# UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA



# TESIS DOCTORAL

Desarrollo de técnicas avanzadas de inteligencia artificial para optimización de coste y minimización de riesgo de compañías aseguradoras

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR


Alberto Gutiérrez Gallego

DIRECTORES

José Ignacio Hidalgo Pérez
Antonio Óscar Garnica Alcázar

# Desarrollo de técnicas avanzadas de inteligencia artificial para optimización de coste y minimización de riesgo de compañías aseguradoras

Alberto Gutiérrez Gallego


**UNIVERSIDAD COMPLUTENSE MADRID**

Memoria presentada para obtener el título de Doctor por la Universidad Complutense de Madrid en el Programa de Doctorado en Ingeniería Informática

Facultad de Informática

*Dirigido por*

José Ignacio Hidalgo Pérez y Antonio Óscar Garnica Alcázar

Madrid, 2025

# Dedicatoria

*A mis padres, por su apoyo y cariño constantes, y por ser siempre un ejemplo a seguir.*

*A mis abuelos, por la fuerza y el apoyo que me han brindado, presentes y en el recuerdo, y por ser un ejemplo de superación y esfuerzo.*

*A mi hermano, por estar siempre a mi lado, por su constante preocupación por mí y por darme la oportunidad de ser un ejemplo para él.*

*A mi novia, por su paciencia en los días difíciles y por ser un apoyo constante.*

Con todo mi cariño y gratitud.
Alberto Gutiérrez

# Agradecimientos

Escribir una tesis doctoral es, aunque a menudo se vea como una tarea individual, un esfuerzo colectivo. Es el trabajo del doctorando, sí, que dedica años a investigar, leer, programar, analizar, escribir y corregir. Pero también es el de sus directores, que orientan, corrigen y acompañan; el de sus compañeros, que comparten preocupaciones e ideas; el de las entidades que hacen posible el marco de trabajo; y, muy especialmente, el de sus seres queridos, que sostienen cuando flaquean las fuerzas.

Por eso, este trabajo no lo he hecho solo. Y estas líneas quieren ser un pequeño homenaje a todas esas personas que han estado cerca durante este largo proceso.

Quiero agradecer a mis directores su dedicación y paciencia. Su esfuerzo, su criterio y su disponibilidad han sido fundamentales en cada etapa de esta tesis, y siempre han estado ahí cuando más falta hacía, aportando claridad, confianza y perspectiva.

Agradezco también profundamente a Biztools y Aictuari, por brindarme la oportunidad de desarrollar esta investigación en el marco de un proyecto de doctorado industrial. Gracias por la libertad, la confianza y el compromiso con la innovación que me han permitido convertir ideas en propuestas reales. Parte de este trabajo forma ya parte activa de productos aplicados, y eso me enorgullece enormemente.

Mi agradecimiento se extiende también al sector médico enmarcado en el proyecto Genobia, por ofrecerme un entorno interdisciplinar que amplió mi visión y enriqueció esta tesis con nuevos enfoques y datos de gran valor. Quiero dedicar una mención especial a la memoria de Antonio López Farré, cuya contribución y cercanía dejaron una huella imborrable durante el desarrollo de este proyecto.

Gracias también a mis compañeros y compañeras de trabajo e investigación, por su interés genuino, por las conversaciones espontáneas que a veces resolvían un problema mejor que cualquier paper, y por estar ahí, incluso sin saberlo, como apoyo continuo.

Y, por supuesto, gracias a mi familia. A mis padres, por enseñarme con su ejemplo que el esfuerzo y la constancia tienen valor por sí mismos. Todo lo que soy tiene raíces en ellos. Y gracias a ti, Verónica, por estar cerca en los días fáciles y en los difíciles, por tu comprensión, por tu fuerza tranquila, por tu alegría, y por recordarme que hay vida más allá de cada línea de código y cada tabla de resultados.

Finalmente, gracias a ti, lector de esta tesis, por tu tiempo, tu interés y por formar parte, de alguna manera, de este camino que, aunque largo, ha merecido la pena.

# Acrónimos

**ADASYN** Adaptive Synthetic.

**AG** Algoritmo Genético.

**AUC** Area Under Curve.

**BUE** Balanced Underbagged Ensemble Approach.

**CA** Enfoque clásico.

**CNN** Condensed Nearest Neighbors.

**CRI** Claim Risk Indicator.

**DL** Deep Learning.

**ENN** Edited Nearest Neighbors.

**FN** False Negative.

**FP** False Positive.

**FPR** False Positive Rate.

**GenObIA-CM** Genética e Inteligencia Artificial al Servicio de la Prevención de la Obesidad.

**GLM** Generalized Linear Models.

**IA** Inteligencia Artificial.

**IMC** Índice de Masa Corporal.

**IR** Imbalanced Ratio.

**KPI** Key Performance Indicator.

**LR** Logistic Regression.

**MCR** Minority Class Ratio.

**ML** Machine Learning.

**MLP** Multi-layer Perceptron.

**MOGP** Multi-Objective Genetic Programming.

**NM1** NearMiss versión 1.

**NM2** NearMiss versión 2.

**NM3** NearMiss versión 3.

**NSGA** Non-dominated Sorting Genetic Algorithm.

**PCM** Performance Curve Mapping.

**PRC** Precision Recall Curve.

**PSO** Particle swarm optimization.

**PUI** Profit Underwriting Indicator.

**RF** Random Forest.

**ROC** Receiver Operating Characteristic.

**RUS** Random Undersampling.

**SHAP** SHapley Additive exPlanations.

**SMOTE** Synthetic Minority Over-sampling Technique.

**SMOTE-ENC** SMOTE-Encoded Nominal and Continuous.

**SMOTE-ENN** SMOTE-Edited Nearest Neighbors.

**STEM** SMOTE Edited Nearest Neighbour Mixup.

**TN** True Negative.

**TNR** True Negative Rate.

**TP** True Positive.

**TPR** True Positive Rate.

**VFDT** Very Fast Decision Tree.

**XGB** Extreme Gradient Boosting.

# Resumen

Las compañías de seguros desempeñan un papel fundamental en la economía al ofrecer productos que protegen a personas y empresas frente a riesgos e imprevistos. No obstante, la evaluación precisa del riesgo y la tarificación adecuada de pólizas requieren modelos predictivos robustos, bien calibrados y capaces de adaptarse a datos complejos y desbalanceados. Aunque el aprendizaje automático ha demostrado ser una herramienta valiosa en este contexto, su aplicación en el sector asegurador sigue enfrentando importantes desafíos, como el sesgo en los datos, el desequilibrio entre clases y la reticencia de las compañías a adoptar modelos con bajo nivel de interpretabilidad.

En esta tesis se desarrollan y analizan diferentes estrategias para mejorar la clasificación en problemas propios del ámbito asegurador mediante el uso de modelos *ensemble*. En primer lugar, se presenta el método *Balanced Underbagged Ensemble* (BUE), diseñado para mitigar el impacto del desbalanceo de clases y mejorar la estabilidad del aprendizaje. Posteriormente, se introduce un clasificador en cascada, aplicado en el marco del proyecto Genobia (clasificación de riesgo de obesidad), que permite estructurar la toma de decisiones en múltiples niveles, aumentando la capacidad de abstención en casos inciertos y mejorando la calidad de la clasificación. A continuación, se propone una optimización del espacio de decisión en modelos *ensemble* mediante la calibración de umbrales. Se comparan distintas estrategias: el enfoque clásico con umbral fijo, el uso de metaclasificadores, la optimización bayesiana de pesos y una nueva metodología basada en optimización multiobjetivo, orientada a mejorar el equilibrio entre la tasa de verdaderos positivos y falsos positivos.

El resultado final es el desarrollo de un clasificador multinivel que extiende las capacidades del clasificador en cascada y permite abordar de forma conjunta la predicción de siniestralidad y la optimización del resultado técnico. Este enfoque organiza el proceso en distintos niveles, donde cada uno se entrena con BUE y se optimiza de forma independiente, adaptándose a distintos objetivos de negocio. Su diseño modular y flexible no solo mejora la toma de decisiones en seguros, sino que también puede extrapolarse a otros sectores con problemas de clasificación complejos.

**Palabras Clave:** Aprendizaje Automático, Modelos *ensemble*, Datos desbalanceados, Clasificación multinivel, Sector asegurador, Predicción de siniestralidad

# Abstract

Insurance companies play a fundamental role in the economy by offering products that protect individuals and businesses against risks and unforeseen events. However, accurately assessing risk and pricing policies appropriately requires robust, well-calibrated predictive models capable of handling complex and imbalanced data. Although machine learning has proven to be a valuable tool in this context, its adoption in the insurance sector still faces significant challenges, such as data bias, class imbalance, and the reluctance of companies to adopt models with low interpretability.

This thesis develops and analyses various strategies to improve classification in insurance-related problems through the use of ensemble models. First, it introduces the Balanced Underbagged Ensemble method (BUE), designed to mitigate the impact of class imbalance and enhance learning stability. Next, it presents a cascade classifier, applied within the framework of the Genobia project (obesity risk classification), which structures decision-making across multiple levels, increasing the model's capacity to abstain in uncertain cases and improving classification quality. Subsequently, an optimisation of the decision space in ensemble models is proposed through threshold calibration. Different strategies are compared: the classic fixed-threshold approach, the use of metaclassifiers, Bayesian weight optimisation, and a novel methodology based on multi-objective optimisation aimed at improving the balance between true positive and false positive rates.

The final outcome is the development of a multi-level classifier that extends the capabilities of the cascade classifier and jointly addresses claims prediction and technical result optimisation. This approach organises the process into distinct levels, each trained with BUE and optimised independently to align with specific business objectives. Its modular and flexible design not only enhances decision-making in insurance but can also be applied to other sectors facing complex classification challenges.

**Keywords:** Machine Learning, Ensemble Models, Imbalanced Data, Multilevel Classification, Insurance Sector, Claim Prediction

# Índice general

# Índice de tablas

# Índice de figuras

# Capítulo 1

# Introdución

## 1.1 Contexto y motivación

Las compañías de seguros desempeñan un papel fundamental en la economía mundial al ofrecer productos que protegen a las personas y empresas frente a riesgos e imprevistos. Estos productos, conocidos como pólizas de seguro, son un contrato entre una compañía aseguradora y el tomador del seguro, donde la aseguradora se compromete a cubrir económicamente ciertos riesgos específicos, según el tipo de cobertura elegida, a cambio del pago de una prima. Proporcionan coberturas para una amplia variedad de elementos de valor: propiedades, vehículos, vida, salud y negocios.

La prima de seguro es el importe que el tomador paga a la aseguradora para recibir la cobertura contratada. Este valor debe ser suficiente para cubrir el coste esperado de los siniestros que puedan ocurrir durante la vigencia de la póliza, así como los costes operativos de la aseguradora. De esta forma se garantiza la sostenibilidad financiera de la compañía. El cálculo de la prima implica analizar los riesgos asociados a las coberturas contratadas dentro de la póliza.

El enfoque tradicional en el sector asegurador para el cálculo de la prima se basa en las matemáticas actuariales, una disciplina que cuantifica el riesgo en términos monetarios. Este enfoque utiliza modelos estadísticos y probabilísticos para estimar la probabilidad de que ocurra el evento asegurado, por ejemplo, un siniestro y su coste asociado. Dentro de las compañías, este cálculo está en manos de los actuarios, profesionales especializados que consideran diversos factores clave, entre los que se incluyen:

- La naturaleza del riesgo. Por ejemplo, en seguros de vida, se consideran tablas de mortalidad y en seguros de vehículos, la probabilidad de accidentes y los costes asociados.
- La duración del contrato. Cuanto mayor sea el período de cobertura, mayor será el riesgo asumido por la aseguradora.

- Los importes mínimos de cobertura establecidos por la legislación vigente. En el caso de los seguros de automóviles, en Europa los límites mínimos de cobertura son de 70 millones de euros para daños personales y 15 millones para daños materiales.
- Información histórica y estadística de la aseguradora, como datos sociodemográficos, siniestralidad previa y experiencia en la gestión de riesgos similares.

Aunque el enfoque tradicional es ampliamente aceptado, enfrenta retos importantes en el contexto actual, marcado por el aumento exponencial de los datos disponibles y un entorno empresarial en constante transformación. La naturaleza cambiante de los riesgos, las expectativas de los consumidores y la competencia dentro del sector han puesto de manifiesto las limitaciones de los modelos tradicionales. En particular, su capacidad limitada para capturar y manejar la complejidad de los datos actuales y adaptarse a escenarios en constante evolución.

Por otro lado, la integración de innovaciones en el sector asegurador presenta desafíos organizativos, tecnológicos y culturales. Muchas aseguradoras muestran resistencia al cambio, priorizando la estabilidad y confiabilidad de sus procesos actuales sobre la adopción de tecnologías nuevas. Este enfoque conservador, aunque comprensible en un sector que opera sometido a un alto nivel de regulación, ralentiza la implementación de soluciones que podrían ofrecer ventajas competitivas significativas.

En este contexto, el sector asegurador se encuentra en un momento clave para su transformación. La investigación y el desarrollo de nuevos enfoques, junto con la exploración de cómo complementar los modelos tradicionales, son factores esenciales para garantizar la evolución del sector asegurador. Adaptarse a las demandas del mercado y aprovechar los avances tecnológicos no solo permitirá a las aseguradoras optimizar sus operaciones, sino también ofrecer productos más precisos y personalizados a las necesidades de los clientes.

Este trabajo se fundamenta en la necesidad de transformar y modernizar los enfoques tradicionales del sector asegurador mediante la incorporación de nuevas herramientas basadas en Inteligencia Artificial (IA). En particular, se centra en el desarrollo de modelos de predicción de siniestralidad y de rentabilidad de clientes, adaptados a las características específicas de cada asegurado. Esta aproximación busca superar las limitaciones de los métodos clásicos, ofreciendo soluciones que permitan optimizar la toma de decisiones en un sector cada vez más competitivo y dinámico.

## 1.2 Inteligencia Artificial en el sector de seguros

La IA y el aprendizaje automático (del inglés *machine learning*, ML) han revolucionado múltiples sectores al ofrecer nuevas herramientas para el análisis de datos y la toma de decisiones. En el campo de los seguros, estas tecnologías han sido aplicadas para optimizar el cálculo de primas, mejorar la detección de fraudes y personalizar las ofertas de seguros. Sin embargo, las compañías aseguradoras son reticentes a incorporarlas.

Una de las principales razones es la preferencia por los modelos actuariales tradicionales, ya que son ampliamente aceptados y destacan por su transparencia e interpretabilidad, cualidades que muchos modelos de IA aún no pueden igualar. Según Wüthrich and Merz [123], los modelos basados en aprendizaje profundo (del inglés *Deep Learning*, DL) presentan el problema de la "caja negra", donde el proceso de toma de decisiones no es fácilmente interpretable. Esto genera incertidumbre entre los actuarios, quienes deben justificar sus decisiones, especialmente en un sector tan altamente regulado como el asegurador.

Otro factor relevante a la reticencia de las aseguradoras es la preocupación sobre la robustez y la fiabilidad de los modelos de IA cuando se implementan en aplicaciones del mundo real. Problemas como el sobreajuste (que se da cuando un modelo se ajusta demasiado a los datos de entrenamiento y obtiene un rendimiento pobre al aplicarse a nuevos datos) y la variabilidad del rendimiento, representan riesgos que no están dispuestas a asumir. Adicionalmente, la inercia organizacional y cultural de las aseguradoras, caracterizada por una tendencia conservadora, dificulta la adopción de tecnologías disruptivas. Esta resistencia se traduce en una implementación cautelosa de la IA, ya que las aseguradoras buscan minimizar riesgos y garantizar la estabilidad de sus procesos.

Aun así, el número de investigaciones científicas en el sector ha aumentado en los últimos años, permitiendo la creación de modelos más robustos, interpretables y explicables. En el trabajo de Xinhua et al. [126], se analiza la aplicación de algoritmos de ML para predecir la probabilidad de siniestros y las compensaciones acumuladas en seguros de automóviles. Los resultados muestran que el uso de técnicas avanzadas, como los modelos *ensemble*, mejoran significativamente la precisión en la predicción de siniestros en comparación con los modelos tradicionales.

Además, los datos masivos generados por los clientes, tales como historiales de conducción, siniestros anteriores y datos telemáticos, ofrecen una oportunidad única para mejorar la evaluación de riesgos. Hanafy and Ming [54] demuestran cómo con toda esta información y modelos de IA basados en árboles, redes neuronales y algoritmos *ensemble*, se consigue enriquecer la precisión en la detección de siniestros y, a su vez, en la personalización de primas de seguro. Además, el estudio destaca tanto la necesidad de abordar los desafíos técnicos y éticos asociados al uso de ML, como la de garantizar la transparencia de los modelos y proteger la privacidad de los datos de los clientes.

## 1.3 Objetivos

El objetivo principal de esta tesis es desarrollar un sistema avanzado de predicción de riesgos en el sector asegurador, integrando técnicas de ML y aprovechando el poder computacional de las unidades de procesamiento gráfico (GPU) para optimizar los modelos predictivos. Este objetivo general se desglosa en los siguientes objetivos específicos:

**Creación de una base de datos robusta**
La creación de una base de datos robusta es el primer paso crítico en el desarrollo de

cualquier modelo de análisis de datos. Este objetivo implica la recopilación, limpieza y normalización de datos históricos de pólizas de seguro, incluyendo variables clave como: características del asegurado, historial de siniestros, historial de recibos y datos contextuales como la ubicación geográfica y el tipo de vehículo. Es fundamental garantizar que la base de datos sea lo suficientemente amplia y diversa para contemplar una variedad de escenarios y riesgos potenciales, que van a permitir que los modelos de predicción tengan un alto grado de precisión y generalización. En la literatura se ha demostrado que la calidad y la preparación de los datos son fundamentales para el éxito de los modelos de ML [59]. Además, una base de datos bien estructurada facilita la implementación de técnicas de análisis avanzado, como el análisis predictivo, permitiendo identificar patrones ocultos [99].

**Creación de un modelo de predicción de siniestralidad**
El segundo objetivo es desarrollar un modelo de predicción de siniestralidad, que permita identificar con precisión si una póliza va a tener al menos un siniestro en los próximos doce meses desde su contratación. Este modelo se basará en técnicas de ML supervisado, sobre todo técnicas basadas en árboles, ya que son modelos de "caja blanca" que permiten una mejor interpretabilidad. Además, integrará modelos *ensemble* con el fin de mejorar la robustez y el rendimiento de las predicciones. La predicción de siniestralidad no solo es esencial para la gestión de riesgos, sino también para la personalización de las primas y la mejora de la eficiencia operativa en las aseguradoras. Este objetivo de desarrollo corresponde al capítulo 6, cuya información está sujeta a un acuerdo de confidencialidad con la empresa, por lo que no se incluye en la versión pública de la tesis.

**Creación de un modelo de predicción de rentabilidad**
El tercer objetivo es desarrollar un modelo de predicción de rentabilidad, definida como la diferencia entre la prima captada neta y el importe de siniestros producidos durante el periodo vigente de la póliza. Se busca predecir si un cliente va a ser rentable o no en los doce meses siguientes a la entrada en vigor de la póliza. Este modelo permitirá a las aseguradoras conocer qué clientes de su cartera generarán beneficio o pérdida y, también, les ayudará a realizar una selección más específica de los clientes de nueva producción, es decir, clientes de nuevo ingreso en la compañía. Al igual que el anterior, este objetivo se desarrolla en el capítulo 6, cuya información permanece restringida por un acuerdo de confidencialidad con la empresa, motivo por el cual no se incluye en la versión pública de la tesis.

## 1.4 Metodología

Para abordar los objetivos de esta investigación, se ha adoptado una metodología basada en la aplicación de técnicas de ML sobre un conjunto de datos históricos de pólizas de seguros de automóviles. El proceso de investigación incluye las siguientes etapas:

a) Recolección de datos. Se ha obtenido un conjunto de datos de pólizas de seguros que

contiene información detallada sobre las características del asegurado, el vehículo, el historial de siniestros y el historial de recibos. Este nivel de granularidad es esencial en el sector asegurador, donde tanto las características de la póliza como los antecedentes de siniestralidad condicionan directamente el riesgo y la rentabilidad.

b) Preprocesamiento de datos. Para abordar esta etapa se ha necesitado colaboración del equipo experto actuarial, que ha ayudado a conocer cada una de las variables utilizadas en este trabajo. Se ha realizado una fase de limpieza de variables al igual que de filas. También se han realizado diferentes fases de ingeniería de variables y filtrado de datos.

c) Selección del modelo. Se han evaluado catorce algoritmos de ML supervisado, seleccionando aquellos con mayor capacidad para adaptarse a la complejidad y al desbalance presentes en los datos aseguradores

d) Entrenamiento y validación del modelo. Dependiendo del problema a abordar se han establecido diferentes conjuntos de entrenamiento, validación y test. Este diseño experimental busca garantizar la robustez de los resultados y minimizar sesgos derivados del fuerte desbalanceo entre pólizas con y sin siniestro, para lo cual se han aplicado distintas estrategias de tratamiento de clases desbalanceadas.

e) Evaluación y optimización del modelo. Se han utilizado diferentes métricas como son el recall de cada clase, la media geométrica de los recalls, el area bajo la curva (AUC, del ingles *Area Under the Curve*), Curvas ROC (del inglés *Receiver Operating Characteristic*), Younden Index, análisis de negocio e interpretación visual. La elección de estas métricas se justifica por la necesidad de equilibrar la capacidad predictiva del modelo con la visión de negocio, teniendo en cuenta que unos buenos resultados en términos métricos no siempre garantizan un impacto positivo en la práctica. En este contexto, resulta fundamental identificar de forma fiable los riesgos sin comprometer la rentabilidad global de la cartera

f) Interpretación y explicabilidad. Para esta fase se han utilizado herramientas como SHAP [118], que permiten interpretar la salida del modelo y explicar qué variables han contribuido en la predicción. Esta capacidad de interpretación resulta fundamental en el contexto asegurador, donde la transparencia y la justificación de las decisiones son requisitos clave tanto para la aceptación empresarial como para el cumplimiento normativo. Asimismo, representa un elemento decisivo para la adopción de estos modelos, en un sector que, por su carácter tradicional, suele mostrar reticencias a la incorporación de soluciones basadas en IA.

En conjunto, esta metodología no solo cubre las fases clásicas de un proyecto de ML, sino que se adapta a las particularidades del sector asegurador. De este modo, cada decisión metodológica, desde el preprocesamiento hasta la selección de métricas, se alinea con los objetivos de la investigación: mejorar la capacidad predictiva en la detección de riesgo y, al mismo tiempo, proporcionar información útil y explicable para la toma de decisiones a

nivel de negocio.

## 1.5 Organización del documento

El documento está organizado de la siguiente forma: el capítulo 1 introduce el sector asegurador y los problemas a abordar. El capítulo 2 recopila el estado del arte, tanto del ámbito asegurador como de los retos específicos a tratar. El capítulo 3 presenta una nueva técnica para el tratamiento de datos desbalanceados [1] en modelos *ensemble*. El capítulo 4 aborda un clasificador en cascada con capacidad de abstención. El capítulo 5 presenta una técnica de optimización de curvas de rendimiento para modelos *ensemble*. Finalmente, el capítulo 6 se implementa un clasificador multinivel que integra técnicas de los capítulos anteriores y constituye la herramienta final utilizada por la empresa.

## 1.6 Financiación

Este trabajo ha sido financiado fundamentalmente por la ayuda de doctorado industrial de la Comunidad de Madrid IND2020/TIC-17435 y por Biztools S.L. Agradecemos también las ayudas del Ministerio de Economía y Competitividad de España PID2021-125549OB-I00 y PDC2022-133429-I00 cofinanciadas por la UE Next Generation, y por los Fondos Estructurales de la UE a través de la ayuda de la Comunidad de Madrid B2017/BMD3773 (GenObIA-CM).

## 1.7 Publicaciones

Las publicaciones generadas a partir de esta tesis son:

- Gutierrez-Gallego, A., Garnica, O., Parra, D., Velasco, J.M. and Hidalgo, J.I. (2025), Balanced Underbagged Ensemble Approach for Classifying Highly Imbalanced Datasets in the Insurance and Financial Sectors. Intelligent Systems in Accounting, Finance and Management. En este artículo se propone un método ensemble específico para el tratamiento de conjuntos de datos altamente desbalanceados en el sector asegurador y financiero. Esta aportación corresponde al capítulo 3, dedicado al método BUE.
- Gutierrez-Gallego, A., Garnica, O., Parra, D., Velasco, J.M. and Hidalgo, J.I. (2025), Optimising Performance Curves for Ensemble Models through Pareto Front Analysis of the Decision Space. Expert Systems. En este artículo se introduce la propuesta de optimización multiobjetivo de umbrales de decisión (PCM) mediante análisis de frentes de Pareto en el espacio de decisión. Esta contribución se desarrolla en el capítulo 5.

[1] En este documento se utiliza el término desbalanceo por coherencia con la jerga técnica habitual en informática y aprendizaje automático, aunque el término correcto en español sería desequilibrio.

- Gutiérrez-Gallego, A., Zamorano-León, J. J., Parra-Rodríguez, D., Zekri-Nechar, K., Velasco, J. M., Garnica, Ó., ... and Hidalgo, J. I. (2024). Combination of machine learning techniques to predict Overweight/Obesity in Adults. Journal of Personalized Medicine, 14(8), 816. Este artículo presenta el clasificador en cascada enfocado en la predicción del sobrepeso/obesidad en adultos de la Comunidad de Madrid. Su contenido esta relacionado directamente con el capítulo 4.
- Parra, D., Gutiérrez-Gallego, A., Garnica, O., Velasco, J. M., Zekri-Nechar, K., Zamorano-León, J. J., ... and Hidalgo, J. I. (2022). Predicting the Risk of Overweight and Obesity in Madrid—A Binary Classification Approach with Evolutionary Feature Selection. Applied Sciences, 12(16), 8251. Este artículo presenta un modelo predictivo del riesgo de sobrepeso u obesidad en la Comunidad de Madrid mediante selección evolutiva de variables. Esta contribución se relaciona con el capítulo 4, donde se introduce el clasificador en cascada aplicado al proyecto GenObIA.
- Parra, D., Gutiérrez-Gallego, A., Velasco, J. M., Villanueva, R. J., and Hidalgo, J. I. (2023, July). Modelling the Transmission Dynamics of Obesity: A Multi-Objective Approach Using Dynamic Structured Grammatical Evolution. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation (pp. 73-74). Esté artículo de conferencia, esta relacionado con el problema de Obesidad presentado en el capítulo 4, y me ha permitido ampliar el conocimiento algorítmico en técnicas evolutivas.
- Parra, D., Gutiérrez, A., Velasco, J. M., Garnica, O., and Hidalgo, J. I. (2022, April). Combining the Properties of Random Forest with Grammatical Evolution to Construct Ensemble Models. In International Conference on the Applications of Evolutionary Computation (Part of EvoStar) (pp. 61-76). Cham: Springer International Publishing. Este artículo de conferencia no está directamente vinculado con esta tesis, aunque ha servido para profundizar en la combinación de técnicas ensemble y gramáticas evolutivas.
- Parra, D., Gutiérrez, A., Garnica, O., Lanchares , J., and Hidalgo, J. I. (2021), "Selección óptima de variables mediante computación evolutiva para algoritmos de clasificación," in XIX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 20/21). Este artículo de conferencia, relacionado con el capítulo 4, propone una estrategia de selección de variables mediante algoritmos evolutivos aplicada a problemas de clasificación, enmarcada dentro del proyecto GenObIA.

# Capítulo 2

# Estado del arte

Las pólizas de seguros ayudan a repartir el riesgo entre los asegurados utilizando la ley de los grandes números, es decir, a medida que aumenta el número de pólizas contratadas, el coste medio de la cartera se normaliza. Esta es la clave principal del funcionamiento de las compañías de seguros, siendo la captación de clientes una de sus principales tareas. Los actuarios son los encargados de valorar y evaluar a los clientes, con el objetivo de fijar precios justos en función del riesgo que representa cada póliza. La forma más fácil de hacerlo es promediar el coste por póliza y aplicar una tarifa plana a toda la cartera, pero el problema es que, a medida que aumenta el número de pólizas, aparecen distintos segmentos de riesgo. Por eso, desde finales del siglo XX, las aseguradoras utilizan modelos lineales generalizados (del inglés *Generalized Linear Models*, GLM) [84] para fijar el precio de sus carteras en función del riesgo de los distintos segmentos a los que se dirige la póliza.

## 2.1 Modelos Lineales Generalizados (GLM)

Los modelos GLM fueron formalizados y extendidos por Nelder y Wedderburn [94] en 1972, lo que proporcionó una base metodológica para su aplicación tanto en seguros como en otras áreas que requieren modelar variables con distribuciones no normales, tales como la distribución binomial para datos de conteo o la de Poisson para eventos poco frecuentes.

Desde entonces, se han consolidado como una herramienta fundamental en el modelado estadístico dentro del sector asegurador. Son una generalización de los modelos de regresión lineal clásicos y destacan por generar parámetros con interpretaciones claras que se agregan multiplicativamente para calcular las primas de seguro.

Estos modelos sirven para estimar el coste de los siniestros, es decir, la parte proporcional de la prima del seguro destinada a cubrir las pérdidas y los gastos asociados a ellos. Este cálculo se basa en la frecuencia (ecuación (2.2)) y gravedad de los siniestros (ecuación (2.3)), que serían las variables de respuesta en dos posibles modelos GLM. La frecuencia

de los siniestros mide la proporción de siniestros respecto a la exposición ganada (ecuación (2.1)), mientras que la gravedad de los siniestros cuantifica el importe medio por siniestro, calculado como el cociente entre el importe total de las pérdidas y el número de siniestros. A continuación se presentan las ecuaciones mencionadas:

$$E_g = \sum_{i=1}^{n} \frac{D_i}{365} \tag{2.1}$$

$$\text{Frecuencia de siniestros} = \frac{N}{E_g} \tag{2.2}$$

$$\text{Gravedad de siniestros} = \frac{S}{N} \tag{2.3}$$

donde:

- $E_g$ es la exposición ganada, que representa el riesgo asumido por la aseguradora ajustado por el tiempo en vigor de las pólizas.
- $n$ es el número total de pólizas.
- $N$ es el número total de siniestros en el período analizado.
- $D_i$ es el número de días que la póliza $i$ estuvo en vigor asumiendo un año estándar de 365 días para la prorrata temporal.
- $S$ es el importe total de las pérdidas por siniestros en el período analizado.

Para construir un modelo GLM se deben abordar los siguientes puntos:

- **Escoger una familia de distribución**. Los modelos GLM permiten que la variable respuesta (variable dependiente que el modelo intenta predecir en función de una combinación lineal de las variables independientes) siga diversas distribuciones de la familia exponencial, lo que proporciona flexibilidad para modelar una variedad de tipos de datos.
  - Binomial. Usada para datos de respuesta binaria (éxito/fracaso, sí/no).
  - Poisson. Apropiada para contar datos, como el número de veces que ocurre un evento.
  - Gamma. Utilizada para datos que son siempre positivos y asimétricos, como los tiempos de espera.
  - Normal. Para respuestas que son continuas y simétricas.
- **Modelado del predictor lineal**. El predictor lineal en un modelo GLM es una combinación lineal de las variables explicativas, similar a la regresión lineal:

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in} \tag{2.4}$$

donde $\eta_i$ es el predictor lineal para la observación $i$, $\beta_0, \beta_1, \ldots, \beta_n$ son los coeficientes del modelo, y $x_{i1}, x_{i2}, \ldots, x_{in}$ son las variables explicativas correspondientes.

- **Determinación de la función de enlace**. La función de enlace es un componente fundamental en los modelos GLM, ya que establece la relación entre la media de la variable respuesta ($\mu_i$) y el predictor lineal definido anteriormente. Esta relación se expresa mediante:

$$g(\mu_i) = \eta_i \tag{2.5}$$

  Esta formulación permite adaptar el modelo a diferentes tipos de variables respuesta, manteniendo una estructura lineal sobre $\eta_i$. La elección de la función de enlace depende de la distribución de la variable respuesta y de la naturaleza del problema. Algunas de las funciones de enlace más comunes son:

  - Función identidad. Se utiliza cuando la variable respuesta es continua y se asume una distribución normal. Es el caso típico de la regresión lineal clásica.

$$g(\mu) = \mu \tag{2.6}$$

  - Función logit. Se emplea cuando la variable respuesta es binaria, y se modela mediante una distribución binomial. Esta es la función de enlace utilizada en la regresión logística, y garantiza que las predicciones estén acotadas entre 0 y 1.

$$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right) \tag{2.7}$$

  - Función logarítmica. Se aplica en contextos donde la variable respuesta representa un recuento (por ejemplo, número de siniestros), modelado mediante una distribución de Poisson. Esta función asegura que las predicciones sean estrictamente positivas.

$$g(\mu) = \log(\mu) \tag{2.8}$$

- **Estimación de los parámetros del modelo,** $\beta$. Se estiman normalmente mediante el método de máxima verosimilitud, que consiste en encontrar los valores de los parámetros que mejor explican los datos observados según el modelo. Es decir, se busca ajustar el modelo de forma que reproduzca de la mejor manera posible la distribución observada en los datos.

- **Diagnóstico del modelo**. Una vez que el modelo ha sido ajustado, es fundamental evaluar su calidad y validez. Para ello, se realiza un análisis de los residuos, que consiste en examinar las diferencias entre los valores observados y los valores predichos por el modelo. Este análisis permite detectar patrones no explicados por el modelo, posibles errores de especificación o violaciones de los supuestos. Además, se comprueba la posible presencia de sobredispersión, especialmente en distribucio-

nes como la de Poisson o la binomial, donde la varianza puede superar la media esperada, afectando la fiabilidad de las estimaciones.

- **Interpretación**. Los resultados de los modelos GLM deben interpretarse en función del contexto del problema. Los coeficientes estimados representan la relación entre cada variable explicativa y la variable respuesta, indicando la dirección y magnitud del efecto. Es fundamental realizar un análisis de significancia estadística sobre dichos coeficientes, con el fin de identificar qué variables tienen un efecto significativo sobre la respuesta.

Los modelos GLM destacan por su transparencia y capacidad interpretativa, permitiendo a los actuarios entender cómo contribuye cada factor a la formación del riesgo. Además, su flexibilidad para adaptarse a diferentes tipos de distribuciones de datos los hace extremadamente útiles en el amplio espectro de aplicaciones dentro del sector asegurador.

Sin embargo, a pesar de sus cualidades, los modelos GLM tienen limitaciones, especialmente cuando se enfrentan a datos con estructuras complejas o interacciones no lineales entre variables. También presuponen la independencia de las observaciones, lo que puede no ser válido en todos los contextos aseguradores.

Con todas estas aportaciones, los modelos GLM han demostrado durante el siglo XXI que son modelos que funcionan y aportan un gran valor dentro de las aseguradoras, pero en las últimas décadas, con la recopilación de datos masivos, se ha abierto una alternativa mediante el uso del ML.

## 2.2 Aprendizaje automático (ML)

ML es un campo de la IA que se centra en el desarrollo de sistemas capaces de aprender y mejorar automáticamente a partir de la experiencia, detectando patrones en los datos y generando modelos predictivos. El concepto fue formalmente definido por Mitchell [89], quien estableció que:

> *Un programa aprende de la experiencia $E$ respecto a una tarea $T$ y una medida de desempeño $P$, si su desempeño en la tarea $T$, medido por $P$, mejora con la experiencia $E$.*

Existen diferentes categorías de ML según el problema a abordar:

- **Aprendizaje supervisado**: Este enfoque se utiliza cuando los datos de entrada ($X$) están relacionados con etiquetas o valores de salida conocidos ($y$). El objetivo es construir un modelo entrenado con los datos conocidos y posteriormente predecir los valores de salida para datos nuevos no etiquetados. Este tipo de aprendizaje es utilizado ampliamente en problemas de clasificación y regresión. El problema se puede formular como:

$$\hat{y} = f(X; \theta) + \varepsilon, \tag{2.9}$$

donde:

- $\hat{y}$ es la predicción del modelo.
- $X$ es el conjunto de variables de entrada.
- $\theta$ representa los parámetros del modelo.
- $\varepsilon$ es el error aleatorio.

Los modelos más empleados son:

- Incremento del gradiente (*Gradient Boosting*). El Incremento del gradiente es una técnica basada en la creación de modelos secuenciales que buscan corregir los errores de sus predecesores. En cada ronda de entrenamiento se genera un modelo débil y se comparan sus predicciones con el resultado correcto esperado. La distancia entre la predicción y el valor correcto representa la tasa de error del modelo. A partir de estos errores se calcula el gradiente de la función de pérdida [93]. De esta manera se construye un modelo aditivo de manera progresiva hacia el escenario buscado.
- Árboles de decisión (*Decision Tree*). Los árboles de decisión son modelos estructurados en forma de árbol, donde cada nodo representa una pregunta sobre una variable de los datos y las ramas indican posibles respuestas. El árbol se ramifica hasta llegar a los nodos hoja, que representan la predicción final. Son fáciles de interpretar y útiles en muchos contextos, aunque pueden sobreajustarse a los datos si no se regulan correctamente [103].
- Clasificador por agregación de modelos (*Bagging Classifier*). El clasificador por agregación de modelos es una técnica que mejora la estabilidad y precisión de modelos individuales entrenando múltiples versiones de un mismo algoritmo con subconjuntos aleatorios del conjunto de datos mediante *bootstrap* (muestreo aleatorio con reemplazo). Posteriormente, las predicciones de estos modelos se combinan (por votación en clasificación o promedio en regresión) para generar una predicción final más robusta [19].
- Bosque aleatorio (*Ramdom Forest*). El bosque aleatorio es una variante de los clasificadores por agregación de modelos. En este caso se utiliza un conjunto de múltiples árboles de decisión que trabajan en paralelo. Cada árbol se entrena con un subconjunto aleatorio de los datos, y al final, las predicciones de todos los árboles se combinan mediante votación (en clasificación) o promedio (en regresión). Esta estrategia reduce la varianza y mejora la estabilidad del modelo en comparación con un único árbol de decisión [17]. A diferencia de los *Bagging Classifiers*, en los que cada árbol tiene acceso a todas las variables para buscar la mejor división en cada nodo, el bosque aleatorio introduce una aleatorización adicional. En cada partición, cada árbol selecciona aleatoriamente un subconjunto de variables, lo que incrementa la diversidad entre árboles

y reduce la correlación entre ellos.

- Regresión logística (*Logistic Regression*). La regresión logística es un modelo utilizado para clasificar datos en dos o más categorías. A diferencia de la regresión lineal, que predice valores continuos, la regresión logística utiliza la función sigmoidea para modelar la probabilidad de pertenencia a una clase. Al establecer un umbral de decisión, convierte estas probabilidades en predicciones de cada clase [69]. Es ampliamente utilizada en problemas de clasificación binaria, como la detección de fraudes o enfermedades.
- Máquinas de vectores de soporte (*Support Vector Machines*). Las Máquinas de vectores de soporte son modelos que buscan encontrar el hiperplano que mejor separa las clases en un espacio de alta dimensión. Utilizan una función de margen óptimo para maximizar la separación entre clases, lo que las hace efectivas en problemas donde los datos no son fácilmente separables de forma lineal [62].

- **Aprendizaje no supervisado**: En este caso, los datos no están etiquetados, y el modelo debe identificar patrones o estructuras inherentes en los datos. El aprendizaje no supervisado se puede modelar como una función:

$$f : X \to Z \tag{2.10}$$

donde:

- $X$ representa el espacio de datos de entrada.
- $Z$ es una representación comprimida o estructurada de los datos.

Esta función busca transformar los datos $X$ en una representación $Z$ que capture variables esenciales, permitiendo reducir la dimensionalidad, encontrar agrupaciones o extraer información latente. Los problemas más comunes en este ámbito son el *clustering* y la reducción de dimensionalidad.

### 2.2.1 Aprendizaje de conjunto

El aprendizaje de conjunto (*ensemble* learning) mejora el rendimiento de los modelos combinando predicciones de múltiples clasificadores en lugar de depender de un solo modelo. Su objetivo es reducir la varianza, el sesgo y mejorar la generalización. La predicción final $\hat{y}$ de un modelo *ensemble* que combina $n$ modelos base $m_i(x)$ puede expresarse como:

$$\hat{y} = f(m_1(x), m_2(x), ..., m_n(x)) \tag{2.11}$$

donde la función $f(\cdot)$ puede representar un promedio ponderado, una votación ma-

yoritaria o un modelo de combinación más sofisticado.

Existen tres enfoques principales para la construcción de modelos *ensemble*:

- *Bagging* (*Bootstrap Aggregating*): se basa en entrenar múltiples modelos independientes sobre subconjuntos aleatorios de los datos y luego combinar sus predicciones. Un ejemplo común es Random Forest.
- *Boosting*: entrena modelos secuenciales donde cada nuevo modelo se centra en corregir los errores cometidos por el anterior. Un ejemplo popular es XGBoost, que mejora la predicción iterando sobre modelos de árboles de decisión con pesos ajustados.
- *Stacking*: combina múltiples modelos utilizando un meta-modelo que aprende a fusionar las predicciones. Se usa en competiciones de ML para mejorar resultados combinando redes neuronales, árboles de decisión y modelos lineales.

El aprendizaje en *ensemble* se ha convertido en una de las técnicas más utilizadas en aplicaciones prácticas, incluyendo detección de fraudes, tarificación en seguros y predicción de riesgos financieros.

### 2.2.2 Métricas de evaluación

En el contexto del ML, las métricas de evaluación son esenciales para medir el desempeño de los modelos y asegurar que cumplen con los objetivos establecidos para una tarea específica [100]. Dependiendo de la naturaleza del problema, ya sea supervisado o no supervisado, las métricas adoptan diferentes formas para capturar con precisión la calidad del modelo.

#### Métricas para aprendizaje supervisado

En los problemas de aprendizaje supervisado, las métricas miden la precisión con la que el modelo predice las etiquetas, $y$, en función de las variables, $X$. Estas métricas varían en función de si el problema es de clasificación o de regresión.

La clasificación busca asignar cada instancia de entrada a una de las clases predefinidas. El procedimiento comúnmente utilizado para medir el desempeño del modelo es comparando las predicciones con los valores reales mediante la matriz de confusión. Para una clasificación binaria, la matriz de confusión se estructura tal y como muestra la tabla 2.1.

| | **Predicción Positiva** | **Predicción Negativa** |
|---|---|---|
| **Clase Positiva** | Verdaderos Positivos (TP) | Falsos Negativos (FN) |
| **Clase Negativa** | Falsos Positivos (FP) | Verdaderos Negativos (TN) |

Tabla 2.1: Matriz de confusión para un problema de clasificación binaria.

Cada celda representa:

- **Verdaderos Positivos** (TP): Casos correctamente clasificados como positivos.
- **Verdaderos Negativos** (TN): Casos correctamente clasificados como negativos.
- **Falsos Positivos** (FP): Casos negativos que el modelo clasificó erróneamente como positivos.
- **Falsos Negativos** (FN): Casos positivos que el modelo clasificó erróneamente como negativos.

A partir de la matriz de confusión, se pueden calcular las siguientes métricas de evaluación:

- **Precisión** (*Accuracy*): Es la fracción de predicciones correctas sobre el total de predicciones realizadas.

$$\text{Precisión} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

- ***Recall***: Es la proporción de instancias de la clase que el modelo ha identificado correctamente.

$$Recall_c = \frac{\text{Número de instancias correctamente clasificadas de la clase } c}{\text{Total de instancias reales de la clase } c} \tag{2.13}$$

  En particular:

  - $Recall_1$ coincide con la Tasa de Verdaderos Positivos (TPR o sensibilidad).Es la proporción de instancias positivas que el modelo ha identificado correctamente.

$$Recall_1 = \text{TPR} = \frac{TP}{TP + FN} \tag{2.14}$$

  - $Recall_0$ coincide con la Tasa de Verdaderos Negativos (TNR o especificidad). Representa la proporción de instancias negativas que el modelo ha identificado correctamente.

$$Recall_0 = \text{TNR} = \frac{TN}{TN + FP} \tag{2.15}$$

- **Tasa de Falsos Positivos** (FPR): Representa la proporción de instancias positivas que el modelo ha identificado erronamente.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2.16}$$

- **F1-Score:** Media armónica de la precisión y *recall*.

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}} \tag{2.17}$$

- **Área Bajo la Curva ROC** (AUC, del inglés *Area Under the Curve*): Representa la capacidad del modelo para distinguir entre clases. La curva ROC ilustra la relación entre TPR y FPR para distintos umbrales de clasificación. El AUC resume su comportamiento general; un valor de 0.5 indica un modelo aleatorio, mientras que un valor de 1.0 corresponde a un modelo perfecto. En la sección 2.5 se aborda en detalle cómo mejorar el rendimiento de los modelos usando el espacio ROC.

$$\text{AUC} = \sum_{i=1}^{n-1} (\text{FPR}_{i+1} - \text{FPR}_i) \cdot \frac{\text{TPR}_{i+1} + \text{TPR}_i}{2} \tag{2.18}$$

  donde $n$ es el número total de puntos evaluados sobre la curva ROC (correspondientes a distintos umbrales de clasificación), e $i$ es el índice que recorre los puntos de la curva ROC ordenados de menor a mayor FPR.

La regresión busca predecir valores continuos a partir de las variables de entrada. Las métricas de evaluación más utilizadas en este tipo de problemas son el error cuadrático medio (MSE), el error absoluto medio (MAE) y el coeficiente de determinación $R^2$. Estas métricas permiten cuantificar la precisión del modelo al estimar los valores reales, evaluando la magnitud y la dirección de los errores cometidos.

