Universidad
Rey Juan Carlos

Tesis Doctoral

Métodos de muestreo para la mejora de rendimiento en clasificadores de aprendizaje automático

Autor:

Víctor C. Aceña Gil

Directores:

Javier Martínez Moguerza

Isaac Martín de Diego

Programa de Doctorado en Tecnologías de la Información y las Comunicaciones

Escuela Internacional de Doctorado

Mayo de 2023

Todos los días tienen un mejor momento, no te saltes el tuyo.
El Chojin.

Buena suerte o mala suerte, ¿quién sabe?

Agradecimientos

El verdadero logro del doctorado no es esta tesis en sí, sino la persona en la que me he convertido para llegar hasta aquí. Afrontar este desafío, siendo consciente de que no soy excepcional sino simplemente uno más de tantos, ha sido posible gracias a los numerosos apoyos que he tenido durante esta etapa. Sin ellos, no habría sido capaz de llegar a buen puerto.

En primer lugar, quiero expresar mi más profundo agradecimiento a Javier e Isaac, mis directores de tesis y, por supuesto, a Jose y Raúl, por apostar por este proyecto que nos ha unido durante tres años de esfuerzo conjunto. En particular, a Javier, por esas locas ideas que han dado lugar a artículos científicos y han impulsado esta tesis. A Isaac, por enseñarme el significado de la investigación, mostrándome cómo debe ser un investigador, compañero y líder. Aún sigo aprendiendo.

A mis padres, quienes me han brindado la oportunidad de construir la escalera que me ha llevado hasta este último peldaño, agradezco su infinita paciencia a lo largo de estos años. Sin su apoyo incondicional durante mi carrera, esto no habría sido posible. Y, por supuesto, a mi hermana, cuyas conversaciones me han inspirado a mantener vivo el interés por seguir aprendiendo.

A mis amigos, especialmente a Irene, Fernando, Diego, Víctor e Isma, quienes nunca fallan, mis compañeros de vida, con los que contar para todo, los infinitos. A Estíbaliz, Laura y Jose, inspiración constante en el camino del doctorado que solo es posible con trabajo y paciencia. A Carol, Álvaro, Adri y Rocío porque aunque la vida nos ha separado un poco, siempre estáis. A Sergio, mi eterno jefe, quien ya lo sabía incluso antes de que me metiera en este lío. A Sergi y Nadia, que aparecen siempre que se los necesito, aunque yo no lo sepa. A Javi, quien me mostró una puerta que no había visto, que llevaba a un lugar que jamás pensé que podría visitar. Finalmente, a Paula, por irrumpir en mi vida con la naturalidad de quien siempre estuvo, tus revisiones y consejos me han hecho mejor en todos los sentidos.

A todos mis compañeros del DSLAB, que han pasado de ser simples colegas a grandes amigos En especial a Carmen, porque somos iguales y nos hemos dado el apoyo que el otro necesitaba. Por último, a Rubén, mi eterno compañero de batalla,

que tanto me ha ayudado y del que tanto he aprendido. Me enseñó a cambiar de R a Python, de Windows a Linux, a programar sin chapuzas (o al menos lo ha intentado), pero sobre todo, a no parar nunca. Totalmente distintos, pero iguales. Separados al nacer.

Una mención especial a las personas que me han mantenido en este mundo. A las médicas del servicio de oncología del Hospital del Henares, sin las cuales yo no estaría aquí, y a las enfermeras del hospital de día que hicieron todo más fácil. A Cinthya, por enseñarme a comprenderme, a perdonarme y a ser justo conmigo mismo.

En última instancia, quiero agradecer a todas aquellas personas que en algún momento me han brindado su ayuda y, debido a mi torpeza, no he sido capaz de mencionar aquí. Afortunadamente, han sido muchas, porque la vida de los que disfrutamos del privilegio de la mediocridad está llena de personas excepcionales que nos proporcionan soporte, guía e impulso para seguir adelante.

Resumen

Esta tesis se lleva a cabo en colaboración con la empresa Madox Viajes, en el marco del proyecto *Pharaoh*, como parte de su estrategia de digitalización. La colaboración entre la empresa y la universidad se logra gracias a la financiación obtenida en la convocatoria de ayudas para Doctorados Industriales en la Comunidad de Madrid, que tiene como objetivo fomentar la colaboración entre el mundo académico y empresarial, facilitando la transferencia directa de conocimientos científicos generados en la universidad a las empresas.

Madox Viajes, fundada en 2008, es una empresa de nicho en el segmento de viajes a medida. Con el fin de mantenerse competitiva en un mercado turístico cada vez más desafiante, la empresa apuesta por la innovación científica, las matemáticas y la tecnología. En este sentido, el proyecto *Pharaoh* busca proporcionar a Madox Viajes los recursos necesarios para innovar en sus operaciones, incorporando componentes tecnológicos y científicos en sus sistemas *Enterprise Resource Planning* (*ERP*), *Customer Relationship Management* (*CRM*) y *eCommerce*, adaptándose a un futuro donde el mercado convergerá hacia el modelo *online* y competirá en igualdad de condiciones con grandes proveedores.

El objetivo empresarial de esta tesis en el proyecto *Pharaoh* es desarrollar un modelo de propensión de compra para el *scoring* de clientes utilizando aprendizaje basado en ejemplos. Para lograr esto, se desarrollarán modelos de aprendizaje automático centrados en mecanismos de muestreo que se adapten a cambios en el comportamiento de clientes y mercado, e identifiquen a los clientes que aporten más información en escenarios de baja tasa de conversión. El objetivo científico es investigar nuevas técnicas de muestreo para mejorar el rendimiento de modelos de *Machine Learning* (*ML*).

En esta investigación, se explora la aplicación de distintas técnicas de muestreo en el *ML* dentro del marco del aprendizaje estadístico, abordando el aprendizaje incremental, el aprendizaje combinado y el análisis de complejidad de los datos. Se presenta una nueva metodología de reentrenamiento para *Support Vector Machine* (*SVM*) basada en subconjuntos soporte, que permite una rápida y precisa actuali-

zación de modelos con nuevos datos, y un marco general de ensamblado llamado *Minimally Overfitted Ensemble* (*MOE*), que mejora la capacidad predictiva tanto de algoritmos base estables como inestables. Además, se introduce la medida de complejidad *Dynamic Disagreeing Neighbors* (*DDN*), que considera la dificultad de clasificar instancias en tres niveles: instancia, clase y conjunto de datos, y se basa en el cálculo de vecindarios dinámicos.

A lo largo de esta tesis, se han llevado a cabo numerosos experimentos y análisis para validar las técnicas y modelos propuestos. Estos experimentos se han realizado utilizando conjuntos de datos reales y sintéticos, y los resultados obtenidos han sido comparados con enfoques y algoritmos del estado del arte. Los hallazgos de estos experimentos han permitido identificar las fortalezas y debilidades de las técnicas propuestas, y han servido como base para realizar ajustes y mejoras en los modelos. Estos resultados también han proporcionado información valiosa sobre cómo las técnicas de muestreo pueden aplicarse de manera efectiva en diferentes contextos y desafíos empresariales.

La aplicación exitosa de técnicas de muestreo en distintos escenarios del *ML* en esta tesis tiene el potencial de impulsar el crecimiento y la competitividad de Madox Viajes. Los modelos desarrollados mejorarán la eficiencia y precisión en la clasificación de clientes, aumentando la tasa de conversión y optimizando la toma de decisiones. Además, ha proporcionado grandes avances científicos en los tres campos del *ML* donde se ha realizado la investigación que han dado lugar a diversas publicaciones científicas en revistas de gran impacto. Esta colaboración entre la universidad y Madox Viajes en el proyecto *Pharaoh* resalta cómo la transferencia de conocimientos científicos entre el mundo académico y empresarial puede generar beneficios mutuos e impulsar la innovación en la industria.

Abstract

This thesis is carried out in collaboration with the company Madox Viajes, within the framework of the *Pharaoh* project, as part of its digitization strategy. The collaboration between the company and the university is made possible thanks to the funding obtained through the call for support for Industrial Doctorates in the Community of Madrid. This initiative aims to promote collaboration between the academic and business worlds, facilitating the direct transfer of scientific knowledge generated in the university to companies.

Madox Viajes, founded in 2008, is a niche company in the custom travel segment. The company focuses on scientific innovation, mathematics, and technology to remain competitive in an increasingly challenging tourism market. In this sense, the *Pharaoh* project seeks to provide Madox Viajes with the necessary resources to innovate in its operations, incorporating technological and scientific components into its *Enterprise Resource Planning* (*ERP*), *Customer Relationship Management* (*CRM*), and *eCommerce* systems, and adapting to a future where the market will converge towards the *online* model, competing on equal terms with large providers.

The business objective of this thesis within the *Pharaoh* project is to develop a purchase propensity model for customer *scoring* using example-based learning. To achieve this, machine learning models will focus on sampling mechanisms that adapt to changes in customer behavior and market conditions and identify customers who provide more information in low conversion rate scenarios. The scientific objective is to investigate new sampling techniques to improve the performance of *Machine Learning* (*ML*) models.

This research explores the application of various sampling techniques in *ML* within the statistical learning framework, addressing incremental learning, combined learning, and data complexity analysis. A new retraining methodology for *Support Vector Machine* (*SVM*) based on support subsets is presented, allowing fast and accurate model updating with new data. Additionally, a general assembly framework called *Minimally Overfitted Ensemble* (*MOE*) is introduced, which improves the predictive capacity of both stable and unstable base algorithms. The complexity measure *Dy-*

namic Disagreeing Neighbors (*DDN*) is also introduced, considering the difficulty of classifying instances at three levels: instance, class, and dataset, and is based on the calculation of dynamic neighborhoods.

Throughout this thesis, numerous experiments and analyses have been carried out to validate the proposed techniques and models. These experiments have been performed using real and synthetic datasets, and the results obtained have been compared with state-of-the-art approaches and algorithms. The findings of these experiments have allowed for the identification of strengths and weaknesses of the proposed techniques and have served as a basis for making adjustments and improvements to the models. These results have also provided valuable insights into how sampling techniques can be effectively applied in different contexts and business challenges.

The successful application of sampling techniques in various *ML* scenarios in this thesis can potentially drive growth and competitiveness for Madox Viajes. The developed models will improve efficiency and accuracy in customer classification, increasing the conversion rate and optimizing decision-making. In addition, it has provided significant scientific advances in the three *ML* fields where research has been conducted, resulting in various scientific publications in high-impact journals. This collaboration between the university and Madox Viajes in the *Pharaoh* project highlights how the transfer of scientific knowledge between the academic and business worlds can generate mutual benefits and drive innovation in the industry.

Índice general

Índice de figuras

Índice de tablas

Listado de acrónimos

AdaBoost *Adaptive Boosting*.

B2B *Business to Business*.

B2C *Business to Consumer*.

Bagging *Bootstrap aggregating*.

CRM *Customer Relationship Management*.

DDN *Dynamic Disagreeing Neighbors*.

DT *Decision Tree*.

ERP *Enterprise Resource Planning*.

ET *Extra Trees*.

FN *Falsos Negativos*.

FP *Falsos Positivos*.

GB *Gradient Boosting*.

IS *Instance Selection*.

MCC *Mathews Correlation Coefficient*.

ML *Machine Learning*.

MOE *Minimally Overfitted Ensemble*.

NCN *Nearest Centroid Neighbors*.

OOB *out-of-bag*.

RF *Random Forest*.

SH *sampling heuristic*.

SMO *Sequential Minimal Optimization.*

SVM *Support Vector Machine.*

VN *Verdaderos Negativos.*

VP *Verdaderos Positivos.*

WRAB *Weighted RAndom Bootstrap.*

Wagging *Weight Aggregation.*

XGBoost *eXtreme Gradient Boosting.*

kDN *k-Disagreeing Neighbors.*

kNN *k-Nearest Neighbour.*

Capítulo 1

Introducción

Esta tesis se ha desarrollado en el marco de la empresa Jof Associates Int S.L. (también conocida como Madox Viajes), como parte del proyecto *Pharaoh*, que es la pieza angular de su estrategia de digitalización. La colaboración entre la empresa y la universidad para este proyecto ha sido posible gracias a la financiación obtenida en la convocatoria de ayudas para la realización de Doctorados Industriales en la Comunidad de Madrid, con referencia IND2019/TIC 17194. El objetivo de estas ayudas es fomentar la colaboración entre el mundo académico y el entorno empresarial, para lograr la transferencia directa de los conocimientos científicos generados en la universidad a las empresas.

Esta alianza universidad-empresa demuestra la importancia de combinar el conocimiento teórico y la investigación académica con la experiencia práctica y las necesidades reales del sector empresarial. Esta colaboración permite que las innovaciones científicas y tecnológicas desarrolladas en la universidad se apliquen de manera efectiva en el mundo empresarial, mejorando así la competitividad y el rendimiento de la compañía. Además, la empresa se beneficia del acceso a recursos académicos especializados y a investigadores altamente capacitados que pueden ayudar a resolver problemas específicos y a desarrollar soluciones a medida. Por otro lado, la universidad se beneficia de la oportunidad de investigar problemas reales y aplicar sus descubrimientos en un entorno empresarial, lo que enriquece la formación de sus estudiantes y fortalece la relación entre la academia y la industria.

La empresa Madox Viajes

Madox Viajes es una empresa de nicho en el segmento de viajes a medida que fue fundada en 2008. Algunos datos relevantes de la compañía son un volumen de negocio de 3M €/año, un total de 2000 clientes, más de 3000 viajes combinados realizados, un local abierto al público situado en Arroyomolinos (Madrid) y tres sitios web para la captura de oportunidades. En sus 15 años de existencia, Madox Viajes ha logrado

competir en igualdad de condiciones con los actores más importantes del sector y ha obtenido importantes logros, como el estándar de calidad de turismo español "Q de Calidad" en 2017.

Para mantenerse competitiva en un mercado turístico cada vez más desafiante, Madox Viajes ha decidido renovarse mediante la innovación científica y apostar por las matemáticas y la tecnología para conseguir sus objetivos. La empresa busca adaptarse a un futuro donde el mercado convergerá hacia el modelo *online* y competirá en igualdad de condiciones con grandes proveedores. Madox Viajes está enfocada en ofrecer la mejor experiencia de viaje a sus clientes y continuar luchando por oportunidades en mercado cada vez más competitivo y desafiante, que se encuentra en constante evolución.

El proyecto *Pharaoh*

La misión del proyecto *Pharaoh* es brindar a la empresa los recursos necesarios para innovar en sus operaciones mediante la incorporación de nuevos componentes tecnológicos y científicos en sus sistemas. Estos componentes son capaces de adaptar su comportamiento según cambia el escenario y los datos subyacentes. La tesis que se ha desarrollado forma parte del proyecto *Pharaoh* y sus resultados serán integrados en los siguientes sistemas de la empresa:

- *Enterprise Resource Planning* (*ERP*): un sistema de gestión empresarial que permite a la organización gestionar y optimizar sus procesos empresariales y recursos en una sola plataforma integrada. Este sistema dispone de funcionalidades de gestión de expedientes y de inteligencia de negocio para la toma de decisiones por parte de la dirección.
- *Customer Relationship Management* (*CRM*): un sistema de gestión de relaciones con los clientes que ayuda a la empresa a administrar las interacciones con sus clientes actuales y potenciales. El *CRM* incluye herramientas gestión de oportunidades y un modelo de *scoring* estático codificado con reglas explícitas de negocio.
- El *eCommerce* es una forma de comercio electrónico que consiste en la compra y venta de productos o servicios a través de Internet. Actualmente en desarrollo, con futuras funcionalidades *Business to Business* (*B2B*) y *Business to Consumer* (*B2C*) para la venta de viajes de todo tipo, incluyendo los más complejos, como los combinados. Utilizando para ello conexiones a servicios web de proveedores para la obtención de precios de forma dinámica.

El proyecto *Pharaoh* se centra en la mejora de las diversas piezas de *software* de la empresa mediante la implementación de nuevas soluciones tecnológicas que funcionen como sistemas de apoyo a la toma de decisiones basados en el conocimiento

extraído de los datos. Al incorporar estos componentes en los sistemas existentes, se busca optimizar sus capacidades y permitir que se ajusten de manera autónoma a las fluctuaciones y cambios en el entorno empresarial sin la necesidad de intervención humana constante. Al hacerlo, esperamos que se refleje en un aumento de las ventas y una mejora en la calidad de las recomendaciones de viajes, con el objetivo de aumentar la eficiencia y mejorar la calidad de las decisiones.

Objetivo

El objetivo empresarial de esta tesis en el proyecto *Pharaoh* es desarrollar un modelo de propensión de compra para el *scoring* de clientes utilizando aprendizaje basado en ejemplos. Este *scoring* permitirá a los agentes comerciales gestionar oportunidades con información detallada, mejorando la eficiencia en el tiempo y aumentando la tasa de conversión. La finalidad es optimizar la gestión de clientes y recursos empresariales para mejorar el rendimiento y los resultados del negocio.

Para lograr esto, se desarrollarán modelos de aprendizaje automático (*Machine Learning* (*ML*)) centrados en mecanismos de muestreo para identificar el conjunto óptimo de clientes, minimizando el espacio de almacenamiento requerido y maximizando el rendimiento de los modelos. Esto simplificará futuros análisis al disponer de una muestra representativa.

Los modelos construidos abordarán enfoques que cubran las necesidades del negocio, como implementar un proceso de actualización y mejora continua, adaptándose a cambios en el comportamiento de clientes y mercado. Además, identificarán qué clase de clientes aporta más información en escenarios de baja tasa de conversión. Es decir, situaciones en las que la proporción de clientes potenciales que finalmente realizan una compra o se convierten en clientes de pago es relativamente pequeña en comparación con el total de clientes potenciales.

El objetivo científico es analizar estrategias para mejorar el rendimiento de clasificadores mediante nuevas técnicas de muestreo. Investigaremos si el muestreo recurrente mejora el rendimiento de clasificadores incrementales y evaluaremos la contribución informativa de cada clase en problemas de clasificación desequilibrados. Además, exploraremos la identificación de observaciones relevantes y la factibilidad de ensamblar algoritmos de *ML* estables, considerando la diversidad del muestreo y el sobreajuste local.

1.1. Aprendizaje basado en ejemplos

El aprendizaje es una habilidad innata del ser humano, tan natural que cualquiera puede comprender su concepto, reconocer cuando alguien está aprendiendo o ha

aprendido algo, e incluso identificar este proceso en otros seres vivos. El aprendizaje es un concepto que ha sido objeto de estudio en numerosos ámbitos de la ciencia. Desde la perspectiva de la psicología, el proceso del aprendizaje ha sido ampliamente investigado, y podemos decir que se trata de un proceso a través del cual se adquiere una habilidad o conocimiento después de haber asimilado cierta información. Desde el punto de vista del constructivismo (Vygotsky y Cole, 1978), el aprendiz es responsable de interpretar y dar sentido al nuevo conocimiento. Por lo tanto, no es un sujeto pasivo que recibe información para almacenar, sino que es un agente activo que construye significados a través de la combinación de su estructura mental y sus experiencias previas. Así, ante un mismo estímulo, la respuesta del individuo puede variar dependiendo de su proceso de aprendizaje.

En el ámbito de la Inteligencia Artificial, el *ML* imita el proceso del aprendizaje humano mediante la experiencia (Spicer y Sanborn, 2019). Para entender este proceso, podemos utilizar el concepto de aprendizaje como un proceso en el que se infieren reglas a partir de ejemplos. Estos ejemplos son los datos en un espacio de características, que podemos considerar como las experiencias de un individuo. El algoritmo que procesa estos datos puede ser visto como el aprendiz, con sus propias singularidades que implican una forma específica de procesar los datos. El proceso de aprendizaje se produce cuando el algoritmo se expone a los datos, lo que finalmente conduce a un modelo de las reglas subyacentes en los datos (construye significados). Este modelo puede estar relacionado con la estructura del espacio de características que definen los datos o con la relación entre este espacio y otro espacio respuesta.

En términos generales, el *ML* se divide en tres tipos de aprendizaje (Ayodele, 2010): supervisado, no supervisado y por refuerzo. En el aprendizaje supervisado, se proporciona un conjunto de datos etiquetados para entrenar un modelo que luego puede predecir nuevos datos sin etiquetar. En el aprendizaje no supervisado, el objetivo es encontrar patrones o relaciones en los datos sin etiquetar. Por último, en el aprendizaje por refuerzo, el modelo aprende a tomar decisiones para maximizar la recompensa que recibe por su desempeño en una tarea.

Es importante destacar que el *ML* se centra en la "máquina" y representa un cambio de paradigma en la programación: significa aprender un algoritmo a partir de datos en lugar de programar el algoritmo manualmente. El éxito de un modelo de *ML* se mide por lo bien que resuelve una tarea, mientras que en la estadística clásica, la motivación de modelización y la evaluación son intrínsecas, basadas en la teoría y la coherencia con los datos. En resumen, la estadística clásica y el *ML* comparten el objetivo de modelar los datos, pero difieren en su enfoque y evaluación.

Aunque la estadística clásica y el *ML* tienen enfoques diferentes (Ij, 2018), existe un enfoque en el *ML* desde la perspectiva de la estadística conocido como aprendizaje estadístico, que es la forma más extendida de entender el *ML* en la actualidad

(Hastie *et al.*, 2009). Este enfoque utiliza herramientas, técnicas y resultados teóricos de la estadística para descubrir leyes y propiedades de los modelos o sus resultados. Podemos entender el aprendizaje estadístico como una aproximación rigurosa al *ML* desde el marco de la estadística y las matemáticas. En definitiva, el aprendizaje estadístico permite una comprensión más profunda del funcionamiento del *ML*. Además, el soporte estadístico es fundamental para el análisis de datos, lo que lleva a una mejor comprensión de estos y, en consecuencia, al desarrollo de algoritmos que utilicen esta información para una modelización de excelente calidad y rendimiento.

Algunas de las aplicaciones más comunes del campo del *ML* (Shinde y Shah, 2018) incluyen el análisis de datos, el reconocimiento de patrones y clasificación, la predicción de resultados futuros, la automatización de tareas y la creación de sistemas de recomendación. En el análisis de datos, el *ML* se utiliza para descubrir relaciones ocultas en conjuntos de datos complejos. En el reconocimiento de patrones, el *ML* se utiliza para identificar patrones y clasificar datos en diferentes categorías. En la predicción, el *ML* se utiliza para predecir resultados futuros basados en datos históricos. En la automatización de tareas, el *ML* se utiliza para automatizar tareas repetitivas y procesos empresariales. Y, en los sistemas de recomendación, el *ML* se utiliza para proporcionar recomendaciones personalizadas a los usuarios.

A pesar de las ventajas que ofrece el aprendizaje automático, como la capacidad de procesar grandes cantidades de datos y encontrar patrones complejos, también tiene algunas desventajas (Malik, 2020). Una de ellas es la necesidad de grandes cantidades de datos para entrenar modelos precisos y efectivos. Otra desventaja es la posible falta de interpretabilidad de los modelos, lo que dificulta la comprensión de cómo se tomaron ciertas decisiones. También es importante tener cuidado con el preprocesamiento de los datos para evitar los sesgos y el sobreajuste (dificultad de generalización). Es importante tener en cuenta tanto las ventajas como las desventajas del *ML* al decidir si es apropiado para una tarea o problema específico.

Actualmente, el *ML* es un campo de investigación en constante evolución con diversas líneas de desarrollo. Algunas de las áreas más destacadas incluyen el procesamiento del lenguaje natural (Khurana *et al.*, 2023), donde los modelos de *ML* se utilizan para comprender y generar lenguaje humano, y la justicia en *ML* (Carey y Wu, 2023), que se centra en garantizar la equidad y la no discriminación en los modelos de *ML*. Además, también se está investigando el aprendizaje incremental u *online*, donde los modelos de *ML* pueden actualizarse y mejorar continuamente a medida que se recopilan nuevos datos (Hoi *et al.*, 2021), y el *ML* interpretable (Belle y Papantonis, 2021), donde se buscan técnicas para hacer que los modelos sean más comprensibles para los humanos. Por último, el aprendizaje por refuerzo es otra área de investigación destacada (Moerland *et al.*, 2023), que se enfoca en la enseñanza de los modelos a través de la interacción con su entorno, con el objetivo de maximizar una recom-

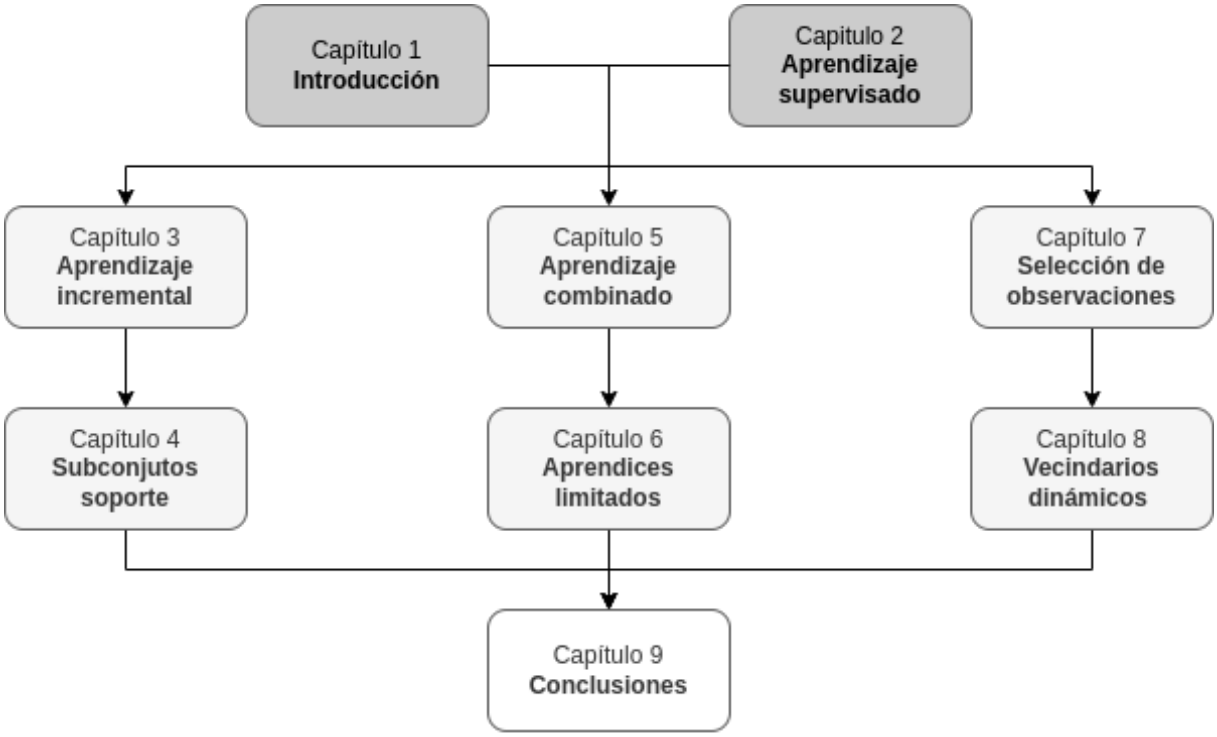


Figura 1.1: *Esquema de la estructura del documento: Los dos primeros capítulos abordan temas generales de la investigación (gris oscuro). Los seis siguientes se organizan en pares, enfocándose en un ámbito específico en cada par (gris claro). Finalmente, el último capítulo presenta las conclusiones (blanco). Las flechas representan el orden de lectura.*

pensa o objetivo específico. Todas estas áreas de investigación tienen el potencial de impulsar significativamente el desarrollo y la aplicación del *ML* en diversos campos, como la medicina, la ingeniería, la logística y la educación, entre otros.

