

Machine Learning aplicado a la detección de fraude con tarjeta de crédito

UOC

David de Vega Martin

Área 1

Tutor/a de TF

Francesc Julve López

Fecha Entrega
Junio de 2023

Universitat Oberta
de Catalunya



Esta obra está sujeta a una licencia de Reconocimiento [3.0 España de Creative Commons](#)

Ficha del Trabajo Final

Título del trabajo:	Machine Learning aplicado a la detección de fraude con tarjeta de crédito
Nombre del autor/a:	David de Vega Martin
Nombre del Tutor/a de TF:	Francesc Julve López
Nombre del/de la PRA:	PEC1 definición y Planificación del Trabajo Final.
Fecha de entrega:	04/2023
Titulación o programa:	Master Ciencia de Datos
Área del Trabajo Final:	Área 1
Idioma del trabajo:	Castellano
Palabras clave	Fraude, tarjeta, crédito
Resumen del Trabajo	
<p>El pago con tarjeta de crédito está en auge como consecuencia de la expansión del comercio electrónico, cambios en los hábitos de consumo y por la reducción paulatina del uso de efectivo impulsada por los distintos gobiernos de los países.</p> <p>Este uso masivo de la tarjeta de crédito la convierte en un objetivo fácil para el fraude ya que permite obtener grandes cantidades de dinero, en un periodo corto de tiempo y con un bajo riesgo de detección.</p> <p>El alto volumen de operaciones hace que los métodos convencionales de detección resulten obsoletos.</p> <p>Es en este punto donde surge los métodos de aprendizaje automático como método más efectivo para combatir el fraude. Sin embargo hay que mencionar que estas técnicas se enfrentan, principalmente, a dos retos:</p> <ul style="list-style-type: none">- La naturaleza evolutiva de los medios y métodos para realizar el fraude.- Un fuerte sesgo en los conjuntos de datos de fraude con tarjeta de crédito, que se encuentran muy desequilibrados. Esto es, presentan un fuerte	

desequilibrio entre clases, operaciones legítimas vs fraudulentas

El objetivo de este TFM consiste en analizar, seleccionar e implementar una serie de algoritmos de aprendizaje automático, realizar un estudio comparativo de los mismos y por último obtener un modelo de predicción, que determine si una transacción es legítima o no.

Abstract

Credit card payment is booming as a result of the expansion of e-commerce, changes in consumer habits and the gradual reduction in the use of cash driven by the different governments of the countries.

This massive use of the credit card makes it an easy target for fraud since it allows to obtain large amounts of money, in a short period of time and with a low risk of detection.

High trading volume makes conventional detection methods obsolete.

It is at this point that machine learning methods emerge as the most effective method to combat fraud. However, it should be mentioned that these techniques mainly face two challenges:

- The evolving nature of the means and methods of carrying out fraud.
- A strong bias in credit card fraud datasets, which are very unbalanced. That is, they present a strong imbalance between classes, legitimate vs. fraudulent operations.

The objective of this TFM is to analyze, select and implement a series of machine learning algorithms, perform a comparative study of them and finally obtain a prediction model, which determines whether a transaction is legitimate or not.

Index

1.	Introducción	5
1.1.	Contexto y justificación del Trabajo	5
1.3	Objetivos del Trabajo	9
1.4	Impacto en sostenibilidad, ético-social y de diversidad	10
1.5	Enfoque y método seguido	10
1.6	Planificación del trabajo	12
1.7	Breve sumario de productos obtenidos	13
1.8	Breve descripción de otros capítulos de la memoria	13
2.	Estado del Arte	14
2.1.	Introducción	14
2.2.	Pre-Procesado.	14
2.3.	Algoritmos.	16
2.4	Métricas de rendimiento.	18
2.5	Caso de uso.	21
3	Metodología	23
4.1.	Análisis Exploratorio de Datos (EDA)	26
4.1.1.	Variable Time	28
4.1.2.	Variables Características V1-V28	30
4.1.3.	Variable Amount	31
4.2.	Ajustes del dataset	32
4.2.1.	Escalado de variables	32
4.2.2.	Separación del dataset	32
4.2.3.	Tratamiento del sesgo de las variables y desequilibrios de clases	33
5.	Algoritmos	40
5.1.	Algoritmos Lineales: Regresión Logística	40
5.2.	Algoritmos de Boosting	43
5.2.1.	AdaBoost	43
5.2.2.	XGBoost	46
5.2.3.	Catboost	48
5.3.	Algoritmos basados en arboles de decisión	50
5.3.1.	Decisión Tree	50

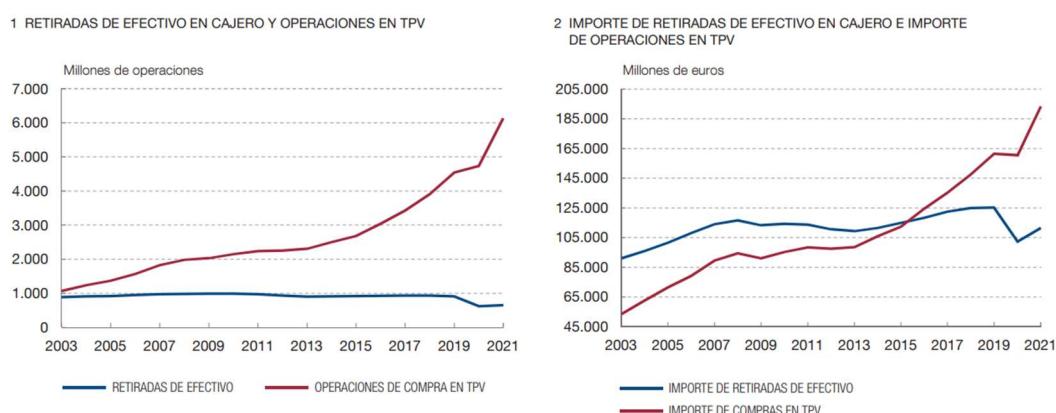
5.3.2. Random Forest	52
5.4. Algoritmos de redes neuronales: MLP	53
5.5. Algoritmos basados en instancia: KNN	55
6. Conclusiones y trabajos futuros.	60
Glosario	63
Bibliografía	64

1. Introducción

1.1. Contexto y justificación del Trabajo

El pago con tarjeta de crédito está en auge como consecuencia de múltiples factores. Principalmente por la expansión del comercio electrónico, por la popularización de dispositivos como los smartphones o los wearables que facilitan dicha tarea. Por los cambios en los hábitos de consumo que la pandemia del COVID 19 ha agilizado. Y por último por la reducción paulatina del uso de efectivo impulsada por los distintos gobiernos de los países [1] [2] [3] [4] [5].

En el gráfico 1, perteneciente a los datos del Sistema de Tarjetas y Medios de Pago elaborado por el Banco de España [6], podemos ver claramente como la cantidad de operaciones de retirada de efectivo se encuentra estancando y disminuyendo paulatinamente. En paralelo a ese fenómeno se disparan el número de operaciones de compra en TPV (Terminal de Punto de Venta), por exclusión todas las que no son retiradas de efectivo en cajero. De hecho en el gráfico de la derecha podemos observar como el monto de operaciones en TPV supera a las retiradas en efectivo desde 2016 siendo esto una tendencia consolidada y que continua al alza.



FUENTE: Banco de España, a partir de los datos de Sistema de Tarjetas y Medios de Pago.

NOTA: Los niveles para 2021 se han estimado en función del comportamiento de la variable para los tres primeros trimestres del año.

Gráfico 1 Fuente Banco de España.

Este uso masivo de la tarjeta de crédito la convierte en un objetivo fácil para el fraude ya que permite obtener grandes cantidades de dinero, en un periodo corto de tiempo y con un bajo riesgo de detección.

El fraude en transacciones con tarjeta de crédito puede definirse como el uso de la tarjeta de una persona por parte de un tercero, para sus fines personales, sin el conocimiento y/o

consentimiento del titular de esta y sin que las autoridades emisoras de la tarjeta tengan constancia de que esto está sucediendo.

El séptimo informe sobre fraude con tarjeta emitido por el Banco Central Europeo (ECB)^[7], analiza las tendencias en el fraude con tarjetas emitidas dentro de la Zona Única de Pagos en euros (Single Euro Payment Área SEPA). El contexto se sitúa en el 2015 a 2019, centrándose en este último. Recoge dos categorías principales de fraude:

2.1 Fraude con presencia física de tarjeta (Card Present)

Esta categoría podríamos contemplar dos comportamientos:

1.1 Transacciones fraudulentas en un terminal de punto de venta (TPV) Physical Point Of Sale, POS.

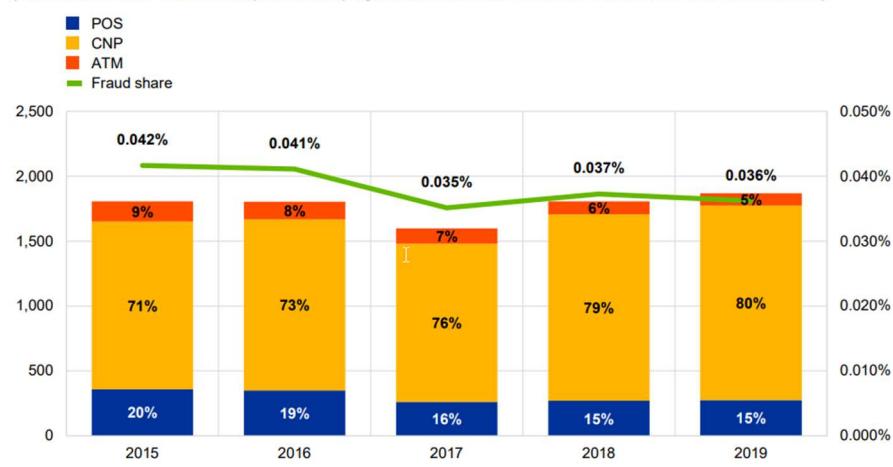
1.2 Extracciones de efectivo fraudulentas en cajeros automáticos, Automated Teller Machines, ATM.

2.2 Fraude sin presencia de tarjeta o remoto (Card No Present, CNP) las efectuadas mediante apropiación de los datos de esta.

El valor total de las transacciones empleando tarjetas emitidas dentro del SEPA ascienden a 5.16 trillones de euros de los cuales eran fraudulentas un total de 1.87 billones de euros a nivel mundial, de las cuales 1.03 billones corresponden exclusivamente al área euro, un 0.036% del total.

El gráfico 2 se muestran las proporciones que suponen cada modalidad. La opción mayoritaria son las operaciones con tarjeta física no presente (CNP) que para nuestro periodo de análisis pasan de un 71% a un 80%. Seguida por las operaciones en TPVs (POS), que pasan de un 20% a un 15%. Por último tenemos las extracciones fraudulentas en cajeros automáticos que bajan de un 9% a un 5%.

(left-hand scale: total value of fraud (EUR millions); right-hand scale: value of fraud as a share of the value of transactions)

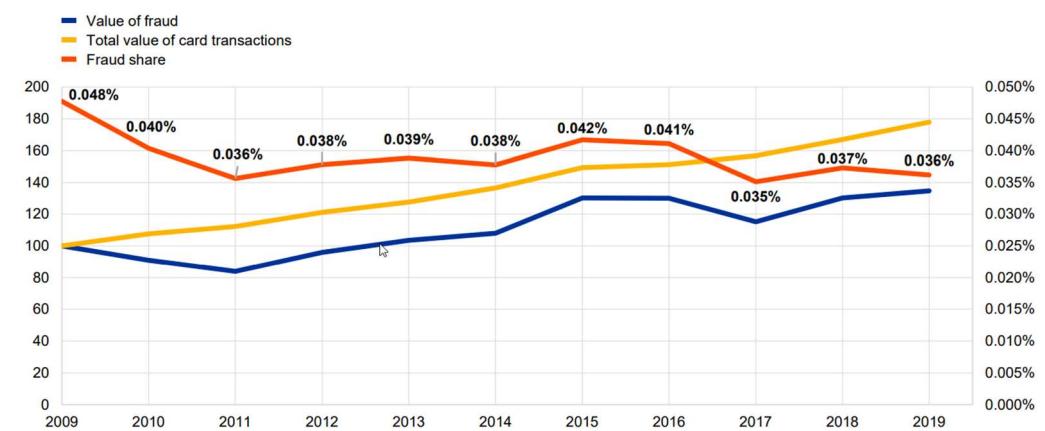


Source: All reporting card payment scheme operators.

Gráfico 2 Fuente Banco Central Europeo (ECB).

Como puede observarse en el grafico 3, en la última década el volumen de operaciones fraudulentas ha aumentado en términos absolutos, en un contexto de crecimiento del número de operaciones y un incremento de relevancia de los pagos por CNP, siendo estos los más afectados. En términos relativos, esto es, como parte del valor de las operaciones globales, el porcentaje, podemos ver cómo ha remitido paulatinamente, especialmente en los últimos años. Sugiriendo esto, el impacto beneficioso de las medidas adoptadas por la industria así como las regulatorias.

(left-hand scale: 2009 value = 100; right-hand scale: value of fraud as a share of the value of transactions)



Source: All reporting card payment scheme operators for each year.

Gráfico 3 Fuente Banco Central Europeo (ECB).

Por último en el grafico 4 se puede ver la evolución de las distintas tipologías de fraude en la última década, conforme a las siguientes modalidades:

1- Lost and stolen fraud.

Fraude por robo o perdida física de la tarjeta.

2- Card not received fraud.

La solicitud de tarjeta fraudulenta, en este caso el delincuente empleando información privada y protegida, rellena la información que le permite solicitar una tarjeta contra la que efectuar cargos.

3- Counterfeit fraud.

Las principales prácticas son:

3.1- Tarjeta falsificada.

Consiste en la creación de un duplicado físico de una tarjeta existente. Para ello emplean herramientas de hacking con las que obtiene los datos de esta. En el duplicado físico se llega a reproducir incluso la banda magnética de la tarjeta, se emplea tanto para compras físicas como online.

3.2- El fraude de intrusión de dispositivo.

El delincuente accede ilegalmente al dispositivo, como un terminal de punto de venta o un cajero automático, para obtener la información de la tarjeta. El delincuente puede emplear herramientas de hacking para instalar software malicioso o dispositivos de skimming para robar la información de la tarjeta.

4- Other Fraud

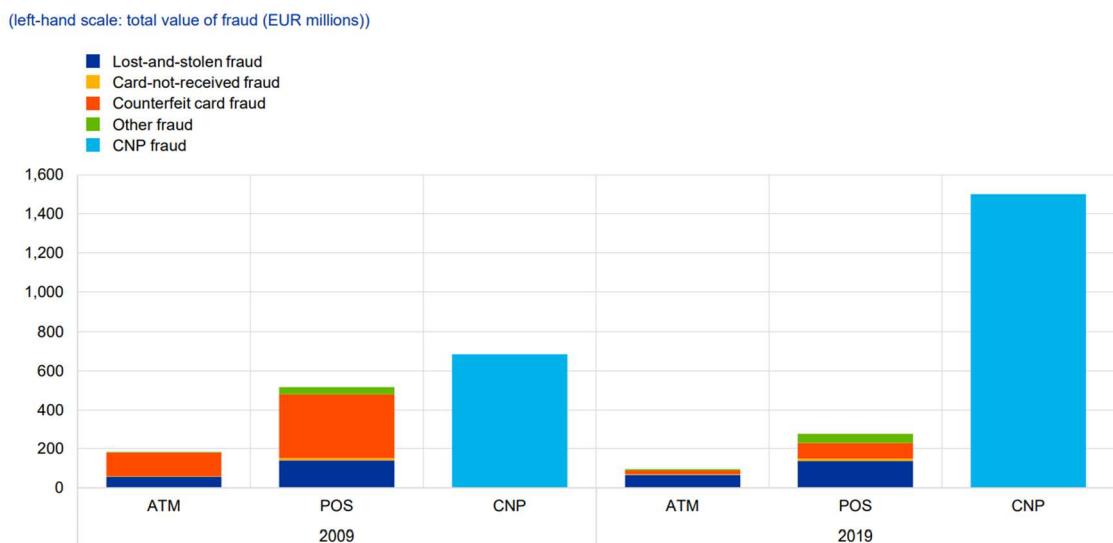
Recoge el resto de las variantes fraudulentas.

La más representativa sería el fraude de telecomunicación. En este caso el delincuente utiliza la tarjeta para cargar contra ella suscripciones a servicios online, compra de contenido digital e incluso consumos domésticos, si bien se concentra en facturas de telecomunicaciones. Esta modalidad es difícil de detectar si el titular no revisa detalladamente sus facturas.

Fácilmente se puede apreciar el fuerte incremento del fraude de tarjeta no presente (CNP) en toda la década.

El fraude de tipo lost and stolen, pérdida o robo de la tarjeta, así como el card not received, tarjeta no solicitada, se mantienen estables a través del tiempo.

El único que realmente se reduce es el counterfeit, el asociado a duplicados de tarjeta físico y el motivado por fraude de intrusión de dispositivo, el cual se reduce por las medidas de seguridad implementadas por la industria, así como las mejoras legislativas en materia de ciberseguridad, como mencionamos anteriormente.



Source: All reporting card payment scheme operators in the respective years.

Gráfico 4 Fuente Banco Central Europeo (ECB).

La reducción del volumen de operaciones fraudulentas que se consigue por los métodos anteriores tiene una naturaleza eminentemente preventiva, esto es, evitar que la operación se produzca, pero eso no siempre es posible y hay que buscar medidas para

detectarlas una vez se han materializado. Por todo lo argumentado hasta ahora, se puede afirmar que la detección de operaciones fraudulentas por medios tradicionales de detección manual es completamente ineficaz tanto por el tiempo como por los recursos requeridos, así como por el constante incremento del volumen de estas.

Es en este punto donde surge los métodos de aprendizaje automático como método más efectivo para combatir el fraude. Sin embargo hay que mencionar que estas técnicas se enfrentan, principalmente, a dos retos:

- La naturaleza evolutiva de los medios y métodos para realizar el fraude.
- Un fuerte sesgo en los conjuntos de datos de fraude con tarjeta de crédito, que se encuentran muy desequilibrados. Esto es, presentan un fuerte desequilibrio entre clases, operaciones legítimas vs fraudulentas.

1.3 Objetivos del Trabajo

Elegir e implementar una serie de algoritmos de aprendizaje automático, realizar un estudio comparativo de los mismos y por último obtener un modelo de predicción, que determine si una transacción es legítima o no.

Este objetivo se articulará mediante una secuencia de pasos que procedemos a describir a continuación:

- Se procederá a buscar en la bibliografía; artículos, estudios y casos de uso; sobre lucha contra el fraude en las transacciones bancarias.
- Análisis de la distribución de los datos del dataset de operaciones bancarias.
- Seleccionar que técnica de submuestreo y sobre muestreo pueden ser aplicables a nuestro dataset, habida cuenta de que se haya fuertemente desequilibrado.
- Desarrollar las técnicas de submuestreo aleatorio que nos permita:
 - Crear un subconjunto de datos donde tengamos una ratio de operaciones legítimas y fraudulentas equilibrado.
 - Detectar las anomalías que se nos puedan presentar así como tratar los valores atípicos en el caso de que estuviesen presentes.
 - Aplicar algoritmos de reducción de dimensionalidad y clustering, proceder a una comparativa y seleccionar el que mejor rendimiento ofrezca a nuestra submuestra.

Con estas actuaciones tendríamos ya nuestro data set preparado para comenzar a desplegar algoritmos.

- Seleccionar los algoritmos clasificadores, proceder a su implementación y posterior análisis de rendimiento, que ofrezcan una mayor precisión.

1.4 Impacto en sostenibilidad, ético-social y de diversidad

El presente trabajo se enmarca en los ODS (Objetivos de Desarrollo Sostenible 2030, ONU) en el área de Comportamiento ético y responsabilidad social (RS), concretamente en el epígrafe ODS-8 Decent work and economic growth.

El impacto económico de un método preciso de detección de fraude es alto tanto por los altos volúmenes de capital defraudado, como por el lucro cesante en concepto de operaciones no realizadas por falta de seguridad o desconfianza en algún eslabón de la transacción.

Por último hay que mencionar que las variables del dataset se encuentran anonimizadas a excepción de fecha e importe, por lo cual no existe información sensible que genere conflicto alguno ético y/o responsabilidad social.

1.5 Enfoque y método seguido

Para la realización de este TFM se va a emplear la metodología CRISP-DM Cross Industry Standard Process for Data Mining. Esta metodología integra todas las tareas que precisa un proyecto de minería de datos. Esta consta de 6 fases:

- 1.- Comprensión del negocio (Business Understanding).

En nuestro caso, el objetivo es la consecución de un modelo de predicción de fraude.

- 2.- Comprensión de los datos (Data Understanding).

Consta de dos etapas:

- Proporcionar una descripción detallada del dataset, en la que se analizara su estructura y distribución.
- Gestionar la calidad de los datos, identificando problemas y planteando soluciones. Las principales dificultades por abordar son el desequilibrio de clases, así como el tratamiento de los valores atípicos.

3.- Preparación de los datos (Data Preparation).

Esta fase se centra en la obtención del dataset más adecuado sobre el que aplicar los modelos.

4.- Modelado de los datos (Modeling).

El fin último es la obtención de un modelo de predicción de fraude que nos permita alcanzar los objetivos del proyecto a través del siguiente proceso:

- Seleccionar las técnicas de modelado más adecuados para nuestro dataset y nuestros objetivos.
- Establecer una estrategia que permita evaluar la calidad del modelo.
- Construir el modelo a partir de las técnicas previamente seleccionadas.

5.- Evaluación (Evaluation).