- **Error Cuadrático Medio** (MSE, del inglés *Mean Squared Error*). Mide el promedio de los errores al cuadrado entre los valores reales $y_i$ y las predicciones del modelo $\hat{y}_i$. Al elevar al cuadrado las diferencias, se penalizan de forma más severa los errores grandes, lo que hace que el MSE sea especialmente útil para evitar desviaciones significativas en las predicciones. Valores bajos de MSE indican un mejor ajuste del modelo sobre los datos observados.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.19}$$

  donde $n$ es el número total de observaciones, $y_i$ es el valor real y $\hat{y}_i$ es la predicción.

- **Raíz del Error Cuadrático Medio** (RMSE, del inglés *Root Mean Squared Error*). Representa la raíz cuadrada del MSE y se interpreta en las mismas unidades que la variable de salida. Al igual que el MSE, penaliza más los errores grandes, pero su escala facilita la interpretación directa del error promedio. Valores bajos de RMSE indican un mejor rendimiento del modelo.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{2.20}$$

- **Error Absoluto Medio** (MAE, del inglés *Mean Absolute Error*). Mide el promedio de los errores absolutos entre los valores reales y las predicciones del modelo. A

diferencia del MSE, todos los errores se ponderan de forma lineal.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.21}$$

- **Coeficiente de determinación,** $R^2$. Mide la proporción de la variabilidad de la variable dependiente que es explicada por el modelo a partir de las variables independientes. Un valor de $R^2 = 1$ indica que el modelo explica perfectamente los datos, mientras que un valor de $R^2 = 0$ implica que el modelo no aporta ninguna mejora respecto a predecir simplemente la media de los valores observados. Cuanto más cercano a 1 sea el valor de $R^2$, mayor será la capacidad explicativa del modelo.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.22}$$

donde $\bar{y}$ es la media de los valores reales.

### Métricas para aprendizaje no supervisado

En problemas no supervisados las métricas evalúan las características aprendidas por el modelo, como la agrupación en clústeres (*clustering*) o la reducción de dimensionalidad.

En *clustering*, el objetivo es agrupar los datos en clústeres (grupos) de forma que los elementos dentro de un mismo clúster sean lo más parecidos entre sí y lo mas diferentes al resto de clústeres. Las métricas más utilizadas son:

- **Índice de Silueta**. Evalúa la calidad de la agrupación midiendo, para cada punto, el grado de adecuación de su asignación al clúster correspondiente. Tiene en cuenta tanto la cohesión (la cercanía a los puntos de su mismo clúster) como la separación (la distancia respecto a los clústeres más cercanos).

$$S(i) = \frac{b(i) - a(i)}{\text{máx}(a(i), b(i))} \tag{2.23}$$

donde $a(i)$ es la distancia promedio entre el punto $i$ y otros puntos de su clúster, y $b(i)$ es la distancia promedio entre el punto $i$ y los puntos del clúster más cercano.

- **Coeficiente de Silueta**. Se obtiene calculando el promedio del índice de silueta de todos los puntos. Este valor resume la calidad del agrupamiento de forma general. Un valor cercano a 1 indica que, en promedio, los puntos están bien agrupados. En cambio, valores cercanos a 0 o negativos reflejan una mala calidad en la asignación de los clústeres.

$$S = \frac{1}{n}\sum_{i=1}^{n}S(i) \tag{2.24}$$

donde $n$ es el número total de observaciones y $S(i)$, el índice de silueta del punto $i$, que viene dado por la ecuación 2.23.

- **Inercia.** Mide la compacidad de los clústeres, es decir, la cercanía promedio entre los puntos de un mismo clúster.

$$\text{Inercia} = \sum_{i=1}^{n} \left\| x_i - \mu_{c(i)} \right\|^2 \tag{2.25}$$

donde $n$ es el número total de instancias de datos, $x_i$ representa el vector de variables del punto $i$ y $\mu_{c(i)}$ es el centroide del clúster al que pertenece el punto $x_i$.

En reducción de dimensionalidad, el objetivo es proyectar los datos en un espacio de menor dimensión manteniendo, en la medida de lo posible, su estructura original. Las métricas más utilizadas para evaluar la calidad de esta representación son:

- **Preservación de vecinos.** Comprueba si los puntos que estaban cerca unos de otros antes de la reducción siguen estando cerca después. Si esto se cumple, significa que la estructura se ha mantenido correctamente. Se calcula como la proporción de vecinos comunes entre el espacio original y el reducido.

$$P(k) = \frac{1}{n} \sum_{i=1}^{n} \frac{|N_k^{\text{orig}}(i) \cap N_k^{\text{red}}(i)|}{k} \tag{2.26}$$

donde $n$ es el número total de puntos, $k$ es el número de vecinos considerados, $N_k^{\text{orig}}(i)$ representa el conjunto de los $k$ vecinos más cercanos al punto $i$ en el espacio original, y $N_k^{\text{red}}(i)$ es el conjunto de los $k$ vecinos más cercanos al mismo punto en el espacio reducido.

- **Error de reconstrucción.** Mide el error al reconstruir el dato original a partir de su versión reducida. Cuanto menor sea este error, mejor se habrá conservado la información tras la reducción de dimensionalidad.

$$\text{Error} = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \tag{2.27}$$

donde $x_i$ es el dato original y $\hat{x}_i$ su reconstrucción.

## 2.3 Aprendizaje automático en la industria del seguro

Los beneficios de la industria del seguro de automóviles dependen principalmente de los beneficios captados a partir de la prima del seguro. Con la competencia entre compañías aseguradoras, la prima está sujeta a restricciones impuestas por el mercado. Por lo tanto, se debe buscar un equilibrio al fijar el valor de la prima.

Existen pocos estudios sobre la aplicación de algoritmos de ML en la industria del seguro de automóviles. Además, bastantes artículos no detallan cómo abordan el problema del desbalanceo de datos, donde una clase presenta un número significativamente mayor de casos en comparación con la otra y no muestran métricas como la precisión y el *recall* de cada clase (en los casos de modelos de clasificación). Esto crea la incertidumbre de si el conjunto de datos desbalanceados sesga la precisión del modelo de ML. Este tipo de problema se aborda más adelante en la sección 2.4.

Para que el mercado del seguro sea rentable, los accidentes deben ser una pequeña proporción de las primas contratadas. Por esta razón, es necesario aplicar una etapa rigurosa de preprocesamiento con técnicas para el tratamiento de datos desbalanceados, antes de iniciar la etapa de modelado de ML.

Guelman [49] aplica un modelo basado en Gradient Boosting para predecir el coste de accidentes automovilísticos utilizando datos de una importante aseguradora canadiense. Una de las principales preocupaciones en este artículo es el desbalanceo de datos, ya que hay muchos más casos sin siniestros que con siniestros. Al estructurar los datos, cada fila del conjunto de datos recoge un período durante el cual un vehículo estuvo expuesto al riesgo de un accidente como culpable. Después del preprocesamiento de datos, generan modelos Gradient Boosting para calcular la frecuencia y severidad de los siniestros. Con los resultados obtenidos para estas métricas, los autores aplican la fórmula para calcular el coste de las pérdidas y la comparan con los valores obtenidos por los modelos GLM. Tras analizar los resultados, recomiendan usar Gradient Boosting como método alternativo a los modelos GLM para construir modelos de coste de pérdidas de seguros.

Liu et al. [78] comparan AdaBoost multiclase, GLM, redes neuronales y algoritmos de máquina de soporte vectorial (SVM, del inglés *Support Vector Machine*) aplicados a un problema de predicción en seguros de automóviles. El conjunto de datos recoge siniestros por daños a la propiedad de terceros de una compañía de seguros en Malasia. Este conjunto de datos se divide en tres clases según el número de siniestros, siendo la clase sin siniestros la mayoría por una diferencia significativa. Los autores no muestran la precisión de los modelos para cada clase objetivo, creando la incertidumbre de si el resultado está sesgado por el desbalanceo de datos.

En Blier-Wong et al. [18] proporcionan una revisión exhaustiva sobre la aplicación de técnicas de ML en el sector del seguro de propiedad y accidentes, con un enfoque particular en la fijación de precios y la reserva (fondos destinados a pagar siniestros pendientes). Se destaca en el artículo cómo los modelos de DL y ML, como las redes neuronales y los algoritmos de *Boosting*, están transformando las prácticas tradicionales de los actuarios, superando las limitaciones de los modelos GLM. Estos modelos permiten una mayor precisión al capturar relaciones no lineales e interacciones más complejas entre las variables, lo que mejora las predicciones del coste de siniestros y cálculo de primas.

También discuten los desafíos relacionados con la interpretabilidad de los modelos complejos, un aspecto crucial para su adopción en una industria tan regulada. Los autores

sugieren que, aunque las redes neuronales ofrecen un rendimiento superior, su "caja negra" plantea problemas de transparencia y explicabilidad, lo cual es esencial para cumplir con las normativas y mantener la confianza del cliente. Para mitigar esto, proponen métodos como la integración de GLMs con redes neuronales, lo que permite mantener la interpretabilidad de los primeros mientras se aprovechan las capacidades predictivas de los segundos. Además, el artículo subraya la importancia de seguir investigando en áreas como la generación de datos sintéticos y la evaluación ética de los modelos predictivos en seguros.

En Roy and George [105] abordan el problema del fraude en el sector de seguros, señalando que representa pérdidas millonarias anuales para la industria. Se diferencian dos tipos principales de fraude: el fraude duro, donde se fabrican siniestros inexistentes, y el fraude blando, que implica la exageración de siniestros legítimos. Tradicionalmente, la detección de fraudes se ha realizado mediante un proceso manual y basado en reglas, lo que resulta ineficiente ante la creciente complejidad y volumen de datos.

Para abordar estos desafíos, los autores proponen la utilización de técnicas de ML, que permitan automatizar el proceso de detección y mejorar la precisión y rapidez de las decisiones. Para ello, construyen modelos predictivos utilizando datos de siniestros, que incluyen detalles como la fecha de ocurrencia del siniestro, la fecha de reporte, el importe reclamado y otros atributos relevantes. Estos datos se transforman y alimentan a los algoritmos ML, y posteriormente son evaluados mediante matrices de confusión.

Los autores concluyen que, entre los algoritmos evaluados, los árboles de decisión y Random Forest ofrecen un rendimiento superior en la detección de fraudes comparados con Naïve Bayes. Sin embargo, sugieren que futuras investigaciones podrían explorar otros algoritmos o combinaciones de técnicas para mejorar aún más la efectividad de los modelos predictivos.

SISBAR [36] es un sistema novedoso de detección de fraude basado en blockchain con permisos y algoritmos de ML para compañías de seguros. Los autores utilizan XGBoost [27] y Very Fast Decision Tree (VFDT) [35] para detectar y clasificar diferentes tipos de siniestros fraudulentos de seguros de automóviles y medir el nivel de riesgo del cliente. Demuestran la eficacia de XGBoost en la predicción del comportamiento futuro de los clientes y en la estimación del número de siniestros. Al identificar estos fraudes, se consigue un ahorro significativo para los clientes responsables, quienes no se verán afectados por penalizaciones injustas.

En [106] examinan la aplicación de técnicas de ML para la predicción de riesgos en seguros, con un enfoque en el uso de clasificadores basados en árboles como Decision Tree, Random Forest y XGBoost. Los autores comparan la efectividad de estos modelos en la clasificación de riesgos de seguros, utilizando datos históricos de la industria. XGBoost destaca como el modelo más preciso, obteniendo un AUC de 0.86 y un F1-score superior a 0.56. Además, el estudio aborda la interpretabilidad de los modelos a través de herramientas como SHAP, facilitando una comprensión más detallada de las decisiones tomadas por

el modelo y proporcionando explicaciones claras sobre la contribución de cada variable en las predicciones. Este enfoque es crucial para garantizar la transparencia y la confianza en las aplicaciones de ML dentro del sector asegurador.

## 2.4 Técnicas para el tratamiento de datos desbalanceados

Un problema que surge al utilizar datos masivos en los modelos de ML es el uso de conjuntos de datos incompletos, desbalanceados o sesgados que, si no son tratados adecuadamente, pueden conducir a una inferencia errónea del modelo.

El desbalanceo de datos se produce cuando un conjunto de datos contiene más muestras de una clase (la clase mayoritaria) en la distribución de la variable objetivo, que de las otras clases (las clases minoritarias). Este desbalanceo puede generar un aprendizaje desequilibrado, en el que el modelo tiende a favorecer a la clase mayoritaria, ignorando las clases minoritarias, lo que da lugar a predicciones erróneas y métricas de evaluación poco representativas.

El uso de datos desbalanceados puede plantear cuatro problemas principales [52]:

- La mayoría de los algoritmos de ML asumen o esperan una distribución balanceada de la variable objetivo. Por lo tanto, los algoritmos proporcionan predicciones imprecisas para la clase objetivo si existe un gran desequilibrio entre clases.
- El uso de métricas de rendimiento globales, como la precisión de la predicción, induce en un sesgo hacia la clase mayoritaria.
- El algoritmo de ML puede considerar las muestras de la clase minoritaria como ruido.
- Dificultad para detectar patrones en la clase minoritaria.

Los conjuntos de datos desbalanceados son comunes en aplicaciones de IA. Por ejemplo, en problemas de clasificación en el ámbito médico como la detección de proteínas [10], la predicción de la expresión génica [130] o el diagnóstico de enfermedades [39, 70]. La falta de precisión en la predicción de estos diagnósticos puede causar problemas graves para la salud del paciente, por lo que es crucial prestar especial atención a la clase minoritaria, que precisamente es la clase crítica.

Otra área donde este problema aparece es en el sector financiero, especialmente en la detección del fraude financiero [2, 3, 22] y de seguros [57, 58]. La detección de fraudes es un problema típico de datos desbalanceados, ya que los casos de fraude son la minoría. Es crucial identificar estos casos, ya que representan pérdidas significativas para las empresas. Además de lo mencionado anteriormente, la preocupación por los datos desbalanceados también afecta a la gestión de energía [51, 75, 133], la tecnología de la información [1,

125, 127] y la gestión de la seguridad [9, 13], entre otros. Por lo tanto, es un problema que debe abordarse para diseñar modelos fiables.

En la literatura se han desarrollado diversas técnicas para abordar el problema de los datos desbalanceados [37]. No obstante, solo un conjunto limitado de estas técnicas, como las estrategias basadas en *Underbagging* u *Overbagging* (explicadas en la sección 2.4.3), obtienen resultados óptimos en escenarios con datos extremadamente desbalanceados, aunque, en general, a costa de tiempos computacionales elevados.

Por otro lado, existen métodos que, si bien reducen el desbalanceo, tienden a generar modelos sobreajustados, lo que compromete la capacidad de generalización del modelo de ML frente a nuevos datos. Estas limitaciones resaltan la importancia de seleccionar cuidadosamente las estrategias de preprocesamiento y modelado en función de las características específicas del conjunto de datos.

### 2.4.1 Técnicas de sobremuestreo

Las técnicas de sobremuestreo se centran principalmente en generar o replicar muestras sintéticas de la clase minoritaria para reducir la diferencia en el número de muestras.

Una técnica comúnmente utilizada es la técnica de sobremuestreo sintético de la clase minoritaria, (SMOTE, del inglés *Synthetic Minority Over-sampling Technique*) [25] que está inspirada en una técnica muy exitosa basada en el reconocimiento de caracteres manuscritos [50].

SMOTE se enfoca en la generación de datos sintéticos al interpolar instancias en el espacio de variables, en lugar de simplemente replicar datos en el espacio de datos original. El algoritmo selecciona muestras aleatorias de la clase minoritaria y genera nuevos puntos sintéticos a lo largo de los segmentos que conectan estas muestras con cualquiera o con todos sus $k$ vecinos más cercanos dentro de la misma clase (figura 2.1).

El procedimiento consiste en seleccionar una muestra de la clase minoritaria y calcular la diferencia entre su vector de variables y el de uno de sus vecinos más cercanos perteneciente a la misma clase. Esta diferencia se multiplica por un número aleatorio entre 0 y 1 y se agrega al vector de la clase minoritaria. Este proceso genera un punto aleatorio entre las dos muestras de clases minoritarias seleccionadas con características similares a ambas muestras.

SMOTE-Encoded Nominal and Continuous (SMOTE-ENC) [91] es una variante de SMOTE diseñada específicamente para manejar conjuntos de datos que contienen tanto variables numéricas como categóricas. SMOTE-ENC emplea un método de interpolación para las variables continuas y otro diferente para las variables categóricas.

SMOTETomek [11] combina Tomek-Links (figura 2.2) y SMOTE (figura 2.1). Tomek-Links [117] es una técnica de submuestreo que mejora la regla del vecino más cercano. Detecta aquellas muestras de la clase mayoritaria con la menor distancia euclidiana a los

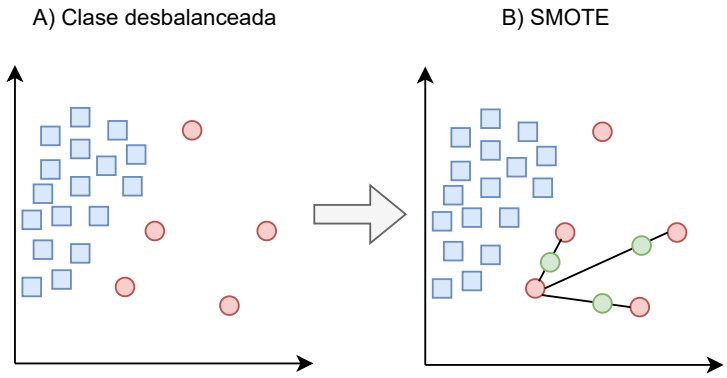



Figura 2.1: Ejemplo del funcionamiento de SMOTE, donde se generan nuevas muestras sintéticas (en verde) a partir de interpolaciones entre muestras reales de la clase minoritaria (en rojo).

datos de la clase minoritaria y las elimina. Luego, se aplica SMOTE a este conjunto de datos.

La combinación de SMOTE y Tomek-Links (figuras 2.2 y 2.3) permite la generación de datos sintéticos de la clase minoritaria y, simultáneamente, elimina aquellas muestras de la clase mayoritaria más cercanas a la clase minoritaria.

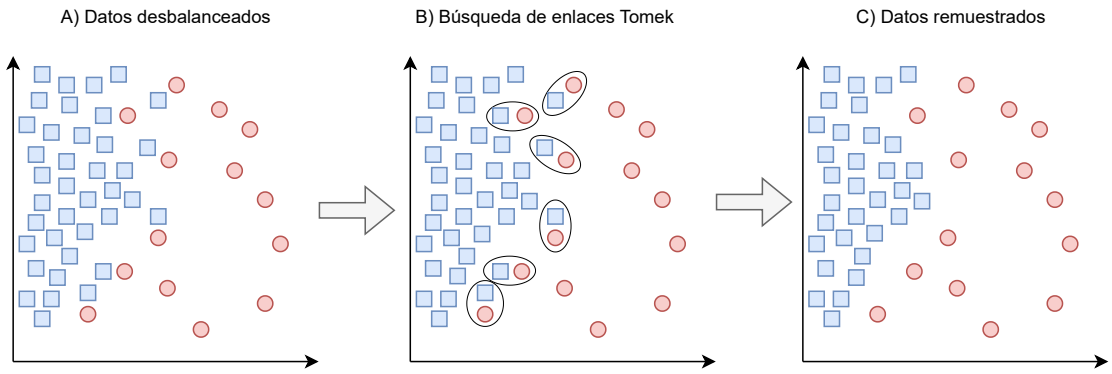



Figura 2.2: Ilustración del proceso de eliminación de Tomek-Links. Primero se detectan los pares de muestras cercanas de clases opuestas (B), y posteriormente se eliminan las muestras de la clase mayoritaria involucradas (C) para limpiar la frontera de decisión.

Borderline SMOTE [53] es otra variante de SMOTE que se centra en identificar muestras de la clase minoritaria cuyos vecinos son muestras de la clase mayoritaria (figura 2.4). Una vez identificadas estas muestras, se consideran como cercanas a la frontera entre clases y se generan datos sintéticos utilizando únicamente estas instancias, reforzando así la zona de decisión.

Mixup [135] es una técnica de sobremuestreo que actúa como regularizador, suavizando los límites de decisión entre clases. Genera ejemplos sintéticos interpolando dos muestras

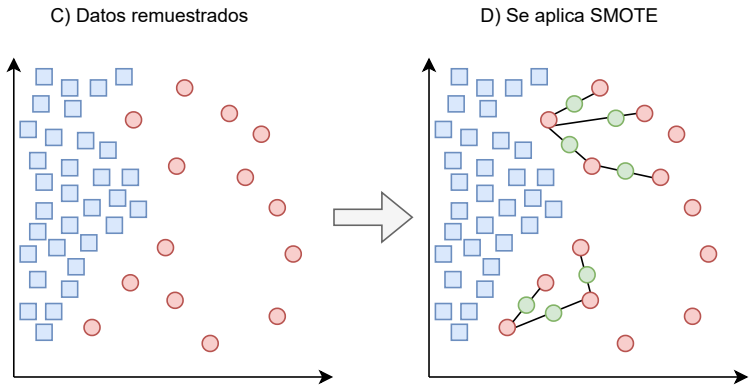



Figura 2.3: SMOTETomek. A partir de la eliminación con Tomek-Links (C), se generan muestras sintéticas (en verde) con SMOTE (D), mejorando el balance y la separación entre clases.

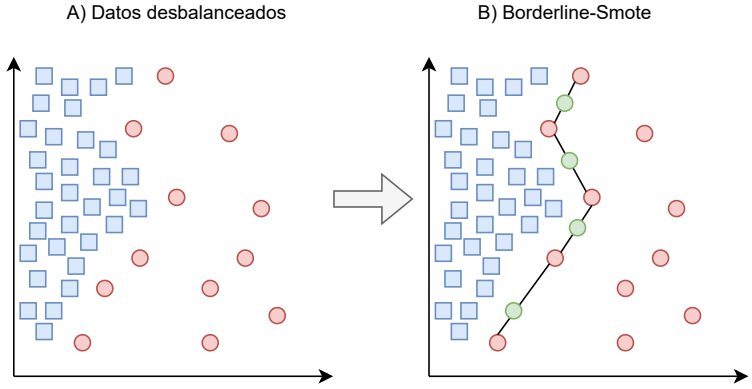



Figura 2.4: Funcionamiento de Borderline-SMOTE, una variante de SMOTE que genera nuevas muestras sintéticas únicamente en la zona froteriza entre clases, reforzando las regiones más conflictivas del espacio de decisión. Los casos de la clase minoritaria (en rojo) que se encuentran próximos a muestras de la clase mayoritaria se identifican como puntos fronterizos (representados mediante líneas que los conectan). A partir de ellos, se generan nuevas muestras sintéticas (en verde).

elegidas al azar, tanto en sus variables como en sus etiquetas. En el caso de las variables continuas, crea combinaciones convexas de ambas muestras, usando un peso aleatorio tomado de una distribución Beta. Para las etiquetas, también se realiza una combinación proporcional, dando lugar a valores intermedios que reflejan una mezcla entre las clases originales.

Adaptive Synthetic (ADASYN) [60] es otra técnica basada en el algoritmo SMOTE. La diferencia clave entre ADASYN y SMOTE es que ADASYN detecta aquellas muestras de la clase minoritaria en áreas dominadas por la clase mayoritaria y genera muestras sintéticas de la clase minoritaria en esta área (figura 2.5). ADASYN aplica una distribución ponderada a las diferentes muestras de la clase minoritaria en función de la dificultad de

aprendizaje en cada área (más vecinos de la clase mayoritaria, más dificultad). De esta manera, genera más datos sintéticos en regiones con mayor complejidad para el modelo. Cabe señalar que, Borderline SMOTE no genera datos sintéticos en las muestras más cercanas a la clase mayoritaria, mientras que ADASYN sí lo hace.

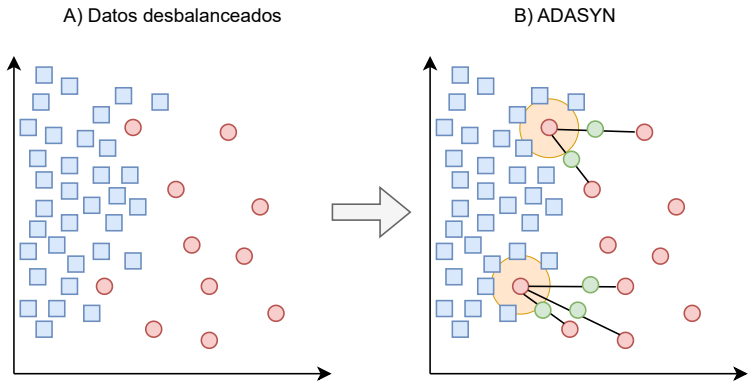



Figura 2.5: Representación de ADASYN, una técnica que genera muestras sintéticas en función de la dificultad de clasificación local, concentrando las nuevas muestras en zonas donde la clase minoritaria está menos representada.

Otra técnica de sobremuestreo es Random OverSampler [85], que genera nuevas muestras de la clase minoritaria replicando las existentes. Una de las principales preocupaciones al emplear este método es el riesgo de sobreentrenamiento, ya que la duplicación excesiva de casos sesgados o anómalos puede distorsionar la representación de la clase minoritaria. Esto podría generar distribuciones poco realistas, afectando a la capacidad del modelo de ML para generalizar adecuadamente, reduciendo su rendimiento frente a nuevos datos.

### 2.4.2 Técnicas de submuestreo

Por otro lado, las técnicas de submuestreo se utilizan para reducir el número de muestras de las clases mayoritarias. Entre ellas se encuentran Tomek Links y Condensed Nearest Neighbors (C-NN)[1]. Son técnicas que combinan la idea de submuestreo con estrategias de edición de instancias para mejorar la calidad del conjunto de datos antes de entrenar un modelo. Tomek Links identifica pares de instancias de clases opuestas que son mutuamente las más cercanas. Cuando se detecta uno de estos pares, se elimina la instancia perteneciente a la clase mayoritaria, con el objetivo de limpiar la frontera de decisión entre clases. Por su parte, CNN reduce la clase mayoritaria manteniendo solo las instancias que son necesarias para clasificar correctamente la clase minoritaria, eliminando las instancias redundantes. Estas técnicas no solo ayudan a equilibrar la distribución de clases, sino que también mejoran la separación entre las clases, lo que puede dar como resultado clasificadores más precisos [12, 55] .

[1]No confundir C-NN con redes neuronales convolucionales.

Otra técnica comúnmente utilizada es el submuestreo aleatorio (RUS, del inglés *Random Undersampling*). Es una de las técnicas más simples y directas para manejar datos desbalanceados. Esta técnica reduce la cantidad de instancias en la clase mayoritaria al seleccionar aleatoriamente un subconjunto de sus muestras. El objetivo es igualar el número de instancias de ambas clases, creando un conjunto de datos balanceado para el entrenamiento. Sin embargo, esta simplicidad tiene un coste: la eliminación aleatoria de datos de la clase mayoritaria puede llevar a la pérdida de información valiosa, lo que podría degradar la capacidad predictiva del modelo. Aunque es una técnica ampliamente utilizada, su principal limitación es que puede omitir patrones importantes presentes en la clase mayoritaria [66].

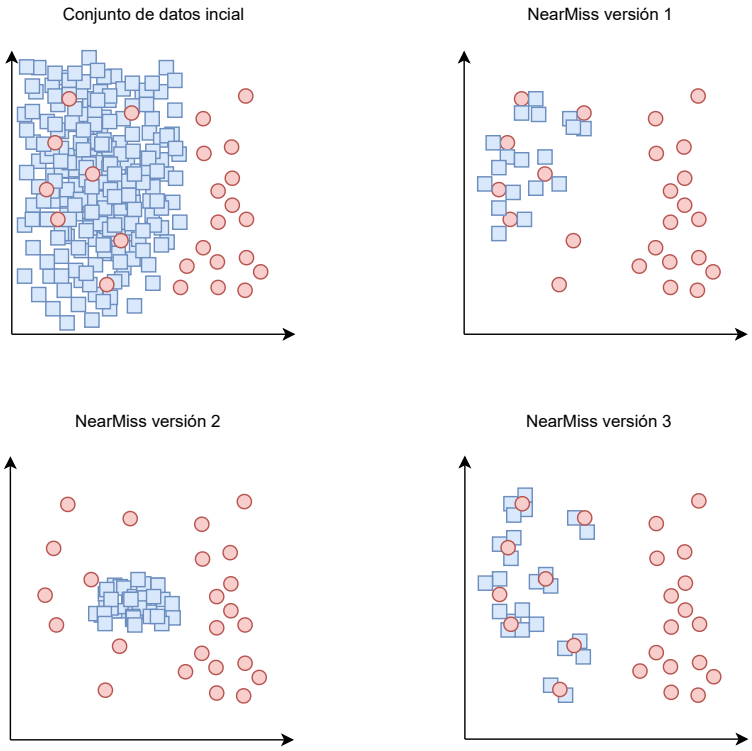



Figura 2.6: Ejemplo de la técnica NearMiss, que aplica submuestreo sobre la clase mayoritaria seleccionando diferentes subconjuntos según criterios de proximidad: versión 1 (más cercanos a la clase minoritaria), versión 2 (con menor distancia global) y versión 3 (balanceando la distribución).

Otra técnica de submuestreo es NearMiss [81]. Esta técnica selecciona instancias de la clase mayoritaria en función de su distancia a la clase minoritaria (figura 2.6). Hay varias variantes de NearMiss, cada una con un criterio de selección diferente:

- NearMiss-1 selecciona muestras de la clase mayoritaria con la menor distancia promedio a las tres muestras más cercanas de la clase minoritaria.
- NearMiss-2 selecciona muestras de la clase mayoritaria con la menor distancia pro-

medio a las tres muestras más lejanas de la clase minoritaria.

- NearMiss-3 implica seleccionar un número dado de muestras de la clase mayoritaria para cada muestra de la clase minoritaria más cercana.

Esta técnica puede ser útil en situaciones donde es crucial preservar la frontera de decisión entre las clases para mejorar la precisión del modelo.

### 2.4.3 Técnicas híbridas

Este tipo de técnicas utilizan diferentes métodos de tratamiento de datos desbalanceados. Entre ellas destacan las siguientes.

SMOTE-Edited Nearest Neighbors (SMOTE-ENN) [128] combina sobremuestreo y el submuestreo. SMOTE-ENN emplea SMOTE para generar nuevas muestras sintéticas de la clase minoritaria y, posteriormente, utiliza ENN para eliminar instancias ruidosas o ambiguas del conjunto de datos. Para cada instancia en el conjunto de datos, ENN identifica sus $k$ vecinos más cercanos y, si la mayoría de estos vecinos pertenece a una clase diferente a la de la instancia en cuestión, ésta es eliminada del conjunto de datos. El objetivo de ENN es eliminar instancias que son clasificadas erróneamente por sus vecinos, funcionando como un método de limpieza del ruido en el conjunto de datos.

SMOTE Edited Nearest Neighbour Mixup (STEM) [56] combina SMOTE-ENN y Mixup. En primer lugar, se ejecuta SMOTE-ENN para equilibrar las clases y eliminar el ruido del conjunto de datos. A continuación, se aplica Mixup con el objetivo de aumentar la diversidad de los datos y mejorar la capacidad de generalización del modelo.

RUSBoost [110] combina el submuestreo aleatorio RUS con el algoritmo de *Boosting* AdaBoost. RUS reduce el tamaño de la clase mayoritaria seleccionando aleatoriamente un subconjunto de sus muestras. Por su parte, AdaBoost combina múltiples clasificadores débiles (es decir, árboles de decisión poco profundos) para crear un clasificador fuerte. RUSBoost genera subconjuntos equilibrados mediante submuestreo aleatorio en cada iteración y, a continuación, entrena un clasificador débil para cada subconjunto. Posteriormente, se calcula el peso del clasificador en función de su rendimiento y se actualizan los pesos de las instancias, dando mayor énfasis a aquellas que fueron clasificadas erróneamente. Finalmente, se construye un clasificador fuerte combinando las predicciones de los clasificadores débiles de manera ponderada.

SMOTEBoost [26] es similar a RUSBoost, pero en lugar de aplicar submuestreo aleatorio, emplea SMOTE para generar cada uno de los subconjuntos utilizados por los clasificadores débiles.

En Jeatrakul et al. [67], se propone una combinación de técnicas de submuestreo utilizando redes neuronales complementarias y sobremuestreo. El autor utiliza tres tipos de algoritmos de ML: máquina de vectores de soporte, k-vecinos más cercanos y redes neuronales complementarias. Estas últimas se combinan con SMOTE, utilizando la media geométri-

ca de las clases y el AUC como métricas de evaluación. Los resultados muestran cómo la combinación de esta red neuronal artificial con SMOTE obtiene mejores resultados que solo aplicando técnicas de sobremuestreo y submuestreo.

### 2.4.4 Técnicas de *Bagging*

Una variante a las técnicas de sobremuestreo y submuestreo son las técnicas basadas en *Bagging* (técnicas *ensemble* que mejoran la estabilidad y precisión de los algoritmos ML al reducir la varianza) y preprocesamiento de datos.

El uso de *Bagging* es especialmente eficaz cuando se combina con algoritmos que tienen alta varianza, como los árboles de decisión. La agregación de múltiples modelos reduce la tendencia de los modelos individuales a sobreajustarse a los datos de entrenamiento, dando como resultado un modelo más generalizable. La clave de estos métodos está en la forma en que se generan las réplicas *bootstrap*, con un enfoque especial en manejar el desequilibrio de clases. Esto garantiza clasificadores eficientes en cada iteración, priorizando la diversidad.

Dentro de este tipo de técnicas, diferenciamos dos:

- *Underbagging*. Esta técnica combina *Bagging* con submuestreo. Consiste en construir diferentes subconjuntos para modelos de ML utilizando técnicas de submuestreo. Generalmente, el procedimiento de submuestreo se aplica solo a la clase mayoritaria; sin embargo, también se puede aplicar el remuestreo con reemplazo de la clase minoritaria para obtener conjuntos más diversos. Cabe destacar que el submuestreo es más probable que ignore algunos casos útiles de la clase mayoritaria ya que en cada subconjunto se excluyen muestras de esta. Por eso es importante crear un número de subconjuntos representativo, y así obtener una mejor representación de la población. *Underbagging* ha demostrado ser eficaz, especialmente cuando se combina con algoritmos de clasificación basados en árboles o K-NN, ya que permite mejorar la detección de la clase minoritaria sin necesidad de modificar los algoritmos base [121].
- *Overbagging*. Esta técnica combina *Bagging* con sobremuestreo. Su objetivo es mejorar la detección de la clase minoritaria generando subconjuntos de entrenamiento más equilibrados para los modelos base. En lugar de reducir la clase mayoritaria (como en *Underbagging*), *Overbagging* refuerza la presencia de la clase minoritaria en cada subconjunto aplicando técnicas de sobremuestreo [122].

Ambas técnicas presentan ventajas significativas, ya que mejoran la capacidad de los clasificadores para gestionar la variabilidad de los datos. Sin embargo, también tienen desventajas. Por ejemplo, *Underbagging* puede conllevar la pérdida de información representativa de la población al reducir la clase mayoritaria durante el entrenamiento, mientras que *Overbagging* puede llevar al sobreajuste si se emplean técnicas de sobremuestreo demasiado agresivas.

Para mitigar estos problemas y mejorar la eficacia del aprendizaje en entornos desbalanceados, se han desarrollado enfoques más avanzados basados en modelos *ensemble*. EasyEnsemble [77] crea múltiples subconjuntos equilibrados mediante la aplicación de RUS y entrena un clasificador Adaboost en cada uno de ellos. Finalmente, se emplea un esquema de votación para combinar las predicciones y obtener la clasificación final.

BalanceCascade [77] combina *Bagging* y *Boosting*, utilizando específicamente AdaBoost, junto con un proceso supervisado de submuestreo para abordar el problema del desbalanceo de clases. Entrena un clasificador en cada subconjunto equilibrado de datos y, posteriormente, elimina secuencialmente las instancias de la clase mayoritaria que han sido clasificadas correctamente con alta confianza. Este proceso se repite para cada bolsa (subconjunto) de datos, enfocándose progresivamente en las instancias más difíciles de clasificar. Finalmente, las predicciones de todos los clasificadores se combinan mediante votación para obtener la clasificación final.

Otro tipo de modelos *ensemble* es Self-paced Ensemble (SPE) [79]. A diferencia de los métodos tradicionales, SPE se centra en la "dificultad de clasificación" de cada muestra, utilizándola para guiar un submuestreo autorregulado durante el entrenamiento. Este proceso iterativo divide las muestras de la clase mayoritaria en grupos según su dificultad y, progresivamente, reduce la importancia de las muestras triviales para enfocarse en las más informativas. SPE construye un *ensemble* de clasificadores, cada uno entrenado con subconjuntos de datos ajustados según la dificultad de clasificación. Este enfoque iterativo, que no requiere el cálculo de distancias entre muestras, mejora la eficiencia computacional y la escalabilidad, lo que lo hace especialmente adecuado para conjuntos de datos de gran tamaño.

La tabla 2.2 presenta un resumen de las técnicas mencionadas para el tratamiento de datos desbalanceados, detallando su tipo, una breve descripción, así como sus principales ventajas y limitaciones.

| **Técnica** | **Tipo** | **Descripción** | **Ventajas** | **Limitaciones** |
|---|---|---|---|---|
| SMOTE | Sobremuestreo | Interpolación entre vecinos cercanos de la clase minoritaria | Prevención de sobreajuste; mejora la generalización | Puede generar ruido si las clases están solapadas |
| SMOTE-ENC | Sobremuestreo | Manejo de variables numéricas y categóricas | Apropiado para datos heterogéneos | Requiere codificación cuidadosa |
| Borderline-SMOTE | Sobremuestreo | Genera muestras en la frontera entre clases | Refuerza zonas difíciles de separar | Ignora muestras alejadas de la frontera de decisión |
| ADASYN | Sobremuestreo | Genera más muestras en zonas difíciles | Se focaliza en regiones complejas | Puede aumentar el ruido si no se controla |
| Mixup | Sobremuestreo | Interpolación de muestras y etiquetas | Suaviza la frontera de decisión | Pierde interpretabilidad individual |
| SMOTETomek | Híbrido | SMOTE + eliminación con Tomek Links | Mejora la separación y balance entre clases | Puede eliminar muestras útiles |
| Random Oversampling | Sobremuestreo | Duplica de forma aleatoria muestras minoritarias | Fácil de aplicar | Riesgo de sobreajuste |
| Random Undersampling (RUS) | Submuestreo | Reducción aleatoria de la clase mayoritaria | Simple y rápida | Pierde información valiosa |
| Tomek Links | Submuestreo | Eliminación de muestras frontera de la clase mayoritaria | Limpia la frontera de decisión | Puede eliminar muestras con información importante |
| Condensed Nearest Neighbors (C-NN) | Submuestreo | Conserva solo muestras esenciales | Reduce redundancia, mejora separación | Puede reducir diversidad |
| NearMiss-1 | Submuestreo | Selecciona muestras de la clase mayoritaria más cercanos a las instancias minoritarias, reforzando los casos frontera | Refuerza la separación entre clases | Sensible al número de vecinos utilizados |
| NearMiss-2 | Submuestreo | Selecciona muestras mayoritarios más alejados de las instancias minoritarias, priorizando zonas más difíciles | Mantiene diversidad en la clase mayoritaria | Cálculo global de distancias costoso |
| NearMiss-3 | Submuestreo | Selecciona, para cada instancia minoritaria, un número fijo de vecinos de la clase mayoritaria | Equilibra la clase mayoritaria en torno a cada instancia minoritaria | Alta carga computacional y menor escalabilidad |
| SMOTE + ENN | Híbrido | Sobremuestreo con limpieza de ruido por vecinos | Reduce ruido y ambigüedad | Depende del número de vecinos |
| STEM | Híbrido | SMOTE-ENN seguido de Mixup | Alta diversidad y robustez | Mayor complejidad computacional |
| SMOTEBoost | Híbrido Ensemble | SMOTE integrado con AdaBoost | Aprovecha *ensembles* con datos balanceados | Coste computacional elevado |
| RUSBoost | Híbrido Ensemble | RUS + AdaBoost | Combina submuestreo y aprendizaje ponderado | Puede perder patrones útiles |
| SMOTE + Redes neuronales complementarias | Híbrido | Combina redes neuronales y SMOTE | Precisión superior | Difícil de interpretar |
| Underbagging | Híbrido Ensemble | Bagging + submuestreo | Mejora representación con alta varianza | Puede eliminar ejemplos clave |
| Overbagging | Híbrido Ensemble | Bagging + sobremuestreo | Mayor presencia minoritaria en subconjuntos | Riesgo de sobreajuste |
| EasyEnsemble | Híbrido Ensemble | RUS + AdaBoost y esquema de votación | Gran robustez; mejora separación | Coste computacional moderado |
| BalanceCascade | Híbrido Ensemble | AdaBoost + submuestreo supervisado iterativo | "Descarta instancias correctamente clasificadas | Proceso secuencial costoso |
| Self-paced Ensemble (SPE) | Híbrido Ensemble | Submuestreo guiado por dificultad | Escalable y eficiente | Más complejo de implementar |

Tabla 2.2: Resumen de técnicas para el tratamiento de datos desbalanceados

## 2.5 El espacio ROC

Mientras que las métricas convencionales como la precisión, la sensibilidad o el F1-score son comúnmente empleadas para evaluar los modelos, a menudo ofrecen una vista limitada de su rendimiento.

Un criterio ampliamente reconocido para evaluar el rendimiento de los algoritmos de clasificación es la curva ROC. La curva ROC es un gráfico definido dentro del espacio ROC [41]: el espacio ROC es el intervalo unitario bidimensional, $\mathcal{U}_{\text{ROC}}$, que representa el producto cartesiano entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) en un problema de clasificación, es decir, $\mathcal{U}_{\text{ROC}} = \text{FPR} \times \text{TPR}$. Los puntos que conforman la curva ROC se obtienen a partir de la toma de decisiones sobre las clases positiva y negativa al variar el valor del umbral [40]. Cada umbral $i$ genera un punto distinto $(\text{fpr}_i, \text{tpr}_i) \in \mathcal{U}_{\text{ROC}}$, y la curva ROC es el conjunto de todos estos puntos. En esencia, la curva ROC refleja el desempeño de los modelos a lo largo de un espectro de umbrales de clasificación y se considera mejor métrica que la precisión al evaluar modelos de clasificación. La curva ROC de un clasificador perfecto es una línea con $\text{tpr}_i = 1$ para todos los umbrales, mientras que para un clasificador aleatorio se cumple $\text{tpr}_i = \text{fpr}_i$.

El análisis de la curva ROC se ha utilizado desde la Segunda Guerra Mundial, cuando los ingenieros eléctricos la definieron para analizar señales de radar y estudiar la relación señal/ruido con el objetivo de detectar objetos enemigos en el campo de batalla [6, 72, 82]. Desde entonces, las curvas ROC se han utilizado para medir la capacidad de un sistema para detectar una señal contra el ruido de fondo. Esto es crucial en sistemas de radar, redes de comunicación y procesos de control de calidad. Por lo tanto, se pueden encontrar numerosos trabajos también en la industria, especialmente para la detección de alarmas [7, 23, 108]. Las curvas ROC además tienen una gran importancia en el campo médico, especialmente en radiología, medicina preventiva y epidemiología clínica, donde se utilizan para evaluar la efectividad de las pruebas diagnósticas en situaciones como el cáncer, la diabetes y las enfermedades cardíacas [46, 73, 80, 86, 92, 96, 115].

En el campo del scoring financiero, es habitual enfrentarse a problemas con datos desbalanceados, razón por la cual la curva ROC se utiliza frecuentemente como medida y criterio de evaluación [61, 76, 124, 136, 137]. En la predicción meteorológica, es común utilizar la curva ROC para evaluar la habilidad de diferentes modelos de pronóstico en eventos meteorológicos específicos, proporcionando una herramienta cuantitativa para comparar y medir el rendimiento de estos modelos en diferentes condiciones meteorológicas. Esto es especialmente importante en escenarios donde es crítico predecir eventos reales (como huracanes o tornados) con precisión, minimizando las falsas alarmas [15, 28, 119]. La curva ROC también se emplea con frecuencia en escenarios de criminología y forense para establecer un marco analítico robusto. Se centra en su capacidad para discriminar y su confiabilidad para producir resultados precisos de manera sistemática [5, 109, 112].

Los puntos que forman el espacio ROC resultan de la toma de decisiones de las clases

positivas y negativas en un punto de corte, también conocido como valor umbral [40]. Típicamente, los modelos de clasificación asignan una puntuación a cada predicción, y se toma una decisión basada en un umbral elegido para determinar la clase correspondiente. La matriz de confusión representa el resultado de la clasificación a partir de un umbral seleccionado para un modelo.

Una métrica comúnmente utilizada junto a la curva ROC es el AUC. Sirve como métrica global para medir el rendimiento del modelo a lo largo de los posibles umbrales de clasificación, en concreto mide la capacidad del modelo para distinguir entre clases. Un modelo con un AUC cercano a 1 tiene un buen rendimiento, mientras que un AUC de 0.5 indica un rendimiento aleatorio.

