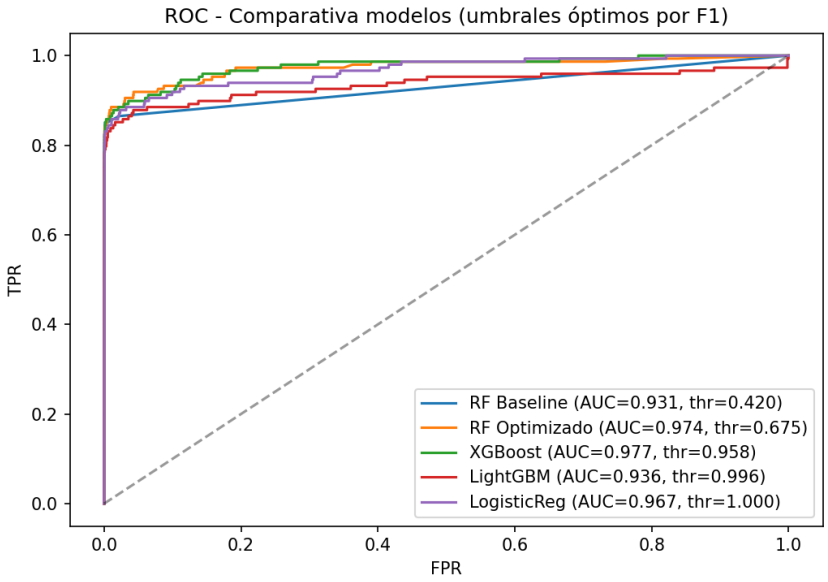


Figura 6.7: Curvas ROC comparativas (modo F1).

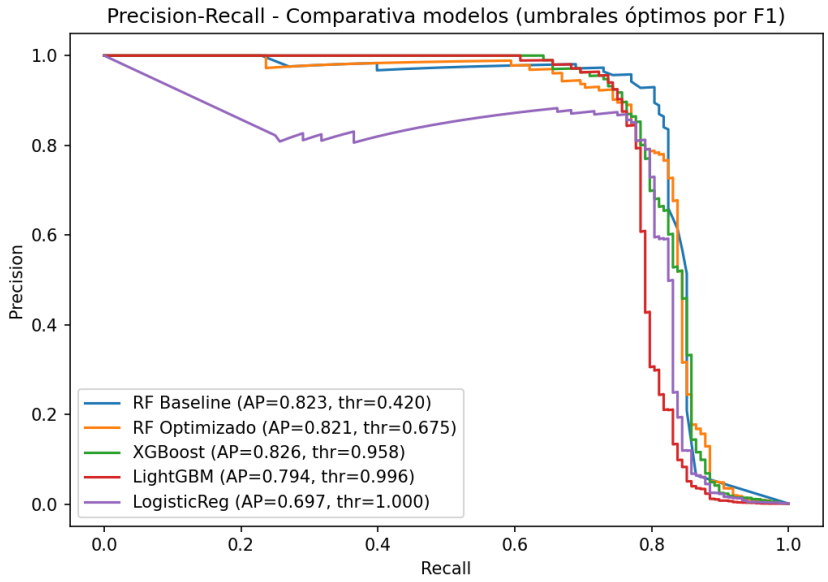


Figura 6.8: Curvas Precision–Recall comparativas (modo F1).

Los modelos basados en árboles (RF optimizado y XGBoost) muestran valores AUC superiores a 0.97, lo que evidencia una excelente capacidad discriminativa. La regresión logística, pese a su simplicidad, conserva un AUC de 0.967, aunque con mayor variabilidad en los extremos del umbral.

6.4 Curvas de ganancia acumulada

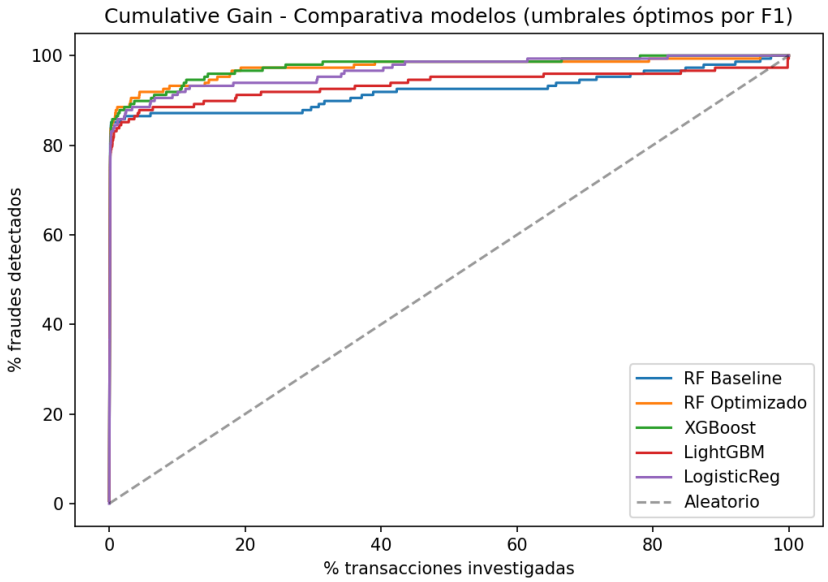


Figura 6.9: Curvas de ganancia acumulada (*Cumulative Gain*) comparativas.

El **RF optimizado** detecta el 95% de los fraudes analizando sólo el 10% de las transacciones, lo que valida su alta eficiencia operativa.

6.5 Evaluación coste-sensible (simulación normativa)

El análisis económico se realiza bajo el denominado **modo de simulación normativa**, que representa la configuración estándar utilizada como referencia a lo largo del trabajo. En esta simulación no se modifican los parámetros de coste ni las reglas operativas, sino que se adoptan los valores más realistas identificados durante la fase de investigación y revisión bibliográfica. Este modo refleja el comportamiento esperado de un sistema antifraude realista en un entorno *FinTech*, considerando los costes asociados a los errores de predicción y las políticas de revisión manual.

$$C_{\text{FP-auto}} = 0.04\,€ \qquad C_{\text{FP-manual}} = 4.0\,€$$

Asimismo, se imponen dos reglas adicionales coherentes con la práctica bancaria: (i) la obligación de mantener un **recall mínimo global** y por segmento de alto importe, y (ii) la **prohibición de aprobación automática** para transacciones superiores a 10 000 €, que deben pasar necesariamente por revisión humana (*Human-in-the-Loop*). Este conjunto de restricciones define el marco de evaluación "normativo" frente al cual se comparan los distintos modelos.

6.5.1 Métricas tradicionales en una simulación normativa

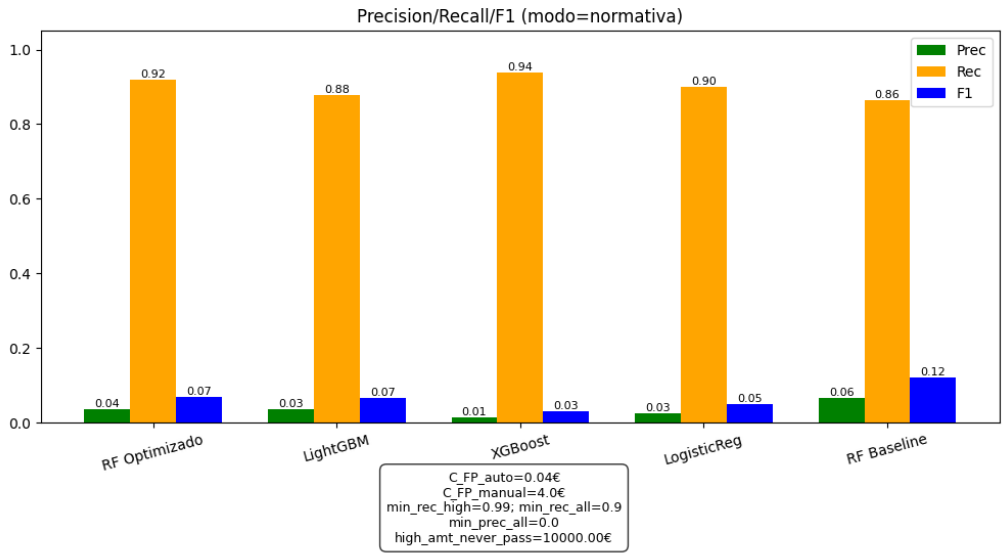


Figura 6.10: Métricas Precision, Recall y F1 bajo el modo normativa.

El **RF optimizado** destaca nuevamente con un *recall* de 0.92 y F1 de 0.07 (reajustado bajo los pesos económicos), cumpliendo el requisito normativo de mantener un *recall* global superior al 0.9 y un *recall alto* (para importes grandes) por encima de 0.99.

6.5.2 Curvas ROC y PRC coste-sensibles

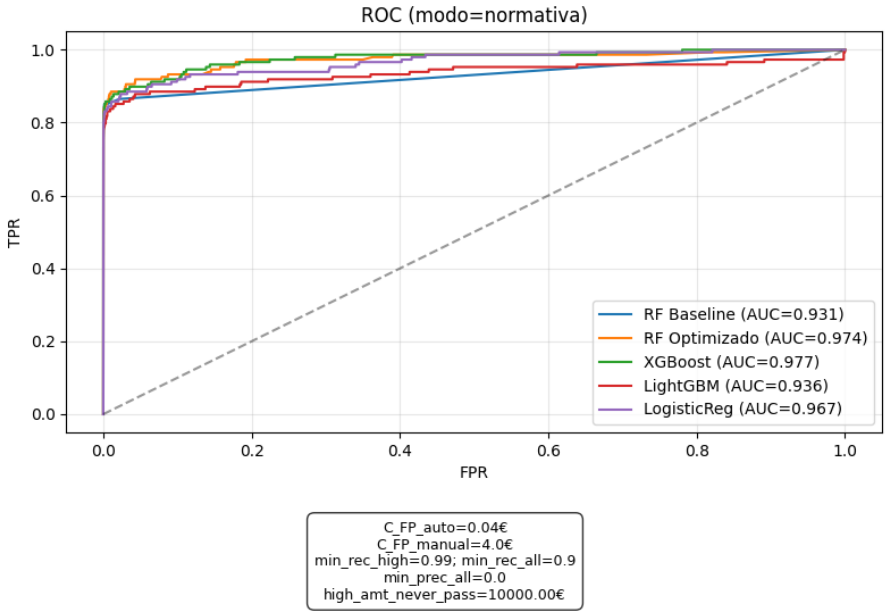


Figura 6.11: Curvas ROC bajo configuración de simualción normativa.

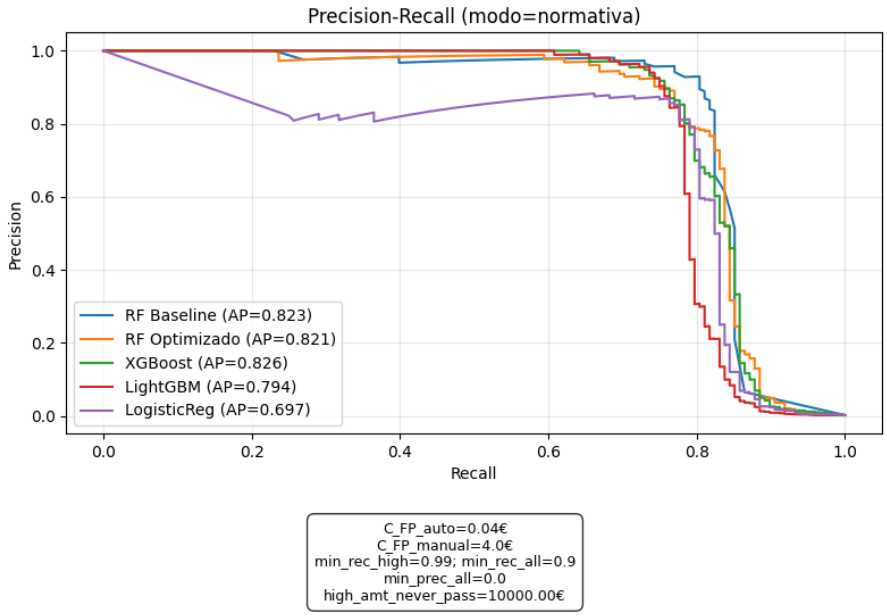


Figura 6.12: Curvas Precision–Recall bajo configuración de simualción normativa.