Esta fase comprende los siguientes pasos:

- Evaluar el modelo o modelos generados hasta el momento.
- Revisar todo el proceso de minería para comprobar que se ha ejecutado respecto a lo planificado.

6.- Despliegue (Deployment).

Es la fase de puesta en producción, en ella, el modelo obtenido se procederá a desplegar en una API.

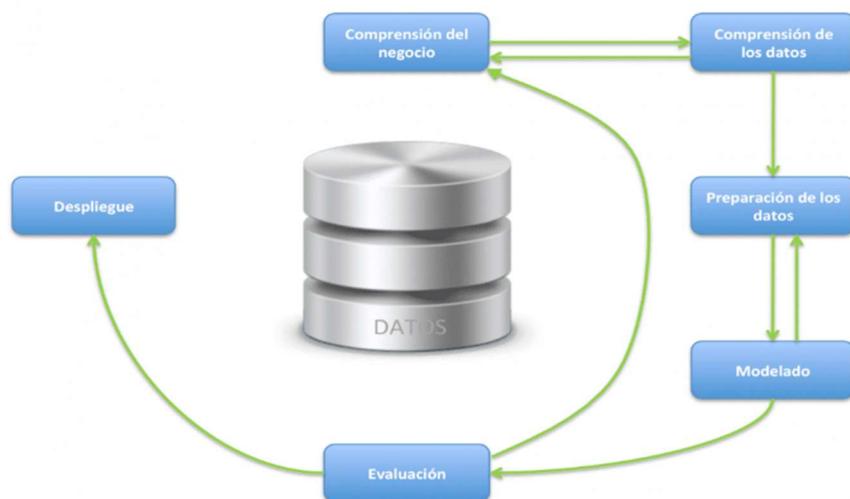


Gráfico 5 Fuente www.adictosaltrabajo.com. Metodología CRISP-DM.

1.6 Planificación del trabajo

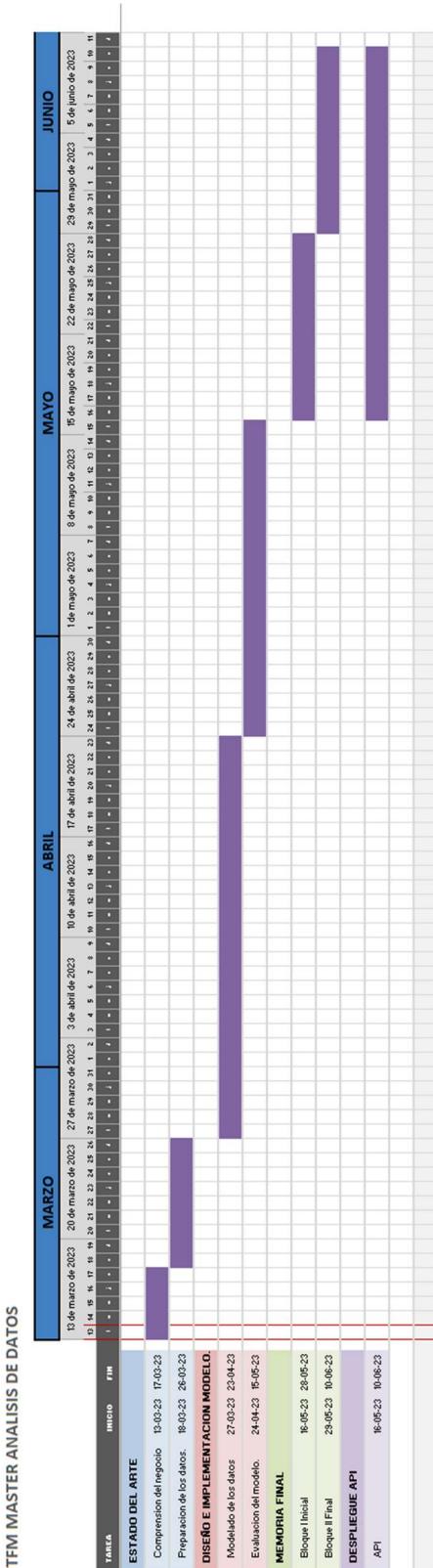


Gráfico 6 - Fuente Diagrama GANTT Desarrollo del proyecto. Elaboración propia.

Gráfico 6 – Grafico Gantt desarrollo del proyecto.

1.7 Breve sumario de productos obtenidos

Entrega 1: PEC 1 - Definición y Planificación del trabajo final.

Es un documento de texto que incluye la introducción, objeto de estudio así como el plan de trabajo que se sigue para la ejecución de este TFM.

Entrega 2: PEC 2 - Estado del Arte

Se componen de un documento de texto donde se efectuará la parte del estado del arte.

Asimismo se procederá a la preparación de los datos del dataset se entregarán archivos de código en formato jupyter, que se complementarán, en caso necesario, con documentos de texto que expliquen las medidas adoptadas.

Entrega 3: PEC 3 - Metodología, diseño, implementación y evaluación del modelo.

En esta fase se establecerá la metodología , se procederá al modelado y evaluación de los datos, para lo cual se entregarán archivos de código en formato jupyter.

Se complementarán con el documento de texto contiene el desarrollo del modelo final obtenido.

Entrega 4: PEC 4 - Redacción de la memoria.

En esta fase se procederá a la redacción de la memoria final en formato texto. Asimismo se entregarán los archivos con código de programación que permiten el análisis realizado.

Entrega 5: Memoria - Memoria, presentación y video presentación.

Documento multimedia en formato video que contiene la presentación formal de este trabajo, así como una copia del material de la presentación en formato de texto.

1.8 Breve descripción de otros capítulos de la memoria

Capítulo 2. Estado del arte:

En primer lugar se efectuará un análisis preliminar del dataset para conocer bien su estructura y distribución, el objetivo es una compresión preliminar de los datos que nos permita identificar los problemas que vamos a encontrar.

Posteriormente se procederá a buscar en la bibliografía los artículos, caso de uso y estudios sobre lucha contra el fraude. En esta fase el objetivo es obtener toda la información posible sobre casos similares y que soluciones se han dado a los problemas previamente identificados de modo que se establezca la metodología a emplear más eficiente para el correcto desarrollo de la tarea.

Capítulo 3. Metodología:

En el que se procederá a definir la metodología, materiales y tareas que se llevaran a cabo para la propuesta de solución.

Capítulo 4. Preparación de los Datos.

En esta parte se procederá a la carga y análisis del dataset, todas las tareas de preprocesamiento, así como las técnicas de submuestreo y/o sobre muestreo que nos permitan la obtención del dataset sobre el que ejecutar los algoritmos.

Capítulo 5. Desarrollo de algoritmos y Análisis de los resultados.

En esta parte se procederá al modelado de los datos, evaluación y análisis del resultado de estos.

Capítulo 6. Conclusiones y trabajo futuro.

Se abordarán las conclusiones y se propondrán líneas para futuras mejoras de este proyecto.

2. Estado del Arte

2.1. Introducción

La detección del fraude con tarjeta de crédito es uno de los apartados más estudiados dentro de la temática de la lucha con el fraude.

A nivel global tenemos 3 materias que estudiar:

- El problema del desbalance del dataset.
- Las técnicas por emplear ya que tenemos múltiples algoritmos.
- Las métricas que seleccionar para medir el rendimiento de estos.

2.2. Pre-Procesado.

El fraude con tarjeta de crédito es un ejemplo de aprendizaje con desequilibrio de clases, en ocasiones la clase de operaciones fraudulentas sea de 0.1% del total del dataset. Esto ocasiona problemas, ya que la mayoría de los algoritmos no están diseñados para operar en estas condiciones y a menudo se produce un sesgo hacia la clase mayoritaria, lo que

supone que en el modelo final la clase minoritaria, fraudulenta en nuestro caso se encuentre mal representada.

Para evitar esto se suele acudir a técnicas de re muestreo las cuales se llevan a cabo como parte del preprocesamiento de datos. Las estrategias principales son: sobre muestreo, submuestreo y técnicas híbridas.

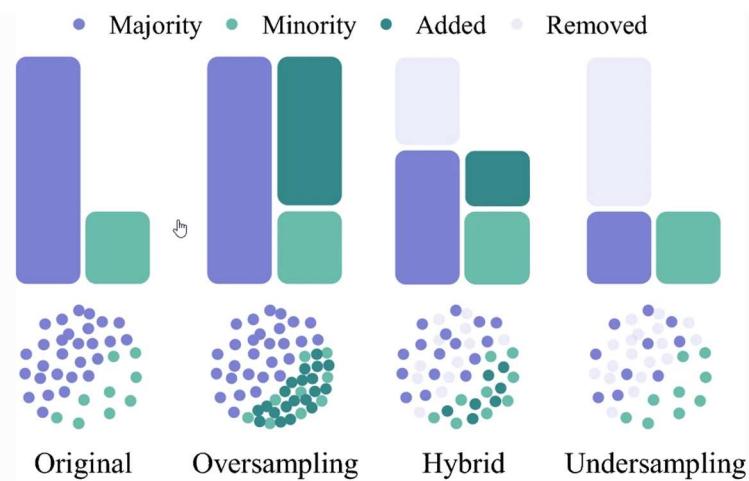


Gráfico 7: Fuente: Imbalanced data preprocessing techniques for machine learning: a systematic mapping study [1]

- **Sobre muestreo:**

Consiste en aumentar la proporción de la clase minoritaria. Puede hacerse de modo aleatorio a partir de los datos existentes o generando nuevos datos sintéticos. En este último caso los métodos más empleados son el SMOTE (técnica de sobre muestreo de minorías sintéticas) y ADASYN (muestreo sintético adaptativo).

- **Submuestreo:**

Consiste en eliminar muestras de la clase mayoritaria. Al igual que en el caso anterior puede hacerse eliminándolas de modo aleatorio, con esta práctica se corre el riesgo de eliminar muestras útiles, en este caso las técnicas más empleadas son eliminación muestras en regiones superpuestas o reemplazando subconjuntos de muestras por sus centroides.

- **Hibridas:**

Esta técnica combina sobre muestreo y submuestreo y contribuye a mejorar el rendimiento de la clasificación.

2.3. Algoritmos.

Existe una gran variedad de literatura de ML aplicado a la detección del fraude, por esto es especialmente valioso la encuesta de C Victoria Priscilla and D Padma Prabha. Credit card fraud detection: a systematic review [3], la cual recopila unos cien trabajos de investigación sobre esta temática identificando las técnicas de ML que se emplearon y cuyo resumen en forma de tabla se recoge en la figura 3.

En su estudio recoge 4 grandes grupos de algoritmos, recopilado, las ventajas y desventajas de cada grupo, los algoritmos que los componen y el número de veces que se emplean. Los más populares por cada grupo son:

1. Aprendizaje Supervisado, Supervised learning:
Máquinas de soporte vectorial, clasificador de red bayesiana, regresión logística, arboles de decisión.
2. Métodos de conjunto:
Bosques aleatorios y boosting.
3. Deep Learning:
Long short-term memory.
Deep Feed Forward NN.
4. Aprendizaje no supervisado:
Hidden Model Markov.
K-Means.

No es el objetivo de este documento examinar todos y cada uno de los algoritmos. Por esta razón es interesante el estudio “Facilitating User Authorization from Imbalanced Data Logs of Credit Cards Using Artificial Intelligence” [4], ya que analiza tres dataset , de distinto tamaño, relacionados con el fraude, y que emplea buena parte de esos algoritmos más comunes. Los Dataset son:

- “Abstract Dataset for Credit Card Fraud Detection” (A)^[5]
Consta de 3075 registros y 11 variables. La proporción de operaciones fraudulentas es un 2.48%, siendo el de menor tamaño.
- “Default of Credit Card Client Dataset” (B). ^[6]
Consta de 30000 observaciones y contempla variables de diversa temática pagos determinados, cuestiones demográficas, datos crediticios, historial de pagos etc. Contempla si hay cumplimiento o no de los pagos del cliente. En este caso la proporción de cliente en no cumplimiento o en mora, o a nuestros efectos categoría fraude seria de un 22,12 %, el más alto de los tres dataset.

Machine learning	Methods	Usage frequency	References	Advantage	Disadvantage
Supervised Learning	Neural Network	10	[8, 9, 15, 17, 20, 27, 32–34, 38]	Highly accurate and reliable	Need to understand and label the input More computation time required for the training phase
	Support Vector Machine	16	[4, 8, 14, 17, 20–24, 27, 30, 38, 41, 48, 52, 53]		
	Bayesian Network Classifiers	15	[4, 9–11, 13, 17, 22, 24, 30, 32, 33, 37–40]		
	K-Nearest Neighbor	5	[9, 13, 22, 25, 30]		
	Logistic Regression	14	[4, 9, 13, 15, 20, 21, 23, 24, 26, 32, 38, 45, 46, 53]		
	Decision Tree	12	[3, 9, 11, 17, 20, 23, 26, 32, 38, 45, 46, 48]		
Unsupervised Learning	Expectation-Maximization	1	[37]	Easy to find unknown patterns and features of data	Computationally complex Less accurate due to unlabeled input
	K-Means	3	[22, 37]		
	Fuzzy C-Means	1	[34]		
	DBSCAN	1	[32]		
	Hidden Markov Model (HMM)	3	[2, 32, 33]		
	Self-Organizing Map (SOM)	1	[32]		
	LINGO	1	[31]		
Ensemble Learning	Random Forest	20	[3, 4, 8–10, 15–17, 20, 21, 23]–[25, 27, 30, 38, 45, 46, 48, 53]	Avoid the overfitting problem and gives better predictions when compared with a single model	Computation time is high Reduces model interpretability due to increased complexity
	Boosting	7	[9, 18, 20, 23, 48, 52, 53]		
	Bagging	3	[10, 12, 22]		
	Voting	1	[10]		
Deep Learning	Stochastic Gradient Descent	1	[9]	No need for feature extraction and labeling of data	A large amount of data is needed to find the pattern Create overfitting problem in the model
	Long short-term memory	3	[4, 16, 19]		
	Deep Feed Forward NN	3	[18, 26, 51]		
	Variational Autoencoder (VAE)	1	[51]		
	Auto Encoder (AE)	2	[29, 41]		
	Restricted Boltzmann Machines	1	[29]		
	Recurrent Neural Network	2	[4, 19]		
	Convolutional Neural Network	1	[27]		
	Generative Adversarial Network	1	[50]		

Gráfico 8: Fuente: C Victoria Priscilla and D Padma Prabha. Credit card fraud detection: a systematic review [2]

- Credit Card Fraud Detection (C) y es el objeto de este TFM, el mayor de los tres.^[7]

En este caso tenemos un total de 284.807 operaciones de las cuales un 0.17% son fraudulentas, siendo el más grande y el que más fuertemente desbalanceado esta de los tres.

El objetivo de este estudio es emplear estos tres dataset para calcular los algoritmos más populares sobre cada uno de ellos, al mismo tiempo obtener las métricas más usuales y analizar los resultados obtenidos.

Los algoritmos que contempla el estudio son:

- RUSboot, (RUSBoost).
Emplea una combinación de undersampling aleatorio (RUS) y el procedimiento AdaBoost, proporcionando un método simple y eficiente para mejorar la clasificación con un conjunto de entrenamiento desbalanceado.
- Customized RUSboot .
- Arboles de Decisión, Decision Tree (DT).
- Regresión Logística, Logistic Regression (LR).
- Perceptrón Multi etiqueta, Multi layer Perceptron (MLP).
- K Vecinos más cercanos, K Near Neighbours (KNN).
- Bosque aleatorio, Random Forest (RF).
- AdaBoost.
- Máquina de soporte vectorial (Supported Vector Machine) SVM.

2.4 Métricas de rendimiento.

En ML hay muchas métricas para evaluar la actuación de un algoritmo.

Dentro de las medidas derivadas de la matriz de confusión tenemos:

La exactitud o accuracy: No es el más indicado para datasets tan fuertemente desequilibrado como los que nos ocupan debido a la alta influencia de los TN (True Negative), en nuestro caso operaciones legítimas. Una alta exactitud no implica un clasificador necesariamente bueno en este caso, ya que componen la clase mayoritaria de forma abrumadora.

La precisión: No se ve tan afectada por las clases fraudulentas, se concentra únicamente en las clases correctamente clasificadas en ambos grupos. Sin embargo al no tener en cuenta las FP (False Negative) se filtran operaciones fraudulentas, lo que también la invalida.

Generalmente, se busca una alta tasa de TP (alta Precisión) y una alta detección de positivos (alto Recall). En la práctica, sin embargo, existe una relación indirecta entre ambas, a medida que aumenta una disminuye la otra, y depende de los objetivos del estudio que combinación encontrar.

Se debe proteger al cliente identificando tantas transacciones fraudulentas como sea posible. Simultáneamente se debe evitar etiquetar incorrectamente demasiadas transacciones para que los usuarios puedan usar sus tarjetas de crédito sin que se le rechacen las transacciones.

Para la emisora de la tarjeta predecir erróneamente una transacción genuina en lugar de fraudulenta también genera un problema ya que tendría que asumirla. La medida F1 combina ambas tanto Precisión como Recall y se calcula como una media armónica.

Estaríamos ante un problema en el que el costo de FN es mayor que el de FP.

Por esta razón, centraríamos la atención en un alto Sensibility / Recall y un alto valor de F1.

Para los tres dataset elegiríamos el modelo Customized RUSboot ya que es el que tiene mejores valores tanto para F1 con 88.6, 97.6 y 99.6 al mismo tiempo que obtiene altos valores de recall con 96.3, 99.6 y 100.

Model name	Sensitivity	Specificity	Precision	F1 score	Model name	Sensitivity	Specificity	Precision	F1 score	Model name	Sensitivity	Specificity	Precision	F1 score
RUSBoost	50.6	99.8	33.4	40.2	RUSBoost	34.6	98.3	85.9	59.4	RUSBoost	34.6	98.3	85.9	59.4
Customized RUSBoost	96.3	85.6	94.2	88.6	Customized RUSBoost	99.6	98.7	95.7	97.6	Customized RUSBoost	99.6	98.7	95.7	97.6
DT	76.5	97.9	72.6	75.4	DT	40.6	81.0	49.5	50.7	DT	40.6	81.0	49.5	50.7
LR	57.0	99.0	86.0	68.7	LR	23.6	97.0	69.6	35.0	LR	23.6	97.0	69.6	35.0
MLP	70.4	99.5	95.8	81.1	MLP	38.5	93.2	61.4	47.3	MLP	38.5	93.2	61.4	47.3
KNN	80.6	99.9	95.1	87.2	KNN	37.8	89.4	50.0	43.1	KNN	37.8	89.4	50.0	43.1
RF	53.2	99.0	82.3	64.5	RF	5.5	99.2	68.2	10.2	RF	5.5	99.2	68.2	10.2
AdaBoost	73.4	99.0	83.7	78.2	AdaBoost	30.8	95.8	67.3	42.3	AdaBoost	30.8	95.8	67.3	42.3
SVM	61.2	99.9	96.8	75.7	SVM	33.2	95.2	67.8	44.5	SVM	33.2	95.2	67.8	44.5

Gráfico 9: Dataset A, B, C

Fuente: Facilitating User Authorization from Imbalanced Data Logs of Credit Cards Using Artificial Intelligence.^[3]

Si hiciésemos un ranking de los distintos métodos, veríamos que este se mantiene independientemente del dataset y del desequilibrio de este.

Otras medidas para evaluar el rendimiento de algoritmos son:

- ROC Curve (Curva Característica Operativa del Receptor).

Es una representación gráfica de la sensibilidad o recall y la especificidad conforme se varía el umbral de discriminación. En el eje x representa la ratio

de falsos positivos y en el eje y el de falsos negativos. Esta herramienta permite seleccionar modelos óptimos y descartar los subóptimos. El área debajo de la curva (AUC) se emplea como resumen de la calidad del modelo. Oscila entre 0 y 1, valores por encima de 0.8 se consideran buenos clasificadores.

- PR Curve (Curva Precisión-Recall).

Se obtiene trazando precisión contra el recall.

Un alto AUC en esta curva implica un alto recall y precisión, alta precisión implica baja tasa de falsos positivos y alta recall con una baja tasa de falsos.

negativos. En este caso el clasificador devuelve valores precisos, lo predicho se ajusta a la realidad.

Un sistema que con un alto recall y una baja precisión devuelve muchos resultados con una alta tasa de valores que son incorrectos.

Un sistema con alta precisión pero bajo recall retorna pocos resultados pero la mayoría de ellos son correctos.

En el caso de nuestros dataset volvemos a elegir Customized RUSboot tanto para curva ROC como PR.

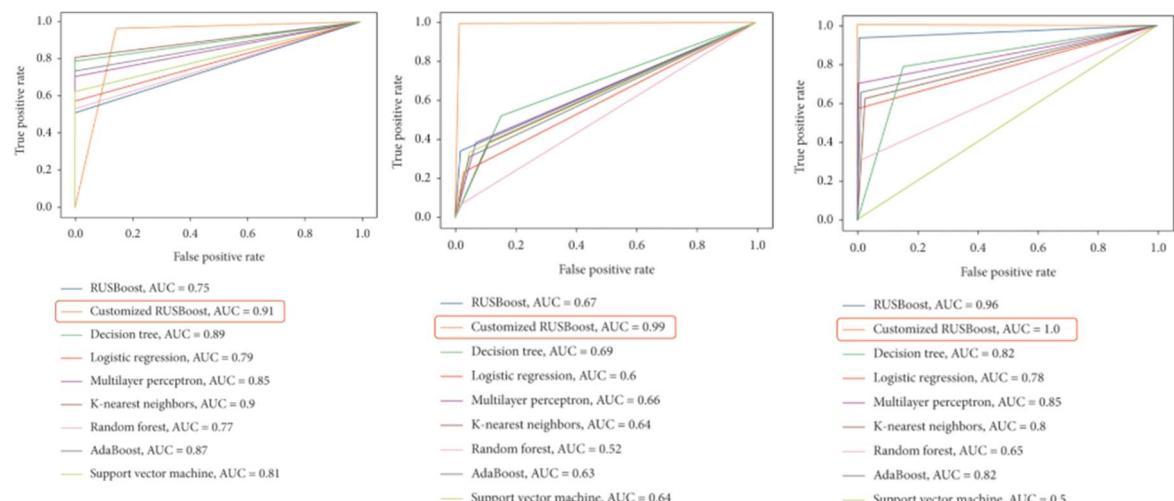


Gráfico 10: Dataset A, B, C.