Otro tipo de gráfico utilizado principalmente en problemas con datos desbalanceados es la curva PRC (Precision Recall Curve), que muestra los valores de precisión correspondientes a los valores de *recall*. Las curvas PRC expresan la susceptibilidad de los clasificadores a conjuntos de datos desbalanceados con indicadores visuales claros, y permiten la interpretación precisa e intuitiva del rendimiento práctico del clasificador [107]. El AUC de PRC también es efectivo en comparaciones de múltiples clasificadores. Una curva PRC tiene una relación uno a uno con una curva ROC correspondiente, es decir, cada punto en cualquiera de las curvas determina de manera única un punto correspondiente en la otra curva. Al igual que el gráfico ROC, el gráfico PRC proporciona una evaluación a nivel de modelo y se define el espacio PRC como el intervalo unitario bidimensional, $\mathcal{U}_{\text{PRC}}$, que representa el producto cartesiano entre el *recall* y la precisión, es decir, $\mathcal{U}_{\text{PRC}} = \text{TPR} \times \text{Prec}$. En la curva precisión-*recall*, cada umbral genera un punto distinto $(\text{tpr}_i, \text{prec}_i) \in \mathcal{U}_{\text{PRC}}$. Aparte de la diferencia en las métricas utilizadas, los conceptos previamente descritos también son aplicables a esta curva.

### 2.5.1 Optimizaciones ROC

Mozer et al. [90] describen un enfoque novedoso para el diseño de clasificadores en el contexto de la predicción de abandono en la industria de las telecomunicaciones. El abandono se refiere a los clientes que cambian de un proveedor de servicios a otro. En su estudio, se exploran cuatro estrategias para optimizar la curva ROC bajo estas restricciones.

La primera es un algoritmo que se centra en mejorar las tasas de TPR y TNR alrededor de puntos específicos de la curva ROC. Inicialmente se entrenan varios modelos, asignando más peso a las muestras en la región enfatizada. Este proceso se repite hasta que el rendimiento en el conjunto de entrenamiento alcanza un punto de estabilidad. La segunda técnica es un algoritmo similar al anterior. Sin embargo, da menos peso a las muestras que no afectan significativamente a las tasas de TPR y TNR en la región de interés. La tercera técnica es la optimización restringida, donde el problema de optimización restringido se convierte en un problema no restringido utilizando el método de Lagrangiano aumentado. Posteriormente, se itera para encontrar el TNR máximo mientras se mantiene un TPR constante. Finalmente, aplican un algoritmo genético para optimizar los parámetros del

clasificador. Al analizar los resultados, sugieren la necesidad de enfoques más robustos para abordar este problema, ya que todos los algoritmos propuestos dieron ganancias similares, lo que podría indicar un límite en la mejora alcanzable con el modelo evaluado.

Gao et al. [45] proponen un marco de aprendizaje en conjunto llamado Ensemble Maximal Figure-of-Merit (E-MFoM) para la optimización de curvas ROC en problemas de clasificación. Aunque la maximización del AUC es un enfoque común, E-MFoM busca una mayor flexibilidad al considerar múltiples reglas de decisión y adaptarse a diversas métricas de rendimiento. Para ello, introduce un enfoque discriminativo que entrena clasificadores optimizando directamente la métrica objetivo. Los resultados muestran un rendimiento superior al de los enfoques convencionales de optimización de AUC. E-MFoM ofrece un enfoque integral para la optimización de curvas ROC, destacando su flexibilidad, capacidad discriminativa y adaptabilidad a diversas métricas de rendimiento.

Pietraszek [97] propone un método para construir clasificadores binarios de abstención de manera óptima mediante el análisis ROC. Estos clasificadores pueden abstenerse de clasificar en ciertos casos para reducir errores en la clasificación. El trabajo presenta tres modelos de optimización: basado en costes, con abstención limitada y con mejora limitada. En el modelo basado en costes, el coste de la mala clasificación se optimiza ajustando el modelo para mantener un equilibrio entre distintas métricas de rendimiento, asegurando que el impacto de los errores sea el menor posible. Para los modelos de abstención limitada y mejora limitada, se introduce un algoritmo eficiente para encontrar el clasificador óptimo, aprovechando las propiedades del Convex Hull ROC. El estudio valida estos métodos utilizando un algoritmo de construcción de curvas ROC y validación cruzada en 15 conjuntos de datos UCI KDD, demostrando su efectividad en la reducción de errores en sistemas de clasificación en la vida real.

Zhang et al. [134] proponen un método de optimización de umbrales utilizando la curva ROC y el agrupamiento de sensibilidad en un sistema de alarma. Empleando el algoritmo de optimización por enjambre de partículas Wang et al. [120], optimizan la función objetivo, incorporando FPR y TPR con pesos específicos. Los resultados de la simulación muestran que el método reduce efectivamente la FPR, disminuye el número de alarmas y clasifica las variables según su sensibilidad. Además, destacan la capacidad del método para dar más tiempo de respuesta a los operadores para situaciones anómalas, mejorando la seguridad del proceso. Otro método comúnmente utilizado para determinar el punto óptimo dentro del codo de la curva ROC es el Índice de Youden [43]. Albright [5] utiliza este índice para abordar un problema en medicina relacionado con las pruebas de cáncer de próstata.

Numerosos esfuerzos de investigación han explorado la optimización del espacio ROC, a menudo empleando algoritmos genéticos multiobjetivo (MOGAs). Por ejemplo, Chatelain et al. [24] utilizaron la optimización multiobjetivo, integrando NSGA-II (Non-dominated Sorting Genetic Algorithm) [31], para optimizar hiperparámetros para un conjunto de modelos con objetivos duales. Lévesque et al. [74] examinaron trayectorias de clasificadores

simples en el espacio ROC, empleando NSGA-II e investigando combinaciones booleanas a través del método Iterative Boolean Combination (IBC). Bhowan et al. [16] utilizaron programación genética multiobjetivo (MOGP, del inglés *Multi-Objective Genetic Programming*) [116] para optimizar el AUC, utilizando la precisión de la clase minoritaria y mayoritaria como objetivos de aprendizaje. Muestran cómo, en algunos problemas, las soluciones obtenidas por su enfoque tienen mejor AUC que los resultados con programación genética.

## 2.6 Optimización de modelos ensemble

Los modelos *ensemble* han demostrado ser una de las estrategias más efectivas en ML, ya que combinan múltiples clasificadores para mejorar la robustez y precisión del modelo final.

Mienye and Sun [88] proporcionan una visión completa sobre los enfoques de aprendizaje por *ensemble*, detallando los métodos de *Bagging*, *Boosting* y Stacking y su desarrollo hasta los algoritmos más recientes, como Random Forest, AdaBoost, Gradient Boosting, XGBoost, LightGBM, y CatBoost. El estudio enfatiza cómo el uso de técnicas como el voto mayoritario y el voto ponderado permite mejorar la precisión de los modelos al combinar predicciones de múltiples modelos base. También discuten estrategias para seleccionar el subconjunto óptimo de clasificadores en un modelo *ensemble*, y presentan tanto métodos estáticos como dinámicos para la selección de los clasificadores base. Estos enfoques han demostrado ser eficaces en aplicaciones complejas, como el diagnóstico médico, la detección de fraude y el análisis de sentimientos.

En cuanto a las aplicaciones prácticas, en el artículo exploran su uso en sectores como la salud y la industria, donde los métodos de *ensemble* han logrado superar el rendimiento de los modelos individuales, especialmente en problemas de clasificación y en entornos con datos de alta complejidad. Sugieren futuras direcciones de investigación, incluyendo la reducción del consumo de recursos en modelos *ensemble* basados en DL, con el objetivo de hacerlos más accesibles para aplicaciones en contextos con recursos limitados.

En relación con la optimización de modelos de *ensemble*, existen diferentes métodos. Yin and Li [129] utilizan la optimización bayesiana (BOA) para ajustar los hiperparámetros de los modelos XGBoost y Random Forest. BOA emplea un proceso de regresión gaussiana y un criterio de selección, que sirve para decidir qué combinaciones de hiperparámetros probar, buscando un equilibrio entre explorar nuevas opciones y aprovechar las que ya han dado buenos resultados. Con BOA, optimizan los hiperparámetros al encontrar el equilibrio entre la exploración y la explotación, mejorando significativamente el rendimiento de los modelos sin evaluar innecesariamente la función objetivo. Los resultados muestran mejoras en la precisión y estabilidad de los modelos optimizados en comparación con sus versiones no optimizadas.

Otra optimización de modelos *ensemble* con un enfoque similar es propuesta por Du et al.

[38], donde presentan el Ensemble Dinámico de Optimización Bayesiana (BODE), que mejora BOA ajustando dinámicamente los pesos y seleccionando modelos del *ensemble* según su rendimiento reciente. BODE actualiza la configuración del *ensemble* usando BOA para explorar y explotar las mejores combinaciones de modelos, adaptándose a patrones cambiantes en datos de series temporales.

En el campo de la salud, Mienye and Jere [87] integran una optimización bayesiana y técnicas de IA explicable en un *ensemble* compuesto por AdaBoost, Random Forest y XGBoost. Utilizando optimización bayesiana, ajustan los hiperparámetros de los modelos para maximizar la precisión en la predicción de enfermedades cardíacas, obteniendo un 98.9 % de sensibilidad y 97.1 % de especificidad en el conjunto de datos Cleveland.

El aprendizaje por apilamiento (Stacking learning) también se ha utilizado en el proceso de optimización de modelos *ensemble*. En Ksiazek et al. [71] se presenta un modelo de ML que combina el aprendizaje por apilamiento con algoritmos genéticos (AG). El proceso de optimización del modelo *ensemble* utiliza un enfoque de dos capas con algoritmos genéticos para la selección de variables y el ajuste de parámetros. La primera capa aplica para seleccionar las variables más relevantes y optimizar los parámetros de varios clasificadores individuales (KNN, Random Forest, Naïve Bayes, entre otros), mejorando su rendimiento. La segunda capa optimiza un meta-clasificador, que combina las salidas de los clasificadores individuales, ajustando sus parámetros para maximizar la precisión del *ensemble*. Este enfoque evolutivo permite una optimización eficiente y una mejora significativa en el rendimiento del modelo *ensemble*.

Shahhosseini et al. [111] realizaron una investigación exhaustiva sobre modelos *ensemble* en ML, centrándose sobre todo en las técnicas de aprendizaje por apilamiento. El principal desafío era seleccionar cuidadosamente los modelos base, ajustar sus hiperparámetros y asignar pesos adecuados a cada modelo. En contraste con las metodologías convencionales que tratan estas tareas como pasos discretos, su investigación introduce un marco integrado. Este enfoque optimiza los hiperparámetros y la asignación de pesos, consiguiendo así una mejora notable en el rendimiento de los modelos de conjunto.

En un estudio reciente, Yuan et al. [131] proponen un método innovador basado en optimización de tensores, mediante el cual representan las salidas de múltiples modelos del modelo *ensemble* como una estructura multidimensional. A partir de esta representación, aplican técnicas de reducción para identificar un subconjunto de modelos representativos que preserve el rendimiento predictivo, minimizando al mismo tiempo la carga computacional. El uso de tensores facilita cálculos algebraicos avanzados, que optimizan la selección de variables y pesos entre los submodelos del modelo *ensemble*. Esto permite al modelo alcanzar un equilibrio entre precisión y eficiencia sin consumir demasiados recursos. Los resultados muestran que la optimización basada en tensores logra un desempeño comparable al de los modelos *ensemble* tradicionales, pero utilizando menos recursos, haciéndolo ideal para aplicaciones de alta dimensionalidad y big data.

En aplicaciones industriales y de logística, el uso de metaheurísticas como el algoritmo de

enjambre de partículas (PSO, del inglés *Particle Swarm Optimization*) y los algoritmos genéticos ha demostrado ser efectivos para la optimización de modelos *ensemble* [102]. En Matloob et al. [83] revisan los avances en el uso de técnicas de aprendizaje por *ensemble* para la predicción de defectos en software, abordando métodos como *Bagging*, *Boosting*, Stacking y diversas configuraciones híbridas y basadas en metaheurísticas. Usan los algoritmos mencionados anteriormente para mejorar la selección y ponderación de los modelos base en los *ensembles*, logrando un balance entre precisión y eficiencia computacional. Los autores concluyen que en la mayoría de los casos, los métodos *ensemble* superan a los clasificadores individuales en términos de precisión y robustez, especialmente en tareas de clasificación complejas y problemas de balanceo de clases. También se destaca la efectividad del uso de técnicas de preprocesamiento, como SMOTE y selección de variables, para mejorar el rendimiento de los *ensemble*.

Como se observa a lo largo de este capítulo, el sesgo derivado de datos desbalanceados sigue siendo uno de los principales retos para garantizar la precisión de los modelos de ML, especialmente cuando se aplican técnicas *ensemble*. Aunque la literatura ha propuesto múltiples soluciones (desde técnicas de sobremuestreo y submuestreo, hasta algoritmos híbridos y métodos basados en metaheurísticas), ninguna de ellas resuelve por completo la complejidad de equilibrar precisión, robustez y eficiencia computacional. La optimización de modelos *ensemble* mediante métodos como PSO, algoritmos genéticos y optimizaciones bayesianas ha logrado avances notables en la selección de variables, pesos y umbrales, pero aún persisten limitaciones cuando se enfrentan a problemas con alto desbalanceo de clases o arquitecturas complejas. Por tanto, la necesidad de nuevas propuestas que integren estrategias de balanceo configurables y robustas dentro de esquemas *ensemble* motiva la contribución de esta tesis, centrada en desarrollar un enfoque optimizado para maximizar la detección de la clase minoritaria sin depender de un único clasificador base.

# Capítulo 3

# Optimización de modelos *ensemble* con datos altamente desbalanceados

El sesgo de los datos es uno de los factores más críticos en los problemas de ML, ya que puede provocar confusión en los modelos e influir en su precisión. Una de las fuentes de sesgo en los conjuntos de datos del mundo real es la distribución desequilibrada de las clases.

La mayoría de las técnicas existentes en la literatura para el tratamiento de datos desbalanceados abarcan el submuestreo o el sobremuestreo del conjunto de datos. Aunque estas técnicas han demostrado ser eficaces para tratar el desequilibrio de clases, tienen ciertas limitaciones. Los métodos de sobremuestreo pueden introducir ruido o generar muestras no representativas, mientras que el submuestreo puede provocar una pérdida de información importante. Métodos como EasyEnsemble y BalanceCascade, aunque eficaces, suelen estar ligados al uso de AdaBoost como clasificador base, y métodos más complejos como SPE pueden ser difíciles de implementar. Por tanto, es necesario crear un equilibrio entre simplicidad y eficiencia, facilitando la configurabilidad y adaptabilidad a diferentes clasificadores. En este capítulo, desarrollamos una propuesta que da respuesta a estas necesidades, ofreciendo una alternativa configurable que se basa en la metodología *Underbagging* para generar subconjuntos balanceados y busca maximizar el *recall* de clases minoritarias, sin estar restringido a un clasificador concreto.

## 3.1 Propuesta

Proponemos un método de *Underbagging* denominado BUE (Balanced Underbagged Ensemble Approach) que evita los sesgos de los algoritmos de clasificación ML en datos desbalanceados. BUE se caracteriza por dos enfoques principales: (i) una técnica híbrida que utiliza métodos de *Underbagging* para tratar datos desbalanceados y (ii) un método para construir un clasificador *ensemble* que proporciona una clasificación sin sesgos.

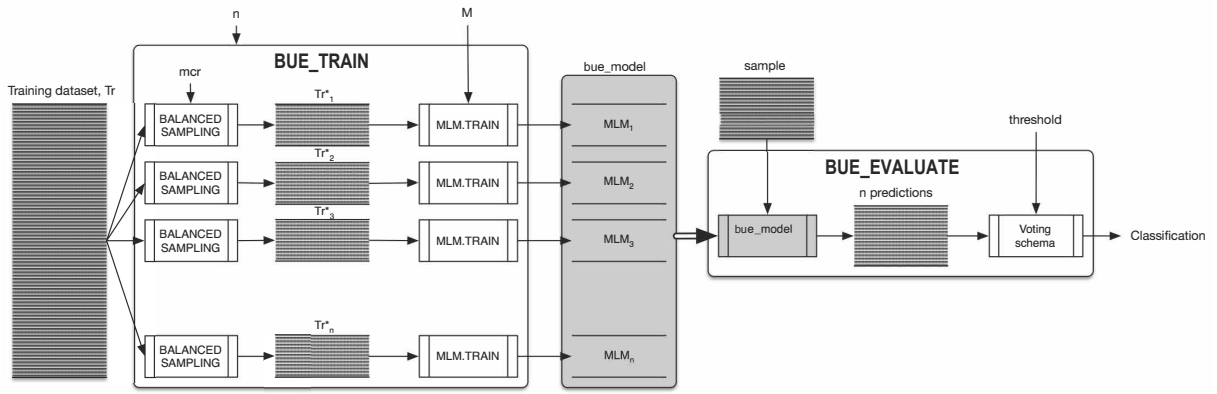



Figura 3.1: Flujo de trabajo de Balanced Underbagged Ensemble Approach. La caja con la etiqueta `BALANCED_SAMPLING` representa el algoritmo 2, `BUE_TRAIN` se muestra en el algoritmo 1, y `BUE_EVALUATE` en el algoritmo 3.

Para comprender el funcionamiento de BUE, la figura 3.1 muestra el flujo de trabajo. BUE tiene dos etapas: `BUE_TRAIN` y `BUE_EVALUATE`.

La primera etapa, `BUE_TRAIN`, entrena un conjunto de `n` modelos de ML (algoritmo 1) que forman el modelo *ensemble* utilizado en `BUE_EVALUATE`. Cada modelo, $\texttt{MLM}_i$, se entrena con un subconjunto de entrenamiento balanceado, $\texttt{Tr}_i^*$, creado mediante la función `BALANCED_SAMPLING` según algoritmo 2.

**Algorithm 1** Entrenamiento del modelo con BUE

**Input**: Algoritmo de ML : M; Conjunto de entrenamiento: Tr; Número de subconjuntos y modelos: n

**Output**: bue_model

```
1: function BUE_TRAIN(M, Tr, n)
2:     mcr ← 0.5                                   ▷ Ratio de la clase minoritaria
3:     bue_model ← {}
4:     for i ← 1 to n do
5:         Tr* ← BALANCED_SAMPLING(Tr, mcr)
6:         MLM ← MLMODEL(M)                        ▷ Instancia del algoritmo ML
7:         bue_model.APPEND(MLM.TRAIN(Tr*)) ▷ Entrenamiento del modelo ML, con
   el subconjunto de entrenamiento
8:     end for
9:     return bue_model
10: end function
```

En la implementación actual de BUE, todos los modelos se obtienen utilizando el mismo algoritmo de ML, $M$. Sin embargo, versiones futuras de BUE podrían considerar la combinación de diferentes tipos de algoritmos de aprendizaje automático.

Los parámetros de entrada del algoritmo 1 son el algoritmo de aprendizaje automático ($M$), el conjunto de entrenamiento original (Tr) y el número de modelos que formarán el *ensemble* (`n`). En cada una de las `n` iteraciones del bucle, se genera un subconjunto balanceado utilizando la función `BALANCED_SAMPLING` de acuerdo con el algoritmo 2. Este

mecanismo garantiza que: (i) la diversidad de la clase mayoritaria esté representada en los n subconjuntos equilibrados, y que (ii) los subconjuntos sean una representación equilibrada del conjunto de datos.

**Algorithm 2** Crea un subconjunto de entrenamiento equilibrado a partir del conjunto de entrenamiento

**Input**: Conjunto de entrenamiento: Tr; Ratio de la clase minoritaria para el nuevo subconjunto de entrenamiento: mcr

**Output** : Subconjunto de entrenamiento, Tr*

1: **function** BALANCED_SAMPLING(Tr, mcr)
2: $n_m \leftarrow$ Number of minority samples from Tr
3: $n_M \leftarrow round$((1-mcr)*($n_m$)/ mcr) ▷ Número de muestras de la clase mayoritaria para Tr*
4: $idx_m \leftarrow$ All sample indexes for minority class in Tr
5: $idx_M \leftarrow$ EXTRACT(Tr, $n_M$) ▷ $n_M$ índices de clase mayoritaria seleccionados aleatoriamente
6: Tr* $\leftarrow$Tr($\{idx_m\} \cup \{idx_M\}$)
7: **return** Tr*
8: **end function**

El algoritmo 2 recibe dos entradas: Tr y un parámetro que indica la proporción de muestras de la clase minoritaria en el subconjunto, `mcr` (minority class ratio). La función devuelve un subconjunto, $\texttt{Tr}_i^*$, compuesto por todas las muestras de la clase minoritaria y una selección aleatoria con reemplazo (*Bagging*) de la clase mayoritaria. Si el número de muestras de la clase minoritaria en el conjunto de datos inicial es $s_m$, se seleccionará un total de $s_M = s_m \cdot (1 - mcr)/mcr$ muestras de la clase mayoritaria.

La complejidad de `BALANCED_SAMPLING` está determinada por la llamada a la función `EXTRACT(Tr,` $s_M$`)`, cuya complejidad es $\mathcal{O}(s_M)$. En consecuencia, la complejidad de algoritmo 1 es $\mathcal{O}(\texttt{n} \cdot s_{MC})$.

Debido al procedimiento de muestreo, cada subconjunto es diferente; en consecuencia, los n clasificadores resultantes están débilmente correlacionados y pueden combinarse en un clasificador *ensemble*, construyendo un clasificador robusto, `BUE_model`, tanto para la clase mayoritaria como para la clase minoritaria.

Una vez entrenado el clasificador *ensemble*, se proporciona una clasificación utilizando un esquema de votación por mayoría cualificada. El algoritmo 3 presenta `BUE_EVALUATE`, que es el procedimiento para evaluar una muestra. El *ensemble* clasifica una instancia como perteneciente a la clase minoritaria si el valor promedio de las predicciones de los n clasificadores supera un umbral; en caso contrario, se clasifica como perteneciente a la clase mayoritaria.

Los parámetros de entrada de `BUE_EVALUATE` son el modelo *ensemble*, `BUE_model`, la muestra a evaluar y el umbral. La función devuelve una predicción para la muestra evaluada. En este caso, la complejidad del algoritmo es $\mathcal{O}(\texttt{n})$.

**Algorithm 3** Evaluación del modelo Balanced Underbagged Ensemble

**Input**: BUE model: bue_model; sample to evaluate: sample; threshold for sample classification: threshold

**Output**: Classification

```
1: function BUE_EVALUATE(bue_model, sample, threshold)
2:     ensemble_results ← {}
3:     for each model ∈ bue_model do
4:         ensemble_results.append(model.EVALUATE(sample))
5:     end for
6:     vote ← (1/n) ∑ ensemble_results
7:     if vote ≥ threshold then
8:         return minority_class
9:     else
10:        return majority_class
11:    end if
12: end function
```

**Algorithm 4** Función principal BUE

**Output**: Class prediction for the test sample

```
1: function BUE_MAIN( )
2:     Tr ← training sample load
3:     sample ← test sample load
4:     M ← ML algorithm
5:     n ← Number of subsets and models
6:     bue_model ← BUE_TRAIN(M, Tr, n)
7:     threshold ← threshold for sample classification
8:     class_prediction ←BUE_EVALUATE(bue_model, sample, threshold)
9:     return class_prediction
10: end function
```

El algoritmo 4 es el encargado de cargar los datos necesarios y llamar a las funciones de entrenamiento y evaluación. Dado que invoca las dos funciones principales de BUE, su complejidad es la mayor de ambas, $\mathcal{O}(\mathtt{n} \cdot s_M)$. Por lo tanto, la complejidad de nuestra propuesta es lineal y depende del tamaño del conjunto de entrenamiento y del número de modelos.

## 3.2 Conjunto de datos

Para evaluar la eficiencia de BUE, se han empleado dos conjuntos de datos altamente desbalanceados. El primero de ellos, denominado `Seguros`, recoge información sobre pólizas de seguros de automóviles y fue proporcionado por Biztools S.L., una empresa española especializada en el desarrollo de software para el sector asegurador. Desde 2010, esta compañía gestiona un corredor digital centrado en la venta de seguros exclusivamente online.

El objetivo principal es predecir la aparición de siniestros futuros. En este conjunto, cada fila representa un año de cobertura de una póliza. Inicialmente, los datos incluían 81 194 registros y 102 variables. Tras aplicar un proceso de preprocesamiento, limpieza, armonización y filtrado, se obtuvo una versión depurada con 69 080 filas y 21 variables. El conjunto presenta un fuerte desbalanceo de clases: solo 681 de las 69 080 observaciones corresponden a pólizas con siniestros, lo que equivale a tan solo un 1.1 % del total.

Para proteger la privacidad de los titulares de las pólizas, se eliminaron todos los identificadores personales durante el preprocesamiento y el conjunto de datos fue anonimizado de acuerdo con las regulaciones de protección de datos, incluida la Regulación General de Protección de Datos de la Unión Europea. El uso de los datos estuvo restringido a fines de investigación bajo estrictos acuerdos de intercambio de datos, garantizando que no hubiera intentos de reidentificar a los individuos.

Debido a la firma de un acuerdo de confidencialidad con la entidad proveedora de los datos, no es posible proporcionar información detallada sobre las variables ni sobre la estructura interna del conjunto de datos.

El segundo conjunto de datos, denominado `Fraude`, contiene transacciones con tarjetas de crédito registradas durante dos días de septiembre de 2013 [98]. Reúne un total de 284 807 operaciones, de las cuales únicamente 492 corresponden a casos de fraude, lo que representa un alto desbalanceo con tan solo un 0.172 % del total. Está compuesto por 30 variables, de las cuales 28 provienen de una transformación PCA realizada previamente por los autores del conjunto. Por motivos de confidencialidad, no se dispone de las variables originales ni del contexto exacto de cada transacción.

## 3.3 Marco experimental y metodología

Para comprobar la calidad de BUE comparamos su eficiencia con once técnicas de muestreo pertenecientes a las principales familias utilizadas para tratar datos desbalanceados, explicadas en la sección 2.4, combinadas con diez algoritmos de ML. Para cada combinación de algoritmo y técnica se lanzaron 40 ejecuciones. Así, para un total de 110 combinaciones, resultaron 4400 ejecuciones. Con estas 40 ejecuciones por combinación se pretende determinar qué técnica funciona mejor independientemente del algoritmo de ML utilizado. Se han elegido 40 ejecuciones ya que el consenso establece que el número mínimo de observaciones para realizar estadísticas significativas debe ser superior a 30. El conjunto de datos original se ha separado en subconjuntos diferentes para cada ejecución, siendo un 70 % correspondiente a entrenamiento y 30 % a test.

Los diez algoritmos de ML utilizados para la comparación son:

- Extreme Gradient Boosting (xgb).
- Random Forest Classifier (rf).
- Logistic Regression (lr).

- Gaussian Naive Bayes (gnb).
- Gradient Boosting (gb).
- Bernoulli Naive Bayes (bnb).
- Extra Tree Classifier (et).
- Decision Trees (dt).
- Bagging Classifier (bc).
- AdaBoost Classifier (adb) [8].

Las once técnicas de muestreo son:

- Balanced Underbagged Ensemble (BUE).
- SMOTE (S).
- SMOTETomek (ST).
- Borderline SMOTE (BS).
- Adaptive Synthetic (ADASYN).
- Random OverSampler (RO).
- Condensed Nearest Neighbors (CNN).
- NearMiss-1 (NM1).
- NearMiss-2 (NM2).
- NearMiss-3 (NM3).
- Datos originales desbalanceados (IMBAL).

Dado que trabajamos con dos conjuntos de datos extremadamente desbalanceados, métricas de rendimiento tradicionales (como la precisión) no son las más adecuadas, ya que pueden ser engañosas en escenarios donde la clase mayoritaria domina. Un clasificador que prediga siempre la clase mayoritaria puede lograr una alta precisión, pero fallar en la identificación de los casos de la clase minoritaria.

Takaya Saito [107] revisa las métricas de evaluación ROC y PRC y muestra como PRC ofrece más ventajas que ROC en caso de datos desbalanceados. Sin embargo, en escenarios muy desbalanceados, la precisión se ve afectada significativamente, ya que un número muy reducido de falsos positivos en la clase minoritaria implica una precisión baja. Esto se refleja en PRC, ya que los valores de la curva están limitados por la precisión, aunque se obtenga un *recall* elevado en cada clase.

En su lugar, utilizamos la media geométrica (GMean) de TPR ($\text{recall}_\text{m}$) y TNR ($\text{recall}_\text{M}$) como la métrica principal de evaluación, Ecuación (3.1), que garantiza un equilibrio en el rendimiento de clasificación entre ambas clases [14].

$$\text{Gmean} = \sqrt{\text{recall}_\text{M} \cdot \text{recall}_\text{m}} \tag{3.1}$$

El uso de métricas como GMean en problemas de clasificación con conjuntos de datos desbalanceados es una práctica común, como se muestra en [48, 114]. Maximizar GMean es especialmente relevante en estos escenarios, ya que refleja la capacidad del modelo para clasificar correctamente tanto las instancias de la clase mayoritaria como las de la clase

minoritaria. Al emplear GMean, garantizamos que tanto $\text{recall}_\text{m}$ como $\text{recall}_\text{M}$ contribuyan equitativamente al proceso de evaluación, evitando un sesgo hacia la clase mayoritaria y permitiendo una comparación justa entre distintos modelos y técnicas de muestreo.

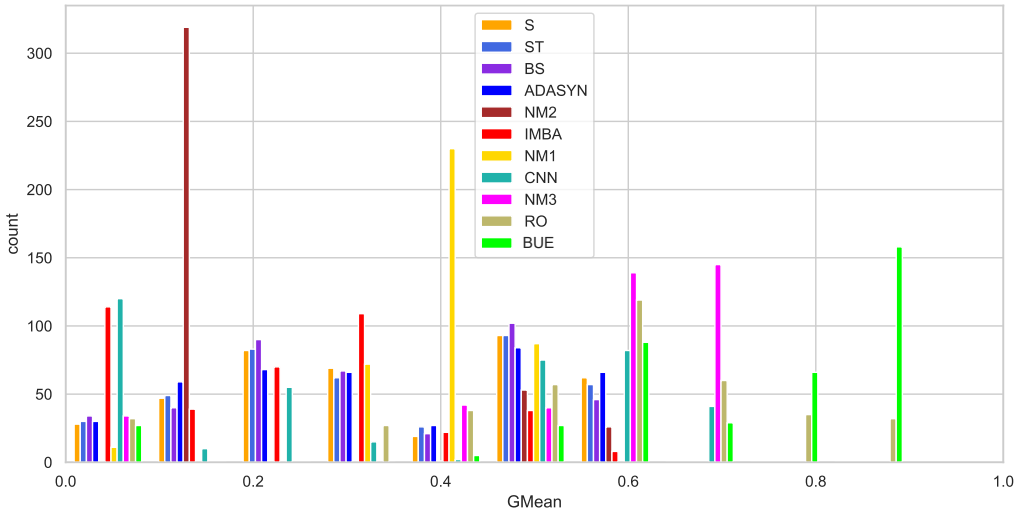
## 3.4 Análisis de resultados

Los resultados que se muestran a continuación corresponden a una configuración de BUE obtenida tras un proceso previo de exploración y ajuste: `n` = 50, `mcr` = 0.5 y `threshold` = 0.8.

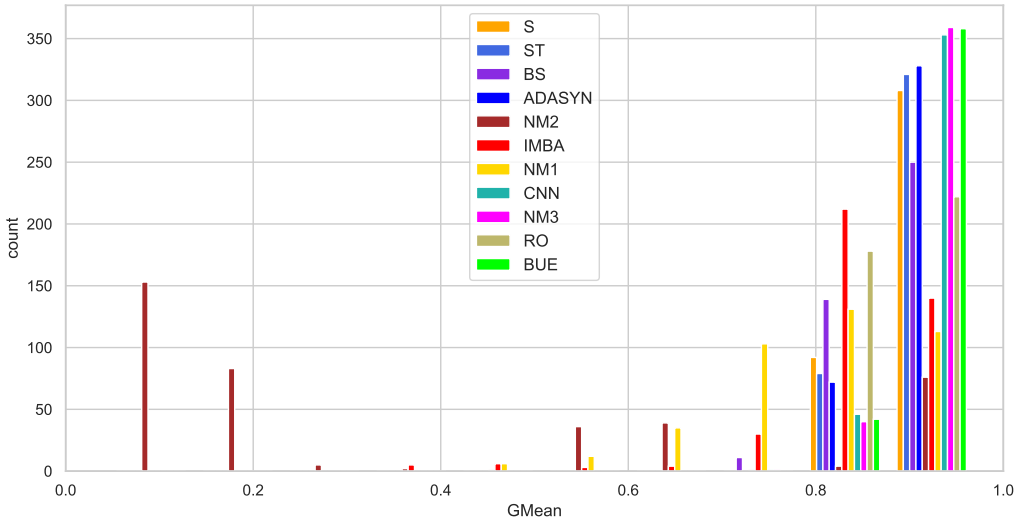
En la figura 3.2 se muestra la distribución de los valores de GMean obtenidos en las 40 ejecuciones de cada técnica de muestreo con los diez algoritmos de ML evaluados en los dos conjuntos de datos. Cada técnica fue evaluada un total de 400 veces, lo que permite observar su consistencia y rendimiento en distintos dominios. Los resultados se agrupan en intervalos de 0.10. Dentro de cada intervalo, se muestra el recuento de resultados de cada técnica de forma separada. Esto permite visualizar claramente la cantidad total de valores dentro de cada rango para cada técnica.

En el conjunto de datos `Seguros`, figura 3.2a, las técnicas evaluadas muestran distribuciones variadas. BUE concentra la mayoría de sus resultados en valores altos de GMean (cercanos a 0.8), logrando un mejor equilibrio entre $\text{recall}_\text{M}$ y $\text{recall}_\text{m}$ en comparación con otras técnicas. Métodos como BS y ADASYN presentan distribuciones más dispersas, con un mayor número de resultados en un rango intermedio de GMean, $[0.4, 0.6]$. En el caso de NM1 y CNN, también se observa una gran acumulación de valores en intervalos intermedios, aunque con una ligera tendencia hacia valores algo más bajos. IMBA muestra un rendimiento bajo, con valores ($\approx 0.2$), lo que demuestra la necesidad de aplicar técnicas para el tratamiento de datos desbalanceados ya que su rendimiento está sesgado por la clase mayoritaria. Técnicas como RO y NM3 también muestran una mayor concentración de resultados en los rangos medios-altos de GMean, aunque no alcanzan los resultados de BUE. En cambio, NM2 destaca por acumular una gran parte de sus resultados en los valores más bajos, lo que refleja un rendimiento poco consistente en este conjunto de datos.

En el conjunto de datos `Fraude`, figura 3.2b, los resultados sin aplicar ninguna técnica (IMBA) muestran un aumento significativo de los valores de GMean respecto al conjunto de datos `Seguros`, con más de la mitad de los valores ubicados en el rango $[0.4, 0.6]$. A diferencia del caso anterior, los resultados están menos dispersos y tienden a concentrarse en valores altos de GMean. BUE mantiene su buen rendimiento, con una gran densidad de resultados en valores de GMean en el rango $[0.9, 1.0]$. La mayoría de las técnicas presentan sus mejores resultados en el rango $[0.7, 0.9]$. Sin embargo, NM2, al igual que en el caso anterior, sigue obteniendo los valores más bajos. Esto puede deberse a su funcionamiento, ya que la selección de muestras de la clase mayoritaria más cercanas a las de la clase minoritaria podría generar una pérdida de información con respecto a la



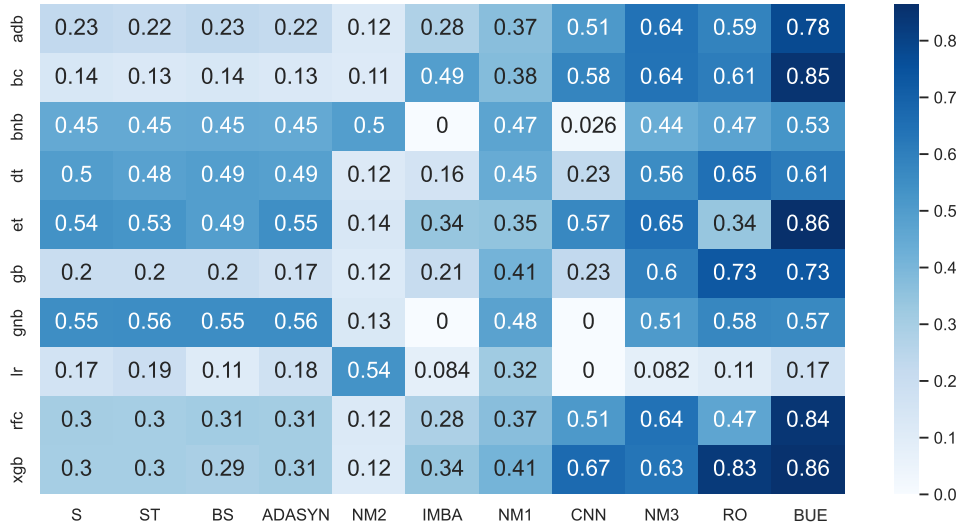

(a) Conjunto de datos `Seguros`



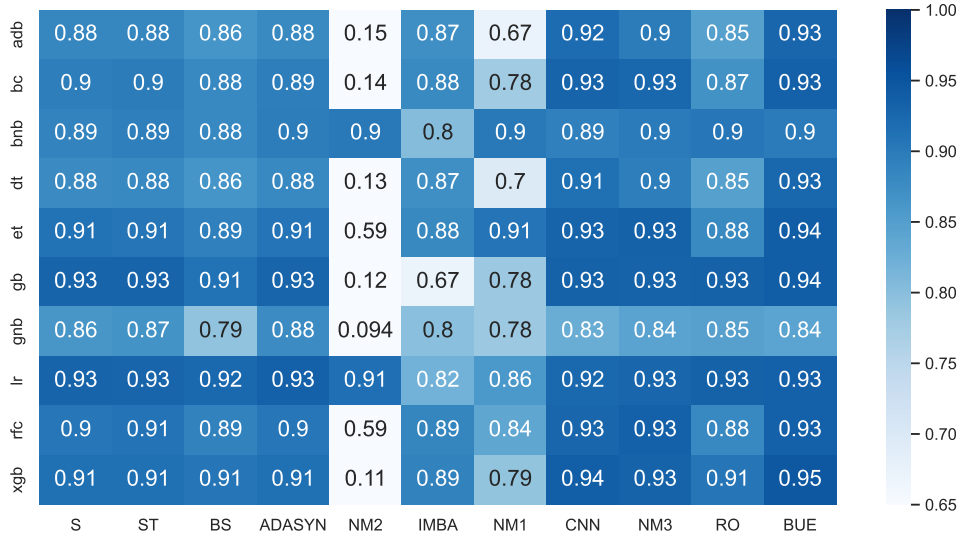

(b) Conjunto de datos `Fraude`

Figura 3.2: Distribución de los resultados para cada una de las técnicas de tratamiento de datos desbalanceados en los dos conjuntos de datos. Los resultados se presentan en intervalos de 0.10; por ejemplo, un GMean de 0.85 o 0.87 aparece en el intervalo 0.80-0.90. Para cada intervalo, se muestra el recuento de resultados de cada técnica de forma independiente.

clase mayoritaria.




(a) Conjunto de datos `Seguros`




(b) Conjunto de datos `Fraude`

Figura 3.3: Resultados promedio de GMean obtenidos por cada combinación de técnica de muestreo y algoritmo de ML.

Adicionalmente, para ampliar la comparativa de los resultados, se ha generado la figura 3.3, que muestra un mapa de calor con la media de los valores GMean obtenidos para cada combinación de técnica de muestreo y algoritmo de ML. Cada técnica se representa en el eje de abscisas, y cada uno de los algoritmos de ML en el eje ordenadas. Cada celda dentro del mapa de calor es la media resultante de las 40 ejecuciones técnica+algoritmo. El rango de valores varía desde 0, representado por el color azul más claro, hasta 1, el más oscuro. El mapa de calor está parcialmente ordenado, con las puntuaciones más altas ubicadas a la derecha de la tabla y las puntuaciones más bajas a la izquierda.

En el conjunto de datos `Seguros` (figura 3.3a), BUE obtiene los mejores resultados para la mayoría de los algoritmos, seguida de RO y NM3. Independientemente de la técnica

de muestreo, el peor algoritmo en este conjunto es lr, ya que la mayoría de sus resultados se aproximan a 0. Estos resultados ponen de manifiesto la efectividad del método BUE, diseñado específicamente para combinar de forma equilibrada el submuestreo y el *bagging* en escenarios con un alto desbalanceo entre clases. A diferencia de otras técnicas, BUE maximiza la detección de la clase minoritaria sin comprometer la robustez del modelo, reforzando la diversidad entre clasificadores base y reduciendo la pérdida de información de la clase mayoritaria. Por ello, BUE se presenta como una alternativa sólida frente a métodos clásicos de muestreo, ofreciendo resultados más estables y precisos en contextos aseguradores donde el desbalanceo de datos es especialmente crítico.

La figura 3.3b presenta un rango de colores diferente debido a los altos valores obtenidos por IMBA en el conjunto de datos `Fraude`. Si se hubiera mantenido la misma escala, apenas se notarían diferencias entre las tonalidades de color de las técnicas. No obstante, BUE sigue obteniendo los mejores resultados, seguido de NM3 y CNN, cuyos valores en algunos casos son similares a los de nuestra propuesta.

La técnica BUE logra los mejores resultados para la mayoría de los algoritmos. Independientemente de la técnica de muestreo utilizada, el peor algoritmo para el conjunto de datos `Seguros` es lr, mientras que para el conjunto de datos `Fraude` es gnb. Las mejores combinaciones de técnicas de muestreo y algoritmos son: BUE+xgb, BUE+et, BUE+bc y BUE+rf para el conjunto de datos `Seguros`; y BUE+xgb, BUE+et y BUE+gb para el conjunto de datos `Fraude`.

También se ha evaluado nuestra propuesta sobre el conjunto de datos Abalone19, considerado uno de los más desbalanceados utilizados en [44], donde se analizan diferentes enfoques *ensemble* aplicados a problemas de desbalanceo de clases. Este conjunto de datos no formó parte de nuestro banco de pruebas principal, pero sí para obtener una comparación directa con las más de diez técnicas consideradas en dicho estudio. Los resultados obtenidos con BUE superaron claramente a los enfoques clásicos presentados, e incluso a las técnicas *ensemble* e híbridas analizadas. Mientras que Galar et al. [44] obtuvieron valores de AUC por debajo de 0.70 en todos los métodos evaluados, nuestro modelo BUE alcanzó un AUC de 0.85 utilizando XGBoost y de 0.82 con Random Forest Classifier.

Aunque Galar et al. [44] evaluaron sus métodos en una amplia colección de 44 conjuntos de datos reales, la mayoría de ellos presentaban un número relativamente reducido de instancias y ratios de desbalanceo (IR, del inglés *Imbalanced Ratio*) moderados. Por ejemplo, Abalone19, el conjunto más desbalanceado de su estudio, presenta un IR de 129.44 y 4174 instancias. En contraste, nuestros conjuntos de datos son más exigentes. El conjunto de seguros utilizado en esta tesis contiene 69 080 instancias con un IR de 100.41, mientras que el conjunto de fraude incluye 284,807 transacciones con un IR extremo de 577.85. Estos valores superan ampliamente las condiciones de desbalanceo de la mayoría de los conjuntos de datos utilizados en trabajos previos.