1.2. Esquema de la tesis

Este documento está organizado en nueve capítulos donde se describen todos los aspectos relevantes de la investigación, así como los resultados alcanzados. Los dos primeros capítulos tratan temas generales, los seis siguientes abordan ámbitos específicos y se organizan por pares. Finalmente, el último capítulo presenta las conclusiones. La figura 1.1 muestra esta estructura y la relación de los capítulos. En este primer capítulo se introduce el contexto y motivación de la investigación, así como el marco de trabajo general. Además, se presentan las principales contribuciones de la tesis.

El capítulo 2 se dedica a la introducción del problema de clasificación, un caso particular del aprendizaje supervisado, y su visión como una tarea predictiva. A lo

largo de este capítulo se desarrollan todos los conceptos teóricos necesarios en el resto de la investigación, incluidas las suposiciones en el contexto probabilístico que implica el aprendizaje estadístico y los fundamentos de los principales algoritmos predictivos que se emplearán más adelante.

El capítulo 3 sirve como introducción del aprendizaje incremental en *ML*, donde explicamos que el desarrollo y mantenimiento de los modelos predictivos es un sistema vivo que requiere de un cuidado y mantenimiento constante. En la sección 3.1 se explican los métodos iterativos existentes en el caso particular de los *Support Vector Machine*s (*SVM*s). A continuación, en el capítulo 4 se presentan los subconjuntos soporte, un resultado teórico con el cual garantizamos que con una muestra del conjunto de entrenamiento inicial y los nuevos datos disponibles en un momento del tiempo se produce un clasificador equivalente al clasificador entrenado con ambos conjuntos de datos completos. En la sección 4.2 se presenta una metodología de reentrenamiento para *SVM*s basada en el concepto de los subconjuntos soporte. El buen rendimiento de esta nueva metodología se expone en la sección 4.3. Finalmente, las lecciones más importantes extraídas de la propuesta se resumen en la sección 4.4.

El capítulo 5 resume los aspectos importantes del aprendizaje combinado para el ensamblaje de modelos. En particular, nos centramos en los ensamblados generativos presentados en la sección 5.1, que se basan en el remuestreo del conjunto de entrenamiento y son la base de nuestra propuesta. El *Minimally Overfitted Ensemble* (*MOE*) es un nuevo marco de aprendizaje combinado basado en el sobreajuste local de los aprendices limitados, presentados en la sección 6.1. La idea es sobreajustar ligeramente modelos en distintas regiones del espacio de características. Para lograrlo, presentamos en la sección 6.2.1 el método de muestreo *Weighted RAndom Bootstrap* (*WRAB*), que garantiza que este mínimo sobreajuste genere fronteras de decisión diversas que al combinarse produzcan un clasificador de gran rendimiento. Estas suposiciones se confirman en la sección 6.3, donde se muestran los resultados de la experimentación que ratifican el buen funcionamiento del *MOE*.

El último bloque se enfoca en el problema de clasificación estático, en el cual se limitan los datos de entrenamiento en función de la dificultad que presenta este conjunto de ser clasificado correctamente. En el capítulo 7 se presenta el problema de la selección de observaciones (*Instance Selection* (*IS*)) y cómo puede ser abordado mediante la clasificación de cada observación en función de su complejidad. En el capítulo 8 se presenta la nueva medida de complejidad *Dynamic Disagreeing Neighbors* (*DDN*) que estima la complejidad de cada observación según la proporción de las clases opuestas que pertenecen a un vecindario. La novedad de esta medida es la forma en la que los vecindarios se calculan dinámicamente y la importancia de cada observación se pondera según la distancia a la observación medida, todo ello se

encuentra detallado en la 8.1. Además, el rendimiento de esta nueva medida se evalúa en la sección 8.2, comparándola con *k-Disagreeing Neighbors* (*kDN*) una medida referencia en el estado del arte. Por último, en la sección 8.2.3 planteamos una heurística de muestreo que emplea la complejidad de cada observación para seleccionar una muestra representativa del conjunto de datos original, y comprobamos el buen rendimiento de *DDN* para este propósito.

El capítulo 9 concluye la investigación presentando un resumen de las conclusiones generales, las principales contribuciones y las oportunidades de mejora que se presentan a futuro. Además, se proporciona una lista de las publicaciones en las que se detallan cada una de las contribuciones y en las que se ha colaborado de forma independiente a la línea de investigación.

1.3. Contribuciones principales

En este trabajo, nos centramos en la aplicación de distintas técnicas de muestreo para resolver el problema de clasificación en el marco del aprendizaje estadístico. Primero se aborda este problema en un entorno de aprendizaje incremental. A continuación, se trata la clasificación en el aprendizaje combinado. Por último, nos aproximamos al problema de clasificación desde la perspectiva del análisis de complejidad de los datos. Las principales contribuciones de esta tesis son una serie de resultados teóricos y los algoritmos que explotan estos resultados utilizando el muestreo como pieza distintiva y fundamental.

Aprendizaje incremental

En general, la disponibilidad de nuevos datos, distintos a los empleados para entrenar y validar el modelo, es una situación común en muchos escenarios de *ML*, como el procesamiento del lenguaje natural, la visión por computadora, la bioinformática, entre otros. Sin embargo, incorporar nuevos datos a un modelo de *ML* previamente entrenado puede resultar un desafío, ya que a menudo se requiere volver a entrenar y ajustar el modelo para reflejar la influencia de los nuevos datos, especialmente cuando el modelo se actualiza regularmente. Por lo tanto, es importante buscar soluciones eficientes y efectivas para actualizar los modelos de *ML* con nuevos datos de forma rápida y precisa. Esto es particularmente importante en algoritmos como el *SVM*, ampliamente utilizados en *ML* debido a sus sólidas bases matemáticas y flexibilidad, donde el entrenamiento de *SVM* es computacionalmente costoso, tanto en tiempo como en memoria. Por lo tanto, la fase de entrenamiento puede ser una limitación en problemas donde el modelo se actualiza regularmente. Como solución, se han propuesto nuevos métodos para entrenar y actualizar *SVM* en el pasado. En este trabajo, introducimos el concepto de subconjunto soporte y una nueva

metodología de reentrenamiento para *SVM*. Un subconjunto soporte es un subconjunto del conjunto de entrenamiento, de manera que reentrenar un modelo de *ML* con este subconjunto y los nuevos datos es equivalente a entrenar con todos los datos. Se evalúa el rendimiento de la propuesta en una variedad de experimentos en conjuntos de datos simulados y reales en términos de tiempo, calidad de la solución, vectores soporte resultantes y cantidad de datos utilizados. Los resultados prometedores proporcionan una nueva línea de investigación para mejorar la efectividad y adaptabilidad de la técnica propuesta, incluyendo su generalización a otros modelos de *ML*.

Aprendizaje combinado

La combinación de algoritmos de *ML* es una solución para construir predictores más robustos que aquellos basados en algoritmos individuales. Sin embargo, algunas aproximaciones sugieren que combinar algoritmos inestables proporciona mejores resultados que combinar algoritmos estables. Por ejemplo, los ensambles generativos, basados en técnicas de remuestreo, han demostrado un rendimiento alto al fusionar la información de los aprendices base inestables. *Random Forest* (*RF*) y *Gradient Boosting* (*GB*) son dos ejemplos conocidos, que combinan *Decision Tree*s (*DT*s) y proporcionan predicciones mejores que las obtenidas usando un solo árbol. Sin embargo, no se han logrado resultados tan exitosos al ensamblar algoritmos estables. Este documento introduce la noción de aprendiz limitado y un nuevo marco general de ensamblado llamado *MOE*, un enfoque de ensamblado basado en remuestreo que construye aprendices ligeramente sobreajustados. El marco propuesto funciona bien con algoritmos base estables e inestables, gracias a un muestreo que proporciona la diversidad necesaria para los algoritmos base estables. Se realiza un análisis de hiperparámetros de la propuesta en datos artificiales. Además, su rendimiento se evalúa en conjuntos de datos reales en comparación con métodos conocidos de *ML*. Los resultados confirman que el marco *MOE* funciona con éxito usando algoritmos base estables e inestables, mejorando en la mayoría de los casos la capacidad predictiva de modelos únicos de *ML* y otros métodos de ensamblado.

Reducción de la complejidad y selección de instancias

Dentro del ámbito de *ML*, *IS* es un proceso de muestreo que consiste en filtrar el ruido y eliminar los datos redundantes. En problemas de clasificación, *IS* es un compromiso entre maximizar el rendimiento y reducir los datos utilizados para el entrenamiento. Las medidas de complejidad proporcionan información sobre la dificultad de clasificar las instancias, lo que las hace adecuadas para *IS* ya que pueden capturar, por ejemplo, puntos ruidosos o fronterizos. Este trabajo introduce la medida de complejidad *DDN*, definida en tres niveles: instancia, clase y conjunto de datos. La *DDN* de una instancia se define como el porcentaje de sus vecinos más cercanos que pertenecen a otras clases. La *DDN* se basa en el cálculo del vecinda-

rio *Nearest Centroid Neighbors* (*NCN*), que se ajusta dinámicamente a la distribución de datos. Además, se tiene en cuenta la distancia de cada vecino para que aquellos más lejanos tengan menos influencia y aquellos más cercanos tengan más influencia para la complejidad de la instancia. La validez de la propuesta se evalúa a través de una serie de experimentos donde se compara con el conocido *kDN* en términos de estabilidad, correlación con el error de clasificación y rendimiento en *IS*. La *DDN* ha mostrado resultados competitivos, estables y robustos a lo largo de los experimentos, mejorando generalmente los obtenidos con la alternativa.

Capítulo 2

Aprendizaje supervisado: Problema de la clasificación

En *ML* un problema de aprendizaje supervisado se considera coloquialmente una tarea de predicción. El término predicción hace referencia a la existencia del valor real que queremos estimar (Molnar, 2022). Por tanto, en el aprendizaje supervisado se estiman una serie de valores que poder comparar con el dato real, denominados habitualmente etiquetas. Así pues, abordaremos un problema de aprendizaje supervisado cuando dispongamos de un conjunto de datos etiquetado. Cuando el problema es de regresión nos encontramos con etiquetas numéricas y, si el problema es de clasificación dispondremos de etiquetas categóricas. Generalmente, consideramos una variable respuesta Y y r variables explicativas o características $\boldsymbol{X} = (X_1, X_2, \ldots, X_r)$ que se relacionan de la siguiente manera:

$$Y = f(\boldsymbol{X}) + \epsilon \tag{2.1}$$

donde f es una función concreta aunque desconocida y ϵ es el término de error aleatorio, independiente de $\boldsymbol{X}$ y de media cero.

El objetivo de los algoritmos de aprendizaje supervisado es aproximarse a esa función de predicción f, mediante la cual poder estimar valores para las etiquetas reales. Todas las posibles funciones de predicción pertenecen al espacio de hipótesis (Hastie *et al.*, 2009). Cada modelo de aprendizaje supervisado dispone de distintos mecanismos para disminuir el espacio de hipótesis, asumiendo una serie de condiciones inherentes al diseño del propio algoritmo. De esta forma, se inducen unos sesgos que condicionan las posibles funciones de predicción que el modelo es capaz de inferir en base a los datos de que dispone. Dentro de este espacio de hipótesis restringido el modelo define la mejor función posible $\hat{f}$ tal que $\hat{Y} = \hat{f}(\boldsymbol{X})$ para cualquier observación $(\boldsymbol{X}, Y)$, minimizando la diferencia existente entre las etiquetas estimadas y las

reales.

El aprendizaje de un modelo, por tanto, es un proceso de búsqueda y optimización. Por ello, hemos de ser capaces de medir cuánto de equivocado o acertado está el algoritmo en sus predicciones. Esto nos lleva al concepto de error de predicción. En un problema de clasificación el error de predicción puede entenderse como la probabilidad de clasificar mal un caso, es decir, asignar una etiqueta errónea. Desde un punto de vista formal, consideremos f la función de predicción que queremos estimar en base a un conjunto de datos de aprendizaje, que podemos escribir como $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ donde las características están representadas por los vectores $\boldsymbol{x_i} \in \mathbb{R}^r$ y las clases por los valores y_i. El error de predicción puede definirse como:

$$L_{01}(y_i, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^{n} I_{\{y_i \neq \hat{y}_i\}} \tag{2.2}$$

donde $\hat{y}_i$ es la clase predicha para la observación i-ésima utilizando la función $\hat{f}$ de predicción construida con el modelo de clasificación, e I es la función indicatriz tal que toma valor 1 si $y_i \neq \hat{y}_i$ y 0 en caso contrario. Nótese, que en este caso se está empleando la función de pérdida 0-1, que pondera con la misma importancia el error cometido en cualquiera de las clases.

Una vez conseguimos estimar la función de predicción y tenemos una medida para evaluar qué tan buena es dicha función, ¿podemos estar seguros de la calidad de las predicciones en un nuevo conjunto de datos? La suficiencia de un modelo para conservar el error observado al entrenarlo es lo que se conoce comúnmente como capacidad de generalización. La solución que se propone dentro del paradigma del *ML* es construir mediante algún método de muestreo aleatorio tres conjuntos de datos (Izenman, 2008). Un conjunto de aprendizaje (o entrenamiento) $\mathcal{L}$, un conjunto de datos donde "todo vale", del cual extraeremos todo el conocimiento posible para procesar los datos, extraer patrones, probar distintos algoritmos de aprendizaje y, en definitiva, aplicar todas las técnicas de minería de datos que sea pertinentes para realizar nuestro análisis. Un conjunto de validación (o desarrollo) $\mathcal{V}$, un conjunto de datos que se utilizará para la selección de modelos, realizar un ajuste fino de ellos, y evaluar los distintos algoritmos probados comparando su capacidad predictiva. Finalmente, un conjunto de test (o evaluación) $\mathcal{T}$, un conjunto de datos que se utilizará para evaluar el rendimiento de un modelo final completamente especificado.

La suposición necesaria en este caso es que la distribución subyacente que genera los tres conjuntos de datos es la misma. Esto quiere decir que, en general, el conjunto de datos que podemos observar en un problema supervisado $\mathcal{D} = \{\mathcal{L} \cup \mathcal{V} \cup \mathcal{T} : \mathcal{L} \cap \mathcal{V} = \mathcal{L} \cap \mathcal{T} = \mathcal{V} \cap \mathcal{T} = \emptyset\}$ es una muestra aleatoria extraída de la distribución de

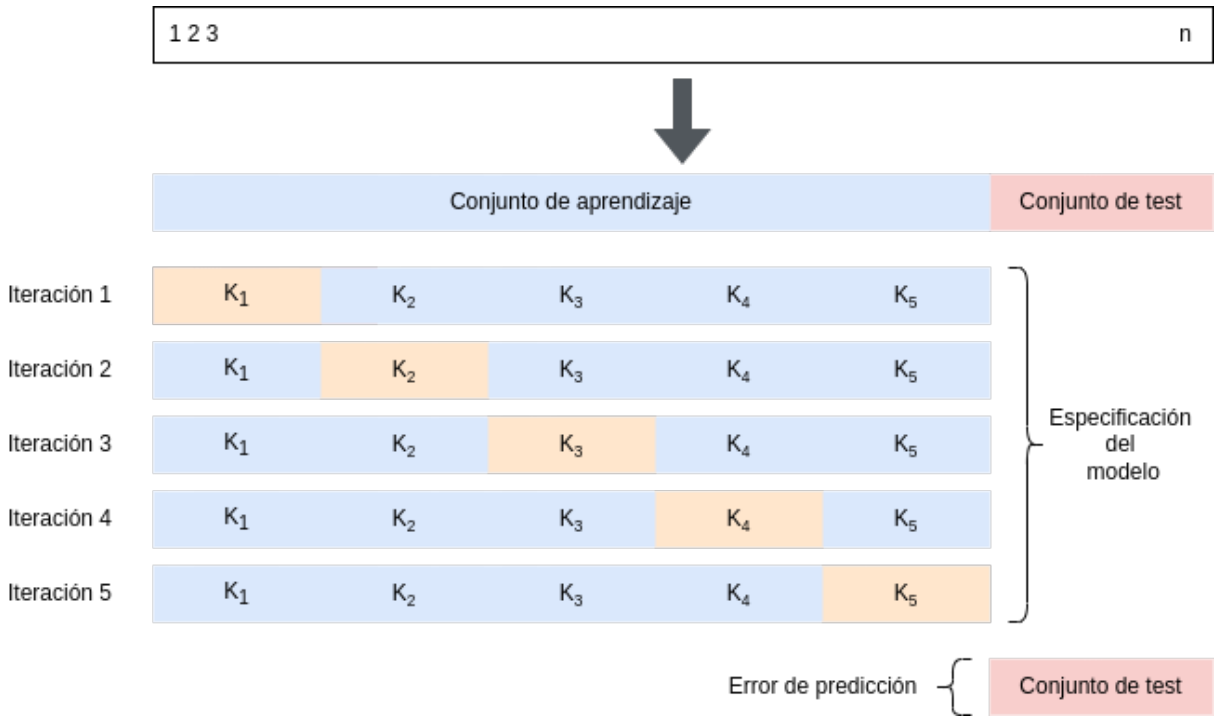


Figura 2.1: *Una ilustración de la validación cruzada de 5 particiones. Un conjunto de datos de n observaciones se divide en conjunto de aprendizaje y test. A continuación, el conjunto de aprendizaje se divide en cinco particiones sin solapamiento. Cada una de las particiones omitidas, representadas en naranja, se utilizan para evaluar y sobre el resto, mostradas en azul, se realiza el ajuste del modelo. El error de predicción en la fase de desarrollo se estima promediando los cinco errores calculados en cada iteración. El error de predicción será el resultante de evaluar el modelo completamente especificado en el conjunto de test, representado en rojo.*

probabilidad conjunta $P(\boldsymbol{X}, Y)$ en un espacio $(r+1)$-dimensional.

En la práctica no siempre puede dividirse el conjunto $\mathcal{D}$ en tres conjuntos de datos para realizar el proceso de entrenamiento, desarrollo y evaluación de manera que estemos seguros de los resultados obtenidos. En ocasiones el conjunto de datos no es lo suficientemente grande para realizar la división y/o es demasiado heterogéneo. En estos casos existen soluciones elegantes como la validación cruzada de k-particiones (Stone, 1974). Este método consiste en dividir el conjunto $\mathcal{D}$ en k particiones sin solapamiento de igual tamaño, eliminar uno de los grupos y ajustar el modelo con los $k-1$ restantes. A continuación, se utiliza la partición omitida para evaluar el error de predicción. Este proceso se repite k veces, eliminando una partición cada vez y se promedian los k errores de predicción para estimar la capacidad de generalización. Este método se ha demostrado eficaz para calibrar el modelo en la fase de validación (Golub *et al.*, 1979), pudiendo evaluar además el error de predicción fielmente. Para garantizar que el error de generalización sea representativo y minimizar el riesgo en el proceso (Rao *et al.*, 2008), la validación cruzada se emplea en la fases de entrena-

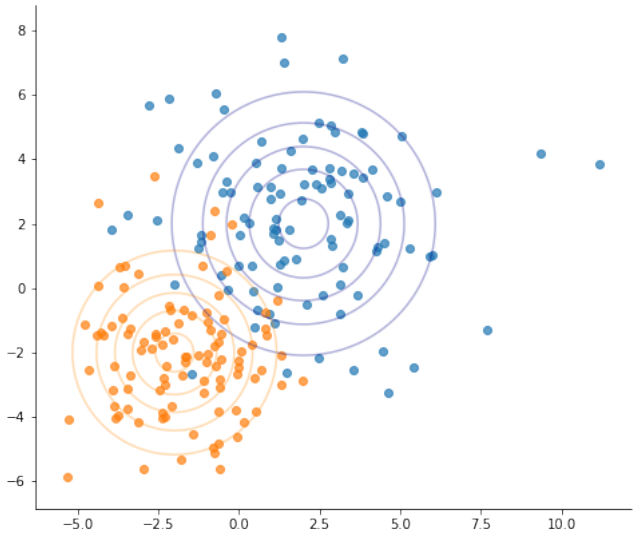


Figura 2.2: *Un conjunto de datos simulados mediante dos distribuciones normales bivariantes formado por 100 puntos para cada una de las clases, indicadas en azul y naranja. Se acompaña cada nube de puntos con líneas de contorno.*

miento y desarrollo, donde se comparan los algoritmos de aprendizaje. Sin embargo, el rendimiento del modelo completamente especificado se comprueba en el conjunto de test. En la figura 2.1 se ilustra el proceso de la validación cruzada de manera esquemática.

En definitiva, un buen clasificador se construye cuando el error medido en el conjunto de test $\mathcal{T}$ es el mínimo posible. Por tanto, la tasa de error en el conjunto de test se calcula aplicando la ecuación (2.2), correspondiente al error de predicción, únicamente sobre las observaciones pertenecientes al conjunto de test. El calculo de esta tasa de error en test queda como sigue:

$$\frac{1}{\text{card}(\mathcal{T})} \sum_{j=1}^{\text{card}(\mathcal{T})} I_{\{y_i \neq \hat{y}_i\}} \tag{2.3}$$

tal que cada observación $(\boldsymbol{x}_j, y_j) \in \mathcal{T}$ y, donde $\text{card}(\mathcal{T})$ representa el número de elementos del conjunto de test.

La tasa de error del conjunto de test de (2.3) es mínima para el clasificador Bayes. Este clasificador asigna cada observación a la clase más probable dado un vector de características. Formalmente, sea G una variable respuesta categórica que toma va-

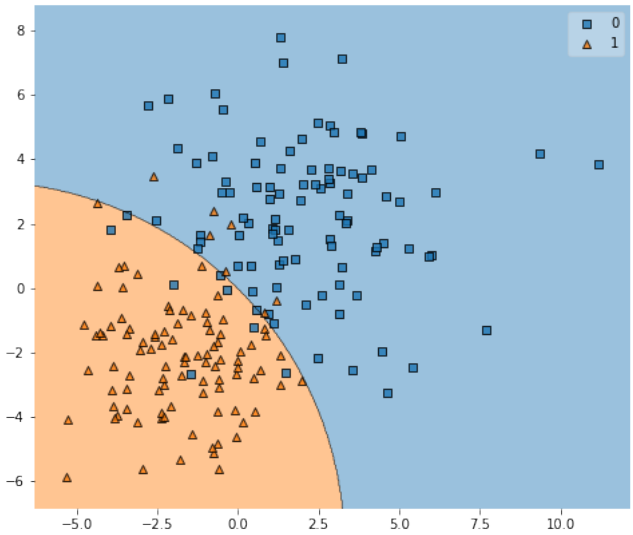


Figura 2.3: *Un conjunto de datos simulados formado por 100 puntos para cada una de las clases, indicadas en azul y naranja. El fondo de color naranja indica la región en la que se asignará a la clase naranja, y el fondo azul indica la región en la que se asignará a la clase azul.*

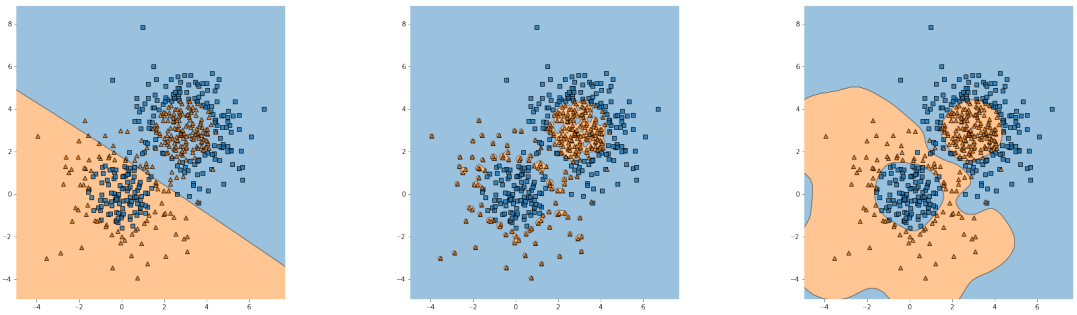
lores en el conjunto $\mathcal{G}$ de las k clases posibles. Empleando la función de pérdida 0-1, el clasificador Bayes se puede definir como:

$$\hat{f}(\boldsymbol{x}) = \mathcal{G}_k \; si \; P(\mathcal{G}_k|\boldsymbol{X} = \boldsymbol{x}) = max_{g\in\mathcal{G}} P(g|\boldsymbol{X} = \boldsymbol{x}) \tag{2.4}$$

El clasificador Bayes minimiza la tasa de error en test (2.3). Este mínimo error se denomina tasa de error Bayes y viene dada por la expresión:

$$1 - E\left(max_k \, P(G = \mathcal{G}_k|\boldsymbol{X})\right) \tag{2.5}$$

donde $E(\cdot)$ es la esperanza matemática que promedia la probabilidad sobre todos los valores posibles de la variable aleatoria $\boldsymbol{X}$. Consideremos un problema de clasificación binaria sencillo, por ejemplo, dos distribuciones normales bivariantes como las mostradas en la figura 2.2. Para este caso, conociendo la distribución que genera los datos, se pueden calcular la probabilidad condicionada de una de las clases para obtener la frontera de decisión inducida por el clasificador Bayes, como queda reflejado en la figura 2.3. En general, los algoritmos de aprendizaje automático que modelan directamente la probabilidad condicionada $P(Y|\boldsymbol{X})$ se denominan modelos



(a) *Subajuste* **(b)** *Sobreajuste* **(c)** *Buen ajuste*

Figura 2.4: *Ejemplo de un clasificador binario en los distintos supuestos de subajuste, sobreajuste y buen ajuste.*

discriminativos y son los más comunes (Bernardo *et al.*, 2007) en la literatura en los problemas de clasificación.La otra alternativa, generalmente más compleja, es estimar directamente la probabilidad conjunta $P(Y, \boldsymbol{X})$ para posteriormente calcular la probabilidad condicionada que necesitamos para predecir.