Fuente: Facilitating User Authorization from Imbalanced Data Logs of Credit Cards Using Artificial Intelligence. [3]

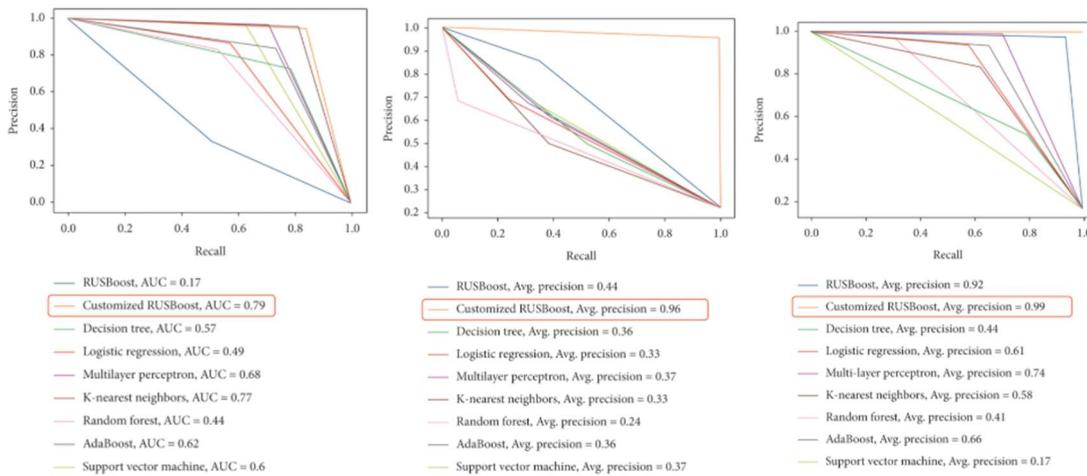


Gráfico 10: Dataset A, B, C.

Fuente: Facilitating User Authorization from Imbalanced Data Logs of Credit Cards Using Artificial Intelligence [3]

2.5 Caso de uso.

El fin último del presente TFM es la obtención de un modelo de predicción de fraude en una API. Por esta razón, a continuación examinamos el caso de la aplicación Sherlock de Revolut.^[7]

Revolut es una Fintech, esto es, una empresa que emplea la tecnología para ofrecer servicios y procesos financieros. Son una alternativa a la banca tradicional y ofertan una tarjeta con la que operar y una cuenta digital en la que guardar el dinero.

Sherlock es la herramienta de lucha contra el fraude con tarjetas basado en aprendizaje automático que Revolut emplea. Supervisa de forma continua y autónoma las transacciones de los usuarios de la firma. Si Sherlock encuentra un cargo sospechoso, bloquea la compra y congela la tarjeta en tiempo real. Mediante una notificación se solicita confirmación de la legitimidad o no de la operación. Si el cargo es legítimo la tarjeta se desbloquea y se repite la compra validándola, si es fraudulenta la tarjeta se cancela y se podría solicitar un reemplazo de esta gratuito.

El objetivo de Sherlock es minimizar las perdidas por fraude de Revolut así como las pérdidas causadas por bloqueos incorrectos. El funcionamiento de Sherlock es un problema de clasificación binaria similar al estudiado hasta ahora y al que se le añade una serie de especificidades que complementan y enriquecen lo anteriormente expuesto.

El origen de datos de Sherlock son las bases de datos de operaciones fraudulentas de la empresa. Para facilitar su manipulación se volcaron diariamente a Big Query en Google Cloud.

Tras esto se procedió a clasificar las operaciones asignándoles una puntuación en base a una serie de características:

- 1.- Características sin procesamiento o mínimo, básicamente, identificación del establecimiento, monto de la operación y hora de transacción.
- 2.- Funciones centradas en el usuario y comparativas con su historial de cliente, consistente en evaluar si la operación en curso difería del histórico del cliente.
- 3.- Funciones centradas en el establecimiento y que comparan la operación actual con el histórico de ese establecimiento y que son similares a las del apartado anterior pero para la otra parte implicada en la transacción, haciendo hincapié en el historial de operaciones fraudulentas de este.

Efectuado esto se obtiene el dataset, se divide y se procede a probar algunos de los algoritmos que hemos empleado en este documento. Finalmente eligen un aumento de gradiente de árboles de decisión por las siguientes razones:

- Las métricas obtenidas superaban a sus rivales así como la velocidad de inferencia.
- Es robusto con datos heterogéneos, de distinta fuente, así como el tratamiento de variables tanto numéricas como categóricas.
- Fácil de implementar por no requerir muchos ajustes de hiperparámetros.

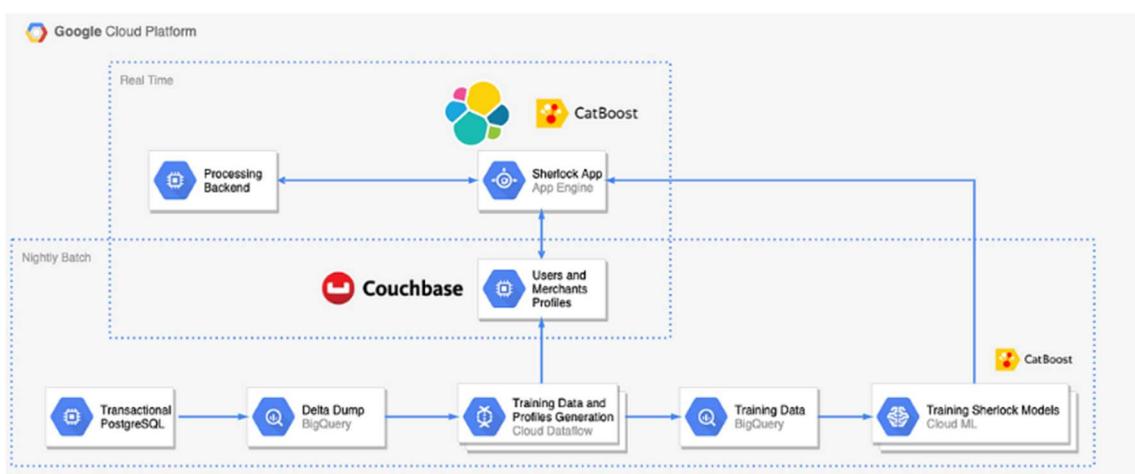


Gráfico 11:

Fuente Building a state-of-the-art card fraud detection system in 9 months. [7]

3 Metodología

Recordamos que para la realización de presente TFM se emplea la metodología CRISP-DM.

En este parte se desglosará las etapas de dicha metodología en capítulos al tiempo que se describe brevemente el contenido que se desarrollará en cada uno de ellos.

Comprensión del negocio.

El objetivo es la consecución de un modelo de predicción de fraude sobre el dataset <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud> alojado en Kaggle.

En el capítulo anterior, el estado del arte se ha efectuado un estudio de las principales técnicas empleadas para el análisis de este tipo de datasets. Esta primera fase se ha desarrollado en el mismo. Se han estudiado los grandes grupos de algoritmos con los que se aborda un problema de esta naturaleza y se ha estudiado un caso de uso concreto, el de Revolut y el algoritmo empleado CatBoost.

Es por ello por lo que puede avanzarse los algoritmos a emplear en la parte de análisis. Intentando huir de las clasificaciones de supervisados no supervisados habitual, se establece la siguiente:

- **A. Lineales** su representante es la **Regresión Logística**.
- **A. Boosting** el caso de uso del estado del arte , el de Revolut, comparte similitudes con el dataset objeto del TFM, por lo que se estudiara esta categoría con especial atención. Eso sí, todas las transformaciones y las decisiones tomadas sobre el dataset afectan por igual a todos los algoritmos seleccionados. Esto es, se le presta mayor atención por ser la solución de un caso de uso real, pero sin orientar las transformaciones, a que la decisión de elección final de modelo se encamine al uso de algún algoritmo de esta categoría.

Entre los candidatos tenemos **AdaBoost**, **XGBoost** y por último **CatBoost**, el empleado por Revolut.

- **A. basados en Arboles de decisión**

En esta categoría tenemos el **Decisión Tree** y el **Random Forest**.

- **A. de Redes Neuronales** como representante de este tipo de algoritmos el **MLP**.

- **A. basados en distancia**, se analizará el **KNN**.

En total 8 algoritmos se postulan como candidatos, con las debidas transformaciones y ajustes, al modelo final. Dicho número pretende representar debidamente los 4 grandes grupos de algoritmos descritos en la parte de estado del arte al tiempo dentro de los límites TFM.

4 Comprensión de los datos.

Esta fase se abordará en el siguiente capítulo Preparación de los Datos en su primera parte Análisis Exploratorio de Datos (EDA).

Puede dividirse en varias partes:

- Un análisis exploratorio de Datos (EDA) donde se procederá a estudiar las variables y grupos de variables que componen el dataset.
- Identificación de patrones ocultos o características de las variables que puedan contribuir a modelar los algoritmos de modo más eficaz.

A ese respecto podemos avanzar que los principales problemas que presenta este dataset son los siguientes:

- El problema fundamental del dataset es el desequilibrio entre las clases que lo componen.
- Las variables de características se encuentran anonimizadas, esto es, se han transformado mediante PCA.
- Las variables presentan una gran cantidad de outliers.
- Por último un número significativo de variables presentan un alto grado de sesgo (skewness).

Preparación de los datos

Las características anteriormente enunciadas condicionan el resto de las fases pendientes. Puede avanzarse las siguientes medidas a modo de introducción para paliarlas o solucionarlas.

Se procederá al escalado de variables a fin de reducir el número de outliers.

En el caso del desequilibrio de clases se procederá a emplear técnicas de sobre muestreo, en concreto, Smote y Adasyn.

En el caso del sesgo se implementará una transformación de Power Transformation tras proceder a un preanálisis de los algoritmos a implementar. Toda esta fase se abordará en la parte final del capítulo 4 tras el análisis exploratorio.

Modelado de los datos

En primer lugar se calcularán todos los algoritmos en su configuración por defecto, esto es, sin explicitar ningún parámetro con las distintas distribuciones.

Esta primera aproximación tiene como objetivo determinar un intervalo de rendimiento máximo y mínimo para cada algoritmo.

Tras ello para cada algoritmo se determinará un parámetro principal y se procederá a calcular de nuevo los algoritmos.

Los modelos con parámetro principal que no alcancen el mínimo del modelo sin parámetros serán eliminados ya que se consideraran soluciones subóptimas.

Efectuado esto y usando la medida de rendimiento seleccionada se efectuará un ranking de los mejores modelos con un parámetro. Esta parte es sin duda la más discrecional ya que las soluciones a desarrollar en ocasiones tendrán que determinarse visualmente pues constituyen un intervalo.

Determinado el mejor modelo con parámetro principal se procederá a aplicar un grid para determinar parámetros secundarios que proporcionen la solución final optimizada.

Evaluación

Esa solución de modelo optimizada con parámetros secundarios, final para cada uno de los modelos, se comparará con la solución inicial sin parámetros. Si esa solución optimizada no es capaz de mejorar el rendimiento del mejor modelo sin parámetros, descartamos ese modelo definitivamente. No tiene sentido considerar ese modelo a candidato final, si existe otro que sin optimizar, y por tanto son mayor recorrido de mejora, le supera.

Con este procedimiento se obtendrá la lista final de candidatos a mejor algoritmo y de ella emergerá el modelo final óptimo.

Despliegue

Lamentablemente esta fase no podrá llevarse a cabo dentro del límite tiempo, constituye la principal mejora futura de este proyecto.

4. Preparación de los datos

4.1. Análisis Exploratorio de Datos (EDA)

En este capítulo se presenta el dataset objeto de este TFM, el cual se encuentra alojado en Kaggle en la siguiente dirección:

<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>.

Contiene las transacciones realizadas por titulares de tarjetas de crédito ubicados en Europa en el lapso de dos días de septiembre de 2013. Se desconoce en qué días se produjo la captura de los datos si son laborables, fin de semana o festivos con las implicaciones que ello comporta en lo que afecta al comportamiento de uso de las tarjetas. La estructura general del dataset se muestra en la Figura 1 y pueden distinguirse los siguientes elementos.

Time: Recoge el momento en que se efectúa la operación. Se expresa en segundos transcurridos desde que se efectuó la anterior.

Características: Comprende las variables denominadas V1 a V28, dichas variables se encuentran anonimizadas mediante PCA, para preservar datos sensibles de la naturaleza de la operación.

Amount: Expresa el importe de la operación expresada en euros.

Class: Es la variable dependiente de tipo dicotómico que recoge la información sobre la legitimidad o no de la operación. (0.- Autentica o legítima y 1.- Fraude).

El dataset se compone de 284807 registros o columnas y 31 variables de las cuales 284315 operaciones legítimas o auténticas que suponen el 99.83% del total. En tanto que tenemos 492 operaciones fraudulentas que suponen el 0.17% del total.

No existen valores nulos en ninguna de las variables que lo componen.

Labels	Original	% Ori	No_Outliers	% No_O
Authentic	284315.0	99.83	203351.0	99.99
Fraud	492.0	0.17	26.0	0.01
TOTAL	284807.0	100.00	203377.0	100.00

Gráfico 12: Resumen total operaciones. Elaboración propia.

A continuación se muestra las métricas principales del dataset y su estructura general.

	count	mean	std	min	25%	50%	75%	max
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	139320.500000	172792.000000
V1	284807.0	3.918649e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930
V2	284807.0	5.682686e-16	1.651309	-72.715728	-0.598550	0.065486	0.803724	22.057729
V3	284807.0	-8.761736e-15	1.516255	-48.325589	-0.890365	0.179846	1.027196	9.382558
V4	284807.0	2.811118e-15	1.415869	-5.683171	-0.848640	-0.019847	0.743341	16.875344
V5	284807.0	-1.552103e-15	1.380247	-113.743307	-0.691597	-0.054336	0.611926	34.801666
V6	284807.0	2.040130e-15	1.332271	-26.160506	-0.768296	-0.274187	0.398565	73.301626
V7	284807.0	-1.698953e-15	1.237094	-43.557242	-0.554076	0.040103	0.570436	120.589494
V8	284807.0	-1.893285e-16	1.194353	-73.216718	-0.208630	0.022358	0.327346	20.007208
V9	284807.0	-3.147640e-15	1.098632	-13.434066	-0.643098	-0.051429	0.597139	15.594995
V10	284807.0	1.772925e-15	1.088850	-24.588262	-0.535426	-0.092917	0.453923	23.745136
V11	284807.0	9.289524e-16	1.020713	-4.797473	-0.762494	-0.032757	0.739593	12.018913
V12	284807.0	-1.803266e-15	0.999201	-18.683715	-0.405571	0.140033	0.618238	7.848392
V13	284807.0	1.674888e-15	0.995274	-5.791881	-0.648539	-0.013568	0.662505	7.126883
V14	284807.0	1.475621e-15	0.958596	-19.214325	-0.425574	0.050601	0.493150	10.526766
V15	284807.0	3.501098e-15	0.915316	-4.498945	-0.582884	0.048072	0.648821	8.877742
V16	284807.0	1.392460e-15	0.876253	-14.129855	-0.468037	0.066413	0.523296	17.315112
V17	284807.0	-7.466538e-16	0.849337	-25.162799	-0.483748	-0.065676	0.399675	9.253526
V18	284807.0	4.258754e-16	0.838176	-9.498746	-0.498850	-0.003636	0.500807	5.041069
V19	284807.0	9.019919e-16	0.814041	-7.213527	-0.456299	0.003735	0.458949	5.591971
V20	284807.0	5.126845e-16	0.770925	-54.497720	-0.211721	-0.062481	0.133041	39.420904
V21	284807.0	1.473120e-16	0.734524	-34.830382	-0.228395	-0.029450	0.186377	27.202839
V22	284807.0	8.042109e-16	0.725702	-10.933144	-0.542350	0.006782	0.528554	10.503090
V23	284807.0	5.282512e-16	0.624460	-44.807735	-0.161846	-0.011193	0.147642	22.528412
V24	284807.0	4.456271e-15	0.605647	-2.836627	-0.354586	0.040976	0.439527	4.584549
V25	284807.0	1.426896e-15	0.521278	-10.295397	-0.317145	0.016594	0.350716	7.519589
V26	284807.0	1.701640e-15	0.482227	-2.604551	-0.326984	-0.052139	0.240952	3.517346
V27	284807.0	-3.662252e-16	0.403632	-22.565679	-0.070840	0.001342	0.091045	31.612198
V28	284807.0	-1.217809e-16	0.330083	-15.430084	-0.052960	0.011244	0.078280	33.847808
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	77.165000	25691.160000
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000	0.000000	1.000000

Gráfico 13: Resumen métricas principales variables de características Elaboración propia.

4.1.1. Variable Time

Las operaciones de los dos días del dataset se recogen en segundos comprendidos en ese lapso. Esto permite reconstruir el histórico por días y por horas y analizarlo debidamente. La tabla adjunta nos muestra una distribución bastante homogénea de las operaciones fraudulentas, 492 en total.

Day	Fraud	Authentic	Total	% Fraud
0	1	281	144506	0.19
1	2	211	139809	0.15

Gráfico 14: Desglose total operaciones dataset. Elaboración propia.

En el grafico se muestra separados por día, por tipo de operación y por franja horaria las operaciones realizadas.

Las operaciones legítimas muestran una distribución similar para ambos días. Cabe mencionar el hecho de que hay una reducción del número de operaciones global efectuado entre las 2 y las 7 am, momento en el que los titulares de estas están durmiendo. En el caso de las operaciones fraudulentas puede apreciarse un notable incremento de estas, acercándose a la media para el todo el día.

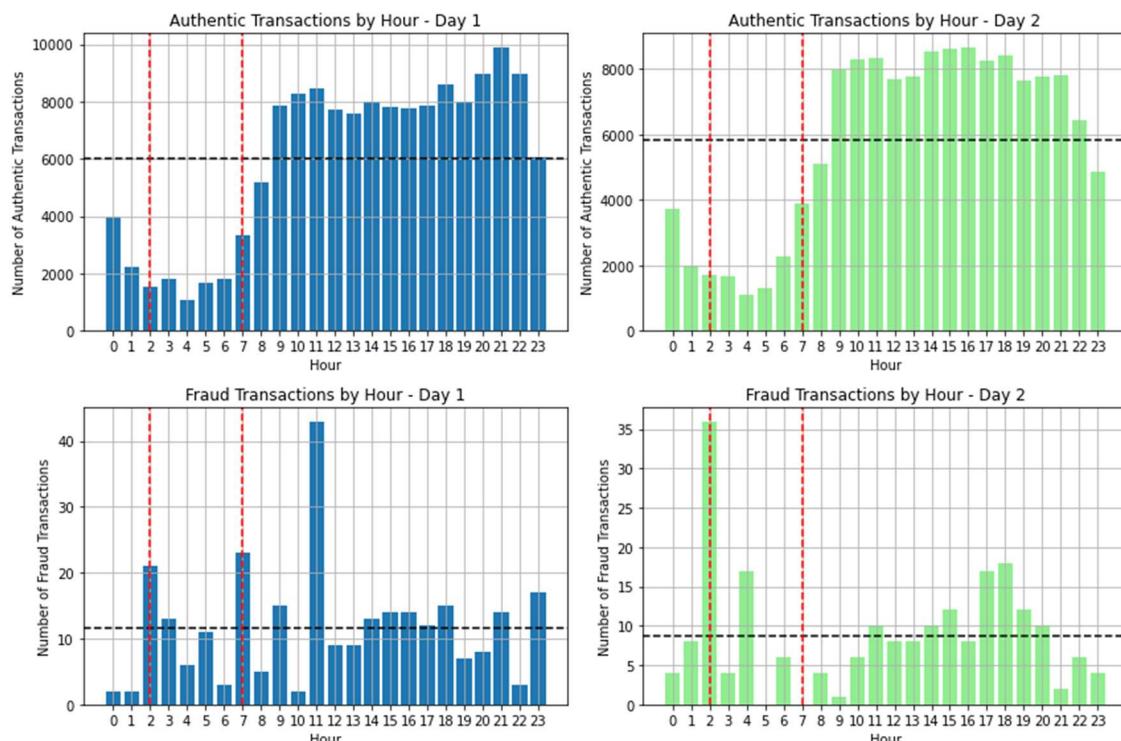


Gráfico 15: Gráficos distribución operaciones auténticas vs fraudulentas por día y por franja horaria. Elaboración propia.

A continuación se muestran el número de operaciones por hora efectuado en ambos días.

La columna % Fraud muestra la proporción de operaciones fraudulentas para esa hora. La media de las operaciones fraudulentas global se sitúa en un 0,17 % y que sirve de referencia para la clasificación que se realiza en la columna Clasif.

El valor de las operaciones fraudulentas de esa franja horaria es inferior a la media nos devolverá el valor 'Below'.

El número de fraudulentas esta entre la media y su doble, esto es 0.17 y 0.34 % devolverá 'Above'

En caso de que supere el valor de 0.35 % tendremos una lectura 'Extreme'

La primera medida a tomar seria implementar un protocolo para evitar la suplantación de la identidad del titular.