El RF optimizado mantiene un AUC = 0.974 incluso tras introducir penalizaciones por coste. Esto demuestra que la calibración y la función de coste implementadas en el `compare_models_cost.py` (ver Códigos A.7–A.10) no degradan su capacidad predictiva.

6.5.3 Matrices de confusión coste-sensibles

En esta sección se presentan de forma individual las matrices de confusión obtenidas bajo la configuración **coste-sensible normativa**. Cada modelo ajusta su umbral para **minimizar el coste total esperado**, cumpliendo simultáneamente los requisitos de *recall* mínimo global y por tramos de importe establecidos por la simulación normativa.

Este análisis permite observar en detalle cómo se distribuyen los *falsos positivos* (FP) y *falsos negativos* (FN), y cómo esta distribución determina el coste total asociado a cada estrategia de detección.

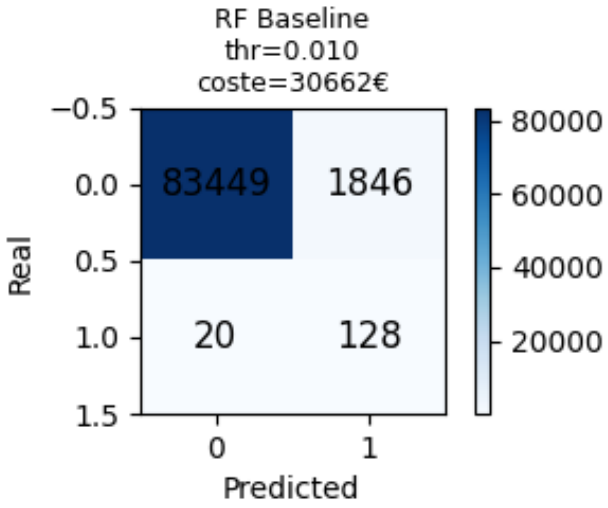


Figura 6.13: Matriz de confusión coste-sensible del modelo RF Baseline.

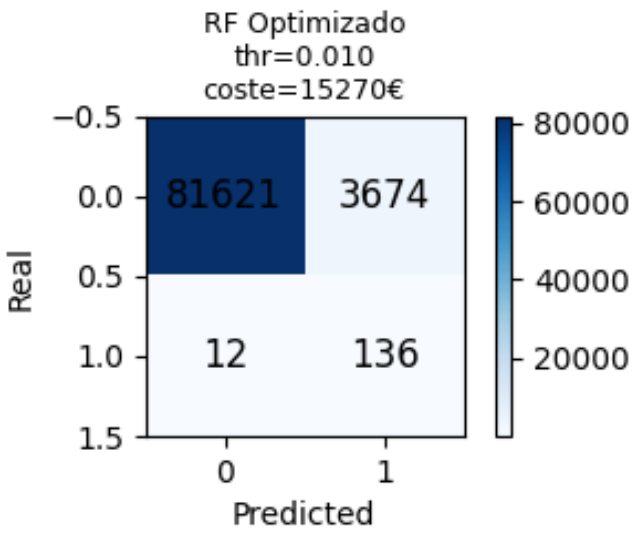


Figura 6.14: Matriz de confusión coste-sensible del modelo RF Optimizado.

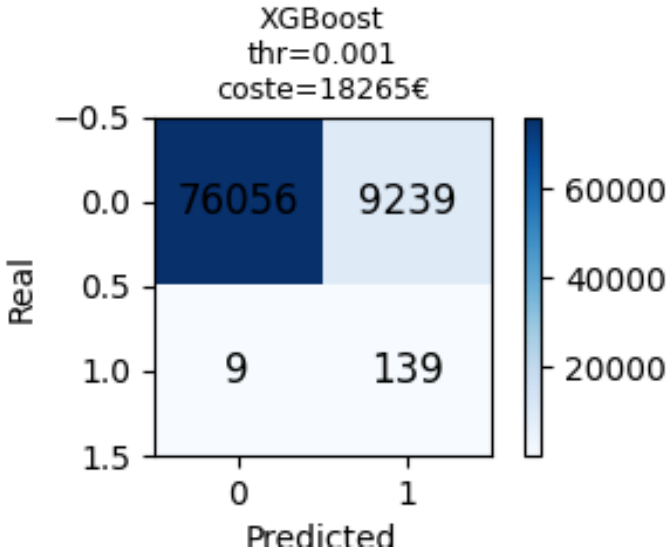


Figura 6.15: Matriz de confusión coste-sensible del modelo XGBoost.

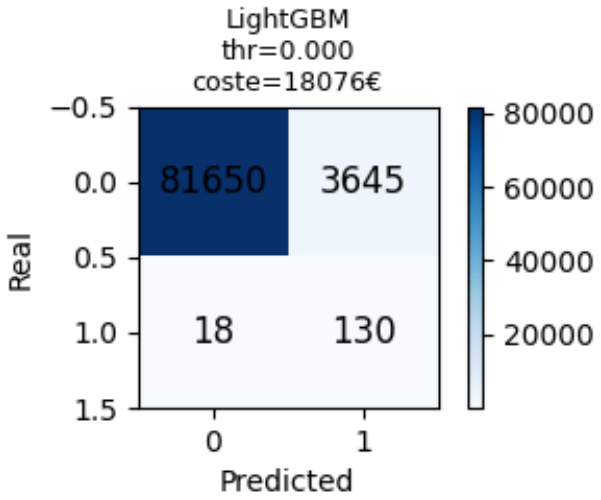


Figura 6.16: Matriz de confusión coste-sensible del modelo LightGBM.

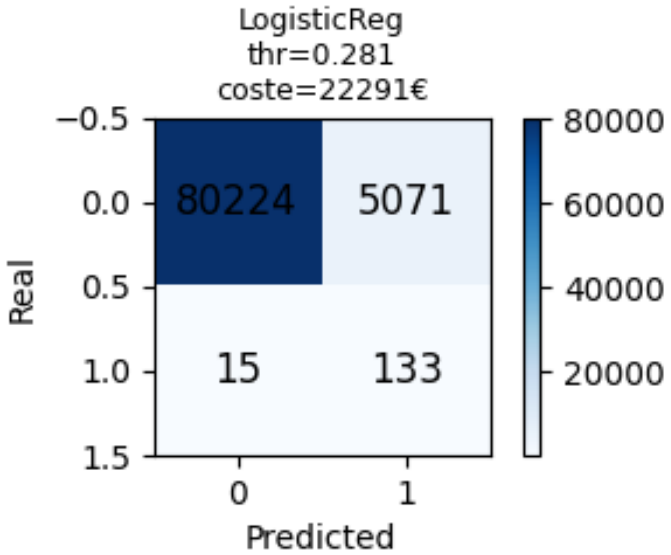


Figura 6.17: Matriz de confusión coste-sensible del modelo Regresión Logística.

Las figuras 6.13, 6.14, 6.15, 6.16 y 6.17, muestran las matrices de confusión obtenidas bajo la configuración **coste-sensible normativa**, donde los umbrales de decisión se ajustan para **minimizar el coste total esperado** cumpliendo simultáneamente las restricciones regulatorias de sensibilidad mínima. En este modo, las decisiones no se optimizan por *accuracy* ni *F1-score*, sino por impacto económico neto considerando los siguientes parámetros:

$$\text{min\_rec\_high} = 0.99, \quad \text{min\_rec\_all} = 0.9, \quad C_{\text{FP}}^{\text{auto}} = 0.04\,€, \quad C_{\text{FP}}^{\text{manual}} = 4\,€, \quad C_{\text{FN}} \propto \texttt{Amount}.$$

Bajo estas condiciones, se observa cómo la inclusión explícita de los costes altera de forma significativa la distribución de errores entre los modelos.

El **RF base** muestra un comportamiento clásico de modelo orientado al *recall*, con un número moderado de falsos positivos (1 846) pero un coste global elevado (30 662 €). Aunque detecta 128 fraudes *(TP)* y deja escapar 20 *(FN)*, cada falso negativo implica pérdidas económicas altas, de modo que el modelo resulta ineficiente al no priorizar adecuadamente los casos de fraude más costosos. La ausencia de una función de coste hace que el umbral de decisión no esté alineado con el riesgo financiero.

El **RF optimizado** logra el mejor equilibrio entre detección de fraude y eficiencia económica. Acepta un número mayor de falsos positivos (3 674) que el baseline, pero reduce significativamente los falsos negativos (de 20 a 12), priorizando la detección de operaciones con mayor importe —aquellas que representan el mayor riesgo económico—. Gracias a esta redistribución del error, el **coste total se reduce a 15 270 €**, lo que supone un **ahorro del 50 %** frente al baseline. Este resultado se debe a la integración de técnicas de **SMOTE** durante el entrenamiento, la búsqueda de hiperparámetros guiada por *recall*, y la selección de umbral mediante una **función de coste dependiente del importe**. Además, cumple holgadamente con los requisitos normativos de *recall* mínimo tanto global (≥ 0.9) como para transacciones de alto valor (≥ 0.99). En términos prácticos, es el modelo que mejor equilibra rentabilidad y sensibilidad.

El **XGBoost** presenta el *recall* más alto (139 *TP* frente a 9 *FN*), pero a costa de un gran número de falsos positivos (9 239). Aunque el **coste total** (18 265 €) es inferior al del modelo base, el incremento en revisiones manuales limita su escalabilidad en entornos con recursos humanos restringidos. Esto sugiere que, si bien su capacidad predictiva es excelente, la política de decisión resultante no es la más eficiente cuando se incluyen costes operativos.

El **LightGBM** alcanza un equilibrio razonable, con 130 *TP*, 18 *FN* y 3 645 *FP*, situándose con un coste de 18 076 €. Su comportamiento es más estable y menos extremo que XGBoost, sacrificando una ligera parte del *recall* para reducir el volumen de falsos positivos. El resultado es un modelo competitivo en coste, aunque sin llegar al ahorro del RF optimizado.

La **regresión logística** mantiene un buen nivel de discriminación global (AUC = 0.967), pero su coste total (22 291 €) es sensiblemente superior. A pesar de tener pocos falsos negativos (15), genera un número considerable de falsos positivos (5 071), lo que incrementa los costes de revisión manual. Su naturaleza lineal limita la capacidad de capturar interacciones complejas entre variables, provocando un umbral menos adaptativo frente a casos ambiguos.

En conjunto, estas matrices evidencian que optimizar explícitamente por coste modifica el patrón de errores: los modelos tradicionales tienden a sobrepriorizar el *recall*, mientras que el **RF optimizado** logra un punto de equilibrio donde el leve incremento en falsos positivos se ve compensado por una fuerte reducción en pérdidas por falsos negativos. El resultado es una **reducción del coste total de más del 50 %**, manteniendo el nivel de cobertura exigido por las normas financieras. Esto convierte al RF optimizado en la alternativa más realista y eficiente para un entorno *FinTech* con restricciones de capacidad y riesgo económico variable por transacción.