Los modelos discriminativos pueden entenderse como una función de probabilidad condicionada que depende de unos parámetros, que son los llamados hiper parámetros del modelo, por tanto, en esta aproximación al problema de clasificación un modelo puede definirse como:

$$P(G|\boldsymbol{X}, \boldsymbol{\theta}) = \Pi_{i=1}^{n} P(y_i|\boldsymbol{x_i}, \boldsymbol{\theta})) \tag{2.6}$$

En la fase de aprendizaje se estima el conjunto de parámetros del modelo $\boldsymbol{\theta}$, en la fase de validación se comprueba la calidad de la solución que aportan esos hiperparámetros en el conjunto de validación y finalmente, elegidos los mejores hiperparámetros se evalúa la capacidad de generalización del modelo completamente especificado. Después de este proceso se puede llegar exactamente a tres conclusiones, a saber, que el modelo esté subajustado, que esté sobreajustado o que esté bien ajustado (Ying, 2019). El sobreajuste ocurre cuando un modelo es demasiado complejo y se ajusta demasiado bien al conjunto de aprendizaje. Es decir, no es capaz de aprender los patrones generales de los datos, si no que aprende de manera muy precisa las características del conjunto de aprendizaje, y pierde la capacidad de generalización. De este modo, tiene un rendimiento peor en el conjunto de test que en el conjunto de aprendizaje. Por otro lado, un modelo subajustado sucede cuando este es demasiado simple y tiene un rendimiento pobre en el conjunto de aprendizaje. Dicho de otro modo, no se ajusta lo suficientemente bien al conjunto de aprendizaje y por lo tanto, también tendrá un rendimiento pobre en el conjunto de test. En

		Predicho	
		+1	-1
Real	+1	VP	FN
	-1	FP	VN

Tabla 2.1: *Matriz de confusión 2x2 para un problema de clasificación binaria.*

general, un modelo sobreajustado es demasiado complejo y un modelo subajustado es demasiado simple. Sin embargo, un modelo bien ajustado es aquel que tiene una complejidad adecuada para aprender los patrones importantes del conjunto de aprendizaje y generalizar bien a nuevos datos. En la figura 2.4 se muestran los tres ejemplos de un clasificador binario subajustado, sobreajusta y bien ajustado en un conjunto de datos sintético.

Podemos concluir que el objetivo del *ML* supervisado es construir un modelo de aprendizaje de manera que las predicciones en un conjunto de datos nuevo sean lo más exactas y precisas posible. Para ello, se ha descrito como prioridad disminuir la tasa de error en el conjunto de test lo máximo posible hasta alcanzar el error Bayes. Este es un supuesto ideal ya que en casos reales no dispondremos de la distribución que genera los datos y, por tanto, también desconocemos el error Bayes. Sin embargo, para un problema de clasificación enfocarse en el error de test definido mediante la ecuación 2.3, el cual se basa en la función de pérdida 0-1 y pondera el error cometido en ambas clases con la misma importancia, puede no ser suficiente (Tharwat, 2021).

Supongamos un problema de clasificación binaria, donde tenemos una clase negativa y otra positiva que se representan formalmente mediante el conjunto $\mathcal{G} = \{-1, +1\}$. El rendimiento de un modelo puede resumirse en una matriz de confusión de $2x2$, como se muestra en la tabla 2.1. La matriz de confusión muestra cuántos elementos de cada clase son correctamente identificados por el modelo y cuántos son identificados de manera incorrecta. La matriz de confusión $2x2$ tiene 4 celdas que representan los siguientes casos:

- *Verdaderos Positivos* (*VP*): Estos son los elementos que son correctamente identificados como pertenecientes a la clase positiva.
- *Falsos Positivos* (*FP*): Estos son los elementos que son incorrectamente identificados como pertenecientes a la clase positiva.
- *Verdaderos Negativos* (*VN*): Estos son los elementos que son correctamente identificados como pertenecientes a la clase negativa.
- *Falsos Negativos* (*FN*): Estos son los elementos que son incorrectamente identificados como pertenecientes a la clase negativa.

La tasa de error considera igual de relevantes los *FN* y los *FP*. Sin embargo, la utilidad de esta medida puede depender de las condiciones específicas del problema que se está resolviendo. Un claro ejemplo donde esto es problemático es en el caso de que los datos estén muy desequilibrados, es decir, que una clase esté mucho menos presente que la otra. La evaluación en función de la tasa de error puede llevar a un modelo mal ajustado que solo tenga en cuenta una de las clases, pero donde se observe un error pequeño. La tasa de error no es más que un resumen de la matriz de confusión en un valor numérico teniendo en cuenta dos de los cuatro elementos disponibles. El complementario de este valor es conocido como exactitud y representa la tasa de acierto del modelo que produce la matriz de confusión (Goodall, 1967).

En general, las métricas de rendimiento resumen en un único valor numérico el rendimiento de un modelo de clasificación. Este valor es calculado a partir de alguno (en ocasiones incluso de todos ellos) de los cuatro elementos que conforman la matriz de confusión. Alguna de las métricas más importantes que se pueden obtener son: la precisión, la tasa de recuperación, la puntuación F_1 (Sasaki y Fellow, 2007) y el coeficiente de correlación de Matthews (*Mathews Correlation Coefficient* (*MCC*)) (Matthews, 1975).

La precisión se refiere a la proporción de resultados positivos identificados por el modelo que son realmente positivos, y se puede calcular como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos. La precisión mide cómo de fiable es el modelo en la identificación de resultados positivos. Por otro lado, la tasa de recuperación mide la capacidad del modelo para encontrar todos los resultados positivos, y se puede calcular como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos. La tasa de recuperación mide cuán completo es el modelo en la detección de casos positivos.

La puntuación F_1 se puede interpretar como un equilibrio entre la precisión y la tasa de recuperación. La clase positiva suele ser la clase de mayor interés, de ahí la importancia de una métrica de rendimiento que combine la precisión y la tasa de recuperación, ya que ambas priman el rendimiento para la correcta predicción de la clase positiva. La puntuación F_1 se computa como media armónica entre ambas métricas:

$$F_1 = 2 \cdot \frac{\text{precisión} \cdot \text{tasa de recuperación}}{\text{precisión} + \text{tasa de recuperación}} = \frac{2VP}{2VP + FP + FN} \tag{2.7}$$

En resumen, la precisión mide la exactitud de los resultados positivos identificados por el modelo, mientras que la tasa de recuperación mide la capacidad del modelo para encontrar todos los resultados positivos. Ambas métricas son importantes en diferentes situaciones, por lo que la F_1 se utiliza para equilibrar ambas y obtener una métrica de rendimiento más completa. Una puntuación F_1 de 1 indica un equilibrio

perfecto entre la precisión y la tasa de recuperación, mientras que una puntuación menor indica que alguna de las dos no alcanzan sus valores máximos.

La métrica F_1 puede no ser adecuada en situaciones donde las clases están desequilibradas debido a que ignora los Verdaderos Negativos (Powers, 2011). Al dar igual peso a la tasa de recuperación y la precisión, esta métrica puede no reflejar adecuadamente la realidad cuando los errores de clasificación tienen costos diferentes (Hand y Christen, 2018). En tales casos, la F_1 puede resultar engañosa y no brindar una visión completa tanto de la predicción del clasificador como de la predicción de la clase verdadera.

Otra métrica de rendimiento ampliamente utilizada en problemas de clasificación binaria es *MCC*, conocida en estadística como el coeficiente phi (Yule, 1912). El *MCC* es un valor compuesto que toma en cuenta todos los valores dentro de la matriz de confusión, y se calcula de la siguiente manera:

$$MCC = \frac{VP \cdot VN - FP \cdot FN}{\sqrt{(VP + FP)(VP + FN)(TN + FP)(VP + FN)}} \tag{2.8}$$

Esta métrica proporciona una visión más completa del rendimiento de un modelo de clasificación binaria que la precisión, la tasa de recuperación o la F_1, en situaciones en las que existe un desequilibrio entre las clases. Esta métrica toma valores entre -1 y 1, donde 1 indica una predicción perfecta, -1 indica una predicción totalmente incorrecta, y 0 indica una predicción aleatoria.

El *MCC* evalúa la correlación entre las predicciones del clasificador y las etiquetas reales. Es una métrica simétrica que refleja tanto la eficacia de la predicción del clasificador como la precisión de la clase verdadera al predecir la predicción del clasificador. A diferencia de la F_1, el *MCC* es más veraz e informativo en situaciones donde las clases están desequilibradas (Chicco y Jurman, 2020). No obstante, en los casos donde el problema fundamental sea clasificar de manera correcta la clase positiva como detección de fraude o diagnosis médica la F_1 sigue siendo una métrica adecuada y precisa.

En conclusión, las métricas de rendimiento exactitud, F_1 y *MCC* son tres métricas comunes para evaluar modelos de clasificación en *ML*. La exactitud mide el número de predicciones correctas, pero no es adecuada para problemas con distribución desequilibrada de clases. La F_1 es una métrica que equilibra precisión y tasa de recuperación, y es útil en problemas que requieren un equilibrio entre verdaderos positivos y falsos positivos. La *MCC* es una métrica que considera tanto precisión y tasa de recuperación como verdaderos y falsos positivos y negativos, y es especialmente útil en problemas con distribución desequilibrada de clases.

Ahora que tenemos una comprensión clara de qué es un clasificador formalmen-

te, qué fases influyen en su construcción y validación, además de las peculiaridades de su evaluación, es hora de introducir los diferentes algoritmos de clasificación que serán relevantes en los próximos capítulos. A continuación se realiza una una revisión breve, así como el análisis de ciertos enfoques de implementación, de los k vecinos más cercanos (*k-Nearest Neighbour* (*kNN*)), árboles de decisión (*DT*) y máquinas de vector soporte (*SVM*).

2.1. *K-Nearest Neighbors*

El *kNN* es un algoritmo de aprendizaje supervisado basado en memoria (Fix y Hodges, 1951). La idea detrás de *kNN* es utilizar los datos existentes para predecir el valor de nuevas observaciones. El algoritmo clasifica los nuevos puntos según la mayoría de las etiquetas de clase de las k observaciones más cercanas en el conjunto de entrenamiento. La definición de cercanía se refiere a una función d que determina la distancia entre dos observaciones x e y. Para que una función sea considerada una distancia, tiene que cumplir la siguientes condiciones (Cover y Thomas, 2006):

- No negatividad: La distancia entre dos puntos nunca puede ser negativa. Formalmente, para dos puntos x e y, la distancia $d(x, y)$ debe ser mayor o igual a cero: $d(x, y) \geq 0$.
- Identidad de los indiscernibles: La distancia entre un punto y sí mismo es cero. Formalmente, para un punto x, la distancia $d(x, x)$ debe ser igual a cero: $d(x, x) = 0$.
- Simetría: La distancia entre dos puntos es la misma independientemente del orden en que se tomen los puntos. Formalmente, para dos puntos x e y, la distancia $d(x, y)$ es igual a la distancia $d(y, x)$: $d(x, y) = d(y, x)$.
- Desigualdad triangular: La distancia entre dos puntos directamente no puede ser mayor que la distancia a través de un tercer punto. Formalmente, para tres puntos x, y e z, la distancia $d(x, z)$ es menor o igual que la suma de las distancias $d(x, y)$ y $d(y, z)$: $d(x, z) \leq d(x, y) + d(y, z)$.

Estas condiciones se conocen como las propiedades métricas y son necesarias para garantizar la consistencia de la medición de la distancia. La distancia Euclídea es la medida de distancia más comúnmente empleada en *kNN*, pero también se pueden utilizar otras medidas de distancia, como la distancia de Manhattan o la distancia de Chebyshev.

La distancia Euclídea $d(x, y)$ entre dos puntos de datos x e y se define como:

$$d(x,y) = \sqrt{\sum_{i=1}^{r}(x_i - y_i)^2} \tag{2.9}$$

donde r es el número de características y x_i e y_i son las i-ésimas características de las observaciones x e y, respectivamente.

La distancia de Manhattan $d(x,y)$ se define como:

$$d(x,y) = \sum_{i=1}^{r}|x_i - y_i| \tag{2.10}$$

y la distancia de Chebyshev $d(x,y)$ se define como:

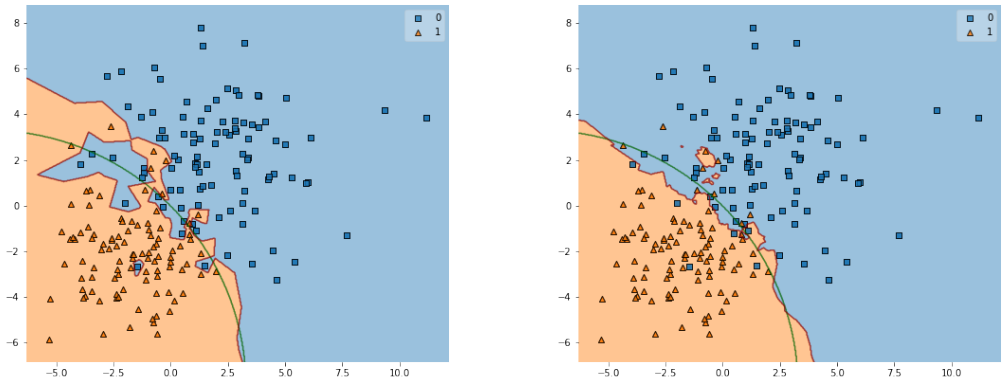
$$d(x,y) = \max_{i=1}^{r}|x_i - y_i| \tag{2.11}$$

Tanto la distancia Euclídea como la distancia Manhattan son una particularización de la distancia de Minkowski. Esta se define como la raíz p-ésima de la suma de las diferencias entre las coordenadas de los puntos elevadas al exponente p. La fórmula para la distancia de Minkowski entre dos puntos x e y en un espacio con d dimensiones y un parámetro p se puede escribir como:

$$d(x,y) = \left(\sum_{i=1}^{r}|x_i - y_i|^p\right)^{\frac{1}{p}} \tag{2.12}$$

donde x_i e y_i son las coordenadas de las observaciones x e y en la dimensión i, respectivamente, r es el número de dimensiones, y p es el parámetro que controla la forma de la distancia de Minkowski. Cuando $p = 1$, la distancia de Minkowski se reduce a la distancia de Manhattan. Cuando $p = 2$, la distancia de Minkowski se reduce a la distancia Euclídea. La distancia de Minkowski es una medida más general que permite controlar la importancia de las diferencias en cada dimensión. Definida la función de distancia y el número de vecinos k, el *kNN* como clasificador funciona de la siguiente manera:

1. Calcular la distancia entre la observación objetivo y cada punto de datos en el conjunto de entrenamiento utilizando la distancia definida.
2. Seleccionar los k ejemplos de entrenamiento más cercanos a la observación objetivo basándose en la distancia calculada en el paso anterior.
3. Asignar la etiqueta de clase más común entre los k ejemplos.
4. Devolver la etiqueta de clase predicha para la observación objetivo.



(a) *Un vecino.* **(b)** *Cinco vecinos.*

Figura 2.5: *Un conjunto de datos simulados formado por 100 puntos para cada una de las clases, indicadas en azul y naranja. El fondo de color naranja indica la región en la que se asignará a la clase naranja, y el fondo azul indica la región en la que se asignará a la clase azul empleando un clasificador* kNN. *Además, la frontera de decisión para el* kNN *está marcada en rojo, y la frontera de decisión de clasificador Bayes está marcada en verde.*

Nótese que la elección del valor k es importante ya que afecta la precisión del algoritmo. Si k es demasiado grande, el algoritmo puede perder detalles importantes en los datos. Si k es demasiado pequeño, el algoritmo puede ser muy sensible al ruido en el conjunto de entrenamiento. No obstante, para $k = 1$ existe un resultado muy importante en la literatura, en el que se asegura que la tasa de error para el $1-NN$ nunca será más de dos veces el error Bayes (Cover y Hart, 1967). Aunque esta afirmación tiene ciertas restricciones, es una buena aproximación de la cota de error presente en los datos. En la figura 2.5 se muestra un ejemplo de la frontera de decisión que genera el *kNN* con uno y cinco vecinos, valores típicos para k. El algoritmo *kNN* es sensible al ruido y los valores atípicos, y la elección adecuada de k es importante para un buen rendimiento. Además, para grandes conjuntos de datos, el entrenamiento puede ser computacionalmente costoso y en situaciones con datos de alta dimensionalidad y muchas clases, el algoritmo puede ser menos preciso. Sin embargo, tiene la ventaja de ser simple y fácil de implementar y no requiere supuestos sobre la distribución de los datos, siendo adecuado para una amplia gama de problemas de clasificación.

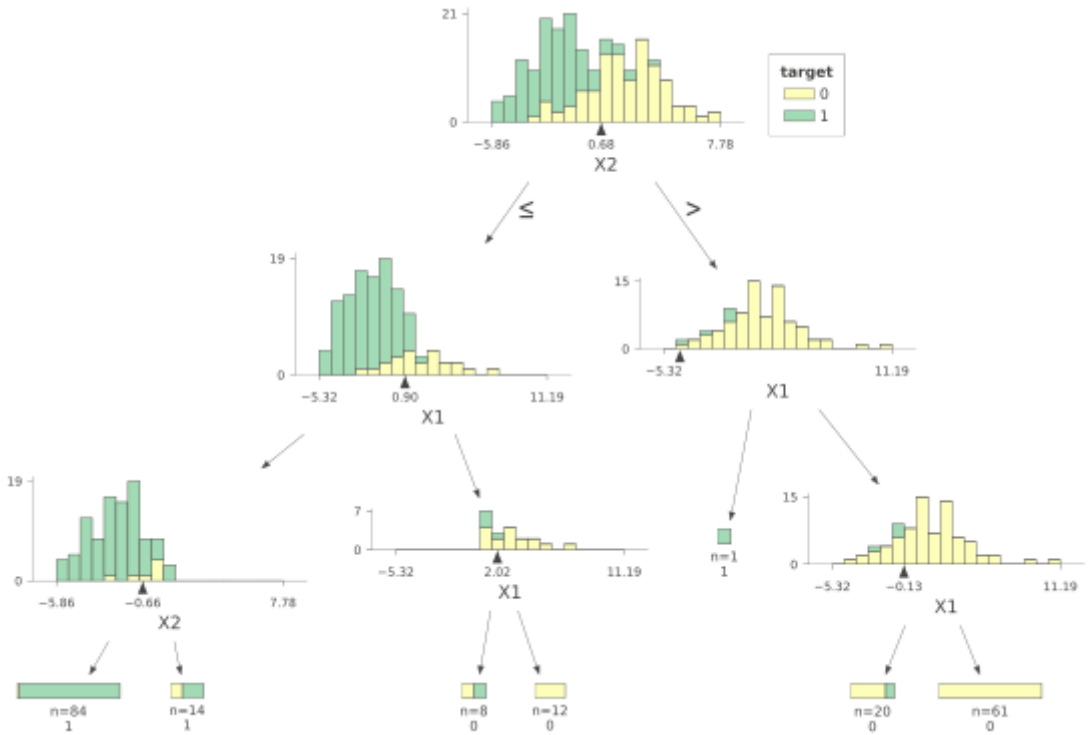


Figura 2.6: *Estructura de un árbol de decisión construido con datos simulados en dos dimensiones. Los colores amarillo y verde representan cada una de las clases. En los nodos finales se muestra el número de observaciones y la clase asignada.*

2.2. *Decision Tree*

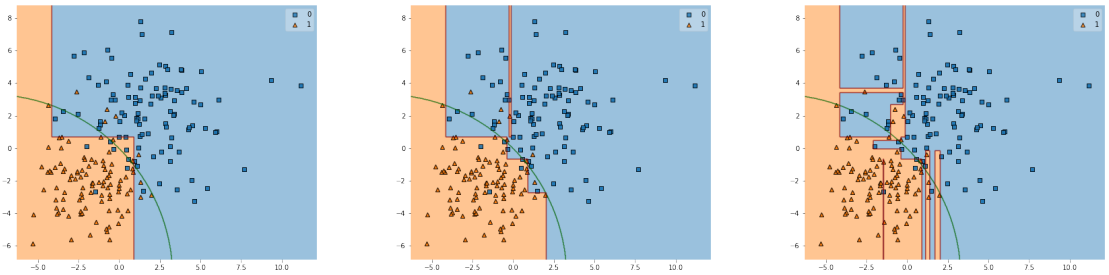
El *DT* es un algoritmo de aprendizaje supervisado basado en criterios de información y reducción de la incertidumbre. La clave de los árboles de decisión reside en el particionamiento recursivo del espacio de características en regiones rectangulares, más pequeñas y homogéneas, de forma que la incertidumbre se reduzca en esas regiones. Para determinar la mejor partición en cada nodo, se utilizan medidas de incertidumbre como la entropía o el índice Gini. Para un conjunto de datos $\mathcal{R}$ con k clases, la entropía, $H(\mathcal{R})$, se define como:

$$H(\mathcal{R}) = -\sum_{i=1}^{k} p_i \log_2(p_i) \tag{2.13}$$

y el índice de Gini, $G(\mathcal{R})$, se define de la siguiente manera:

$$G(\mathcal{R}) = \sum_{i=1}^{k} p_i(1 - p_i) \tag{2.14}$$

para ambas medidas de incertidumbre, p_i es la proporción de observaciones per-



(a) *Profundidad tres.* **(b)** *Profundidad cinco.* **(c)** *Profundidad diez.*

Figura 2.7: *Un conjunto de datos simulados formado por 100 puntos para cada una de las clases, indicadas en azul y naranja. El fondo de color naranja indica la región en la que se asignará a la clase naranja, y el fondo azul indica la región en la que se asignará a la clase azul empleando un clasificador* DT. *Además, la frontera de decisión para el* DT *está marcada en rojo, y la frontera de decisión de clasificador Bayes está marcada en verde.*

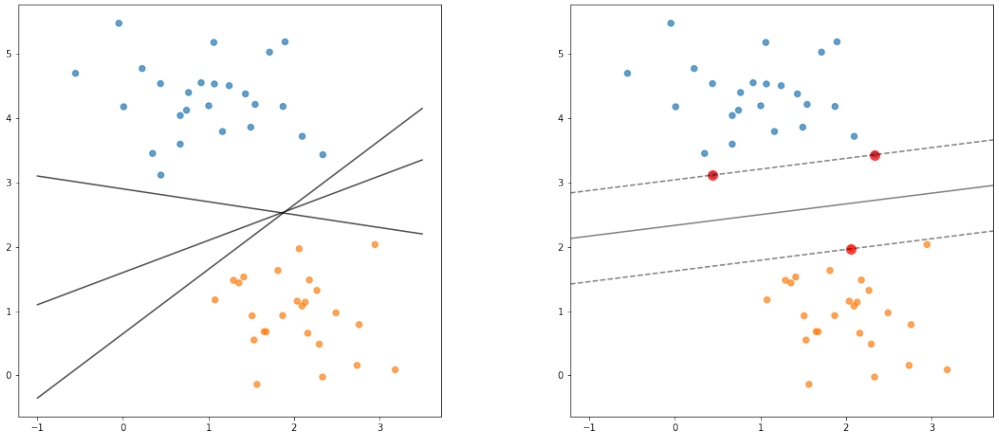
tenecientes a la clase i.

El proceso de construcción del árbol se inicia con una característica que se considera la mejor para separar los datos en dos regiones. Luego, para cada una de las regiones se vuelve a dividir. Para ello, se evalúan el resto de características y se selecciona la que produce la mejor división. Este proceso se repite para cada grupo resultante hasta que se alcanza un criterio de parada, habitualmente la profundidad máxima del árbol. En la figura 2.6 se representa un árbol que divide el espacio en siete regiones. Primero se construye el nodo inicial mediante $X2$, la división aquí proporciona dos ramas a otros dos nodos intermedios que se vuelven a dividir en función de la variable $X1$. Este proceso se repite hasta llegar a los siete nodos finales u hojas del árbol. Estas hojas tienen asignada la etiqueta de la clase mayoritaria.

Para seleccionar el mejor corte para las variables de los nodos intermedios se calcula la reducción de la incertidumbre de cada nodo. Supongamos que una de las posibles divisiones de la variable X_j es $X_j \leq c$ y $X_j > c$, donde c es algún valor de X_j. Sea también $\mathcal{R}$ la región del espacio que buscamos dividir en $\mathcal{R}_L$ y $\mathcal{R}_R$ las dos regiones resultantes. El mejor valor c para dividir X_j será aquel que maximice la ganancia de información, definida como la reducción de alguna medida incertidumbre I:

$$\Delta I(c, \mathcal{R}) = i(\mathcal{R}) - \frac{card(\mathcal{R}_R)}{card(\mathcal{R})} \cdot i(\mathcal{R}_R) - \frac{card(\mathcal{R}_L)}{card(\mathcal{R})} \cdot i(\mathcal{R}_L) \tag{2.15}$$

Una vez construido el árbol, se puede usar para hacer predicciones para nuevos casos. Para hacer una predicción, se sigue el camino a través del árbol, desde la raíz,



(a) *Varios hiperplanos de separación.*

(b) *Hiperplano de máxima separación. Vectores soporte marcados en rojo.*

Figura 2.8: *Ejemplos de los posibles hiperplanos separadores para un conjunto de datos linealmente separable, y la solución arrojada por una* SVM.

siguiendo las decisiones basadas en las características y se llega a una hoja. La predicción se realiza a partir de la etiqueta asignada a la hoja correspondiente. En la figura 2.7 se muestra la frontera de decisión de un árbol de distintas profundidades, comparando con la frontera ofrecida por el clasificador Bayes para datos simulados.