		Fraud	Authentic	Total	% Fraud	Clasif		index	Fraud	Authentic	Total	% Fraud	Clasif	
Day	Hour						Day	Hour						
Day 1	0	2.000000	3962	3964	0.050000	Below	Day 2	0	24	4.000000	3727	3731	0.110000	Below
	1	2.000000	2215	2217	0.090000	Below		1	25	8.000000	1995	2003	0.400000	Extreme
	2	21.000000	1555	1576	1.330000	Extreme		2	26	36.000000	1716	1752	2.050000	Extreme
	3	13.000000	1808	1821	0.710000	Extreme		3	27	4.000000	1667	1671	0.240000	Above
	4	6.000000	1076	1082	0.550000	Extreme		4	28	17.000000	1110	1127	1.510000	Extreme
	5	11.000000	1670	1681	0.650000	Extreme		5	46	0.000000	1309	1309	0.000000	Below
	6	3.000000	1828	1831	0.160000	Below		6	29	6.000000	2264	2270	0.260000	Above
	7	23.000000	3345	3368	0.680000	Extreme		7	47	0.000000	3875	3875	0.000000	Below
	8	5.000000	5174	5179	0.100000	Below		8	30	4.000000	5093	5097	0.080000	Below
	9	15.000000	7863	7878	0.190000	Above		9	31	1.000000	7959	7960	0.010000	Below
	10	2.000000	8286	8288	0.020000	Below		10	32	6.000000	8304	8310	0.070000	Below
	11	43.000000	8474	8517	0.500000	Extreme		11	33	10.000000	8329	8339	0.120000	Below
	12	9.000000	7723	7732	0.120000	Below		12	34	8.000000	7680	7688	0.100000	Below
	13	9.000000	7576	7585	0.120000	Below		13	35	8.000000	7772	7780	0.100000	Below
	14	13.000000	8016	8029	0.160000	Below		14	36	10.000000	8531	8541	0.120000	Below
	15	14.000000	7822	7836	0.180000	Above		15	37	12.000000	8613	8625	0.140000	Below
	16	14.000000	7772	7786	0.180000	Above		16	38	8.000000	8659	8667	0.090000	Below
	17	12.000000	7870	7882	0.150000	Below		17	39	17.000000	8267	8284	0.210000	Above
	18	15.000000	8592	8607	0.170000	Below		18	40	18.000000	8414	8432	0.210000	Above
	19	7.000000	7987	7994	0.090000	Below		19	41	12.000000	7643	7655	0.160000	Below
	20	8.000000	8972	8980	0.090000	Below		20	42	10.000000	7766	7776	0.130000	Below
	21	14.000000	9881	9895	0.140000	Below		21	43	2.000000	7806	7808	0.030000	Below
	22	3.000000	8974	8977	0.030000	Below		22	44	6.000000	6458	6464	0.090000	Below
	23	17.000000	6065	6082	0.280000	Above		23	45	4.000000	4852	4856	0.080000	Below

Gráfico 16: Desglose operaciones con probabilidad de fraude asociada a la franja horaria sobre el fraude medio horario y clasificación. Elaboración propia.

4.1.2. Variables Características V1-V28

Este grupo de variables se ha tratado con un PCA para preservar la identidad del titular y los datos sensibles inherentes a la operación.

Por esta razón no procede efectuar un profundo análisis , ya que como consecuencia del PCA no existen altas correlaciones entre las variables.

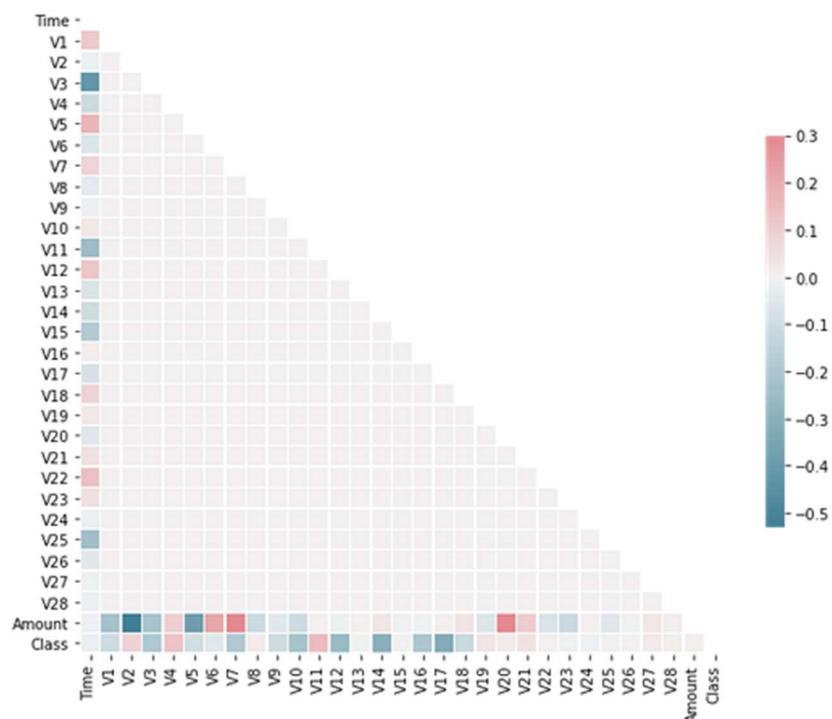


Gráfico 17: Heatmap de matriz de correlación de variables V1-V28. Elaboración propia.

Si hay que destacar el elevado sesgo (skewness) presente en las variables, cuyo tratamiento se abordara posteriormente.

La siguiente tabla recoge aquellas variables que presentan sesgo, considerando como tal, todo aquel que exceda de +/- 1. Con tal criterio, puede observarse que está presente en 17 de las 28 variables oscilando entre un - 8.52 y 11.19.

Features	V8	V23	V2	V17	V1	V5	V12	V3	V20	V14	V27	V16	V10	V6	V7	V21	V28
0 Skewness	-8.52	-5.88	-4.62	-3.84	-3.28	-2.43	-2.28	-2.24	-2.04	-2.0	-1.17	-1.1	1.19	1.83	2.55	3.59	11.19

Gráfico 18: Listado valor del sesgo variables de características. Elaboración propia.

4.1.3. Variable Amount

Este dataset está fuertemente desequilibrado por lo que la clase total, responde al comportamiento de la clase mayoritaria como puede observarse en las siguientes gráficas y la tabla adjunta.

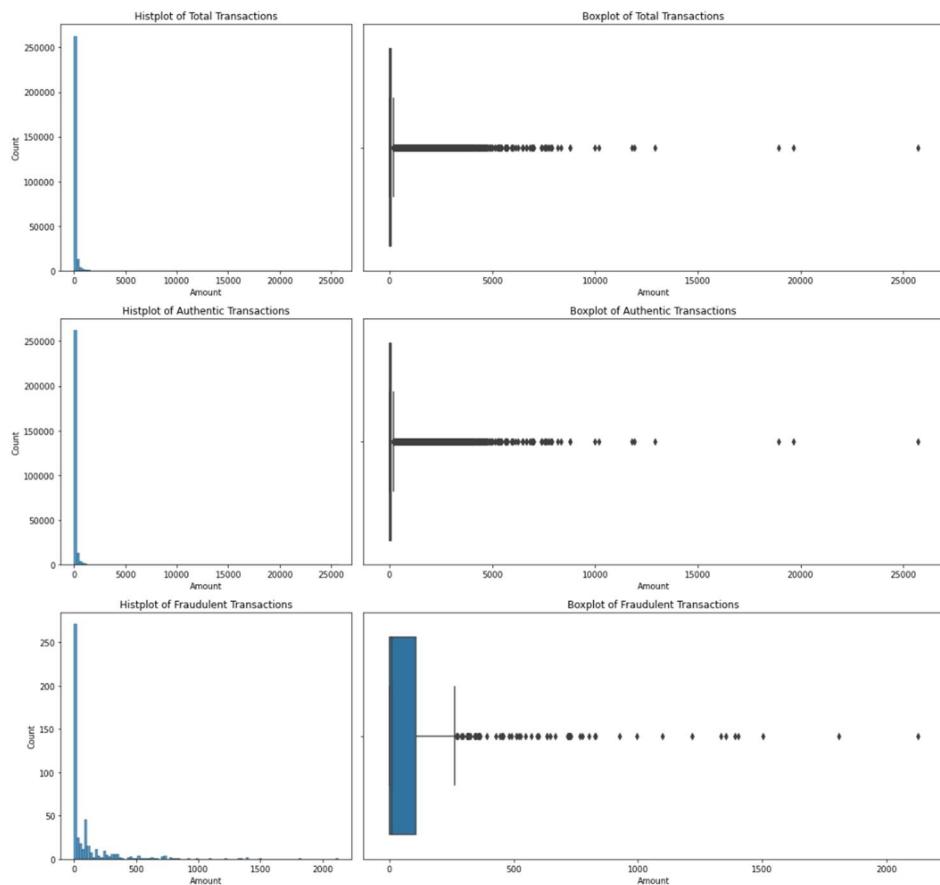


Gráfico 19: Histograma y boxplot operaciones auténticas, fraudulenta y total
Elaboración propia.

	Authentic	Fraud
count	284315.000000	492.000000
mean	88.291022	122.211321
std	250.105092	256.683288
min	0.000000	0.000000
25%	5.650000	1.000000
50%	22.000000	9.250000
75%	77.050000	105.890000
max	25691.160000	2125.870000

Gráfico 20: Métricas distribución de operaciones auténticas vs fraudulentas.
Elaboración propia.

4.2. Ajustes del dataset

Finalmente tras analizar todas las variables queda efectuar las transformaciones para dejar listo el dataset para ejecutar los algoritmos.

4.2.1. Escalado de variables

Para paliar el problema de los outliers de la variable Amount se somete la variable a una transformación Robust Scaler, con objeto de centrar los valores alrededor de cero y obtener una dispersión relativa similar. Se recurre a esta transformación por su eficacia para la aplicación de algoritmos que se basan en las distancias, los sensibles a la escala de características y métodos basados en el gradiente. En el grafico puede apreciarse como se reduce la diferencia de escala en la variable Amount

4.2.2. Separación del dataset

Asimismo, se procede a separar el dataset en conjunto de entrenamiento (train) y de validación (test), en una proporción 80%-20%, por ser la empleada habitualmente para este tipo de estudios. La tabla muestra el estado en el que queda ambas particiones

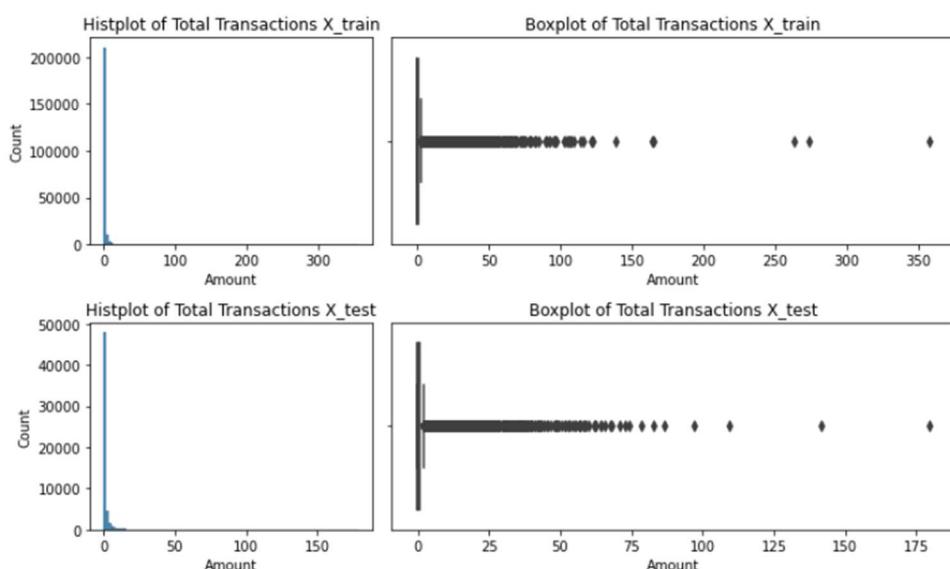


Gráfico 21: Histplot y Boxplot de dataset train y test. Elaboración propia.

Labels	Train	%	Test	%	Total	%
Authentic	227845.0	80.00	56962.0	20.00	284807.0	99.83
Fraud	394.0	80.08	98.0	19.92	492.0	0.17
TOTAL	228239.0	80.00	57060.0	20.00	285299.0	100.00

Gráfico 22: Distribución de operaciones dataset train y test. Elaboración propia.

4.2.3. Tratamiento del sesgo de las variables y desequilibrios de clases

Las dos principales conclusiones extraídas del EDA son:

- Existencia de sesgo en varias variables, en algunas de cierta importancia.
- Gran desequilibrio entre las clases legitima y fraude.

A fin de reducir el sesgo se ha implementado una transformación de tipo Power Transformation, en adelante PT, que proporciona el paquete sklearn. Su efectividad dependerá en gran medida del tipo de algoritmo que se esté analizando.

Para reducir el problema del desequilibrio de clases se efectuarán transformaciones con smote y adasyn.

Con esta lógica se obtienen 6 distribuciones:

- OR_origin dataset original, con desequilibrio de clases y con variables con sesgo
- OR_smote dataset original con clases equilibradas mediante smote y con variables con sesgo.
- OR_adasyn con clases equilibradas mediante adasyn y con variables con sesgo
- PT_original dataset con reducción de sesgo con PT y con desequilibrio de clases.
- PT_smote dataset con reducción de sesgo con PT y equilibrio de clases mediante smote.
- PT_adasyn con reducción de sesgo con PT y equilibrio de clases mediante adasyn.

El objetivo es obtener unas primeras conclusiones sobre que distribuciones funcionan mejor para un determinado algoritmo, estando este con su configuración por default. Esto es, establece una primera clasificación del rendimiento del algoritmo:

- Nivel interno: Todos los ajustes de parámetros que supongan un rendimiento en términos de F2 Score inferior al default se consideraran subóptimas y desechadas automáticamente.
- Nivel externo: Permitirá obtener un primer ranking de algoritmos en términos de F2 Scores.

A continuación se muestran los resultados obtenidos para todos los algoritmos con el formato que se va a emplear en el resto de TFM.

En las mismas, se muestra nombre del algoritmo, descripción que contiene la distribución empleada, los parámetros del algoritmo, Default en este caso, acompañados por las siguientes medidas :

- ROC-AUC.
- Derivadas de la matriz de confusión: Accuracy, Precision, Recall.
- F1-Score y F2-Score.
- Matriz de confusión.

Dado que se van a comparar distribuciones distintas, con clases equilibradas y sin equilibrar se va a emplear F2-Score y la matriz de confusión, ya que son los elementos que más intuitivamente permiten visualizar los resultados. En concreto, la razón de emplear F2-Score o F-beta score, obedece a su formulación.

La precisión es la proporción de resultados positivos correctos entre todos los resultados positivos predichos por el modelo. Mide la exactitud del modelo al predecir los casos positivos. Por otro lado, el recall es la proporción de resultados positivos correctos entre todos los casos positivos reales. Mide la capacidad del modelo para encontrar todos los casos positivos.

El F2-score da más importancia a la capacidad del modelo para encontrar los casos positivos que a su precisión. La fórmula para calcular el F2-score es la siguiente:

$$\text{F2-score} = (1 + 2^2) * ((\text{precision} * \text{recall}) / ((2^2 * \text{precision}) + \text{recall}))$$

El valor F2-score oscila entre 0 y 1, donde 1 indica una precisión y un recall perfectos, y 0 indica un desempeño pobre.

El F2-score es útil en casos donde el recall es más importante que la precisión. Por ejemplo, en la detección de enfermedades, es fundamental minimizar los casos de falsos negativos, es decir, asegurarse de que no se pasen por alto pacientes con enfermedades o el caso que nos ocupa la detección del fraude con tarjeta de crédito.

Los FN suponen un problema grave para la entidad emisora ya que tendría que reembolsarlos al titular. Los FP son un perjuicio grave para el cliente ya que se bloquearía la tarjeta ocasionándole, dependiendo del contexto, un grave perjuicio.

En este caso, se daría más importancia a tener un alto recall, incluso a costa de tener una precisión ligeramente menor y es especialmente útil en casos donde la detección de casos positivos es crítica.

Como complemento la matriz de confusión es un elemento que permite visualizar fácilmente las diferencias en los F2-Scores alcanzado por cada modelo.

1.- Regresión Logística

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	regression_logistic	OR origin	Default	0.826416	0.999175	0.831169	0.653061	0.731429	0.682303	[[56851 13],[34 64]]
1	regression_logistic	OR smote	Default	0.946029	0.973596	0.056747	0.918367	0.106888	0.227503	[[55368 1496],[8 90]]
2	regression_logistic	OR adasyn	Default	0.922704	0.916857	0.018880	0.928571	0.037007	0.087299	[[52135 4729],[7 91]]
3	regression_logistic	PT origin	Default	0.836603	0.999175	0.814815	0.673469	0.737430	0.697674	[[56849 15],[32 66]]
4	regression_logistic	PT smote	Default	0.944807	0.971156	0.052174	0.918367	0.098738	0.212565	[[55229 1635],[8 90]]
5	regression_logistic	PT adasyn	Default	0.920330	0.912117	0.017878	0.928571	0.035081	0.082999	[[51865 4999],[7 91]]

Gráfico 23: Tabla inicial modelo Regresión Logística. Elaboración propia.

Existe una alta variabilidad de los resultados.

Como puede observarse las mejores soluciones se alcanza para los modelos PT_origin y OR_origin con R2-Score 0.6976 y 0.6823 respectivamente.

Los valores de F2-Score, con sobre muestreo, son muy baja ya que la clase FP no se captura correctamente las clases FN se clasifican correctamente pero a costa de tener un recuento de FP inaceptable.

2.- AdaBoost

Este algoritmo presenta una distribucion de F2 Score más compacta respecto al anterior. Aun así , puede apreciarse que las distribuciones con desequilibrio de clases , esto es OR_origin y PT_origin son las que obtienen valores más altos entre 0.72-0.75.

Las transformaciones con smote y adasyn obtienen valores muy parecidos ya sea con sesgo o sin él, de un 0.62 - 0.65 para smote y 0.62 - 0.63 con adasyn.

Model	Description	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	adaboost	OR origin	0.862025	0.999087	0.739583	0.724490	0.731959	0.727459 [[56839, 25], [27, 71]]]
0	adaboost	OR smote	0.886586	0.997279	0.363636	0.775510	0.495114	0.632280 [[56731, 133], [22, 76]]]
0	adaboost	OR adasyn	0.871385	0.997437	0.376289	0.744898	0.500000	0.622867 [[56743, 121], [25, 73]]]
0	adaboost	PT origin	0.877314	0.999105	0.732673	0.755102	0.743719	0.750507 [[56837, 27], [24, 74]]]
0	adaboost	PT smote	0.906976	0.997314	0.372093	0.816327	0.511182	0.658979 [[56729, 135], [18, 80]]]
0	adaboost	PT adasyn	0.876513	0.997507	0.385417	0.755102	0.510345	0.633562 [[56746, 118], [24, 74]]]

Gráfico 24: Tabla inicial modelo AdaBoost. Elaboración propia.

3.- XgBoost

Presenta una gran homogeneidad de valores F2-Score y con lecturas realmente altas. Maneja realmente bien todas las opciones, independientemente de la presencia o no de sesgo y de presencia o ausencia desequilibrio de clases, todas se ubican en una banda muy estrecha de entre 0.82 y 0.84.

Hay que destacar especialmente el caso de OR_smote, es el primer caso en que FN se ubica en el umbral cercano a las 10 observaciones de FN, 11 concretamente, y todo ello con un valor de FP de 34. Esto es, la diagonal secundaria de la matriz de confusión presenta valores muy bajos, lo que posibilita un F2 Score muy elevado y un ROC-AUC, medida secundaria, realmente elevado. En la explicación y análisis de esta medida, no nos vamos a detener en este momento del estudio. Se empleará más adelante cuando haya que discriminar entre soluciones mucho más cercanas en términos de F2 Score, y será la que se emplee, llegado el caso, para dilucidar que algoritmo elegir.

Esta solución es particularmente interesante, más si cabe, en la fase tan temprana de parámetros default, ya que es muy consistente.

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	xgboost	OR origin	Default	0.908102	0.999561	0.919540	0.816327	0.864865	0.835073
1	xgboost	OR smote	Default	0.943579	0.999210	0.719008	0.887755	0.794521	0.847953
2	xgboost	OR adasyn	Default	0.923188	0.999175	0.721739	0.846939	0.779343	0.818540
3	xgboost	PT origin	Default	0.908102	0.999561	0.919540	0.816327	0.864865	0.835073
4	xgboost	PT smote	Default	0.928290	0.999192	0.724138	0.857143	0.785047	0.826772
5	xgboost	PT adasyn	Default	0.933331	0.999087	0.685484	0.867347	0.765766	0.823643

Gráfico 25: Tabla inicial modelo XgBoost. Elaboración propia.

4.- CatBoost

Este algoritmo es neutro respecto al sesgo para todas las distribuciones y el equilibrio o desequilibrio de clases afecta ligeramente los resultados. No obstante las distribuciones con desequilibrio de clases son las que mejor resultado F2 Score obtiene.

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	catboost	OR origin	Default	0.903017	0.999579	0.940476	0.806122	0.868132	0.829832
1	catboost	OR smote	Default	0.933093	0.998613	0.562914	0.867347	0.682731	0.782689
2	catboost	OR adasyn	Default	0.922924	0.998648	0.572414	0.846939	0.683128	0.772812
3	catboost	PT origin	Default	0.903017	0.999579	0.940476	0.806122	0.868132	0.829832
4	catboost	PT smote	Default	0.912896	0.998964	0.658537	0.826531	0.733032	0.786408
5	catboost	PT adasyn	Default	0.917734	0.998455	0.532468	0.836735	0.650794	0.750916

Gráfico 26: Tabla inicial modelo CatBoost. Elaboración propia.