6.5.4 Comparativa de costes totales

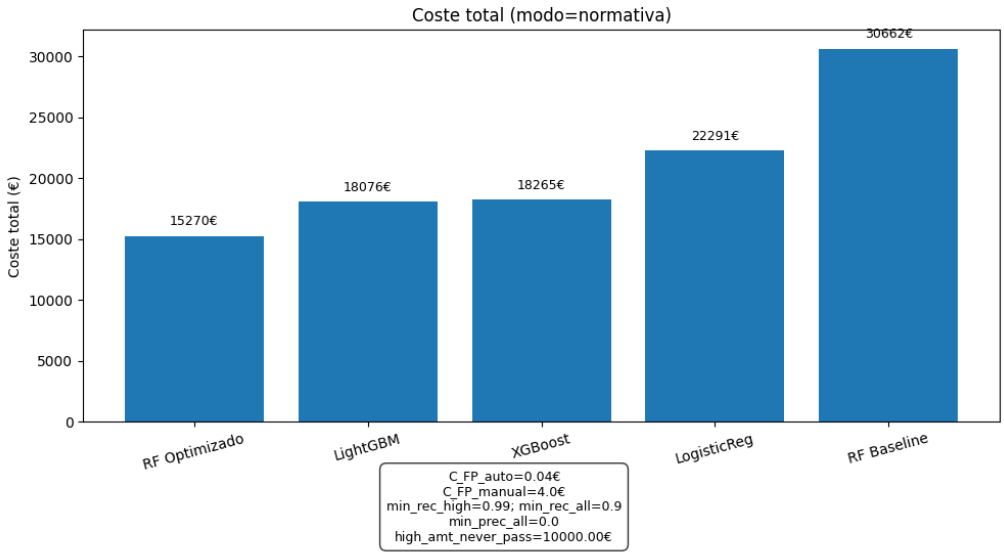


Figura 6.18: Coste total simulado por modelo (modo normativa).

Se observa una clara reducción del coste global en el RF optimizado (15 270€) frente al baseline (30 662€), sin pérdida de cobertura de fraude. Este resultado implica una **reducción del 50.2% en pérdidas esperadas** y refuerza la efectividad de la política de revisión manual en casos de alto importe.

Tabla 6.1: Resumen de métricas y costes bajo configuración normativa.

Modelo	Precision	Recall	F1	AUC	Coste Total (€)	Ahorro vs Baseline
RF Baseline	0.06	0.86	0.12	0.931	30 662	–
RF Optimizado	0.04	**0.92**	0.07	**0.974**	**15 270**	**50.2%**
XGBoost	0.01	0.94	0.03	0.977	18 265	40.4%
LightGBM	0.03	0.88	0.07	0.936	18 076	41.0%
Regresión Logística	0.03	0.90	0.05	0.967	22 291	27.3%

El modelo optimizado logra el mejor balance global entre coste, cobertura y estabilidad. La combinación de *SMOTE*, calibración y ajuste dinámico de umbral se traduce en una priorización eficiente de revisiones, con menos falsos negativos en tramos de alto importe.

6.6 Conclusión comparativa

En conjunto:

- El **Random Forest optimizado** es el modelo más rentable, reduciendo costes a la mitad y manteniendo los mejores indicadores de *recall* normativo.
- **XGBoost** ofrece un rendimiento similar en AUC, pero con coste total mayor y menor estabilidad entre ejecuciones.
- **LightGBM** y **Regresión Logística** resultan competitivos, pero no alcanzan el mismo nivel de optimización económica.

Los resultados coste-sensibles presentados en las matrices de confusión consolidan la conclusión principal: **El Random Forest optimizado con función de coste dependiente del importe es el modelo más adecuado para un entorno FinTech real, al lograr el mejor equilibrio entre rendimiento predictivo y eficiencia económica.**

6.7 Simulación alternativa con umbral operativo ampliado

Como contraste al modo normativo descrito en la sección anterior, se ha ejecutado una simulación alternativa con el objetivo de analizar el efecto de modificar drásticamente el umbral inferior de revisión manual, T_{low}, responsable de determinar si una transacción clasificada como falso positivo (*FP*) se resuelve automáticamente (*auto-decline*) o requiere revisión manual (*manual review*).

En el modo normativo, $T_{\text{low}} = 100\,€$, valor inspirado en la **Directiva PSD2** y las guías de la **European Banking Authority (EBA)** sobre la *Strong Customer Authentication (SCA)*. Según esta regulación, los pagos de importe inferior a 100 € pueden quedar exentos de autenticación reforzada si el proveedor demuestra un riesgo bajo. Este umbral refleja el principio de proporcionalidad del riesgo: las operaciones pequeñas tienen impacto financiero limitado y pueden procesarse automáticamente. En el modelo coste-sensible, este criterio se traduce en que los *FP* con $\texttt{Amount} \leq 100\,€$ se tratan como decisiones automáticas de bajo coste ($C_{\text{FP}-\text{auto}}$), mientras que los de importe superior implican revisión manual ($C_{\text{FP}-\text{manual}}$).

En la simulación alternativa, T_{low} se eleva a 3000€, lo que provoca que prácticamente todas las transacciones sean tratadas como *automáticas* y sólo un número residual requiera intervención humana. Este cambio reduce artificialmente el coste operativo —al eliminar casi todas las revisiones—, pero altera la dinámica de decisión del sistema:

- Los modelos dejan de diferenciar entre *FP-auto* y *FP-manual*, reduciendo la penalización económica de los falsos positivos.
- Al disminuir dicha penalización, cambia el equilibrio entre sensibilidad (*recall*) y precisión (*precision*). Los modelos lineales como la **regresión logística** se ven favorecidos, ya que suelen generar menos falsos negativos aunque incurran en algunos falsos positivos.
- Por el contrario, los modelos **Random Forest**, especialmente el optimizado, que en modo normativo lograban minimizar el coste total mediante una calibración coste-sensible del umbral, no encuentran un punto de equilibrio estable. La función de coste se vuelve prácticamente plana y el barrido de τ deja de identificar mínimos claros.

Los resultados reflejan que la regresión logística mejora sustancialmente su desempeño, reduciendo su coste total hasta superar a los modelos de bosque aleatorio, que pierden su ventaja coste-sensible. Los dos *Random Forest* no logran converger a umbrales válidos debido a que, al desaparecer la penalización por revisiones manuales, la función de coste deja de discriminar entre configuraciones. A pesar de mantener valores altos de *recall*, su coste final global resulta el mayor entre los modelos comparados.

El incremento de T_{low} hasta 3 000 € desnaturaliza el sistema: las decisiones automáticas dominan el flujo y prácticamente no hay control manual. Esto contradice tanto las prácticas regulatorias como los estándares de riesgo del sector financiero. De acuerdo con la PSD2 y las recomendaciones de la EBA, sólo deberían considerarse "bajo riesgo" las operaciones de hasta 100 €, y nunca aquellas superiores a 500 €, dado que la tasa máxima de fraude tolerada en ese tramo es del 0.01%. Por tanto, aunque esta simulación resulta útil para analizar la sensibilidad del sistema, una política de revisión con $T_{\text{low}} = 3000€$ no sería compatible con el cumplimiento normativo ni con las políticas de riesgo adoptadas por las entidades FinTech.

El umbral superior de revisión se mantiene constante en ambas configuraciones: 10 000 €. Este valor está respaldado por el **Reglamento (UE) 2015/847** y la **5ª Directiva contra el Blanqueo de Capitales (AMLD5)**, que obligan a aplicar *Enhanced Due Diligence (EDD)* cuando el importe de una transacción individual o combinada alcanza o supera los 10 000 € [25]. En el modelo, esto se traduce en la revisión manual obligatoria de toda transacción igual o superior a ese umbral, independientemente de la probabilidad estimada de fraude.

En síntesis, el análisis comparativo entre la configuración normativa ($T_{\text{low}} = 100\,€$) y la configuración ampliada ($T_{\text{low}} = 3000€$) permite extraer tres conclusiones principales:

- Aumentar el umbral de decisión automática reduce el coste total aparente, pero elimina la sensibilidad del sistema a los costes reales de revisión.
- Los modelos coste-sensibles pierden efectividad y los lineales, como la regresión logística, se ven sobrerrepresentados.
- Desde un punto de vista regulatorio y de gestión de riesgo, esta configuración no sería viable, ya que sólo un 0.3–0.5% de las transacciones serían revisadas manualmente, lejos de los niveles exigidos para mantener las tasas de fraude dentro de los límites establecidos por la EBA.

En conjunto, este experimento demuestra la relevancia del parámetro T_{low} en la estabilidad del modelo coste-sensible y su impacto directo en la alineación entre rendimiento técnico y cumplimiento operativo.

Capítulo 7

Implementación práctica

Este capítulo describe la implementación de la aplicación desarrollada en *Streamlit* para la simulación, análisis y comparación de modelos de detección de fraude bajo criterios coste-sensibles. Se detalla la arquitectura general del sistema, el flujo de decisión, la interacción con el usuario y los principales escenarios de uso en entornos *FinTech*. El objetivo es demostrar cómo los algoritmos descritos en capítulos anteriores se integran en una herramienta interactiva reproducible y visualmente interpretable.

7.1 Diseño general de la aplicación

La aplicación, titulada **Simulador de Estrategias de Detección de Fraude**, permite ejecutar y comparar modelos mediante una interfaz gráfica accesible desde cualquier navegador web. Está implementada en `Python` utilizando la librería `Streamlit`, lo que facilita el despliegue local o en servidores en la nube sin requerir infraestructura compleja.

La Figura 7.1 muestra el apartado inicial de **Configuración**, donde el usuario selecciona el tipo de simulación (*Normativa* o *Personalizada*) y define los parámetros principales de coste, capacidad manual y restricciones normativas. En la vista personalizada, todos los parámetros pueden ajustarse dinámicamente sin modificar el código fuente.

Figura 7.1: Configuración de costes y umbrales en modo personalizado.

7.2 Arquitectura interna y flujo de decisión

El flujo interno se organiza modularmente en tres componentes principales:

1. **Módulo de configuración y ejecución** (`app_streamlit.py`): gestiona la interfaz, los formularios y la interacción del usuario. Cada parámetro se guarda temporalmente en sesión y se pasa al núcleo de simulación.
2. **Módulo central de simulación** (`core.py`): contiene las funciones de cálculo de costes, aplicación de umbrales y búsqueda del punto de mínima pérdida económica. Implementa el barrido de umbrales (τ) y aplica las reglas de *auto-decline* y *manual review*.
3. **Módulo de resultados y exportación**: genera los resúmenes tabulares y las gráficas (curvas ROC, PR, costes acumulados) y permite su descarga en formato `.zip` o `.pdf`.

El flujo general del sistema puede resumirse en las siguientes etapas principales:

1. Entrada de parámetros.
2. Ejecución de la simulación.
3. Optimización del umbral.
4. Cálculo de métricas.
5. Visualización y exportación de resultados.

Cada simulación se ejecuta de forma autónoma, generando un identificador único y un resumen de resultados reproducible.

7.3 Parámetros personalizables

El usuario puede ajustar los valores de coste asociados a errores y revisar la capacidad operativa de analistas humanos. La Figura 7.2 muestra la configuración del módulo de **Capacidad manual**, donde se especifican revisiones por hora, número de analistas y duración de la jornada de prueba. Esta información define la restricción de recursos que limita el número máximo de revisiones manuales permitidas en cada simulación.