En resumen, el algoritmo de un árbol de decisión funciona dividiendo el espacio de características en regiones más pequeñas y haciendo una predicción basada en la variable objetivo en cada región. Esto los hace fáciles de entender y explicar. Además, no son sensibles a valores atípicos ni datos faltantes y pueden manejar tanto datos categóricos como numéricos. Sin embargo, son modelos inestables que tienden al sobreajuste. En capítulos posteriores veremos que este hecho puede suponer una gran ventaja para el aprendizaje combinado.

2.3. *Support Vector Machines*

Las *SVM* fueron concebidos para problemas de clasificación binaria que eran separables linealmente (es decir, las dos clases pueden ser separadas por un hiperplano) (Vapnik, 1982). Imaginando un caso sencillo en dos dimensiones donde tenemos dos conjuntos de puntos linealmente separables, existen infinitas líneas rectas que los separan, como se muestra en la figura 2.8a. Estas líneas rectas se generalizan como hiperplanos de separación, el objetivo de la *SVM* es encontrar el hiperplano de máxima separación entre las observaciones de cada una de las clases. En la figu-

ra 2.8b se muestra este hiperplano de máxima separación, además se marcan unos puntos particulares que se denominan vectores soporte. Estos puntos son la clave en la solución de una *SVM*, dado que sólo la posición de estos puntos importa, otra observación más allá del margen que definen los vectores soporte, y que esté bien clasificado, no modifica el ajuste del modelo.El motivo es que estos puntos no influyen en la función de pérdida empleada para ajustar el modelo, por lo que su ubicación y cantidad no son relevantes siempre y cuando no atraviesen el margen.

Dado un conjunto de aprendizaje $\mathcal{L} = \{(\boldsymbol{x_i}, y_i) : i = 1, 2, \ldots, n\}$, sobre el par $(\boldsymbol{X}, Y)$ donde $\boldsymbol{X} \in \mathbb{R}^r$ e $Y \in \{-1, 1\}$. Este conjunto puede ser separado linealmente por un hiperplano de la forma:

$$\{x : f(x) = \boldsymbol{\omega}^T \boldsymbol{x} + b = 0\} \tag{2.16}$$

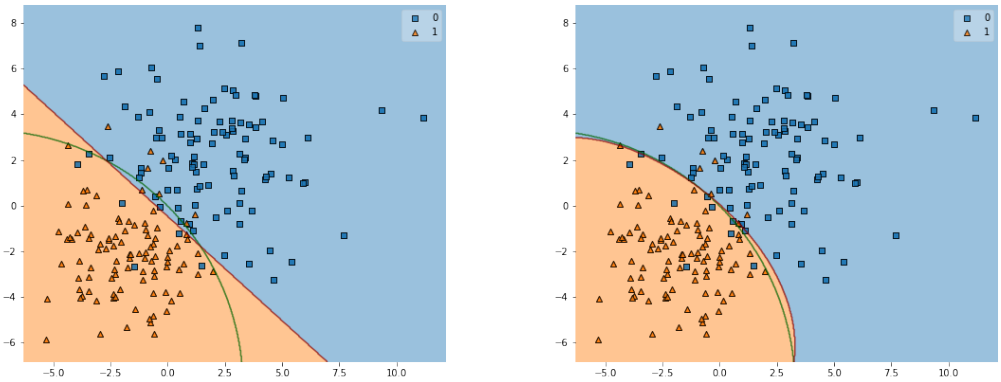
done $\boldsymbol{\omega}$ es el vector de pesos, y b es el sesgo del hiperplano. La solución de la *SVM* es la que garantiza que este hiperplano y su observación más cercana es máxima, es decir, el hiperplano de máxima separación. Este hiperplano se encuentra al resolver el siguiente problema de optimización:

$$\begin{aligned} \underset{\boldsymbol{\omega}}{\text{minimizar}} \quad & \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} \\ \text{sujeto a} \quad & y_i(\boldsymbol{\omega}^T\boldsymbol{x} + b) \geq 1 \\ & i = 1, \ldots, n \end{aligned} \tag{2.17}$$

Sin embargo, en la mayoría de los problemas de clasificación binaria de *ML*, no es posible separar correctamente las clases linealmente. En estos casos, el problema de optimización original para entrenar una *SVM* se relaja para permitir instancias mal clasificadas. Por lo tanto, esta relajación produce soluciones de margen suave. Para hacer esto, introducimos el concepto de una variable de holgura, ξ_i y un parámetro de regularización. Este parámetro de regularización toma la forma de una constante de ajuste que controla el tamaño de las variables de holgura y equilibra los dos términos en la función de minimización. Ampliando el sistema (2.3) con estos dos conceptos, el nuevo problema de optimización primal de la *SVM* queda como sigue:

$$\begin{aligned} \underset{\boldsymbol{\omega}, b, \boldsymbol{\xi}}{\text{minimizar}} \quad & \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + C\sum_{i=1}^{t}\xi_i \\ \text{sujeto a} \quad & y_i(\boldsymbol{\omega}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, n \end{aligned} \tag{2.18}$$

donde $C > 0$ es el parámetro de regularización (si $C = \infty$, la *SVM* proporciona una solución de margen duro). No obstante, con esta reformulación sólo podemos



(a) *Kernel lineal.* **(b)** *Kernel no lineal.*

Figura 2.9: *Un conjunto de datos simulados formado por 100 puntos para cada una de las clases, indicadas en azul y naranja. El fondo de color naranja indica la región en la que se asignará a la clase naranja, y el fondo azul indica la región en la que se asignará a la clase azul empleando un clasificador* SVM. *Además, la frontera de decisión para la* SVM *está marcada en rojo, y la frontera de decisión de clasificador Bayes está marcada en verde.*

abordar casos donde las clases estén solapadas, pero no se resuelve la no linealidad de los datos. Para ello, supongamos una transformación no lineal $\phi : \mathbb{R}^r \mapsto \mathbb{H}$, donde $\mathbb{H}$ puede ser de muy alta dimensión, incluso de dimensión infinita. Por tanto, sustituyendo los x_i por sus transformaciones $\phi(x_i)$ en 2.3, obtenemos la *SVM* para casos no lineales:

$$\begin{aligned} \underset{\omega,b,\xi}{\text{minimizar}} \quad & \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{t}\xi_i \\ \text{sujeto a} \quad & y_i(\omega^T\phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1,\ldots,n \end{aligned} \tag{2.19}$$

donde $\phi(x_i)$ mapea x_i en un espacio de alta dimensión. Generalmente consideramos que es un espacio de Hilbert de funciones reales en $\mathbb{R}$ con producto escalar y norma.

Aunque las *SVM* de margen suave permiten problemas no separables linealmente, no pueden detectar relaciones no lineales. Para superar esta limitación, las *SVM* se generalizaron para resolver problemas no lineales (Cortes y Vapnik, 1995). En la figura 2.9 se muestran las fronteras de decisión generadas para una *SVM* lineal y una *SVM* no lineal con kernel gaussiano. Además, se puede apreciar la flexibilidad que otorga el uso de kernels a las *SVM*, de manera que se puede aproximar muy bien la

frontera Bayes.

En la generalización al caso no lineal, se utiliza la técnica del multiplicador de Lagrange para obtener la solución del problema dual de Wolfe. Por lo tanto, el método solo tiene que calcular el vector $\boldsymbol{\alpha}$ de multiplicadores de Lagrange, que es la solución del problema dual. La principal ventaja de este enfoque es la introducción de la función kernel. Esto permite capturar relaciones no lineales en los datos, sin construir la función $\phi(\cdot)$. El problema dual se establece de la siguiente:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimizar}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha} - 1_n^T \boldsymbol{\alpha} \\ \text{sujeto a} \quad & \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n, \end{aligned} \tag{2.20}$$

donde H es una matriz cuadrada de orden n y $H_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, tales que $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i^T)\phi(\boldsymbol{x}_j)$ es la función kernel. La solución del problema primal se puede calcular usando $\hat{\boldsymbol{\alpha}}$ que resuelve (2.3) de la siguiente manera:

$$\hat{\boldsymbol{\omega}} = \sum_{i \in sv} y_i \alpha_i \phi(\boldsymbol{x_i}) \tag{2.21}$$

y el clasificador se define como:

$$sign(\boldsymbol{\omega}^T \phi(\boldsymbol{x}) + b) = sign\left(\sum_{i \in sv} y_i \alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b\right) \tag{2.22}$$

donde el conjunto sv contiene aquellas instancias tales que $\alpha_i \neq 0$ en la solución dual y se conocen como vectores soporte.

Los métodos de optimización por programación cuadrática (QP) utilizados para resolver el problema dual de las *SVM* requieren calcular la matriz completa del kernel. Esto se vuelve inmanejable desde un punto de vista computacional a medida que aumenta el número de observaciones. Se han desarrollado varios algoritmos que dividen el problema en partes activas e inactivas para solucionar esta limitación. La parte activa se refiere a menudo como el conjunto de trabajo o conjunto activo. Existen varias estrategias para calcular el "conjunto de trabajo": fragmentación de tamaño variable (Vapnik, 1982), descomposición de tamaño fijo (Osuna *et al.*, 1997) y optimización mínima secuencial (*Sequential Minimal Optimization* (*SMO*)), que es una versión extrema de la descomposición en la que solo se seleccionan dos instancias en cada iteración (Platt, 1998). De hecho, *SMO* sigue siendo el núcleo de los métodos de entrenamiento más presentes para algoritmos de resolución más rápidos de *SVM* (Torres-Barrán *et al.*, 2021).

Existen muchas soluciones de software para resolver el problema de optimización de la *SVM* basados en los métodos mencionados anteriormente. La implementación de software *SVM* más popular fue SVM^{light} (Joachims, 1998), basada en el método de descomposición. Sin embargo, en los últimos años, *LibSVM* ha tomado el liderazgo gracias a sus implementaciones en los lenguajes de programación más utilizados en *ML*, como R o Python. *LibSVM* se basa en el *SMO* (Chang y Lin, 2011), con nuevas heurísticas para seleccionar el *conjunto de trabajo* (Fan *et al.*, 2005) y acelerar así la búsqueda de los vectores soporte que definen una *SVM*.

Capítulo 3

Aprendizaje incremental

Hasta ahora hemos tratado los modelos predictivos como algo estático, dependientes de un ajuste y una evaluación permanentes e inamovibles. No obstante, los modelos predictivos de *ML* están presentes en una gran variedad de dominios, como la energía, el turismo, la economía y los sectores ambientales. En estos casos, los modelos suelen utilizarse durante un período de tiempo. Es posible que estos modelos necesiten ser reajustados de acuerdo a los nuevos datos. Por lo tanto, pueden tener que lidiar con patrones que no existían en el entrenamiento original (Gama *et al.*, 2014). En algunos problemas este tipo de cambios en los datos pueden estar inducidos por la propia presencia de un modelo predictivo, donde las decisiones tomadas en función de este afectan a los datos que se recogerán en el futuro. Por ejemplo, consideremos un modelo de *ML* para la atención médica que estima la probabilidad de morir en una cirugía. En este contexto, los pacientes con alta probabilidad de morir probablemente no se someterán a la cirugía. En consecuencia, la distribución de los datos cambiará con el tiempo, lo que podría reducir la calidad del modelo si subestima la información sobre estos pacientes de alto riesgo. Por lo tanto, el modelo de *ML* debería ser reajustado con los nuevos pacientes sin subestimar a los anteriores (es decir, a los pacientes que no se someterán a la cirugía debido a la toma de decisiones basada en el modelo).

Existen diferentes estrategias que se pueden utilizar para reajustar modelos, como el aprendizaje en línea con micro lotes o datos en *streaming* y el reentrenamiento por lotes programado en el tiempo (Bifet *et al.*, 2018). Aunque la actualización de los algoritmos en línea es rápida, no se usan comúnmente ya que el error del modelo suele ser mayor que cuando se utilizan otras aproximaciones (Gama, 2012), estos suelen ser bastante útiles en los dominios donde los patrones son muy volátiles y se necesita un reajuste casi instantáneo. Es importante tener en cuenta que, en este caso, las observaciones se utilizan solo una vez y luego se descartan. Por otro lado, la actualización del modelo utilizando todos los datos podría proporcionar los mejores

resultados. Sin embargo, esto se vuelve inviable a medida que aumenta el número de observaciones. La estrategia de aprendizaje incremental más común es un equilibrio entre estos dos enfoques, donde el modelo se actualiza con micro lotes de datos. Estas técnicas se aplican en soluciones de *ML* en *streaming*, un área específica del *ML* que se ocupa de problemas donde los datos son altamente volátiles y requieren, si es que se necesita, poca ingeniería de características. En otros problemas, el modelo se entrena primero con el conjunto de datos inicial y luego se actualiza periódicamente con los nuevos datos. La frecuencia de esta actualización periódica depende del dominio de aplicación.

Se han propuesto varias aproximaciones a estos problemas en el contexto de las *SVM*s, debido a que son uno de los métodos más utilizados para problemas de clasificación y regresión debido a sus sólidas bases matemáticas y capacidad de generalización. Como resultado, las *SVM*s se han convertido en una opción predeterminada en la mayoría de los problemas de *ML* (Bouaafia *et al.*, 2020). La fuerza que proporcionan también plantea un problema en su implementación, ya que encontrar una solución válida para una *SVM* requiere resolver un problema de optimización cuadrática. Además, el costo computacional requerido para su resolución aumenta en presencia de kernels no lineales. Por lo tanto, se han desarrollado algoritmos para resolver el problema de la *SVM* más eficientes. A pesar de que estos algoritmos no garantizan la convergencia local, obtienen la solución global óptima en tiempos de entrenamiento adecuados.

Una aproximación simple para el reentrenamiento de *SVM*s es inicializar el algoritmo en un estado que esté más cerca de la solución final. Esta aproximación, llamada *alpha seeding* (DeCoste y Wagstaff, 2000), es la base de métodos más complejos como los métodos incrementales (Laskov *et al.*, 2006). Estos métodos insertan la nueva información en el estado anterior, de manera similar a la técnica de *shrinking*. En las *SVM*s en línea, se requiere un enfoque diferente para resolver el problema cuadrático ya que utilizan cada instancia solo una vez (Shalev-Shwartz *et al.*, 2011) o, en algunos casos, lotes de datos procesado una vez pero al mismo tiempo (Gu *et al.*, 2018). Por lo tanto, en todas estas aproximaciones se modifica el método para seleccionar los candidatos a vectores soporte (por ejemplo, introduciendo regularización en la función objetivo o muestreo selectivo para el conjunto de trabajo (Loosli *et al.*, 2007)).

3.1. Aprendizaje incremental para SVM

El desarrollo de métodos iterativos para resolver *SVM*, como *SMO*, dio lugar a una nueva aproximación para el reentrenamiento de la *SVM*. Por ejemplo, *alpha seeding* es un método que utiliza la solución actual del problema dual (es decir, el vector de

alfas) como el estado inicial para el siguiente reentrenamiento (DeCoste y Wagstaff, 2000). El método de *alpha seeding* ha sido utilizado en otros problemas, como la descomposición de *SVM*s lineales (Kao *et al.*, 2004) y la optimización de hiperparámetros con validación cruzada de k particiones (Lee *et al.*, 2004). En este último caso, el problema se plantea como un problema de reentrenamiento, donde el proceso se optimiza utilizando las soluciones de las particiones anteriores como estado inicial en las siguientes particiones. Además, se han construido nuevas versiones de *alpha seeding* para una búsqueda eficiente de hiperparámetros (Wen *et al.*, 2017).

Las aproximaciones actuales basadas en *alpha seeding* tienen dos principales inconvenientes. En primer lugar, requieren modificar la implementación de *SVM* para permitir establecer un estado inicial. Por lo tanto, debería alcanzarse un amplio consenso sobre los beneficios de esta aproximación antes de incluirla en las implementaciones de *SVM* más comunes. En segundo lugar, las implementaciones actuales de *SVM* inicializan el gradiente de la función de pérdida al comienzo del algoritmo. Esta inicialización es más costosa en *alpha seeding* ya que el método no puede asumir que todos los alfas son inicialmente cero. Por lo tanto, en algunos casos, el coste de inicialización de *alpha seeding* supera la reducción del tiempo requerido en el entrenamiento. En cuanto al aprendizaje incremental, el *alpha seeding* ha sido utilizado como base para varios métodos de reentrenamiento de *SVM*. La conversión de *alpha seeding* a una *SVM* incremental se ha centrado en dos áreas principales: cómo gestionar las observaciones anteriores y el muestreo selectivo.

En el abordaje para la gestión de datos previos, los métodos utilizan observaciones que representan los vectores soporte del entrenamiento anterior y descartan los demás. Además, aumentan el costo de la mala clasificación de los vectores soporte anteriores en relación con las nuevas observaciones (Syed *et al.*, 1999). Mantener la información sobre las condiciones *KKT* de los datos previamente vistos mejoró este enfoque (Cauwenberghs y Poggio, 2001). Esta noción también es adecuada para conjuntos secuenciales como el *boosting* (Kashef, 2021). Estas aproximaciones nos llevan a concluir que los vectores soporte no representan todo el estado de una *SVM*. Otro enfoque similar a *alpha seeding* es *LaSVM* (Bordes *et al.*, 2005), que consiste en una reorganización de *SMO* que también sirve como algoritmo en línea. Por lo tanto, cada par de muestras se considera solo una vez y afirma lograr una solución comparable a un algoritmo de resolución de *SVM* tradicional. En una configuración de entrenamiento por lotes, más iteraciones pueden aumentar la calidad de la solución mediante el uso de recursos computacionales adicionales.

En muestreo selectivo, la implementación de *SVM* suele modificarse para personalizar la función de coste e incluir una estrategia de muestreo selectivo que acelere la convergencia y mejore el rendimiento. Existen diferentes maneras de realizar muestreo selectivo en algoritmos de *SVM* incremental. Varios de los métodos cuen-

tan con sólidas bases matemáticas sobre las que apoyar su construcción y resultados. Por ejemplo, seleccionando el subconjunto de datos que maximiza la diferencia entre la función de pérdida actual y una estimación de la nueva función de pérdida (Yang *et al.*, 2007), o evaluando las condiciones *KKT* en cada nuevo punto (Fine y Scheinberg, 2002). Por otro lado, otros métodos dependen de heurísticas en el muestreo. En este contexto, se han propuesto heurísticas basadas en la distancia entre las muestras dentro del espacio de características del kernel (Ma *et al.*, 2017), y la distancia entre las muestras y el hiperplano óptimo (Loosli *et al.*, 2007). Un hallazgo relevante en la última heurística es que las instancias más cercanas al hiperplano son mejores para estimar la nueva frontera de decisión.

Un enfoque muy conocido para resolver la *SVM* mediante un muestreo aleatorio es el método de Nyström. Este método se basa en la selección aleatoria para aproximar el problema original y reducir así el tamaño del problema. La calidad de la solución depende de la aproximación realizada. Se han estudiado muchos mecanismos de muestreo para el método de Nyström (Kumar *et al.*, 2012). Aunque en la mayoría de los casos existe una brecha entre la solución óptima y los métodos basados en Nyström, también se ha estudiado el aprendizaje en línea con técnicas de aproximación del kernel, lo que ofrece mejoras en la predicción (Lu *et al.*, 2016). Otra forma de reducción de dimensiones es la *SVM* de menores cuadrados principales, que permite actualizaciones rápidas en tiempo real para flujos de datos (Artemiou *et al.*, 2021). En particular, para conjuntos de datos desequilibrados de gran escala, la reducción de la matriz kernel lleva a reducciones significativas en el tiempo de entrenamiento (Richhariya y Tanveer, 2020).

Como hemos mencionado anteriormente, la necesidad de modificar la implementación de *SVM* implica una limitación en la adopción de estas técnicas. Además, en algunos casos, estas técnicas no pueden proporcionar una precisión similar al entrenamiento con el conjunto de datos completo. También, la idea de seleccionar instancias relevantes más allá de los vectores soporte no es única para *SVM* incremental. Las técnicas de aprendizaje activo (como el muestreo selectivo) se centran en reducir el número de instancias en el entrenamiento de *SVM*s (D'Addabbo y Maglietta, 2015). Aunque las técnicas de reducción de dimensiones no son una solución universal para el entrenamiento de *SVM*s, son una herramienta valiosa en el arsenal del aprendizaje incremental. La elección de la técnica adecuada dependerá del conjunto de datos específico y de los requisitos de precisión y tiempo de entrenamiento.

Capítulo 4

Subconjuntos soporte

En este capítulo se propone un metodología de aprendizaje incremental para *SVM* basado en un nuevo concepto, el subconjunto soporte. Un subconjunto soporte es un subconjunto del conjunto de aprendizaje, de manera que al entrenar con este subconjunto y otro conjunto de datos nuevo, no visto por el modelo, la función de decisión generada es la misma que si empleáramos el conjunto de aprendizaje al completo. El subconjunto soporte amplía la idea de utilizar un subconjunto de observaciones relevantes (Cauwenberghs y Poggio, 2001), pero garantizando la equivalencia de la frontera de decisión resultante al entrenar con todos los datos. Esta nueva metodología de reentrenamiento también incluye *alpha seeding* y muestreo selectivo. Sin embargo, no incurre en la sobrecarga de inicialización del gradiente porque solo considera el subconjunto soporte y los nuevos datos.

4.1. Subconjuntos soporte

En esta sección se introduce la noción de subconjuntos soporte y subconjunto soporte mínimo, que proporcionan la muestra representativa mínima para un reentrenamiento efectivo. Además, se desarrollan una serie de resultados teóricos que garantizan el buen funcionamiento de la metodología de reentrenamiento, basada en la estimación de estos conjuntos, que se presentará en la siguiente sección.

Sea $\mathcal{L} = \{(\boldsymbol{x_i}, y_i) : i = 1, 2, \ldots, r\}$ el conjunto de entrenamiento tal que $\boldsymbol{x_i} \in \mathbb{R}^n$ y $y_i \in \mathcal{G}$, el conjunto de etiquetas. Sea $\mathcal{L}'$ un nuevo conjunto de datos $\mathcal{L}' = \{(\boldsymbol{x'_i}, y'_i) : i = 1, 2, \ldots, l\}$ tal que $\boldsymbol{x'_i} \in \mathbb{R}^n$ y $y'_i \in \mathcal{G}$, donde $\mathcal{L}' \nsubseteq \mathcal{L}$. Sea $f(\cdot \mid m, \mathcal{L})$ la función de decisión inducida por un modelo m de *ML*, entrenado en el conjunto $\mathcal{L}$. Refiriéndonos a un modelo de *ML* definido como un algoritmo de aprendizaje con hiperparámetros fijos. Las funciones de decisión $f^1 : \mathbb{R}^n \longrightarrow \mathcal{G}$ y $f^2 : \mathbb{R}^n \longrightarrow \mathcal{G}$ son equivalentes si $f^1(\boldsymbol{x}) = f^2(\boldsymbol{x})$, $\forall \boldsymbol{x} \in R$ donde $R \subseteq \mathbb{R}^n$.

Definición 1. *(Subconjunto soporte) Sea $\mathcal{L}_S \subseteq \mathcal{L}$ un subconjunto del conjunto de entrenamiento. Sea $f^1(\cdot \mid m, \mathcal{L} \cup \mathcal{L}')$ y $f^2(\cdot \mid m, \mathcal{L}_S \cup \mathcal{L}')$ dos funciones de decisión inducidas por el modelo de* ML *m. Entonces, $\mathcal{L}_S$ es un subconjunto soporte para la terna $\{\mathcal{L}, \mathcal{L}', f^1(\cdot \mid m)\}$, si y solo si, $f^1(\cdot \mid m)$ y $f^2(\cdot \mid m)$ son equivalentes en cualquier $R \subseteq \mathbb{R}^n$.*

Hay que tener en cuenta que $\mathcal{L}$ y $\mathcal{L}'$ no necesariamente son generados por la misma distribución de probabilidad.

Teorema 1. *Sea $\mathcal{X}_S$ el conjunto parcialmente ordenado de todos los subconjuntos soporte, entonces $\mathcal{X}_S$ es un conjunto finito tal que card$(\mathcal{X}_S) \geq 1$.*

Demostración. Sea $\mathcal{L}$ el conjunto de datos de entrenamiento, se sigue que $1 \leq \text{card}(\mathcal{L}_S) \leq r$, tal que $r \in \mathbb{N}$ y $\mathcal{X}_S = \{\{\mathcal{L}_S^i\}_{i=1}^m : \text{card}(\mathcal{L}_S^1) \leq \text{card}(\mathcal{L}_S^2) \leq \ldots \leq \text{card}(\mathcal{L}_S^m), \mathcal{L}_S^i \in \mathcal{P}(\mathcal{L}) \setminus \emptyset\}$. Donde $\mathcal{P}(\cdot)$ es el conjunto de las partes de un conjunto. Por lo tanto, es obvio que $\max(\text{card}(\mathcal{X}_S)) = \sum_{m=1}^{r} \binom{r}{m} = 2^r - 1$, por lo tanto existe una biyección $g : \mathcal{X}_S \longrightarrow \{1, 2, \ldots, t\}$, tal que $t = \text{card}(\mathcal{X}_S) \leq 2^r - 1$, con $t \in \mathbb{N}$.

Se ha demostrado que el conjunto $\mathcal{X}_S$ es finito, sin embargo el conjunto vacío también es finito. En base a la definición de subconjunto soporte, $\exists \mathcal{L}_S$ tal que $f_m\{\mathcal{L} \cup \mathcal{L}'\} \equiv f_m\{\mathcal{L}_S \cup \mathcal{L}'\}$, por lo tanto $\mathcal{L} = \mathcal{L}_S \in \mathcal{X}_S$ y $\text{card}(\mathcal{X}_S) \geq 1$.