5.- Decisión Tree

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	decision_tree	OR origin	Default	0.872203	0.999070	0.722772	0.744898	0.733668	0.740365
1	decision_tree	OR smote	Default	0.881413	0.997121	0.347222	0.765306	0.477707	0.616776
2	decision_tree	OR adasyn	Default	0.850915	0.997244	0.350254	0.704082	0.467797	0.585739
3	decision_tree	PT origin	Default	0.882468	0.999228	0.781250	0.765306	0.773196	0.768443
4	decision_tree	PT smote	Default	0.907099	0.997560	0.398010	0.816327	0.535117	0.674536
5	decision_tree	PT adasyn	Default	0.871367	0.997402	0.372449	0.744898	0.496599	0.620748

Gráfico 27: Tabla inicial modelo Decision Tree. Elaboración propia.

Presenta una gran variabilidad de resultados los mejores valores se obtienen en todas las distribuciones que no presentan corrección de sesgo y funciona mejor también para los datos originales, esto es, con desequilibrio de clases que con la corrección.

Si que hay que indicar que la clase FN se mantiene bastante constante en todas las distribuciones. Sin embargo en los casos que hay equilibrio de clases los FP se disparan al cuádruple

6.-Random Forest

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	random_forest	OR origin	Default	0.903017	0.999579	0.940476	0.806122	0.868132	0.829832 [[56859 5],[19 79]]
1	random_forest	OR smote	Default	0.913160	0.999491	0.870968	0.826531	0.848168	0.835052 [[56852 12],[17 81]]
2	random_forest	OR adasyn	Default	0.902965	0.999473	0.877778	0.806122	0.840426	0.819502 [[56853 11],[19 79]]
3	random_forest	PT origin	Default	0.913221	0.999614	0.941860	0.826531	0.880435	0.847280 [[56859 5],[17 81]]
4	random_forest	PT smote	Default	0.913151	0.999473	0.861702	0.826531	0.843750	0.833333 [[56851 13],[17 81]]
5	random_forest	PT adasyn	Default	0.902947	0.999438	0.858696	0.806122	0.831579	0.816116 [[56851 13],[19 79]]

Gráfico 28: Tabla inicial modelo Random Forest. Elaboración propia.

Proporciona resultados muy homogéneos y altos para todas las distribuciones, independientemente de la presencia o ausencia de sesgo y de equilibrio o no de clases. Todas están entre 0.81 y 0.84, con valores de ROC-AUC también elevados. La diferencia entre distribuciones radica en que en el caso de las distribuciones con equilibrio de clases, FP dobla el número de valores respecto a original.

7.- MLP

Este algoritmo presenta gran homogeneidad de resultados recuento de F2_Score.

Sin embargo las instancias de OR en el grupo FN son más elevadas.

Es el único en el que el F2-Score máximo se encuentra en la distribución OR

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	mlp	OR origin	Default	0.923241	0.999280	0.761468	0.846939	0.801932	0.828343 [[56838 26],[15 83]]
1	mlp	OR smote	Default	0.892637	0.999192	0.754902	0.785714	0.770000	0.779352 [[56839 25],[21 77]]
2	mlp	OR adasyn	Default	0.897520	0.998771	0.609375	0.795918	0.690265	0.750000 [[56814 50],[20 78]]
3	mlp	PT origin	Default	0.892760	0.999438	0.875000	0.785714	0.827957	0.802083 [[56853 11],[21 77]]
4	mlp	PT smote	Default	0.887553	0.999210	0.767677	0.775510	0.771574	0.773931 [[56841 23],[22 76]]
5	mlp	PT adasyn	Default	0.902894	0.999333	0.806122	0.806122	0.806122	0.806122 [[56845 19],[19 79]]

Gráfico 29: Tabla inicial modelo MLP. Elaboración propia

8.- KNN

El algoritmo solo presenta un límite superior para OR_origin de 0.807 e inferior de 0.7385 para PT_adasyn.

Los mejores resultados los presenta para OR_origin y PT_original, esto es, funciona mejor con la distribución original de clases desequilibrada.

Obtiene unos resultados aceptables en términos de F2-Score , para el resto de los casos, ya que caracteriza la clase FN pero se ve lastrada por FP con más de 100 valores.

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	knn	OR origin	Default	0.892787	0.999491	0.905882	0.785714	0.841530	0.807128	[[56856 8],[21 77]]
1	knn	OR smote	Default	0.937764	0.997770	0.427861	0.877551	0.575251	0.725126	[[56749 115],[12 86]]
2	knn	OR adasyn	Default	0.937764	0.997770	0.427861	0.877551	0.575251	0.725126	[[56749 115],[12 86]]
3	knn	PT origin	Default	0.887685	0.999473	0.904762	0.775510	0.835165	0.798319	[[56856 8],[22 76]]
4	knn	PT smote	Default	0.942919	0.997893	0.443878	0.887755	0.591837	0.739796	[[56755 109],[11 87]]
5	knn	PT adasyn	Default	0.942910	0.997876	0.441624	0.887755	0.589831	0.738540	[[56754 110],[11 87]]

Gráfico 30: Tabla inicial modelo KNN. Elaboración propia.

En todos los algoritmos a excepción de MLP, el efecto de la transformación PT tiene un efecto positivo o neutro, por lo que a fin de simplificar el análisis a realizar, será la única que se emplee. A continuación se muestran un primer ranking, en el que valorar el potencial de cada uno de ellos.

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	xgboost	OR smote	Default	0.943579	0.999210	0.719008	0.887755	0.794521	0.847953	[[56830 34],[11 87]]
1	random_forest	PT origin	Default	0.913221	0.999614	0.941860	0.826531	0.880435	0.847280	[[56859 5],[17 81]]
2	catboost	OR origin	Default	0.903017	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859 5],[19 79]]
3	catboost	PT origin	Default	0.903017	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859 5],[19 79]]
4	mlp	OR origin	Default	0.923241	0.999280	0.761468	0.846939	0.801932	0.828343	[[56838 26],[15 83]]
5	knn	OR origin	Default	0.892787	0.999491	0.905882	0.785714	0.841530	0.807128	[[56856 8],[21 77]]
6	decision_tree	PT origin	Default	0.882468	0.999228	0.781250	0.765306	0.773196	0.768443	[[56843 21],[23 75]]
7	adaboost	OR origin	Default	0.877349	0.999175	0.762887	0.755102	0.758974	0.756646	[[56841 23],[24 74]]
8	regression_logistic	PT origin	Default	0.836603	0.999175	0.814815	0.673469	0.737430	0.697674	[[56849 15],[32 66]]

Gráfico 31: Tabla inicial total modelos con parámetros default. Elaboración propia.

Análisis del Ranking Inicial

El ranking de los modelos presenta un claro corte en dos grupos diferenciados, basado en su capacidad para equilibrar la clasificación correcta de fraudes (minimizar FN, falsos negativos) y otras métricas clave.

1. Algoritmos con F2 Score > 0.82 y FN < 20

Este grupo representa los modelos con mayor potencial de detección efectiva de fraudes, priorizando el recall (detección de fraudes) sobre la precisión. En la problemática del fraude, reducir los falsos negativos (FN) es crucial. Aquí destacan los modelos basados en árboles de decisión y boosting, que son conocidos por manejar bien problemas con desbalance de clases.

a) XGBoost (OR_smote)

- Mejor modelo en el ranking.
- Logra el mayor F2 Score 0.847953, con una tasa de falsos negativos FN muy baja 11, lo que lo hace el modelo más eficiente en detección de fraudes.

- El uso de SMOTE para equilibrar las clases ha sido clave para mejorar su capacidad predictiva. El ROC-AUC de 0.9435 indica una excelente capacidad de discriminación muy superior al desempeño de los mejores algoritmos del ranking.

b) Random Forest (PT_origin)

- Tiene un F2 Score de 0.847280. Su capacidad para detectar fraudes es sólida, aunque tiene un pequeño incremento en FN con 17.

c) CatBoost (OR_origin y PT_origin)

- Obtiene el mismo F2 Score de 0.829832 y solo 19 falsos negativos es robusto y eficiente independientemente de la presencia o no de sesgo.

d) MLP (OR_origin)

- Obtiene un buen rendimiento con un F2 Score de 0.828343, y mantiene FN en 15.

Este grupo es liderado claramente por los modelos de boosting y árboles de decisión. La capacidad de estos algoritmos para manejar problemas de desequilibrio de clases y su habilidad inherente para capturar relaciones complejas entre las variables los convierte en la mejor opción para este problema de fraude con tarjetas.

2. Algoritmos con F2 Score < 0.8 y FN >= 20

- a) KNN (OR_origin): Aunque muestra una precisión alta, 0.9058 y un buen F2 Score 0.8415, su recall de 0.7857 y 21 FN lo hacen menos efectivo frente a otros modelos más robustos.
- b) Decision Tree (PT_origin): La reducción del sesgo mejora ligeramente su rendimiento, pero su F2 Score, 0.768443 y 23 FN lo colocan por debajo de modelos más avanzados como XGBoost.
- c) AdaBoost (OR_origin): Con 24 FN y un F2 Score de 0.756646, no alcanza el rendimiento de otros métodos de boosting.
- d) Regresión Logística (PT_origin): Este modelo tiene dificultades en un entorno no lineal y desequilibrado, con el F2 Score más bajo (0.697674) y 32 FN, lo que reduce su capacidad para detectar fraudes.

Conforme lo expuesto ya tenemos unas conclusiones iniciales:

- Técnicas de Boosting y Árboles de Decisión: Los modelos de boosting, como XGBoost y CatBoost, son los más adecuados para este problema, independientemente de la presencia de sesgo o del equilibrio de clases. Su capacidad para manejar desequilibrios y la complejidad de las relaciones entre variables los hace superiores.

- SMOTE se ha demostrado eficaz para mejorar el rendimiento de los modelos, es el caso de XGBoost (OR_smote), que encabeza el ranking.
- La reducción del sesgo en las variables mediante PT ha mostrado mejoras notables en algunos modelos, como Random Forest y Decision Tree.
- ADASYN no ha aportado ninguna solución final al ranking inicial. No supone mejora global.

5. Algoritmos

5.1. Algoritmos Lineales: Regresión Logística

Este modelo se ha entrenado haciendo uso de la función:

[sklearn.linear_model.LogisticRegression](#) [8]

El parámetro C en la regresión logística controla la fuerza de la regularización en el modelo, razón por la que se empleara como parámetro principal.

Un valor más bajo de C indica una regularización más fuerte, mientras que un valor más alto de C indica una regularización más débil. La regularización es una técnica que ayuda a prevenir el sobreajuste (overfitting) en los modelos de aprendizaje automático, evitando que se ajusten demasiado a los datos de entrenamiento.

En la regresión logística, la regularización se logra mediante un término de penalización añadido a la función de coste. Un valor bajo de C impone una mayor penalización, lo que puede llevar a un modelo más simplificado y fronteras de decisión más restrictivas. Por otro lado, un valor alto de C disminuye la penalización, permitiendo al modelo ajustarse más a los datos de entrenamiento.

En caso de underfitting, se puede aumentar C para reducir la regularización y permitir un mejor ajuste a los datos. Para combatir el overfitting, se puede disminuir C para aumentar la regularización y simplificar el modelo.

Los valores de los parámetros principales de todos los algoritmos que se analizaran se han obtenido por aproximación vía ensayo y error, en pruebas externas, previas a la redacción de este documento.

Por último, se presenta el grid de gráficos en el que se va a sustentar el análisis de cada algoritmo. Está organizado en dos filas y tres columnas, donde cada gráfico representa el rendimiento del modelo en términos de F2 Score frente a diferentes valores del parámetro, en este caso, C hiperparámetro de regularización de la regresión logística.

1. Organización del Grid:

- Filas:
 - La primera fila contiene resultados para las distribuciones con sesgo, etiquetadas como OR_, la distribución original.

- La segunda fila corresponde a los resultados de las distribuciones con reducción de sesgo, denominadas como PT_ , aplicación de la Power Transformation para corregir sesgos en las variables.
- Columnas:
 - La primera columna muestra los resultados de las distribuciones _origin, es decir, los datos originales con desequilibrio de clases entre transacciones legítimas y fraudulentas.
 - La segunda columna corresponde a las distribuciones con equilibrio de clases utilizando la técnica _smote (Synthetic Minority Over-sampling Technique).
 - La tercera columna presenta los resultados para las distribuciones equilibradas mediante _adasyn (Adaptive Synthetic Sampling).

2. Descripción del Gráfico Individual

Cada gráfico sigue la misma estructura, y está diseñado para permitir la comparación del F2 Score del modelo en distintos escenarios.

Ejes:

- El eje X representa el valor del parámetro principal del algoritmo, C en este caso, que regula la penalización en el modelo de regresión logística. Los valores del parámetro comprenden un intervalo estudiado previamente, y que proporciona los mejores valores en términos de F2 Score del parámetro. En este caso, para C, van desde 0.001 hasta 3, mostrando cómo diferentes niveles de regularización afectan el rendimiento del modelo.
- El eje Y muestra el F2 Score, con valores entre 0 y 1, donde 1 es el rendimiento perfecto.

Líneas de Referencia:

- La línea azul claro con marcadores circulares representa el rendimiento del modelo evaluado en esa configuración particular de distribución de datos y ajuste de clases.
- La línea horizontal azul oscuro representa el F2 Score máximo obtenido por el mejor algoritmo analizado en el paso previo, Default. Esta línea sirve como un benchmark global que permite comparar el rendimiento del modelo actual con el mejor modelo previamente evaluado.
- La línea horizontal roja muestra el F2 Score del modelo en su versión Default ,con parámetros no optimizados, dentro de la misma distribución de datos. Esto permite observar la influencia del ajuste del hiperparámetro C en el rendimiento del modelo.

Área de Mejora (Intervalo):

- El área sombreada en azul claro representa los valores de C para los cuales el modelo logra un F2 Score superior al obtenido en su configuración por defecto. Es decir, este intervalo muestra los rangos donde el modelo se mejora al ajustar C frente a su versión inicial, por defecto.

Título: Comprende 3 líneas.

- En la línea superior se muestra la distribución empleada.
- En la línea media se muestra el intervalo de parámetros, si es que existe, que mejora el F2 Score del modelo original Default.
- La línea inferior recoge el valor máximo de F2 Score de la distribución, y el valor de parámetro con el que se alcanza.

El rango de parámetros que se empleara para este algoritmo es:

C_values = [0.1,0.5,1,1.5,2,2.5,3]

Análisis del Modelo de Regresión Logística

La PT en las distribuciones de datos muestra un impacto mínimo, variando según el balance de clases.

- Distribución Original con Sesgo: PT mejora ligeramente el rendimiento, sugiriendo que optimiza el modelo pese al sesgo.
- Distribución SMOTE: PT disminuye el rendimiento, introduciendo posiblemente más ruido que información útil.
- Distribución ADASYN: idéntico caso a smote.

Análisis del Desequilibrio de Clases

- Distribución Original: El F2 Score 0.68-0.69 muestra un rendimiento aceptable, con convergencia hacia el comportamiento estándar del modelo.
- Distribución SMOTE: El F2 Score cae a 0.21-0.22 debido a un aumento de falsos positivos.
- Distribución ADASYN: Los resultados son aún peores, con un F2 Score de 0.08, también afectado por falsos positivos.

Mejor Solución del Algoritmo

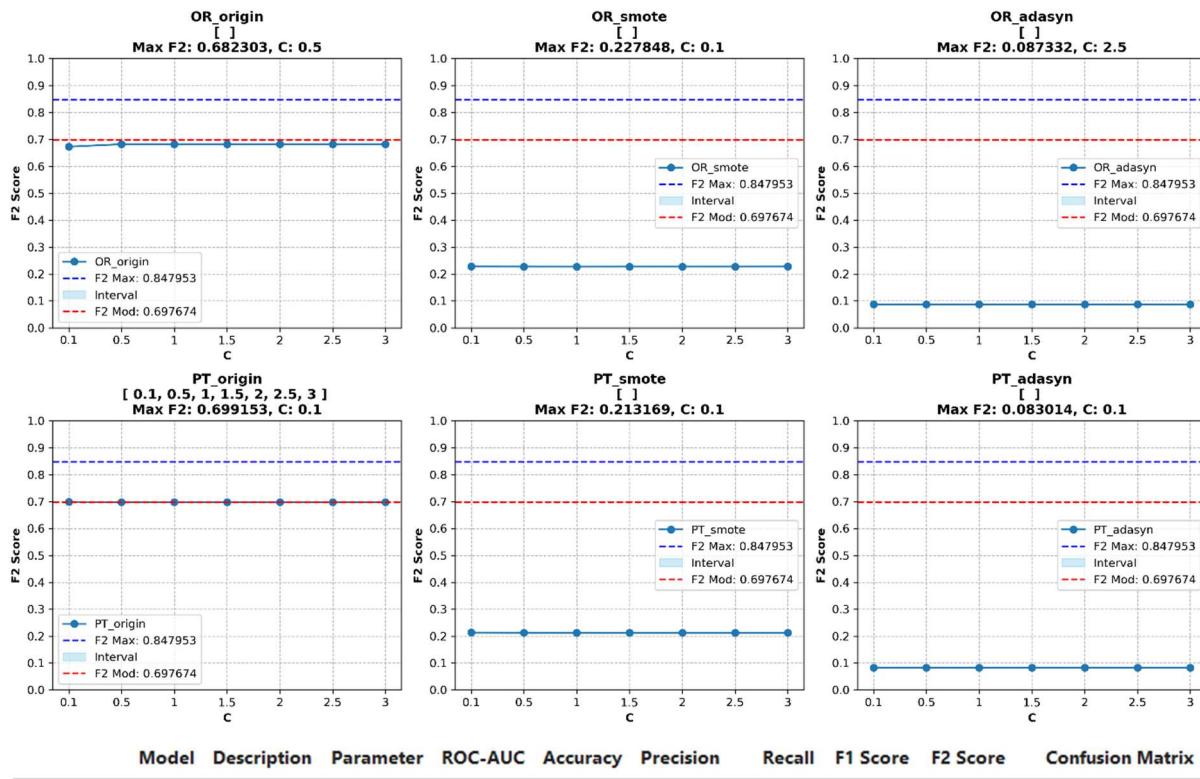
La mejor configuración es PT con la distribución original. Con un C=0.1, el F2 Score es de 0.6991, aunque inferior a otros algoritmos en su versión estándar.

Conclusiones

La PT tiene un impacto mínimo y es más efectiva con datos sesgados. SMOTE y ADASYN reducen el rendimiento significativamente debido a los falsos positivos.

PT_Origin F2 Score de 0.6991, mejora ligeramente el modelo, pero no es una solución óptima para aplicaciones prácticas.

regression_logistic_total: F2 Score Results



	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	regression_logistic	PT origin	C=0.1	0.836612	0.999192	0.825	0.673469	0.741573	0.699153	[[56850 14], [32 66]]

Gráfico 32: Modelo Regresión Lineal evolución F2-Score frente al parámetro principal.
Elaboración propia.

5.2. Algoritmos de Boosting

5.2.1. AdaBoost

El modelo implementado es un clasificador AdaBoost utilizando la función `sklearn.ensemble.AdaBoostClassifier`.^[9]

El parámetro principal seleccionado para entrenar el algoritmo es "n_estimators", que determina la cantidad de clasificadores débiles utilizados en el ensamble. El valor por defecto es 50. Aumentar el número de estimadores puede hacer que el modelo sea más complejo y capaz de adaptarse mejor a los datos de entrenamiento, pero también puede aumentar el tiempo de entrenamiento y el riesgo de sobreajuste.

Además del número de estimadores, otros parámetros importantes en AdaBoost son el clasificador débil base (`base_estimator`), la tasa de aprendizaje (`learning_rate`) y el algoritmo utilizado (`algorithm`). Estos parámetros también tienen un impacto significativo en el rendimiento del modelo y pueden requerir ajuste dependiendo del problema específico. Parámetro `n_estimators` = [10, 30, 50, 70, 100]

Tratamiento del Sesgo y Análisis del F2 Score

- Los resultados del F2 Score en Adaboost son consistentes, variando entre 0.79 y 0.86 en todas las configuraciones. Esto refleja un comportamiento estable, en todos los escenarios de sesgo. Un patrón recurrente es que el F2 Score disminuye conforme aumenta el parámetro n_estimators. Es decir, aunque el rendimiento inicial es bueno, a medida que se incrementan los estimadores más allá de 30, el F2 Score comienza a empeorar.
- Los mejores resultados se obtienen en el rango de 10 a 30 estimadores, donde el F2 Score supera al mejor modelo por defecto. Sin embargo, pruebas adicionales confirman que aumentar el número de estimadores no es una estrategia eficaz para mejorar el rendimiento, a diferencia de lo que suele ocurrir en otros contextos.

Tratamiento de Sesgos (OR vs PT)

- PT en Origin:

El rendimiento se mantiene constante en cualquier escenario de sesgo, esto es PT para este algoritmo no aporta mejora sustancial, salvo n_estimators 10.

- PT en SMOTE:

Obtiene un F2 Score alto todas las distribuciones con 0.86 es una combinación efectiva, únicamente por detrás de OR_smote.

- PT en ADASYN: El equilibrado mejora el F2 Score 0.84 sobre original.

Esto sugiere que PT, en general, tiende a mejorar el rendimiento en configuraciones equilibradas.