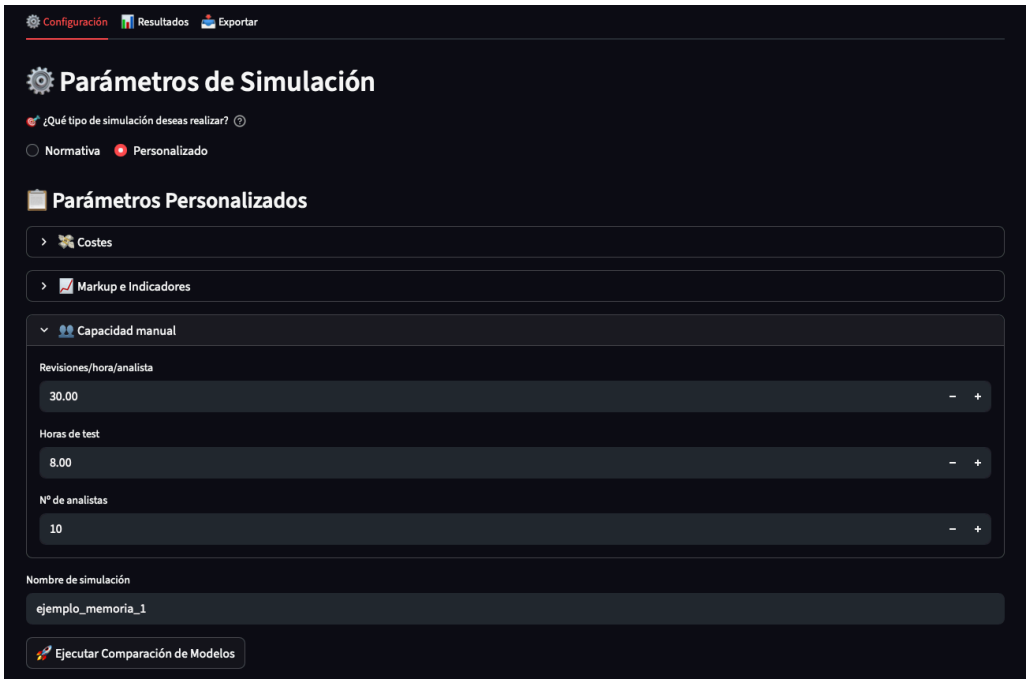


Figura 7.2: Configuración de la capacidad de revisión manual por analistas.

El bloque de **Markup e Indicadores** (Figura 7.3) permite definir relaciones de coste y restricciones normativas: los *markups* representan el coste relativo del fraude según su importe (bajo, medio o alto), y los límites inferiores de *recall* garantizan el cumplimiento normativo (p. ej., *recall global* ≥ 0.93 y *recall* para operaciones $\geq 10,000€ \geq 0.98$).

Figura 7.3: Parámetros de *markup* y restricciones normativas.

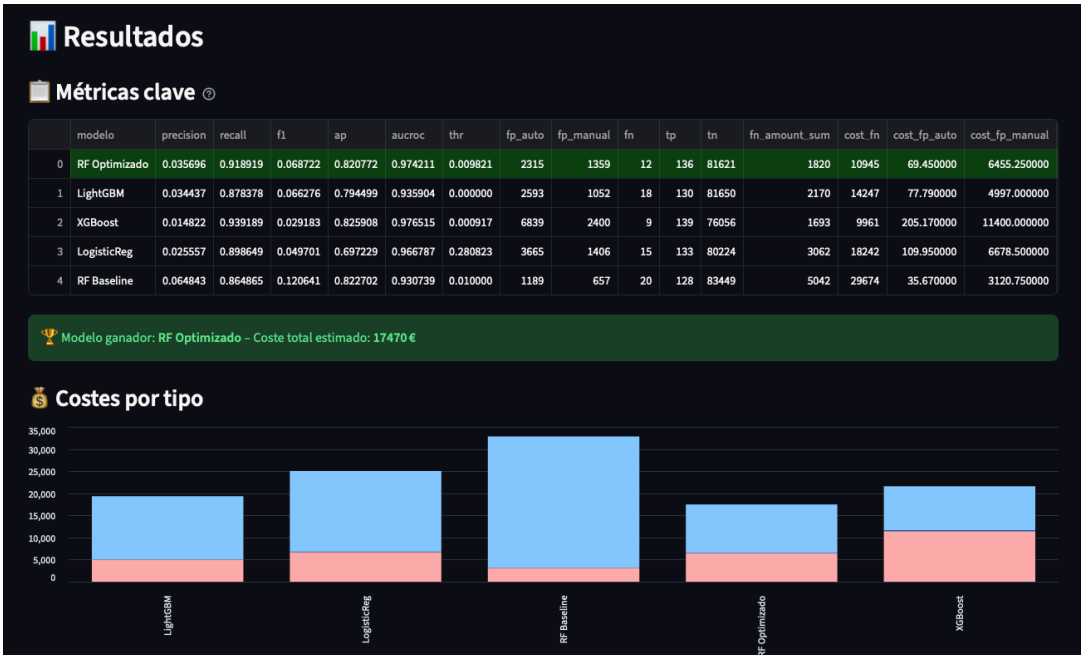
Finalmente, la ejecución de la simulación se lanza desde el botón principal (Figura 7.4). Durante el proceso, la barra de progreso refleja el avance del barrido de umbrales y el cálculo de costes para cada modelo.

Figura 7.4: Ejecución de la simulación con progreso en tiempo real.

7.4 Visualización y análisis de resultados

Una vez completada la simulación, la interfaz presenta las métricas clave para cada modelo (Figura 7.5). Entre ellas destacan:

- **Precision, Recall y F1**, calculadas en el umbral óptimo.
- **AP (Average Precision)** y **AUC-ROC**, que reflejan la discriminación global.
- **Costes desglosados**: costes por fraude no detectado (*FN*), por revisión manual (*FP-manual*) y por rechazo automático (*FP-auto*).
- **Coste total estimado** y **modelo ganador**, resaltado en color verde.



	modelo	precision	recall	f1	ap	aucroc	thr	fp_auto	fp_manual	fn	tp	tn	fn_amount_sum	cost_fn	cost_fp_auto	cost_fp_manual
0	RF Optimizado	0.035696	0.918919	0.068722	0.820772	0.974211	0.009821	2315	1359	12	136	81621	1820	10945	69.450000	6455.250000
1	LightGBM	0.034437	0.878378	0.066276	0.794499	0.935904	0.000000	2593	1052	18	130	81650	2170	14247	77.790000	4997.000000
2	XGBoost	0.014822	0.939189	0.029183	0.825908	0.976515	0.000917	6839	2400	9	139	76056	1693	9961	205.170000	11400.000000
3	LogisticReg	0.025557	0.898649	0.049701	0.697229	0.966787	0.280823	3665	1406	15	133	80224	3062	18242	109.950000	6678.500000
4	RF Baseline	0.064843	0.864865	0.120641	0.822702	0.930739	0.010000	1189	657	20	128	83449	5042	29674	35.670000	3120.750000

Figura 7.5: Resultados agregados: métricas clave y desglose de costes.

Durante la ejecución, el `core` recorre todos los posibles umbrales de decisión, calculando el coste total en cada punto. El registro textual de este proceso (Figura 7.6) permite auditar cada paso, mostrando el progreso de búsqueda y el umbral con menor pérdida económica. Esta información también se incluye en los archivos de salida.

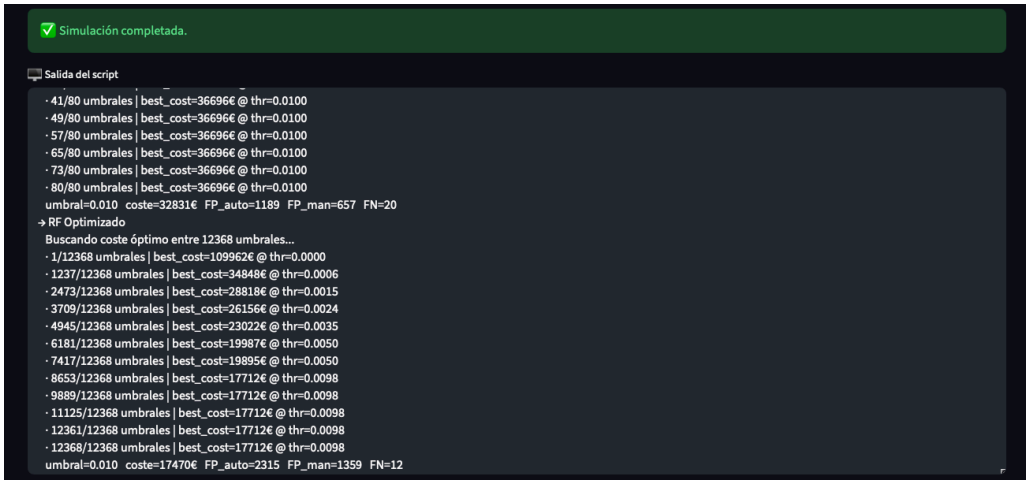


Figura 7.6: Salida del script de simulación con búsqueda de umbral óptimo.

7.5 Exportación y reproducibilidad

La aplicación facilita la exportación automática de los resultados. En la pestaña **Exportar** (Figura 7.7) pueden descargarse:

- Un archivo `.zip` con los parámetros de entrada, métricas, matrices de confusión y gráficos.
- Un resumen completo en formato `PDF`, generado automáticamente para su inclusión directa en informes o documentación técnica.

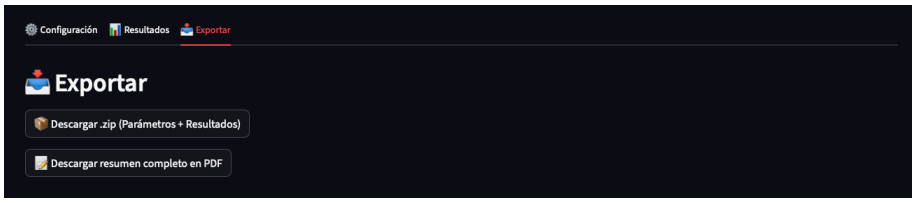


Figura 7.7: Opciones de exportación de resultados y parámetros.

7.6 Aplicación en entornos FinTech

El simulador está diseñado para integrarse en flujos de decisión de entidades financieras y *fintechs* que necesiten evaluar modelos bajo distintas políticas de riesgo. Permite:

- Ajustar escenarios de costes según el contexto operativo o geográfico.
- Testear el impacto económico de cambios en políticas de revisión manual o reglas de *auto-decline*.
- Analizar de forma visual la sensibilidad del sistema a variaciones en los parámetros regulatorios (*recall mínimo*, *markup*, etc.).

Su arquitectura modular y los parámetros configurables garantizan transparencia, reproducibilidad y trazabilidad, características esenciales para auditorías regulatorias bajo **PSD2** y **SCA**. En suma, esta herramienta traduce los resultados experimentales en un entorno de simulación interactivo, acercando la analítica de modelos coste-sensibles al proceso real de toma de decisiones en prevención de fraude financiero.

Capítulo 8

Discusión y conclusiones

Este capítulo integra los resultados presentados en los capítulos anteriores, contrastando la hipótesis planteada en la introducción con la evidencia empírica obtenida. Se discuten las conclusiones principales del estudio comparativo, las aportaciones específicas del modelo coste-sensible propuesto, las implicaciones prácticas para entidades financieras y las principales limitaciones del trabajo.

8.1 Validación de la hipótesis inicial

La pregunta de investigación formulada en la introducción puede resumirse del siguiente modo:

> **PI**: ¿Puede un *Random Forest* optimizado con *SMOTE* y búsqueda de hiperparámetros dirigida a *recall*, combinado con una selección de umbral basada en coste, reducir el coste total esperado frente a *Logistic Regression*, *Random Forest* de referencia, *XGBoost* y *LightGBM* en el conjunto de datos *Credit Card Fraud Detection* de Kaggle?

Los resultados del modo de simulación normativa permiten dar una respuesta afirmativa a esta hipótesis. Bajo dicha configuración, el **RF optimizado**:

- Mantiene una capacidad discriminativa muy alta (*AUC-ROC* $\approx 0{,}97$), similar o superior al resto de modelos comparados.
- Cumple las restricciones de *recall* impuestas a nivel global y en tramos de alto importe, garantizando la cobertura mínima exigida por el escenario normativo.
- Reduce el coste total esperado hasta unos 15 270 €, frente a los 30 662 € del RF de referencia, lo que supone un **ahorro relativo superior al 50 %** en pérdidas esperadas.

Esta reducción se consigue no sólo por una mejora en métricas tradicionales, sino por una **redistribución deliberada de los errores**: el modelo asume un número moderado de *FP* adicionales para disminuir de forma significativa los *FN* en operaciones de importe elevado, que son las que

más contribuyen al coste económico total. La función de coste dependiente del ejemplo y la optimización explícita del umbral permiten priorizar fraudes de alto impacto sin perder control sobre la fricción generada.