□

Corolario 1.1. *El conjunto de todos los subconjuntos soporte, $\mathcal{X}_S$, es numerable.*

Definición 2. *(Subconjunto soporte mínimo) Sea $\mathcal{X}_S$ el conjunto de todos los subconjuntos soporte y $\mathcal{L}_S^1 \subset \mathcal{X}_S$. Si card$(\mathcal{L}_S^1) \leq$ card$(\mathcal{L}_S^i)$ $\forall \mathcal{L}_S^i \subset \mathcal{X}_S$, tal que $i = 1, 2, \ldots, m$. Entonces $\mathcal{L}_S^1$ es un subconjunto soporte mínimo.*

Lema 2. *El subconjunto soporte mínimo siempre existe, y no necesariamente es único.*

Demostración. La existencia es trivial debido al Teorema 1. La no unicidad se demuestra con el siguiente contraejemplo.

Sea $\mathcal{L} = \{((0,0),1),((2,-2),1),((2,2),-1),((-1,1),1)\}$ el conjunto de datos de entrenamiento y sea $\mathcal{L}' = ((4,0),-1)$ un nuevo conjunto de datos en un problema de clasificación binaria. Se puede verificar que el hiperplano de margen máximo inducido por un *SVM* es

$$H : \left(-\frac{1}{2}, -\frac{1}{2}\right)^T \cdot \boldsymbol{x} + 1 = 0$$

para $\mathcal{L} \cup \mathcal{L}'$. Sea $\mathcal{X}_S = \{\mathcal{L}_S^1 = \mathcal{L}, \mathcal{L}_S^2 = \{((0,0),1),((2,-2),1),((2,2),-1)\}, \mathcal{L}_S^3 = \{((-1,1),1), ((2,-2),1),((2,2),-1)\}\}$ el conjunto parcialmente ordenado de todos los subconjuntos soporte. Se sigue de la definición de subconjunto soporte que todas las funciones de decisión $f(\cdot \mid \mathcal{L}_S^i \cup \mathcal{L}')$, con $i = 1, 2, 3$, son equivalentes. En este caso particular, el

menor cardinal de los elementos de $\mathcal{X}_S$ es alcanzado tanto por $\mathcal{L}_S^2$ como por $\mathcal{L}_S^3$, lo que implica que el mínimo no es único. □

Definición 3. *(Subconjunto soporte máximo) Sea $\mathcal{X}_S$ el conjunto de todos los subconjuntos soporte y $\mathcal{L}_S^1 \subset \mathcal{X}_S$. Si card$(\mathcal{L}_S^m) >$ card$(\mathcal{L}_S^i)$ $\forall \mathcal{L}_S^i \subset \mathcal{X}_S \setminus \mathcal{L}_S^i$, entonces $\mathcal{L}_S^m$ es el subconjunto soporte máximo.*

Lema 3. *El subconjunto soporte máximo siempre existe y es único.*

Demostración. Supongamos que hay dos subconjuntos soporte Máximos diferentes $\mathcal{L}_S^{m_1}$ y $\mathcal{L}_S^{m_2}$. Dado dos conjuntos A y B, si $B \subseteq A$ y card(A) = card(B), entonces se sigue que $A = B$. Por lo tanto, $\mathcal{L}_S^{m_1} = \mathcal{L}$ y $\mathcal{L}_S^{m_2} = \mathcal{L}$ porque son ambos un subconjunto soporte máximo, lo que implica que $\mathcal{L}_S^{m_1} = \mathcal{L}_S^{m_2}$ y contradice el hecho de que son dos conjuntos diferentes. Por lo tanto, el subconjunto soporte máximo es único y $\mathcal{L}_S^m = \mathcal{L}$. La existencia del subconjunto soporte máximo es trivial debido a la prueba del Teorema 1. □

4.2. Metodología de reentrenamiento

En esta sección, se presenta la metodología de reentrenamiento propuesta para *SVM*s. Esta nueva metodología se basa en la noción de subconjunto soporte y en enfoques actuales de la literatura, como *alpha seeding* y el muestreo selectivo. El esquema de la metodología de reentrenamiento, que se muestra en la figura 4.1 y consta de cinco etapas. En la primera etapa, la *SVM* se entrena con el conjunto de datos inicial. Esta etapa no difiere de los métodos de entrenamiento actuales y se describe en la sección 4.2.1. Luego, en la segunda etapa, se estima un subconjunto soporte. El método propuesto de estimación de subconjunto soporte se describe en la sección 4.2.2. La estimación del subconjunto soporte se realiza la primera vez para la *SVM* inicial y se repite en cada iteración para la *SVM* reentrenado. Cuando hay nuevos datos disponibles y se estima el subconjunto soporte, el método calcula el nuevo conjunto de datos de reentrenamiento. Este proceso se divide en dos pasos: primero, se evalúan los nuevos datos y luego se selecciona una muestra de ellos. La evaluación y el cálculo de muestreo selectivo se describen en la sección 4.2.3 y en la sección 4.2.4, respectivamente. Finalmente, este conjunto de datos se utiliza para reentrenar la *SVM* en la última etapa utilizando el procedimiento de la sección 4.2.5.

4.2.1. Entrenamiento: entrenamiento inicial de la SVM

Consideremos un problema de clasificación binaria. Sea $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ donde $\boldsymbol{x_i} \in \mathbb{R}^r$ y $y_i \in \{-1, 1\}$, el conjunto de datos inicial.

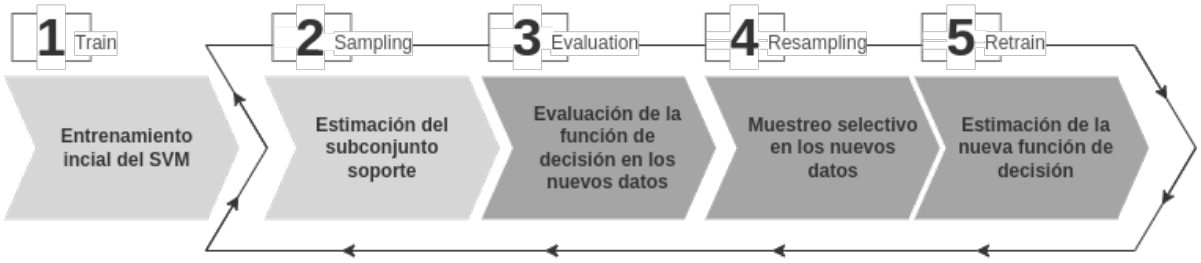


Figura 4.1: *Etapas de la metodología de reentrenamiento. La primera etapa se ejecuta solo una vez en el entrenamiento inicial, mientras que las otras etapas se aplican en cada iteración. El subconjunto soporte en la segunda etapa se calcula después de que la* SVM *se haya entrenado y los pasos desde el* 3 *hasta el* 5 *se ejecutan cuando se disponga de nuevas observaciones.*

En esta etapa, el objetivo es construir una función $f : \mathbb{R}^r \longrightarrow \mathbb{R}$, inducida por la *SVM* (como se explica en la sección 2.3). El hiperplano óptimo que define la frontera de decisión se calcula de la siguiente manera:

$$f(\boldsymbol{x}) = \sum_{i \in sv} y_i \alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b \tag{4.1}$$

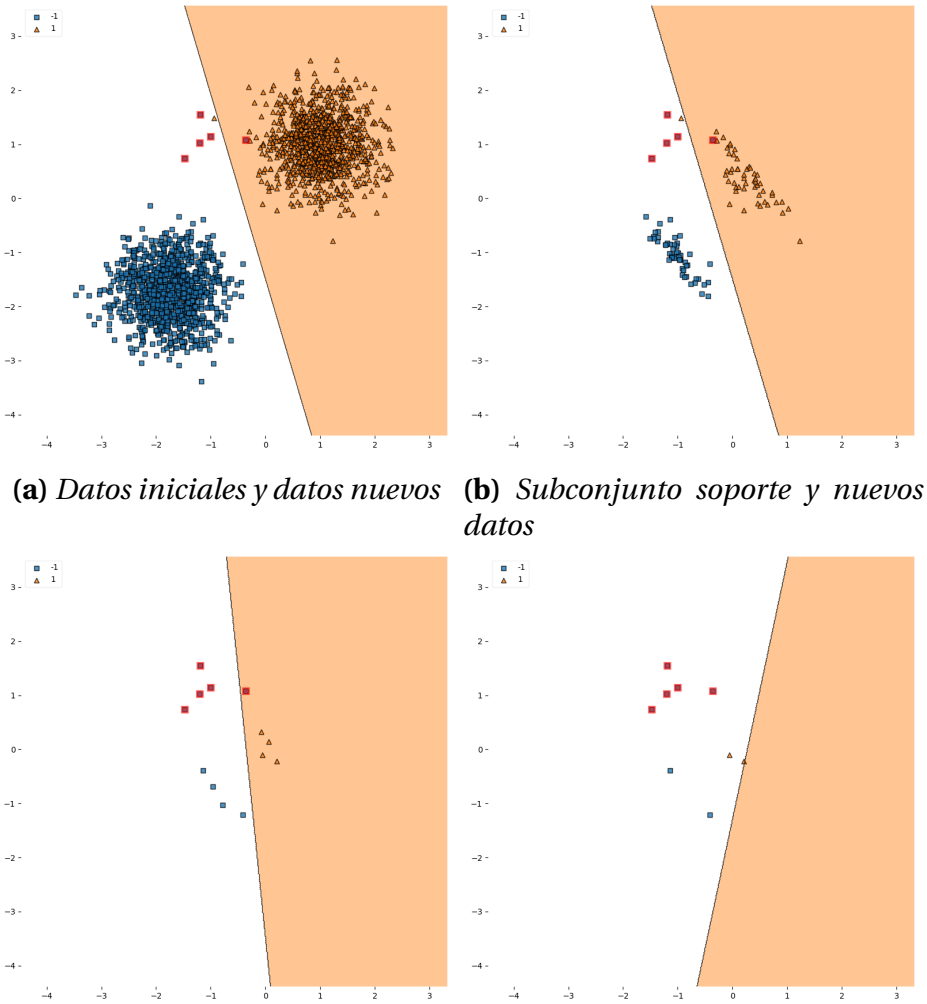
donde sv es el conjunto de vectores soporte y $\boldsymbol{\alpha}$ es la solución del problema de optimización dual.

4.2.2. Muestreo: estimación del subconjunto soporte

Definimos sv^+ y sv^-, como los conjuntos de vectores soporte que pertenecen a las clases positiva y negativa, respectivamente. Nótese que, $sv = sv^+ \cup sv^-$ y $sv^+ \cap sv^- = \emptyset$.

Sea $\mathcal{D}^+ = \{\{\boldsymbol{x_i}, y_i\}_{i=1}^s : f(\boldsymbol{x_1}) \leq f(\boldsymbol{x_2}) \leq \ldots \leq f(\boldsymbol{x_s}), f(\boldsymbol{x_i}) \geq 0, \boldsymbol{x_i} \notin sv\}$ el conjunto parcialmente ordenado de observaciones en $\mathcal{D}$, asignadas a la clase positiva. Sea $\mathcal{D}^- = \{\{\boldsymbol{x_i}, y_i\}_{i=1}^t : f(\boldsymbol{x_1}) \geq f(\boldsymbol{x_2}) \geq \ldots \geq f(\boldsymbol{x_t}), f(\boldsymbol{x_i}) < 0, \boldsymbol{x_i} \notin sv\}$ el conjunto parcialmente ordenado de observaciones en $\mathcal{D}$, asignadas a la clase negativa. Para ambos conjuntos, la relación de orden se corresponde con la distancia al hiperplano óptimo *SVM*.

Como se mencionó anteriormente, el hiperplano de máximo margen de dos conjuntos de datos que comparten todos los vectores soporte es el mismo. En base a esto, el objetivo es estimar el subconjunto soporte, de tal manera que, al agregar los nuevos datos, el hiperplano resultante también sea el mismo que al agregar estos nuevos datos al conjunto inicial. En el mejor de los casos, el subconjunto soporte estimado será un subconjunto soporte mínimo.



(a) *Datos iniciales y datos nuevos*

(b) *Subconjunto soporte y nuevos datos*

(c) *Subconjunto soporte (sin muestra mínima) y nuevos datos*

(d) *Vectores soporte iniciales y nuevos datos*

Figura 4.2: *Hiperplanos generados por una* SVM *entrenado con diferentes subconjuntos de datos. Los nuevos datos se muestran en rojo. El hiperplano en a y b son iguales, pero difieren del calculado con los vectores soporte y los nuevos datos.*

La función de decisión para una *SVM* se representa por el hiperplano de máximo margen definido por el conjunto sv. El hiperplano obtenido a partir de este conjunto será el mismo que el obtenido con todos los datos. Además, existe una relación entre los subconjuntos soporte y el conjunto sv. Por lo tanto, si los vectores soporte están contenidos en el subconjunto soporte, el hiperplano óptimo será el mismo que el obtenido con todos los datos. Por consiguiente, el conjunto sv es un subconjunto de cada estimación del subconjunto soporte. Sin embargo, se consideran observaciones adicionales para asegurar que el hiperplano no cambie cuando se agreguen nuevas observaciones. Estas observaciones, junto con el conjunto sv, constituyen el subconjunto soporte.

La estimación del subconjunto soporte, $\hat{\mathcal{D}}_S$, se lleva a cabo mediante muestreo

selectivo de acuerdo a la distancia al hiperplano óptimo. En este capítulo, proporcionamos una heurística que determina el número de observaciones del subconjunto soporte estimado. El número de observaciones seleccionadas está relacionado con el número de vectores soporte de cada clase. Estas observaciones se seleccionan por distancia creciente al hiperplano. Sin embargo, es conveniente garantizar un número mínimo de observaciones, n_0, que pertenezcan al subconjunto soporte.

La figura 4.2 proporciona un ejemplo para motivar la inclusión de otras observaciones en la estimación del subconjunto soporte, junto con los vectores soporte. Consideremos un caso linealmente separable en el que se presenta una solución óptima, para los datos iniciales y nuevos, en la figura 4.2a. Cuando se obtiene el nuevo hiperplano utilizando el conjunto original de sv y los nuevos datos, hay un cambio considerable en el hiperplano (figura 4.2d). En la figura 4.2c, la heurística sin n_0, no proporciona suficientes observaciones para constituir una buena estimación del subconjunto soporte. Sin embargo, cuando se obtiene el nuevo hiperplano utilizando el método propuesto (figura 4.2b), el hiperplano original no cambia. El número mínimo de observaciones debe seleccionarse en función de la complejidad de los datos y la cantidad de nuevos datos recibidos. Por lo tanto, n_0 debe ser suficientemente grande cuando hay pocos vectores soporte en relación con los nuevos datos. El objetivo es evitar que el tamaño del subconjunto soporte sea demasiado pequeño y que el hiperplano resultante no sea equivalente al obtenido con todos los datos.

Obsérvese que el número de vectores soporte que pertenecen a cada clase puede ser diferente (véase, por ejemplo, la figura 4.3). Si el método de estimación del subconjunto soporte no tiene en cuenta la proporción de vectores soporte entre clases, entonces la nueva frontera de decisión podría cambiar significativamente. Por lo tanto, el número de observaciones en la estimación del subconjunto soporte para cada clase dependerá del número de vectores soporte que pertenecen a esa clase.

Formalmente, el subconjunto soporte se estima como:

$$\hat{\mathcal{D}}_S = \mathcal{D}_S^+ \cup \mathcal{D}_S^- \tag{4.2}$$

donde $\mathcal{D}_S^+ = \{\{\boldsymbol{x}_i, y_i\}_{i=1}^{l+} : l+ = \min(\max(\text{card}(sv^+), n_0), \text{card}(\mathcal{D}^+)), \boldsymbol{x_i} \in X^+\} \cup sv^+$ y $\mathcal{D}_S^- = \{\{\boldsymbol{x}_i, y_i\}_{i=1}^{l-} : l- = \min(\max(\text{card}(sv^-), n_0), \text{card}(\mathcal{D}^-)), \boldsymbol{x_i} \in \mathcal{D}^-\} \cup sv^-$.

Nótese que algunas observaciones mal clasificadas pueden pertenecer a los conjuntos $\mathcal{D}_S^+$ y $\mathcal{D}_S^-$ para el problema de optimización de margen suave. Estas observaciones mal clasificadas suelen representar un porcentaje mínimo del subconjunto soporte y son relevantes en el entrenamiento. En resumen, a menos que haya menos observaciones en el conjunto de entrenamiento, esta muestra garantiza un mínimo de n_0 observaciones a cada lado del hiperplano óptimo.

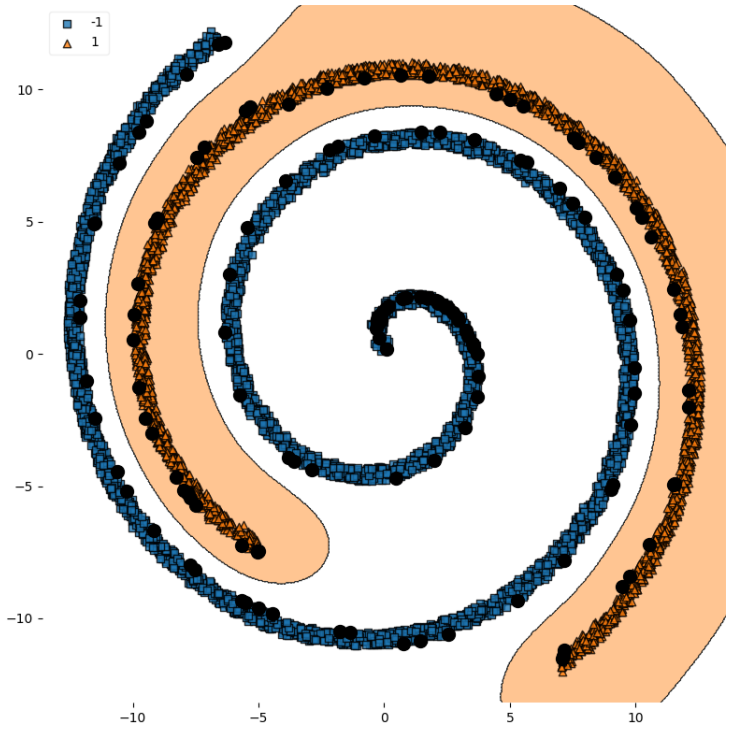


Figura 4.3: *Ejemplo de número desequilibrado de vectores soporte por clase,* 95 *negativos y* 50 *positivos. Los vectores soporte están en negro. Este gráfico muestra que la clase negativa necesita muchos más vectores soporte, ya que su distribución es más compleja que la de la clase positiva.*

4.2.3. Evaluación: Valor de la función de decisión de los nuevos datos

Sea $\mathcal{N} = \{(\boldsymbol{x}'_i, y'_i), : i = 1, 2, \ldots, m\}$ donde $\boldsymbol{x}'_i \in \mathbb{R}^r$ y $y'_i \in \{-1, 1\}$, el conjunto de nuevos datos. Sea $\mathcal{N}_b = \{\{\boldsymbol{x}'_i, y'_i\}_{i=1}^{b}) : |f(\boldsymbol{x}'_i)| \leq 1\}$ el conjunto de observaciones acotadas en $\mathcal{N}$. Sea $\mathcal{N}_{nb} = \{\{\boldsymbol{x}'_i, y'_i\}_{i=1}^{nb}) : |f(\boldsymbol{x}'_i)| > 1\}$ el conjunto de observaciones no acotadas en $\mathcal{N}$. Además, sea $\mathcal{N}_{nbm} = \{\{\boldsymbol{x}'_i, y'_i\}_{i=1}^{nbm}) : f(\boldsymbol{x}'_i) \cdot y'_i < 0\}$, tal que $\mathcal{N}_{nbm} \subset \mathcal{N}_{nb}$, el conjunto de observaciones no acotadas y mal clasificadas de los nuevos datos.

En la etapa de evaluación, el objetivo es evaluar las nuevas observaciones en base a los hiperplanos de la *SVM* actual y si están correctamente clasificadas o no. Por lo tanto, a cada observación en los nuevos datos $\mathcal{N}$ se le asigna al conjunto acotado o no acotado, $\mathcal{N}_b$ y $\mathcal{N}_{nb}$, respectivamente. Además, el conjunto $\mathcal{N}_{nbm}$ contiene todas las observaciones de $\mathcal{N}_{nb}$ que están mal clasificadas.

4.2.4. Re-muestreo: Muestreo selectivo de los nuevos datos

El método propuesto para estimar los subconjuntos soporte descrito en la sección 4.1 requiere la terna $\{\mathcal{L}, \mathcal{L}', f_m\}$ en su cálculo. El conjunto $\mathcal{L}'$ es un subconjunto de los nuevos datos $\mathcal{N}$. Además, $\mathcal{L}'$ es generalmente un subconjunto estricto de $\mathcal{N}$. El sub-

conjunto $\mathcal{L}'$ podría ser igual a los nuevos datos $\mathcal{N}$ principalmente en dos escenarios. En primer lugar, si $\mathcal{L}'$ está generado por la misma distribución de probabilidad que $\mathcal{L}$. Por lo tanto, el subconjunto soporte estimado contiene la información relevante tanto de $\mathcal{L}'$ como de $\mathcal{L}$. En segundo lugar, las distribuciones de probabilidad de $\mathcal{L}'$ y $\mathcal{L}$ son diferentes. En este segundo escenario, si el tamaño del subconjunto soporte es similar o menor al tamaño de los nuevos $\mathcal{L}'$, la *SVM* resultante sería diferente a una *SVM* entrenado con todos los datos (es decir, los nuevos datos podrían definir completamente el hiperplano de separación). Para superar esta limitación, restringimos el tamaño del subconjunto $\mathcal{L}'$ mediante el muestreo selectivo.

Se considera una heurística simple para determinar si los nuevos datos y los datos anteriores provienen de la misma distribución de probabilidad. Consideramos que las distribuciones de probabilidad no son las mismas si hay observaciones no acotadas mal clasificadas en los nuevos datos.

En el primer escenario, la solución anterior solo necesita ajustes menores. Por lo tanto, los únicos candidatos para ser los nuevos vectores soporte son las observaciones acotadas en $\mathcal{N}$. Por lo que si no hay observaciones no acotadas, $\mathcal{N}_{nbm} = \emptyset$, entonces $\mathcal{L}' = \mathcal{N}_b$.

En el segundo escenario, cuando el tamaño del subconjunto $\mathcal{L}'$ es mayor que $n_{min} = \max(|sv|, n_0)$, se selecciona una muestra aleatoria del subconjunto de $\mathcal{N}$ con tamaño n_{min}. Esta heurística reduce el peso de los nuevos datos en el hiperplano de separación cuando el tamaño de los nuevos datos es mayor o similar al subconjunto soporte. Más formalmente, $\mathcal{L}' = \{\{\boldsymbol{x}_i', y_i\}_{i=1}^{n_{min}}\}$ es un conjunto de observaciones en $\mathcal{N}$.

Finalmente, se utiliza el subconjunto $\mathcal{L}'$ para volver a entrenar la *SVM*. Por lo tanto, consideraremos el nuevo conjunto para volver a entrenar $\mathcal{L}_r = \mathcal{L}' \cup \hat{\mathcal{L}}_S$.

4.2.5. Reentrenamiento: Estimación de la nueva función de decisión

En esta etapa, se presenta el método de reentrenamiento basada en el conjunto de datos calculado en las etapas anteriores. Este reentrenamiento utiliza *alpha seeding* para acelerar el proceso de entrenamiento. El reentrenamiento se presenta como un nuevo problema de clasificación binaria. La información de reentrenamiento es:

$$\mathcal{R} = \{(\boldsymbol{x_i}, y_i, \alpha_i, \nabla f(\boldsymbol{x_i})) : i = 1, 2, \ldots, \text{card}(\mathcal{L}_r)\}$$

donde $(\boldsymbol{x}_i, y_i) \in \mathcal{L}_r$, α_i es distinto de cero si y sólo si la observación es un vector soporte. El gradiente se inicializa de la siguiente manera:

$$\nabla f(\boldsymbol{x_i}) = \sum_{j \in sv} y_i y_j \alpha_j K(\boldsymbol{x_i}, \boldsymbol{x_j}) - 1$$

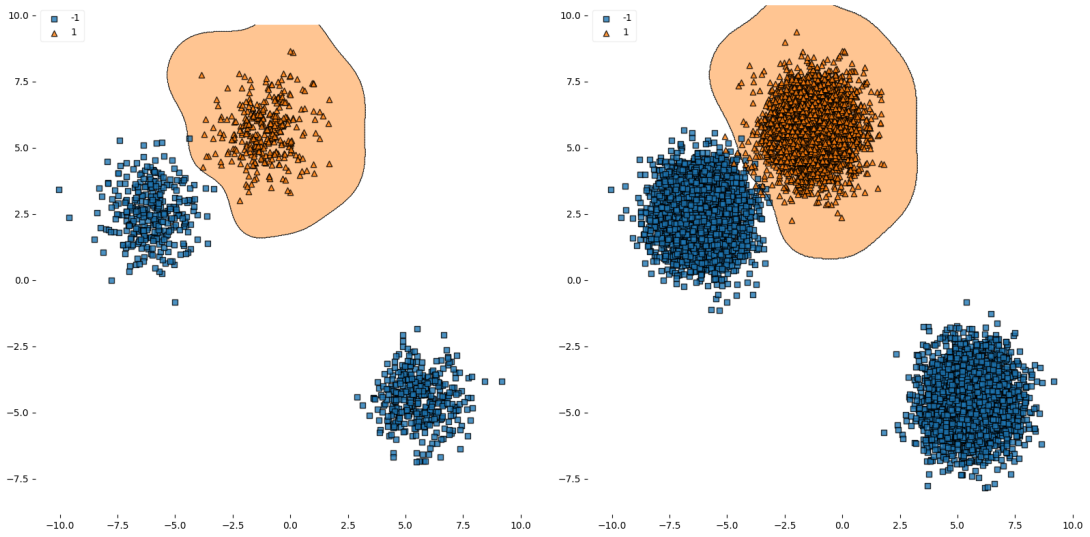
Hay que tener en cuenta que si los alfas son iguales a 0, el gradiente es -1 para todas las observaciones. En la mayoría de las implementaciones basadas en *SMO*, los alfas se inicializan en 0 porque siempre están dentro de la región factible. Por lo tanto, la inicialización del gradiente es muy rápida (es decir, inicializar a un valor constante). Sin embargo, al volver a entrenar con *alpha seeding*, algunos alfas no son 0, y como consecuencia, el gradiente para todas las observaciones tiene que ser calculado. Este cálculo inicial del gradiente es costoso y, en algunos casos, puede tomar más tiempo que el proceso de entrenamiento. Por lo tanto, el *alpha seeding* no siempre obtiene mejores tiempos que el entrenamiento sin *alpha seeding*.