Análisis del Desequilibrio/Equilibrio de Clases (OR vs SMOTE o ADASYN)

El equilibrio de clases mediante técnicas de resampling como SMOTE y ADASYN tiene un impacto positivo en el rendimiento de Adaboost. Mientras que la distribución original muestra resultados aceptables, las configuraciones con resampling logran mejoras notables.

- SMOTE: Se obtiene un F2 Score máximo de 0.86, siendo más efectivo en el rango de 10-30 estimadores. A partir de ahí, el rendimiento empieza a disminuir, lo que sugiere que aumentar los estimadores más allá de este punto añade más ruido que valor.
 - ADASYN: Existe mejora el rendimiento del modelo con un F2 Score de 0.84 en su mejor configuración, siendo más efectivo en el rango de 10-30 estimadores
- El equilibrado de clases mejora el rendimiento en un rango limitado del parámetro, entre 10-30 estimadores.

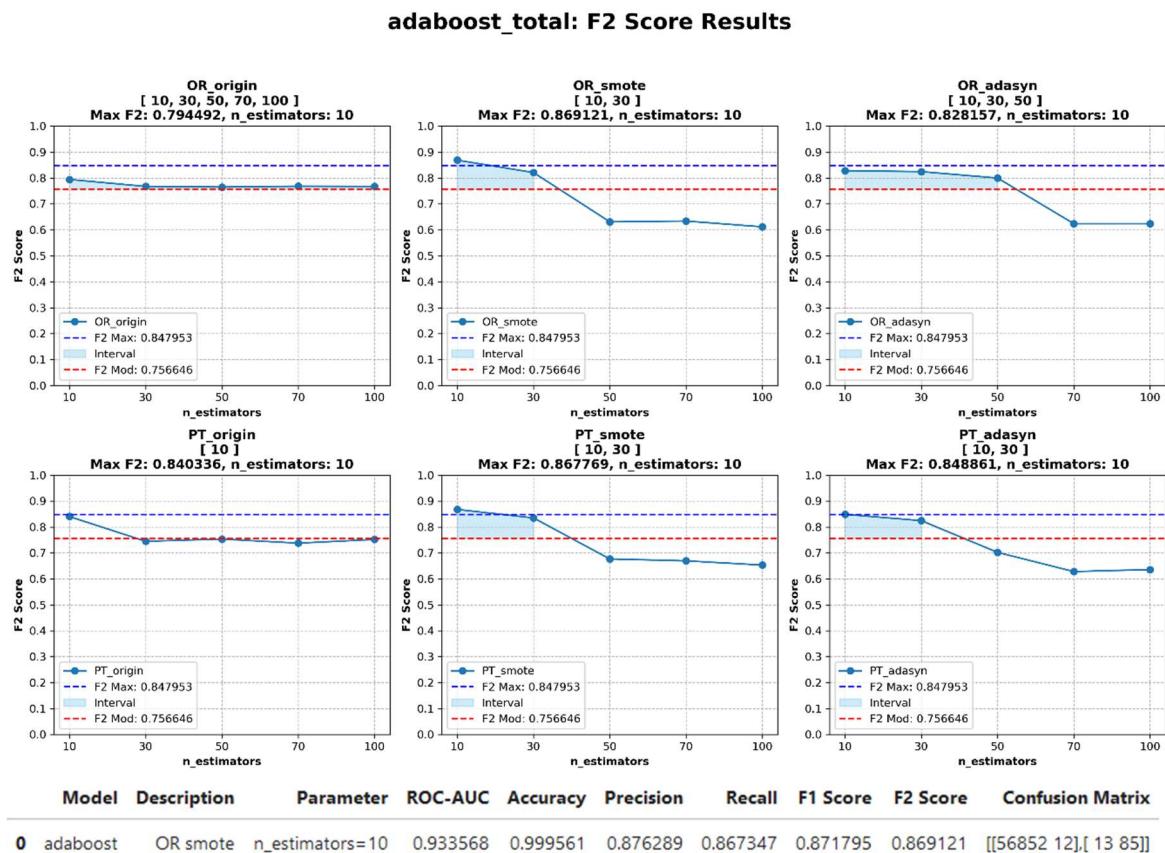


Gráfico 33: Modelo AdaBoost evolución F2-Score frente al parámetro principal.
Elaboración propia.

Mejor Solución del Algoritmo

Se obtiene con OR_SMOTE y n_estimators=10, logrando mejorar el mejor registro de F2 Score en comparación con el mejor modelo default.

Sin embargo, es importante destacar que los mejores resultados se logran con un número reducido de estimadores, y la estrategia tradicional de aumentar estimadores, para aumentar el F2 Score, no es efectiva aquí, ya que el rendimiento cae al incrementarlos más allá de 30, esto implica un potencial de mejora limitado.

Conclusión

El algoritmo Adaboost es sensible al número de estimadores, con un rendimiento óptimo entre 10 y 30. Las técnicas de equilibrio de clases como SMOTE y ADASYN, combinadas con PT, resultan ser muy eficaces en este modelo. Sin embargo, el ajuste de estimadores no presenta un gran potencial de mejora, ya que más allá de 30, el rendimiento decae. Por tanto, la mejor configuración identificada, OR_SMOTE con n_estimators=10, es la opción recomendada para la optimización de hiperparámetros en fases futuras. Sin embargo, será necesario explorar otros aspectos del modelo para maximizar su rendimiento en etapas avanzadas.

5.2.2. XGBoost

La función empleada para entrenar este algoritmo es: XGBClassifier [10]

El parámetro más importante en XGBoost es el "learning_rate" (tasa de aprendizaje). Controla la contribución de cada árbol al modelo final y afecta la velocidad de convergencia durante el entrenamiento.

Un learning_rate bajo permite un ajuste más lento pero posiblemente más preciso, mientras que un learning_rate alto acelera el proceso de ajuste pero puede llevar a un sobreajuste. Es crucial para el equilibrio entre el rendimiento y la capacidad de generalización del modelo en XGBoost

El learning_rate debe ajustarse junto con otros parámetros clave, como el número de árboles y la profundidad máxima, para encontrar el equilibrio óptimo en el rendimiento del modelo. El algoritmo XgBoost es el que obtiene el segundo mejor valor de F2-Score con 0.8350. El rango del parámetro principal learning_rate = [0.001,0.01,0.1,0.5,1,3].

XGBoost muestra un rendimiento diferenciado según se utilice o no una estrategia de equilibrio de clases. En general, todas las configuraciones mejoran con respecto al modelo por defecto, e incluso en algunos casos superan al mejor algoritmo en su fase por defecto.

Tratamiento del Sesgo (OR vs PT)

La aplicación de Power Transform (PT) no mejora el rendimiento de XGBoost, e incluso lo empeora en la mayoría de los casos. El modelo gestiona bien los datos sin PT, lo que sugiere que la reducción de sesgo no es necesaria.

- Al incrementar el valor de learning_rate por encima de 1, el F2 Score se deteriora en todas las distribuciones, tanto con sesgo (OR) como con sesgo reducido (PT). A partir de ese valor, el rendimiento cae por debajo del modelo por defecto.
- En todas las distribuciones, la configuración OR (sin PT) supera a PT, confirmando que la transformación no aporta mejoras en este modelo.

Tratamiento del Desequilibrio de Clases (OR vs SMOTE o ADASYN)

El comportamiento de XGBoost varía significativamente según se trate o no el desequilibrio de clases.

- Distribuciones sin equilibrio de clases (OR): El rendimiento es más lineal hasta que el learning_rate alcanza 1, tras lo cual el F2 Score cae abruptamente. Aunque supera al modelo por defecto, no llega a ser mejor que el mejor algoritmo en fase por defecto.
- Distribuciones con equilibrio de clases (SMOTE o ADASYN): Aquí, el rendimiento sigue una curva parabólica inversa. El F2 Score alcanza su máximo entre learning_rate 0.5 y 1, especialmente con SMOTE, mejorando incluso al mejor algoritmo por defecto. ADASYN sigue un patrón similar, pero sin superar al mejor resultado en la fase default. El rango útil en estas configuraciones es más estrecho, concentrándose en learning_rate alrededor de 0.5.

Análisis de la Mejor Solución del Algoritmo

La mejor configuración se obtiene con OR_SMOTE y learning_rate = 0.5, logrando un F2 Score de 0.8562. Este resultado supera al mejor algoritmo por defecto, siendo el rendimiento más alto en todas las pruebas de XGBoost. Sin embargo, el comportamiento del modelo sigue una parábola inversa, indicando que se ha alcanzado un punto de inflexión, lo que limita el potencial de mejora mediante ajustes adicionales de hiperparámetros. Aunque puede avanzarse a la siguiente fase de optimización, las posibilidades de mejorar significativamente este resultado son reducidas.

xgboost_total: F2 Score Results

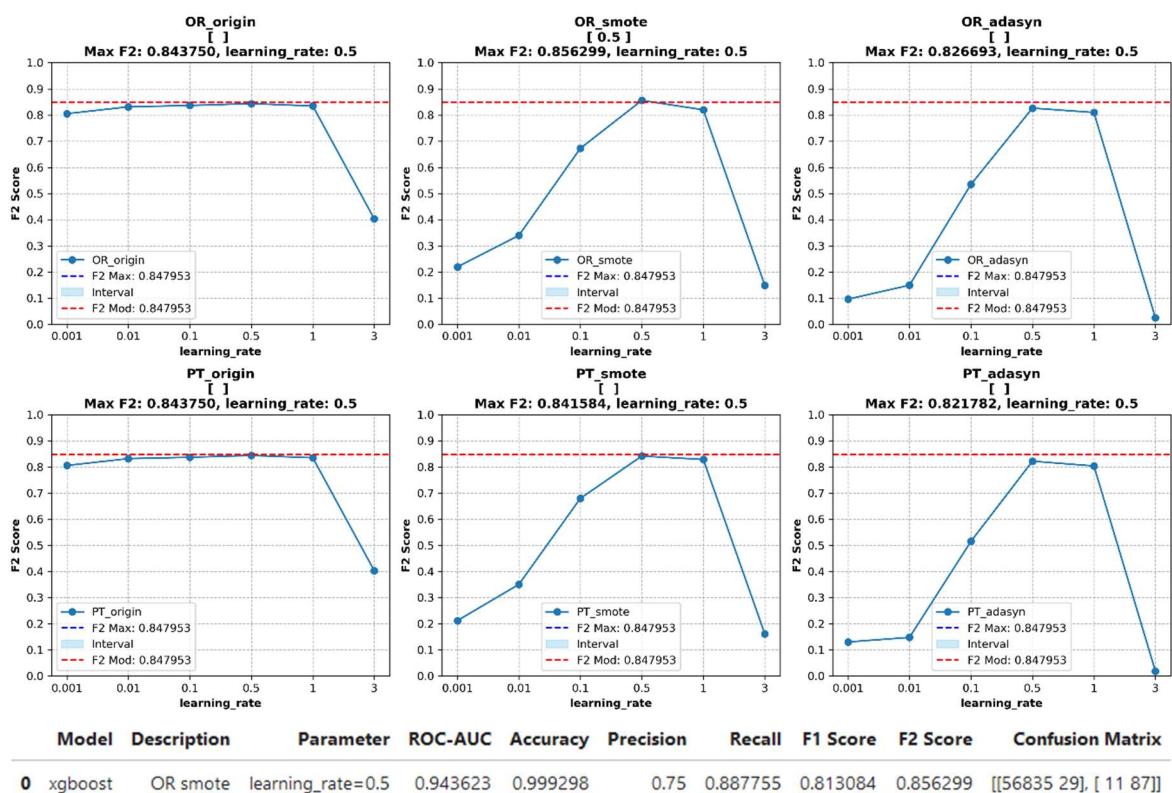


Gráfico 34: Tabla inicial modelo XgBoost con parámetro principal learning_rate.
Elaboración propia

5.2.3. Catboost

El parámetro principal en CatBoost^[11] es "n_estimators", determina el número de árboles construidos durante el entrenamiento, lo que influye en la complejidad y capacidad de generalización del modelo.

Más estimadores capturan patrones complejos y aumentan la precisión, pero también pueden llevar a sobreajuste en conjuntos de datos pequeños. Cada árbol construido en el modelo busca corregir errores y aprender detalles específicos de los datos de entrenamiento. Sin embargo, un número excesivo de árboles aumenta el costo computacional y el tiempo de entrenamiento, y puede dificultar la generalización a nuevos datos. Encontrar el valor óptimo para "n_estimators" implica equilibrar rendimiento, generalización y eficiencia.

Descripción General del Rendimiento

El algoritmo CatBoost presenta un rendimiento predecible y lineal en términos de F2 Score al aumentar el número de estimadores (n_estimators). Su valor por defecto de 100 ya ofrece un rendimiento competitivo, manteniéndose por encima de 0.8 en la distribución original desbalanceada. En contraste, con técnicas de sobre muestreo como SMOTE o ADASYN, el F2 Score no alcanza estos niveles, lo que sugiere que CatBoost gestiona mejor los datos sin estas técnicas.

El rendimiento en su configuración por defecto está a la par con el de los mejores algoritmos, lo que indica que es un modelo robusto sin necesidad de ajustes intensivos.

Tratamiento del Sesgo (OR vs PT)

El tratamiento de sesgo mediante Power Transform (PT) no tiene un impacto significativo en el rendimiento. Las distribuciones originales (OR) tienden a superar las versiones con PT y otros algoritmos competidores. Las mejoras se concentran en el rango de estimadores entre 100 y 250, pero fuera de este tramo, el rendimiento cae por debajo del algoritmo por defecto.

Conclusión sobre el sesgo

El tratamiento de sesgo no aporta una ventaja clara. Las configuraciones originales ofrecen mejores resultados de forma consistente.

Tratamiento del Equilibrio de Clases (OR vs SMOTE o ADASYN)

CatBoost maneja eficazmente los datos desbalanceados sin necesidad de aplicar técnicas de resampling como SMOTE o ADASYN. Los resultados de las distribuciones originales superan las versiones con equilibrio de clases, lo que sugiere que estas técnicas no mejoran significativamente el rendimiento.

Las técnicas de sobre muestreo no aportan mejoras significativas y pueden ser descartadas en este contexto.

Análisis de la Mejor Solución del Algoritmo

La mejor configuración se encuentra con n_estimators= 150, tanto en OR origin como en PT origin. Ambas opciones superan la configuración por defecto y el mejor algoritmo comparado. Esta configuración maximiza el rendimiento sin necesidad de equilibrar las clases o aplicar transformaciones de sesgo.

La mejor solución se obtiene con la configuración original, sin sobre-muestreo.

Conclusiones Generales

CatBoost destaca por su rendimiento robusto en su configuración por defecto y no necesita ajustes de equilibrio de clases. Sus principales ventajas son:

1. Comportamiento predecible: Ajustes en los estimadores no producen cambios drásticos, lo que lo convierte en un modelo fiable.
2. Distribución original superior: Las versiones con y sin tratamiento de sesgo y sin sobre muestreo ofrecen los mejores resultados.
3. SMOTE y ADASYN no recomendados: Las técnicas de equilibrio de clases no mejoran el rendimiento de CatBoost en este caso.

En resumen, la mejor estrategia es mantener la configuración por defecto o utilizar n_estimators= 200 con la distribución original, obteniendo así el mejor rendimiento posible.

catboost_total: F2 Score Results

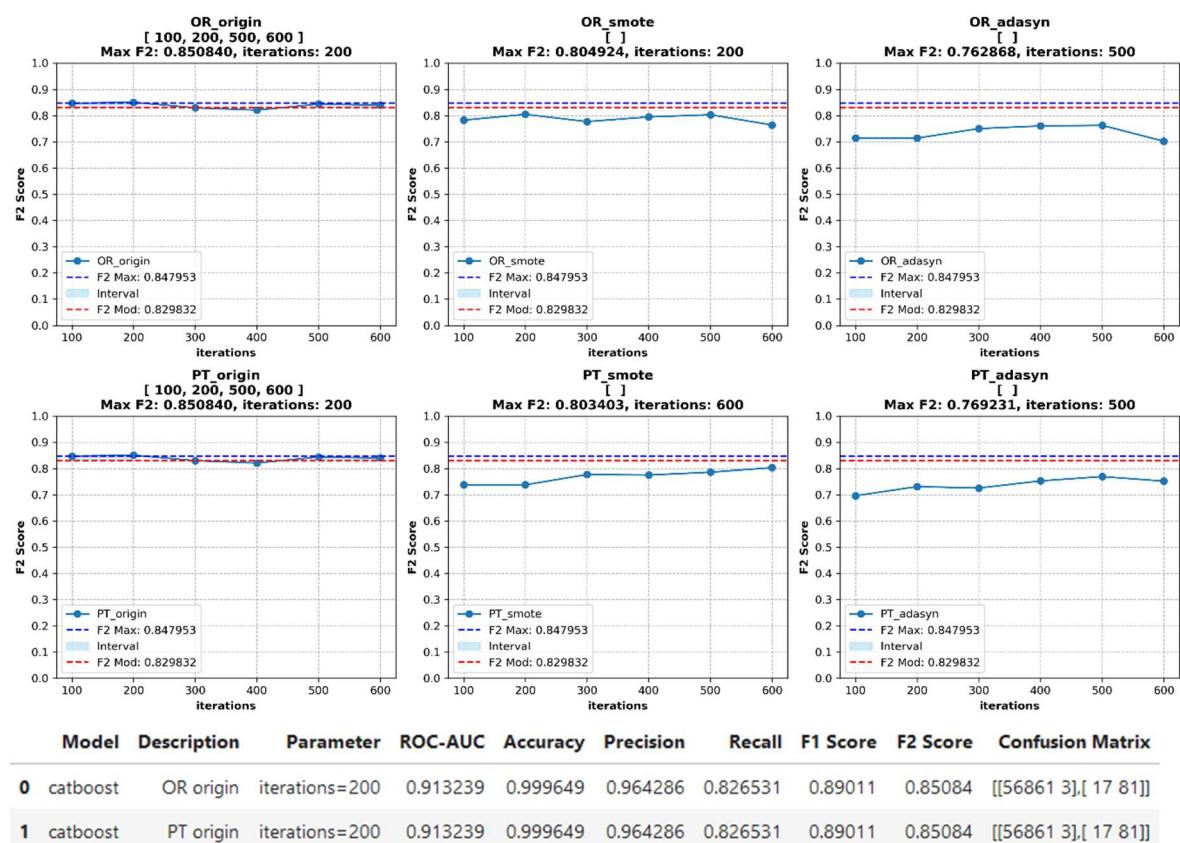


Gráfico 35: Modelo CatBoost evolución F2-Score frente parámetro principal. Elaboración propia.

5.3. Algoritmos basados en arboles de decisión

5.3.1. Decisión Tree

Este algoritmo se ha entrenado empleando la función [sklearn.treeDecisionTreeClassifier](#).^[12]

El parámetro "max_depth" en un árbol de decisión establece la profundidad máxima permitida para el árbol. Un valor más alto indica un árbol más profundo, capaz de capturar relaciones complejas y detalladas en los datos. Sin embargo, un aumento excesivo en la profundidad puede llevar a sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos. Por otro lado, una profundidad demasiado baja puede resultar en un modelo subajustado que no captura relaciones complejas. Encontrar el valor óptimo para max_depth implica un equilibrio entre complejidad y generalización.

El intervalo que se va a emplear es max_depth_values = [1,2,3,4,5]. El valor F2-Score obtenido para este algoritmo con la configuración por defecto era 0.7777

A continuación se muestran los valores de la distribución. Se presenta la misma circunstancia, las soluciones con sobre muestreo presentan recuentos de F2-Score extremadamente bajos por mala clasificación de la clase FP.

Tratamiento del Sesgo (OR vs PT)

La aplicación de Power Transform (PT) a las distribuciones tiene un impacto mixto en el rendimiento del modelo, dependiendo de la configuración de clases:

- Distribución Original (Sesgo): La aplicación de PT muestra una ligera mejora en el rendimiento, indicando que la transformación puede ayudar a mitigar el sesgo en los datos originales.
- Distribución SMOTE: Al combinar PT con SMOTE, el F2 Score disminuye, lo que sugiere que el sobre muestreo introduce ruido adicional en lugar de mejorar el modelo.
- Distribución ADASYN: En este caso, PT mejora levemente el rendimiento, lo que indica que ADASYN se beneficia más de la transformación en comparación con SMOTE.

Análisis del Desequilibrio / Equilibrio de Clases

Los resultados varían significativamente según la distribución de clases:

- Distribución Original (Sesgo): El F2 Score se mantiene en el 0.80, superando apenas el valor por defecto del modelo, pero a un alto costo computacional, especialmente al aumentar el parámetro max_depth. Este rendimiento sugiere que, aunque aceptable, el modelo tiene un margen de mejora limitado.
- Distribuciones Equilibradas:
 - SMOTE: El F2 Score cae drásticamente a un rango de 0.26-0.27, debido a un aumento en los falsos positivos, lo que indica que SMOTE introduce más ruido que valor.
 - ADASYN: Los resultados son aún peores, con un desplome a 0.06-0.10, lo que lo convierte en un método ineficaz, similar a lo observado en otros algoritmos como la regresión logística.

Análisis de la Mejor Solución del Algoritmo

La mejor configuración del modelo se obtiene al aplicar PT en la distribución original, alcanzando un F2 Score de 0.8071. Si bien mejora ligeramente el rendimiento por defecto, está lejos del mejor algoritmo probado, y a un coste computacional considerable.

Conclusiones

El uso de Power Transform (PT) no mejora el rendimiento, siendo más efectivo en la distribución original sesgada. Técnicas de sobre muestreo como SMOTE y ADASYN ofrecen resultados muy bajos debido al aumento de falsos positivos.