### 3.4.1 Análisis de sensibilidad de hiperparámetros

Con el fin de evaluar la robustez de BUE respecto a sus hiperparámetros, se realizó un análisis de sensibilidad exhaustivo variando: el número de modelos del *ensemble* (`n`), el ratio de clase minoritaria en cada subconjunto de entrenamiento (`mcr`) y el umbral de votación (`threshold`). Los valores evaluados fueron:

- $n \in \{30, 40, 50, 60, 70, 100\}$
- $mcr \in \{0.4, 0.45, 0.5, 0.55, 0.6\}$
- $threshold \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$

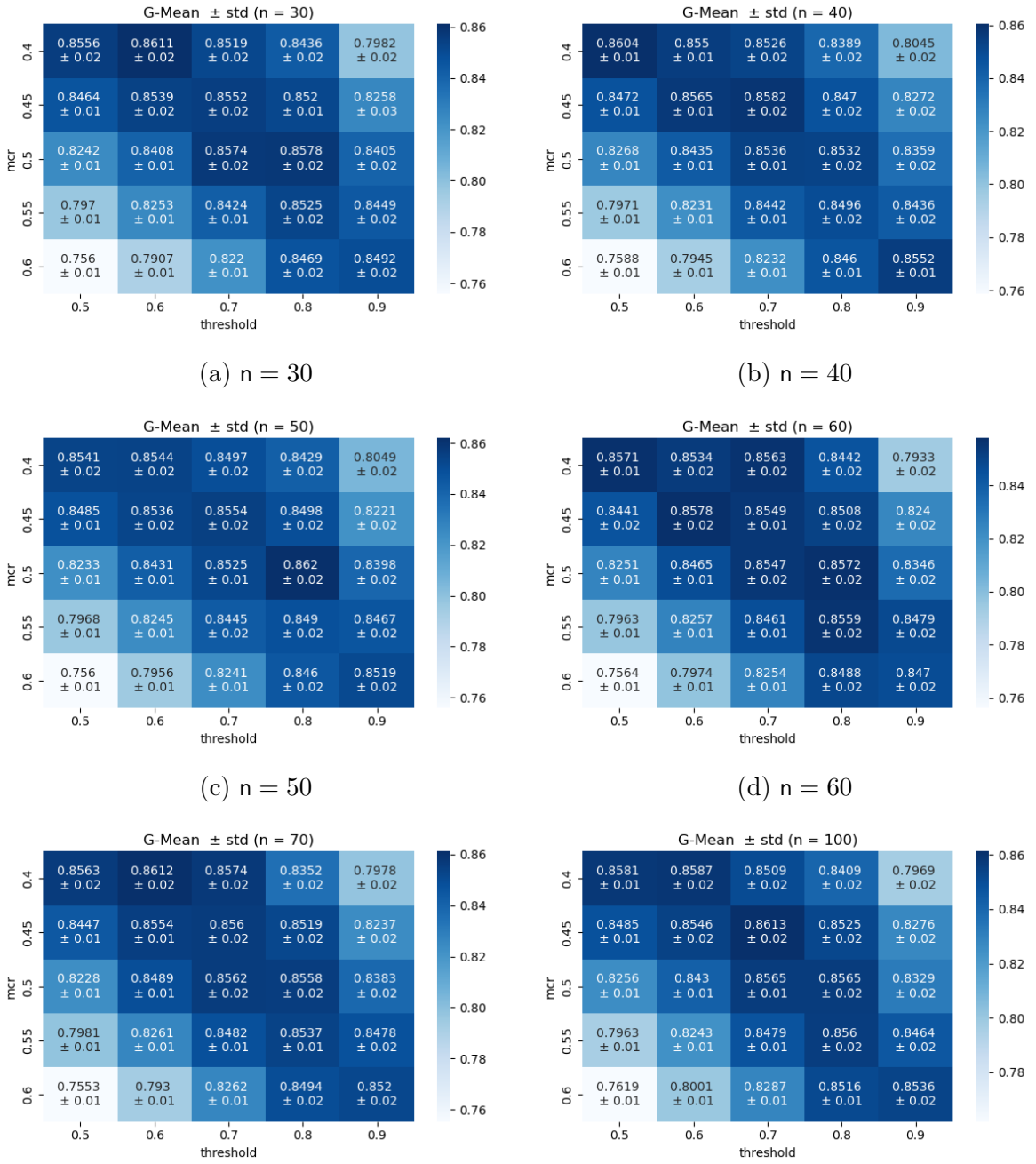
utilizando `xgb` como clasificador base. Para cada configuración se registraron la media y la desviación estándar de GMean en 40 ejecuciones independientes. Los resultados, mostrados como mapas de calor en la figura 3.4, confirman que BUE se mantiene estable y competitivo en un rango amplio de valores.

El umbral de votación (`threshold`) muestra una influencia clara en el rendimiento: valores en torno a 0.8 tienden a producir mejores GMean de forma consistente para todos los tamaños de *ensemble*, logrando un equilibrio entre los *recall* de la clase mayoritaria y minoritaria. Umbrales más bajos incrementan los falsos positivos, mientras que umbrales más altos penalizan el *recall* de la clase minoritaria.

Respecto al tamaño del *ensemble* (`n`), los valores comprendidos entre 40 y 70 ofrecen un rendimiento sólido. Incrementar `n` por encima de 70 aporta ganancias marginales, y `n`=50 muestra el comportamiento más consistente a través de distintas combinaciones de `mcr` y `threshold`. Por otro lado, valores más pequeños como `n`=30 introducen una mayor varianza.

En cuanto a `mcr`, el valor óptimo observado es 0.5, que corresponde a un muestreo equilibrado de ambas clases para cada subconjunto de entrenamiento. Valores inferiores degradan el *recall* de la clase minoritaria, mientras que valores superiores reducen la diversidad de instancias de la clase mayoritaria incluidas en el muestreo.

En conjunto, la configuración seleccionada (`n`=50, `mcr`=0.5, `threshold`=0.8) queda empíricamente validada como robusta, con un buen rendimiento y con baja sensibilidad ante cambios moderados en los hiperparámetros.




(a) n = 30

(b) n = 40

(c) n = 50

(d) n = 60

(e) n = 70

(f) n = 100

Figura 3.4: Análisis de sensibilidad de BUE respecto a sus hiperparámetros. Cada mapa de calor muestra GMean medio ± desviación estándar (40 ejecuciones) para combinaciones de `threshold` (columnas) y `mcr` (filas), con distintos valores de `n`.

### 3.4.2 Análisis estadístico

Para comparar estadísticamente el rendimiento de las técnicas de muestreo se ha utilizado la prueba de Friedman y un método bayesiano, siguiendo la propuesta de Calvo y Santafé Rodrigo [20], [21].

La prueba de Friedman es una técnica de comparaciones múltiples que tiene como objetivo detectar diferencias significativas entre el comportamiento de dos o más muestras evaluadas. El primer paso en la aplicación de la prueba de Friedman es convertir las técnicas evaluadas en rankings, agrupando los valores observados para cada combinación de técnica y algoritmo. Una vez obtenidos los rangos para cada modelo, se aplica el Estadístico de Friedman (Ecuación (3.2)) [34] que sigue una distribución $\chi^2$ con $k-1$ grados de libertad, donde k es el número de técnicas a evaluar, n el número de conjuntos de evaluación y $R_j$ es la suma de los rangos obtenidos por la técnica $j$,

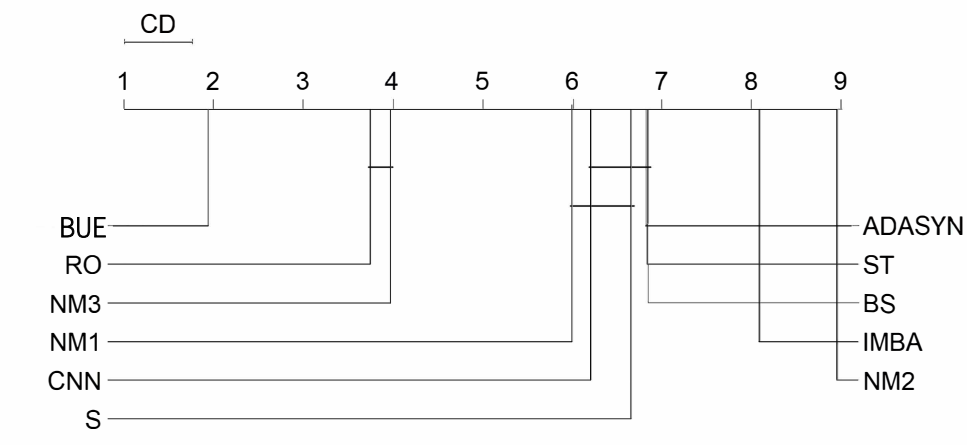
$$\chi_F^2 = \frac{12}{nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3n(k+1) \tag{3.2}$$

Para ambos conjuntos de datos se obtuvo un valor del estadístico de Friedman superior al valor crítico correspondiente, lo que indica la existencia de diferencias estadísticamente significativas entre las técnicas evaluadas. En el conjunto de datos *Seguros*, el estadístico de Friedman alcanzó un valor de $\chi_F^2 = 40.53$, que supera el valor crítico para $k-1=10$ grados de libertad y un $\alpha$ de 0.05, 18.31. De forma análoga, en el conjunto de datos `Fraude` se obtuvo un valor de $\chi_F^2 = 61.03$, que también supera ampliamente el umbral crítico. Por tanto, en ambos casos se rechazó la hipótesis nula de igualdad entre técnicas, y se aplicó la prueba post-hoc de Nemenyi, cuyos resultados se recogen en la figura 3.5.

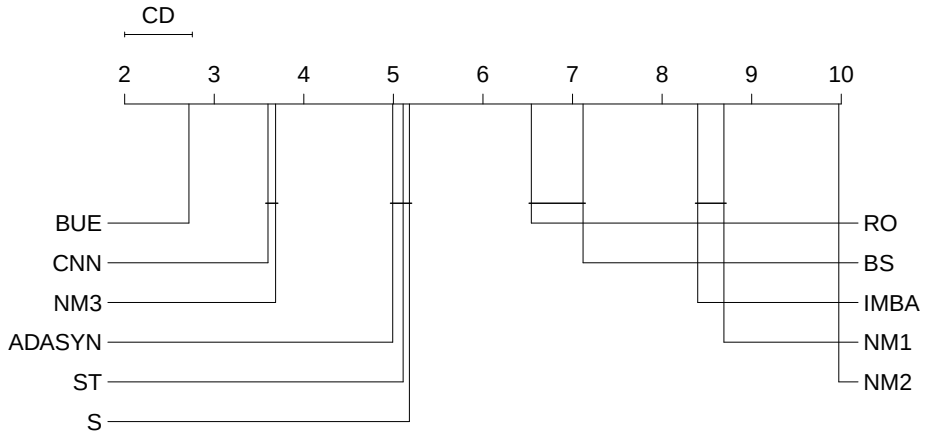
La figura 3.5 muestra el ranking de las técnicas de muestreo, donde el valor más cercano a 1 corresponde a la mejor técnica. La línea horizontal entre las técnicas, CD, es la diferencia crítica, que indica la no existencia de diferencias significativas entre las técnicas conectadas por la línea. En ambos conjuntos de datos, `Seguros` y `Fraude`, BUE obtiene el mejor ranking, mostrando diferencias significativas al no estar conectada con ninguna de las técnicas cercanas. Además, en ambos conjuntos de datos, las técnicas más cercanas a BUE no presentan diferencias significativas entre sí. Estos resultados confirman que BUE es consistentemente la mejor técnica para ambos conjuntos de datos.

Por otro lado, el método bayesiano de comparación de rankings no solo permite estimar la probabilidad de que una técnica supere a otra, sino que también construye un ranking global basado en la posición esperada de cada técnica. Asigna a cada técnica una posición promedio en el ranking y proporciona intervalos de credibilidad que reflejan el grado de incertidumbre asociado a dicha estimación. En la figura 3.6 se muestra la probabilidad de que cada técnica de muestreo sea la mejor, denominada probabilidad de ganar (*probability of winning*), junto con su desviación estándar para los resultados obtenidos con todos los algoritmos ML utilizando GMean como función objetivo. BUE obtiene la mayor probabi-

lidad de ser la mejor técnica en ambos conjuntos de datos, y su desviación estándar no se superpone con las técnicas más cercanas.

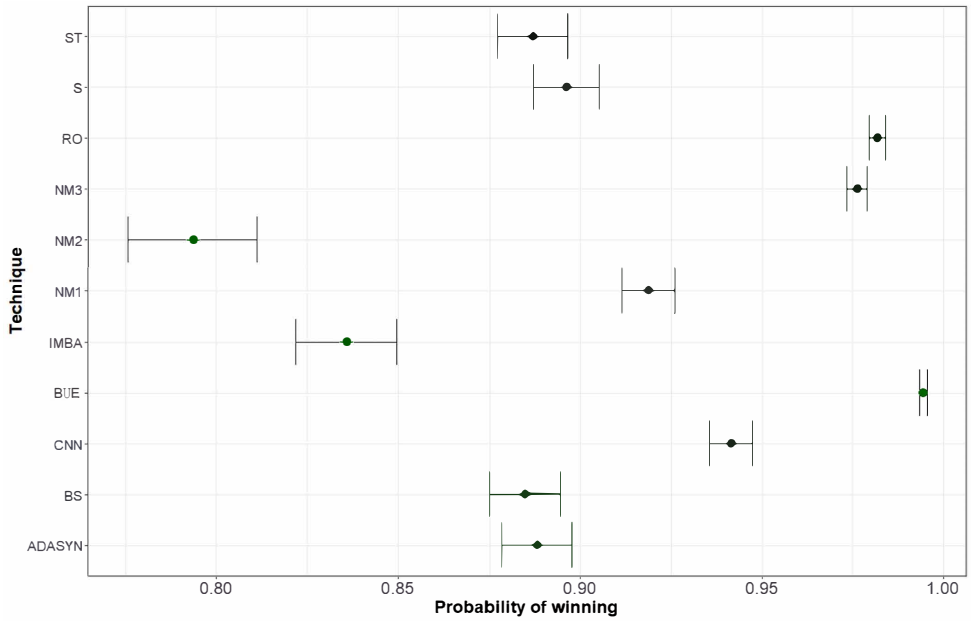



(a) Conjunto de datos `Seguros`

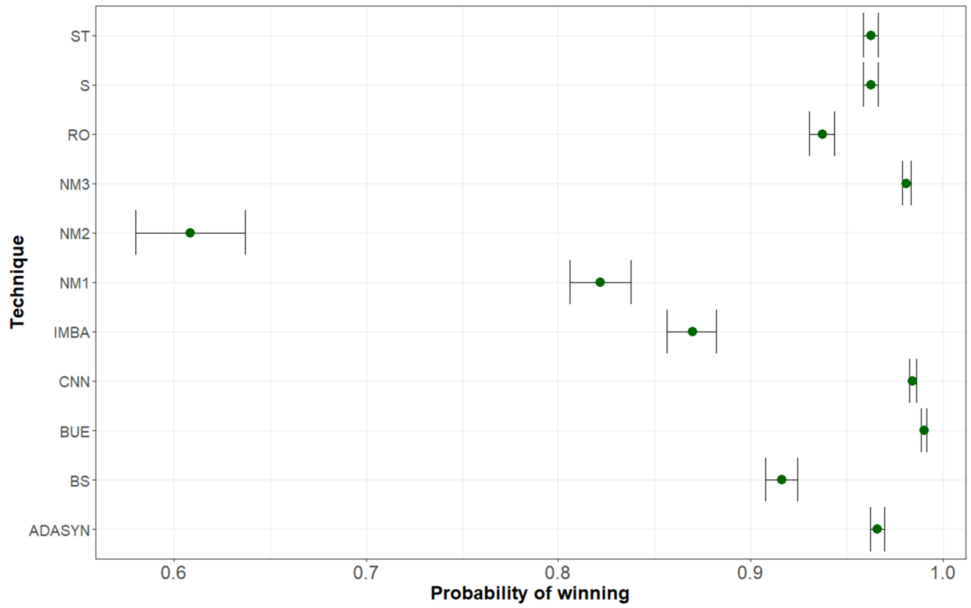



(b) Conjunto de datos `Fraude`

Figura 3.5: Ranking y diferencias críticas de las once técnicas de muestreo.




(a) Conjunto de datos `Seguros`




(b) Conjunto de datos `Fraude`

Figura 3.6: Probabilidad de ganar para las once técnicas de muestreo.

### 3.4.3 Comparativa de los tiempos de ejecución

Se han comparado los tiempos de ejecución de las diferentes técnicas de muestreo. En la tabla 3.1 se muestran los tiempos de ejecución de cada técnica, calculados como la suma del tiempo empleado por la técnica aplicada y el tiempo de entrenamiento del algoritmo de ML.

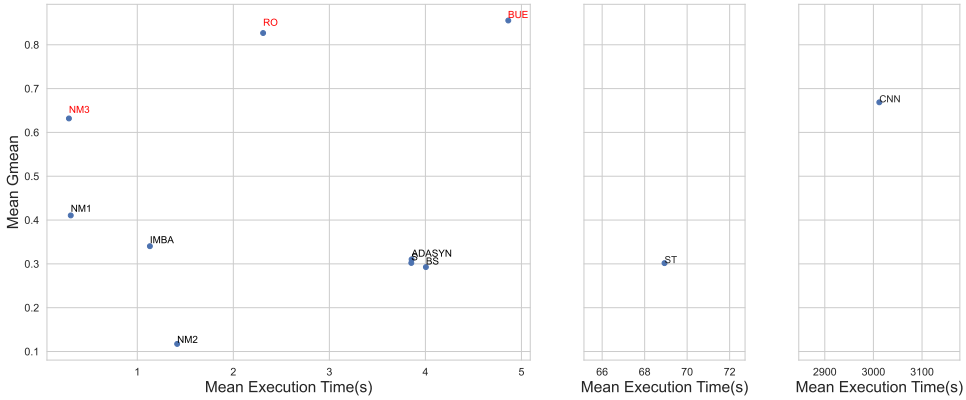
La tabla 3.1 muestra el tiempo de ejecución de cada técnica, calculado como la suma del tiempo empleado por la técnica aplicada y el tiempo de entrenamiento del algoritmo de ML, considerando que, en nuestro caso, se construye un *ensemble* de 50 modelos por cada

algoritmo de ML, mientras que las demás técnicas entrenan únicamente un modelo, lo que permite una comparación justa y equitativa.

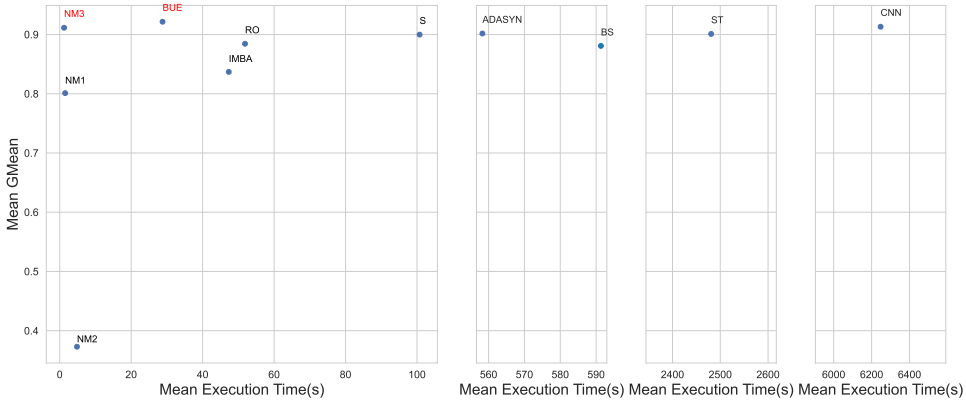
| **Conjunto de datos `Seguros`** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | ST | BS | ADASYN | NM2 | IMBA | NM1 | CNN | NM3 | RO | BUE |
| **Mean Time (s)** | 3.85 | 68.93 | 4.00 | 3.85 | 1.41 | 1.13 | 0.30 | 3012.12 | 0.28 | 2.31 | 4.86 |
| **Median Time (s)** | 3.74 | 68.83 | 3.99 | 3.84 | 1.42 | 1.11 | 0.30 | 3007.51 | 0.29 | 2.31 | 4.83 |
| **Conjunto de datos `Fraude`** | | | | | | | | | | | |
| | S | ST | BS | ADASYN | NM2 | IMBA | NM1 | CNN | NM3 | RO | BUE |
| **Mean Time (s)** | 89.04 | 2602.74 | 635.77 | 597.28 | 4.77 | 39.19 | 1.50 | 6242.41 | 1.19 | 48.37 | 27.79 |
| **Median Time (s)** | 28.70 | 2458.85 | 34.73 | 35.05 | 4.61 | 11.58 | 1.55 | 6414.88 | 1.02 | 12.83 | 17.89 |

Tabla 3.1: Tiempos de ejecución, en segundos, para cada técnica Los tiempos se obtienen como la suma del tiempo de aplicación de la técnica y el tiempo de ejecución del algoritmo de ML.

Los experimentos se realizaron en un sistema con un procesador Intel Core i7-7700, 64 GB de memoria RAM DDR4 y sistema operativo Ubuntu 22.04.1 LTS.




(a) Conjunto de datos `Seguros`




(b) Conjunto de datos `Fraude`

Figura 3.7: Media de valores GMean vs. tiempo medio de ejecución, en segundos, para las once técnicas de muestreo. En rojo se muestran las soluciones no dominadas, es decir, aquellas que no son superadas simultáneamente en ambos criterios por ninguna otra.

Finalmente, en la figura 3.7 se muestra la media de los valores GMean frente al tiempo de ejecución de cada técnica para las técnicas de muestreo analizadas en ambos conjuntos de datos. Los resultados indican que el coste computacional varía significativamente según

el método aplicado. Técnicas como CNN y ST presentan los tiempos de ejecución más altos, especialmente en el conjunto de datos `Fraude`, aunque sus valores de GMean no necesariamente superan a los de alternativas más eficientes.

Por otro lado, BUE, NM3 y RO alcanzan los mejores valores de GMean con tiempos de ejecución considerablemente más bajos, lo que las posiciona como las mejores opciones. En contraste, NM2 e IMBA, aunque son los métodos más rápidos, presentan valores bajos de GMean.

Estos resultados destacan la importancia de considerar tanto la eficiencia computacional como el rendimiento predictivo. BUE y NM3 se encuentran en el frente de Pareto en ambos conjuntos de datos y, en consecuencia, son las mejores alternativas. Aunque BUE tiene tiempos de ejecución más elevados, esto se debe al entrenamiento y evaluación secuencial de los `n` modelos. Sin embargo, BUE admite paralelización, lo que mejoraría significativamente los tiempos de ejecución en una implementación paralela. Por lo tanto, nuestra propuesta ofrece el mejor equilibrio entre precisión y tiempo de ejecución.

## 3.5 Conclusiones

Este capítulo presenta una técnica de muestreo híbrida, BUE, diseñada para abordar datos altamente desbalanceados en problemas de clasificación. Evaluamos la eficacia de nuestra propuesta comparándola con diez técnicas de procesamiento de datos desbalanceados combinadas con diez modelos de clasificación en dos conjuntos de datos del mundo real. Un conjunto de datos `Seguros` para la predicción de siniestros y un conjunto de datos para la detección de fraude financiero. A partir de los resultados se ha comparado el rendimiento de las técnicas de procesamiento de datos desbalanceados ML utilizando métodos bayesianos. En este análisis, BUE obtiene la mejor clasificación y la mayor probabilidad de ser la mejor técnica, seguida de NM3. También se ha demostrado que entre NM3 y las técnicas con resultados similares no existen diferencias significativas.

Nuestra propuesta alcanza los mejores valores de GMean, aunque sus tiempos de ejecución son más altos en comparación con NM3. NM3 se caracteriza por su rapidez en ambos conjuntos de datos, aunque sus resultados en el conjunto de datos `Seguros` son peores que en el conjunto de datos `Fraude`, y siempre peores que los de nuestra propuesta. Esto se debe al funcionamiento de NM3, ya que selecciona ejemplos de la clase mayoritaria en función de su proximidad a los de la clase minoritaria. En el caso del conjunto de datos `Fraude`, los perfiles fraudulentos tienden a compartir características similares, mientras que los perfiles no fraudulentos son en su mayoría homogéneos. Este comportamiento favorece a NM3, ya que su estrategia de selección es más eficaz en contextos en los que las clases están bien diferenciadas, como ocurre en el conjunto de datos `Fraude`. Sin embargo, en el conjunto de datos `Seguros`, donde las características de las clases son más heterogéneas, su rendimiento es limitado.

Para cuantificar la mejora, comparamos nuestra propuesta con la técnica de referencia para

datos desbalanceados (IMBA) utilizando XGBoost, el clasificador con mejor rendimiento en ambos conjuntos de datos. En el conjunto de datos `Seguros`, nuestra propuesta mejora en un 152.94 % el valor medio obtenido en GMean por IMBA, mientras que en el conjunto de datos `Fraude` logra una mejora del 15.85 %.

Esto se traduce en un desempeño superior en la clasificación de ambas clases, lo cual es crucial en aplicaciones financieras y de seguros, donde la identificación de reclamaciones o transacciones fraudulentas debe realizarse sin penalizar injustamente a clientes legítimos. Además, los tiempos de ejecución de BUE son competitivos en comparación con otras técnicas. Con estos resultados, nuestra propuesta es capaz de abordar problemas de clasificación binaria extremadamente desbalanceados.

Las ventajas de nuestra propuesta son su buen rendimiento, su capacidad de configuración y su posibilidad de paralelización, reduciendo aún más el tiempo de ejecución. Por contra los parámetros de BUE deben ajustarse empíricamente. El alto número de parámetros y sus relaciones no lineales requieren una búsqueda heurística para encontrar los mejores valores, aumentando el tiempo total necesario para ajustar la técnica. Asimismo, el potencial de este enfoque para problemas multiclase debe ser estudiado más a fondo.

BUE muestra un excelente potencial para problemas de clasificación con datos desbalanceados en la predicción de reclamaciones de seguros y la detección de fraudes, y esperamos que mantenga su desempeño en otros conjuntos de datos para problemas de clasificación. Como trabajo futuro, planeamos aplicar esta técnica en otros proyectos de nuestro grupo, en particular en conjuntos de datos médicos.

# Capítulo 4

# Clasificador en cascada

La obtención de datos en el contexto de Seguros es una tarea difícil. Por ello, los trabajos se comenzaron con datos provenientes del proyecto `GenObIA` de los cuales ya se disponía de un amplio registro de muestras. `GenObIA` [30], es una iniciativa multidisciplinar centrada en el análisis de factores de riesgo asociados a enfermedades crónicas no transmisibles, con especial énfasis en el sobrepeso y la obesidad. En este contexto, se desarrolló un clasificador en cascada para predecir si un individuo va a padecer `sobrepeso/obesidad` en función de sus hábitos y patologías. Habitualmente se considera que una persona tiene `sobrepeso/obesidad` si su Índice de Masa Corporal, IMC, es igual o superior a 25, por lo que se trata de un problema de clasificación binaria. El IMC se calcula dividiendo el peso de la persona (en kilogramos) entre el cuadrado de su altura (en metros):

$$\text{IMC} = \frac{\text{peso (kg)}}{\text{altura (m)}^2} \tag{4.1}$$

En este capítulo se describe el desarrollo de este clasificador en cascada, que sirve como base metodológica para la propuesta final de tesis, detallada en el capítulo 6.

## 4.1 Propuesta

Para abordar el desarrollo del clasificador, se llevó a cabo una fase inicial de validación cruzada utilizando nueve algoritmos de ML. El objetivo era obtener una base de resultados como punto de partida. Los algoritmos utilizados fueron Random Forest, Logistic Regression, Gradient Boosting, Bernoulli Naive Bayes, Extra Trees, Decision Tree, Bagging Classifier y AdaBoost

Con Random Forest y Gradient Boosting se llegó a obtener una precisión de 0.70, pero con valores TPR bajos, 0.72 y 0.7 respectivamente. Esto suponía un problema ya que no diagnosticar un caso de `sobrepeso/obesidad` tiene un riesgo elevado frente a clasificar una persona `sin sobrepeso` como `sobrepeso/obesidad`. Teniendo esto en cuenta, se diseñó un clasificador con abstención, el clasificador en cascada, con el objetivo de incrementar la

tasa de acierto de personas con `sobrepeso/obesidad` sin disminuir la tasa de acierto de personas `sin sobrepeso`. Este tipo de clasificadores es útil en contextos donde es preferible abstenerse de decidir antes que cometer errores graves.

El clasificador en cascada consta de $n$ niveles, donde cada nivel incorpora un algoritmo de ML. Para cada nivel se establecen dos umbrales, uno para determinar cuándo considerar a una persona con `sobrepeso/obesidad` y otro para considerarla como `sin sobrepeso`. Estos umbrales se aplican sobre la puntuación de pertenencia a una clase (`sobrepeso/obesidad` y `sin sobrepeso`) generada por los modelos de ML. Cada nivel del clasificador en cascada es entrenado con el mismo conjunto de entrenamiento. En nuestro caso utilizamos un 70 % del conjunto de datos para entrenamiento y un 30 % de los datos para test.

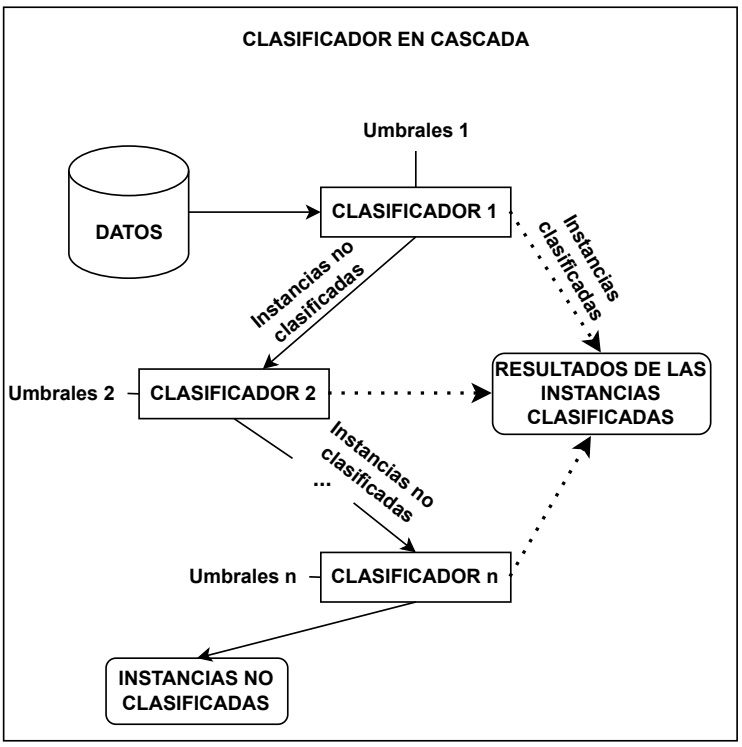



Figura 4.1: Estructura del clasificador en cascada entrenado. Cada nivel está compuesto por un algoritmo de ML junto con sus umbrales correspondientes, y clasifica las instancias de entrada en función de estos. Las instancias que cumplen los umbrales se consideran como clasificadas (flechas continuas), mientras que las que no lo hacen se etiquetan como no clasificadas (flechas discontinuas) y se transfieren al siguiente nivel. Como resultado se obtienen las instancias clasificadas y en el caso de haberlas, instancias no clasificadas.

Su funcionamiento, una vez entrenado y establecidos los umbrales de cada nivel, es el siguiente (figura 4.1 ):

a) El clasificador 1 recibe los datos y procede a evaluarlos.

b) Los resultados se dividen en dos grupos: por un lado, los casos no clasificados, aquellos en los que la puntuación de pertenencia a una clase no alcanza los umbrales

establecidos; y, por otro, los casos clasificados.

c) Los casos no clasificados se convierten en la entrada del siguiente nivel y son evaluados por el algoritmo de ML.

d) Se repite el paso b).

e) En aquellos casos en los que, tras aplicar todos los niveles del clasificador en cascada, las instancias no consiguen ser clasificadas según los umbrales establecidos, se consideran no clasificadas. En estos casos, fueron los profesionales médicos quienes evaluaron la asignación a una u otra clase.

## 4.2 Conjunto de datos

Para la realización de este proyecto, el consorcio `GenObIA` proporcionó los datos de individuos pertenecientes a centros asociados (hospitales, universidades y centros deportivos). Estos se obtuvieron mediante encuestas que recopilaban información sobre hábitos de actividad física, nutrición y patologías.

El conjunto de datos original estaba compuesto por un total de 1179 muestras y 93 variables:

- 1 identificador de individuo.
- 13 variables de información general sobre el individuo, como el peso, edad, educación, estrés, etc.
- 7 variables relacionadas con el consumo de bebidas alcohólicas, haciendo una distinción entre bebidas destiladas y fermentadas.
- 7 variables sobre hábitos del tabaco, como número de cigarrillos, pipas, puros y, en el caso de exfumadores, el tiempo transcurrido desde que dejaron de fumar.
- 15 variables sobre patologías, como tipos de cáncer, apnea del sueño, diabetes mellitus tipo 2, entre otras.
- 34 variables sobre hábitos nutricionales como información sobre las porciones de diferentes tipos de alimentos y los puntos de adherencia a la dieta mediterránea según estas porciones.
- 16 variables relacionadas con el ejercicio físico y su intensidad.

Tras una etapa de preprocesamiento y reuniones con el equipo médico, se redujo el número de variables a 42 (tabla 4.1). Esta tabla representa cada una de las variables resultantes con su descripción y el grupo al que pertenecen. Se eliminaron aquellas variables redundantes relacionadas con los hábitos nutricionales, por estar directamente relacionadas con la variable Adherencia a la Dieta Mediterránea. Por el mismo motivo, se eliminaron variables que contenían información del ejercicio físico, ya que estas variables estaban relacionadas

con las variables IPAQ y CalIPAQ, que recogen información sobre el ejercicio realizado y las calorías quemadas, respectivamente. Esta última también acabó siendo eliminada, ya que emplea el peso como variable para su cálculo. Dado que la variable objetivo también se basaba en el peso, su inclusión podría inducir a los modelos a "hacer trampas", identificando relaciones artificiales y favoreciendo predicciones basadas en una variable que, en la práctica, ya contiene parte de la respuesta.

Del mismo modo que para el conjunto de datos en `Seguros` del capítulo anterior, se abordaron rigurosamente las consideraciones éticas del conjunto de datos con el fin de garantizar la protección de la privacidad de los participantes y el cumplimiento de las normas éticas. Todos los participantes proporcionaron su consentimiento informado, y el estudio fue aprobado por el Comité Regional de Ética en la Investigación con Medicamentos de la Comunidad de Madrid (Código de aprobación: 06/2018, Fecha de aprobación: 28 de junio de 2018). Los datos fueron recolectados de forma anónima y la información sensible, incluidos los factores biológicos y sociodemográficos, se manejó con estricta confidencialidad.

## 4.3 Marco experimental y metodología

Para evaluar el desempeño del clasificador en cascada se ha comparado su rendimiento frente a nueve algoritmos de ML con abstención, es decir, aplicando umbrales para clasificar cada instancia a una clase. Tanto estos algoritmos como el clasificador en cascada han sido evaluados sobre cuatro variantes del conjunto de datos, cada una con una combinación distinta de variables:

- La variante `38` incluye todas las variables tras el proceso de filtrado, excepto cuatro variables relacionadas con hábitos nutricionales.
- La variante `37c` contiene todas las variables del caso anterior excepto el centro. Se decidió eliminar esta variable porque los modelos entrenados con la variante 38 tendían a clasificar en función del lugar de recogida de los datos (el centro en el que se había recogido la información del individuo) introduciendo un sesgo en las predicciones.
- La variante `37a` es similar, pero excluye la variable edad, con el fin de evitar el sesgo asociado a esta variable, ya que las personas jóvenes tienden a presentar una menor prevalencia de sobrepeso que las de mayor edad.
- La variante `36` contiene todas las variables de la variante `38` excepto edad y centro.
- La variante `26` corresponde a un conjunto de datos sin patologías ni centro, incorporando además nuevos hábitos nutricionales. Se decidió eliminar las patologías, ya que algunas eran consecuencia y no causa del sobrepeso, lo que podría afectar la interpretación del modelo.

En la tabla 4.2 se muestran las variables utilizadas para cada variante.

| ID | Variable | Descripción | Tipo |
|---|---|---|---|
| 1 | sex | Sexo de la persona | Información general |
| 2 | age | Edad de la persona en años | Información general |
| 3 | center | Origen del dato | Información general |
| 4 | pop | Volumen de la población donde reside la persona | Información general |
| 5 | edu | Nivel académico alcanzado por la persona | Información general |
| 6 | earning | Nivel de ingresos de la persona | Información general |
| 7 | job | Tipo de trabajo realizado por la persona | Información general |
| 8 | stress | estrés autopercibido por la persona | Información general |
| 9 | sleep.8 | La persona duerme más de ocho horas | Información general |
| 10 | spirit | La persona toma bebidas espirituosas | Información general |
| 11 | spiritWEEK | Unidades de bebidas espirituosas por semana | Bebidas alcohólicas |
| 12 | wine_beer | La persona bebe cerveza o vino | Bebidas alcohólicas |
| 13 | beerWEEK | Unidades de cerveza por semana | Bebidas alcohólicas |
| 14 | wineWEEK | Unidades de vino tinto por semana | Bebidas alcohólicas |
| 15 | whiteWEEK | Unidades de vino blanco por semana | Bebidas alcohólicas |
| 16 | pinkWEEK | Unidades de vino rosado por semana | Bebidas alcohólicas |
| 17 | smoke | La persona fuma | Tabaco |
| 18 | nsmoke | Cigarrillos consumidos al día | Tabaco |
| 19 | pipe | Tabaco de pipa consumido al día | Tabaco |
| 20 | cigar | Puros consumidos al día | Tabaco |
| 21 | exsmokerY | Tiempo transcurrido desde que un fumador dejó de fumar en años | Tabaco |
| 22 | exsmokerUNK | La persona ha dejado de fumar pero no recuerda desde cuándo. | Tabaco |
| 23 | cancer | La persona ha padecido o padece cáncer | Patologías |
| 24 | cancer_mam | La persona ha padecido o padece cáncer de mama. | Patologías |
| 25 | cancer_col | La persona ha padecido o padece cáncer de colon | Patologías |
| 26 | cancer_pros | La persona ha padecido o padece cáncer de próstata. | Patologías |
| 27 | cancer_lung | La persona ha padecido o padece cáncer de pulmón. | Patologías |
| 28 | cancer_other | La persona ha padecido o padece otro tipo de cáncer. | Patologías |
| 29 | heart_attack | La persona ha sufrido un infarto agudo de miocardio. | Patologías |
| 30 | heart_angina | La persona ha sufrido una angina de pecho. | Patologías |
| 31 | heart_failure | La persona ha sufrido un fallo cardíaco | Patologías |
| 32 | diabetes | La persona padece diabetes mellitus de tipo 2 | Patologías |
| 33 | metabolic_syn | La persona padece síndrome metabólico | Patologías |
| 34 | apnea | La persona padece apnea del sueño | Patologías |
| 35 | asthma | La persona tiene asma | Patologías |
| 36 | COPD | La persona padece una enfermedad pulmonar obstructiva crónica. | Patologías |
| 37 | ADH | La persona tiene adherencia a la dieta mediterránea. | Hábitos nutricionales |
| 38 | IPAQ | Puntuaciones de los sujetos en el Cuestionario Internacional de Actividad Física (IPAQ) | Ejercicio físico |
| 39 | vege | Raciones de verduras consumidas por el individuo al día | Hábitos nutricionales |
| 40 | soda | Porciones de bebidas gaseosas y/o azucaradas consumidas por el sujeto al día | Hábitos nutricionales |
| 41 | legume | Raciones de legumbres consumidas por el sujeto a la semana | Hábitos nutricionales |
| 42 | milk | Porciones de leche o productos lácteos consumidos por el sujeto al día | Hábitos nutricionales |

Tabla 4.1: Representación de las columnas que forman el conjunto de datos utilizado. Variables empleadas por el clasificador en cascada tras el preprocesamiento del conjunto de datos inicial.

| Variante | Variables utilizadas |
|---|---|
| `38` | [1-38] |
| `37c` | [1-2] ∪ [4-38] |
| `37a` | 1 ∪ [3-38] |
| `36` | 1 ∪ [4-38] |
| `26` | [1-2] ∪ [4-11] ∪ [13-22] ∪ [37-42] |

Tabla 4.2: Variables utilizadas para cada variante del conjunto de datos.

Para cada combinación de algoritmo y variante del conjunto de datos se realizaron 100 ejecuciones, con el fin de asegurar una evaluación robusta del rendimiento de cada uno de ellos. Los algoritmos empleados en esta comparativa son: Random Forest, Logistic Regression, Gradient Boosting, Bernoulli Naive Bayes, Extra Trees, Decision Tree, Bagging Classifier y AdaBoost, además del clasificador en cascada implementado.

Los umbrales de clasificación utilizados para cada uno de los algoritmos de ML y el clasificador en cascada fueron: 70 % para considerar la instancia como sobrepeso y 80 % como `sin sobrepeso`; en el caso de no superar dichos umbrales, la instancia se considera como no clasificada.

Para el clasificador en cascada se implementaron 2 versiones, una con tres niveles y otra con cuatro niveles, con el fin de ver la evolución del rendimiento y el porcentaje de instancias clasificadas según se añaden niveles. La primera emplea Gradient Boosting, Random Forest y Logistic Regression, mientras que la segunda utiliza Gradient Boosting, Random Forest, Bagging Classifier y Logistic Regression, seleccionados por haber obtenido los mejores resultados en el proceso previo de validación cruzada. Se estableció el límite en cuatro niveles, ya que a partir del quinto, al incorporar algoritmos con los mismos umbrales de clasificación, aumentaba el número de instancias clasificadas erróneamente en ambas clases, lo que provocaba una disminución del *recall*. Aunque se incrementaba el porcentaje de muestras clasificadas, también aumentaba el número de errores, debido a que las instancias que no se clasificaban en los primeros niveles presentaban una mayor incertidumbre y resultaban más difíciles de clasificar.

Para evaluar el rendimiento de los algoritmos se utilizaron dos métricas adicionales aparte de las clásicas: calidad de clasificación ($Q_{\text{class}}$) y la calidad de rechazo o abstención ($Q_{\text{rej}}$). Estas métricas permiten cuantificar tanto la efectividad del clasificador como su capacidad para abstenerse en situaciones de baja certeza [29, 63].

- **Calidad de la clasificación** ($Q_{\text{class}}$): mide la proporción de decisiones correctas, incluyendo tanto las clasificaciones como las abstenciones correctas, sobre el total de instancias.El valor de $Q_{\text{class}}$ se encuentra entre 0 y 1, siendo más cercano a 1 cuando el clasificador toma decisiones correctas, y más próximo a 0 cuando la mayoría de las decisiones son erróneas.

$$Q_{\text{class}} = \frac{C + R_{\text{cor}}}{N}$$

donde:

- $C$: número de instancias correctamente clasificadas
- $N$: número total de instancias
- $R_{\text{cor}}$: número de rechazos acertados. Se considera un rechazo acertado cuando la puntuación de pertenencia a clase se sitúa entre 0.5 y el umbral de decisión correspondiente a la clase, y la instancia pertenece en realidad a la clase opuesta. Es decir, bajo un umbral estándar de 0.5 (habitualmente empleado por defecto en los algoritmos de clasificación) dicha instancia habría sido clasificada erróneamente, y por tanto el rechazo se considera correcto.

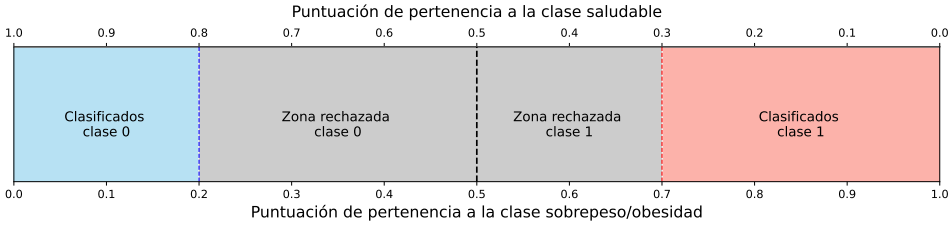



Figura 4.2: Intervalos de clasificación y rechazo

- **Calidad del rechazo** ($Q_{\text{rej}}$): evalúa la calidad de la abstención al medir si las instancias rechazadas tienden a coincidir con casos que el modelo habría clasificado erróneamente. Un valor cercano a 0 indica que el modelo está rechazando muestras que habría clasificado correctamente; por el contrario, un valor cercano o superior a 1 indica que el rechazo se está efectuando correctamente, ya que la mayoría de las instancias rechazadas habrían sido clasificadas erróneamente.

$$Q_{\text{rej}} = \frac{R_{\text{cor}}}{N_{\text{rej}}} \cdot \frac{N}{E + R_{\text{cor}}}$$

donde:

- $N_{\text{rej}}$: número total de instancias rechazadas
- $N$: número total de instancias
- $E$: número de errores cometidos por el modelo
- $R_{\text{cor}}$: número de rechazos correctos

## 4.4 Análisis de resultados

La tabla 4.3 recoge los resultados obtenidos tras 100 ejecuciones para cada combinación de algoritmo de ML y variante del conjunto de datos. En ella se muestra la precisión

del mejor y del peor modelo, la media de precisión, la desviación estándar (STD) que indica la variabilidad del rendimiento, la precisión y el *recall* medios para cada clase: clase 0 (personas clasificadas como `sin sobrepeso`) y clase 1 (personas clasificadas con `sobrepeso/obesidad`), el porcentaje de instancias clasificadas (CLAS) y la calidad de clasificación ($Q_{\text{class}}$) y rechazo ($Q_{\text{rej}}$).

Para comparar los resultados de una forma más clara, se representaron los valores medios obtenidos por cada algoritmo, agregados sobre las distintas variantes de datos, en un mapa de calor (figura 4.3). En ella se muestran distintas métricas de evaluación como la precisión media, *recall*, precisión, la proporción de instancias clasificadas, $Q_{\text{class}}$ y $Q_{\text{rej}}$. Los clasificadores en cascada (CASCADA 3 y CASCADA 4) destacan por obtener valores

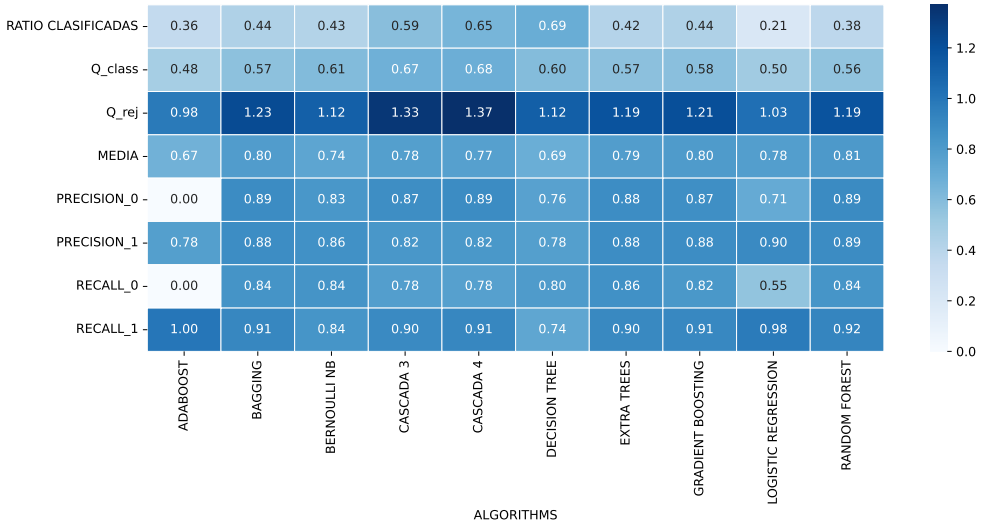



Figura 4.3: Mapa de calor para los diferentes algoritmos que muestra las media de para diferentes metricas agregadas por las diferentes variantes de conjunto de datos.

equilibrados en todas las métricas, con un alto ratio de instancias clasificadas (0.59 y 0.65, respectivamente), superando a modelos como Adaboost (0.36) o Logistic Regression (0.21). Además de clasificar un mayor número de instancias, logran obtener los mejores valores en $Q_{\text{class}}$, mientras que otros algoritmos con menos ratio de clasificación apenas llegan a superar 0.55.