El coste computacional de la inicialización del gradiente aumenta a medida que aumenta la muestra. La propuesta solo emplea el conjunto de datos de reentrenamiento $\mathcal{L}_r$, reduciendo así el tiempo de inicialización del gradiente considerablemente. Este conjunto de datos suele ser más pequeño que el conjunto de datos completo. Por lo tanto, el procedimiento de reentrenamiento produce una función de decisión equivalente utilizando solo un subconjunto de los datos y partiendo de una solución inicial más cercana a la solución óptima. En consecuencia, el método se reentrena más rápidamente y requiere menos recursos computacionales (por ejemplo, tiempo de CPU y memoria).

4.3. Experimentos y resultados

El rendimiento de la propuesta se valida en conjuntos de datos simulados y reales comparándolo con alternativas disponibles en la literatura para distintas métricas de rendimiento. Las alternativas consisten en entrenar utilizando todos los datos disponibles, volver a entrenar utilizando *alpha seeding* y, para el caso incremental se compara con el *LaSVM*, uno de los algoritmos en línea más reconocidos basados en *SVM*s (Bordes *et al.*, 2005). La implementación base de *SVM* para la propuesta y las alternativas no incrementales es *LibSVM* (Chang y Lin, 2011) debido a su amplia adopción en la comunidad de *ML* y sus garantías de convergencia global. En los experimentos, el *SVM* se parametriza con los hiperparámetros predeterminados en la biblioteca *scikit-learn* (Pedregosa *et al.*, 2011). El parámetro n_0 en la heurística del subconjunto soporte se ha establecido en 50 mediante pruebas empíricas en varias distribuciones normales bivariantes.

Los dos primeros experimentos analizan el comportamiento de la metodología de reentrenamiento para diferentes balances entre los datos iniciales y nuevos. En el primer experimento, se consideran datos artificiales que representan varios escenarios, con y sin cambio de distribuciones. En el segundo experimento, se utilizan conjuntos de datos reales conocidos. Por lo tanto, el *SVM* se entrena primero con un conjunto inicial y luego se actualiza con los nuevos datos. El conjunto de entrenamiento inicial



(a) *Datos iniciales* **(b)** *Datos iniciales y datos nuevos*

Figura 4.4: *Distribución de datos para el primer experimento, para el tamaño del conjunto de datos* $n = 10^4$ *y* $p = 0,1$. *Se muestra la diferencia entre la superficie de decisión óptima para los datos iniciales y todos los datos.*

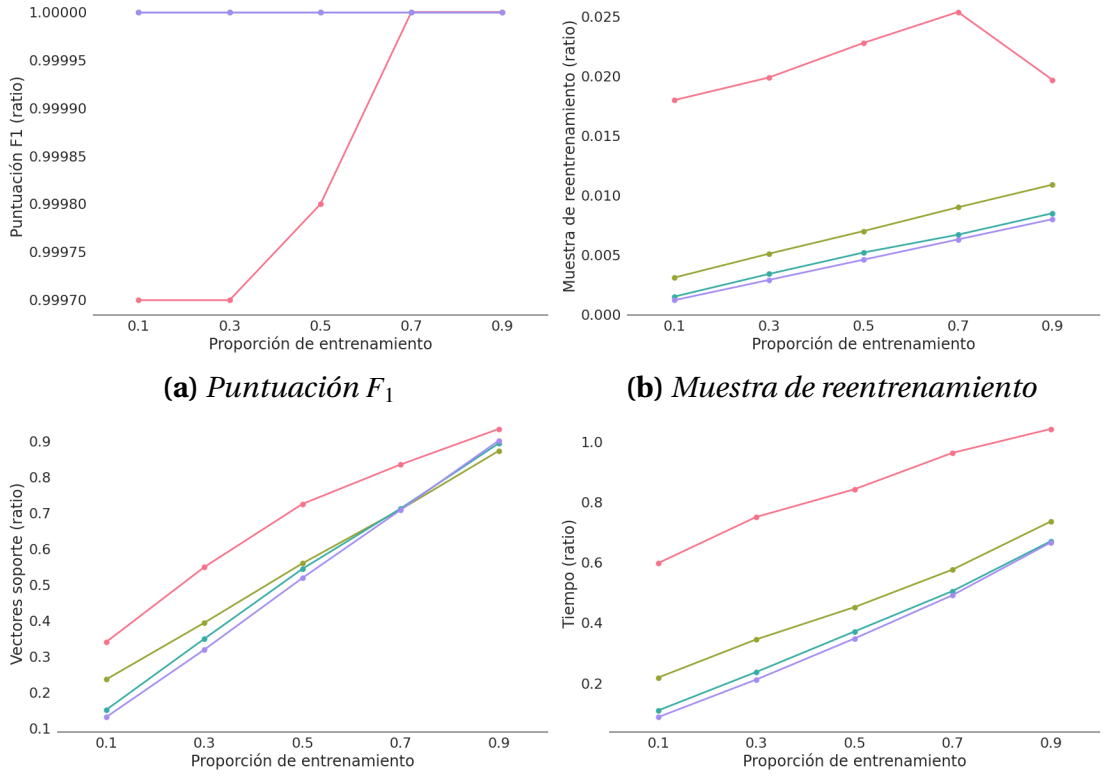
representa una proporción p del conjunto de datos completo, mientras que el paso de reentrenamiento utiliza el resto $1 - p$, para varios valores de p.

El tercer experimento simula un modelo de *SVM* que se despliega y se vuelve a entrenar después de que se disponga de nuevos datos. Esta situación es común en los modelos de *ML* desplegados, que se entrenan con un historial y luego se actualizan periódicamente con nuevos datos.

El proceso de reentrenamiento se evalúa utilizando varias métricas: calidad de la solución, tiempo de entrenamiento (segundos), número de vectores soporte y la proporción de datos considerados en el proceso de reentrenamiento. La calidad de la solución se mide en términos de la puntuación F_1. Todos los algoritmos se llevaron a cabo en una máquina Ubuntu 18.04 con un CPU Intel(R) Core(TM) i7-8700 de 3.20 GHz.

4.3.1. Comportamiento de la metodología de reentrenamiento en escenarios controlados

En los experimentos con conjuntos de datos simulados, se consideran tres conjuntos de datos artificiales con tamaño n. Los resultados para cada métrica en estos conjuntos de datos se muestran como la relación de la propuesta con la mejor alternativa. Es decir, el cociente obtenido al dividir el desempeño de la propuesta por el desempeño de la mejor alternativa. La puntuación F_1 se calcula sobre todos los datos. En la relación de la puntuación F_1, los valores mayores a 1 son preferibles, ya que



(a) *Puntuación* F_1 **(b)** *Muestra de reentrenamiento*

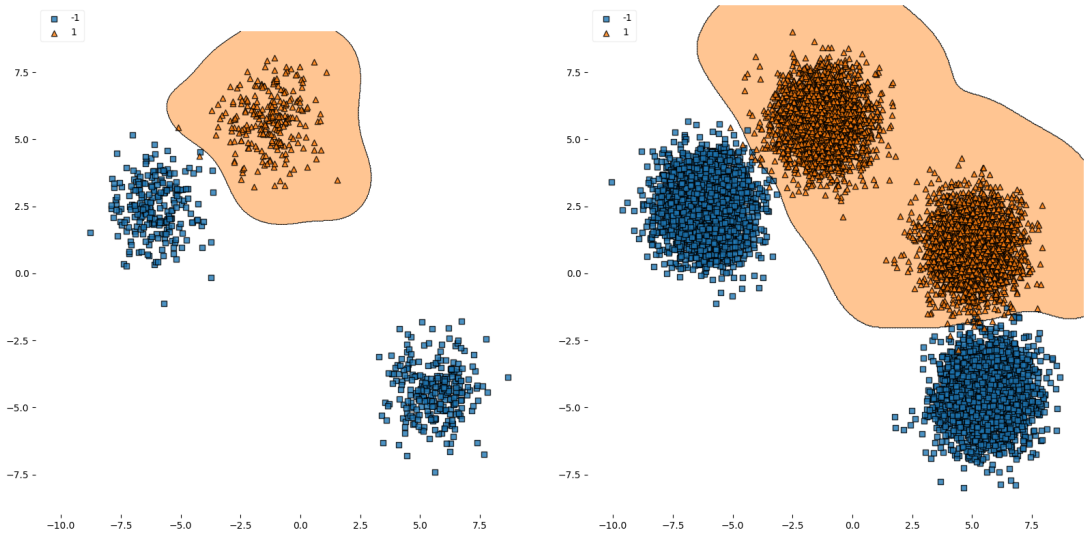
(c) *Vectores soporte* **(d)** *Tiempo*

Figura 4.5: *Los resultados para cada métrica en el primer experimento, se muestran como la relación de la propuesta con la mejor alternativa. Cada tamaño de muestra se muestra en un color diferente: rojo para* 10^4*, verde para* 10^5*, azul para* $5 \cdot 10^5$ *y morado para* 10^6*.*

eso implica que la propuesta tuvo un rendimiento mejor. Por otro lado, en lo que respecta al tiempo, a la muestra de reentrenamiento y al número de vectores soporte, los valores inferiores a 1 indican que la propuesta tuvo un mejor rendimiento.

En el primer conjunto de datos, se consideran tres distribuciones normales bivariantes. Las medias de las distribuciones normales bivariantes se muestrean de una distribución uniforme entre $[0, 10]$, y la matriz de covarianzas es $\mathbb{I}$ (es decir, matriz identidad). Estas distribuciones normales bivariantes se asignan aleatoriamente de manera que la clase negativa contenga dos distribuciones y la clase positiva una. El conjunto de datos inicial y el conjunto de datos de reentrenamiento para el caso $n = 10^4$ y $p = 0{,}1$ se muestra en la figura 4.4.

Los resultados del primer experimento se muestran en la figura 4.5. La propuesta obtiene una puntuación F_1 similar al de las alternativas al mismo tiempo que proporciona tiempos de reentrenamiento más pequeños, utilizando menos datos y vectores soporte. La propuesta redujo el tiempo de reentrenamiento considerablemente, logrando en el mejor de los casos una reducción de 68 a 7 segundos en $n = 10^6$ y $p = 0{,}1$.



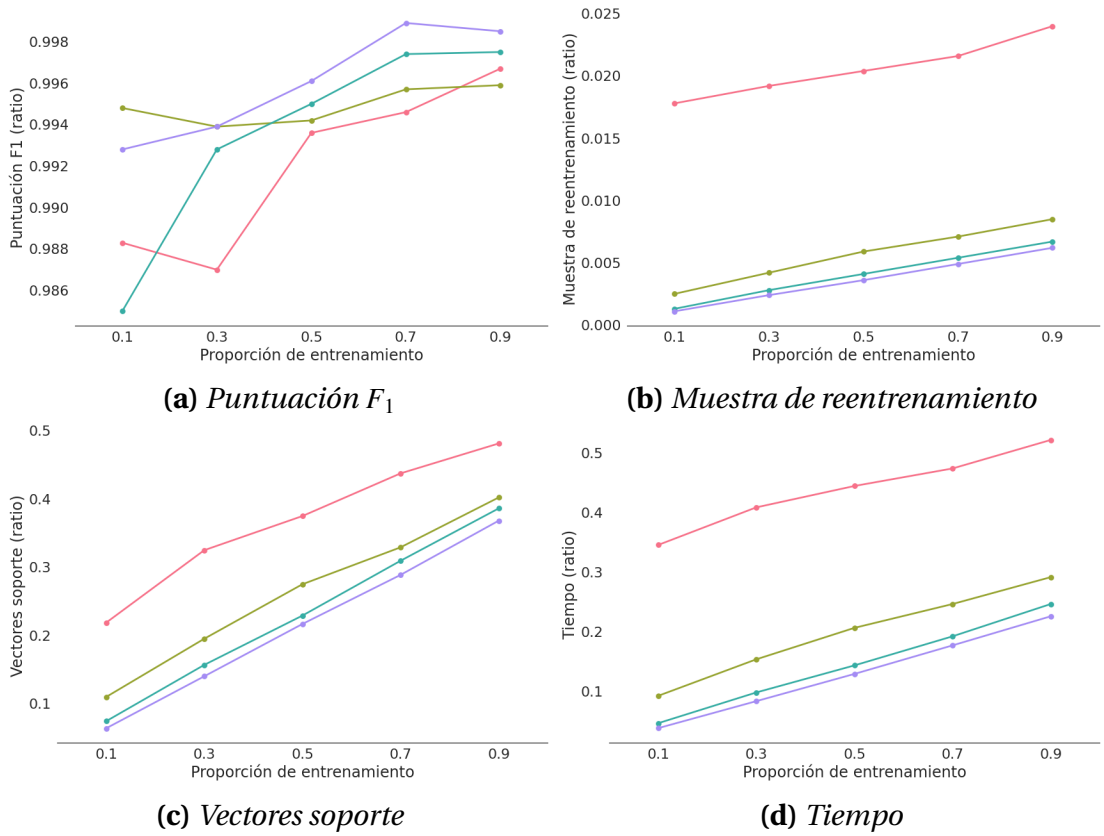
(a) *Datos iniciales* **(b)** *Datos iniciales y datos nuevos*

Figura 4.6: *Distribución de los datos para el segundo experimento, para el tamaño del conjunto de datos* $n = 10^4$ *y* $p = 0{,}1$. *Se muestra la diferencia entre la superficie de decisión óptima para los datos iniciales y todos los datos.*

Además, como los conjuntos de datos son densos, el tamaño de la muestra utilizado por el método propuesto es mucho más pequeño, siendo en el peor de los casos el 2,5 % del total.

En el segundo conjunto de datos simulado, se consideran cuatro distribuciones normales bivariantes. Este experimento simula un escenario en el que el método se entrena inicialmente con una muestra de tres distribuciones normales bivariantes y el proceso de reentrenamiento con una distribución normal bivariante y el resto de los datos de las otras distribuciones. Por lo tanto, el reentrenamiento incluye una muestra de los nuevos datos que pertenecen a una región inicialmente no representada del espacio de características. Las medias de las tres distribuciones normales bivariantes iniciales se muestrean de una distribución uniforme en el intervalo $[0, 10]$, y la matriz de covarianzas es $\mathbb{I}$. Estas distribuciones normales bivariantes se asignan aleatoriamente de manera que la clase negativa contenga dos distribuciones y la clase positiva una. Las medias de la distribución restante se seleccionan manualmente ocupar una región del espacio actualmente asignada a la clase opuesta. Esta última distribución pertenece a la clase positiva. Un ejemplo de este ajuste para los datos iniciales y los nuevos para el caso $n = 10^4$ y $p = 0{,}1$ se muestra en la figura 4.6.

Los resultados del segundo conjunto de datos se muestran en la figura 4.7. En este experimento, el método logra puntuaciones F_1 ligeramente peores, debido a los fuertes cambios en la distribución. Sin embargo, la propuesta disminuye un mínimo del 50 % en los vectores soporte y alrededor del 60 % en el tiempo, y utiliza el 2,5 % de todos los datos.



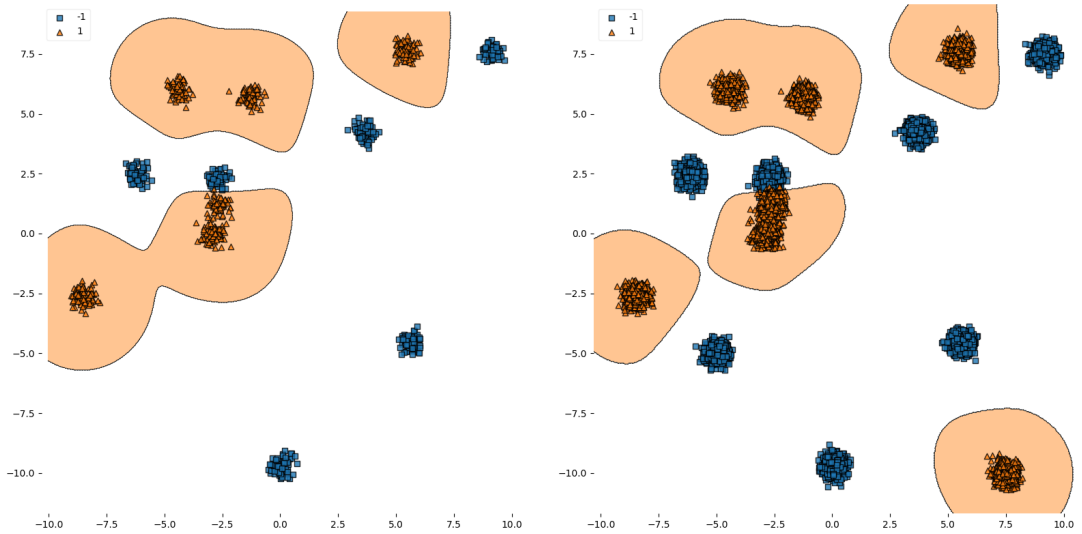
(a) *Puntuación* F_1 **(b)** *Muestra de reentrenamiento*

(c) *Vectores soporte* **(d)** *Tiempo*

Figura 4.7: *Los resultados para cada métrica en el segundo experimento, se muestran como la relación de la propuesta con la mejor alternativa. Cada tamaño de muestra se muestra en un color diferente: rojo para* 10^4*, verde para* 10^5*, azul para* $5 \cdot 10^5$ *y morado para* 10^6*.*

En el último experimento con conjunto de datos simulado, el propósito es similar al del segundo conjunto de datos simulado, pero utilizando más distribuciones normales bivariantes. En este caso, se consideran catorce distribuciones normales bivariantes, pero la muestra inicial solo contiene observaciones de doce de ellas. En las primeras doce distribuciones, sus medias se muestrean de una distribución uniforme entre $[0, 10]$, mientras que su matriz de covarianzas es $0{,}25 \cdot \mathbb{I}$. Las dos distribuciones restantes se colocan manualmente, asignando una a cada clase, y se colocan en una región que pertenece a la clase negativa (es decir, una de ellas se clasifica incorrectamente). La figura 4.8 proporciona un ejemplo de este ajuste para los datos iniciales y los nuevos para el caso $n = 10^4$ y $p = 0{,}1$.

Los resultados para este último experimento se muestran en la figura 4.9. La propuesta supera a la mejor alternativa en todas las métricas consideradas con $p \geq 0{,}3$. Obsérvese que la propuesta proporciona una puntuación F_1 similar mientras utiliza solo el 15 % de la muestra original, el 40 % de los vectores soporte y solo el 30 % del tiempo de la mejor alternativa.



(a) *Datos iniciales* **(b)** *Datos iniciales y datos nuevos*

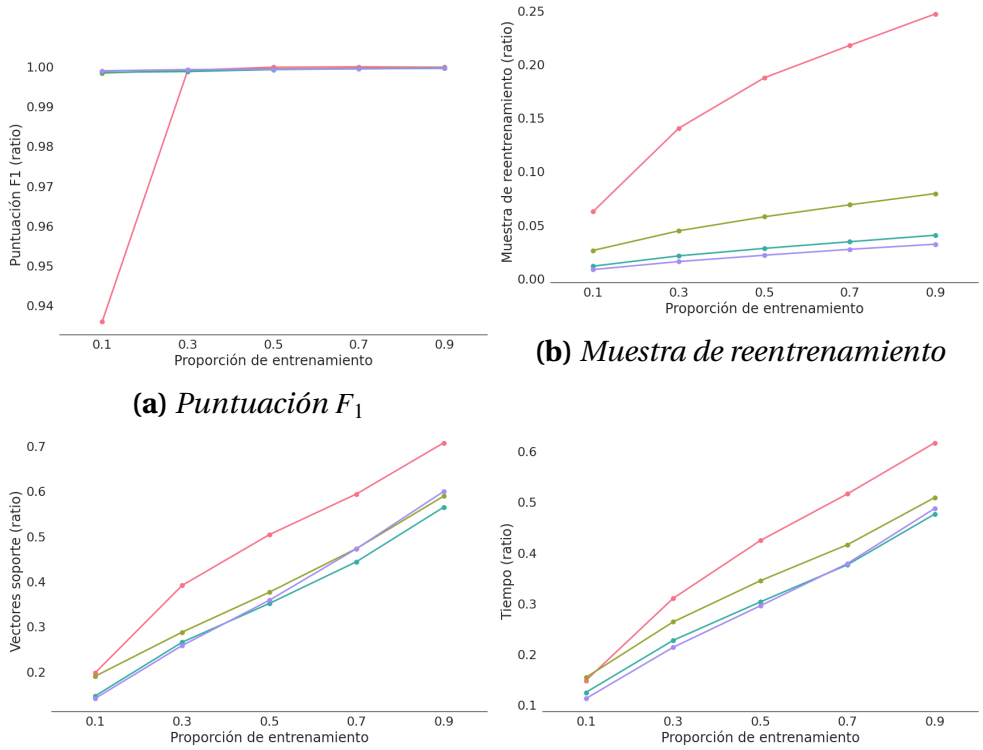
Figura 4.8: *Distribución de los datos para el tercer experimento, para el tamaño del conjunto de datos* $n = 10^4$ *y* $p = 0{,}01$. *Se muestra la diferencia entre la superficie de decisión óptima para los datos iniciales y todos los datos.*

4.3.2. Comportamiento de la metodología de reentrenamiento con datos reales

En esta sección, se evalúa la propuesta en quince conjuntos de datos reales mostrados en la tabla 4.1. Los conjuntos de datos se obtuvieron del repositorio UCI (Dua y Graff, 2017) y *LibSVM Data* (Chang, 2008). Contienen un número diferente de observaciones, características y ratio de observaciones entre las clases. Todos los problemas considerados son binarios.

Estos experimentos analizan las métricas de rendimiento considerando tres proporciones diferentes p (0,3, 0,5, 0,7) entre los datos iniciales y nuevos. Los conjuntos de datos se dividen en entrenamiento (80 %) y test (20 %). Obsérvese que la puntuación F_1 se calcula en el conjunto de test, mientras que las otras métricas miden aspectos del proceso de reentrenamiento.

En el primer experimento real, consideramos $p = 0{,}7$ y los resultados se informan en la tabla 4.2. En este caso, todos los métodos logran una puntuación F_1 similar, excepto en los conjuntos de datos muy desequilibrados y extremadamente desequilibrados, lo que revela una ligera reducción del rendimiento. La propuesta requiere menos vectores soporte en trece de los quince casos. El tiempo de reentrenamiento es similar en todos los métodos con las excepciones de los conjuntos de datos *covtype, skin_nonskin, ijcnn1* y *w8a*. El conjunto de datos *covtype* toma más tiempo porque se consideró el 100 % del conjunto de datos, por lo que incurre en un gran gasto de tiempo de la etapa de evaluación en comparación con las alternativas. En los con-



(a) *Puntuación* F_1

(b) *Muestra de reentrenamiento*

(c) *Vectores soporte*

(d) *Tiempo*

Figura 4.9: *Los resultados para cada métrica en el tercer experimento, se muestran como la relación de la propuesta con la mejor alternativa. Cada tamaño de muestra se muestra en un color diferente: rojo para* 10^4*, verde para* 10^5*, azul para* $5 \cdot 10^5$ *y morado para* 10^6*.*

Conjuntos	N. observaciones	N. características	Proporción clase minoritaria
a9a	32561	124	0,24
banknote	1371	5	0,44
breastcancer	569	31	0,37
cod-rna	59535	9	0,33
covtype	58101	55	0,49
housevotes84	232	17	0,47
ionosphere	351	35	0,36
ijcnn1	233392	23	0,10
mammographic mases	830	5	0,49
r2	116	10	0,45
skin_nonskin	245057	4	0,21
sonar	208	61	0,47
splice	1000	61	0,48
w7a	24692	301	0,03
w8a	49749	301	0,03

Tabla 4.1: *Detalle de los conjuntos de datos reales. El número de observaciones resulta después de eliminar aquellas que tienen valores faltantes. El conjunto de datos covtype corresponde al 10 % de los datos originales de covtype.*

Conjunto	Método									
	***LibSVM* (alpha seeding)**			***LibSVM***			**Subconjunto soporte**			
	F_1	n. sv	tiempo	F_1	n. sv	tiempo	F_1	n. sv	tiempo	muestra
a9a	**0,63**	9415	44,46	**0,63**	9391	**44,08**	**0,63**	**9065**	47,86	0,70
banknote	**1,00**	89	**0,00**	**1,00**	89	**0,00**	**1,00**	**82**	0,01	0,25
breastcancer	**0,90**	124	**0,00**	**0,90**	124	0,01	**0,90**	**118**	0,01	0,64
cod-rna	**0,80**	15361	**20,34**	**0,80**	15362	24,05	**0,80**	**14451**	22,04	0,70
covtype	**0,73**	32669	**126,08**	**0,73**	32669	137,15	**0,73**	**32668**	175,21	1,00
housevotes84	**1,00**	73	**0,00**	**1,00**	**72**	**0,00**	**1,00**	76	0,01	1,00
ijcnn1	**0,94**	10285	69,05	**0,94**	10239	64,79	0,93	**8623**	**39,21**	0,19
ionosphere	**0,98**	103	**0,00**	**0,98**	103	**0,00**	**0,98**	**102**	0,01	0,87
mammographic	**0,74**	**443**	**0,01**	**0,74**	**443**	**0,01**	**0,74**	**443**	0,02	1,00
r2	**0,76**	**89**	**0,00**	**0,76**	**89**	**0,00**	**0,76**	**89**	0,01	1,00
skin_nonskin	**1,00**	2439	12,08	**1,00**	2439	14,35	**1,00**	**1724**	**6,03**	0,03
sonar	**0,69**	**127**	**0,00**	**0,69**	**127**	**0,00**	**0,69**	**127**	0,01	1,00
splice	**0,88**	562	**0,04**	**0,88**	**561**	**0,04**	**0,88**	563	0,07	1,00
w7a	**0,66**	2340	16,59	**0,66**	2307	15,21	0,64	**1967**	**13,33**	0,26
w8a	**0,76**	3751	102,29	**0,76**	3683	93,84	0,73	**3042**	**41,72**	0,21

Tabla 4.2: *Métricas de rendimiento de los tres métodos comparados en conjuntos de datos reales para $p = 0{,}7$. Los mejores resultados en negrita para cada conjunto de datos.*

juntos de datos *skin_nonskin*, *ijcnn1* y *w8a*, el tiempo es considerablemente menor porque se requiere solo del 3 % al 21 % del conjunto de datos. Obsérvese que en nueve de los quince conjuntos de datos, no se requiere toda la muestra y el método utiliza, en promedio, el 11 % menos de vectores soporte.