La mejor configuración, OR_Origin con un F2 Score de 0.8071, supera levemente la versión por defecto, pero no es suficiente para considerarla una mejora significativa, por lo que este algoritmo no tiene el suficiente potencial de mejora.

decision_tree_total: F2 Score Results

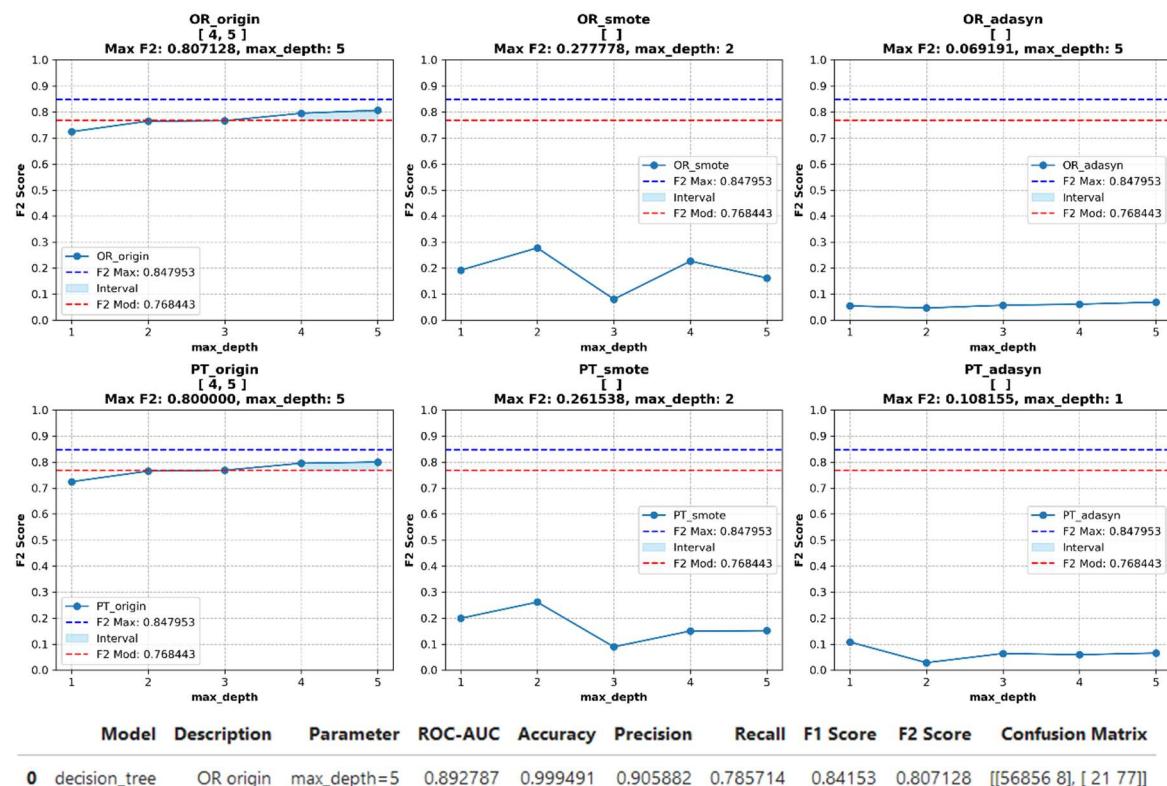


Gráfico 36: Modelo decision_tree F2-Score frente al parámetro principal max_depth.
Elaboración propia

5.3.2. Random Forest

En el algoritmo Random Forest de Scikit-learn^[13], el parámetro `n_estimators` se utiliza para especificar el número de árboles que se utilizarán en el bosque.

El parámetro `n_estimators` tiene un impacto significativo en el rendimiento y la capacidad de generalización del modelo. Un mayor número de árboles tiende a mejorar la precisión del modelo, reducir la varianza y hacer que las predicciones sean más estables. Sin embargo, agregar más árboles también aumenta el costo computacional del entrenamiento y la inferencia del modelo. Existen varias estrategias para seleccionar el valor apropiado de `n_estimators`, a saber, experiencias anteriores similares, búsquedas aleatorias o gridsearch y por último las curvas de validación.

random_forest_total: F2 Score Results

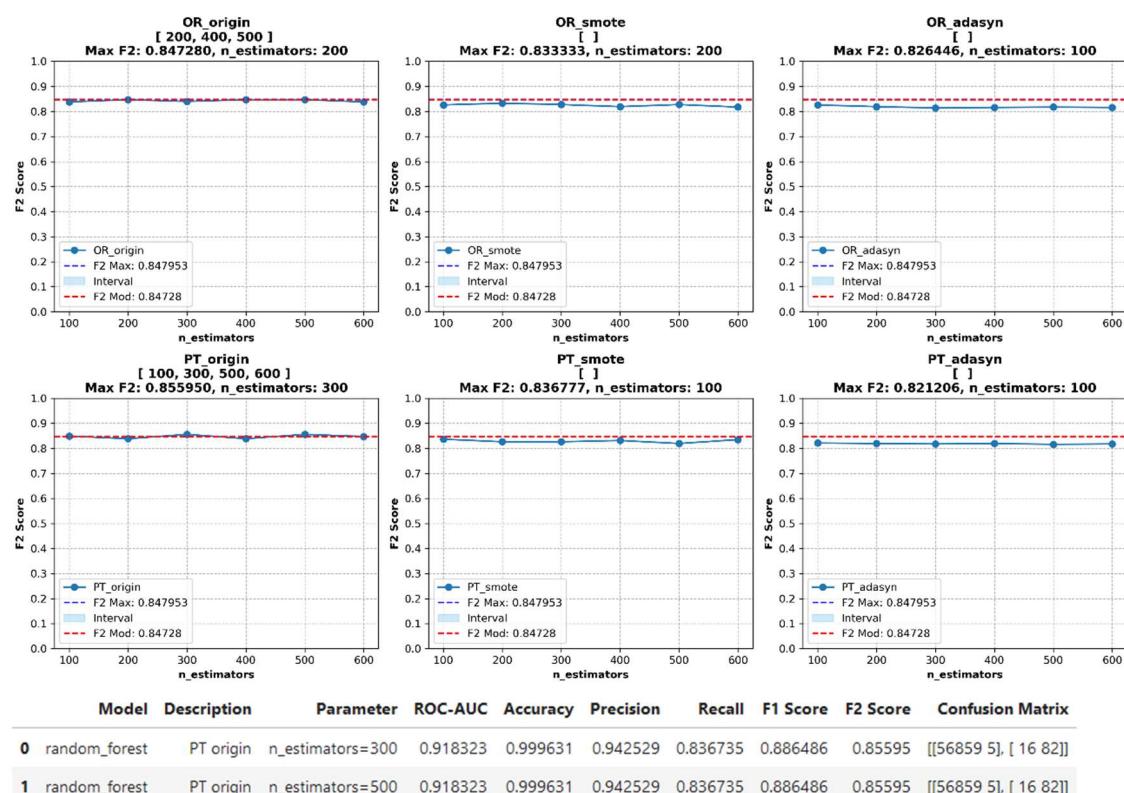


Gráfico 37: Tabla inicial modelo Random Forest parámetro principal `n_estimators`. Elaboración propia.

El análisis del algoritmo refleja una tendencia lineal en su rendimiento, con un F2 Score que oscila entre 0.81 y 0.85 en todas las configuraciones probadas, ya sea con o sin corrección de sesgo o equilibrio de clases. Este comportamiento estable y predecible indica que, aunque el rendimiento es alto, el algoritmo presenta poco margen de mejora.

Rendimiento y Estabilidad:

El algoritmo destaca por su estabilidad en todas las configuraciones, con un rendimiento casi constante. Sin embargo, esta falta de variabilidad limita su potencial de optimización, en comparación con otros modelos que muestran un mayor rango de mejora.

Rendimiento en Configuración Default:

En su versión por defecto, el algoritmo ya alcanza un F2 Score competitivo, superior a otros modelos desde el inicio. La mejor configuración, PT_origin con $n_estimators = 300$ y 500 llega a 0.8559, similar al rendimiento por defecto, pero sin una mejora significativa.

Coste Computacional:

El alto rendimiento del modelo viene acompañado de un coste computacional elevado, lo que lo hace menos eficiente en comparación con alternativas que logran un balance más favorable entre desempeño y recursos.

Recomendación:

El uso de este algoritmo puede no ser justificado debido a su limitado retorno tras la optimización de parámetros y su elevado coste computacional.

5.4. Algoritmos de redes neuronales: MLP

Este modelo se ha entrenado haciendo uso de la función:

[sklearn.neural_network.MLPClassifier](#)^[14].

El parámetro principal en un Perceptrón Multicapa (MLP) es el `hidden_layer_sizes`, que especifica la arquitectura de capas ocultas de la red neuronal. Este parámetro determina el número de capas ocultas y el número de neuronas en cada capa oculta.

La arquitectura del MLP tiene un impacto significativo en su capacidad para aprender y modelar relaciones complejas en los datos. Al ajustar el parámetro `hidden_layer_sizes`, puedes controlar la capacidad del modelo para capturar representaciones más sofisticadas y complejas.

Número de capas ocultas: Elige la cantidad de capas ocultas que deseas incluir en el MLP. Cada capa oculta proporciona una representación intermedia de los datos y permite que el modelo aprenda características más abstractas y de mayor nivel. Un mayor número de capas ocultas puede aumentar la capacidad del modelo para capturar relaciones no lineales complejas en los datos. Sin embargo, un número excesivo de capas puede llevar al sobreajuste.

Número de neuronas en cada capa oculta: Determina la cantidad de neuronas en cada capa oculta. Un mayor número de neuronas permite una mayor capacidad de

representación, ya que cada neurona aprende características específicas y distintas de los datos de entrada. Sin embargo, un exceso de neuronas puede llevar al sobreajuste, especialmente si la cantidad de datos de entrenamiento es limitada. Es importante encontrar un equilibrio entre el número de neuronas y la cantidad de datos disponibles.

Combinación de capas: La elección de la arquitectura adecuada implica encontrar la combinación óptima de capas ocultas y neuronas en cada capa. Algunas arquitecturas comunes incluyen una única capa oculta, múltiples capas ocultas con la misma cantidad de neuronas o arquitecturas más complejas con diferentes tamaños de capas ocultas. No hay una regla estricta sobre qué arquitectura es la mejor, ya que depende del problema y los datos específicos.

El valor de hidden_layer_sizes = [(50,), (100,), (120,), (150,)]

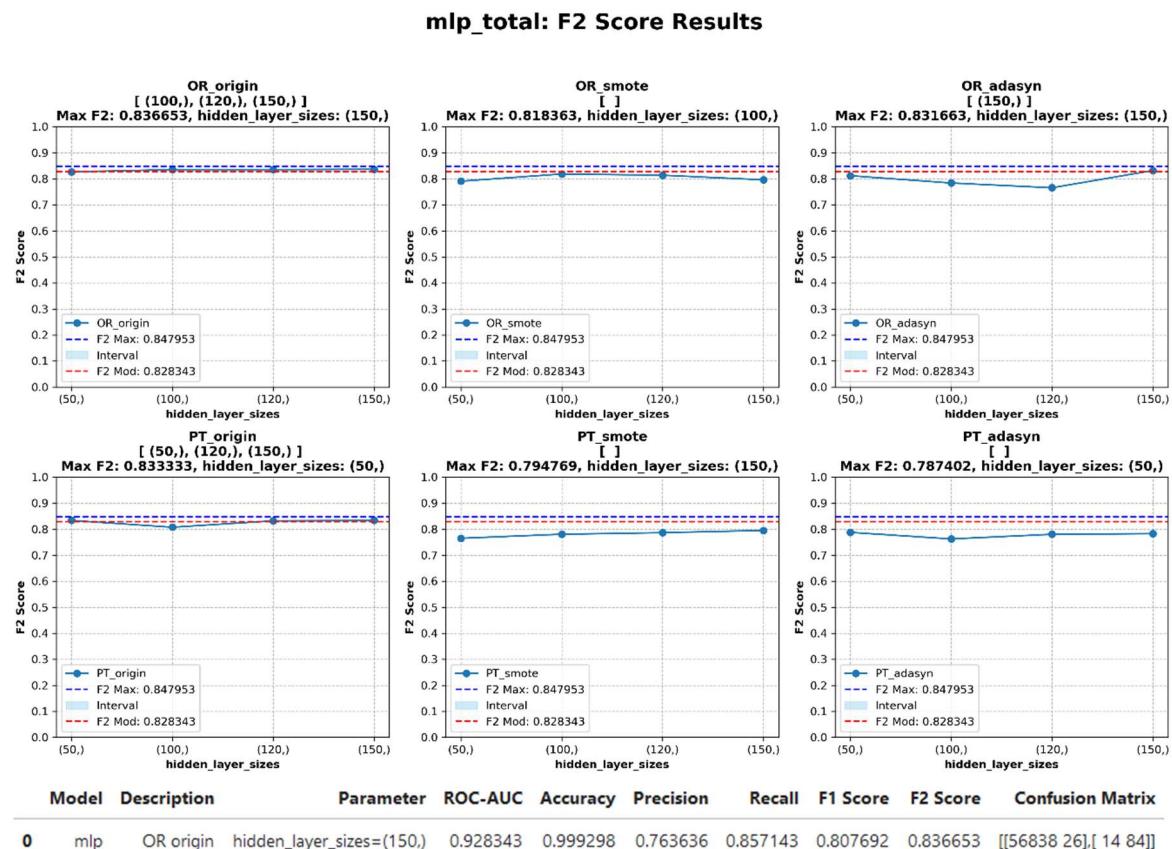


Gráfico 38: Modelo MLP evolución F2-Score frente al parámetro principal.

Elaboración propia.

El gráfico de rendimiento de este algoritmo muestra una tendencia lineal y predecible en todas las distribuciones, con un F2 Score inicial de 0.8366, cercano al mejor modelo por defecto. Sin embargo, no presenta mejoras significativas tras la optimización de parámetros.

Rendimiento Lineal

El algoritmo mantiene un rendimiento estable con F2 Scores entre 0.8366 y 0.7874, lo que sugiere que, aunque es confiable, carece de capacidad de evolución. Esta estabilidad indica que no es ideal para situaciones que requieran mejoras continuas.

Tratamiento del Sesgo (OR vs PT)

El algoritmo funciona mejor con las variables originales y sesgo intacto, mientras que la transformación de poder (PT) no aporta mejoras significativas, incluso perjudicando el rendimiento en algunos casos.

Tratamiento del Desequilibrio de Clases (OR vs SMOTE o ADASYN)

Las mejores distribuciones que superan el rendimiento por defecto se encuentran en la distribución original (OR). Sin embargo, ninguna configuración alcanza los niveles del mejor modelo por defecto, lo que refuerza la limitada capacidad de optimización.

Análisis de la Mejor Solución

La mejor configuración se obtiene en OR_origin con un F2 Score de 0.8366 y hidden_layer_sizes = 150, lo que confirma que el modelo trabaja mejor con variables sesgadas y un desequilibrio de clases, pero sin mejoras significativas.

Conclusiones

1. Comportamiento lineal y predecible, con un rendimiento constante pero sin margen de mejora.
2. El sesgo original en las variables favorece al modelo, ya que la PT no es eficaz.
3. Las distribuciones originales superan al modelo por defecto, pero sin mejoras sustanciales.
4. Baja capacidad de optimización, lo que lo convierte en una opción menos atractiva frente a otros algoritmos más flexibles.

En resumen, el algoritmo es estable pero carece de capacidad de evolución, por lo que no es adecuado para escenarios que requieran optimización continua.

5.5. Algoritmos basados en instancia: KNN

El parámetro principal en el algoritmo K-Nearest Neighbors (KNN)^[15] es n_neighbors. Este parámetro especifica el número de vecinos más cercanos que se utilizarán para tomar una decisión de clasificación o regresión.

La elección adecuada del valor de n_neighbors es crucial, ya que puede tener un impacto significativo en el rendimiento y la precisión del modelo KNN. Aquí hay algunas consideraciones clave sobre el parámetro n_neighbors:

Número de vecinos: Elige el número de vecinos más cercanos que se considerarán en la clasificación o regresión. Un valor pequeño de n_neighbors (como 1) significa que

solo se considerará el vecino más cercano, lo que puede hacer que el modelo sea sensible al ruido en los datos y a las fluctuaciones locales. Por otro lado, un valor grande de `n_neighbors` (como 10 o más) suaviza la influencia de vecinos individuales y puede perder algunos detalles locales en los datos. Es importante encontrar un equilibrio entre tener en cuenta suficientes vecinos para obtener una estimación precisa y evitar el sobreajuste.

Tamaño del conjunto de datos: El valor de `n_neighbors` también puede depender del tamaño del conjunto de datos. Si tienes un conjunto de datos pequeño, es posible que desees utilizar un valor más pequeño de `n_neighbors` para evitar un ajuste excesivo. Por el contrario, si tienes un conjunto de datos grande, puedes permitir un valor mayor de `n_neighbors` para capturar mejor la variabilidad de los datos.

Distribución de clases o valores: También debes considerar la distribución de las clases (en clasificación) o los valores objetivo (en regresión) en tu conjunto de datos. Si tienes clases o valores desequilibrados, es posible que debas ajustar el valor de `n_neighbors` para asegurarte de que cada clase o rango de valores tenga suficientes vecinos representativos.

Los gráficos de F2 Score muestran una tendencia descendente para todas las distribuciones que de hecho presentan valores muy similares.

knn_total: F2 Score Results

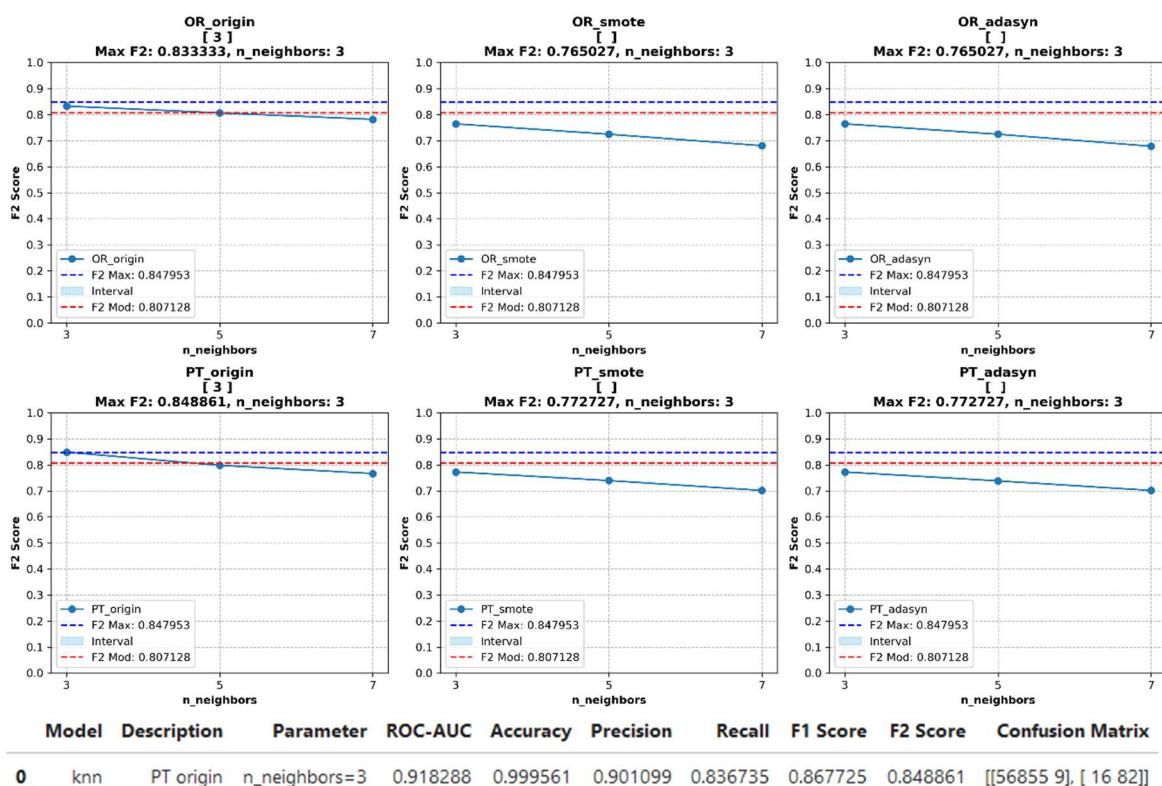


Gráfico 39: Modelo KNN evolución F2-Score frente al parámetro principal.
Elaboración propia.

El modelo KNN tiene un comportamiento muy similar a los dos anteriores partes de un F2-Score elevado, pero la adición de parámetro principal no contribuye a mejorar la puntuación de este.

De hecho puede comprobarse como en todas las distribuciones conforme aumenta el valor del parámetro disminuye el F2 Score, en todo el mayor valor lo proporciona en el inicio del rango con n_estimators para salir de la zona del valor por defecto del algoritmo.

Puede apreciarse como el tratamiento del sesgo de las variables permite alcanzar valores más altos para todas las distribuciones.

Funciona mejor con desequilibrio de clases frente a los modelos de smote y adasyn. El modelo no tiene potencial de mejora, su mejor puntuación la obtiene para F2 Score 0.85 con distribución PT_origin y valor de parámetro n_neighbours = 3.

La tabla final obtenida tras el entrenamiento y optimización de todos los modelos, 294 en total, ordenada por F2-Score, se muestra a continuación:

Existen claramente dos grupos de algoritmos con rendimiento F2 Score superior a 0.85 e inferior.