En cuanto a la calidad de rechazo, los modelos en cascada obtienen también los mejores valores con 1.33 y 1.37. Estos resultados indican que los umbrales establecidos están funcionando en la identificación de ejemplos conflictivos o ambiguos: se rechazan principalmente las instancias que podrían haber sido mal clasificadas, mejorando así la calidad general del clasificador.

En cuanto a la precisión y el *recall*, los modelos en cascada muestran un buen equilibrio entre clases. Ambos alcanzan valores similares de precisión para las clases 0 y 1 (en torno a 0.82–0.89), y mantienen niveles de *recall* bastante equilibrados (entre 0.78 y 0.91). Además, destacan especialmente por su alto *recall* en la clase de `sobrepeso/obesidad`. Es cierto que algoritmos como Logistic Regression o Random Forest obtienen mejores

| Algoritmos | Var | Mejor | Peor | Media | std | Prec 0 | Prec 1 | Recall 0 | Recall 1 | Clas (%) | $Q_{\text{class}}$ | $Q_{\text{rej}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADB | 38 | 0.7132 | 0.5308 | 0.6161 | 0.0369 | 0.0000 | 0.7132 | 0.0000 | 1.00 | 45.7100 | 0.5014 | 0.9184 |
| BG | 38 | 0.8714 | 0.6715 | 0.7955 | 0.0334 | 0.8727 | 0.8706 | 0.8136 | 0.91 | 46.4300 | 0.5456 | 1.2706 |
| BNB | 38 | 0.8417 | 0.6605 | 0.7470 | 0.0363 | 0.8039 | 0.8696 | 0.8200 | 0.85 | 44.7300 | 0.6292 | 1.1113 |
| DT | 38 | 0.8050 | 0.6061 | 0.7007 | 0.0344 | 0.7875 | 0.8228 | 0.8182 | 0.79 | 64.6800 | 0.7252 | 0.9702 |
| ET | 38 | 0.8750 | 0.7252 | 0.7996 | 0.0311 | 0.9400 | 0.8226 | 0.8103 | 0.94 | 41.1700 | 0.6122 | 1.1622 |
| GB | 38 | 0.8651 | 0.7415 | 0.7997 | 0.0293 | 0.9091 | 0.8415 | 0.7547 | 0.94 | 45.7300 | 0.6157 | 1.1906 |
| LR | 38 | 0.8776 | 0.6596 | 0.7860 | 0.0452 | 0.0000 | 0.8776 | 0.0000 | 1.00 | 19.1500 | 0.5287 | 0.9602 |
| RF | 38 | 0.8923 | 0.7222 | 0.8219 | 0.0318 | 0.8852 | 0.8986 | 0.8852 | 0.89 | 43.4100 | 0.5445 | 1.2856 |
| CC 3 | 38 | 0.8642 | 0.7092 | 0.7889 | 0.0284 | 0.8889 | 0.8485 | 0.7887 | 0.92 | 60.3300 | 0.7095 | 1.2919 |
| CC 4 | 38 | 0.8394 | 0.7109 | 0.7810 | 0.0252 | 0.8553 | 0.8291 | 0.7647 | 0.89 | **67.6100** | 0.6834 | 1.4031 |
| ADB | 37a | 0.8958 | 0.5882 | 0.7545 | 0.0634 | 0.0000 | 0.8958 | 0.0000 | 1.00 | 16.8000 | 0.4644 | 1.0342 |
| BG | 37a | 0.8790 | 0.7348 | 0.7980 | 0.0280 | 0.8723 | 0.8831 | 0.8200 | 0.91 | 43.4700 | 0.5943 | 1.2357 |
| BNB | 37a | 0.8358 | 0.6600 | 0.7486 | 0.0373 | 0.8133 | 0.8644 | 0.8841 | 0.78 | 47.3000 | 0.5932 | 1.1693 |
| DT | 37a | 0.7399 | 0.5989 | 0.6753 | 0.0313 | 0.7132 | 0.7766 | 0.8142 | 0.66 | 75.0500 | 0.5933 | 1.0977 |
| ET | 37a | 0.8712 | 0.6950 | 0.7868 | 0.0317 | 0.8448 | 0.8919 | 0.8596 | 0.88 | 45.0400 | 0.5741 | 1.2511 |
| GB | 37a | 0.8718 | 0.7422 | 0.8032 | 0.0317 | 0.8039 | 0.9242 | 0.8913 | 0.85 | 40.9300 | 0.5660 | 1.1990 |
| LR | 37a | 0.9500 | 0.5435 | 0.7734 | 0.0696 | 1.0000 | 0.9412 | 0.7500 | 1.00 | 7.9800 | 0.5577 | 0.9351 |
| RF | 37a | 0.8980 | 0.7563 | 0.8186 | 0.0308 | 0.8974 | 0.8983 | 0.8537 | 0.92 | 35.6500 | 0.5916 | 1.1725 |
| CC 3 | 37a | 0.8405 | 0.6900 | 0.7661 | 0.0332 | 0.8689 | 0.8235 | 0.7465 | 0.91 | 57.2300 | 0.6519 | 1.3248 |
| CC 4 | 37a | 0.8391 | 0.6860 | 0.7591 | 0.0284 | 0.8986 | 0.8000 | 0.7470 | 0.92 | **62.6800** | 0.6842 | 1.3182 |
| ADB | 37c | 0.7092 | 0.5368 | 0.6223 | 0.0380 | 0.0000 | 0.7092 | 0.0000 | 1.00 | 47.9900 | 0.4831 | 0.9593 |
| BG | 37c | 0.8740 | 0.7244 | 0.7981 | 0.0329 | 0.8475 | 0.8971 | 0.8772 | 0.87 | 44.2100 | 0.5756 | 1.2180 |
| BNB | 37c | 0.8433 | 0.6623 | 0.7495 | 0.0370 | 0.8413 | 0.8451 | 0.8281 | 0.85 | 47.8300 | 0.6074 | 1.1708 |
| DT | 37c | 0.7824 | 0.6164 | 0.7043 | 0.0338 | 0.7736 | 0.7931 | 0.8200 | 0.74 | 68.2600 | 0.6308 | 1.1286 |
| ET | 37c | 0.8605 | 0.7077 | 0.7881 | 0.0326 | 0.8033 | 0.9118 | 0.8909 | 0.83 | 44.9800 | 0.5756 | 1.2024 |
| GB | 37c | 0.8723 | 0.7483 | 0.8042 | 0.0271 | 0.8500 | 0.8889 | 0.8500 | 0.88 | 48.6500 | 0.5880 | 1.2629 |
| LR | 37c | 0.8837 | 0.6829 | 0.7910 | 0.0410 | 0.8750 | 0.8857 | 0.6364 | 0.96 | 29.8500 | 0.4807 | 1.0975 |
| RF | 37c | 0.8824 | 0.7483 | 0.8197 | 0.0288 | 0.8958 | 0.8732 | 0.8269 | 0.92 | 41.5600 | 0.5661 | 1.2199 |
| CC 3 | 37c | 0.8533 | 0.7198 | 0.7926 | 0.0279 | 0.8667 | 0.8404 | 0.8387 | 0.86 | 63.3400 | 0.6593 | 1.4149 |
| CC 4 | 37c | 0.8730 | 0.7143 | 0.7842 | 0.0287 | 0.9268 | 0.8318 | 0.8085 | 0.93 | **66.8200** | 0.6892 | 1.3915 |
| ADB | 36 | 0.8800 | 0.5592 | 0.7350 | 0.0630 | 0.0000 | 0.8800 | 0.0000 | 1.00 | 17.5700 | 0.4672 | 1.0240 |
| BG | 36 | 0.8852 | 0.6986 | 0.7968 | 0.0335 | 0.9032 | 0.8667 | 0.8750 | 0.89 | 43.0800 | 0.5951 | 1.2200 |
| BNB | 36 | 0.8547 | 0.6667 | 0.7458 | 0.0356 | 0.8254 | 0.8889 | 0.8966 | 0.81 | 43.9400 | 0.6302 | 1.0958 |
| DT | 36 | 0.7657 | 0.5714 | 0.6892 | 0.0360 | 0.7653 | 0.7662 | 0.8065 | 0.71 | 56.6100 | 0.5117 | 1.1211 |
| ET | 36 | 0.8626 | 0.6835 | 0.7703 | 0.0335 | 0.8413 | 0.8824 | 0.8689 | 0.85 | 45.1700 | 0.5744 | 1.2111 |
| GB | 36 | 0.8919 | 0.7295 | 0.8038 | 0.0336 | 0.9167 | 0.8800 | 0.7857 | 0.95 | 38.0500 | 0.5300 | 1.1881 |
| LR | 36 | 0.9167 | 0.5000 | 0.7852 | 0.0683 | 0.7500 | 0.9500 | 0.7500 | 0.95 | 16.5900 | 0.4759 | 1.0714 |
| RF | 36 | 0.8772 | 0.7252 | 0.8114 | 0.0349 | 0.8889 | 0.8696 | 0.8163 | 0.92 | 38.9700 | 0.5499 | 1.2301 |
| CC 3 | 36 | 0.8345 | 0.6832 | 0.7750 | 0.0317 | 0.8525 | 0.8205 | 0.7879 | 0.87 | 51.3200 | 0.6884 | 1.2740 |
| CC 4 | 36 | 0.8393 | 0.6793 | 0.7623 | 0.0320 | 0.8493 | 0.8316 | 0.7949 | 0.87 | **60.0900** | 0.6834 | 1.3538 |
| ADB | 26 | 0.6887 | 0.5096 | 0.6062 | 0.0336 | 0.0000 | 0.6887 | 0.0000 | 1.00 | 49.9300 | 0.4677 | 0.9797 |
| BG | 26 | 0.9032 | 0.6746 | 0.7881 | 0.0326 | 0.9348 | 0.8846 | 0.8269 | 0.95 | 42.4500 | 0.5456 | 1.1956 |
| BNB | 26 | 0.8395 | 0.5926 | 0.7102 | 0.0460 | 0.8529 | 0.8298 | 0.7838 | 0.88 | 29.8600 | 0.5824 | 1.0439 |
| DT | 26 | 0.7463 | 0.6000 | 0.6794 | 0.0318 | 0.7634 | 0.7299 | 0.7299 | 0.76 | 81.9900 | 0.5607 | 1.2874 |
| ET | 26 | 0.9247 | 0.6923 | 0.7990 | 0.0408 | 0.9722 | 0.8947 | 0.8537 | 0.98 | 33.4700 | 0.5356 | 1.1004 |
| GB | 26 | 0.8605 | 0.7014 | 0.7906 | 0.0308 | 0.8667 | 0.8551 | 0.8387 | 0.88 | 46.4300 | 0.6091 | 1.1881 |
| LR | 26 | 0.8478 | 0.6667 | 0.7766 | 0.0355 | 0.9091 | 0.8286 | 0.6250 | 0.96 | 31.2400 | 0.4704 | 1.1102 |
| RF | 26 | 0.8889 | 0.7107 | 0.7944 | 0.0337 | 0.8929 | 0.8868 | 0.8065 | 0.94 | 30.5700 | 0.5505 | 1.0305 |
| CC 3 | 26 | 0.8315 | 0.7250 | 0.7782 | 0.0245 | 0.8919 | 0.7885 | 0.7500 | 0.91 | 62.0400 | 0.6540 | 1.3414 |
| CC 4 | 26 | 0.8541 | 0.7030 | 0.7742 | 0.0296 | 0.9028 | 0.8230 | 0.7647 | 0.93 | **65.4700** | 0.6837 | 1.3765 |

Tabla 4.3: Resultados de las 100 ejecuciones de cada algoritmo de ML para cada una de las variantes (Var) de datos. Para cada combinación de algoritmo y variante, se muestra la precisión del mejor y del peor modelo, el promedio de precisión obtenido junto con su desviación estándar (std), el promedio de precisión (prec) y *recall* de cada clase, el porcentaje de instancias clasificadas (Clas) y la calidad de clasificación ($Q_{\text{class}}$) y rechazo ($Q_{\text{rej}}$). Los algoritmos evaluados son AdaBoost (ADB), Bagging Classifier (BG), Bernoulli Naive Bayes (BNB), Decision Tree (DT), Extra Trees (ET), Gradient Boosting (GB), Logistic Regression (LR), Random Forest (RF), Cascada 3 (CC 3) y Cascada 4 (CC 4).

resultados en esta clase concreta, pero lo hacen clasificando muchas menos instancias, con ratios de clasificación de apenas 0.21 y 0.38 respectivamente.

Entre los modelos que clasifican un mayor número de instancias destacan el clasificador en cascada de tres niveles, el clasificador en cascada de cuatro niveles y el modelo Decision Tree. Al comparar todas las métricas entre ellos, se observa que Decision Tree clasifica ligeramente más instancias que la versión de cuatro niveles, pero en el resto de métricas ambas versiones del clasificador en cascada obtienen mejores resultados. La única excepción se encuentra en el *recall* de la clase `sin sobrepeso`, donde presentan una ligera caída de aproximadamente dos puntos porcentuales. Sin embargo, esta disminución se ve compensada por un excelente *recall* en la clase de `sobrepeso/obesidad` (0.90 y 0.91).

Estos resultados demuestran que las versiones del clasificador en cascada son especialmente adecuadas para identificar situaciones de riesgo (sobrepeso y obesidad), sin que ello suponga una penalización significativa en la detección de individuos `sin sobrepeso`. La incorporación de múltiples niveles, manteniendo los mismos umbrales de decisión, permite clasificar un mayor número de instancias sin comprometer la calidad de la clasificación ni la calidad de rechazo, aspecto especialmente relevante en el contexto médico en el que se enmarca esta propuesta.

A continuación se muestran varias figuras que relacionan diferentes métricas de rendimiento con el porcentaje de instancias clasificadas (figuras 4.4 - 4.7). El objetivo es analizar cómo varía el comportamiento de los distintos algoritmos respecto al número de instancias que el sistema decide clasificar. Se incluyen métricas como el GMean, el *recall* para ambas clases (`sin sobrepeso` y `sobrepeso/obesidad`) y la precisión global.

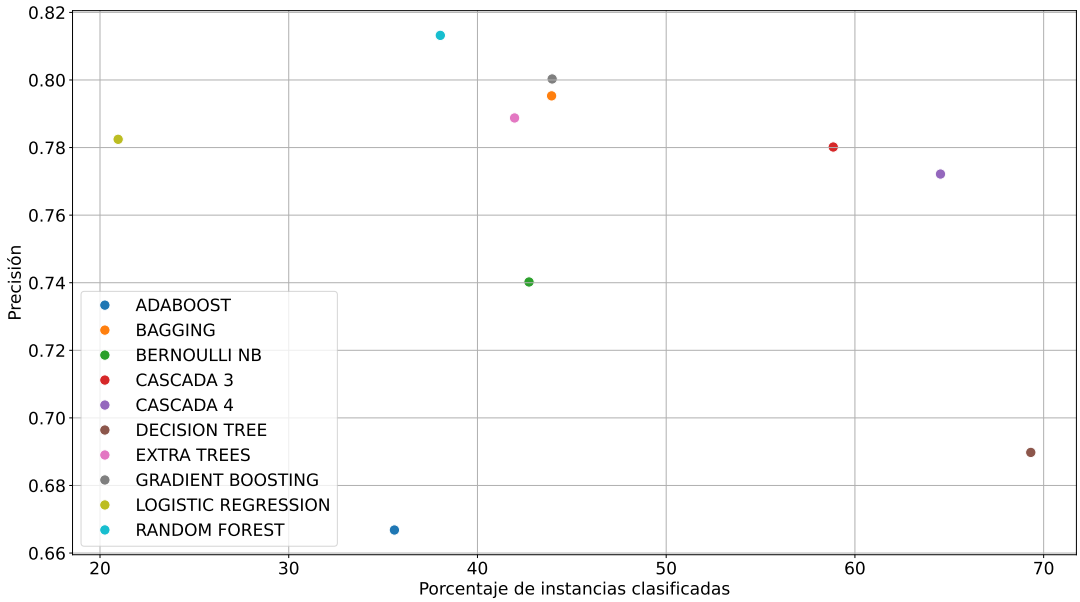



Figura 4.4: Precisión de los algoritmos vs porcentaje de instancias clasificadas.

Al observar los resultados de precisión (figura 4.4), se aprecia que los valores más altos corresponden a algoritmos como Random Forest, Gradient Boosting o Bagging Classifier.

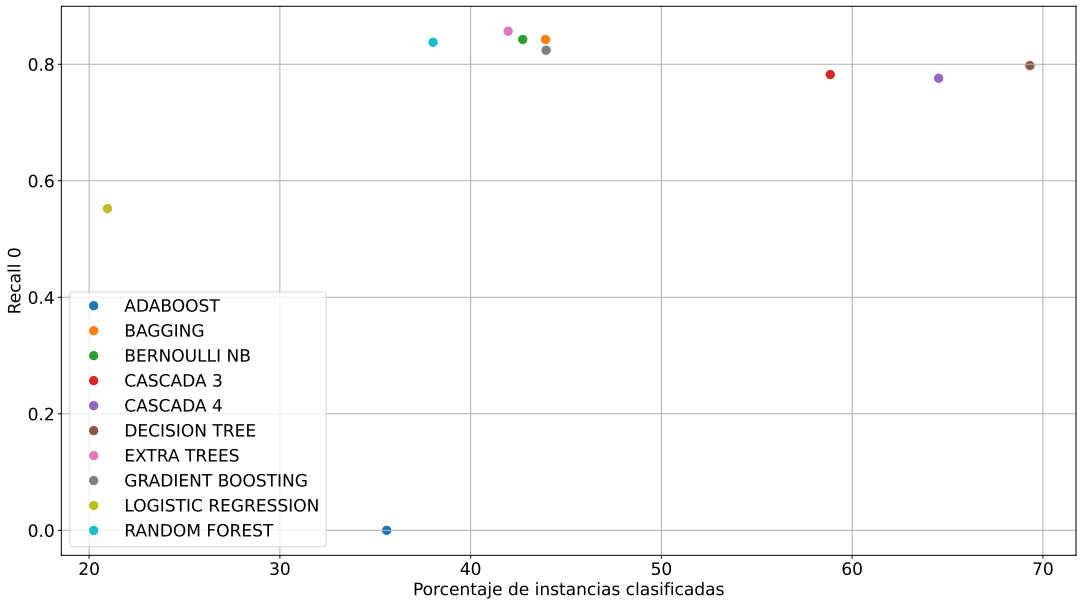



Figura 4.5: *Recall* de la clase `sin sobrepeso` vs porcentaje de instancias clasificadas.

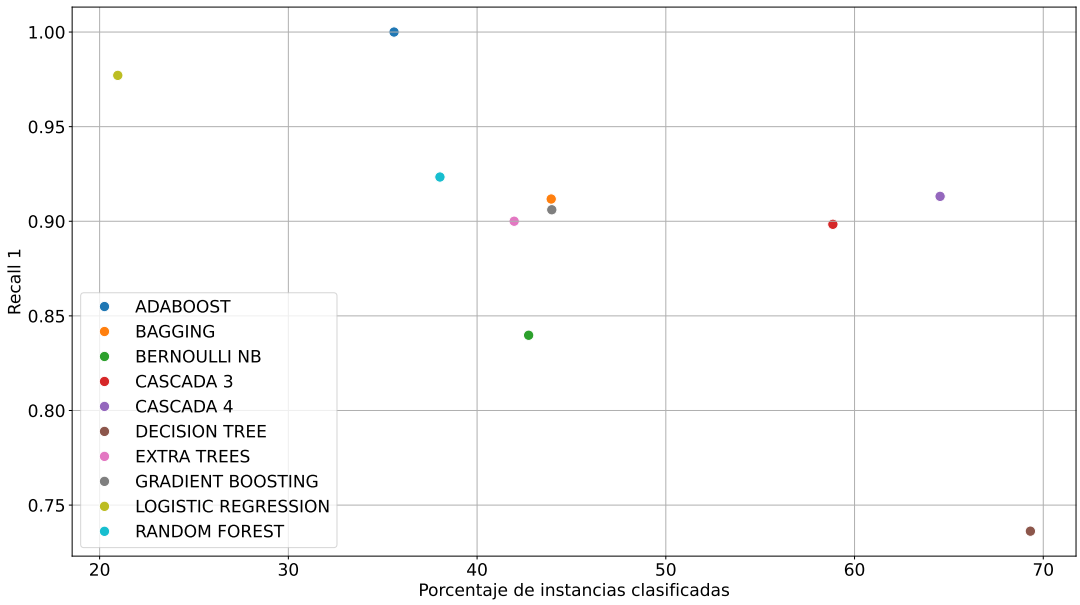



Figura 4.6: *Recall* de la clase `sobrepeso/obesidad` vs porcentaje de instancias clasificadas.

No obstante, estos resultados se alcanzan con porcentajes de clasificación relativamente bajos, que en la mayoría de los casos apenas superan el 40 % de las instancias. Algo similar ocurre en las métricas de *recall* para ambas clases (figuras 4.5 y 4.6), donde estos algoritmos obtienen un rendimiento igual e incluso superior al de los clasificadores en cascada. Sin embargo, este rendimiento elevado se produce sobre un subconjunto reducido de muestras.

Por otro lado, los modelos en cascada consiguen un buen equilibrio entre precisión y *recall*, clasificando un mayor número de instancias. Aunque Decision Tree es capaz de clasificar aún más muestras y ofrece buenos resultados en la clase `sin sobrepeso`, su rendimiento

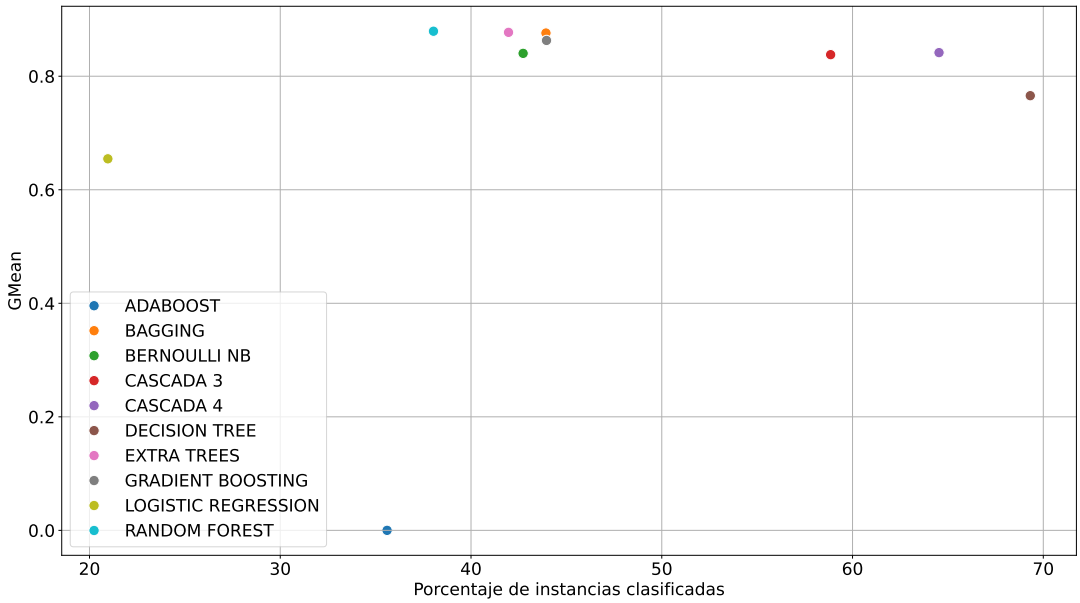



Figura 4.7: Gmean vs porcentaje de instancias clasificadas.

al identificar casos de `sobrepeso/obesidad` es inferior. Por eso, los clasificadores en cascada se presentan como la opción más adecuada, especialmente en un entorno médico donde clasificar como `sin sobrepeso` a una persona con `sobrepeso` puede tener un coste importante.

Como alternativa, si no se desea mantener la abstención después del último nivel del clasificador en cascada, es posible aplicar directamente uno de los algoritmos con mejor rendimiento sobre las muestras no clasificadas. En nuestro caso, usando Random Forest se consiguió alcanzar un *recall* de 0.77 para la clase con `sobrepeso/obesidad` y de 0.85 para la clase `sin sobrepeso`. Aunque estos resultados no son malos, se decidió mantener el clasificador en cascada con abstención, con el objetivo de garantizar una mayor confianza y seguridad en las predicciones, dejando solo un número reducido de casos para ser evaluados con mayor rigurosidad por parte del equipo médico. Al reducir la carga de trabajo de los profesionales sanitarios en más de un 60 %, manteniendo una precisión cercana al 88 % en algunas variantes, se consigue un beneficio considerable.

### 4.4.1 Análisis estadístico

Con el objetivo de comprobar si existían diferencias significativas entre los algoritmos evaluados, se llevó a cabo un análisis estadístico considerando tanto la calidad de clasificación como la calidad del rechazo para cada una de las variantes de datos.

En el caso de $Q_{\text{class}}$, los resultados mostraron diferencias estadísticamente significativas entre los algoritmos en todas las variantes de datos. Se obtuvieron los siguientes estadísticos de Friedman: $\chi_F^2 = 748.90$ para la variante `26`, $\chi_F^2 = 674.57$ para la `36`, $\chi_F^2 = 678.04$ para la `37a`, $\chi_F^2 = 697.77$ para la `37c` y $\chi_F^2 = 692.90$ para la variante `38`, todos ellos con valores de $p < 0.001$.

Para identificar qué pares de algoritmos presentaban diferencias significativas, se aplicó el test post-hoc de Nemenyi. Las figuras 4.8-4.12 muestran los diagramas de comparación por rangos en cada variante. En todas las variantes, las dos versiones del clasificador en cascada se sitúan entre los modelos con mejor rendimiento promedio, superando al resto de algoritmos evaluados. Modelos como Random Forest, Gradient Boosting o Bagging muestran un rendimiento intermedio, pero sin alcanzar a los clasificadores en cascada.

En particular, se observan diferencias significativas de las dos versiones del clasificador en cascada frente al resto de algoritmos en todas las variantes excepto en la 26, donde las líneas de comparación del test de Nemenyi indican que no existen diferencias estadísticamente significativas entre los clasificadores en cascada y el modelo Decision Tree.

En todas las variantes, las dos versiones del clasificador en cascada se mantienen muy próximas en el ranking y no presentan diferencias significativas entre sí, lo que indica que ambas alternativas ofrecen un comportamiento robusto y consistente en términos de $Q_{\text{class}}$.

Este mismo análisis se repitió para la métrica $Q_{\text{rej}}$, evaluando nuevamente los algoritmos en cada una de las variantes de datos. También en este caso, los resultados del test de Friedman revelaron diferencias estadísticamente significativas entre los algoritmos en todas las variantes. Los resultados fueron: $\chi_F^2 = 573{,}51$ para la variante `26`, $\chi_F^2 = 634{,}09$ para la `36`, $\chi_F^2 = 534{,}44$ para la `37a`, $\chi_F^2 = 571{,}28$ para la `37c` y $\chi_F^2 = 585{,}61$ para la variante `38`, todos con valores de $p < 0{,}001$.

Las figuras 4.13 - 4.17 muestran los diagramas de comparación por rangos derivados del test de Nemenyi para $Q_{\text{rej}}$. En ellos se observa nuevamente que los modelos en cascada obtienen los mejores rangos promedio, situándose en las primeras posiciones en todos los conjuntos y demostrando diferencias significativas frente al resto de algoritmos.

En todos los conjuntos, no se observan diferencias significativas entre las dos versiones del clasificador en cascada, lo que refuerza su buen funcionamiento en la tarea de rechazo.

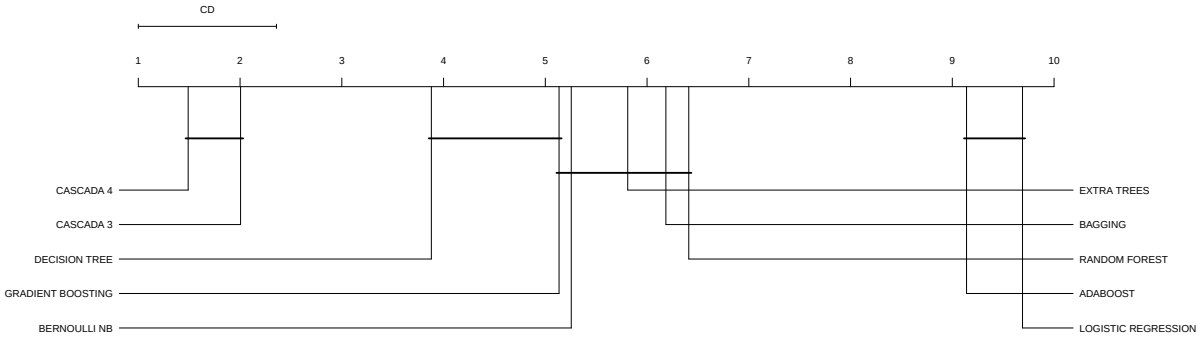



Figura 4.8: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{class}}$ para la variante `38` del conjunto de datos.

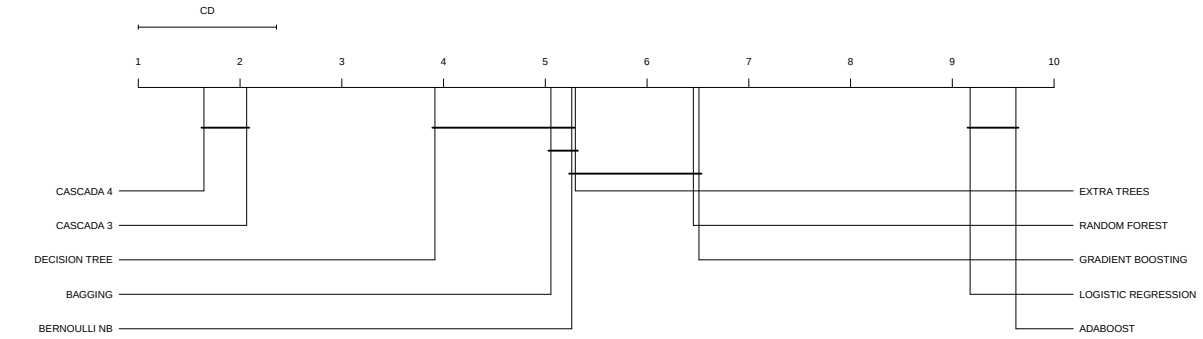



Figura 4.9: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{class}}$ para la variante `37a` del conjunto de datos.

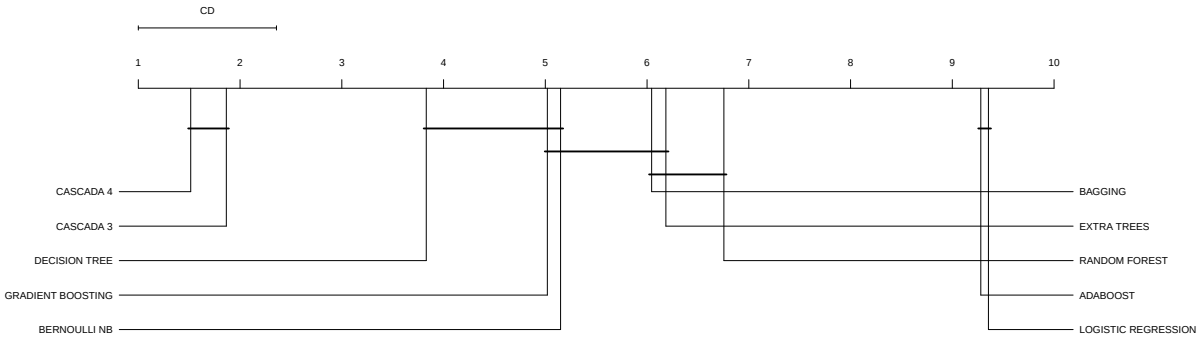



Figura 4.10: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{class}}$ para la variante `37c` del conjunto de datos.

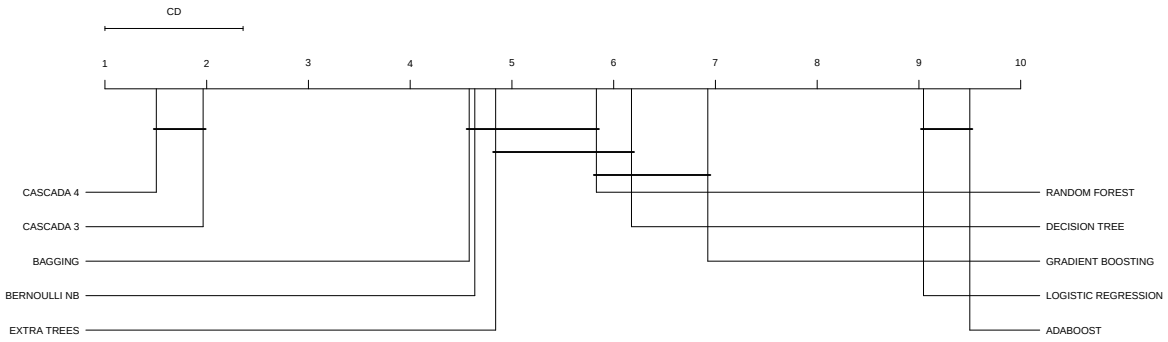



Figura 4.11: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{class}}$ para la variante `36` del conjunto de datos.

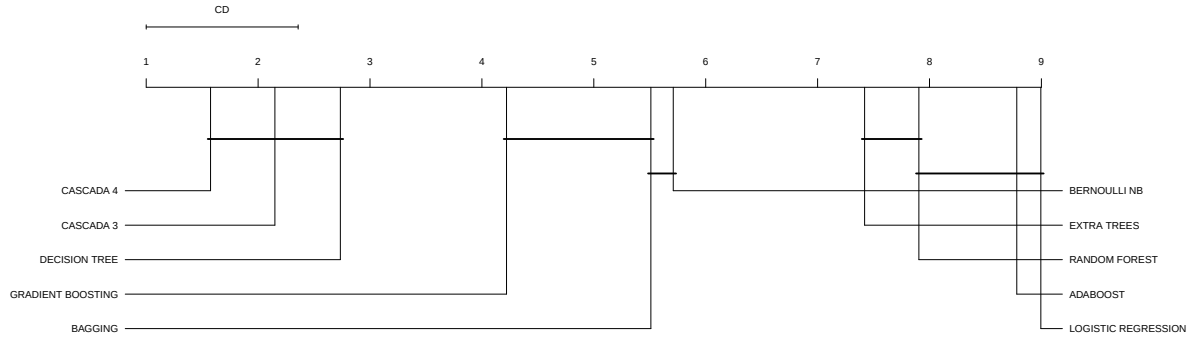



Figura 4.12: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{class}}$ para la variante `26` del conjunto de datos.

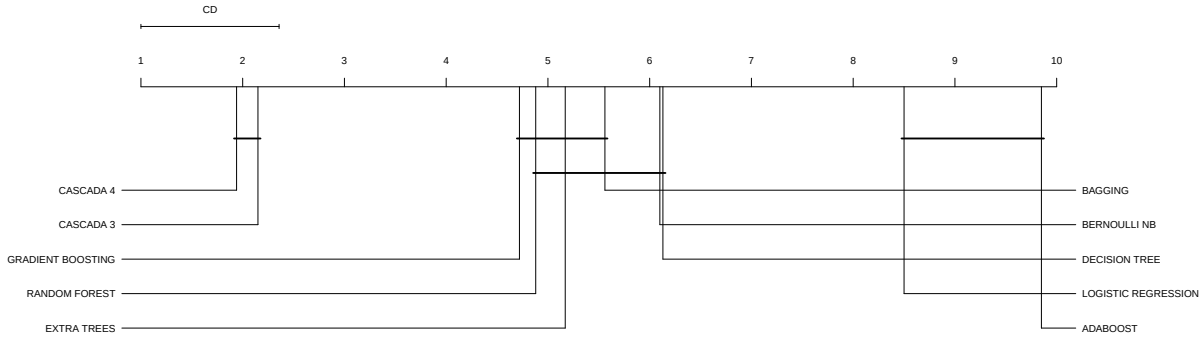



Figura 4.13: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{rej}}$ para la variante `38` del conjunto de datos.

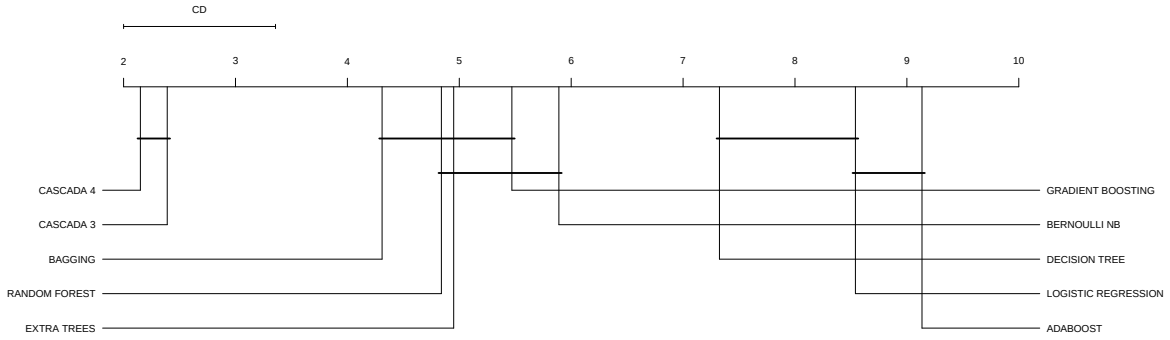



Figura 4.14: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{rej}}$ para la variante `37a` del conjunto de datos.

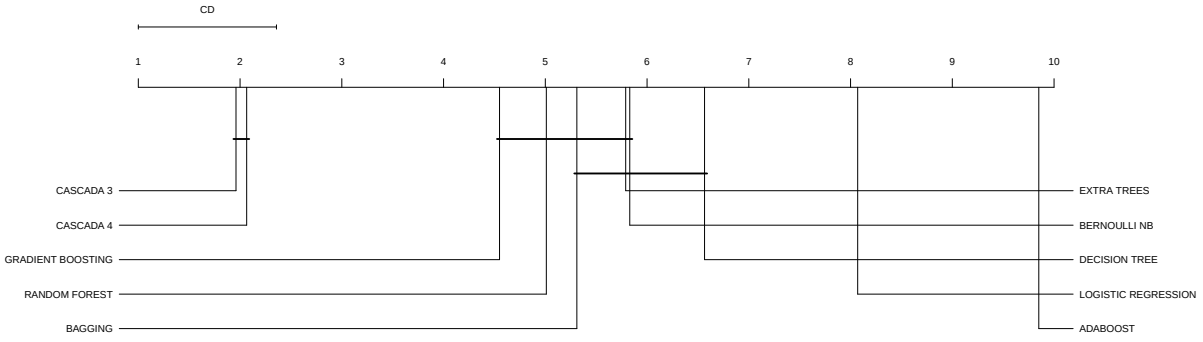



Figura 4.15: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{rej}}$ para la variante `37c` del conjunto de datos.

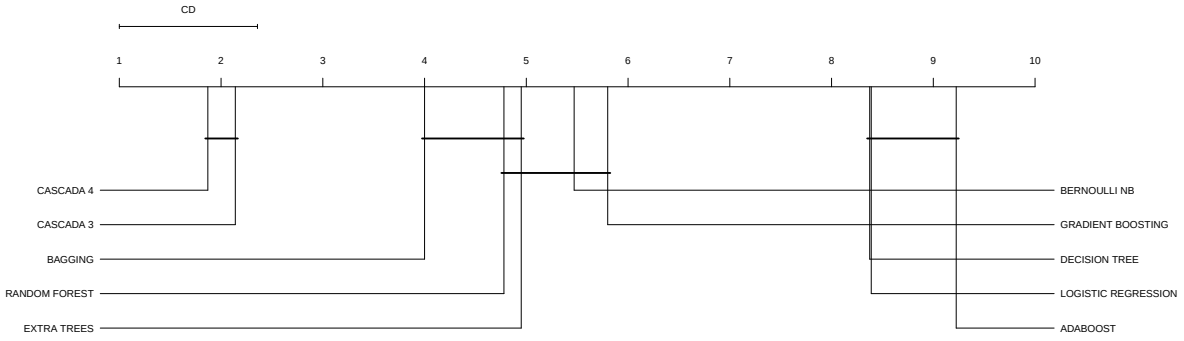



Figura 4.16: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{rej}}$ para la variante `36` del conjunto de datos.

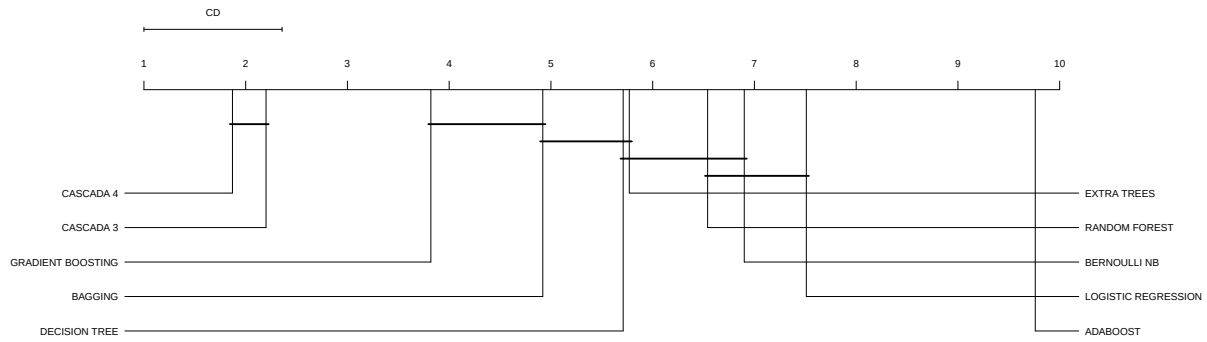



Figura 4.17: Ranking y diferencias críticas entres los algoritmos de ML para la métrica $Q_{\text{rej}}$ para la variante 26 del conjunto de datos.

Al igual que en el capítulo 3 también se aplicó un método bayesiano con el objetivo de estimar la probabilidad de que cada algoritmo sea el mejor para cada una de estas métricas. Este enfoque permite una interpretación más directa de los resultados, ya que cuantifica la probabilidad de superioridad de cada método frente al resto.

En el caso de $Q_{\text{class}}$, los clasificadores en cascada destacan (figuras 4.18-4.22), ya que ambos modelos presentan una probabilidad de ganar superior al resto. En algunas variantes, modelos como Decision Tree o Gradient Boosting aparecen en el tercer lugar con una probabilidad de ganar cercana a la de los clasificadores en cascada, pero sin solapamiento entre los intervalos de incertidumbre.

En cuanto a $Q_{\text{rej}}$, las dos versiones del clasificador en cascada vuelven a obtener los mejores resultados (figuras 4.23-4.27). Ninguna otra técnica muestra solapamiento en los intervalos de incertidumbre con estos dos clasificadores, demostrando su superioridad. Estos resultados se alinean con los obtenidos mediante el test de Friedman y el análisis post-hoc de Nemenyi, lo que demuestra que el clasificador en cascada ofrece la mejor calidad tanto en clasificación como en rechazo.

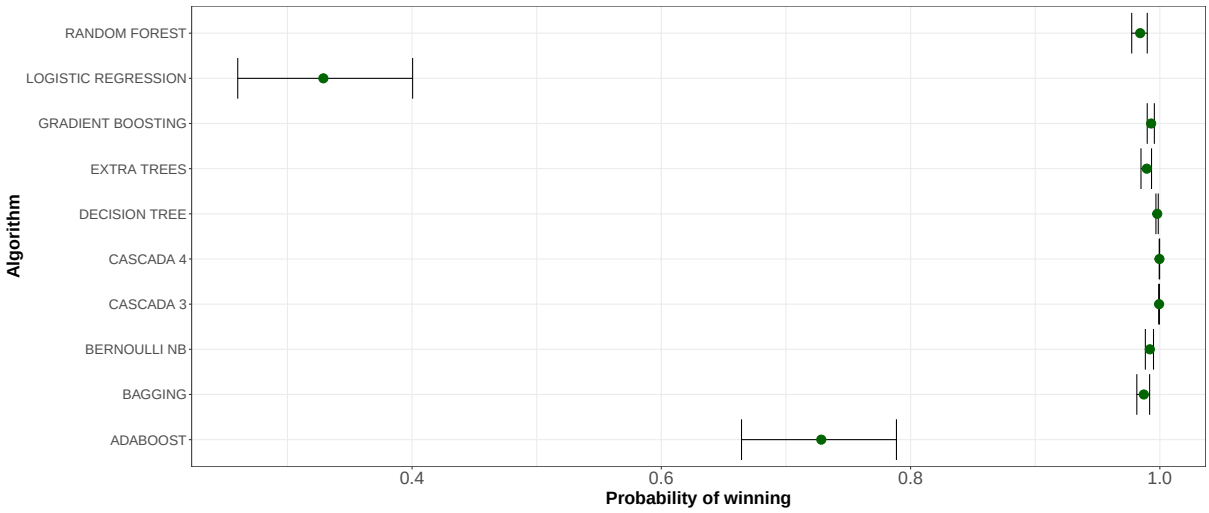



Figura 4.18: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{class}}$ en la variante `38` del conjunto de datos.