En el segundo experimento real, la relación entre el conjunto de datos inicial y los nuevos datos es $p = 0{,}5$. Los resultados se informan en la tabla 4.3. La propuesta supera a las alternativas en cuanto al tiempo de reentrenamiento en los conjuntos de datos *a9a*, *cod-rna*, *skin_nonskin*, *injcnn1*, *w7a* y *w8a*. Esta reducción de tiempo se explica mediante el uso de menos datos en el proceso de reentrenamiento. Obsérvese que el método solo utiliza la muestra completa en los conjuntos de datos *housevotes84*, *r2* y *sonar*. Por lo tanto, la propuesta utiliza menos datos cuando es posible, pero recurre al conjunto de datos completo cuando no puede reducir la información. La propuesta también reduce, en promedio, el 20 % el número de vectores soporte. También hay una ligera disminución en la puntuación F_1 de la propuesta en los conjuntos de datos *a9a*, *breastcancer*, *ijcnn1*, *skin_nonskin* y *splice*, y una disminución notable en los conjuntos de datos muy desequilibrados *w7a* y *w8a*. Esto podría explicarse por una estrategia de muestreo agresiva en la fase de remuestreo, que podría descartar observaciones de influyentes.

En el último escenario, véase la tabla 4.4, el tamaño de los nuevos datos es considerablemente mayor ($p = 0{,}3$). En este experimento, la propuesta utiliza, en promedio, el 55 % de datos menos y el 41 % de vectores soporte menos. Como consecuencia, hay una notable reducción en la puntuación F_1 en cinco conjuntos de datos. En los

Conjunto	Método									
	***LibSVM* (alpha seeding)**			***LibSVM***			**Subconjunto soporte**			
	F_1	n. sv	tiempo	F_1	n. sv	tiempo	F_1	n. sv	tiempo	muestra
a9a	**0,63**	9415	45,61	**0,63**	9391	44,08	0,62	**6644**	**29,22**	0,51
banknote	**1,00**	89	**0,00**	**1,00**	89	**0,00**	**1,00**	**69**	0,01	0,24
breastcancer	**0,90**	124	**0,00**	**0,90**	124	0,01	0,88	**97**	0,01	0,59
cod-rna	**0,80**	15361	22,40	**0,80**	15362	24,05	**0,80**	**10613**	**13,56**	0,50
covtype	**0,73**	32669	**130,28**	**0,73**	32669	137,15	**0,73**	**28480**	132,68	0,86
housevotes84	**1,00**	73	**0,00**	**1,00**	**72**	**0,00**	**1,00**	73	0,01	1,00
ijcnn1	**0,94**	10285	73,69	**0,94**	10239	64,79	0,92	**6842**	**29,63**	0,15
ionosphere	**0,98**	103	**0,00**	**0,98**	103	**0,00**	**0,98**	**100**	0,01	0,85
mammographic	**0,74**	**443**	**0,01**	**0,74**	**443**	**0,01**	**0,74**	**395**	0,02	0,87
r2	**0,76**	**89**	**0,00**	**0,76**	**89**	**0,00**	**0,76**	**89**	0,01	1,00
skin_nonskin	**1,00**	2439	13,38	**1,00**	2439	14,35	0,99	**1370**	**4,62**	0,02
sonar	**0,69**	**127**	**0,00**	**0,69**	**127**	**0,00**	**0,69**	**127**	0,01	1,00
splice	**0,88**	562	**0,04**	**0,88**	561	**0,04**	0,87	**522**	0,06	0,91
w7a	**0,66**	2363	17,66	**0,66**	2307	15,21	0,56	**1503**	**10,19**	0,21
w8a	**0,76**	3751	109,00	**0,76**	3683	93,84	0,70	**2540**	**31,54**	0,17

Tabla 4.3: *Métricas de rendimiento de los tres métodos comparados en conjuntos de datos reales para $p = 0{,}5$. Los mejores resultados en negrita para cada conjunto de datos.*

Conjunto	Método									
	***LibSVM* (alpha seeding)**			***LibSVM***			**Subconjunto soporte**			
	F_1	n. sv	tiempo	F_1	n. sv	tiempo	F_1	n. sv	tiempo	muestra
a9a	**0,63**	9415	46,13	**0,63**	9391	44,08	**0,63**	**4106**	**14,23**	0,31
banknote	**1,00**	89	**0,00**	**1,00**	89	**0,00**	**1,00**	**60**	0,01	0,23
breastcancer	**0,90**	124	**0,00**	**0,90**	124	0,01	0,89	**71**	0,01	0,49
cod-rna	**0,80**	15361	24,17	**0,80**	15362	24,05	**0,80**	**6705**	**7,02**	0,31
covtype	**0,73**	32669	134,54	**0,73**	32669	137,15	**0,73**	**17770**	**59,10**	0,53
housevotes84	**1,00**	73	**0,00**	**1,00**	72	**0,00**	**1,00**	**62**	0,01	0,83
ijcnn1	**0,94**	10285	75,20	**0,94**	10239	64,79	0,90	**4892**	**19,75**	0,10
ionosphere	**0,98**	103	**0,00**	**0,98**	103	**0,00**	**0,98**	**82**	0,01	0,65
mammographic	**0,74**	**443**	**0,01**	**0,74**	**443**	**0,01**	0,71	**244**	**0,01**	0,52
r2	**0,76**	**89**	**0,00**	**0,76**	**89**	**0,00**	**0,76**	**89**	0,01	1,00
skin_nonskin	**1,00**	2439	14,79	**1,00**	2439	14,35	0,99	**971**	**3,23**	0,02
sonar	0,69	**127**	**0,00**	0,69	**127**	**0,00**	**0,71**	**118**	0,01	0,90
splice	**0,88**	562	0,04	**0,88**	561	0,04	0,84	**362**	**0,03**	0,56
w7a	**0,66**	2340	16,95	**0,66**	2307	15,21	0,51	**1003**	**6,20**	0,14
w8a	**0,76**	3751	106,46	**0,76**	3683	93,84	0,66	**1738**	**20,54**	0,11

Tabla 4.4: *Métricas de rendimiento de los tres métodos comparados en conjuntos de datos reales para $p = 0{,}3$. Los mejores resultados en negrita para cada conjunto de datos.*

diez conjuntos de datos restantes, la puntuación F_1 de la propuesta es similar a las alternativas, o mejor como en el conjunto de datos *sonar*. Por lo tanto, el método logra puntuaciones F_1 similares en 10 de los 15 casos utilizando solo el 53 % de los datos. En cuanto al tiempo, la propuesta tomó un tiempo similar con respecto a las alternativas en aquellos conjuntos de datos cuya duración de entrenamiento fue menor a un segundo. Sin embargo, logró una reducción significativa en el tiempo en aquellos

conjuntos de datos con tiempos de reentrenamiento más altos: *a9a*, *cod-rna*, *covtype*, *ijcnn1*, *skin_nonskin*, *w7a* y *w8a*. En estos conjuntos de datos, la propuesta mejoró los tiempos de reentrenamiento entre el 22 % y el 47 %.

4.3.3. Simulación de reentrenamiento iterativo

A continuación, consideraremos un escenario de reentrenamiento iterativo en el que estén disponibles nuevos datos varias veces. Este experimento se lleva a cabo en los conjuntos de datos donde la propuesta es menos precisa y que obtuvieron la mayor reducción de la puntuación F_1 en experimentos anteriores: *breastcancer*, *ijcnn1*, *mammographic*, *skin_nonskin*, *splice*, *w7a* y *w8a*. Por lo tanto, este experimento ayudará a comprender esta reducción y a evaluar si las ventajas de la propuesta en los experimentos anteriores degradarán el modelo en varios reentrenamientos consecutivos. Este experimento también incluye los conjuntos de datos *a9a* y *covtype*, que son algunos de los más complejos computacionalmente y con mayores tiempos de ejecución. Esto permite evaluar más precisamente la reducción esperada en el tiempo a lo largo de varias iteraciones. El conjunto de datos se divide en un conjunto de datos inicial (40 %) y seis nuevos conjuntos de datos (10 % cada uno). En estos nuevos conjuntos de datos, el 20 % de los datos se reserva para fines de test. Los hiperparámetros del *SVM* se seleccionan utilizando la validación cruzada de 10 particiones en el conjunto de datos inicial. Los valores óptimos obtenidos son el kernel gaussiano para todos los conjuntos de datos, $C = \{1000, 1, 10, 100, 100, 1, 1\}$ y $\gamma = \{0{,}001, 0{,}01, 0{,}001, 1, 0{,}001, 0{,}01, 0{,}01\}$, para *a9a*, *breastcancer*, *covtype*, *ijcnn1*, *mammographic*, *skin_nonskin* y *splice*, respectivamente.

La tabla 4.5 muestra los resultados del entrenamiento iterativo. Para cuatro de los nueve conjuntos de datos, *LaSVM* y los tres métodos basados en *LibSVM* tienen puntuaciones F_1 satisfactorias. Para el conjunto de datos *breastcancer*, todos los métodos obtienen una puntuación F_1 y un número de vectores soporte similares (es decir, funciones de decisión similares). Sin embargo, *LaSVM* logra mejores resultados globales para las medidas de muestra y tiempo. En *mammographic*, el F_1 promedio es similar en los cuatro métodos, pero el mejor resultado alterna entre *LaSVM* y los métodos basados en *LibSVM* durante cinco iteraciones. Para las demás medidas de rendimiento, *LaSVM* supera a los demás métodos. Téngase en cuenta que la propuesta utiliza menos muestras que las alternativas de *LibSVM*. Para el conjunto de datos *skin_nonskin*, el F_1 es comparable entre todos los métodos, la propuesta genera menos vectores soporte y utiliza menos tiempo de entrenamiento en general, pero *LaSVM* utiliza el tamaño muestral más pequeño de todas. En *splice*, *LaSVM* supera a los métodos basados en *LibSVM* en 0,02 para el promedio de F_1, obtiene las mejores medidas para muestra y tiempo y utiliza ligeramente menos vectores soporte. Para los conjuntos de datos: *a9a*, *covtype* y *ijcnn1*, la propuesta obtiene resultados de

		***LibSVM* (alpha seeding)**				***LibSVM***				***LaSVM***				**Subconjunto soporte**			
	Iter.	F_1	n. sv	muestra	tiempo	F_1	n. sv	muestra	tiempo	F_1	n. sv	muestra	tiempo	F_1	n. sv	muestra	tiempo
a9a	1	**0,62**	5654	15629	64,69	**0,62**	**5644**	15629	81,08	0,40	12617	**2605**	**14,03**	**0,62**	5693	10881	47,90
	2	**0,71**	6581	18233	91,26	**0,71**	**6564**	18233	125,70	0,40	14773	**2604**	**17,35**	**0,71**	6609	12643	58,84
	3	**0,59**	7454	20837	131,62	**0,59**	**7429**	20837	200,11	0,39	16892	**2604**	**20,69**	0,58	7472	14230	75,22
	4	**0,70**	8386	23441	203,45	**0,70**	**8355**	23441	329,69	0,40	19032	**2604**	**23,95**	**0,70**	8413	15758	100,12
	5	**0,67**	9328	26045	352,28	**0,67**	**9290**	26045	566,40	0,38	21278	**2604**	**27,31**	**0,67**	9355	17368	123,39
	6	**0,59**	10232	28649	642,94	**0,59**	**10184**	28649	1012,89	0,38	23468	**2604**	**31,24**	**0,59**	10236	18988	163,16
breastcancer	1	**0,91**	**73**	272	**0,00**	**0,91**	**73**	272	**0,00**	**0,91**	**73**	**45**	**0,00**	**0,91**	**73**	210	0,01
	2	**0,89**	83	317	**0,00**	**0,89**	83	317	**0,00**	**0,89**	**82**	**45**	**0,00**	**0,89**	83	218	0,01
	3	**1,00**	87	362	**0,00**	**1,00**	87	362	**0,00**	0,89	**85**	**45**	**0,00**	**1,00**	87	228	0,01
	4	0,67	**87**	407	**0,00**	0,67	**87**	407	**0,00**	**0,86**	90	**45**	**0,00**	0,67	**87**	232	0,01
	5	**1,00**	**94**	452	**0,00**	**1,00**	**94**	452	**0,00**	**1,00**	**94**	**45**	**0,00**	**1,00**	**94**	197	0,01
	6	**1,00**	96	497	**0,00**	**1,00**	96	497	**0,00**	**1,00**	**93**	**45**	**0,00**	**1,00**	96	196	0,01
covtype	1	**0,71**	**27887**	27888	133,80	**0,71**	**27887**	27888	183,78	0,69	**27887**	**4648**	**40,85**	**0,71**	**27887**	27888	157,63
	2	**0,73**	**32533**	32536	111,43	**0,73**	**32533**	32536	256,06	0,69	32534	**4648**	**48,84**	**0,73**	**32533**	32536	148,17
	3	**0,73**	**37178**	37184	205,29	**0,73**	**37178**	37184	351,88	0,19	37181	**4648**	**57,38**	**0,73**	**37178**	37184	249,96
	4	**0,73**	**41822**	41832	223,46	**0,73**	**41822**	41832	446,37	0,26	41827	**4648**	**65,22**	**0,73**	**41822**	41832	283,68
	5	**0,74**	**46463**	46480	355,30	**0,74**	**46463**	46480	564,16	0,33	46472	**4648**	**73,17**	**0,74**	**46463**	46480	419,15
	6	**0,75**	**51097**	51128	512,88	**0,75**	**51097**	51128	683,36	0,71	51111	**4648**	**82,90**	**0,75**	**51097**	51128	580,25
ijcnn1	1	**0,97**	3331	112028	142,29	**0,97**	3266	112028	161,53	0,89	42580	18672	295,52	**0,97**	**3206**	**8946**	**11,64**
	2	**0,96**	3638	130700	189,76	**0,96**	3513	130700	216,70	0,89	47707	18672	372,08	**0,96**	**3293**	**9618**	**4,42**
	3	**0,98**	4004	149371	256,94	**0,98**	3794	149371	277,75	0,91	52712	18671	466,47	0,97	**3370**	**9879**	**4,72**
	4	**0,98**	4385	168042	311,93	**0,98**	4097	168042	348,41	0,92	57519	18671	515,67	**0,97**	**3464**	**10110**	**4,86**
	5	**0,98**	4696	186713	380,20	**0,98**	4315	186713	435,50	0,94	62228	18671	589,59	**0,98**	**3548**	**10392**	**5,15**
	6	**0,98**	5003	205384	436,45	**0,98**	4524	205384	492,28	0,93	66911	18671	642,84	0,97	**3682**	**10644**	**5,34**
mammographic	1	**0,86**	181	398	**0,00**	**0,86**	181	398	**0,00**	**0,86**	**175**	**66**	**0,00**	**0,86**	181	370	0,01
	2	0,71	217	464	**0,00**	0,71	216	464	0,01	**0,75**	**210**	**66**	**0,00**	0,71	215	427	0,01
	3	**0,89**	261	530	0,01	**0,89**	261	530	0,01	0,64	**212**	**66**	**0,00**	**0,89**	262	490	0,01
	4	0,76	286	596	0,01	0,76	286	596	0,01	**0,79**	**199**	**66**	**0,00**	0,76	289	556	0,01
	5	**0,89**	317	662	0,01	**0,89**	317	662	0,01	0,73	**194**	**66**	**0,00**	**0,89**	318	622	0,02
	6	0,82	350	728	0,01	0,82	350	728	0,01	**0,89**	**197**	**66**	**0,00**	0,82	351	688	0,02
skin_nonskin	1	**0,95**	15428	117626	38,14	**0,95**	15430	117626	45,90	**0,95**	15176	**19604**	**30,93**	**0,95**	**14863**	40461	35,23
	2	**0,95**	17471	137230	49,64	**0,95**	17471	137230	61,47	**0,95**	17181	**19604**	39,50	0,94	**16412**	44589	**27,47**
	3	**0,96**	19599	156834	63,05	**0,96**	19599	156834	78,58	**0,96**	19265	**19604**	52,72	0,95	**18177**	49236	**32,97**
	4	**0,96**	21470	176438	78,22	**0,96**	21469	176438	98,39	**0,96**	21143	**19604**	58,34	**0,96**	**20111**	54531	**39,83**
	5	**0,97**	23230	196042	94,85	**0,97**	23232	196042	119,82	**0,97**	22945	**19604**	61,48	**0,97**	**21928**	59826	**47,88**
	6	**0,97**	24865	215646	112,41	**0,97**	24864	215646	141,65	**0,97**	24673	**19604**	60,15	**0,97**	**23679**	63460	**55,23**
splice	1	0,74	349	480	0,02	0,74	349	480	0,02	**0,80**	**344**	**80**	**0,00**	0,74	349	480	0,03
	2	0,91	392	560	0,02	0,91	391	560	0,02	**0,92**	**384**	**80**	**0,00**	0,91	394	560	0,04
	3	**0,74**	442	640	0,02	**0,74**	440	640	0,03	**0,74**	**435**	**80**	**0,00**	**0,74**	444	640	0,05
	4	**0,84**	472	720	0,03	**0,84**	**471**	720	0,03	**0,84**	473	**80**	**0,00**	**0,84**	474	720	0,05
	5	**0,82**	515	800	0,03	**0,82**	511	800	0,03	0,78	**500**	**80**	**0,01**	**0,82**	517	800	0,06
	6	0,90	552	880	0,04	0,90	547	880	0,04	**0,95**	**540**	**80**	**0,01**	0,90	553	880	0,07
w7a	1	0,65	857	11852	3,75	0,65	857	11852	3,74	0,07	1670	**1976**	2,37	**0,67**	**769**	2081	**0,50**
	2	0,76	962	13828	4,87	0,76	962	13828	4,87	0,06	1928	**1976**	2,78	**0,81**	**877**	2155	**0,51**
	3	**0,75**	1131	15803	6,57	**0,75**	1131	15803	6,53	0,07	2163	**1975**	3,24	**0,75**	**1019**	2461	**0,83**
	4	**0,72**	1248	17778	7,84	**0,72**	1248	17778	8,09	0,06	2449	**1975**	4,01	**0,72**	**1130**	2854	**1,05**
	5	**0,83**	1352	19753	9,67	**0,83**	1352	19753	9,69	0,05	2705	**1975**	4,26	**0,83**	**1228**	3175	**1,26**
	6	0,72	1466	21728	11,71	0,72	1466	21728	11,41	0,06	2995	**1975**	4,68	**0,75**	**1309**	3452	**1,49**
w8a	1	0,60	1480	23879	12,37	0,60	1437	23879	12,17	0,59	3130	3980	9,75	**0,63**	**1308**	**3468**	**1,38**
	2	**0,88**	1644	27859	15,98	**0,88**	**1556**	27859	15,48	**0,88**	2945	3980	10,30	0,86	1613	**3624**	**1,79**
	3	**0,77**	1837	31839	23,53	**0,77**	**1726**	31839	19,70	**0,77**	2466	**3980**	9,13	0,73	1823	4505	**2,65**
	4	0,72	2023	35819	31,16	0,72	**1904**	35819	25,08	**0,75**	2649	**3980**	8,26	0,72	1944	5136	**3,56**
	5	0,75	2171	39799	39,66	0,75	**2043**	39799	32,83	**0,78**	3080	**3980**	9,57	0,71	2059	5464	**3,68**
	6	**0,68**	2365	43779	50,77	**0,68**	**2190**	43779	47,39	**0,68**	3595	**3980**	11,35	0,65	2208	5822	**4,31**

Tabla 4.5: *Métricas de rendimiento de los cuatro métodos comparados, en cada iteración a través del reentrenamiento secuencial simulado. Los mejores resultados en negrita para cada iteración.*

puntuación F_1 comparables a *LibSVM* y supera con creces a *LaSVM*. En *covtype*, la propuesta no puede reducir las muestras. Por lo tanto, los tiempos son ligeramente peores que en el *alpha seeding* pero mejores que *LibSVM*, y el número de vectores soporte es el mismo para los tres métodos. En el conjunto de datos *a9a*, el número de vectores soporte es menor para *LibSVM*, pero la muestra utilizada es menor al 70 % en cada iteración, y la propuesta reduce el tiempo de entrenamiento entre un 84 % y un 26 %. Para el conjunto de datos *ijcnn1*, la propuesta genera menos vectores soporte en cada iteración. La muestra requerida se reduce drásticamente hasta en un 95 % y el tiempo de reentrenamiento hasta en un 99 %. Para el conjunto de datos *w7a*, el rendimiento de *LaSVM* es muy pobre. Por el contrario, la propuesta mejora todas las métricas de rendimiento en comparación con las alternativas. Por ejemplo, el méto-

do de subconjunto soporte reduce, en promedio, la muestra utilizada en un 83 %, el tiempo de entrenamiento hasta en un 87 % y logra una mejora del 1,7 % en la puntuación F_1. En *w8a*, *LaSVM* supera al resto en cuanto a muestra y puntuación F_1, con tiempos de entrenamiento suficientes. *LibSVM* obtiene los mejores resultados en el número de vectores soporte y una puntuación F_1 satisfactoria. Sin embargo, la propuesta obtiene los mejores resultados en la medida de tiempo (hasta una reducción del 91 %) pero un rendimiento F_1 2,5 % peor en general.

Los métodos basados en *LibSVM* superan a *LaSVM* en la puntuación F_1 general. El método *LaSVM* es ligeramente más preciso en solo dos casos. Sin embargo, los métodos basados en *LibSVM* ganan claramente en tres de los siete casos, con diferencias hasta de 0,78. La propuesta alcanza valores F_1 similares a *LibSVM*, con dos excepciones, y obtiene ventajas en los conjuntos de datos más grandes. La propuesta utiliza menos tiempo de entrenamiento y requiere menos muestras, generando soluciones con menos vectores soporte. La propuesta no mejora los tiempos de entrenamiento de *LibSVM* para los conjuntos de datos más pequeños, o esta es mínima, aunque los tiempos de entrenamiento son insignificantes (menos de un segundo).

4.4. Lecciones aprendidas

La complejidad computacional del algoritmo de resolución de *SVM* es $O(n^3)$ en el peor de los casos (Bordes *et al.*, 2005) y $O(n^2)$ en el mejor. El esquema de reentrenamiento basado en la estimación de subconjunto soporte reduce la complejidad a $O(m^3)$, donde $m = \beta n$ y $\beta \in (0,1]$. En el caso en que $\beta = 1$, no hay mejora en la complejidad computacional. La equivalencia de la solución está casi garantizada cuando la proporción entre los datos iniciales y nuevos es igual o mayor al 50 %. De hecho, la metodología propuesta proporciona una solución con menos vectores soporte que utilizando el conjunto completo.

En la mayoría de los casos, con una pequeña proporción inicial entre los datos iniciales y los nuevos datos, la estimación de un subconjunto de soporte disminuye significativamente el tiempo de reentrenamiento y el número de muestras necesarias con un rendimiento excelente. Para conjuntos de datos de pequeño tamaño, la metodología de reentrenamiento no reduce notablemente las muestras utilizadas hasta que la proporción entre el tamaño inicial y la cantidad de nuevos datos sea lo suficientemente baja. Sin embargo, en conjuntos de datos de gran tamaño, la reducción de la muestra es más prominente en diferentes niveles de proporción.

El componente aleatorio en la fase de remuestreo puede subestimar la deriva de la distribución de los nuevos datos y tiene un ligero impacto en la función de decisión final. Por ejemplo, en el conjunto de datos *a9a* con $p = 0,5$, la puntuación F_1 se redu-

ce en $0,01$, y en el conjunto de datos *sonar* con $p = 0,3$, la puntuación F_1 aumenta en $0,02$. Esta diferencia entre las funciones de decisión no es grave, pero la incertidumbre generada podría abordarse mediante el uso de otras técnicas de muestreo.

Es relevante destacar el comportamiento de la propuesta para conjuntos de datos desequilibrados, muy desequilibrados y extremadamente desequilibrados. El rendimiento de la metodología de reentrenamiento en estos casos es muy alto, logrando soluciones de alta calidad con reducciones de muestreo hasta del 95 % y reducciones en el tiempo de reentrenamiento hasta del 99 %. Nótese que, para los muy desequilibrados, la puntuación F_1 difiere, siendo mejor en un caso y peor en otro, en comparación con *LibSVM*. Además, en *w8a*, que tiene una puntuación F_1 más baja en el último experimento, el número de vectores soporte obtenidos por la propuesta es mayor. En consecuencia, en estos casos, el subconjunto de soporte no capta adecuadamente la complejidad del conjunto de datos. Además, para conjuntos de datos desequilibrados, el subconjunto soporte resultante es un conjunto mucho más pequeño y tiende a estar lo más equilibrado posible. Por lo tanto, los subconjuntos soporte tienden a reequilibrar las clases en conjuntos de datos donde estas no lo están.

Capítulo 5

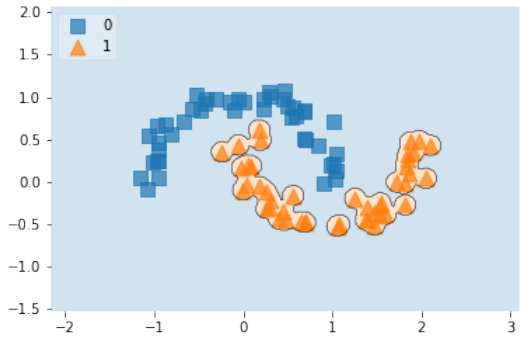
Aprendizaje combinado

Como bien sabemos, los algoritmos de clasificación proporcionan predicciones aprendiendo de los datos del pasado, y el hecho de realizar una predicción lleva asociado un error. En el capítulo 2 hemos analizado cómo construir e interpretar este error de generalización cuando lo reducimos a un único número. Sin embargo, aún no hemos tratado el origen o descomposición de este error. Podemos dividir el error de generalización en dos tipos de errores: reducible e irreducible. El error reducible es evitable, mientras que el error irreducible siempre estará presente. Por lo tanto, el error de generalización se puede reducir disminuyendo el error reducible.