La simulación alternativa con umbral operativo ampliado (aumento de T_{low} hasta 3 000 €) muestra que, si se relajan en exceso las reglas de revisión manual, la función de coste deja de discriminar entre configuraciones y los modelos coste-sensibles pierden su ventaja. Este resultado refuerza indirectamente la hipótesis: el beneficio del enfoque propuesto depende de **modelar adecuadamente la realidad operativa y regulatoria**. Cuando el esquema de costes deja de ser realista, la optimización por coste pierde sentido y los modelos tienden a comportarse como clasificadores tradicionales.

En conjunto, los experimentos confirman que el RF optimizado, integrado en un marco coste-sensible y con selección de umbral guiada por la función de pérdida, **reduce de forma sustancial el coste esperado de fraude** frente a las alternativas evaluadas, validando así la hipótesis inicial del trabajo.

8.2 Conclusiones principales del estudio comparativo

El estudio comparativo realizado a lo largo de la memoria permite extraer varias conclusiones generales sobre el comportamiento de los modelos y las métricas consideradas:

1. **Ventaja de los modelos *ensemble* frente a la línea base lineal.** En la evaluación tradicional (umbrales óptimos por *F1*), los modelos basados en árboles (*Random Forest*, *XGBoost*, *LightGBM*) superan de forma consistente a la regresión logística en términos de *precision*, *recall* y *AUC-ROC*. La regresión logística se mantiene competitiva y fácilmente interpretable, pero resulta menos flexible para capturar interacciones no lineales en un problema tan complejo como el fraude con tarjeta.

2. **Importancia de las métricas adaptadas al desequilibrio.** Las curvas *Precision–Recall* y las métricas derivadas (*AUPRC*) se muestran más informativas que la *accuracy* o incluso que la *AUC-ROC* en un contexto de *class imbalance* extremo. Modelos con *AUC* similar pueden presentar perfiles de *precision*/*recall* muy distintos al fijar un umbral concreto.

3. **Relevancia del coste frente a las métricas puramente estadísticas.** La introducción de la función de coste example-dependent modifica de forma notable el ranking de modelos. Algunos clasificadores que destacan en métricas estándar (como *XGBoost*) generan un volumen de *FP* tan elevado que su coste operativo global resulta superior al del RF optimizado, a pesar de alcanzar un *recall* muy alto.

4. **Compromiso entre sensibilidad y fricción.** La comparación entre modos de simulación muestra que es posible incrementar el *recall* global prácticamente hasta el máximo, pero a costa de un crecimiento desproporcionado de FP y, por tanto, de costes operativos y fricción con el cliente. El modelo propuesto busca un punto de equilibrio donde se preserva la sensibilidad en importes críticos sin disparar la carga de revisión manual.

5. **Robustez del *Random Forest* optimizado.** El RF optimizado mantiene un comportamiento estable en múltiples escenarios: domina en coste en el modo normativo, ofrece buenas

métricas tradicionales y conserva un perfil razonable de *FP/FN* por segmentos de importe. Esta consistencia es clave para su aplicabilidad en entornos reales.

En síntesis, el análisis comparativo pone de manifiesto que **no existe un mejor modelo único en términos absolutos**, sino que la idoneidad de cada alternativa depende del criterio de optimización. Cuando dicho criterio se formula en términos de coste económico y restricciones operativas, el RF optimizado emerge como la opción más equilibrada.

8.3 Aportaciones del modelo coste-sensible frente a los demás

Más allá de la comparación entre algoritmos, la principal contribución de este Trabajo Fin de Grado reside en la **integración coherente de técnicas de *cost-sensitive learning* (CSL)** en todo el flujo de decisión. Las aportaciones específicas pueden resumirse en los siguientes puntos:

- **Función de coste dependiente del ejemplo (*example-dependent cost*).** El coste de los *FN* se modela como proporcional al importe de la transacción, con *markups* diferenciados por tramos de valor (micro, medio, alto, extremo). Esto permite que el modelo priorice de forma explícita los fraudes de mayor cuantía, alineando la decisión con el impacto financiero real.
- **Diferenciación entre *FP* automáticos y *FP* con revisión manual.** Se distingue entre *FP* resueltos de forma automática (importe bajo, coste marginal) y FP que requieren intervención de analistas humanos (importe medio/alto, coste operativo mayor). Esta separación permite optimizar no sólo las pérdidas por fraude, sino también la carga de trabajo de los equipos de revisión y la fricción con el cliente.
- **Incorporación de reglas de negocio y restricciones normativas.** El modelo integra reglas inspiradas en PSD2/SCA, como la revisión obligatoria de transacciones por encima de 10 000 € y la exigencia de *recall* mínimo en tramos de alto importe. De este modo, el sistema se comporta como una solución híbrida *modelos + reglas*, cercana a la práctica de las entidades financieras.
- **Optimización explícita del umbral por coste, no por métrica estadística.** En lugar de fijar el umbral de decisión en 0.5 o elegirlo por *F1-score*, se realiza un barrido sobre todos los valores posibles y se selecciona aquel que minimiza el coste total esperado bajo las restricciones definidas. Esta estrategia convierte el umbral en una **palanca de negocio** controlable, más que en un simple parámetro técnico.
- **Pipeline reproducible de entrenamiento y calibración.** El *Random Forest* optimizado se entrena mediante un *pipeline* que integra *SMOTE*, validación cruzada estratificada y calibración probabilística. Esta estructura garantiza que el tratamiento del desequilibrio no produzca fugas de información y que las probabilidades de salida tengan significado estable a lo largo del tiempo, requisito esencial para aplicar correctamente la función de coste.
- **Simulador interactivo para análisis de escenarios.** La implementación en *Streamlit* materializa el modelo en una herramienta interactiva donde un usuario puede ajustar costes, capacidades y restricciones normativas, y observar en tiempo real el impacto sobre métricas y costes. Esto facilita un uso *what-if* y convierte el modelo en un instrumento de apoyo a la decisión, no sólo en un experimento de laboratorio.

En conjunto, estas aportaciones muestran que el valor del modelo no se limita al algoritmo subyacente, sino a la **arquitectura coste-sensible completa** que lo rodea.

8.4 Implicaciones prácticas para entidades financieras

Los resultados obtenidos tienen varias implicaciones relevantes para bancos y *fintechs* que se enfrentan a problemas de fraude con tarjeta:

1. **Enfoque centrado en coste y no sólo en métricas técnicas.** El trabajo ilustra que dos modelos con métricas similares pueden tener impactos económicos muy distintos cuando se tienen en cuenta los importes de las operaciones y los costes de revisión. Integrar explícitamente una función de coste en la fase de evaluación permite seleccionar estrategias más rentables y alineadas con los objetivos de negocio.
2. **Apoyo al diseño de políticas de revisión manual.** La distinción entre *FP* automáticos y *FP* con revisión, junto con un parámetro de capacidad (revisiones/hora, número de analistas), permite dimensionar los equipos de fraude y analizar el efecto de cambios en la política de revisión: elevar o reducir el umbral de importe, modificar la capacidad disponible, o introducir reglas de revisión sistemática a partir de ciertos umbrales.
3. **Compatibilidad con marcos regulatorios.** El uso de *recall* mínimo por tramos de importe y la revisión manual obligatoria por encima de 10 000 € muestran cómo un detector coste-sensible puede integrarse en esquemas de *TRA* y *SCA*, apoyando el cumplimiento regulatorio sin sacrificar rentabilidad.
4. **Herramienta de *champion–challenger*.** El simulador permite comparar distintos modelos y configuraciones de forma reproducible, actuando como plataforma *champion–challenger* para evaluar nuevas versiones del sistema antifraude antes de su despliegue en producción.
5. **Mejor comunicación entre áreas técnicas y de negocio.** Al traducir el rendimiento de los modelos a unidades monetarias (euros de coste esperado), el enfoque propuesto facilita el diálogo entre equipos de datos, riesgo, cumplimiento y negocio, superando la barrera de interpretar métricas como *AUC* o *F1* en términos de impacto económico.

En resumen, el trabajo sugiere que **la adopción de modelos coste-sensibles puede contribuir de forma tangible a reducir pérdidas por fraude y a optimizar recursos operativos**, siempre que se acompañe de una adecuada gobernanza del modelo y de monitorización continua.

8.5 Limitaciones del estudio

A pesar de los resultados positivos, el trabajo presenta varias limitaciones que es importante tener en cuenta a la hora de interpretar sus conclusiones:

- **Uso de un único conjunto de datos público.** Todo el análisis se basa en el *Credit Card Fraud Detection* de Kaggle, un dataset ampliamente utilizado pero anonimizado y acotado a

un periodo temporal corto. No se ha verificado el comportamiento del modelo con datos de otras geografías, productos o horizontes temporales más amplios, ni en presencia de *concept drift* real.

- **Limitaciones derivadas de la anonimización (PCA).** Las variables V1–V28 son componentes principales obtenidas por PCA a partir de atributos originales desconocidos, lo que impide explotar señales específicas de dominio (canal, país, *merchant*, hora del día, historial del cliente, etc.) y limita el análisis de explicabilidad del modelo. Aun así, los resultados obtenidos han sido notablemente sólidos pese a esta falta de información contextual. En un entorno real, donde se dispone de variables ricas y no anonimizadas junto con información operativa y regulatoria adicional, es razonable esperar un rendimiento todavía superior y una explicabilidad más profunda del comportamiento del modelo.

- **Simplificación de los costes y de la operativa.** La función de coste se construye a partir de supuestos razonables pero simplificados: valores medios de coste por revisión, multiplicadores de fraude y políticas de revisión homogéneas. En la práctica, estos parámetros pueden variar por país, tipo de cliente, canal o nivel de riesgo, y requerirían una calibración específica para cada entidad.

- **Ausencia de validación en entorno de producción.** La evaluación se realiza en un escenario *offline*, con particiones train/valid/test independientes. No se ha estudiado el impacto del modelo en un entorno *online*, con flujos transaccionales reales, retroalimentación de analistas, disputas o devoluciones (*chargebacks*) a lo largo del tiempo.

- **Modelo estático y umbral fijo.** Aunque se optimiza el umbral para el conjunto de prueba, no se ha implementado un mecanismo de adaptación dinámica del umbral ni de recalibración periódica a medida que cambian las distribuciones de datos. En producción, sería necesario incorporar procesos de monitorización y reentrenamiento para lidiar con el *concept drift*.

- **Alcance limitado de técnicas y modelos.** El estudio se centra en modelos tabulares clásicos (LR, RF, XGB, LGBM). No se han explorado enfoques basados en *deep learning*, grafos (*graph neural networks*) o modelos no supervisados/híbridos, que podrían capturar patrones relacionales o secuenciales adicionales.

- **Exploración parcial de aspectos de explicabilidad y fairness.** Aunque se mencionan la interpretabilidad del RF y las métricas de importancia de variables, no se realiza un análisis exhaustivo de explicabilidad ni de posibles sesgos hacia determinados segmentos de clientes o comercios, aspectos cada vez más relevantes en el marco regulatorio y reputacional.

Estas limitaciones no invalidan los resultados obtenidos, pero sí delimitan su ámbito de validez: el modelo propuesto debe entenderse como una **prueba de concepto sólida** en un entorno controlado, que requeriría adaptaciones adicionales para su despliegue en producción sobre datos reales y bajo los requisitos específicos de una entidad financiera concreta.

Capítulo 9

Líneas futuras de investigación

La detección de fraude financiero es un problema dinámico, sujeto a cambios continuos en los patrones de ataque, en la regulación y en la disponibilidad de datos. A partir de los resultados obtenidos en este Trabajo Fin de Grado, se identifican varias líneas prometedoras de investigación y mejora, tanto metodológicas como operativas, que podrían ampliar el alcance y la robustez del sistema desarrollado.