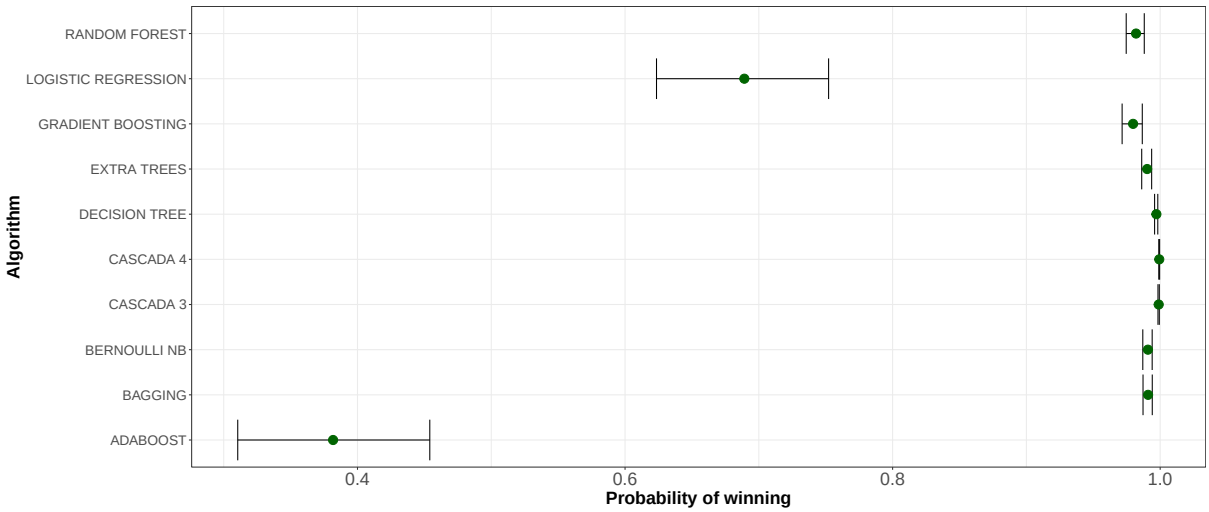



Figura 4.19: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{class}}$ en la variante `37a` del conjunto de datos.

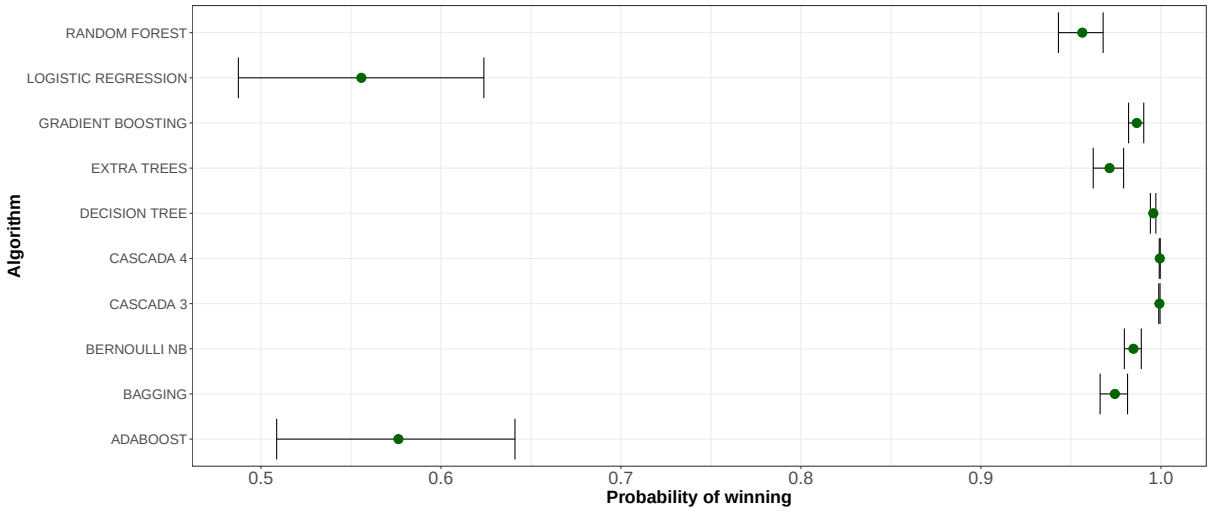



Figura 4.20: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{class}}$ en la variante `37c` del conjunto de datos.

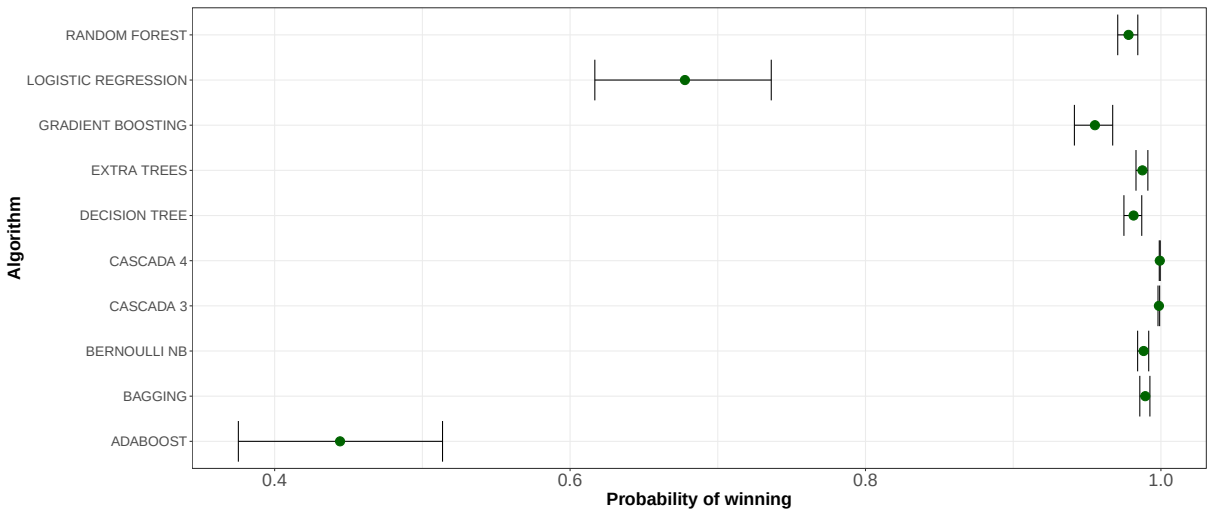



Figura 4.21: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{class}}$ en la variante `36` del conjunto de datos.

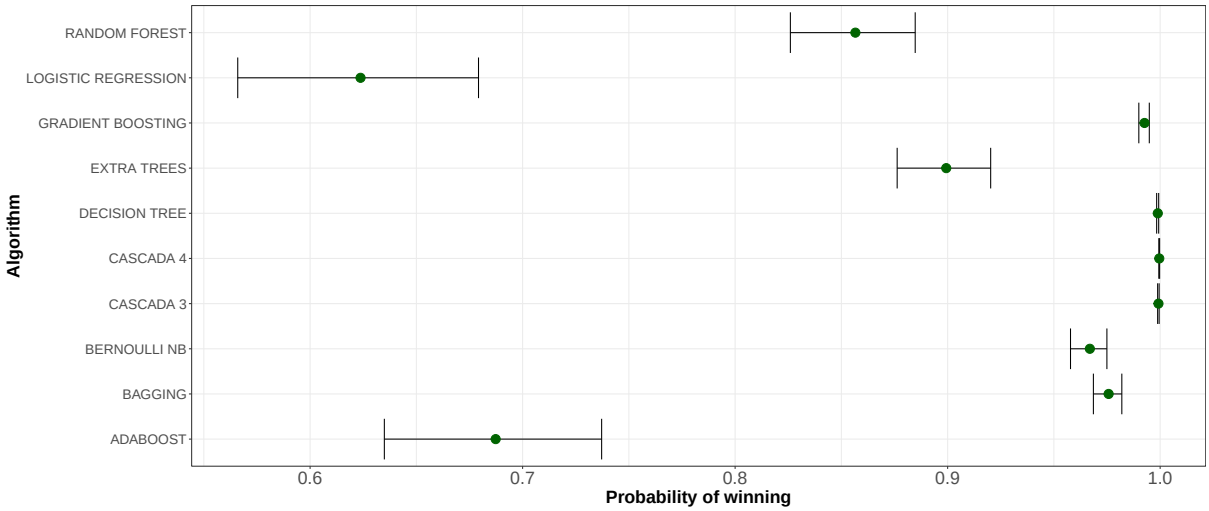



Figura 4.22: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{class}}$ en la variante `26` del conjunto de datos.

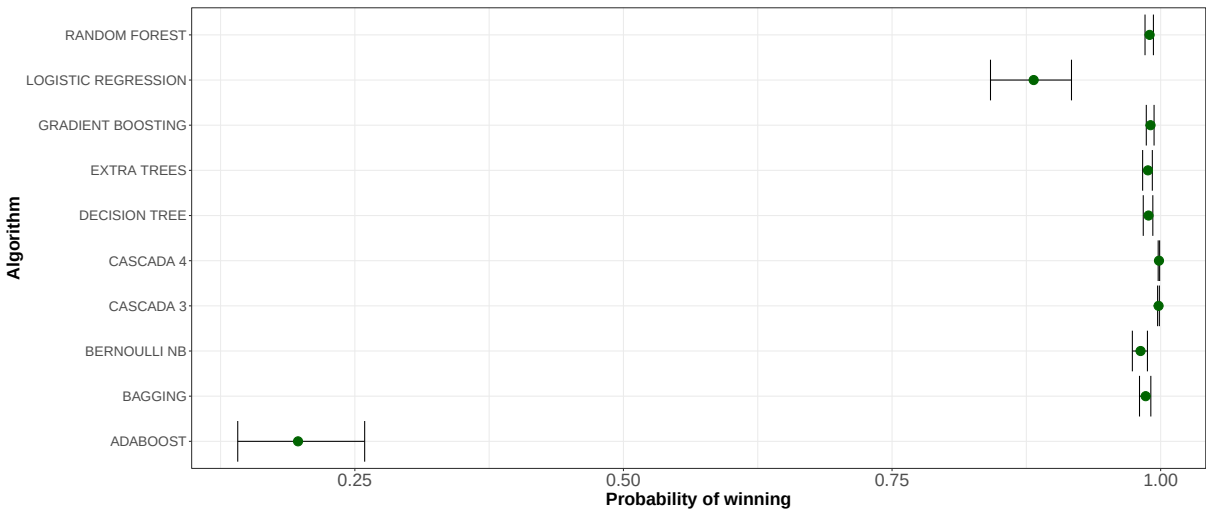



Figura 4.23: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{rej}}$ en la variante `38` del conjunto de datos.

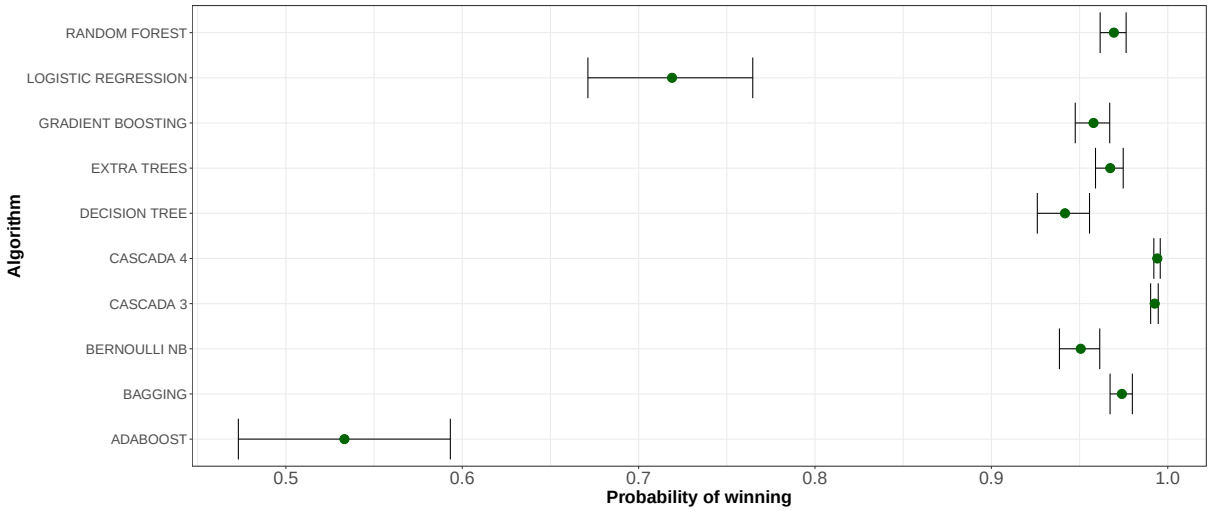



Figura 4.24: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{rej}}$ en la variante `37a` del conjunto de datos.

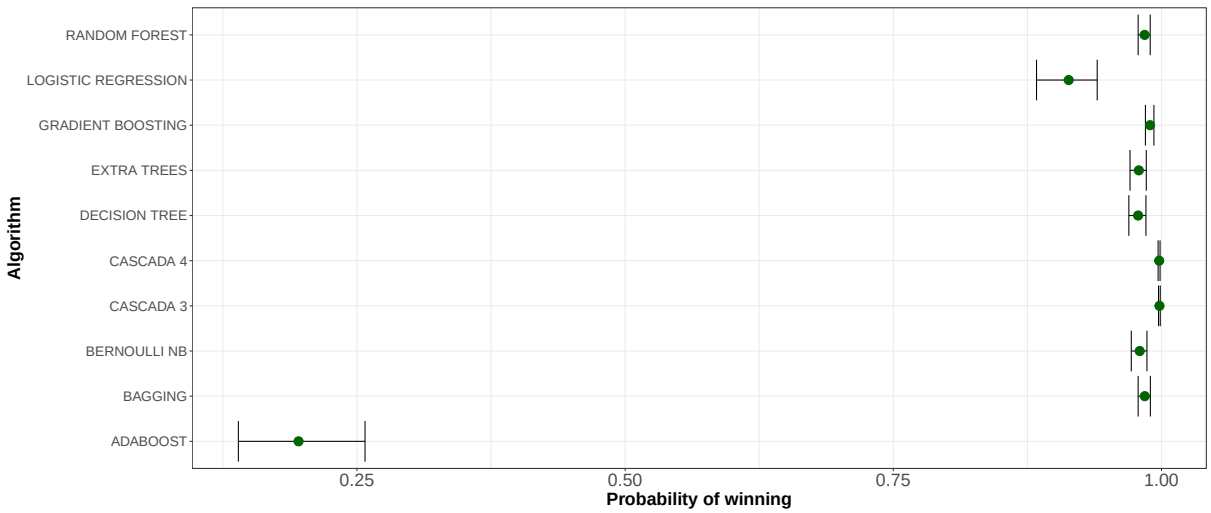



Figura 4.25: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{rej}}$ en la variante `37c` del conjunto de datos.

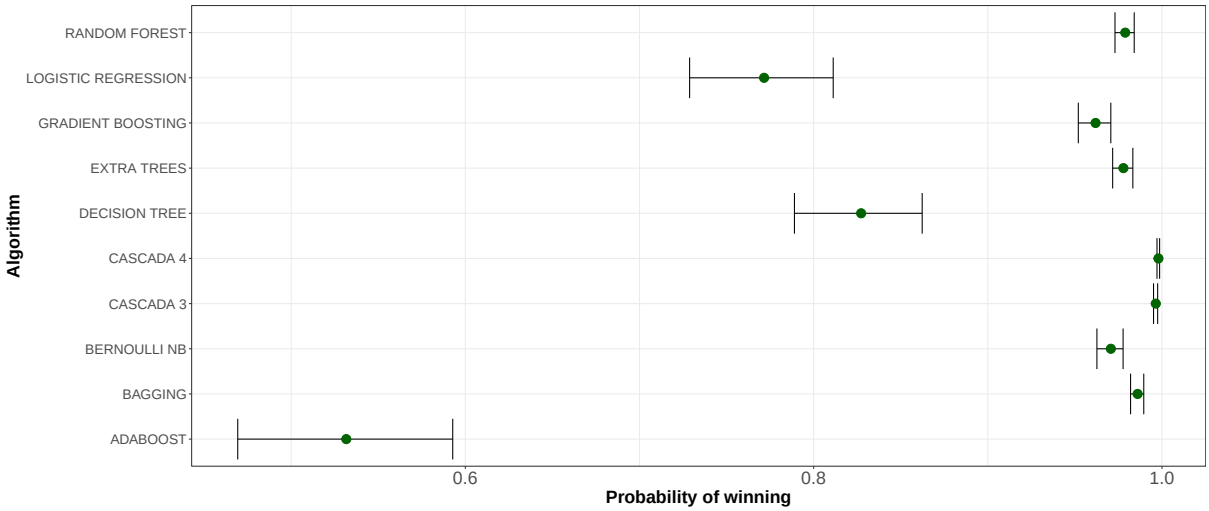



Figura 4.26: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{rej}}$ en la variante `36` del conjunto de datos.

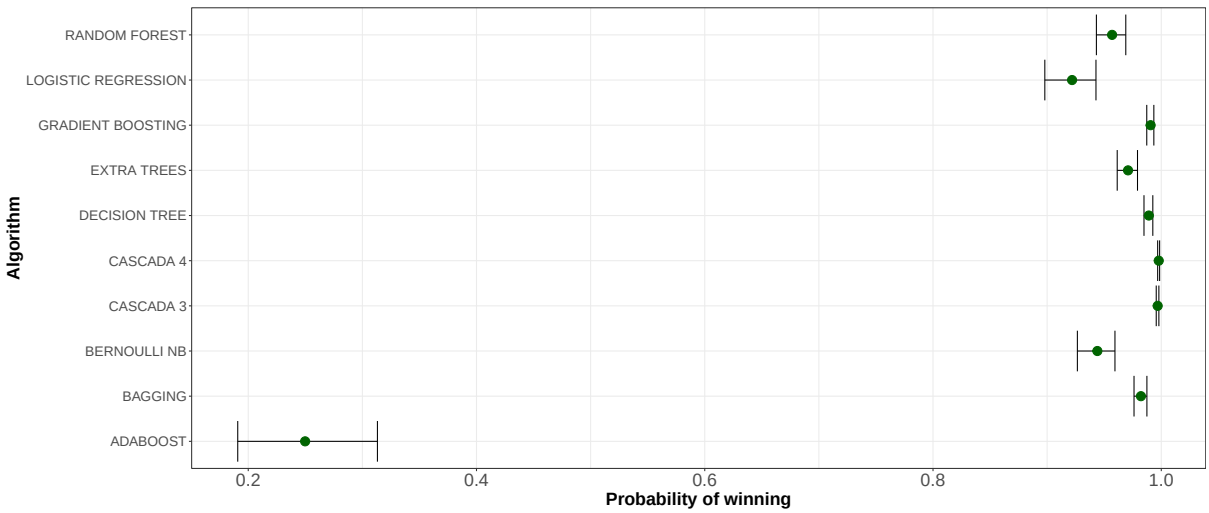



Figura 4.27: Probabilidad de ganar para cada uno de los algoritmos de ML para la métrica $Q_{\text{rej}}$ en la variante `26` del conjunto de datos.

Todos los experimentos fueron realizados en un procesador Intel(R) Core(TM) i7-10870H a 2.20 - 5.0 GHz con 32 GB de memoria RAM en Windows 10. Los experimentos se ejecutaron con ocho hilos en paralelo para aprovechar todos los núcleos del procesador sin afectar al rendimiento y sin ninguna otra tarea en ejecución al mismo tiempo.

## 4.5 Conclusiones

Se ha desarrollado un clasificador en cascada con abstención que utiliza diferentes algoritmos de ML y umbrales para determinar la pertenencia a una clase, con el objetivo de mejorar la detección de casos de `sobrepeso/obesidad` a partir de información relacionada con hábitos, ejercicio físico y patologías. El modelo ha sido evaluado sobre un conjunto de datos recopilado en el marco del proyecto `GenObIA`, y se han empleado múltiples variantes del mismo para comprobar su robustez frente a diferentes combinaciones de variables.

Los resultados han demostrado que el modelo es especialmente eficaz a la hora de reducir los falsos negativos, un aspecto clave en contextos médicos. Además, consigue clasificar un número elevado de muestras en comparación con el resto de algoritmos, manteniendo una buena precisión y un *recall* muy alto en la clase de `sobrepeso/obesidad`.

En términos de calidad de clasificación y de rechazo, el clasificador en cascada también obtiene los mejores resultados. Esto confirma no solo que el clasificador en cascada clasifica más, sino que lo hace mejor que el resto y que las decisiones de abstención se aplican con sentido, rechazando los casos dudosos para que puedan ser analizados posteriormente por el equipo médico. En este contexto, es importante priorizar la detección de individuos con `sobrepeso/obesidad`, ya que no detectarlos puede tener consecuencias clínicas importantes. En cambio, clasificar erróneamente como positivo a una persona `sin sobrepeso` implica un riesgo menor.

Además del análisis clásico, se ha llevado a cabo una evaluación estadística mediante el test de Friedman y un enfoque bayesiano. Ambos métodos han confirmado diferencias significativas entre los algoritmos para cada una de las variantes, situando al clasificador en cascada como la opción más robusta tanto en calidad de clasificación como en la gestión del rechazo.

Estos resultados avalan la validez del clasificador en cascada como solución eficaz para la predicción del sobrepeso y la obesidad. Asimismo, su diseño modular y adaptable lo convierte en una base sólida para desarrollar estrategias más avanzadas, como el clasificador Multinivel presentado en el capítulo 6.

En cuanto al número de niveles, los experimentos mostraron que tanto la configuración de tres como la de cuatro niveles del clasificador en cascada ofrecen mejoras sustanciales frente al resto de algoritmos utilizados. La versión de cuatro niveles logra clasificar un mayor porcentaje de instancias con una precisión media similar a la de tres niveles pero

con mejores valores de $Q_{\text{class}}$ y $Q_{\text{rej}}$. Sin embargo, al añadir un quinto nivel, se observó un incremento notable de errores en ambas clases, debido a la mayor incertidumbre de las instancias no clasificadas, empeorando la precisión del clasificador y disminuyendo el *recall* de ambas clases. Por ello, se fijó en cuatro el número máximo de niveles, como punto de equilibrio entre porcentaje de instancias clasificadas y fiabilidad de clasificación.

# Capítulo 5

# Optimización de modelos *ensemble* de clasificación binaria mediante el estudio de las curvas de rendimiento

En el ámbito del aprendizaje automático, los modelos *ensemble* se han consolidado como una estrategia eficaz para mejorar el rendimiento predictivo mediante la combinación de múltiples modelos base. Sin embargo, la forma en la que se toman las decisiones dentro de estos modelos *ensemble*, habitualmente mediante esquemas de votación con umbrales fijos, puede limitar su potencial, especialmente en escenarios con un alto desbalanceo de clases.

Una forma común de medir el rendimiento de estos modelos es mediante curvas ROC o PRC ya que ofrecen una visión detallada del comportamiento de los clasificadores bajo diferentes umbrales de decisión.

En este capítulo abordamos el problema de optimización de modelos *ensemble* explorando el espacio de decisión, definido por los clasificadores base y el esquema de decisión del modelo *ensemble*. Esta propuesta permite identificar configuraciones óptimas de umbrales sin necesidad de reentrenar los modelos.

## 5.1 Propuesta: Performance Curve Mapping (PCM)

Para enmarcar adecuadamente la propuesta presentada en este capítulo, es necesario introducir una definición formal del modelo *ensemble*. En términos generales, un modelo *ensemble* puede representarse como $E = \{M, D\}$, donde $M$ es un conjunto de $n$ modelos individuales, $M = \{m_1, m_2, \ldots, m_n\}$, y $D$ es el esquema de decisión que combina las salidas de los modelos base para generar la predicción final.

Cada modelo $m_i$ está caracterizado por un conjunto propio de parámetros $\{p_i^k\}^{k=1\ldots k_i}$, mientras que el esquema de decisión $D$ también dispone de su propio conjunto de pará-

metros $\{p_D^k\}^{k=1...k_D}$. Por tanto, una instancia del modelo $E$ se define como la colección de parámetros $p = \{\{p_i^k\}, \{p_D^k\}\} \in \mathcal{X}$, donde $\mathcal{X}$ es el espacio de decisión con todas las posibles configuraciones del modelo. Denotamos el conjunto de todas las instancias posibles del modelo *ensemble* como $\mathcal{C}_E(\mathcal{X})$.

Para sistematizar este análisis, se ha introducido el concepto de *curva de Rendimiento*, que generaliza las curvas clásicas utilizadas en evaluación de modelos de clasificación. Definimos la *curva de Rendimiento* como cualquier representación que exprese el compromiso entre dos métricas en conflicto dentro de un entorno de clasificación.

En consecuencia, también se extiende el concepto de espacio ROC (detallado en el capítulo 2.5) a *espacio de Rendimiento*, que se denota como $\mathcal{U}$, definido como el producto cartesiano de las métricas evaluadas. Por ejemplo, si las métricas son TPR y FPR, entonces $\mathcal{U} \equiv \mathcal{U}_{ROC}$, y cada punto $y \in \mathcal{U}$ tendrá la forma $(tpr_i, fpr_i)$. De igual modo, si se utilizan TPR y la precision del modelo, se tiene $\mathcal{U} \equiv \mathcal{U}_{PRC}$, con $y = (\text{tpr}_i, \text{prec}_i)$.

Con el objetivo de evaluar el rendimiento de cada configuración del modelo *ensemble*, se define la función $f_E^{\mathcal{S}}$, que permite mapear cualquier instancia en el espacio de decisión del modelo hacia su correspondiente punto en el *espacio de Rendimiento*. Esta función toma una instancia $E_i \in \mathcal{C}_E(\mathcal{X})$, la aplica sobre un conjunto de datos $\mathcal{S}$, y devuelve un punto $y \in \mathcal{U}$

$$f_E^{\mathcal{S}} : \mathcal{C}_E(\mathcal{X}) \overset{\mathcal{S}}{\mapsto} \mathcal{U}$$

A partir de esta formalización, se desarrolla la propuesta `Performance Curve Mapping` (`PCM`), que consiste en definir la *curva de Rendimiento* como el frente de Pareto resultante de un proceso de optimización multiobjetivo sobre el espacio de decisión. El conjunto de puntos que forman el *espacio de Rendimiento* se define como:

$$Y = \{y \in \mathcal{U} \ : \ y = f_E^{\mathcal{S}}(p), \ p \in \mathcal{X}\} \tag{5.1}$$

Y a partir de este conjunto definimos la *curva de Rendimiento*, (PC, del inglés *Performance Curve*) como:

$$\text{PC} = \{y' \in Y \ : \ \{y'' \in Y \ : \ y'' \succ y', \ y' \neq y''\} = \emptyset\} \tag{5.2}$$

donde la relación $\succ$ indica dominancia; un punto $y''$ domina a otro $y'$ si es mejor en al menos una métrica y no peor en las demás.

A partir de ahora, y con el objetivo de detallar el funcionamiento de `PCM`, se ha utilizado la curva ROC como caso de estudio, ya que es comúnmente utilizada en problemas de clasificación binaria. Este caso de uso permite mostrar paso a paso la aplicación de `PCM` en la construcción de la *curva de Rendimiento*.

Para comprender adecuadamente la generación de las curvas ROC, es fundamental entender primero el funcionamiento de un modelo de clasificación binaria. En estos modelos,

uno de los parámetros $\{p_i^k\}$ es el umbral, $T_i$, que se utiliza para determinar la pertenencia de una instancia a la clase positiva o negativa. Este mismo principio se extiende a los modelos *ensemble*, donde el proceso de decisión final suele implementarse mediante un esquema de votación, en el que se fija un umbral global $T_D$; si la proporción de predicciones positivas entre los modelos individuales supera dicho umbral, entonces el conjunto clasifica la instancia como positiva.

Habitualmente, se utiliza el valor $T_i = 0.5$ como umbral por defecto para cada modelo base $m_i$, empleando únicamente el umbral $T_D$ como variable de ajuste para construir la curva ROC del modelo *ensemble*, variando sus valores dentro del intervalo unitario $I = [0, 1]$. Esta práctica implica que la generación de la curva ROC del modelo $E$ quede condicionada por los valores predeterminados de los parámetros $\{p_i^k\}$ definidos durante la configuración inicial del clasificador, en este caso limitando la optimización del modelo a $T_D$.

Con `PCM` se analizan de forma conjunta los umbrales de clasificación de los modelos base que componen el modelo *ensemble* y el umbral del esquema de decisión. Se parte de la premisa de que los modelos base están preconfigurados, es decir, que sus hiperparámetros han sido previamente ajustados y su entrenamiento ya ha concluido.

En este contexto, se define el espacio de decisión como $\mathcal{X} = I^{n+1}$, donde cada vector $(T_1, \ldots, T_n, T_D)$ representa una posible configuración de umbrales: uno para cada modelo base $m_i$, y uno adicional, $T_D$, asociado al esquema de votación. Este espacio es separable, ya que la modificación del umbral de un modelo $m_i$ no interfiere en el comportamiento de los demás. Esto permite descomponer el problema en $n$ subproblemas independientes, más uno adicional para el esquema de votación. Formalmente, el espacio de configuraciones posibles se expresa como $\mathcal{C}_E(\mathcal{X}) = \left(\bigtimes_{i=1}^{n} \mathcal{C}_{m_i}(\mathcal{I})\right) \times \mathcal{C}_D(\mathcal{I})$.

Dado que el entrenamiento de los modelos no se ve afectado por los valores de los umbrales, es posible explorar $\mathcal{C}_E(\mathcal{X})$ sin necesidad de reentrenar ni recalcular las predicciones. Esta ventaja nos permite aplicar directamente un algoritmo de optimización multiobjetivo sobre el espacio de decisión, con la finalidad de encontrar configuraciones de umbrales que maximicen simultáneamente el TPR y minimicen el FPR. El resultado de este proceso es un mapeo del espacio de decisión al espacio objetivo, donde el conjunto de soluciones no dominadas constituyen la curva ROC del modelo *ensemble*.

La figura 5.1 ofrece una representación visual del método propuesto, facilitando la comprensión del proceso de exploración y optimización del espacio de decisión. En ella, un punto $C$ del espacio de decisión, $\mathcal{X}$, representa una instancia concreta del modelo $E$, definida por un vector de umbrales $(T_1, ..., T_n, T_D)$. Cada valor $T_i$ del vector está asociado al umbral de decisión de un modelo base $m_i$ dentro del modelo *ensemble*, mientras que $T_D$ corresponde al umbral aplicado por el esquema de votación. Al aplicar cada umbral $T_i$ sobre su respectivo modelo base, se obtienen los valores de TPR y FPR que permiten construir la curva ROC individual de cada $m_i$.

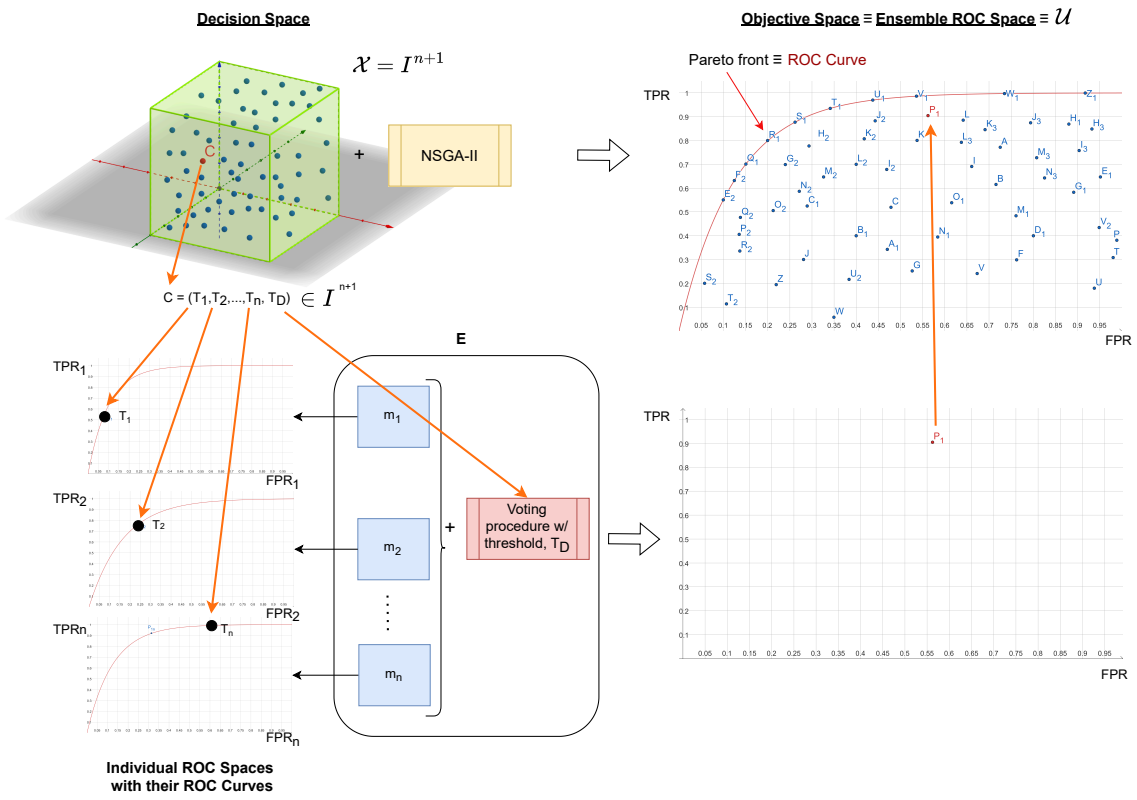



Figura 5.1: Proceso de `Performance Curve Mapping`. Los puntos en el espacio de decisión ($\mathcal{X}$) representan todas las posibles combinaciones de umbrales para los modelos en el modelo *ensemble*, incluyendo el umbral para el esquema de votación. En otras palabras, abarcan todas las posibles instancias del modelo *ensemble* ($E$) con diferentes umbrales. Un algoritmo multiobjetivo, como NSGA-II, explora este espacio, produciendo una nube de puntos en el espacio objetivo. La curva ROC del clasificador es el frente de Pareto.

Mediante un algoritmo de optimización multiobjetivo, concretamente NSGA-II (del inglés, *Non-dominated Sorting Genetic Algorithm*) [33] se explora el espacio de decisión, $\mathcal{X}$. Este algoritmo evolutivo está diseñado para identificar un conjunto diverso y representativo de soluciones óptimas, clasificándolas por niveles de no dominancia y manteniendo la diversidad a través de técnicas como *crowding distance* (que evita la concentración excesiva de soluciones similares) y elitismo (que asegura la preservación de las mejores soluciones encontradas). Como resultado de este proceso, todas las soluciones generadas por NSGA-II conforman el espacio ROC del modelo *ensemble*, es decir, el conjunto de configuraciones de umbrales que reflejan el comportamiento de $E$ en términos de TPR y FPR. A partir de las soluciones no dominadas dentro de este espacio se construye la curva ROC de $E$.

## 5.2 Conjunto de datos

Para validar el método propuesto, se han utilizado dos problemas de clasificación diferentes, uno de los cuales presenta un desbalanceo de clases considerable. Ambos conjuntos de

**Algorithm 5** Algoritmo NSGA-II

**Input:** Problem Definition, Population Size $N$, Number of Generations $G$
**Output:** Pareto-optimal Front ▷ Frente final no dominado
**Initialize** Population $P_0$ with $N$ random individuals ▷ Población aleatoria
**Evaluate** Objective Functions for $P_0$ ▷ Evaluación individuos
**Perform Fast Non-Dominated Sorting** on $P_0$ ▷ Clasificación según dominancia
**for each** Front $F$ in $P_0$ **do**
    **Compute Crowding Distance** for individuals in $F$
**end for**
**Generate** Offspring Population $Q_0$ from $P_0$ ▷ Creación descendencia inicial
**for** $t = 1$ to $G$ **do** ▷ Bucle principal del algoritmo
    **Combine** Parent Population $P_t$ and Offspring Population $Q_t$ to create $R_t$ ▷ Unión de padres e hijos
    **Perform Fast Non-Dominated Sorting** on $R_t$ ▷ Reordenar población combinada
    **Initialize** New Population $P_{t+1} \leftarrow \emptyset$ ▷ Nueva población vacía
    $i \leftarrow 1$
    **while** $|P_{t+1}| + |F_i| \leq N$ **do**
        **Compute Crowding Distance** for individuals in $F_i$
        **Add** Front $F_i$ to $P_{t+1}$ ▷ Agregar frente completo
        $i \leftarrow i + 1$
    **end while**
    **Sort** $F_i$ by Crowding Distance (Descending) ▷ Ordenar para completar población
    **Select** Top Individuals to Fill $P_{t+1}$ ▷ Seleccionar mejores individuos
    **Generate** Offspring Population $Q_{t+1}$ from $P_{t+1}$ using: ▷ Nueva descendencia con operadores genéticos
        - **Tournament Selection** ▷ Selección por torneo
        - **SBX (Simulated Binary Crossover)** ▷ Cruce binario simulado
        - **Polynomial Mutation** ▷ Mutación polinómica
    **Evaluate** Objective Functions for $Q_{t+1}$ ▷ Evaluación nueva descendencia
**end for**
**Return** Non-Dominated Front $F_1$

datos ya fueron empleados en los capítulos 3.2 y 4.2; no obstante, con el objetivo de facilitar la lectura, a continuación se resumen nuevamente sus principales características.

El primer conjunto de datos, `Seguros`, contiene información relacionada con pólizas de automóvil. Cada año de una póliza de seguro está representado como una fila dentro del conjunto de datos. El objetivo es identificar pólizas que generen al menos un siniestro en los próximos doce meses después de su contratación. Inicialmente, el conjunto de datos original tiene un total de 81 194 filas y 102 variables. Después de la fase de preprocesamiento, limpieza y filtrado de variables, el conjunto de datos se reduce a 69 080 filas y 21 variables. Los datos están altamente desbalanceados: 681 de las 69 080 filas son reclamaciones, lo que significa que la clase mayoritaria representa el 98.9 % de los casos.

El segundo conjunto de datos, `GenObIA`, contiene información clínica de 1179 participantes mayores de 18 años, reclutados de 14 centros, incluidos hospitales y universidades de

Madrid (España). El objetivo es detectar individuos en riesgo de desarrollar sobrepeso u obesidad, considerando como sobrepeso aquellos con un IMC $\geq$ 25. El dataset utilizado consta de 37 variables, reducidas de las 97 originales. De los 1179 participantes, 612 (52.0 %) tenían sobrepeso/obesidad y 567 (48.0 %) no, mostrando una distribución relativamente equilibrada entre las clases.

## 5.3 Marco experimental y metodología

Con el objetivo de evaluar el método propuesto, se desarrolló un modelo *ensemble* compuesto por un total de 50 modelos XGBoost, todos ellos configurados con los mismos hiperparámetros, detallados en la tabla 5.1. Como esquema de decisión, se empleó una votación por mayoría cualificada.

| | |
|---|---|
| Objective | Binary Logistic |
| Booster | gbtree |
| Colsample by Tree | 1 |
| Learning Rate | 0.300000012 |
| Max. Depth | 6 |
| Min. Child Weight | 1 |
| Num. of Estimators | 100 |
| $\alpha$ | 0 |
| $\lambda$ | 1 |
| Tree Method | Exact |

Tabla 5.1: Configuración de los hiperparámetros para XGBoost

La elección de XGBoost como modelo base responde a dos factores. En primer lugar, este algoritmo es ampliamente reconocido por su eficiencia computacional y su capacidad para manejar conjuntos de datos de gran tamaño y con una alta dimensionalidad, sin comprometer el rendimiento predictivo [27, 47, 101]. En segundo lugar, en el estudio realizado en el capítulo 3 con el conjunto de datos `Seguros`, XGBoost obtuvo resultados superiores frente a diez algoritmos de ML diferentes, confirmando su robustez. No obstante, aunque presentamos los resultados con XGBoost, es importante destacar que la elección del modelo base para entrenar e implementar `PCM` podría haber sido cualquier otro algoritmo.

Partiendo del modelo *ensemble* previamente entrenado, se ha evaluado su rendimiento utilizando el método propuesto frente a cinco métodos alternativos. A continuación, se describen con detalle las configuraciones empleadas en cada uno de los métodos analizados:

a) En primer lugar, el método propuesto, `PCM`, se basa en la aplicación del algoritmo NSGA-II (véase Algoritmo 5) para explorar de forma eficiente el espacio de decisión. La configuración de NSGA-II empleada se recoge en la tabla 5.2. Tras un total de 750

generaciones, el conjunto de soluciones no dominadas define la *curva de Rendimiento* asociada al modelo.

b) El segundo método corresponde a la configuración clásica del modelo *ensemble*, que denominamos `CA`. En este caso, tanto los umbrales individuales $T_i$ como el umbral del esquema de votación $T_D$ se fijan en 0.5, sin aplicar ningún tipo de ajuste o proceso de optimización posterior. Este planteamiento sirve como referencia o punto de partida para evaluar las mejoras introducidas por los demás métodos.

c) El tercer método, denominado `Gmean`, consiste en la optimización del GMean mediante la aplicación de un algoritmo genético (AG) [64]. En este caso, la función objetivo busca maximizar la media geométrica de TPR y TNR. El algoritmo genético ajusta de forma conjunta los umbrales $T_i$ y $T_D$, evaluando el rendimiento global del *ensemble* y seleccionando, tras 500 generaciones, el mejor individuo encontrado. La configuración detallada de este algoritmo se presenta en la tabla 5.2. Cabe destacar que esta estrategia solo se aplica para la optimización de curvas ROC.

d) El cuarto método, denominado `StackOpt`, corresponde al aprendizaje por apilamiento (*Stacking learning*) [95]. En este caso, se entrena un metaclasificador a partir de las predicciones de los modelos base del modelo *ensemble*, es decir, utilizando como entrada las salidas generadas por cada $m_i$. Se analizaron cuatro opciones de metaclasificador: XGBoost, Random Forest, Logistic Regression y Multilayer perceptron.

e) El quinto método, `WOpt`, es una optimización bayesiana de pesos mediante la herramienta Optuna [4]. Esta técnica busca ajustar automáticamente la ponderación de las predicciones de cada modelo base del modelo *ensemble*, maximizando el rendimiento global de la combinación resultante. A diferencia de los métodos anteriores, se reconfigura la influencia relativa de cada modelo base.

f) El último método analizado, denominado `HiperOpt`, emplea una optimización bayesiana para ajustar los hiperparámetros de los modelos base que constituyen el modelo *ensemble*, utilizando la herramienta Optuna. Esta estrategia tiene como objetivo mejorar el rendimiento individual de cada clasificador antes de llevar a cabo la optimización conjunta del *ensemble*. Esta optimización se aplica junto con los metodos `PCM` y `CA`.

Tanto en el método `Gmean` como en `PCM`, cada individuo dentro de la población evolutiva está compuesto por un total de 51 genes, donde los primeros 50 representan los valores $T_i$ correspondientes a cada uno de los modelos XGBoost que conforman $M$, y el gen adicional codifica $T_D$.

Los seis métodos analizados utilizan exactamente los mismos conjuntos de datos, y cada proceso de optimización se ha ejecutado diez veces, es decir, utilizando un fold (pliegue) de datos de entrenamiento y prueba diferente en cada ejecución (*10-folds*). Es importante destacar que, para cada fold, tanto el conjunto de entrenamiento como el de prueba son idénticos entre los seis métodos, así como el modelo *ensemble* previamente entrenado.

| | AG | NSGA-II |
|---|---|---|
| Objetivos | Max GMean | Max. TPR<br>Min. FPR |
| Longitud del cromosoma | 51 | 51 |
| Selección | Torneo binario [113] | |
| Método de cruce | Un punto | Simulación Binaria [32] |
| Ratio de cruce | 0.9 | 1 |
| Método de mutación | Individual | Polinomial [132] |
| Ratio de mutación | 0.019 | 0.019 |
| Generaciones | 500 | 750 |
| Población | 100 | 200 |

Tabla 5.2: Configuración para el AG (`Gmean`) y NSGA-II (`Performance Curve Mapping`).

De este modo, la única diferencia entre los métodos radica en la fase de optimización, lo que permite una comparación justa y controlada del impacto de cada estrategia sobre el rendimiento final del clasificador.

La figura 5.2 ilustra el flujo de trabajo seguido por los diferentes métodos para cada una de las ejecuciones. Para cada conjunto de entrenamiento, se generan $n = 50$ subconjuntos balanceados mediante la técnica BUE descrita en el capítulo 3 y con ellos se entrenan los modelos base que conforman $M$. A partir del conjunto $M$ de modelos base, se construye el modelo *ensemble* $E$ incorporando el esquema de votación.