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	adaboost	OR smote	n_estimators=10	0.933568	0.999561	0.876289	0.867347	0.871795	0.869121	[[56852 12], [13 85]]
1	xgboost	OR smote	learning_rate=0.5	0.943623	0.999298	0.750000	0.887755	0.813084	0.856299	[[56835 29], [11 87]]
2	random_forest	PT origin	n_estimators=300	0.918323	0.999631	0.942529	0.836735	0.886486	0.855950	[[56859 5], [16 82]]
3	random_forest	PT origin	n_estimators=500	0.918323	0.999631	0.942529	0.836735	0.886486	0.855950	[[56859 5], [16 82]]
4	catboost	OR origin	iterations=200	0.913239	0.999649	0.964286	0.826531	0.890110	0.850840	[[56861 3], [17 81]]
5	catboost	PT origin	iterations=200	0.913239	0.999649	0.964286	0.826531	0.890110	0.850840	[[56861 3], [17 81]]
6	knn	PT origin	n_neighbors=3	0.918288	0.999561	0.901099	0.836735	0.867725	0.848861	[[56855 9], [16 82]]
7	mlp	OR origin	hidden_layer_sizes=(150),	0.928343	0.999298	0.763636	0.857143	0.807692	0.836653	[[56838 26], [14 84]]
8	decision_tree	OR origin	max_depth=5	0.892787	0.999491	0.905882	0.785714	0.841530	0.807128	[[56856 8], [21 77]]
9	regression_logistic	PT origin	C=0.1	0.836612	0.999192	0.825000	0.673469	0.741573	0.699153	[[56850 14], [32 66]]

Gráfico 40: Tabla final modelos fase parámetro. Elaboración propia.

1- Modelos con F2 Score > 0.85

Está conformado exclusivamente por algoritmo de árboles de decisión, random forest y especialmente por los de Boosting. La corrección de errores de clasificación iterativa que implementan los hace altamente efectivos para este tipo de problemas no solo en estudios académicos como el presente, si no en los reales como ya se ha mencionado en el caso del Sherlock de Revolut. Ofrecen un adecuado compromiso de precisión y recall, proporcionando modelos precisos y robustos.

En ese top-3 puede verse:

- La mejor solución la proporciona Adaboost con un total de 26 instancias mal clasificadas de las que son FN 14. Esta sería la solución que elegir en caso de

que se busque la solución óptima, en términos de F2 Score. Hay que recordar que este algoritmo en la fase default clasificó el penúltimo de la lista.

- XgBoost es la mejor solución cuando los FN penalizan muy fuerte y los FP tienen una ponderación menor. Esta solución consta de 40 instancia mal clasificadas, a priori un numero demasiado alto , pero por el contrario es la solución con el FN más bajo de todas con 11 el ROC-AUC más elevado de todos con 0.94.
- CatBoost proporciona una solución parecida a la de AdaBoost con 26 instancias mal clasificadas, la diferencia radica en que en este caso el FN es de 16,el más alto de los tres, lo que penaliza notablemente el valor F2 Score.

El ultimo algoritmo con F2 Score mayor a 0.85 es el random forest 0.8504 también con 28 instancias mal clasificadas y el segundo FN más alto de todos con 15.

Hay que destacar que el top 3 de algoritmos obtienen su mejor performance en la primera mitad del rango del parámetro principal. La estrategia de optimización de hiperparámetro clásica es aumentar el valor del parámetro principal, dado que esto no se produce puede intuirse que las soluciones contempladas están muy cerca de ser la más optima.

2- Modelos con F2 Score < 0.85

Se compone de cuatro algoritmos knn, mlp, decisión_tree y regression_logistic comparten toda la característica de que tenían un pobre desempeño por default y muy baja mejora con la adición del parámetro principal.

En condiciones normales se omitirían todos aquellos modelos en fase de parámetros cuyo F2 Score fuera menor, que el F2 Score del mejor modelo en fase Default, esto es XgBoost OR_smote con un valor de 0.8479, que redondeando que en el punto de corte antes expuesto. Por esa razón, ahora en la fase de hiperparámetro se va a optimizar únicamente los que pasen de ese valor y se incorporaran aquellas soluciones que mejoren el F2 Score del algoritmo en fase de parámetro principal. Para facilitar la visualización los parámetros tienen abreviatura.

A continuación se procede a la optimización mediante hiper parámetros de los algoritmos del primer grupo, se aplicará únicamente a los algoritmos de boosting, XgBoost, AdaBoost, y CatBoost. Random Forest no mejoro el F2 Score con la adicción de parámetro principal por lo que no tiene sentido optimizar parámetros que puedan tener un resultado marginal.

Optimización XgBoost

Se procede a calcular modelos con los siguientes rangos de parámetro:

learning_rate (lr)= [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]

n_estimators (n_est)= [100, 200, 300, 400]

max_depth (md)= [3, 5, 7]

Se filtran las soluciones con F2 Score ≥ 0.85 para todos los algoritmos.

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	xgboost	OR smote [th=0.5, n_est=200, lr=0.3, md=7]	0.982876	0.999473	0.826923	0.877551	0.851485	0.866935	[[56846, 18], [12, 86]]
1	xgboost	OR smote [th=0.5, n_est=300, lr=0.1, md=7]	0.978151	0.999438	0.811321	0.877551	0.843137	0.863454	[[56844, 20], [12, 86]]
2	xgboost	OR smote [th=0.5, n_est=300, lr=0.3, md=7]	0.982525	0.999473	0.833333	0.867347	0.850000	0.860324	[[56847, 17], [13, 85]]

Gráfico 41: Tabla XgBoost fase hiper parámetro. Elaboración propia.

La mejor solución implica pasar de un F2 Score de 0.8562 a un 0.8669, mejorando el F2 Score de cualquier modelo de la fase de parámetro. Esto se consigue con una FN más de FN 11 a 12, pero con una reducción de FP de 10 observaciones, se pesa de 29 a 18. Esa reducción de FP permite pasar de un ROC-AUC de 0.9436 a un 0.9828.

La solución es muy sólida, ya que el principal hándicap que tenía era el mayor FN de su clase. Esta optimización alinea el FN con sus rivales manteniendo el mejor TP.

Optimización modelo CatBoost

Rango de parámetros empleados:

learning_rate (lr) = [0.01, 0.05, 0.1]

n_estimators (n_est)= [100, 150, 200, 250, 300]

max_depth (md)= [3, 5, 7]

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	Catboost	OR origin [th=0.5, n_est=100, lr=0.1, md=5]	0.975536	0.999649	0.953488	0.836735	0.891304	0.857741	[[56860, 4], [16, 82]]

Gráfico 42: Tabla CatBoost fase hiper parámetro. Elaboración propia.

La mejor solución obtenida tiene un F2 Score de 0.8577 frente al 0.8541 de la fase de parámetros, la optimización es muy modesta y consiste en reducir 2 observaciones la FN , pasando de 6 a 4.

Optimización modelo AdaBoost

Rango de parámetros empleados

learning_rate = [0.01, 0.05, 0.1, 0.5, 1]

n_estimators = [10, 15, 20, 25, 30]

max_depth = [3, 5, 7]

No ha proporcionado ninguna solución que mejore la anterior de fase de parámetros.

6. Conclusiones y trabajos futuros.

Se han entrenado un total de 510 modelos en este estudio.

La tipología de algoritmo, que mejor solución proporciona para este tipo de problema, es sin duda, un algoritmo de boosting. Los tres algoritmos finalistas constituyen la totalidad de esta clase.

Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	adaboost	OR smote	n_estimators=10	0.933568	0.999561	0.876289	0.867347	0.871795	0.869121 [[56852, 12], [13, 85]]
1	xgboost	OR smote	[th=0.5, n_est=200, lr=0.3, md=7]	0.982876	0.999473	0.826923	0.877551	0.851485	0.866935 [[56846, 18], [12, 86]]
2	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=5]	0.975536	0.999649	0.953488	0.836735	0.891304	0.857741 [[56860, 4], [16, 82]]

Gráfico 42: Tabla Ranking Final fase hiper parámetro. Elaboración propia.

A este respecto hay que mencionar que no existe una distribución ideal con la que se obtiene este resultado, tenemos dos algoritmos que entran en la clasificación con sesgo en las variables XgBoost y AdaBoost, en tanto que CatBoost sin él.

El equilibrado de clases mediante adasyn, ya se intuía que no contribuye a resolver el problema ya en fase default, quedo demostrado en la fase de parámetros en la que ni una sola adasyn, entro en el ranking final de la fase.

En cuanto al equilibrado mediante smote, presentaba una tasa de soluciones en el ranking final del 50%. Sin embargo, suponen el 66% de las soluciones del trio finalistas ocupando los dos primeros puestos. Sus soluciones han sido las más consistentes y las que mejor comportamiento han demostrado en las fases de parámetros e hiperparámetro.

El modelo catboost puede desecharse aunque tiene la mayor precision 0.9535 y el menor número de FP 4, y el menor recall de los tres modelos, con un valor de 0.8367. El modelo es muy preciso al marcar fraude pero sería el que más operaciones fraudulentas no detectadas tendría con 16 falsos negativos, más que cualquier otro modelo.

Es excelente en minimizar las falsas alertas (falsos positivos), pero su incapacidad para detectar tantos fraudes como los otros modelos lo hace menos deseable para un escenario donde la detección de operaciones fraudulentas es la prioridad absoluta.

AdaBoost es una opción atractiva por su excelente equilibrio entre precisión, 0.8763, y recall, 0.8673, lo que lo hace ideal si se busca detectar fraudes minimizando falsos positivos. Si bien tiene un recall ligeramente menor que XGBoost, su precisión es considerablemente mejor, lo que significa que se generarán menos alertas erróneas. Su F1 Score, 0.8718, muestra un balance sólido por lo que es el mejor modelo. Con solo 13 falsos negativos, está cerca de XGBoost, convirtiéndolo en el más adecuado si se busca un buen equilibrio entre detección de fraudes y reducir falsas alertas.

Adaboost , con parámetro n_estimators= 10, con distribución OR_smote.

Obtiene un F2 Score 0.8691, un ROC_AUC de 0.9335, y un total de 25 observaciones mal clasificadas de las cuales 12 son FP y 13 FN. A continuación se muestra curva ROC, variables mas importantes del modelo y el árbol de decisión que se genera.

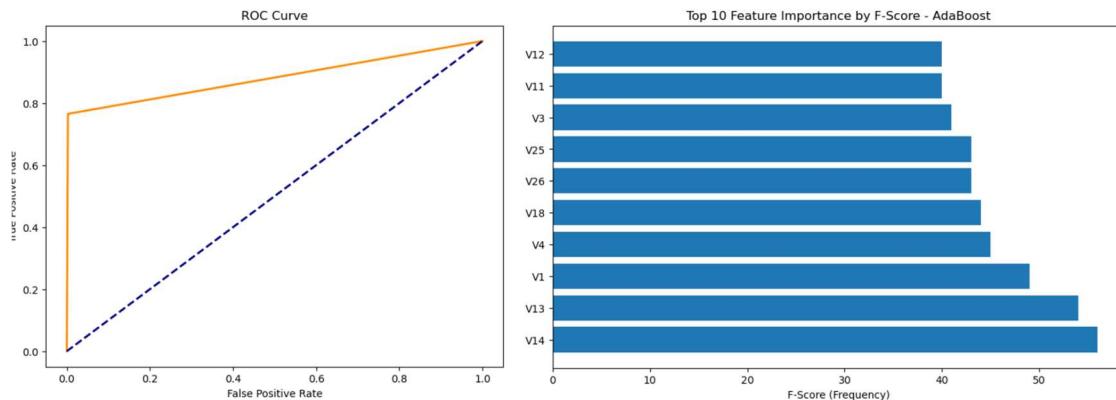


Gráfico 43: Curva ROC y variables del modelo AdaBoost. Elaboración propia.

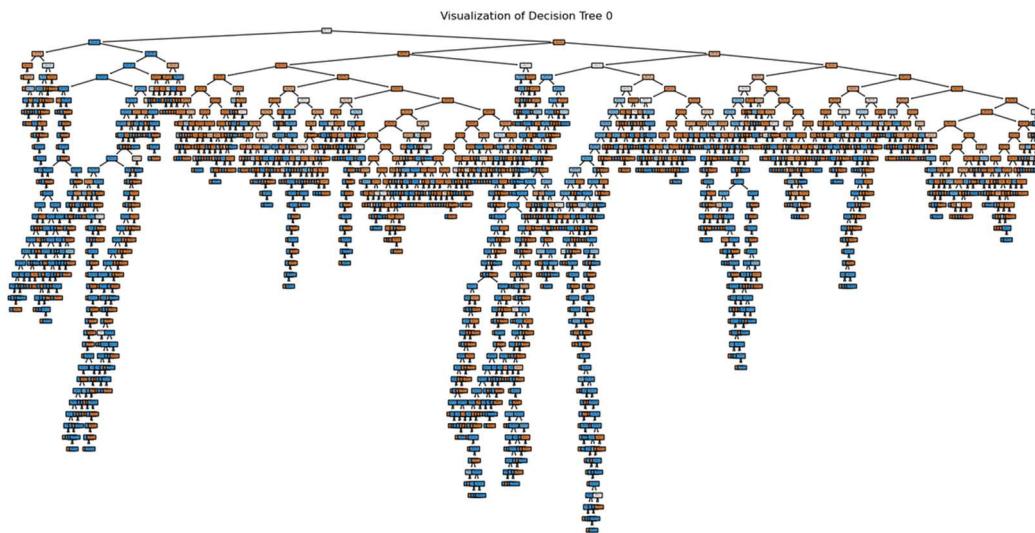


Gráfico 44: Árbol modelo AdaBoost. Elaboración propia.

XGBoost es ideal si el objetivo es maximizar la detección de fraudes y minimizar falsos negativos. Su recall, 0.8776, es el más alto, lo que garantiza que detectará más fraudes que otros modelos. Su precisión, 0.8269, es inferior a la de AdaBoost, lo que genera más falsos positivos, esta desventaja es aceptable en aras a una mayor detección de fraudes, incluso con más alertas erróneas. Su ROC-AUC, 0.9829 demuestra una excelente capacidad para discriminar fraudes, y con solo 12 falsos negativos, es el modelo más eficaz para asegurar que la menor cantidad de fraudes no sea detectada.

XgBoost proporciona la mejor solución para este dataset, la mejor configuración:
Distribución: OR_smote, Thresold = 0.5, n_estimators = 200, learning_ratio = 0.3,
max_depth = 7.

Obtiene un F2 Score 0.8669, un ROC_AUC de 0.9828, y un total de 30 observaciones mal clasificadas, de las cuales 18 son FP y 12 FN. Clasifica correctamente el 80% de las operaciones fraudulentas.

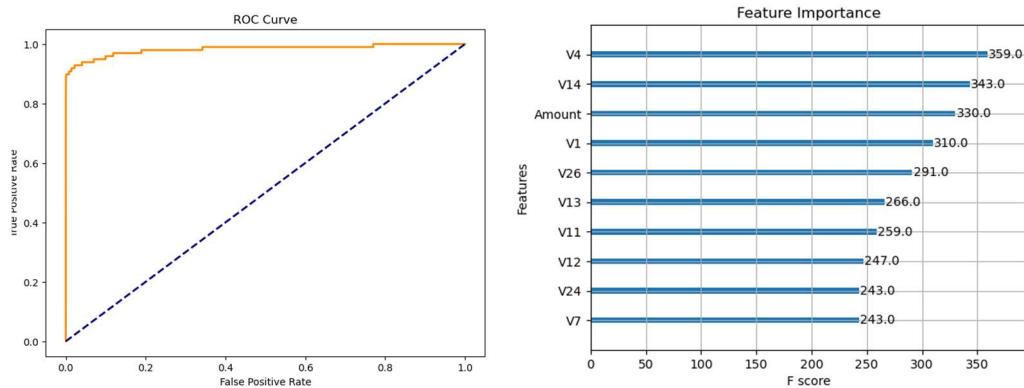


Gráfico 44: Curva ROC y variables del modelo XgBoost. Elaboración propia.

Por último se muestra, el árbol que genera el modelo. Emplea 9 variables y sería fácil de implementar en una API externa, y que constituye la tarea pendiente de este TFM.

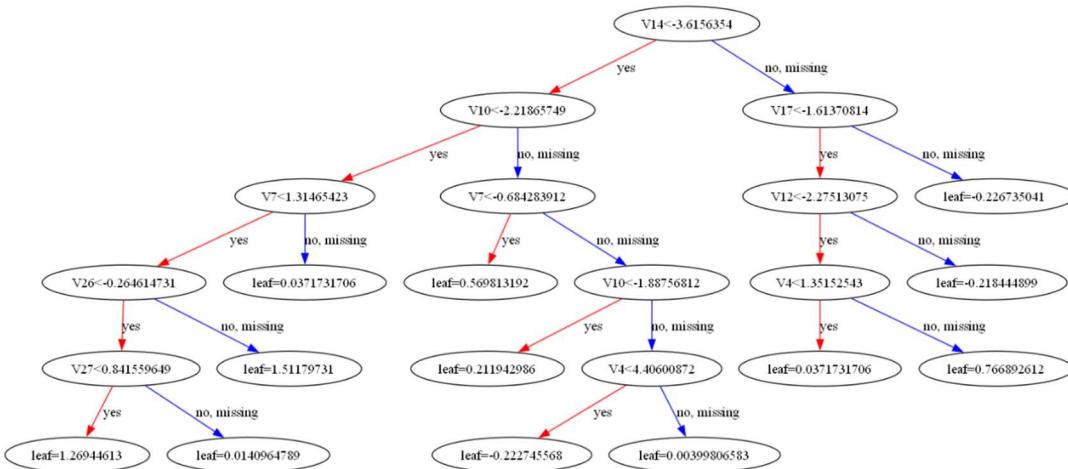


Gráfico 46: Detalle árbol modelo XgBoost. Elaboración propia.

Glosario

ATM: Automated Teller Machine, cajero automático de una entidad bancaria.

CNP: Card Not Present, tarjeta de crédito no está presente (físicamente).

FP: False Negative

FN: False Positive

KNN : K Near Neighbours

MLP: Multi Layer Perceptron

POS: Physical Point Of Sale, terminal de punto de venta ver TPV.

SEPA: Single euro Payment Area, Zona Única de Pagos de la Unión Europea.

TN : True Negative.

TPV: Terminal de punto venta.

TP : True Positive

Bibliografía

- [1] Diario: Expansión. Uso tarjeta crédito en España.URL:
<https://www.expansion.com/empresas/banca/2019/08/06/5d494423e5fdeac61f8b4589.html> Fecha: 24-06-2023.
- [2] Diario: Cinco Días. El País. URL:
https://cincodias.elpais.com/cincodias/2021/06/25/economia/1624641440_600138.html Fecha: 24-06-2023.
- [3] Diario: El diario. URL:
https://www.eldiario.es/economia/tarjeta-frente-efectivo-sigue-disparado-pandemia-miles-cajeros-cerrados_1_9148909.html Fecha: 24-06-2023.
- [4] Diario: Criptonoticias. URL:
<https://www.criptonoticias.com/regulacion/union-europea-busca-restringir-uso-dinero-efectivo-3-anos/> Fecha: 24-06-2023.
- [5] Diario: Expansión. URL:
<https://www.expansion.com/fiscal/2022/12/14/6398d147468aeb290e8b474e.html> Fecha: 24-06-2023.
- [6] Banco de España. Memoria Vigilancia y Supervisión de las infraestructuras de mercado 2021. URL:
https://www.bde.es/f/webbde/Secciones/Publicaciones/PublicacionesAnuales/MemoriaSupervisionBancaria/21/MemoriaSupervision2021_Cap5.pdf Fecha: 11-03-2023.
- [7] Banco Central Europeo. Seventh report on card fraud. October 2021.
URL:
<https://www.ecb.europa.eu/pub/pdf/cardfraud/ecb.cardfraudreport202110~cac4c418e8.es.pdf>.
Fecha 24-06-2023.
1. Gráfico 1
Imbalanced data preprocessing techniques for machine learning: a systematic mapping study.
Vitor Werner de Vargas, Jorge Arthur Schneider Aranda, Ricardo dos Santos Costa, Paulo Ricardo da Silva Pereira2, Jorge Luis Victoria Barbosa 1,2
<https://link.springer.com/article/10.1007/s10115-022-01772-8> Fecha = 24-06-2023.
 2. C Victoria Priscilla and D Padma Prabha. Credit card fraud detection: a systematic review. In *International Conference on Information, Communication and Computing Technology*, 290–303. Springer, 2019.

3. Facilitating User Authorization from Imbalanced Data Logs of Credit Cards Using Artificial Intelligence.
Vinay Arora, Rohan Singh Leekha, Kyungroul Lee, and Aman Kataria
<https://www.hindawi.com/journals/misy/2020/8885269/>
Fecha = 24-06-2023.
4. Dataset small size Abstract data set for Credit card fraud detection.
<https://www.kaggle.com/datasets/shubhamjoshi2130f/abstract-data-set-for-credit-card-fraud-detection> Fecha = 24-06-2023.
5. Dataset medium size Default of Credit Card Clients Dataset.
<https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset>
Fecha = 24-06-2023.
6. Dataset big size Credit Card Fraud Detection.
<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
Fecha = 24-06-2023.
7. <https://www.couchbase.com/press-releases/building-a-state-of-the-art-card-fraud-detection-system-in-9-months/> Fecha = 24-06-2023.
8. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
Fecha = 24-06-2023.
9. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>
Fecha = 24-06-2023.
10. https://xgboost.readthedocs.io/en/stable/get_started.html
Fecha = 24-06-2023.
11. <https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/>
Fecha = 24-06-2023.
12. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
Fecha = 24-06-2023.
13. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
Fecha = 24-06-2023.

14. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
Fecha = 24-06-2023.

15. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
Fecha = 24-06-2023.