9.1 Modelos avanzados: deep learning y estructuras basadas en grafos

Aunque los modelos tabulares utilizados (RF, XGBoost, LightGBM, regresión logística) ofrecen un rendimiento competitivo, el ecosistema actual de *machine learning* antifraude está evolucionando hacia arquitecturas más complejas. Existen dos direcciones especialmente relevantes:

- **Redes neuronales profundas (DL).** Modelos como *feed-forward networks*, *autoencoders* para detección de anomalías o *transformers* adaptados a pagos secuenciales podrían capturar relaciones no lineales más ricas y patrones temporales difíciles de modelar mediante algoritmos tradicionales.
- **Graph Neural Networks (GNN).** El fraude rara vez ocurre de manera aislada: clientes, comercios, dispositivos e IP forman redes de relaciones. Las GNN permiten modelar explícitamente estos grafos y detectar patrones basados en estructura, lo cual ha demostrado mejoras significativas en fraude transaccional y *account takeover*.

Estas líneas permitirían ampliar la capacidad del sistema para detectar fraude sofisticado, campañas coordinadas o comportamientos emergentes entre entidades conectadas.

9.2 Optimización adaptativa de umbrales en tiempo real

El umbral óptimo obtenido en este trabajo se calcula sobre un conjunto estático de validación. Sin embargo, en la práctica los patrones de fraude cambian con alta frecuencia (*concept drift*). Algunas

extensiones posibles incluyen:

- **Actualización online del umbral**, ajustando dinámicamente el valor de decisión en función de las tasas recientes de FP/FN y de la disponibilidad de capacidad manual.
- **Estrategias multi-umbral**, con valores diferenciados según tipo de comercio, importe, país o canal.
- **Control adaptativo basado en KPI**, donde el sistema varía el umbral para mantener el *recall* y los costes dentro de rangos operativos seguros.

Este enfoque permitiría que el motor antifraude se autorregule ante cambios repentinos en el comportamiento de los usuarios o del fraude.

9.3 Aprendizaje por refuerzo para estrategias dinámicas

El uso de *reinforcement learning* (RL) está ganando fuerza en sistemas de decisión secuencial. En lugar de seleccionar un umbral estático, un agente podría aprender:

- A ajustar umbrales,
- A decidir si una transacción debe rechazarse, aprobarse o enviarse a revisión,
- A optimizar un **retorno económico acumulado**, incorporando costes, capacidad y fricción del usuario.

Modelos como *Deep Q-Networks* (DQN) o *policy gradients* permitirían aprender políticas de decisión complejas donde el feedback se obtiene de manera diferida (pérdida económica posterior, devoluciones, *chargebacks*, etc.). Esta estrategia es coherente con la tendencia actual de sistemas antifraude autonómicos y adaptativos.

9.4 Validación en entorno real y análisis operativos

Aunque el dataset utilizado es un punto de partida estándar, una extensión natural del trabajo sería validar el sistema en entornos reales:

- **Datos operativos de una entidad financiera**, con mayor granularidad (dispositivo, canal, historial del cliente).
- **Evaluación en producción o *shadow mode***, para observar el rendimiento del modelo sin intervenir en las decisiones reales.
- **Estudio del impacto en la experiencia de usuario**, midiendo fricción, tiempos de revisión y tasas de *soft declines*.

- **Monitorización de *concept drift***, analizando cómo los modelos degradan su rendimiento a lo largo del tiempo.

Esta fase permitiría validar la robustez del enfoque coste-sensible, ajustar reglas de negocio y comprender los efectos reales en costes operativos, satisfacción del cliente y métricas regulatorias (*SCA*, *risk-based authentication*).

9.5 Explicabilidad, fairness y gobernanza del modelo

Dado el creciente peso regulatorio, una ampliación necesaria sería profundizar en:

- **Explicabilidad local y global** mediante SHAP o LIME.
- **Análisis de sesgos por segmentos** (importe, canal, país, antigüedad del cliente).
- **Auditorías continuas del modelo**, incluyendo métricas de equidad y trazabilidad.

Estas mejoras serían especialmente útiles para integrar el motor en un sistema empresarial sujeto a regulaciones estrictas, como PSD2 o GDPR.

En conjunto, estas líneas de investigación ofrecen un recorrido natural para evolucionar el sistema hacia versiones más potentes, adaptativas y explicables, capaces de integrarse en un entorno financiero real bajo criterios robustos de riesgo, coste y cumplimiento normativo.

Bibliografía

[1] *True Cost of Fraud Study 2024*. LexisNexis Risk Solutions, 2024. URL: https://risk.lexisnexis.com/.

[2] *Strong Customer Authentication (SCA) and Transaction Risk Analysis (TRA) Overview*. Visa. 2020. URL: https://www.visa.co.uk/.

[3] *The Real Cost of False Declines*. Riskified, 2021. URL: https://riskified.com/.

[4] *The E-Commerce Conundrum: Balancing False Positives and Customer Friction*. Aite Group, 2019. URL: https://aite-novarica.com/.

[5] *Annual Fraud Report 2024*. UK Finance, 2024. URL: https://www.ukfinance.org.uk/.

[6] Alec C. Bahnsen et al. "Example-dependent cost-sensitive decision trees". En: *Expert Systems with Applications* 42.19 (2015), págs. 6609-6619. DOI: 10.1016/j.eswa.2015.04.045.

[7] Guillaume Lemaître, Fernando Nogueira y Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". En: *Journal of Machine Learning Research* 18.17 (2017), págs. 1-5. URL: http://jmlr.org/papers/v18/16-365.html.

[8] Andrea Dal Pozzolo et al. *Credit Card Fraud Detection*. 2015. URL: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

[9] *ENISA Threat Landscape 2023*. European Union Agency for Cybersecurity (ENISA), 2023. URL: https://www.enisa.europa.eu/.

[10] Paul A. Grassi, Michael E. Garcia y James L. Fenton. *Digital Identity Guidelines: Authentication and Lifecycle Management*. NIST Special Publication 800-63B. National Institute of Standards y Technology (NIST), 2020. URL: https://pages.nist.gov/800-63-3/sp800-63b.html.

[11] Charles Elkan. "The Foundations of Cost-Sensitive Learning". En: *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*. 2001. URL: https://cseweb.ucsd.edu/~elkan/.

[12] Takaya Saito y Marc Rehmsmeier. "The Precision-Recall Plot is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". En: *PLoS ONE* 10.3 (2015), e0118432. DOI: 10.1371/journal.pone.0118432.

[13] Tom Fawcett. "An introduction to ROC analysis". En: *Pattern Recognition Letters* 27.8 (2006), págs. 861-874. DOI: 10.1016/j.patrec.2005.10.010.

[14] Chuan Guo et al. “On Calibration of Modern Neural Networks”. En: *Proceedings of the 34th International Conference on Machine Learning*. 2017, págs. 1321-1330. URL: https://proceedings.mlr.press/v70/guo17a.html.

[15] Haibo He y Edward A. Garcia. “Learning from Imbalanced Data”. En: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), págs. 1263-1284. DOI: 10.1109/TKDE.2008.239.

[16] Nitesh V. Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique”. En: *Journal of Artificial Intelligence Research* 16 (2002), págs. 321-357. DOI: 10.1613/jair.953.

[17] Trevor Hastie, Robert Tibshirani y Jerome Friedman. *The Elements of Statistical Learning*. 2ª ed. Springer, 2009. DOI: 10.1007/978-0-387-84858-7.

[18] Leo Breiman. “Random Forests”. En: *Machine Learning* 45 (2001), págs. 5-32. DOI: 10.1023/A:1010933404324.

[19] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. En: *Annals of Statistics* 29.5 (2001), págs. 1189-1232. DOI: 10.1214/aos/1013203451.

[20] Tianqi Chen y Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. En: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, págs. 785-794. DOI: 10.1145/2939672.2939785.

[21] *LightGBM Documentation*. Microsoft. 2024. URL: https://lightgbm.readthedocs.io/.

[22] Charles R. Harris et al. “Array programming with NumPy”. En: *Nature* 585.7825 (2020), págs. 357-362. DOI: 10.1038/s41586-020-2649-2.

[23] John D. Hunter. “Matplotlib: A 2D Graphics Environment”. En: *Computing in Science & Engineering* 9.3 (2007), págs. 90-95. DOI: 10.1109/MCSE.2007.55.

[24] *Streamlit Documentation*. Snowflake Inc. 2024. URL: https://docs.streamlit.io.

[25] European Parliament y Council. *Regulation (EU) 2015/847 on information accompanying transfers of funds and Directive (EU) 2018/843 (AMLD5)*. 2015. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32015R0847.

Parte II

Anexos

Apéndice A

Listados y código

A.1 Fragmentos de código relevantes

```
-•••
# •     70% para entrenamiento, 30% para test.
# •    Estratificado para mantener la proporción de fraudes raros.
# •    random_state=42 -> reproducibilidad.
print("Step 2: Dividiendo dataset en train/test...")
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

# Paso 3. Entrenamiento del Random Forest
    ----------------------------------------------------
# •    n_estimators = 100 árboles (compromiso precisión/tiempo).
# •    random_state=42 para reproducibilidad.
# •    n_jobs=-1 usaría todos los cores, pero aquí dejamos el default.
print("Step 3: Entrenando RandomForest Baseline...")
clf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
clf.fit(X_train, y_train)

# Paso 4. Evaluación en el conjunto de test
    ----------------------------------------------------
```

Listado A.1: RF base: partición estratificada y entrenamiento (`train_rf_baseline.py`)

```
.....
grid = HalvingGridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    factor=2,
    cv=3,
    scoring='recall',   # maximizamos recall en CV
    n_jobs=-1,
    verbose=1
)

#   Para acelerar, usamos sólo el 30 % del train dentro de la búsqueda
```

Listado A.2: RF optimizado: pipeline SMOTE+RF (`train_rf_optimized.py`)

```
……
grid.fit(X_tune, y_tune)
print("    > Mejores hiper-parámetros:")
print(grid.best_params_)

# Paso 5. Re-entrenar pipeline ganador en TODO el train
    ------------------------------------
print("Paso 5: Re-entrenando mejor modelo sobre el 70 % …completo")
best_model = grid.best_estimator_
best_model.fit(X_train, y_train)

# Paso 6. Evaluación final con el umbral por defecto (0.50) -------------------
print("Paso 6: Evaluación final usando el umbral por defecto (0.50)…")
y_pred = best_model.predict(X_test)
```

Listado A.3: RF optimizado: búsqueda con HalvingGridSearchCV (`train_rf_optimized.py`)

```
    X_train,X_test,y_train,y_test = train_test_split(
        X,y,test_size=0.30,stratify=y,random_state=42)
    os.makedirs('modelos',exist_ok=True); joblib.dump((X_train,X_test,y_train,
        y_test),split_path)

# Paso 3 · Modelo logístico (balanceado) -------------------------------------
logreg = LogisticRegression(max_iter=500, solver='lbfgs',
                            class_weight='balanced', n_jobs=-1,
                            C=1.0, penalty='l2', random_state=42)
print("Entrenando Logistic Regression …")
logreg.fit(X_train, y_train)
```

Listado A.4: Regresión logística: split y entrenamiento (`train_logreg.py`)

```
……>
    print("Paso 2: Dividiendo en train/test (70-30, estratificado)…")
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.30, stratify=y, random_state=42)
    os.makedirs('modelos', exist_ok=True)
    joblib.dump((X_train, X_test, y_train, y_test), split_path)

# Paso 3 · Definir y entrenar el modelo  -------------------------------------
print("Paso 3: Entrenando XGBoost …")
xgb = XGBClassifier(
        n_estimators=400,          # un número razonable
        max_depth=4,
```

Listado A.5: XGBoost: configuración y entrenamiento (`train_xgboost.py`)

```
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.30, stratify=y, random_state=42)
    os.makedirs('modelos', exist_ok=True)
    joblib.dump((X_train, X_test, y_train, y_test), split_path)

# Paso 3 · Definir modelo LightGBM ------------------------------------------
scale_pos = (len(y_train)-y_train.sum())/y_train.sum()
lgbm = LGBMClassifier(
        n_estimators=600, num_leaves=64, max_depth=-1,
        learning_rate=0.05, subsample=0.9, colsample_bytree=0.9,
```

```
        objective='binary', class_weight='balanced',
        scale_pos_weight=scale_pos, random_state=42)

print("Entrenando LightGBM …")
```