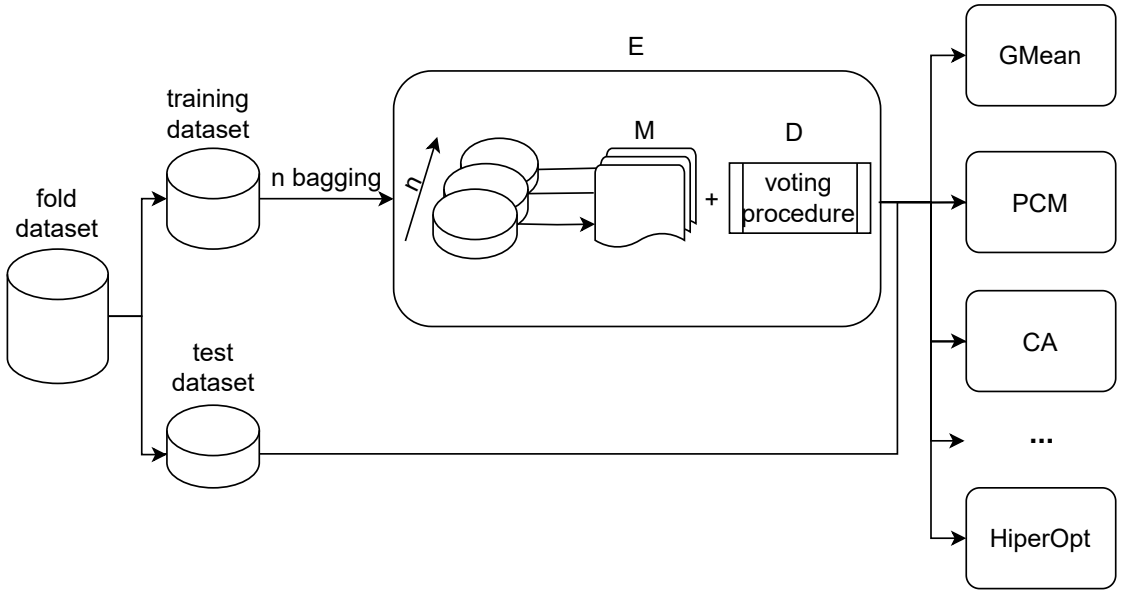



Figura 5.2: Flujo de trabajo de las ejecuciones para cada fold. El conjunto de datos para cada fold se divide en conjuntos de entrenamiento y prueba. El clasificador *ensemble* ($E$) se entrena utilizando $n$ subconjuntos del conjunto de entrenamiento. Posteriormente, se aplican seis optimizaciones sobre el modelo *ensemble* entrenado y el conjunto de prueba.

Una vez definidos los modelos, evaluamos el rendimiento de los distintos métodos em-

pleando un conjunto de métricas que ofrecen una visión integral del comportamiento del modelo *ensemble*. En primer lugar, se realizaron comparaciones visuales de las *curvas de Rendimiento*, con el fin de analizar la distribución de las soluciones en el *espacio de Rendimiento* y resaltar aquellas regiones en las que `PCM` ofreció mejoras sustanciales respecto a los métodos alternativos. Estas representaciones gráficas permitieron apreciar, de forma intuitiva, cómo cada estrategia gestionó el compromiso entre las métricas involucradas (como TPR y FPR en las curvas ROC).

En segundo lugar, se empleó el AUC de cada *curva de Rendimiento* como métrica cuantitativa para valorar la capacidad de discriminación del modelo. Esta métrica resume el rendimiento del clasificador a lo largo de todos los posibles umbrales de decisión y resultó especialmente relevante en presencia de un fuerte desbalanceo entre clases.

Por último, se utilizó el Índice de Youden, $J$ (ecuación (5.3)) [42], para identificar el punto de corte óptimo en la curva ROC. Esta métrica resulta útil en escenarios donde tanto los falsos positivos como los falsos negativos conllevan costes significativos.

$$J = \text{TPR} + \text{TNR} - 1 \tag{5.3}$$

Con la incorporación de estas métricas, aseguramos una evaluación completa y robusta que captura distintos aspectos del rendimiento del modelo *ensemble* bajo cada uno de los métodos analizados.

## 5.4 Análisis de resultados en curvas ROC

Las curvas ROC permiten representar visualmente el compromiso entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) para distintos umbrales de decisión. Esta comparativa visual se realizó con los métodos `PCM`, `CA` y `Gmean`, y los dos conjuntos de datos descritos anteriormente.

### 5.4.1 Comparación visual

La figura 5.3 muestra las curvas ROC obtenidas para `CA` y `PCM` en los diez folds para ambos conjuntos de datos. También se representan las soluciones obtenidas por `Gmean` como puntos. En el caso de `PCM`, se incluyen todas las soluciones generadas durante las 750 generaciones del proceso evolutivo. Los resultados evidencian una ventaja consistente de `PCM`, ya que supera todas las soluciones obtenidas con `CA` en la totalidad de los folds.

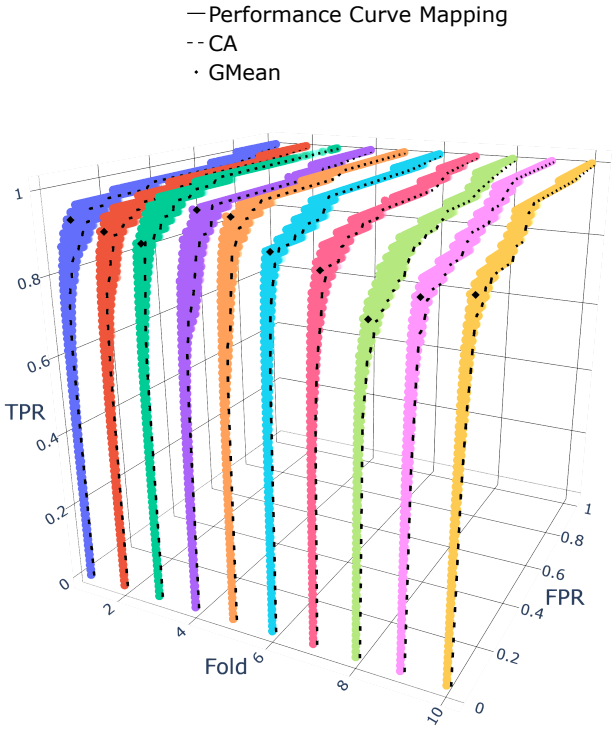
En el conjunto de datos `Seguros` (figura 5.3a), las soluciones generadas mediante el método `Gmean` logran mejorar los mejores puntos alcanzados por `CA` para un mismo nivel de FPR. No obstante, todas estas soluciones son igualadas o superadas por `PCM`, que alcanza sistemáticamente mejores valores tanto de TPR como de FPR.

Una tendencia similar se observa en el conjunto de datos `GenObIA` (figura 5.3b). Aunque el

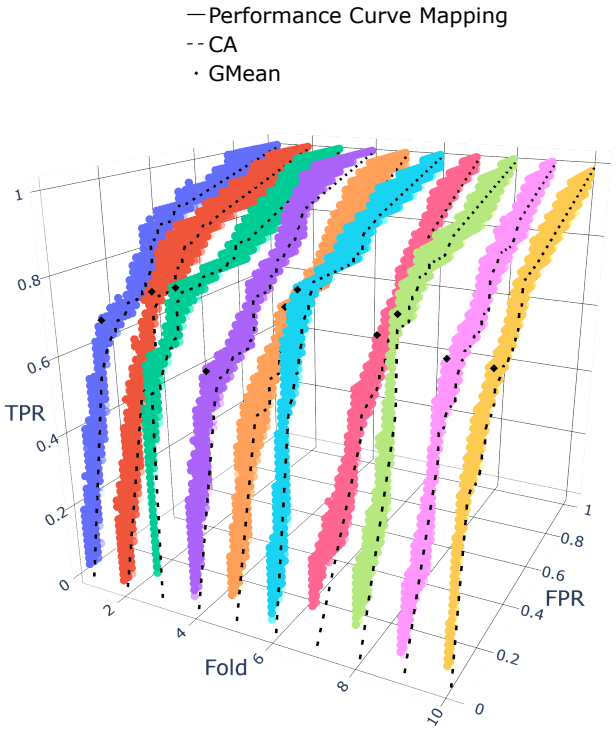
rendimiento de CA es peor en este conjunto, el método Gmean consigue mejorar los valores de TPR y FPR en todos los folds. Sin embargo, también en este caso, PCM logra soluciones que igualan o superan a las obtenidas por Gmean en cada fold.

Para facilitar la interpretación visual de los resultados, la figura 5.4 muestra una vista frontal (desde el eje FPR) de la figura 5.4, representando los resultados de CA, PCM y Gmean para los conjuntos Seguros (figura 5.4a) y GenObIA (figura 5.4b). A diferencia de la figura anterior, en la que se representaban todas las soluciones obtenidas por PCM, en este caso solo se consideran las soluciones no dominadas de cada fold para construir la envolvente (en color verde), que delimita el conjunto de puntos extremos alcanzados por las curvas ROC. Por su parte, la envolvente azul corresponde a las soluciones obtenidas mediante el método CA. Los puntos y triángulos negros corresponden a las soluciones obtenidas con Gmean. El área de la envolvente se utiliza como medida de la dispersión de las soluciones obtenidas, donde valores más pequeños indican una menor variabilidad entre ejecuciones.

Como se aprecia en ambas figuras, PCM domina claramente la región superior izquierda del espacio ROC, lo que implica una mejora simultánea del TPR y del FPR frente a Gmean y CA. Para el conjunto de datos Seguros, el área cubierta por la envolvente de PCM alcanza un valor de 0.0467 frente al 0.0883 de CA, lo que supone una reducción del 47.00 %. En el caso del conjunto de datos GenObIA, la envolvente de PCM es de 0.1338 frente al 0.1649 del método CA, representando una reducción del 18.86 %. Esta disminución del área en ambos conjuntos de datos indica que las soluciones obtenidas por PCM presentan una menor dispersión, lo que sugiere un comportamiento más consistente y estable del modelo frente a variaciones en los cojuntos de entrenamiento y prueba.

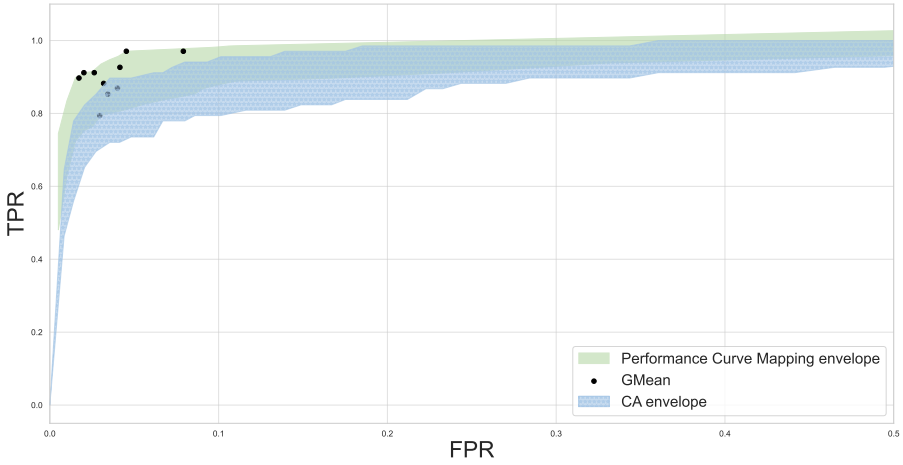



(a) Conjunto de datos `Seguros`.

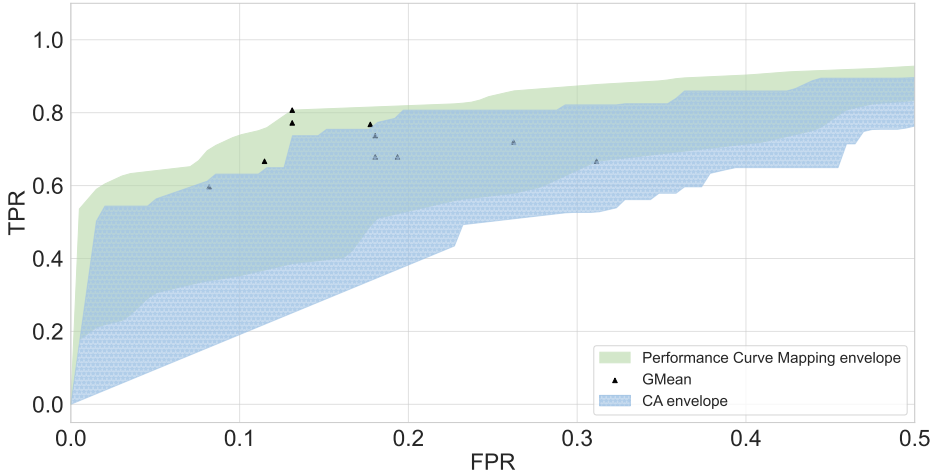



(b) Conjunto de datos `GenObIA`.

Figura 5.3: Curvas ROC de los métodos `Performance Curve Mapping` (`PCM`) y `CA` para los diez folds. Para `PCM` se muestran todas las soluciones obtenidas tras las 750 generaciones del algoritmo evolutivo.




(a) Conjunto de datos `Seguros`.




(b) Conjunto de datos `GenObIA`.

Figura 5.4: Proyecciones de las figuras 5.3a y 5.3b. Representan envolventes que abarcan los puntos más extremos de las curvas ROC para `PCM` (en color verde) y `CA`(en color azul), mientras que los resultados de `Gmean` se muestran como puntos y triángulos individuales. Solo se han considerado los frentes de Pareto de las soluciones de `PCM` para cada fold.

### 5.4.2 Comparación del rendimiento con GMean

Tras un primer contacto visual de los resultados en la tabla 5.3 se recogen los valores máximos de GMean obtenidos por `PCM`, `Gmean` y `CA`. En el conjunto de datos `Seguros`, `PCM` obtiene un valor medio de 0.9337, superando tanto a `Gmean` (0.9299) como a `CA` (0.9070). En el conjunto de datos `GenObIA`, `PCM` también obtiene el mejor resultado, con un valor promedio GMean de 0.7651. Este desempeño supera ligeramente al de `Gmean` (0.7627) y de forma más notable al de `CA` (0.7353).

| | Seguros | | | GenObIA | | |
|---|---|---|---|---|---|---|
| Fold | PCM | Gmean | CA | PCM | Gmean | CA |
| 1 | **0.9485** | 0.9423 | 0.9154 | **0.7693** | 0.7682 | 0.7445 |
| 2 | **0.9430** | 0.9422 | 0.9197 | **0.7969** | 0.7771 | 0.7601 |
| 3 | **0.9499** | 0.9389 | 0.9314 | **0.8189** | **0.8189** | 0.8019 |
| 4 | **0.9454** | **0.9454** | 0.9140 | **0.7399** | **0.7399** | 0.7120 |
| 5 | **0.9626** | **0.9626** | 0.9320 | **0.7284** | **0.7284** | 0.7203 |
| 6 | **0.9451** | **0.9451** | 0.9270 | **0.8374** | 0.8372 | 0.8051 |
| 7 | **0.9305** | 0.9242 | 0.9081 | **0.6775** | **0.6775** | 0.6313 |
| 8 | **0.8895** | 0.8777 | 0.8551 | **0.7975** | 0.7946 | 0.7760 |
| 9 | **0.9083** | 0.9075 | 0.8831 | **0.7397** | **0.7397** | 0.6769 |
| 10 | **0.9136** | **0.9136** | 0.8845 | **0.7457** | **0.7457** | 0.7246 |
| **media** | **0.9337** | 0.9299 | 0.9070 | **0.7651** | 0.7627 | 0.7353 |
| **std. dev.** | **0.0216** | 0.0233 | 0.0238 | 0.0457 | **0.0444** | 0.0517 |

Tabla 5.3: Valores máximos de GMean para cada fold utilizando los enfoques `Performance Curve Mapping` (`PCM`), `CA` y `Gmean`. En **negrita** se representan los mejores resultados para cada conjunto de datos.

Para determinar si estas diferencias son estadísticamente significativas, se aplicó una prueba t de Student pareada con un nivel de confianza del 95 % ($\alpha = 0.05$). En el conjunto de datos `Seguros`, el valor obtenido para `PCM` frente `Gmean` fue $t = 2.45$ con un valor $p = 0.0365$, lo que permite rechazar la hipótesis nula al utilizar un umbral de significación ($\alpha = 0.05$). Estos resultados indican que las diferencias observadas entre ambos métodos son estadísticamente significativas. Por tanto, `PCM` presenta diferencias significativas también frente a `CA`, con un valor $t = 14.18$ y un valor $p = 1.83 \times 10^{-7}$. Del mismo modo, la comparación entre `Gmean` y `CA` obtuvo un valor $t = 9.77$ y un $p = 4.33 \times 10^{-6}$, lo que también indica diferencias estadísticamente significativas entre estos dos métodos.

Por el contrario, en el conjunto de datos `GenObIA` se obtuvo un valor $t = 1.23$ con un valor $p = 0.251$, lo que no permite rechazar la hipótesis nula. En este caso, no se puede afirmar que las diferencias observadas entre `PCM` y `Gmean` sean estadísticamente significativas. No obstante, sí se observaron diferencias significativas entre `PCM` y `CA` ($t = 6.01$, $p = 0.0002$), así como entre `Gmean` y `CA` ($t = 5.37$, $p = 0.0004$), lo que demuestra una mejora de rendimiento de ambos métodos frente al enfoque base.

### 5.4.3 Comparación del rendimiento con AUC

A continuación, se presenta un análisis de resultados con la métrica AUC para evaluar el rendimiento del modelo *ensemble* bajo distintos métodos de optimización. Este análisis se estructura en cuatro bloques diferenciados.

1) Comparación de los resultados obtenidos con la propuesta `PCM` frente a `CA` utilizando la configuración original de los modelos base (ver tabla 5.1).

2) Comparación de los resultados obtenidos por `HiperOpt`+`PCM` e `HiperOpt`+`CA` frente a `PCM` y `CA`.

3) Comparación de los resultados de `StackOpt` con `PCM` y `CA`.

4) Comparación de los resultados de `WOpt`, con `PCM` y `CA`.

Este análisis progresivo permite valorar, por un lado, el impacto individual de cada método, y por otro, su efectividad relativa frente a la propuesta `PCM`, que constituye el eje central de este estudio.

En primer lugar, la tabla 5.4 recoge los valores de AUC para `PCM` y `CA`, . Para esta comparativa solo se tienen en cuenta las columnas que aparecen sin la etiqueta `HiperOpt`. Los resultados de `PCM` superan de forma sistemática al enfoque clásico en todos los folds y en ambos conjuntos de datos.

En el conjunto de datos `Seguros`, `PCM` alcanza un AUC medio de 0.9733 frente al 0.9502 de `CA`, lo que representa una mejora del 46.39 % dentro del rango de mejora alcanzable, definido como la diferencia entre el valor inicial de AUC (`CA`) y el valor máximo teórico (1). En el conjunto de datos `GenObIA`, `PCM` alcanza un valor medio de 0.8338, superando al 0.7635 obtenido por `CA` y representando una mejora del 29.73 % dentro del rango de mejora alcanzable.

Con el objetivo de comparar estadísticamente los resultados AUC, se realizaron para cada conjunto de datos pruebas t de Student para muestras pareadas. En el caso del conjunto de datos `Seguros`, la prueba se realizó con un nivel de confianza del 99 % ($\alpha = 0.01$) y 18 grados de libertad, obteniéndose un valor $t = 10.35$ con un valor $p < \alpha$, lo que permite rechazar la hipótesis nula y confirmar la existencia de diferencias significativas entre ambos métodos.

De forma análoga, en el conjunto de datos `GenObIA` se obtuvo un valor $t = 17.51$, también con un $p < \alpha$, reforzando la conclusión de que la propuesta `PCM` ofrece una mejora significativa respecto al enfoque clásico `CA` en ambos escenarios.

A continuación, se aborda el segundo bloque del análisis. Dado que inicialmente se empleó la misma configuración de hiperparámetros para todos los modelos base (véase tabla 5.1), se propuso la incorporación de un proceso de optimización bayesiana, mediante la herramienta Optuna (`HiperOpt`), con el objetivo de ajustar de forma individual los hiperparámetros de los modelos base que componen el modelo *ensemble*. Esta optimiza-

| | Seguros | | | | GenObIA | | | |
|---|---|---|---|---|---|---|---|---|
| Fold | PCM HiperOpt | CA HiperOpt | PCM | CA | PCM HiperOpt | CA HiperOpt | PCM | CA |
| 1 | **0.9760** | 0.9550 | 0.9740 | 0.9534 | **0.8820** | 0.8073 | 0.8599 | 0.8001 |
| 2 | **0.9846** | 0.9646 | 0.9812 | 0.9603 | **0.9019** | 0.8193 | 0.8609 | 0.7876 |
| 3 | **0.9935** | 0.9820 | 0.9908 | 0.9775 | **0.8918** | 0.8267 | 0.8727 | 0.8216 |
| 4 | **0.9803** | 0.9581 | 0.9775 | 0.9545 | **0.8738** | 0.7706 | 0.8398 | 0.7602 |
| 5 | **0.9917** | 0.9775 | 0.9902 | 0.9734 | **0.8446** | 0.7578 | 0.8087 | 0.7280 |
| 6 | **0.9848** | 0.9661 | 0.9822 | 0.9632 | **0.9073** | 0.8564 | 0.8861 | 0.8182 |
| 7 | **0.9720** | 0.9431 | 0.9696 | 0.9434 | **0.7656** | 0.6869 | 0.7503 | 0.6594 |
| 8 | **0.9541** | 0.9094 | 0.9468 | 0.9084 | **0.8654** | 0.8145 | 0.8384 | 0.7762 |
| 9 | **0.9585** | 0.9331 | 0.9546 | 0.9319 | **0.8253** | 0.7557 | 0.7896 | 0.7098 |
| 10 | **0.9715** | 0.9418 | 0.9651 | 0.9362 | **0.8992** | 0.8015 | 0.8309 | 0.7740 |
| media | **0.9767** | 0.9531 | 0.9733 | 0.9502 | **0.8657** | 0.7897 | 0.8338 | 0.7635 |
| std. Dev. | 0.0130 | 0.0463 | 0.0144 | 0.0208 | 0.0438 | 0.04815 | 0.0412 | 0.0511 |

Tabla 5.4: Valores AUC para cada fold utilizando los enfoques HiperOpt, PCM y CA. Los mejores resultados para cada fold y conjunto de datos están resaltados en **negrita**.

ción preliminar pretende mejorar la configuración de cada modelo base antes de aplicar cualquier estrategia de optimización.

La optimización se realizó de manera independiente para cada modelo, manteniendo la premisa de no introducir dependencias entre ellos. Esta decisión se fundamenta en trabajos previos que han demostrado que una optimización conjunta de los modelos base puede reducir la diversidad y generar correlaciones artificiales, lo cual perjudica la capacidad de generalización del modelo *ensemble* [65, 68, 104].

En la tabla 5.4 se presentan los resultados obtenidos tras aplicar esta optimización de hiperparámetros, tanto para el enfoque PCM como para CA. Los valores obtenidos con HiperOpt+PCM y HiperOpt+CA muestran una mejora general del rendimiento frente a sus versiones sin HiperOpt. No obstante, HiperOpt+PCM sigue superando claramente a HiperOpt+CA, lo que refuerza la efectividad de la técnica propuesta.

En concreto, HiperOpt+PCM alcanza un valor medio de AUC de 0.9767 en el conjunto de datos Seguros, frente al 0.9531 de HiperOpt+CA. Para el conjunto de datos GenObIA, los valores medios son de 0.8657 y 0.7897, respectivamente. En la tabla 5.5 se presenta una comparación detallada entre los métodos PCM y CA, tanto con como sin HiperOpt. Para cada comparación se muestra la mejora relativa obtenida dentro del rango de mejora alcanzable.

La diferencia entre HiperOpt+PCM y PCM es del 12.73 % para el conjunto de datos Seguros y del 19.19 % para el conjunto de datos GenObIA. Al comparar la mejora relativa obtenida por HiperOpt+PCM frente a CA, se observa un incremento notable en el rendimiento. En el conjunto de datos Seguros, HiperOpt+PCM alcanza una mejora del 53.21 %, lo que supone una diferencia de aproximadamente 6.8 puntos porcentuales respecto a los 46.39 %

| Comparación | M Seguros (%) | M GenObIA (%) PCM |
|---|---|---|
| (HiperOpt+PCM) vs. (HiperOpt+CA) | 50.32 % | 36.14 % |
| (HiperOpt+PCM) vs. (CA) | 53.21 % | 43.21 % |
| (HiperOpt+PCM) vs. (PCM) | 12.73 % | 19.19 % |
| (PCM) vs. (CA) | 46.39 % | 29.73 % |
| (PCM) vs. (HiperOpt+CA) | 43.07 % | 20.97 % |
| (HiperOpt+CA) vs. (CA) | 5.92 % | 11.01 % |

Tabla 5.5: Mejora relativa (M) en términos AUC dentro del rango de mejora alcanzable.

logrados únicamente con PCM. De forma análoga, en el conjunto de datos GenObIA, la diferencia es de 13.5 puntos porcentuales, pasando del 29.73 % con PCM al 43.21 % con HiperOpt+PCM.

Estos resultados indican que, si bien la optimización de hiperparámetros mediante Optuna tiene un efecto positivo adicional, la mayor parte de la mejora global se consigue gracias a la estrategia de optimización del espacio de decisión implementada con PCM.

Por otro lado, la comparación entre HiperOpt+CA y CA muestra una mejora menor (5.92 % en el conjunto de datos Seguros y 11.01 % en el conjunto de datos GenObIA), lo que sugiere que el ajuste de hiperparámetros por sí solo tiene un efecto limitado en comparación con PCM.

Para comprobar si existen diferencias significativas entre los resultados obtenidos por HiperOpt+PCM y PCM, se ha aplicado una prueba t de Student pareada con un nivel de confianza del 99.0 %. En el conjunto de datos Seguros, el valor obtenido fue $t = 6.52$ con un valor $p < \alpha$, mientras que en el conjunto de datos GenObIA se obtuvo $t = 9.52$, también con $p < \alpha$. Estos valores permiten rechazar la hipótesis nula en ambos casos, lo que indica que las diferencias observadas entre ambos métodos son estadísticamente significativas.

El tercer bloque del análisis se centra en la comparación del método de aprendizaje por apilamiento (StackOpt) frente a los métodos PCM y CA. En este caso, se evalúa el rendimiento de distintos metaclasificadores entrenados a partir de las predicciones de los modelos base del modelo *ensemble*.

En las tablas 5.6 y 5.7 se presentan los valores AUC obtenidos para cuatro metaclasificadores utilizados en StackOpt: XGBoost (XGB), Random Forest (RF), Regresión Logística (LR) y Perceptrón Multicapa (MLP), junto con los valores obtenidos por PCM y el enfoque clásico (CA).

Los resultados muestran que ninguno de los metaclasificadores logra superar el desempeño alcanzado por PCM. En el conjunto de datos Seguros, el mejor resultado entre los metaclasificadores se obtiene con MLP (media de AUC = 0.9442), que queda por debajo tanto del valor alcanzado por CA (0.9502) como del resultado de PCM (0.9733). En el caso del

conjunto de datos GenObIA, el comportamiento es similar, aunque con un matiz relevante: el mejor metaclasificador, en este caso la Regresión Logística, alcanza un AUC medio de 0.7661, superando levemente al obtenido por CA, que alcanza 0.7635. No obstante, ambos valores siguen claramente por debajo del rendimiento alcanzado por PCM, con un AUC medio de 0.8338.

Dado que los resultados obtenidos previamente muestran que PCM mejora de forma estadísticamente significativa al método clásico CA, y ya que WOpt apenas logra igualar o superar ligeramente a CA en algunos folds, sin mostrar mejoras consistentes, se asume que las diferencias observadas entre PCM y StackOpt también son estadísticamente significativas.

Estos resultados sugieren que, al menos en los casos analizados, un esquema de votación con umbrales optimizados puede ser más eficaz que un metaclasificador. Una posible explicación de este comportamiento podría ser la pérdida de balanceo de clases durante la fase de entrenamiento del metaclasificador, ya que este se entrena sobre el conjunto completo de datos, a diferencia de los modelos base del modelo *ensemble*, que se entrenan con subconjuntos balanceados generados mediante la técnica BUE.

| | StackOpt | | | | PCM | CA |
|---|---|---|---|---|---|---|
| Fold | XGB | RF | LR | MLP | | |
| 0 | 0.899581 | 0.938011 | 0.943068 | 0.947093 | **0.9740** | 0.9534 |
| 1 | 0.886233 | 0.943698 | 0.953141 | 0.954483 | **0.9812** | 0.9603 |
| 2 | 0.928474 | 0.962977 | 0.972289 | 0.973349 | **0.9908** | 0.9775 |
| 3 | 0.866924 | 0.919070 | 0.950155 | 0.954068 | **0.9775** | 0.9545 |
| 4 | 0.925855 | 0.954178 | 0.965162 | 0.968176 | **0.9902** | 0.9734 |
| 5 | 0.932741 | 0.946742 | 0.954528 | 0.956725 | **0.9822** | 0.9632 |
| 6 | 0.889581 | 0.931869 | 0.939037 | 0.941978 | **0.9696** | 0.9434 |
| 7 | 0.857465 | 0.896717 | 0.907245 | 0.911975 | **0.9468** | 0.9084 |
| 8 | 0.853593 | 0.889735 | 0.916873 | 0.919047 | **0.9546** | 0.9319 |
| 9 | 0.778144 | 0.892948 | 0.911452 | 0.914762 | **0.9651** | 0.9362 |
| media | 0.881859 | 0.927595 | 0.941295 | 0.944166 | **0.9733** | 0.9502 |
| std. Dev. | 0.046505 | 0.026607 | 0.022556 | 0.021948 | 0.0144 | 0.0208 |

Tabla 5.6: Resultados AUC de los métodos StackOpt, PCM y CA en el conjunto de datos Seguros. En **negrita** se resaltan los mejores resultados

Por último, se analiza el enfoque WOpt, que consiste en ajustar los pesos de los modelos base del modelo *ensemble* mediante optimización bayesiana. Este método busca mejorar la combinación final de predicciones ponderando adecuadamente la contribución de cada modelo base que forma el modelo *ensemble*.

La tabla 5.8 muestra los resultados AUC para WOpt, PCM y CA en ambos conjuntos de datos. En el conjunto de datos Seguros, WOpt obtiene un valor medio de 0.9507, que representa una ligera mejora respecto a CA (0.9502), pero sigue estando por debajo del resultado

| | StackOpt | | | | PCM | CA |
|---|---|---|---|---|---|---|
| Fold | XGB | RF | LR | MLP | | |
| 0 | 0.790624 | 0.793069 | 0.793788 | 0.788323 | **0.8599** | 0.8001 |
| 1 | 0.785016 | 0.800546 | 0.786022 | 0.777107 | **0.8609** | 0.7876 |
| 2 | 0.812913 | 0.813057 | 0.824274 | 0.821973 | **0.8727** | 0.8216 |
| 3 | 0.739431 | 0.759563 | 0.786022 | 0.783722 | **0.8398** | 0.7602 |
| 4 | 0.715991 | 0.724044 | 0.740006 | 0.737417 | **0.8087** | 0.7280 |
| 5 | 0.803854 | 0.814351 | 0.817947 | 0.815933 | **0.8861** | 0.8182 |
| 6 | 0.654156 | 0.651855 | 0.673857 | 0.682197 | **0.7503** | 0.6594 |
| 7 | 0.766705 | 0.770017 | 0.763537 | 0.768433 | **0.8384** | 0.7762 |
| 8 | 0.718606 | 0.705933 | 0.705357 | 0.707949 | **0.7896** | 0.7098 |
| 9 | 0.756733 | 0.766833 | 0.770199 | 0.774883 | **0.8309** | 0.7740 |
| media | 0.754403 | 0.759927 | 0.766101 | 0.765794 | **0.8338** | 0.7635 |
| std. Dev. | 0.048654 | 0.052244 | 0.047767 | 0.044542 | 0.0412 | 0.0511 |

Tabla 5.7: Resultados AUC de los métodos StackOpt, PCM y CA en el conjunto de datos GenObIA. En **negrita** se resaltan los mejores resultados

alcanzado por PCM (0.9733). Un comportamiento similar se observa en el conjunto de datos GenObIA, donde WOpt alcanza un AUC de 0.7755, frente a los 0.7635 de CA y los 0.8338 de PCM.

| | Seguros | | | GenObIA | | |
|---|---|---|---|---|---|---|
| Fold | WOpt | PCM | CA | Optuna weights | PCM | CA |
| 0 | 0.9502 | **0.9740** | 0.9534 | 0.8041 | **0.8599** | 0.8001 |
| 1 | 0.9590 | **0.9812** | 0.9603 | 0.7946 | **0.8609** | 0.7876 |
| 2 | 0.9765 | **0.9908** | 0.9775 | 0.8320 | **0.8727** | 0.8216 |
| 3 | 0.9538 | **0.9775** | 0.9545 | 0.7960 | **0.8398** | 0.7602 |
| 4 | 0.9740 | **0.9902** | 0.9734 | 0.7509 | **0.8087** | 0.7280 |
| 5 | 0.9636 | **0.9822** | 0.9632 | 0.8274 | **0.8861** | 0.8182 |
| 6 | 0.9455 | **0.9696** | 0.9434 | 0.6856 | **0.7503** | 0.6594 |
| 7 | 0.9180 | **0.9468** | 0.9084 | 0.7718 | **0.8384** | 0.7762 |
| 8 | 0.9289 | **0.9546** | 0.9319 | 0.7151 | **0.7896** | 0.7098 |
| 9 | 0.9374 | **0.9651** | 0.9362 | 0.7778 | **0.8309** | 0.7740 |
| media | 0.9507 | **0.9733** | 0.9502 | 0.7755 | **0.8338** | 0.7635 |
| std. Dev. | 0.0178 | 0.0144 | 0.0208 | 0.0469 | 0.0412 | 0.0511 |

Tabla 5.8: Resultados AUC de WOpt, PCM y CA en ambos conjuntos de datos.

Cabe destacar que, a pesar de ajustar los pesos asignados a cada modelo base mediante optimización bayesiana, el enfoque WOpt no logra mejorar de forma notable el rendimiento respecto a CA. Esto puede deberse a que el esquema de votación por mayoría cualificada ya

actúa como un mecanismo eficaz de agregación, aprovechando la diversidad de los modelos base. Además, una reasignación de pesos sobre modelos ya equilibrados y previamente entrenados con datos balanceados mediante la técnica BUE aporta un margen de mejora limitado para este método.

Al igual que en el caso de `StackOpt`, no se han realizado pruebas estadísticas adicionales entre `PCM` y `WOpt`. Dado que `PCM` ha demostrado mejoras significativas respecto a `CA`, y que `WOpt` no logra superar a `CA`, puede asumirse que `PCM` también presenta un rendimiento superior frente a `WOpt`.

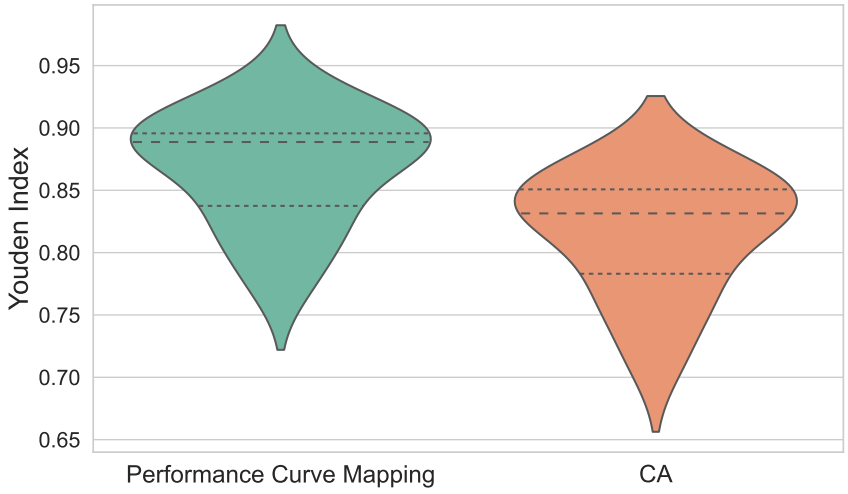
### 5.4.4 Comparación del rendimiento con el Índice de Youden

Además del análisis basado en AUC, se ha evaluado el rendimiento de `PCM` y `CA` mediante el índice de Youden, una métrica que permite valorar simultáneamente el TPR y el FPR.

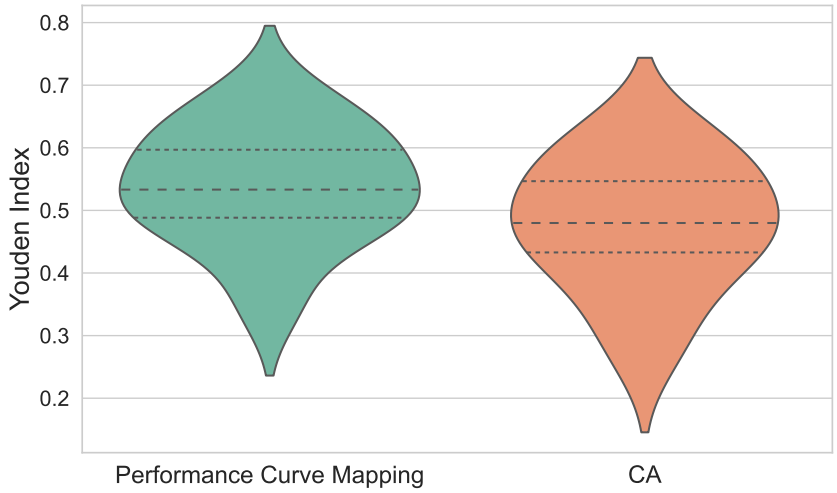
La figura 5.5 presenta diagramas de violín que recogen la distribución de los valores máximos del índice de Youden obtenidos a lo largo de los folds para los conjuntos `Seguros` y `GenObIA`. En ambos casos, `PCM` presenta una mediana superior y un rango intercuartílico más estrecho que `CA`, lo que refleja un comportamiento más estable y preciso.

En términos cuantitativos, para el conjunto de datos `Seguros`, `PCM` alcanza un índice de Youden promedio de 0.8639, frente a los 0.8159 obtenidos con `CA`. Para el conjunto de datos `GenObIA`, los valores medios son de 0.5381 y 0.4760, respectivamente, lo que refuerza la superioridad de `PCM` también en contextos más complejos.

Para comprobar si las diferencias observadas son estadísticamente relevantes, se ha aplicado una prueba t de Student pareada. En el conjunto de datos `Seguros`, se obtuvo un valor $t = 15.28$ con $p < \alpha$, y en el conjunto de datos `GenObIA`, $t = 6.37$ también con $p < \alpha$. En ambos casos se rechaza la hipótesis nula, lo que confirma que el enfoque `PCM` también proporciona una mejora significativa frente al enfoque `CA` en esta métrica.




(a) Conjunto de datos `Seguros`.




(b) Conjunto de datos `GenObIA`.

Figura 5.5: Distribución del índice de Youden para los enfoques `PCM` y `CA`.

## 5.5 Análisis de resultados en curvas PRC

Además de aplicar PCM en las ROC, adicionalmente evaluamos su rendimiento sobre las curvas PRC para los conjuntos de datos Seguros y GenObIA. La tabla 5.9 presenta los valores de AUC-PRC obtenidos con PCM y CA en todos los folds. AUC-PRC sigue la fórmula AUC de la sección 2.5, pero utilizando la precisión de la clase positiva en vez de TPR y el *recall* en vez de FPR.

### 5.5.1 Comparación del rendimiento con AUC-PRC

| | Seguros | | GenObIA | |
|---|---|---|---|---|
| Fold | PCM | CA | PCM | CA |
| 1 | **0.7406** | 0.5701 | **0.8627** | 0.8106 |
| 2 | **0.6991** | 0.5210 | **0.8596** | 0.7938 |
| 3 | **0.7622** | 0.5600 | **0.9008** | 0.8464 |
| 4 | **0.5899** | 0.4655 | **0.8488** | 0.7810 |
| 5 | **0.7039** | 0.5219 | **0.8230** | 0.7502 |
| 6 | **0.7112** | 0.5646 | **0.8785** | 0.8351 |
| 7 | **0.7122** | 0.5191 | **0.7468** | 0.6860 |
| 8 | **0.5965** | 0.4789 | **0.8162** | 0.7651 |
| 9 | **0.5470** | 0.4220 | **0.7782** | 0.7225 |
| 10 | **0.4782** | 0.4138 | **0.8412** | 0.7903 |
| media | **0.6541** | 0.5037 | **0.8356** | 0.7781 |
| std. dev. | 0.0943 | 0.0567 | 0.0463 | 0.0496 |

Tabla 5.9: Valores AUC-PRC para cada fold usando PCM y CA. En **negrita** se resaltan los mejores resultados.

Para el conjunto de datos Seguros, PCM alcanza un AUC-PRC promedio de 0.6541, en comparación con 0.5037 para CA, lo que corresponde a una mejora del 30.30 % dentro del rango de mejora alcanzable. La mayor ganancia por fold se obtiene en el fold 3, donde PCM mejora el AUC-PRC de 0.5600 a 0.7622 (45.95 % de mejora dentro del rango de mejora alcanzable). Este incremento directo en el valor del AUC-PRC refleja una mejora sustancial en la capacidad del modelo para detectar muestras positivas (clase minoritaria) en contextos con un alto desbalanceo de clases, una característica crítica en este conjunto de datos. Para el conjunto de datos GenObIA, PCM obtiene un AUC-PRC promedio de 0.8356, en comparación con 0.7781 para CA, lo que representa una mejora relativa del 25.94 %.

De nuevo para comprobar si existen diferencias significativas entre los resultados obtenidos por PCM y CA en términos de AUC-PRC, se ha aplicado una prueba t de Student pareada con un nivel de confianza del 99.0 %. En el conjunto de datos Seguros, el valor obtenido fue $t = 11.11$ con un valor $p < \alpha$, mientras que en el conjunto de datos GenObIA se obtuvo $t = 19.97$, también con $p < \alpha$, demostrando de nuevo que existen diferencias significativas

entre PCM y CA pero en este caso utilizando la métrica AUC-PRC.

Además, en la figura 5.6 se muestran las curvas PRC correspondientes a los enfoques PCM y CA. En este caso, únicamente se presenta el frente de Pareto obtenido por PCM. Cabe destacar que, aunque el conjunto de datos Seguros presenta un elevado desbalanceo de clases, es precisamente en este donde se observa el mayor porcentaje de mejora. Este resultado confirma que PCM no solo es eficaz en el contexto de las curvas ROC, sino que también mantiene un rendimiento destacado al aplicarse sobre las curvas PRC.

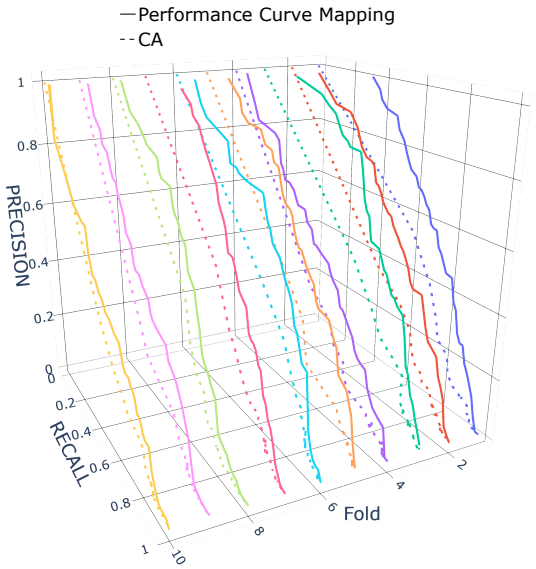
Al igual que en el análisis realizado sobre las curvas ROC, también se evaluó el rendimiento de los enfoques StackOpt y WOpt sobre las curvas PRC. En este contexto, se buscó comprobar si la combinación de predicciones mediante aprendizaje por apilamiento o el ajuste de pesos con optimización bayesiana lograban superar los resultados obtenidos por PCM. A continuación, se presentan los resultados comparativos de AUC-PRC para ambos conjuntos de datos. En primer lugar, las tablas 5.10 y 5.11 muestran los resultados de

| | StackOpt | | | | PCM | CA |
|---|---|---|---|---|---|---|
| Fold | XGB | RF | LR | MLP | | |
| 0 | 0.299262 | 0.307828 | 0.310151 | 0.355024 | **0.7406** | 0.5701 |
| 1 | 0.289125 | 0.247380 | 0.257933 | 0.311232 | **0.6991** | 0.5210 |
| 2 | 0.366702 | 0.359680 | 0.377870 | 0.433272 | **0.7622** | 0.5600 |
| 3 | 0.291734 | 0.206501 | 0.207953 | 0.252529 | **0.5899** | 0.4655 |
| 4 | 0.311491 | 0.324829 | 0.363991 | 0.390118 | **0.7039** | 0.5219 |
| 5 | 0.389434 | 0.324997 | 0.301900 | 0.356155 | **0.7112** | 0.5646 |
| 6 | 0.443638 | 0.416872 | 0.394708 | 0.457136 | **0.7122** | 0.5191 |
| 7 | 0.292259 | 0.227533 | 0.216747 | 0.274605 | **0.5965** | 0.4789 |
| 8 | 0.216843 | 0.190937 | 0.167981 | 0.198371 | **0.5470** | 0.4220 |
| 9 | 0.158310 | 0.154501 | 0.157493 | 0.173845 | **0.4782** | 0.4138 |
| media | 0.305880 | 0.276106 | 0.275673 | 0.320229 | **0.6541** | 0.5037 |
| std. Dev. | 0.081720 | 0.083427 | 0.087068 | 0.095412 | 0.0943 | 0.0567 |

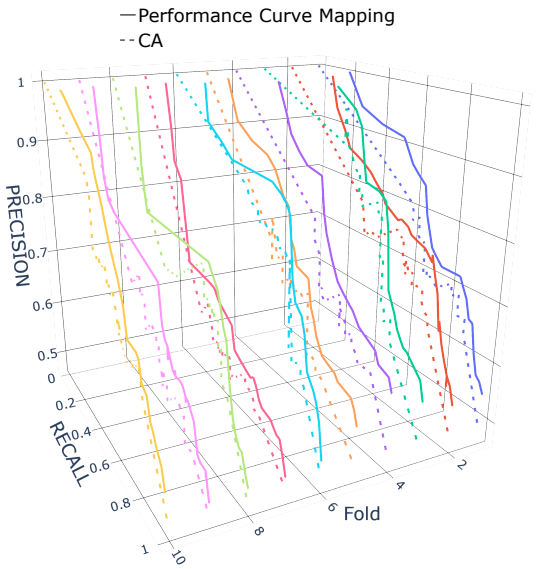
Tabla 5.10: Resultados AUC-PRC de StackOpt, PCM y CA en el conjunto de datos Seguros. En **negrita** se resaltan los mejores resultados

AUC-PRC obtenidos con StackOpt frente a PCM y CA. En el conjunto de datos Seguros, los cuatro metaclasificadores analizados (XGB, RF, LR y MLP) presentan valores claramente inferiores a los obtenidos por CA y por PCM. El mejor resultado de StackOpt (0.3550 con MLP) se sitúa muy por debajo de los 0.5037 alcanzados por CA y de los 0.6541 logrados por PCM. Esta diferencia es particularmente significativa si se considera el alto desbalanceo presente en este conjunto, lo que sugiere que StackOpt no logra capturar adecuadamente los casos de la clase minoritaria.