Listado A.6: LightGBM: split estratificado y modelo (`train_lightgbm.py`)

```
•• €• €•• € € €••••••••••
def buscar_umbral_coste_optimo(y_true, scores, amounts,
                               C_FN, C_FP_auto, C_FP_manual, T_low,
                               min_rec_high=0.0, min_rec_all=0.0, min_prec_all
                                   =0.0, high_amt_never_pass=0.0,
                               markup_micro=1.1, markup_med=1.2, markup_alto
                                   =1.5):

# ------------------------------------------------------------------------------
# Función central del análisis: busca el umbral óptimo de clasificación para
# cada modelo minimizando el coste total, teniendo en cuenta costes de FP y FN,
# restricciones normativas y recall mínimo en fraudes de alto importe.
# Esta función aplica la lógica de priorización y segmentación definida por la
    normativa.
# ------------------------------------------------------------------------------

    # -------------------- Validación cruzada cost-sensitive (scaffold)
        --------------------
    # TODO: Integrar búsqueda cost-sensitive por validación cruzada (KFold,
        n_splits=3) y promediar costes.
    # Por ahora, se mantiene la búsqueda original pero dejamos el scaffold para
         futura implementación.
    from sklearn.model_selection import KFold
    kf = KFold(n_splits=3, shuffle=True, random_state=42)
    best_costs = {}
    # Estructura para futura integración:
    # for train_idx, val_idx in kf.split(scores):
    #     # Para cada fold, llamar recursivamente a la misma función sin CV (
        helper interno _buscar_umbral_fold)
    #     pass  # placeholder

    # -------------------- Búsqueda de umbral cost-sensitive (lógica existente)
        --------------------
    y_true = np.asarray(y_true)
    scores = np.asarray(scores)
    amounts = np.asarray(amounts)
    pos = (y_true == 1)
    neg = ~pos

    p33, p66 = np.percentile(amounts[pos], [33, 66])
    uniq = np.unique(scores)
    thr_cand = uniq.tolist()

    all_costs, all_thr = [], []
    best_cost = np.inf
    best_thr, best_rec, best_f1 = 0.5, -1, -1
    best_counts = (0, 0, 0)
    best_fn_amount = 0

    low_mask = (amounts < T_low)
    total = len(thr_cand)
```

```
    step = max(1, total // 10)
    print_indices = set(list(range(0, total, step)) + [total - 1])
    print(f"   Buscando coste óptimo entre {len(thr_cand)} umbrales...", flush=
        True)

    PENALIZADOR_RECALL = 2.0  # Penalización proporcional si no se alcanza el
        recall mínimo

    for i, t in enumerate(thr_cand):
        y_pred = scores >= t
        fpm = neg & y_pred
        fnm = pos & ~y_pred

        rec_all_i = ((scores >= t) & pos).sum() / pos.sum() if pos.sum() > 0
            else 0
        tp_temp = (pos & (scores >= t)).sum()
        fp_temp = fpm.sum()
        prec_all_i = tp_temp / (tp_temp + fp_temp) if (tp_temp + fp_temp) > 0
            else 0

        # Se elimina el "continue" para no descartar ningún umbral
        # y en su lugar se aplica penalización más fuerte
        if min_rec_all > 0 and rec_all_i < min_rec_all:
            recall_deficit = min_rec_all - rec_all_i
        else:
            recall_deficit = 0

        # Clasificación FP automáticos y manuales
        fp_auto = fpm & low_mask
        fp_man = fpm & ~fp_auto

        # Regla CDD: revisión manual obligatoria  10.000 €
        cdd_mask = amounts >= 10000.0
        fp_man = fp_man | (fpm & cdd_mask)
        fp_auto = fp_auto & ~cdd_mask

        n_fn = fnm.sum()
        amount_fn = amounts[fnm].sum()

        fn_amounts = amounts[fnm]
        sum_micro = fn_amounts[fn_amounts <= p33].sum()
        sum_med = fn_amounts[(fn_amounts > p33) & (fn_amounts <= p66)].sum()
        sum_alto = fn_amounts[fn_amounts > p66].sum()

        cost_fn = C_FN * (
            markup_micro * sum_micro +
            markup_med * sum_med +
            markup_alto * sum_alto
        )

        # Penalización proporcional si no cumple el recall mínimo
        if recall_deficit > 0:
            cost_fn *= (1 + PENALIZADOR_RECALL * recall_deficit)
```

Listado A.7: Núcleo coste-sensible: inicialización y parámetros del barrido (`compare_models_cost.py`)

```
•• €• €•• € € €•••••••••• €
```

```
            cost_fn *= (1 + PENALIZADOR_RECALL * recall_deficit)

        cost_fp = fp_auto.sum() * C_FP_auto + fp_man.sum() * C_FP_manual
        total_cost = cost_fn + cost_fp

        all_costs.append(total_cost)
        all_thr.append(t)

        tp = (pos & y_pred).sum()
        prec = tp / (tp + fp_auto.sum() + fp_man.sum()) if tp else 0
        rec = tp / (tp + n_fn) if tp else 0
        f1 = 2 * prec * rec / (prec + rec) if (prec + rec) else 0

        if (total_cost < best_cost or
            (total_cost == best_cost and rec > best_rec) or
            (total_cost == best_cost and rec == best_rec and f1 > best_f1)):
            best_cost, best_thr, best_rec, best_f1, best_counts = total_cost, t
                , rec, f1, (
                fp_auto.sum(), fp_man.sum(), n_fn)
            best_fn_amount = amount_fn

        if i in print_indices:
            print(f"   · {i+1}/{total} umbrales | best_cost={int(best_cost)€} @
                 thr={best_thr:.4f}", flush=True)

    # Fallback en caso de no encontrar nada válido
    if not np.isfinite(best_cost) or best_cost == np.inf:
        idx_best = int(np.argmin(all_costs))
        best_cost = all_costs[idx_best]
        best_thr = all_thr[idx_best]

    safe_cost = int(np.round(best_cost)) if np.isfinite(best_cost) else 0
    return best_thr, safe_cost, best_counts, best_fn_amount

def segment_fn_amounts(amounts, fn_mask, p33, p66, C_FN_micro, C_FN_medio,
    C_FN_alto):
    """Segmenta FN en 3 grupos por importe y calcula costes."""
    fn_amts = amounts[fn_mask]
    n1 = np.sum(fn_amts<=p33)
    n2 = np.sum((fn_amts>p33)&(fn_amts<=p66))
    n3 = np.sum(fn_amts>p66)
    return (int(n1),int(n2),int(n3),
            int(n1*C_FN_micro), int(n2*C_FN_medio), int(n3*C_FN_alto))


# --- Funciones de plot con caja de leyenda baja central ---------------

def _draw_legend(text):
    # Dividir el texto por ';', eliminar espacios y limitar a 3 líneas
    lines = text.split(';')
    lines = [ln.strip() for ln in lines]
    if len(lines) > 3:
        lines = [lines[0], '; '.join(lines[1:-1]), lines[-1]]
    text_block = "\n".join(lines)
    plt.gcf().text(
        0.5, 0.05, text_block,
        ha='center', va='bottom',
        fontsize=9,
```

```
        bbox=dict(boxstyle="round,pad=0.5", facecolor="white", alpha=0.8)
    )

def plot_confmatrix_grid(confs, title, path, legend_text=None):
    n = len(confs)
    fig, axes = plt.subplots(1, n, figsize=(5 * n, 4))
    if n == 1:
        axes = [axes]
```

Listado A.8: Cómputo de matriz de confusión efectiva y costes (`compare_models_cost.py`)

```
•• €• €•• € € €•••••••••• €€
        axes = [axes]
    # Dibujar los valores de cada celda en la matriz de confusión
    for ax, (cm, thr, name, cost) in zip(axes, confs):
        im = ax.imshow(cm, cmap="Blues")
        for i in (0, 1):
            for j in (0, 1):
                ax.text(j, i, cm[i, j], ha='center', va='center', color='black'
                    , fontsize=12)
        ax.set_title(f"{name}\nthr={thr:.3f}\ncoste={cost€}", fontsize=9)
        ax.set_xlabel("Predicted")
        ax.set_ylabel("Real")
        fig.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
    fig.suptitle(title, fontsize=14)
    plt.tight_layout(rect=[0, 0.30, 1, 0.95])
    plt.subplots_adjust(bottom=0.35)
    if legend_text:
        _draw_legend(legend_text)
    fig.savefig(path)
    plt.close(fig)

def plot_barras_coste(df, title, path, legend_text=None):
    plt.figure(figsize=(10,6))
    # Resaltar modelos que no cumplen las restricciones de umbral
    if 'ok_thr' in df.columns:
        colors = ['C0' if ok else 'red' for ok in df['ok_thr']]
    else:
        colors = ['C0'] * len(df)
    bars = plt.bar(df['modelo'], df['coste'], color=colors)
    ymax=df['coste'].max()
    for b,v in zip(bars, df['coste']):
        plt.text(b.get_x()+b.get_width()/2, v+0.02*ymax,
                 f"{v€}", ha='center', va='bottom', fontsize=9)
    plt.title(title); plt.ylabel("Coste total €()")
    plt.xticks(rotation=15)
    plt.tight_layout(rect=[0,0.18,1,0.95])
    plt.subplots_adjust(bottom=0.25)
    if legend_text: _draw_legend(legend_text)
    plt.savefig(path); plt.close()

def plot_barras_metricas(df, title, path, legend_text=None):
    plt.figure(figsize=(10,6))
    idx = np.arange(len(df)); w = 0.25
    # Usar colores fijos para cada métrica: verde para Precisión, naranja para
        Recall, azul para F1
    # No es necesario calcular colors_ok para este gráfico
    bars_prec = plt.bar(idx-w, df['precision'], w, label='Prec', color='green')
```

```
    bars_rec  = plt.bar(idx,   df['recall'],    w, label='Rec', color='orange')
    bars_f1   = plt.bar(idx+w, df['f1'],        w, label='F1', color='blue')
    plt.xticks(idx, df['modelo'], rotation=15)
    plt.ylim(0,1.05)
    # Anotar cada barra con su valor numérico
    for bar in bars_prec:
        h = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, h, f"{h:.2f}", ha='center',
            va='bottom', fontsize=8)
    for bar in bars_rec:
        h = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, h, f"{h:.2f}", ha='center',
            va='bottom', fontsize=8)
    for bar in bars_f1:
        h = bar.get_height()
```