Una posible explicación, al igual que ocurría con las curvas ROC, radica en que, a diferencia de los modelos base de PCM, entrenados con conjuntos balanceados mediante BUE, el metaclasificador de StackOpt se entrena directamente sobre el conjunto completo de




(a) Conjunto de datos `Seguros`.




(b) Conjunto de datos `GenObIA`.

Figura 5.6: Curvas PRC de los enfoques `Performance Curve Mapping` (`PCM`) y `CA` para los diferentes folds.

entrenamiento. Esto puede generar un sesgo hacia la clase mayoritaria, limitando su capacidad para identificar ejemplos relevantes de la clase minoritaria. En el conjunto de datos `GenObIA`, donde el desbalanceo es menor, las diferencias son menos acusadas, pero `PCM` mantiene una ventaja consistente en todos los folds.

De forma similar, la tabla 5.12 presenta los resultados de AUC-PRC obtenidos mediante

| | StackOpt | | | | PCM | CA |
|---|---|---|---|---|---|---|
| Fold | XGB | RF | LR | MLP | | |
| 0 | 0.802694 | 0.800205 | 0.758663 | 0.753280 | **0.8627** | 0.8106 |
| 1 | 0.801298 | 0.798771 | 0.764641 | 0.761375 | **0.8596** | 0.7938 |
| 2 | 0.853241 | 0.855103 | 0.861480 | 0.861699 | **0.9008** | 0.8464 |
| 3 | 0.757374 | 0.778399 | 0.779131 | 0.774676 | **0.8488** | 0.7810 |
| 4 | 0.742054 | 0.738558 | 0.732922 | 0.727746 | **0.8230** | 0.7502 |
| 5 | 0.823574 | 0.840452 | 0.799295 | 0.795733 | **0.8785** | 0.8351 |
| 6 | 0.665988 | 0.666521 | 0.674410 | 0.679787 | **0.7468** | 0.6860 |
| 7 | 0.775054 | 0.766710 | 0.741613 | 0.746161 | **0.8162** | 0.7651 |
| 8 | 0.726061 | 0.717442 | 0.681212 | 0.685540 | **0.7782** | 0.7225 |
| 9 | 0.777469 | 0.778175 | 0.775449 | 0.774584 | **0.8412** | 0.7903 |
| media | 0.772481 | 0.774034 | 0.756882 | 0.756058 | **0.8356** | 0.7781 |
| std. Dev. | 0.053289 | 0.056165 | 0.054749 | 0.052862 | 0.0463 | 0.0496 |

Tabla 5.11: Resultados AUC-PRC obtenidos por StackOpt, PCM y CA en el conjunto de datos GenObIA. En **negrita** se resaltan los mejores resultados

| | Seguros | | | GenObIA | | |
|---|---|---|---|---|---|---|
| Fold | WOpt | PCM | CA | WOpt | PCM | CA |
| 0 | 0.4298 | **0.7406** | 0.5701 | 0.7791 | **0.8627** | 0.8106 |
| 1 | 0.3770 | **0.6991** | 0.5210 | 0.7817 | **0.8596** | 0.7938 |
| 2 | 0.4609 | **0.7622** | 0.5600 | 0.8680 | **0.9008** | 0.8464 |
| 3 | 0.3077 | **0.5899** | 0.4655 | 0.7920 | **0.8488** | 0.7810 |
| 4 | 0.4689 | **0.7039** | 0.5219 | 0.7416 | **0.8230** | 0.7502 |
| 5 | 0.4124 | **0.7112** | 0.5646 | 0.8112 | **0.8785** | 0.8351 |
| 6 | 0.4534 | **0.7122** | 0.5191 | 0.6854 | **0.7468** | 0.6860 |
| 7 | 0.3177 | **0.5965** | 0.4789 | 0.7529 | **0.8162** | 0.7651 |
| 8 | 0.2730 | **0.5470** | 0.4220 | 0.6918 | **0.7782** | 0.7225 |
| 9 | 0.2075 | **0.4782** | 0.4138 | 0.7838 | **0.8412** | 0.7903 |
| media | 0.3708 | **0.6541** | 0.5037 | 0.7687 | **0.8356** | 0.7781 |
| std. Dev. | 0.0853 | 0.0943 | 0.0567 | 0.0543 | 0.0463 | 0.0496 |

Tabla 5.12: Resultados AUC-PRC obtenidos por los métodos WOpt, PCM y CA para los conjuntos de datos Seguros y GenObIA. En **negrita** se resaltan los mejores resultados

el método WOpt. En el conjunto de datos Seguros, WOpt alcanza un valor medio de 0.3708, que sigue siendo inferior al valor de CA (0.5037) y notablemente menor que el de PCM (0.6541). En el conjunto de datos GenObIA, se observa una mejora respecto a CA (0.7687 frente al 0.7781), aunque insuficiente para alcanzar el rendimiento de PCM, que logra un valor medio de 0.8356.

Estos resultados refuerzan la evidencia de que la mejora en el rendimiento proviene, principalmente, de la estrategia de optimización del espacio de decisión que implementa PCM. Ni la combinación lineal mediante StackOpt ni la reponderación de modelos base con WOpt

logran explotar con igual eficacia el modelo *ensemble* en estos conjuntos de datos.

## 5.6 Tiempos de ejecución

Todas las ejecuciones se llevaron a cabo en una CPU Intel(R) Core(TM) i7-10870H de 2.20 GHz con 8 núcleos. El tiempo de ejecución medio de `Gmean` es de 4596 s, mientras que el de `PCM` es de 24 184 s. Es decir, `PCM` requiere aproximadamente 5.26× el tiempo de optimización basado en `Gmean`. Los tiempos de `PCM` son superiores debido al amplio espacio de búsqueda y al uso del algoritmo de optimización multiobjetivo NSGA-II.

Si bien es cierto que el enfoque `PCM` presenta un mayor coste computacional durante la fase de optimización, es importante destacar que este coste se concentra únicamente en el proceso posterior al entrenamiento de los modelos. La solución es prácticamente instantánea en un entorno de producción, ya que la optimización no se realiza en tiempo real, sino que se lleva a cabo como parte de la calibración del modelo tras el entrenamiento inicial.

En aplicaciones prácticas, como en la industria de seguros, los modelos no necesitan ser actualizados u optimizados diariamente, ya que los datos suelen ser relativamente estables a corto plazo. La recalibración del modelo, incluida la optimización mediante `PCM`, se puede programar para realizarse en intervalos definidos, como cada varios meses, dependiendo de la naturaleza del negocio y la variabilidad de los datos. Por ejemplo, en el sector de seguros, el entorno operativo y los perfiles de riesgo no cambian lo suficiente como para requerir ajustes frecuentes, lo que significa que el proceso de optimización puede planificarse y ejecutarse esporádicamente sin incurrir en costes computacionales significativos.

## 5.7 Conclusiones

En este capítulo se ha abordado la optimización de modelos *ensemble* en problemas de clasificación binaria. Tradicionalmente, la configuración estándar establece un umbral de decisión de 0.5 para todos los modelos base del *ensemble*, lo que restringe el análisis a una única variable, el umbral del esquema de decisión, dentro del *espacio de Rendimiento*. Para superar esta limitación, se ha desarrollado `PCM`, un enfoque que redefine la *curva de Rendimiento* como el frente de Pareto de un problema de optimización multiobjetivo.

Se ha evaluado `PCM` en dos problemas de clasificación binaria: (i) predicción de siniestros en seguros de automóviles con un conjunto de datos altamente desbalanceado (conjunto de datos `Seguros`), y (ii) predicción del riesgo de obesidad en un conjunto de datos clínico más equilibrado (conjunto de datos `GenObIA`). La propuesta se ha comparado con cinco métodos alternativos: la configuración clásica con umbral 0.5 (`CA`), un algoritmo genético diseñado para maximizar la media geométrica del TPR y TNR (`Gmean`), el aprendizaje por apilamiento con metaclasificadores (`StackOpt`), una optimización de pesos mediante

Optuna (`WOpt`) y la optimización individual de hiperparámetros para cada modelo base utilizando Optuna (`HiperOpt`).

Los resultados obtenidos demuestran la efectividad de `PCM`, particularmente en el conjunto de datos `Seguros`, donde se registraron mejoras sustanciales. En el caso del conjunto de datos `GenObIA`, `PCM` también mostró un rendimiento competitivo, mostrando adaptabilidad ante escenarios con menor volumen de datos o mayor complejidad. En concreto, `PCM` logró una mejora del 29.73 % en AUC, 25.94 % en AUC-PRC y 11.88 % en el Índice de Youden frente a `CA`, dentro del rango de mejora alcanzable. En el conjunto de datos `Seguros`, las mejoras fueron aún más notables: 46.39 % en AUC, 30.30 % en AUC-PRC y 26.07 % en el Índice de Youden.

Estos resultados contrastan con los resultados AUC y AUC-PRC obtenidos por los métodos `StackOpt` y `WOpt`, los cuales no lograron superar el rendimiento alcanzado por `PCM` en ninguna de las curvas de *Rendimiento* analizadas. En cambio, la optimización de hiperparámetros mediante Optuna sí permitió mejorar los resultados de `PCM` y `CA` en términos AUC. Para el conjunto de datos `Seguros`, la combinación de `HiperOpt`+`PCM` alcanzó una mejora (dentro del rango de mejora alcanzable) del 12.73 % respecto a `PCM` mientras que en el conjunto `GenObIA` la mejora fue aún mayor, alcanzando un 19.19 %.

Comparando la combinación de `HiperOpt`+`CA` contra `CA`, las mejoras fueron considerablemente menores: un 5.92 % en `Seguros` y un 11.01 % en `GenObIA`. Estos resultados indican que, si bien la optimización de hiperparámetros contribuye al rendimiento global del modelo, el grueso de la mejora se debe a `PCM`. La combinación de ambos enfoques, `HiperOpt`+`PCM`, se presenta como la mejor opción para maximizar el rendimiento del modelo *ensemble*.

En conjunto, los hallazgos de este capítulo sugieren que `PCM` representa una estrategia más eficaz para optimizar modelos *ensemble* que el análisis tradicional basado en curvas ROC o PRC. Además, introduce un nuevo paradigma para la evaluación y calibración de modelos *ensemble*, basado en la exploración del espacio de decisión. Nuestra propuesta permite identificar configuraciones óptimas de umbrales que maximizan el rendimiento en múltiples métricas.

Un aspecto relevante de `PCM` es su independencia respecto al modelo base, lo que facilita su aplicación en distintos esquemas de modelos *ensemble*. Además, ha mostrado un comportamiento robusto tanto en conjuntos de datos desbalanceados como en escenarios más equilibrados, siendo una herramienta especialmente útil en dominios como la medicina, las finanzas o el sector asegurador, donde la precisión y la capacidad de decisión son fundamentales.

Aunque la aplicación de `PCM` requiere un mayor coste computacional durante la fase de optimización, esta se limita exclusivamente al proceso posterior al entrenamiento. En entornos de producción, el uso del modelo *ensemble* optimizado con `PCM` no presenta una penalización adicional, ya que la optimización se realiza de forma offline como parte del

proceso de calibración.

En dominios como el sector asegurador, donde los datos de pólizas se reciben continuamente para la tarificación y adquisición de clientes, esta estructura permite un equilibrio entre eficiencia computacional y rendimiento del modelo. El flujo continuo de datos se emplea de inmediato para predicciones y solo se incorporan al reentrenamiento cuando se dispone de suficiente información etiquetada.

La frecuencia óptima de recalibración de los modelos con `PCM` depende de la variabilidad y distribución de los datos entrantes. Se pueden considerar distintas estrategias de recalibración según las necesidades del negocio y la disponibilidad de datos:

- Recalibración diaria. Un proceso en segundo plano podría ajustar el modelo cada noche utilizando los datos de siniestros y gastos recientemente disponibles. Esto es útil en entornos altamente dinámicos, aunque puede introducir una mayor carga computacional.
- Recalibración semanal. Actualizar el modelo al final de cada semana equilibra adaptabilidad y eficiencia al incorporar una cantidad más estable de información nueva.
- Recalibración anual. Ésta permitiría capturar de manera más precisa los cambios graduales que se producen en los datos a lo largo del tiempo, por ejemplo, en el sector asegurador los costes o recobros anuales asociados a las pólizas tienden a materializarse con el tiempo.

Finalmente, aunque los resultados obtenidos validan la eficacia de `PCM` en los casos analizados, su extensión a otros dominios como el procesamiento de lenguaje natural o la visión por computador constituye una línea prometedora de investigación futura.

# Capítulo 6

# Clasificador multinivel

[CONTENIDO CENSURADO]

**Este capítulo ha sido omitido en la versión pública del documento por contener información sujeta a acuerdos de confidencialidad.**

# Capítulo 7

# Conclusiones

A lo largo de esta tesis se han desarrollado distintas propuestas metodológicas con el objetivo de mejorar la capacidad predictiva de los modelos aplicados al sector asegurador, tanto desde la perspectiva de la siniestralidad como de la rentabilidad. El trabajo ha ido evolucionando desde enfoques más tradicionales hasta configuraciones más sofisticadas, basadas en técnicas de *ensemble* y calibración personalizada, respondiendo a retos reales de negocio y datos altamente desbalanceados.

El trabajo comenzó con la propuesta del método *Balanced Underbagged Ensemble* (BUE), diseñado para mejorar la representatividad de las clases minoritarias sin comprometer la estabilidad del modelo. BUE forma un ensemble de clasificadores entrenados con subconjuntos balanceados,lo que garantiza que todas las regiones del espacio de entrada estén bien representadas y, al mismo tiempo, se reduzca el sesgo hacia la clase mayoritaria.

Los resultados ponen de manifiesto la efectividad del método BUE, diseñado específicamente para combinar de forma equilibrada el submuestreo y el *bagging* en escenarios con un alto desbalanceo entre clases. A diferencia de otras técnicas, BUE maximiza la detección de la clase minoritaria sin comprometer la robustez del modelo, reforzando la diversidad entre clasificadores base y reduciendo la pérdida de información de la clase mayoritaria. Por ello, BUE se presenta como una alternativa sólida frente a métodos clásicos de muestreo, ofreciendo resultados más estables y precisos en contextos aseguradores donde el desbalanceo de datos es especialmente crítico.

En la segunda fase se desarrolló un clasificador en cascada que permite dividir la decisión en niveles jerárquicos. Esta arquitectura está especialmente orientada a introducir mecanismos de abstención razonada en aquellas instancias donde el modelo detecta un alto grado de incertidumbre. El impacto práctico es notable ya que se mejora la calidad de clasificación global al reducir los errores en los niveles superiores y, además, permite ajustar el equilibrio entre las decisiones clasificadas y las rechazadas según el contexto.

En el marco del proyecto `GenObIA`, esta arquitectura fue evaluada frente a nueve algoritmos clásicos, mostrando un rendimiento superior tanto en precisión como en *recall* para la

clase positiva. Además, se definieron métricas específicas como $Q_{\text{class}}$ y $Q_{\text{rej}}$, que permiten evaluar tanto la capacidad predictiva como la precisión de la abstención.

En una tercera etapa se abordó uno de los aspectos menos explorados en la literatura, la optimización del espacio de decisión en modelos *ensemble*. Para ello se propuso `Performance Curve Mapping`, una metodología que plantea el análisis clásico de curvas ROC y PRC como un problema multiobjetivo de optimización.

A lo largo de este trabajo se ha demostrado que la configuración tradicional de umbrales fijos en modelos *ensemble* limita el aprovechamiento de todo su potencial. Para superar esta restricción, `PCM` realiza una exploración sobre el espacio de decisión, permitiendo identificar configuraciones más ajustadas a las características de cada escenario. Los resultados obtenidos validan la eficacia de `PCM` para mejorar métricas clave como AUC y AUC-PRC, especialmente en contextos altamente desbalanceados, y confirman su robustez en diferentes dominios. Una aportación destacable es su independencia respecto al modelo base, lo que facilita su aplicación en distintos esquemas *ensemble* y sectores donde la precisión y la capacidad de decisión resultan críticas, como el asegurador, el financiero o el clínico. Además, su implementación práctica se adapta a entornos con flujo continuo de datos, ya que la fase de optimización se realiza de forma offline sin afectar al rendimiento operativo del sistema. Finalmente, se abre una línea prometedora para extender este enfoque a otros campos, como el procesamiento de lenguaje natural o la visión por computador, ampliando así su alcance y contribución al desarrollo de modelos predictivos más precisos y fiables.

Como aportación final[1], se planteó una arquitectura multinivel que integra las fortalezas de los métodos anteriores mediante el uso de BUE en cada nivel y la adaptación del comportamiento del modelo según distintos objetivos de negocio. Esta arquitectura se orienta a optimizar la rentabilidad y gestionar el riesgo en carteras aseguradoras mediante dos indicadores clave: el CRI y el PUI. Ambos KPIs se implementaron considerando variantes que responden a distintos objetivos estratégicos, como la reducción de la frecuencia siniestral, la maximización del resultado técnico o un equilibrio entre ambos. Los resultados obtenidos muestran que el clasificador multinivel mejora la capacidad de identificar pólizas de alto riesgo manteniendo bajo control el número de pólizas rechazadas, lo que contribuye a una gestión más eficaz y rentable de la cartera.

A lo largo de esta tesis se ha demostrado que es posible construir un marco predictivo sofisticado y flexible, que supere a las técnicas tradicionales tanto en rendimiento como en adaptabilidad. Las propuestas presentadas son complementarias entre sí y pueden aplicarse de manera modular o conjunta en función del escenario. Además, todos los desarrollos han sido validados con datos reales, lo que otorga a esta investigación un valor práctico tangible.

La capacidad de adaptar el modelo a distintas metas (control de siniestralidad, optimiza-

[1]Las aportaciones detalladas en este párrafo son presentadas en el capítulo 6, que está sujeto a un acuerdo de confidencialidad

ción del resultado técnico o equilibrio) refuerza su aplicabilidad en la industria aseguradora, donde los objetivos pueden variar entre carteras o contextos. Además, se ha trabajado para mantener la estabilidad entre ejecuciones, minimizar errores críticos e incorporar mecanismos de rechazo, todo ello para que las decisiones sean más seguras, eficientes y fáciles de explicar.

En definitiva, esta tesis no solo ha contribuido al avance teórico en modelado predictivo sobre datos desbalanceados, sino que ha construido un marco práctico, adaptable e interpretativo que responde a las necesidades reales del sector asegurador. Las técnicas desarrolladas pueden servir como base para nuevos sistemas de recomendación de pólizas, evaluación de riesgo o diseño de productos aseguradores personalizados.

# Bibliografía


[1] Ahmed Abbasi and Hsinchun Chen. A comparison of fraud cues and classification methods for fake escrow website detection. *Information Technology and Management*, 10(2):83–101, 2009. doi: 10.1007/s10799-009-0059-0.

[2] Chirath Abeysinghe, Jianguo Li, and Jing He. A classifier hub for imbalanced financial data. In *Australasian Database Conference*, pages 476–479. Springer, 2016. doi: 10.1007/978-3-319-46922-5_43.

[3] AlsharifHasan Mohamad Aburbeian and Huthaifa I Ashqar. Credit card fraud detection using enhanced random forest classifier for imbalanced data. In *International conference on advances in computing research*, pages 605–616. Springer, 2023.

[4] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[5] Thomas D Albright. How to make better forensic decisions. *Proceedings of the National Academy of Sciences*, 119(38):e2206567119, 2022.

[6] Saman Atapattu, Chintha Tellambura, and Hai Jiang. Analysis of area under the roc curve of energy detection. *IEEE Transactions on Wireless Communications*, 9 (3):1216–1225, 2010. doi: 10.1109/TWC.2010.03.091085.

[7] Saman Atapattu, Chintha Tellambura, and Hai Jiang. Analysis of area under the roc curve of energy detection. *IEEE Transactions on wireless communications*, 9 (3):1216–1225, 2010.

[8] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[9] Amos Azaria, Ariella Richardson, Sarit Kraus, and VS Subrahmanian. Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data. *IEEE Transactions on Computational Social Systems*, 1(2):135–155, 2014. doi: 10.1109/TCSS.2014.2377811.

[10] Feng Bao, Yue Deng, and Qionghai Dai. Acid: Association correction for imbalanced

data in gwas. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(1):316–322, 2016. doi: 10.1109/TCBB.2016.2608819.

[11] Gustavo EAPA Batista, Ana LC Bazzan, Maria Carolina Monard, et al. Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18, 2003. doi: 10.1145/3205651.3205658.

[12] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[13] Punam Bedi, Neha Gupta, and Vinita Jindal. I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems. *Applied Intelligence*, 51(2):1133–1151, September 2020. ISSN 1573-7497. doi: 10.1007/s10489-020-01886-y. URL http://dx.doi.org/10.1007/s10489-020-01886-y.

[14] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, 3(10), 2013.

[15] Zied Ben-Bouallegue. Seamless prediction of high-impact weather events: a comparison of actionable forecasts. *arXiv preprint arXiv:2312.01673*, 2023.

[16] Urvesh Bhowan, Mengjie Zhang, and Mark Johnston. Auc analysis of the pareto-front using multi-objective gp for classification with unbalanced data. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 845–852, 2010.

[17] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[18] Christopher Blier-Wong, Hélène Cossette, Luc Lamontagne, and Etienne Marceau. Machine learning in p and c insurance: A review for pricing and reserving. *Risks*, 9(1), 2021. ISSN 2227-9091. doi: 10.3390/risks9010004. URL https://www.mdpi.com/2227-9091/9/1/4.

[19] Leo Breiman. Bagging predictors. 421:–20, 09 1994. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

[20] Borja Calvo and Guzmán Santafé Rodrigo. scmamp: Statistical comparison of multiple algorithms in multiple problems. *The R Journal, Vol. 8/1, Aug. 2016*, 2016.

[21] Borja Calvo, Josu Ceberio, and Jose A Lozano. Bayesian inference for algorithm ranking analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 324–325, 2018. doi: 10.1145/3205651.3205658.

[22] Francesco Cartella, Orlando Anunciacao, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. Adversarial attacks for tabular data: Application to fraud detection and imbalanced data. *arXiv preprint arXiv:2101.08030*, 2021.

[23] Chein-I Chang, Shao-Shan Chiang, Qian Du, Hsuan Ren, and A Ifarragaerri. An roc analysis for subpixel detection. In *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)*, volume 5, pages 2355–2357. IEEE, 2001.

[24] Clément Chatelain, Sébastien Adam, Yves Lecourtier, Laurent Heutte, and Thierry Paquet. A multi-model selection framework for unknown and/or evolutive misclassification cost problems. *Pattern Recognition*, 43(3):815–823, 2010.

[25] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. doi: 10.1613/jair.953.

[26] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*, pages 107–119. Springer, 2003.

[27] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785.

[28] Yi Chen and Fulu Tao. Potential of remote sensing data-crop model assimilation and seasonal weather forecasts for early-season crop yield forecasting over a large area. *Field Crops Research*, 276:108398, 2022. ISSN 0378-4290. doi: https://doi.org/10.1016/j.fcr.2021.108398. URL `https://www.sciencedirect.com/science/article/pii/S0378429021003440`.

[29] Filipe Condessa, José Bioucas-Dias, and Jelena Kovačević. Performance measures for classification systems with rejection. *Pattern Recognition*, 63:437–450, 2017.

[30] Consorcio Genobia. Proyecto genobia: Estrategias predictivas para la prevención del sobrepeso y obesidad en población adulta. `https://www.proyectogenobia.org`, 2020. Accedido el 26 de marzo de 2025.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002. doi: 10.1109/4235.996017.

[32] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[33] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[34] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical

tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[35] Sharmishta Desai, Sourav Roy, Brina Patel, Samruddhi Purandare, and Minal Kucheria. Very fast decision tree (vfdt) algorithm on hadoop. In *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, pages 1–7. IEEE, 2016. doi: 10.1109/ICCUBEA.2016.7860037.

[36] Najmeddine Dhieb, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. A secure ai-driven architecture for automated insurance systems: Fraud detection and risk measurement. *IEEE Access*, 8:58546–58558, 2020. doi: 10.1109/ACCESS.2020.2983300.

[37] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003.

[38] Liang Du, Ruobin Gao, Ponnuthurai Nagaratnam Suganthan, and David ZW Wang. Bayesian optimization based dynamic ensemble for time series forecasting. *Information Sciences*, 591:155–175, 2022.

[39] Rashmi Dubey, Jiayu Zhou, Yalin Wang, Paul M Thompson, Jieping Ye, Alzheimer's Disease Neuroimaging Initiative, et al. Analysis of sampling techniques for imbalanced data: An n= 648 adni study. *NeuroImage*, 87:220–241, 2014. doi: 10.1016/j.neuroimage.2013.10.005.

[40] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.10.010. URL https://www.sciencedirect.com/science/article/pii/S016786550500303X. ROC Analysis in Pattern Recognition.

[41] Peter A Flach. The geometry of roc space: understanding machine learning metrics through roc isometrics. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 194–201, 2003.

[42] Ronen Fluss, David Faraggi, and Benjamin Reiser. Estimation of the youden index and its associated cutoff point. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 47(4):458–472, 2005.

[43] Ronen Fluss, David Faraggi, and Benjamin Reiser. Estimation of the youden index and its associated cutoff point. *Biometrical Journal*, 47(4):458–472, 2005. doi: https://doi.org/10.1002/bimj.200410135. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bimj.200410135.

[44] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-,

boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[45] Sheng Gao, Chin-Hui Lee, and Joo Hwee Lim. An ensemble classifier learning approach to roc optimization. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 679–682, 2006. doi: 10.1109/ICPR.2006.246.

[46] Soutik Ghosal and Zhen Chen. Discriminatory capacity of prenatal ultrasound measures for large-for-gestational-age birth: A bayesian approach to roc analysis using placement values. *Statistics in biosciences*, 14(1):1–22, 2022.

[47] Filippos Giannakas, Christos Troussas, Akrivi Krouska, Cleo Sgouropoulou, and Ioannis Voyiatzis. Xgboost and deep neural network comparison: The case of teams performance. In *Intelligent Tutoring Systems: 17th International Conference, ITS 2021, Virtual Event, June 7–11, 2021, Proceedings 17*, pages 343–349. Springer, 2021.

[48] Fares Grina, Zied Elouedi, and Eric Lefevre. Re-sampling of multi-class imbalanced data using belief function theory and ensemble learning. *International Journal of Approximate Reasoning*, 156:1–15, 2023.

[49] Leo Guelman. Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Systems with Applications*, 39(3):3659–3667, 2012. doi: 10.1016/j.eswa.2011.09.058.

[50] Thien M Ha and Horst Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997. doi: 10.1109/34.589216.

[51] Guo Haixiang, Li Yijing, Li Yanan, Liu Xiao, and Li Jinling. Bpso-adaboost-knn ensemble learning algorithm for multi-class imbalanced data classification. *Engineering Applications of Artificial Intelligence*, 49:176–193, 2016. doi: 10.1016/j.engappai.2015.09.011.

[52] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017. ISSN 0957-4174. doi: 10.1016/j.eswa.2016.12.035.

[53] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing*, pages 878–887, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31902-3. doi: 10.1007/11538059_91.

[54] Mohamed Hanafy and Ruixing Ming. Machine learning approaches for auto insurance big data. *Risks*, 9(2), 2021. ISSN 2227-9091. doi: 10.3390/risks9020042. URL https://www.mdpi.com/2227-9091/9/2/42.

[55] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

[56] Yumnah Hasan, Fatemeh Amerehi, Patrick Healy, and Conor Ryan. Stem rebalance: A novel approach for tackling imbalanced datasets using smote, edited nearest neighbour, and mixup. In *2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 3–9, 2023. doi: 10.1109/ICCP60212.2023.10398660.

[57] Amira Kamil Ibrahim Hassan and Ajith Abraham. Modeling insurance fraud detection using imbalanced data classification. In *Advances in Nature and Biologically Inspired Computing: Proceedings of the 7th World Congress on Nature and Biologically Inspired Computing (NaBIC2015) in Pietermaritzburg, South Africa, held December 01-03, 2015*, pages 117–127. Springer, 2015.

[58] Amira Kamil Ibrahim Hassan and Ajith Abraham. Modeling insurance fraud detection using imbalanced data classification. In *Advances in nature and biologically inspired computing*, pages 117–127. Springer, 2016. doi: 10.1007/978-3-319-27400-3_11.

[59] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Boosting and additive trees. *The elements of statistical learning: data mining, inference, and prediction*, pages 337–387, 2009.

[60] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008. doi: 10.1109/IJCNN.2008.4633969.

[61] Hongliang He, Wenyu Zhang, and Shuai Zhang. A novel ensemble method for credit scoring: Adaption of different imbalance ratios. *Expert Systems with Applications*, 98:105–117, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.01.012. URL `https://www.sciencedirect.com/science/article/pii/S0957417418300125`.

[62] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13 (4):18–28, 1998.

[63] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: A survey. *Machine Learning*, 113(5): 3073–3110, 2024.

[64] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[65] Ariel Jaffe, Ethan Fetaya, Boaz Nadler, Tingting Jiang, and Yuval Kluger. Unsupervised ensemble learning with dependent classifiers. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2016.

[66] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on artificial intelligence*, volume 56, pages 111–117, 2000.

[67] Piyasak Jeatrakul, Kok Wai Wong, and Chun Che Fung. Classification of imbalanced data by combining the complementary neural network and smote algorithm. In *Neural Information Processing. Models and Applications: 17th International Conference, ICONIP 2010, Sydney, Australia, November 22-25, 2010, Proceedings, Part II 17*, pages 152–159. Springer, 2010.

[68] Alan Jeffares, Tennison Liu, Jonathan Crabbé, and Mihaela van der Schaar. Joint training of deep ensembles fails due to learner collusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[69] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[70] Bartosz Krawczyk, Mikel Galar, Łukasz Jeleń, and Francisco Herrera. Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Applied Soft Computing*, 38:714–726, 2016. doi: 10.1016/j.asoc.2015.08.060.

[71] Wojciech Ksiazek, Mohamed Hammad, Paweł Pławiak, U. Rajendra Acharya, and Ryszard Tadeusiewicz. Development of novel ensemble model using stacking learning and evolutionary computation techniques for automated hepatocellular carcinoma detection. *Biocybernetics and Biomedical Engineering*, 40(4):1512–1524, 2020. ISSN 0208-5216. doi: https://doi.org/10.1016/j.bbe.2020.08.007. URL https://www.sciencedirect.com/science/article/pii/S0208521620300991.

[72] Sumit Kumar, Gaurav Sundaram, Snehan Shourya, Rajib Kumar Jha, Rashmi Ranjan Maharana, and Garima Saini. A novel signal detector based on approximated fractional integrator in frequency domain. In *2023 International Conference on Fractional Differentiation and Its Applications (ICFDA)*, pages 1–5, 2023. doi: 10.1109/ICFDA58234.2023.10153303.

[73] Chap T Le. A solution for the most basic optimization problem associated with an roc curve. *Statistical Methods in Medical Research*, 15(6):571–584, 2006. doi: 10.1177/0962280206070637. URL https://doi.org/10.1177/0962280206070637. PMID: 17260924.

[74] Julien-Charles Lévesque, Audrey Durand, Christian Gagné, and Robert Sabourin. Multi-objective evolutionary optimization for generating ensembles of classifiers in the roc space. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 879–886, 2012.

[75] Yang Li, Jiting Cao, Yan Xu, Lipeng Zhu, and Zhao Yang Dong. Deep learning based on transformer architecture for power system short-term voltage stability assessment with class imbalance. *Renewable and Sustainable Energy Reviews*, 189: 113913, 2024.

[76] Wanan Liu, Hong Fan, and Meng Xia. Tree-based heterogeneous cascade ensemble model for credit scoring. *International Journal of Forecasting*, 39(4):1593–1614, 2023. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2022.07.007. URL https://www.sciencedirect.com/science/article/pii/S0169207022001054.

[77] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[78] Yue Liu, Bing-Jie Wang, and Shao-Gao Lv. Using multi-class adaboost tree for prediction frequency of auto insurance. *Journal of Applied Finance and Banking*, 4 (5):45, 2014.

[79] Zhining Liu, Wei Cao, Zhifeng Gao, Jiang Bian, Hechang Chen, Yi Chang, and Tie-Yan Liu. Self-paced ensemble for highly imbalanced massive data classification. In *2020 IEEE 36th international conference on data engineering (ICDE)*, pages 841–852. IEEE, 2020.

[80] Lee B Lusted. Logical analysis in roentgen diagnosis: memorial fund lecture. *Radiology*, 74(2):178–193, 1960.

[81] Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, pages 1–7. ICML, 2003.

[82] J. Marcum. A statistical theory of target detection by pulsed radar. *IRE Transactions on Information Theory*, 6(2):59–267, 1960. doi: 10.1109/TIT.1960.1057560.

[83] Faseeha Matloob, Taher M Ghazal, Nasser Taleb, Shabib Aftab, Munir Ahmad, Muhammad Adnan Khan, Sagheer Abbas, and Tariq Rahim Soomro. Software defect prediction using ensemble learning: A systematic literature review. *IEEe Access*, 9:98754–98771, 2021.

[84] Peter McCullagh and John A Nelder. *Generalized linear models*. Routledge, 2019. doi: 10.1201/9780203753736.

[85] Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with imbalanced data. *Data mining and knowledge discovery*, 28:92–122, 2014.

[86] Charles E Metz. Roc methodology in radiologic imaging. *Investigative radiology*, 21 (9):720–733, 1986.

[87] Ibomoiye Domor Mienye and Nobert Jere. Optimized ensemble learning approach with explainable ai for improved heart disease prediction. *Information*, 15(7), 2024. ISSN 2078-2489. doi: 10.3390/info15070394. URL https://www.mdpi.com/2078-2489/15/7/394.

[88] Ibomoiye Domor Mienye and Yanxia Sun. A survey of ensemble learning: Concepts,

algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149, 2022. doi: 10.1109/ACCESS.2022.3207287.

[89] Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[90] Michael C Mozer, Robert Dodier, Michael Colagrosso, Cesar Guerra-Salcedo, and Richard Wolniewicz. Prodding the roc curve: Constrained optimization of classifier performance. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/2cd4e8a2ce081c3d7c32c3cde4312ef7-Paper.pdf.

[91] Mimi Mukherjee and Matloob Khushi. Smote-enc: A novel smote-based method to generate synthetic data for nominal and continuous features. *Applied System Innovation*, 4(1), 2021. ISSN 2571-5577. doi: 10.3390/asi4010018. URL https://www.mdpi.com/2571-5577/4/1/18.

[92] Francis Sahngun Nahm. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology*, 75(1):25–36, 2022.

[93] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[94] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 135(3):370–384, 1972.

[95] Bohdan Pavlyshenko. Using stacking approaches for machine learning models. In *2018 IEEE second international conference on data stream mining & processing (DSMP)*, pages 255–258. IEEE, 2018.

[96] Margaret Sullivan Pepe. An interpretation for the roc curve and inference using glm procedures. *Biometrics*, 56(2):352–359, 2000.

[97] Tadeusz Pietraszek. On the use of roc analysis for the optimization of abstaining classifiers. *Machine learning*, 68(2):137–169, 2007. doi: 10.1007/s10994-007-5013-y.

[98] Andrea Dal Pozzolo. Credit card fraud detection dataset, 2015. URL https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. Accessed: October, 2024.

[99] Foster Provost and Tom Fawcett. Data science and its relationship to big data and data-driven decision making. *Big data*, 1(1):51–59, 2013.

[100] Oona Rainio, Jarmo Teuho, and Riku Klén. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14(1):6086, 2024.

[101] Santhanam Ramraj, Nishant Uzir, R Sunil, and Shatadeep Banerjee. Experimenting xgboost algorithm for prediction and classification of different datasets. *International Journal of Control Theory and Applications*, 9(40):651–662, 2016.

[102] Sahar Saeed Rezk and Kamal Samy Selim. Metaheuristic-based ensemble learning: an extensive review of methods and applications. *Neural Computing and Applications*, pages 1–29, 2024.

[103] Lior Rokach and Oded Maimon. Decision trees. *Data mining and knowledge discovery handbook*, pages 165–192, 2005.

[104] Andrew Ross, Weiwei Pan, Leo Celi, and Finale Doshi-Velez. Ensembles of locally independent prediction models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5527–5536, 2020.

[105] Riya Roy and K. Thomas George. Detecting insurance claims fraud using machine learning techniques. In *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*, pages 1–6, 2017. doi: 10.1109/ICCPCT.2017.8074258.

[106] Rahul Sahai, Ali Al-Ataby, Sulaf Assi, Manoj Jayabalan, Panagiotis Liatsis, Chong Kim Loy, Abdullah Al-Hamid, Sahar Al-Sudani, Maitham Alamran, and Hoshang Kolivand. Insurance risk prediction using machine learning. In Yap Bee Wah, Michael W. Berry, Azlinah Mohamed, and Dhiya Al-Jumeily, editors, *Data Science and Emerging Technologies*, pages 419–433, Singapore, 2023. Springer Nature Singapore. ISBN 978-981-99-0741-0.

[107] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015. doi: 10.1371/journal.pone.0118432.

[108] F. Schoefs, A. Clément, and A. Nouy. Assessment of roc curves for inspection of random fields. *Structural Safety*, 31(5):409–419, 2009. ISSN 0167-4730. doi: https://doi.org/10.1016/j.strusafe.2009.01.004. URL `https://www.sciencedirect.com/science/article/pii/S0167473009000034`.

[109] Nicholas Scurich and Richard S. John. Three-way rocs for forensic decision making. *Statistics and Public Policy*, 10(1):2239306, 2023. doi: 10.1080/2330443X.2023.2239306.

[110] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 40(1):185–197, 2009.

[111] Mohsen Shahhosseini, Guiping Hu, and Hieu Pham. Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Machine Learning with Applications*, 7:100251, 2022.

[112] Andrew M. Smith and Tess M.S. Neal. The distinction between discriminability and reliability in forensic science. *Science Justice*, 61(4):319–331, 2021. ISSN 1355-0306. doi: https://doi.org/10.1016/j.scijus.2021.04.002. URL `https://www.sciencedirect.com/science/article/pii/S1355030621000435`.

[113] Artem Sokolov and Darrell Whitley. Unbiased tournament selection. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1131–1138, 2005.

[114] Paria Soltanzadeh, M Reza Feizi-Derakhshi, and Mahdi Hashemzadeh. Addressing the class-imbalance and class-overlap problems by a metaheuristic-based undersampling approach. *Pattern Recognition*, page 109721, 2023.

[115] John A Swets. Indices of discrimination or diagnostic accuracy: their rocs and implied models. *Psychological bulletin*, 99(1):100, 1986.

[116] Hisashi Tamaki, Hajime Kita, and Shigenobu Kobayashi. Multi-objective optimization by genetic algorithms: A review. In *Proceedings of IEEE international conference on evolutionary computation*, pages 517–522. IEEE, 1996.

[117] I Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(6):448–452, 1976. doi: 10.1109/TSMC.1976.4309523.

[118] Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of shap explanations. *Journal of Artificial Intelligence Research*, 74: 851–886, 2022.

[119] K Venkatachalam, Pavel Trojovskỳ, Dragan Pamucar, Nebojsa Bacanin, and Vladimir Simic. Dwfh: An improved data-driven deep weather forecasting hybrid model using transductive long short term memory (t-lstm). *Expert Systems with Applications*, 213:119270, 2023.

[120] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft computing*, 22:387–408, 2018.

[121] Le Wang, Meng Han, Xiaojuan Li, Ni Zhang, and Haodong Cheng. Review of classification methods on unbalanced data sets. *Ieee Access*, 9:64606–64628, 2021.

[122] Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE symposium on computational intelligence and data mining*, pages 324–331. IEEE, 2009.

[123] Mario V Wüthrich and Michael Merz. *Statistical foundations of actuarial learning and its applications*. Springer Nature, 2023.

[124] Jin Xiao, Yu Zhong, Yanlin Jia, Yadong Wang, Ruoyi Li, Xiaoyi Jiang, and Shouyang Wang. A novel deep ensemble model for imbalanced credit scoring in internet finance. *International Journal of Forecasting*, 40(1):348–372, 2024. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2023.03.004. URL `https://www.sciencedirect.com/science/article/pii/S0169207023000353`.

[125] Xi Xiao, Wentao Xiao, Dianyan Zhang, Bin Zhang, Guangwu Hu, Qing Li, and

Shutao Xia. Phishing websites detection via cnn and multi-head self-attention on imbalanced datasets. *Computers & Security*, 108:102372, 2021.

[126] Wang Xinhua, Yan Qing, Jia Lianqin, and J. A. GKhongwar. Prediction of auto insurance claim probability and cumulative compensation based on machine learning algorithm. In Bernard J. Jansen, Qingyuan Zhou, and Jun Ye, editors, *Proceedings of the 2nd International Conference on Cognitive Based Information Processing and Applications (CIPA 2022)*, pages 697–702, Singapore, 2023. Springer Nature Singapore. ISBN 978-981-19-9376-3.

[127] Biao Xu, Yao Wang, Xiuwu Liao, and Kaidong Wang. Efficient fraud detection using deep boosting decision trees. *Decision Support Systems*, 175:114037, 2023.

[128] Zhaozhao Xu, Derong Shen, Tiezheng Nie, and Yue Kou. A hybrid sampling algorithm combining m-smote and enn based on random forest for medical imbalanced data. *Journal of Biomedical Informatics*, 107:103465, 2020.

[129] Jiangning Yin and Nan Li. Ensemble learning models with a bayesian optimization algorithm for mineral prospectivity mapping. *Ore Geology Reviews*, 145:104916, 2022. ISSN 0169-1368. doi: https://doi.org/10.1016/j.oregeorev.2022.104916. URL `https://www.sciencedirect.com/science/article/pii/S0169136822002244`.

[130] Hualong Yu, Jun Ni, Yuanyuan Dan, and Sen Xu. Mining and integrating reliable decision rules for imbalanced cancer gene expression data sets. *Tsinghua Science and technology*, 17(6):666–673, 2012. doi: 10.1109/TST.2012.6374368.

[131] Jinghui Yuan, Weijin Jiang, Zhe Cao, Fangyuan Xie, Rong Wang, Feiping Nie, and Xuelong Li. Achieving more with less: A tensor-optimization-powered ensemble method. *arXiv preprint arXiv:2408.02936*, 2024.

[132] Guo-Qiang Zeng, Jie Chen, Li-Min Li, Min-Rong Chen, Lie Wu, Yu-Xing Dai, and Chong-Wei Zheng. An improved multi-objective population-based extremal optimization algorithm with polynomial mutation. *Information Sciences*, 330:49–73, 2016. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2015.10.010. URL `https://www.sciencedirect.com/science/article/pii/S0020025515007276`. SI Visual Info Communication.

[133] Chaobo Zhang, Junyang Li, Yang Zhao, Tingting Li, Qi Chen, Xuejun Zhang, and Weikang Qiu. Problem of data imbalance in building energy load prediction: Concept, influence, and solution. *Applied Energy*, 297:117139, 2021.

[134] Guixin Zhang, Zhenlei Wang, and Hua Mei. Sensitivity clustering and roc curve based alarm threshold optimization. *Process Safety and Environmental Protection*, 141:83–94, 2020. ISSN 0957-5820. doi: https://doi.org/10.1016/j.psep.2020.03.029. URL `https://www.sciencedirect.com/science/article/pii/S0957582019320841`.

[135] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mi-

xup: Beyond empirical risk minimization, 2018. URL https://arxiv.org/abs/1710.09412.

[136] Yu Zhong and Huiling Wang. Internet financial credit scoring models based on deep forest and resampling methods. *IEEE Access*, 11:8689–8700, 2023. doi: 10.1109/ACCESS.2023.3239889.

[137] Ying Zhou, Long Shen, and Laura Ballester. A two-stage credit scoring model based on random forest: Evidence from chinese small firms. *International Review of Financial Analysis*, 89:102755, 2023. ISSN 1057-5219. doi: https://doi.org/10.1016/j.irfa.2023.102755. URL https://www.sciencedirect.com/science/article/pii/S1057521923002715.