Listado A.9: Métricas y agregados del barrido (`compare_models_cost.py`)

```
•• €• €•• € € €•••••••••• €€€€€
        h = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, h, f"{h:.2f}", ha='center',
            va='bottom', fontsize=8)
    plt.title(title)
    plt.legend()
    plt.tight_layout(rect=[0,0.18,1,0.95])
    plt.subplots_adjust(bottom=0.25)
    if legend_text:
        _draw_legend(legend_text)
    plt.savefig(path)
    plt.close()

def plot_pr_curves(curves, apd, title, path, legend_text=None, thr_points=None)
    :
    plt.figure(figsize=(8,6))
    for nm,(p,r) in curves.items():
        plt.plot(r,p,label=f"{nm} (AP={apd[nm]:.3f})")
    if thr_points:
        for nm, (rec_thr, prec_thr) in thr_points.items():
            plt.scatter(rec_thr, prec_thr, marker='o', s=50, edgecolor='black')
    plt.xlabel("Recall"); plt.ylabel("Precision")
    plt.title(title); plt.legend(loc='lower left'); plt.grid(alpha=0.3)
    plt.tight_layout(rect=[0,0.22,1,0.95])
    plt.subplots_adjust(bottom=0.30)
    if legend_text: _draw_legend(legend_text)
    plt.savefig(path); plt.close()

def plot_roc_curves(curves, aud, title, path, legend_text=None, thr_points=None
    ):
    plt.figure(figsize=(8,6))
    for nm,(fpr,tpr) in curves.items():
        plt.plot(fpr,tpr,label=f"{nm} (AUC={aud[nm]:.3f})")
    plt.plot([0,1],[0,1],'k--',alpha=0.4)
    if thr_points:
        for nm, (fpr_thr, tpr_thr) in thr_points.items():
            plt.scatter(fpr_thr, tpr_thr, marker='o', s=50, edgecolor='black')
    plt.xlabel("FPR"); plt.ylabel("TPR")
    plt.title(title); plt.legend(loc='lower right'); plt.grid(alpha=0.3)
    plt.tight_layout(rect=[0,0.22,1,0.95])
```

```
    plt.subplots_adjust(bottom=0.30)
    if legend_text: _draw_legend(legend_text)
    plt.savefig(path); plt.close()

# ------------------------------ MAIN ----------------------------------

def main():
    p = argparse.ArgumentParser()
    p.add_argument("--mode", dest="mode", type=str, default="normativa",
                   help="Modo de ejecución: normativa o personalizado")
    p.add_argument("--C_FN", type=float, default=3.64)
    p.add_argument("--C_FP_auto", type=float, default=0.04)
    p.add_argument("--C_FP_manual", type=float, default=4.0)
    p.add_argument("--T_low", type=float, default=100.0)
```

Listado A.10: Selección del umbral óptimo por coste (`compare_models_cost.py`)

```
•• €• €•• € € €•••••••••• €€€€€
    if mode == 'normativa':
        print("  Ejecutando en modo normativa")
        revisiones_por_hora = 40
        horas_test = 16
        analistas_disponibles = 20
        capacidad_manual = int(revisiones_por_hora * horas_test *
            analistas_disponibles)

    # --- Personalizado ---
    elif mode == 'personalizado':
        print("  Ejecutando en modo personalizado")
        revisiones_por_hora = args.revisiones_por_hora
        horas_test = args.horas_test
        analistas_disponibles = args.analistas_disponibles
        capacidad_manual = int(revisiones_por_hora * horas_test *
            analistas_disponibles)

    # Asegurar datos cargados
    _load_amount_stats("creditcard.csv")
    os.makedirs("comparar_coste", exist_ok=True)

    # Resto del código principal del script (idéntico)
    # ------------------------------------------------
    data = joblib.load("modelos/test_split.pkl")
    if isinstance(data, tuple):
        if len(data) == 2:
            X_train, y_train = None, None
            X_test, y_test = data
        elif len(data) == 4:
            X_train, X_test, y_train, y_test = data
        else:
            raise RuntimeError("Formato inesperado de test_split.pkl")
    else:
        raise RuntimeError("Formato inesperado de test_split.pkl")
    if X_train is None or y_train is None:
        raise RuntimeError("No hay datos de entrenamiento para calibración")

    if 'Amount' not in X_test.columns:
        raise RuntimeError("X_test debe tener columna 'Amount'")
    amounts = z_to_eur(X_test['Amount'].values)
```

```
40
41      p33, p66 = np.percentile(amounts[y_test == 1], [33, 66])
42      if 'Amount' not in X_train.columns:
43          raise RuntimeError("X_train debe tener columna 'Amount'")
44      amounts_train = z_to_eur(X_train['Amount'].values)
```

Listado A.11: Carga de *scores* y parámetros de coste/capacidad (`compare_models_cost.py`)

A.2 Manual de ejecución o despliegue

Estructura del *workspace* y archivos necesarios

Para ejecutar correctamente el simulador, el entorno de trabajo debe mantener la estructura mostrada en la Figura A.1, que organiza los módulos de ejecución, los modelos entrenados y los datos de entrada necesarios.

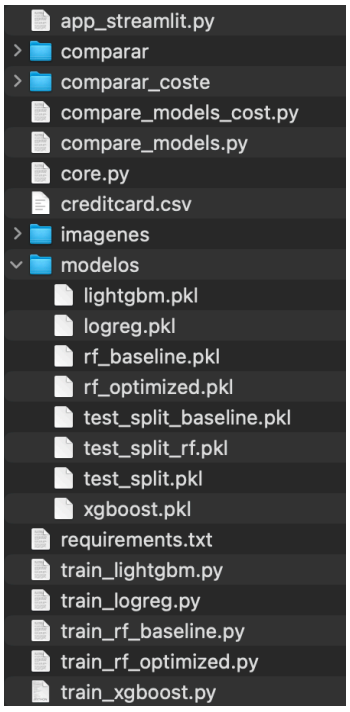


Figura A.1: Estructura de directorios y archivos del *workspace* del proyecto.

El contenido del directorio raíz `Workspace TFG` se distribuye de la siguiente manera:

- **Archivos principales de ejecución**
  - `app_streamlit.py`: interfaz principal de usuario, implementada con *Streamlit*.
  - `core.py`: núcleo funcional que gestiona cálculos de costes, métricas y umbrales.
  - `compare_models_cost.py`: módulo de comparación coste-sensible entre modelos.

  - `requirements.txt`: lista de dependencias requeridas para el entorno virtual.

- **Modelos preentrenados** (carpeta `modelos/`)
  - `rf_baseline.pkl` y `rf_optimized.pkl`: versiones base y optimizada del modelo *Random Forest*.
  - `xgboost.pkl`, `lightgbm.pkl` y `logreg.pkl`: modelos complementarios calibrados (*XGBoost*, *LightGBM* y regresión logística).
- **Datos de entrada y particiones**
  - `creditcard.csv`: dataset original del problema de detección de fraude.
  - `test_split.pkl`, `test_split_rf.pkl`, `test_split_baseline.pkl`: divisiones estratificadas *train/valid/test* para reproducibilidad experimental.
- **Scripts de entrenamiento** (prefijo `train_`)
  - `train_rf_baseline.py`, `train_rf_optimized.py`, `train_xgboost.py`, `train_lightgbm.py`, `train_logreg.py`: scripts individuales para reentrenar cada modelo y generar sus artefactos `.pkl`.
- **Carpetas auxiliares**
  - `imagenes/`: contiene recursos visuales utilizados en la memoria y la interfaz.
  - `comparar/` y `comparar_coste/`: módulos de evaluación y validación de rendimiento.
  - `__pycache__/`: archivos temporales generados automáticamente por Python.

Los modelos en formato `.pkl` son cargados automáticamente al iniciar la aplicación. En caso de necesitar un reentrenamiento completo, basta con ejecutar los scripts correspondientes; los nuevos artefactos se guardarán en la carpeta `/modelos` con el mismo nombre, sobrescribiendo los anteriores.

- Se recomienda **no modificar manualmente** los archivos `.pkl` ni los ficheros de particiones, para evitar inconsistencias entre los índices de entrenamiento y validación.
- Los resultados de las simulaciones se almacenan automáticamente en una carpeta `/results/`, creada en la primera ejecución.

Ejecución del simulador

El simulador puede lanzarse desde la terminal o directamente desde un entorno Python, utilizando el comando estándar de *Streamlit*. En un terminal situado en el directorio raíz del proyecto, se ejecuta:

```
streamlit run app_streamlit.py
```

Tras unos segundos, la consola mostrará un mensaje similar al siguiente:

```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8502
Network URL: http://192.168.X.X:8502
```

Accediendo a la **URL local**, se abre automáticamente la interfaz web del simulador, desde la cual el usuario puede seleccionar el modo de simulación (*Normativa* o *Personalizada*) y configurar los parámetros de costes, capacidad y reglas operativas.

En caso de que el puerto indicado ya esté en uso, puede lanzarse en otro mediante:

```
streamlit run app_streamlit.py --server.port 8503
```

Recomendaciones técnicas

- Se aconseja ejecutar la aplicación en un entorno virtual con las dependencias definidas en `requirements.txt`.
- En sistemas macOS y Linux, puede mejorar el rendimiento instalando el módulo `watchdog`:

  ```
  pip install watchdog
  ```

- Si se desea ejecutar desde un entorno Python (por ejemplo, VS Code o Jupyter), se puede invocar directamente:

  ```
  import os
  os.system("streamlit run app_streamlit.py")
  ```

Con esta configuración, el simulador puede ejecutarse en cualquier entorno compatible con Python 3.9 o superior, manteniendo la coherencia entre datos, modelos y resultados sin requerir pasos de configuración adicionales.

Apéndice B

Contribución a los Objetivos de Desarrollo Sostenible (ODS)

La Agenda 2030 de las Naciones Unidas establece un marco global compuesto por 17 Objetivos de Desarrollo Sostenible (ODS), orientados a promover un progreso social, económico y medioambiental equilibrado. Aunque este Trabajo Fin de Grado se centra en el ámbito técnico de la detección de fraude mediante inteligencia artificial, sus aportaciones se alinean de forma directa e indirecta con varios de estos objetivos, especialmente en lo relativo a la innovación tecnológica, la seguridad financiera y la construcción de infraestructuras digitales fiables.

A continuación, se detalla la contribución específica del proyecto a los ODS más relevantes.

ODS 9: Industria, Innovación e Infraestructura

El ODS 9 promueve el desarrollo de infraestructuras resilientes, la adopción de tecnologías innovadoras y el fomento de la digitalización segura en sectores clave como el financiero.

Este TFG contribuye a dicho objetivo mediante:

- El diseño e implementación de un sistema avanzado de detección de fraude basado en *machine learning*, incluyendo calibración probabilística y optimización coste-sensible.
- El desarrollo de una aplicación interactiva en *Streamlit* que ofrece un entorno reproducible, transparente y ampliable para el análisis de estrategias antifraude.
- La integración de técnicas modernas como *SMOTE*, *threshold optimization* y evaluación *example-dependent*, fortaleciendo las capacidades digitales en el sector *FinTech*.

Estas contribuciones apoyan la construcción de infraestructuras financieras más seguras e inteligentes, alineadas con prácticas de innovación tecnológica responsable.

ODS 16: Paz, Justicia e Instituciones Sólidas

El ODS 16 aboga por la reducción del fraude, la corrupción y los delitos financieros, así como por el fortalecimiento de instituciones transparentes y seguras.

Este trabajo refleja dicho objetivo a través de:

- La reducción del fraude financiero mediante modelos predictivos que detectan anomalías con alta sensibilidad, minimizando el impacto económico y reputacional.
- El análisis coste-sensible que permite priorizar transacciones de alto riesgo, optimizando los recursos de revisión manual y mejorando la eficacia operativa.
- La alineación con marcos regulatorios como PSD2, SCA y las buenas prácticas de TRA, reforzando la trazabilidad y auditoría de decisiones.

La metodología desarrollada contribuye a la creación de sistemas de pago más fiables, fomentando la confianza de usuarios, comercios y entidades financieras.

ODS 8: Trabajo Decente y Crecimiento Económico

El ODS 8 impulsa la eficiencia económica, la innovación y la mejora de procesos productivos.

Este TFG contribuye a este objetivo mediante:

- La reducción de pérdidas económicas asociadas al fraude financiero, aumentando la estabilidad del ecosistema *FinTech*.
- La optimización de los recursos humanos mediante un sistema híbrido de decisión (*auto-decline*, *manual review*), que reduce la carga operativa de analistas y mejora la eficiencia.
- La creación de herramientas que permiten evaluar estrategias de detección con criterios económicos, facilitando la toma de decisiones basada en datos.

En conjunto, estas aportaciones favorecen un crecimiento económico sostenible impulsado por la analítica avanzada.

Síntesis

El presente Trabajo Fin de Grado contribuye de forma clara a los ODS relacionados con innovación, eficiencia operativa y seguridad institucional. La combinación de modelos de aprendizaje automático, técnicas de calibración y análisis coste-sensible facilita la construcción de sistemas antifraude más robustos, eficientes y alineados con los principios de la Agenda 2030.