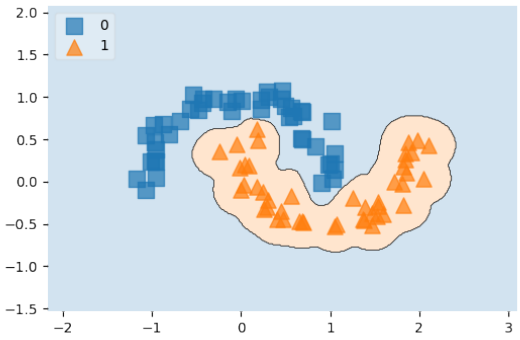
El error reducible también se puede dividir en dos componentes: el error de sesgo y el error de varianza. El sesgo es la diferencia entre la predicción del modelo y el valor real. La varianza es la variabilidad en las predicciones cuando el modelo se entrena con conjuntos de entrenamiento diferentes. Hay un balance entre los errores de sesgo y varianza; cuando uno de ellos aumenta, el otro disminuye y viceversa. Este hecho se vuelve más evidente cuando el alto sesgo corresponde al subajuste y la alta varianza implica el sobresajuste. Sin embargo, la descomposición de sesgo-varianza para problemas de clasificación no es sencilla de evaluar, especialmente para métodos de ensamblado (Valentini, 2004).

No se trata de una novedad asegurar que siempre que se ajusta un modelo de *ML*, el objetivo es lograr un bajo error de generalización. Por lo tanto, el modelo se ajusta en algún punto entre el sobresajuste y el subajuste, el buen ajuste. Hay muchos niveles de sobresajuste que van desde fuertemente sobresajustado hasta bien ajustado, y lo mismo ocurre con el subajuste. Ambas situaciones conducen a modelos indeseables. La figura 5.1 ilustra el concepto de diferentes niveles de ajuste para el conocido conjunto de datos de media luna.

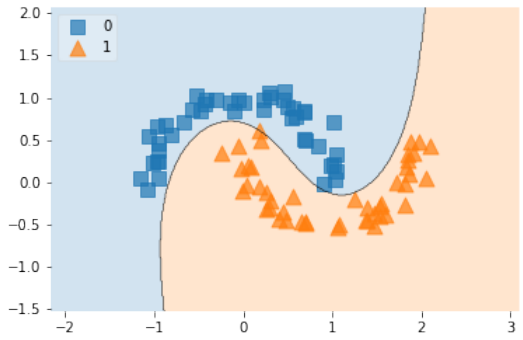
Recordemos que un modelo de *ML* bien ajustado obtiene el mismo error en los conjuntos de entrenamiento y test, y además, coincide con el error Bayes. En la prác-



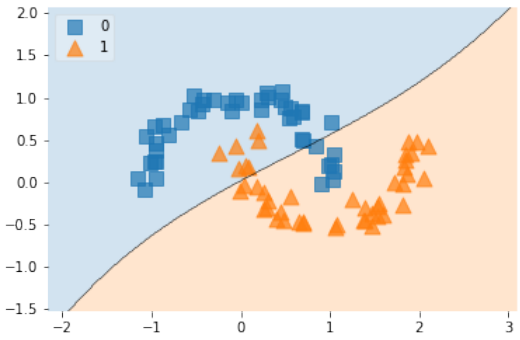
(a) *Clasificador fuertemente sobreajustado*



(b) *Clasificador levemente sobreajustado*



(c) *Clasificador bien ajustado*



(d) *Clasificador subajustado*

Figura 5.1: *Representación de diferentes niveles de sobreajuste a través de diferentes configuraciones de hiperparámetros de* SVM. *Hay cuatro funciones de decisión que representan clasificadores fortemente sobreajustados, ligeramente sobreajustados, bien ajustados y subajustados, respectivamente.*

tica, es deseable que los errores de entrenamiento y test sean ambos pequeños y similares. Por un lado, cuando el error de entrenamiento es mucho menor que el error de test aparece el fenómeno de sobreajuste. Por otro lado, se alcanza el subajuste cuando tanto el error de entrenamiento como el error de test son altos. Un modelo de *ML* deseable busca un equilibrio entre estos dos escenarios.

Los ensamblados son un grupo de técnicas de *ML* que combinan varios modelos reducidos (denominados aprendices individuales) en lugar de un modelo general. Los métodos de ensamblado más conocidos son el *Bootstrap aggregating* (*Bagging*) (Breiman, 1996) y el *Boosting* (Freund *et al.*, 1996). El rendimiento de los métodos basados en ensamblados es generalmente mejor que el de los modelos individuales (y probablemente más complejos). La razón principal de este mejora del rendimiento en ensamblados tipo *Bagging* y *Boosting* es la diversidad de los aprendices individuales. Esta diversidad se obtiene perturbando el conjunto de aprendizaje. En *Bagging*, se consigue la diversidad entrenando cada aprendiz individual en un conjunto de

datos formado por n muestras con reemplazamiento y luego combinando las predicciones mediante un esquema de votación. En los conjuntos basados en *Boosting*, se consigue la diversidad reponderando el conjunto de aprendizaje. Estos pesos se calculan en función de los errores de cada aprendiz individual. En el *Boosting*, estos aprendices individuales se conocen como aprendices débiles. Intuitivamente, un aprendiz débil es un modelo de aprendizaje automático que proporciona predicciones ligeramente mejores que al azar (Schapire, 1990). Finalmente, en lo que respecta al sesgo y la varianza, el proceso de *Bagging* está diseñado para reducir la varianza, mientras que el proceso de *Boosting* reduce el sesgo en las primeras iteraciones y la varianza en las últimas.

Los ejemplos más comunes de *Bagging* y *Boosting* son *Random Forest* (*RF*) (Breiman, 2001) y *eXtreme Gradient Boosting* (*XGBoost*) (Chen *et al.*, 2015), respectivamente. Ambos usan *Decision Tree*s (*DT*s) como aprendices base. La principal cualidad que hace que los *DT*s sean unos aprendices base muy buenos es su inestabilidad. Un modelo de *ML* se considera inestable si un pequeño cambio en la entrada (conjunto de aprendizaje) produce grandes cambios en la salida (predicción). Es bien conocido que los modelos inestables, como los *DT*s, funcionan bien en ensamblados (Ting *et al.*, 2011; Z.-H. Zhou, 2012), a diferencia de los modelos estables como *kNN* o *SVM*.

Un modelo inestable tiene una alta varianza y un bajo sesgo. En otras palabras, los modelos inestables tienden a sobresajustarse. En cuanto al sobresajuste, se ha podido observar empíricamente que los árboles sin podar tienen mejores resultados que los árboles podados para *Bagging* (Bauer y Kohavi, 1999). En particular, el algoritmo *RF* (Breiman, 2001) hace crecer cada árbol lo más grande posible sin podar. Más recientemente, (Han *et al.*, 2020) verifica que los árboles más grandes tienen mejores resultados en *RF*.

5.1. Ensamblados generativos

Una gran variedad de esquemas de combinación y técnicas de ensamblado están disponibles en la literatura de *ML*. Estas técnicas se pueden agrupar de diferentes maneras dependiendo del criterio seleccionado. Por simplicidad, este trabajo sigue la taxonomía de conjuntos propuesta en (Way *et al.*, 2012; Valentini y Masulli, 2002). En esta taxonomía, hay dos niveles fundamentales: métodos de ensamblado generativos y no generativos. La distinción entre estos dos niveles depende de la prevalencia de los métodos de generación y combinación.

Los ensamblados no generativos se centran en la combinación de los aprendices base. Por lo tanto, cada aprendiz se ajusta previamente para lograr un rendimiento

Método de remuestreo	Algoritmo	Características clave
Bootstrap aggregating (*Bagging*)	*Random Forest* (*RF*)	Subconjunto de características aleatorio
	Extra Trees (*ET*)	Subconjunto de características aleatorio puntos de corte aleatorio
Boosting	*Adaptive Boosting* (*AdaBoost*)	Predicción ponderada
	Gradient Boosting (*GB*)	Predicción ponderada función de pérdida de minimización por descenso más pronunciado
	LightGBM	Muestreo basado en gradiente de un solo lado construcción exclusiva de características
	eXtreme Gradient Boosting (*XGBoost*)	Regularización Esbozo cuantil ponderado

Tabla 5.1: *Los ensamblados generativos más populares basados en* Bagging *y* Boosting.

alto. Algunos ejemplos de este tipo de ensamblados son métodos que producen una predicción por votación mayoritaria (Lam y Suen, 1997; Perrone y Cooper, 1992), métodos que combinan el resultado probabilístico por una regla de decisión bayesiana (Xu *et al.*, 1992; Stork *et al.*, 2001) y métodos que seleccionan el mejor subconjunto de aprendices base (Partridge y Yates, 1996). Hay que tener en cuenta que, en todos estos ejemplos, se enfatiza cómo se combinan los aprendices (Hassan *et al.*, 2021). Los ensamblados no generativos han demostrado que combinar la información del aprendiz adecuado (Yu *et al.*, 2021) supera los resultados obtenidos con otros métodos individuales de aprendizaje automático como *SVM* (Yu *et al.*, 2018) o *DT* (Yu *et al.*, 2019) en problemas específicos como el filtrado colaborativo de dominios cruzados.

Los ensamblados generativos producen un grupo de aprendices perturbando el algoritmo base o el conjunto de aprendizaje para cada aprendiz base, centrando su atención en la diversidad y el rendimiento de los aprendices. Por lo tanto, el énfasis está en cómo se construyen los aprendices base, no en la combinación de los resultados individuales. Ejemplos de ensamblados generativos son los métodos de remuestreo (Breiman, 1996; Freund *et al.*, 1996), la selección de características (Rodriguez *et al.*, 2006), la mezcla de expertos (Titsias y Likas, 2002) y los métodos altamente aleatorios (Breiman, 2001; Geurts *et al.*, 2006). Todos estos métodos están diseñados para garantizar la diversidad de los aprendices base. Además, una condición necesaria y suficiente para que los ensamblados generativos tengan mejores resultados que un aprendiz individual es que los aprendices individuales sean precisos (tengan mejores resultados que el azar) y diversos (Hansen y Salamon, 1990).

Los ensamblados generativos más populares son, como era de esperar, *Bagging* y *Boosting*. Han sido ampliamente estudiados y utilizados a lo largo de los años para tareas de clasificación o regresión (Rokach, 2019; González *et al.*, 2020). Hay varios ejemplos conocidos de ensamblados basados en *Boosting* como *AdaBoost* (Freund y Schapire, 1997), *LightGBM* (Ke *et al.*, 2017) y *XGBoost* (Chen *et al.*, 2015), mientras que en los ensamblados basados en *Bagging* la opción predeterminada es *RF* (Breiman, 2001). Todos estos métodos tienen algo en común, están basados en remuestreo y utilizan un *DT* como aprendiz base. Las principales ventajas de los *DT*s son su

robustez contra los valores atípicos y el ruido, y su capacidad para capturar patrones no lineales. Por otro lado, los *DT*s son inestables y suelen sobresajustarse. Sin embargo, estos inconvenientes junto con sus ventajas hacen que los *DT*s sean una buena opción como aprendices base en ensamblados generativos (Dietterich, 2000).

Los hiperparámetros en los métodos basados en *Boosting* tienen una gran influencia en el rendimiento del método. En cambio, la configuración predeterminada de *RF*, en la mayoría de las implementaciones, logra una buena precisión y se puede mejorar solo ligeramente modificando el número de estimadores o el número de características incluidas en cada aprendiz base. Debido a las ventajas de *RF*, hay intentos de *Bagging* utilizando aprendices base estables para emular su rendimiento. La tabla 5.1 muestra un resumen de los conjuntos generativos más comunes y sus principales características. Por ejemplo, en (S.-J. Wang *et al.*, 2009) se consideran conjuntos de *SVM*. En este caso, el *Bagging SVM* lineal supera al *Bagging SVM* gaussiano, como se espera ya que los kernel lineales producen un *SVM* más inestable que los kernel gaussianos. Otros enfoques encontraron mejoras modificando los datos de entrenamiento utilizando diferentes tipos de submuestreo y sobremuestreo (Liu *et al.*, 2011; Q. Wang *et al.*, 2017). Este tipo de muestreo se puede aplicar a conjuntos utilizando como aprendices base redes neuronales bayesianas (Lázaro *et al.*, 2020), donde las técnicas tradicionales de diversidad y reequilibrio trabajan juntas para mejorar el rendimiento de una sola red neuronal bayesiana.

Existen alternativas interesantes a *Boosting* y *Bagging* en la literatura, como *Weight Aggregation* (*Wagging*) (Bauer y Kohavi, 1999), que perturba el conjunto de datos agregando ruido gaussiano al vector de pesos y, por lo tanto, modificando cada réplica. Sin embargo, el rendimiento de los métodos de ensamblado basados en *Wagging* y *Bagging* es similar (Bauer y Kohavi, 1999). Además, aumenta la complejidad del modelo ya que se debe considerar los parámetros de distribución para agregar el ruido. Estudios más recientes para ensamblar modelos estables han recurrido a nuevos métodos de muestreo basados en *Bagging* para lograr la diversidad y precisión requeridas en los diferentes aprendices (Y. Zhang *et al.*, 2019). Sin embargo, este último método no es generalmente aplicable ya que utiliza características específicas del algoritmo base. Por lo tanto, es necesario desarrollar técnicas para aumentar la diversidad para lograr ensamblados efectivos, incluso para aprendices base complicados de ensamblar como *kNN* o *SVM*.

Capítulo 6

Aprendices limitados y sobreajuste local

En este capítulo presentamos el concepto de aprendiz limitado y un nuevo marco general de aprendizaje combinado llamado *MOE*. Los aprendices limitados son aprendices base que se sobresajustan a la muestra de entrenamiento. El enfoque propuesto se basa en sobresajustar mínimamente los aprendices individuales, un caso particular de los aprendices limitados. Además, se propone el muestreo *WRAB*, un nuevo método de re-muestreo que proporciona la diversidad necesaria para construir un modelo de ensamblado con cualquier algoritmo de *ML* (estable o inestable). Los predictores básicos obtenidos a partir de esta construcción son aprendices limitados. La propuesta se evalúa en veinticinco conjuntos de datos reales y su rendimiento se compara con modelos de estado del arte de *ML*, incluyendo varios métodos de *Bagging* y *Boosting*.

6.1. Aprendices limitados

En esta sección, presentamos el concepto del aprendiz limitado. Es un aprendiz que se sobreajusta a la muestra en la que ha sido construido, pero predice suficientemente bien. Antes de definirlo formalmente, consideremos un conjunto de aprendizaje $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ donde $\boldsymbol{x_i} \in \mathbb{R}^r$ ($r \in \mathbb{N}$) y $y_i \in \{-1, 1\}$. Supongamos un método de muestreo (S), con o sin reemplazo. Sea $\mathcal{L}_k^{(S)} = \{(\boldsymbol{x}'_i, y'_i) : i = 1, 2, \ldots, m\}$ la k-ésima muestra del conjunto de aprendizaje, donde $m \leq n$ y $\boldsymbol{x}'_i \in \mathbb{R}^{\leq r}$. Teniendo en cuenta que, si el muestreo se realiza sobre las características, existe una proyección $\phi : \mathbb{R}^r \longrightarrow \mathbb{R}^{\leq r}$ tal que $\phi(\boldsymbol{x}_i) \longmapsto \boldsymbol{x}'_i$, $\forall(\boldsymbol{x}'_i, y'_i) \in \mathcal{L}_k^{(S)}$ donde $(\boldsymbol{x}_i, y_i) \in \mathcal{L}$.

Por lo tanto, cualquier aprendiz se construye utilizando la muestra $\mathcal{L}_k^{(S)}$ y se denota por $f_k : \mathbb{R}^{\leq r} \longrightarrow \{-1, 1\}$, una función sobreyectiva tal que $f_k(\boldsymbol{x}'_i) \longmapsto y_i$ donde $(\boldsymbol{x}'_i, y_i) \in \mathcal{L}_k^{(S)}$. Además, el aprendiz se prueba en $\mathcal{T} \subseteq \mathcal{L} - \mathcal{L}_k^{(S)}$ (es decir, las observaciones *out-of-bag* (*OOB*) para la muestra k (Breiman, 2001)).

Definición 4. *(Predictor aleatorio). El predictor aleatorio es un modelo que predice exclusivamente de acuerdo a la distribución de probabilidad del objetivo.*

Definición 5. *(Aprendiz ingenuo). Un aprendiz ingenuo es un aprendiz que devuelve la etiqueta observada para las observaciones en la muestra de entrenamiento y es un predictor aleatorio en nuevas muestras.*

Definición 6. *(Aprendiz sobreajustado). Un aprendiz sobreajustado f_k es un aprendiz tal que $Error_{test}(f_k) > Error_{train}(f_k)$.*

Empíricamente, podemos agregar una variable de holgura para relajar la condición de que el error de entrenamiento debe ser menor que el error de test (por ejemplo, requerir que la diferencia entre ellos sea mayor que un valor ϵ).

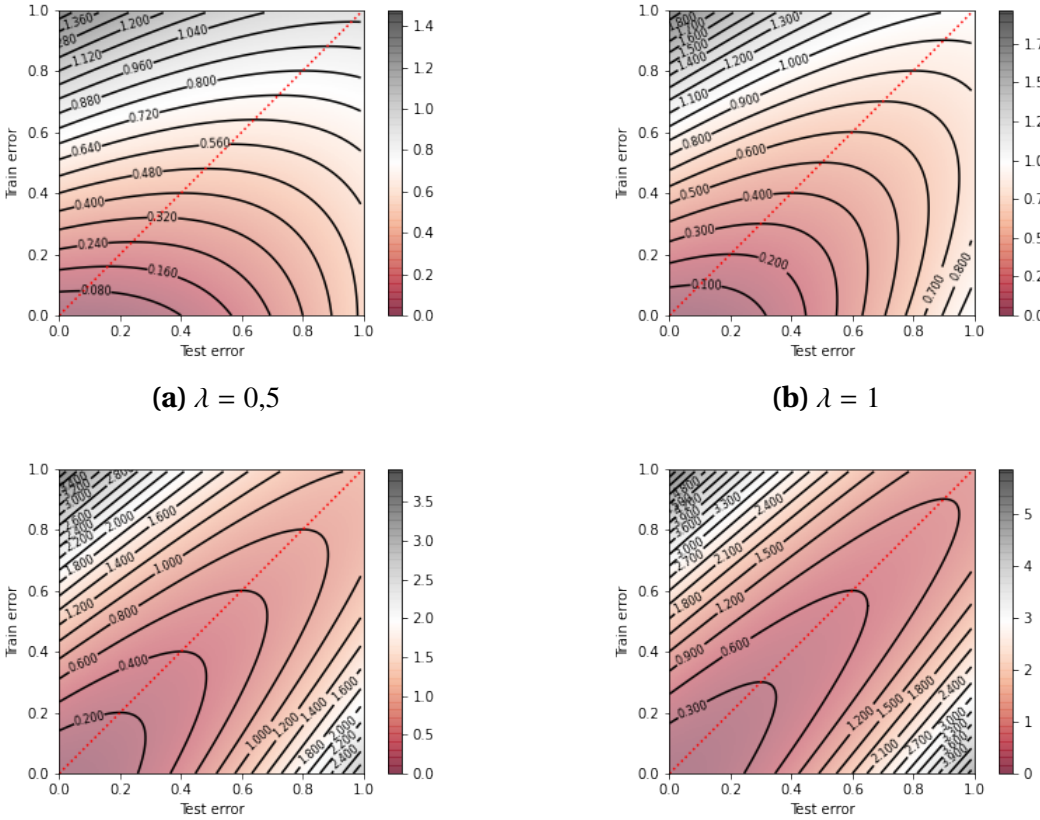
Definición 7. *(Aprendiz limitado). Un aprendiz limitado es un aprendiz sobreajustado que es mejor que un predictor aleatorio en nuevas muestras.*

Hay que tener en cuenta que, los aprendices limitados tienen un error de generalización menor que los aprendices ingenuos por construcción, ya que son mejores que un predictor aleatorio. Los únicos aprendices base en el estado del arte que se pueden considerar como aprendices limitados son los *DT*s utilizados para construir un *RF*. En el *RF*, el tamaño máximo de los *DT*s no está limitado y no se podan. Por lo tanto, estos *DT*s están sobreajustados por construcción (Breiman, 2001). Así pues, los *DT*s en un *RF* son casos particulares de aprendices limitados que sobreajustan tanto como sea posible.

6.2. *Minimally Overfitted Ensemble*

En esta sección se presenta *MOE*, un nuevo marco de ensamblados basado en los conceptos de un modelo con mínimo sobreeajuste y aprendices limitados. El *MOE* se fundamenta en la idea de que el ensamblaje de aprendices ligeramente sobreeajustados aumentará el rendimiento predictivo.

Una forma común de elegir un modelo bien ajustado es definiendo una heurística que identifique el error mínimo en test sujeto a una diferencia mínima entre el error en entrenamiento y el error en test. De esta manera, se evita simultáneamente el sobreajuste tanto en el conjunto de entrenamiento como en el de test. Otra forma es minimizar el error promedio a través de diferentes conjuntos de test usando técnicas como la *validación cruzada de k-particiones* (Wahba *et al.*, 1999). En la práctica, no hay una solución única para encontrar el mejor modelo de *ML* porque el error teórico casi siempre es desconocido. Sin embargo, es universalmente aceptado que la solución puede ser encontrada a través del análisis de diferentes tipos de errores.



(a) $\lambda = 0{,}5$ **(b)** $\lambda = 1$

(c) $\lambda = 3$ **(d)** $\lambda = 5$

Figura 6.1: *Líneas de contorno de la función objetivo del mínimo sobreeajuste para diferentes valores del parámetro λ. La escala de color representa el valor de la función, más cercano a cero es mejor. La región factible del problema de optimización está delimitada a la derecha de la línea roja punteada.*

Siguiendo este enfoque, el mínimo sobreajuste para clasificación binaria se puede definir en términos de los errores de entrenamiento y test. Dado que el objetivo es favorecer el mínimo sobreajuste, el enfoque más simple es minimizar una función de pérdida que tenga en cuenta el error de entrenamiento y el error de test. En esta función, el error de entrenamiento es la principal contribución y el error de test se utilizará para controlar el nivel de sobreajuste, siempre y cuando el error de test sea mayor que el error de entrenamiento.

Definición 8. *(Modelo mínimamente sobreajustado). Dado un conjunto $\mathcal{F}$ de modelos de* ML *ajustados con el mismo conjunto de entrenamiento y evaluados en el mismo conjunto de test, el modelo con mínimo sobreeajuste es la solución de:*

$$\begin{aligned} \arg\min_{f \in \mathcal{F}} L_{mo}(\lambda, f) &= Error_{train}(f) + \lambda(Error_{test}(f) - Error_{train}(f))^2 \\ &\textit{sujeto a } Error_{test}(f) > Error_{train}(f) \end{aligned} \tag{6.1}$$

tal que $\lambda \in (0, \infty)$.

Hay que tener en cuenta que $L_{mo}(\lambda, f)$ es la función de pérdida para el sobreeajuste sujeta a $Error_{test}(f) > Error_{train}(f)$. El modelo f que minimiza L_{mo}, para un λ fijo, es el que tiene mínimo sobreeajuste de entre los que pertenencen al conjunto $\mathcal{F}$. El parámetro λ permite controlar el nivel de sobreeajuste que se considera el mínimo. Los valores más cercanos a cero favorecen modelos sobreeajustados, mientras que los valores altos penalizan el sobreeajuste. Los efectos del parámetro λ se muestran en la figura 6.1. El grado de sobreeajuste disminuye a medida que aumenta el valor de λ. Es una pregunta subjetiva cuál es el grado de sobreeajuste que corresponde al mínimo sobreeajuste. Por lo tanto, el valor de λ debe establecerse de acuerdo con el problema específico que queramos resolver, en consecuencia dependerá del dominio de aplicación. Los valores empíricos de λ serán principalmente unas pocas unidades o como mucho unas pocas decenas de unidades, pero nunca cercanas a cero, y el cardinal de $\mathcal{F}$ será lo suficientemente grande. Observamos que cuanto más pequeño sea el conjunto de aprendizaje, menos modelos diferentes habrá en $\mathcal{F}$.

El algoritmo 6.1 presenta el pseudo-código del marco *MOE*. Al igual que en *Bagging*, el marco propuesto comienza obteniendo m muestras del conjunto de aprendizaje con un método de muestreo (S). Luego, se entrena un aprendiz base para cada muestra generada $\mathcal{L}_k^{(S)}$ y se selecciona el aprendiz con mínimo sobreeajuste. Finalmente, se realiza el proceso de combinación de predicciones usando la regla de voto mayoritario. Obsérvese que el *MOE* incorpora un paso adicional que incluye limitaciones sobre los aprendices seleccionados. En *Bagging*, los hiperparámetros son los mismos para cada aprendiz individual. Sin embargo, en el *MOE*, dado un conjunto de hiperparámetros $\mathcal{H}$, se elige la configuración de hiperparámetros que proporciona el aprendiz con mínimo sobreeajuste. Por lo tanto, para cada muestra $\mathcal{L}_k^{(S)}$, el método proporciona el aprendiz limitado con mínimo sobreeajuste según la ecuación (6.1) (es decir, m aprendices limitados). El tiempo de cómputo del *MOE* es $O(m \cdot card(\mathcal{H}) \cdot (t + p_1 + p_2))$, donde t es la complejidad del entrenamiento del aprendiz base, p_1 es la complejidad de la predicción de $\mathcal{L}_k^{(S)}$ con el aprendiz base, p_2 es la complejidad de la predicción de las observaciones *OOB* y $card(\mathcal{H})$ es el tamaño del conjunto de hiperparámetros.

Es evidente que los aprendices construidos de acuerdo a la Definición 8 cumplen con las condiciones de aprendices limitados. Asumiendo que λ está establecido en

Algoritmo 6.1 *Minimally Overfitted Ensemble*

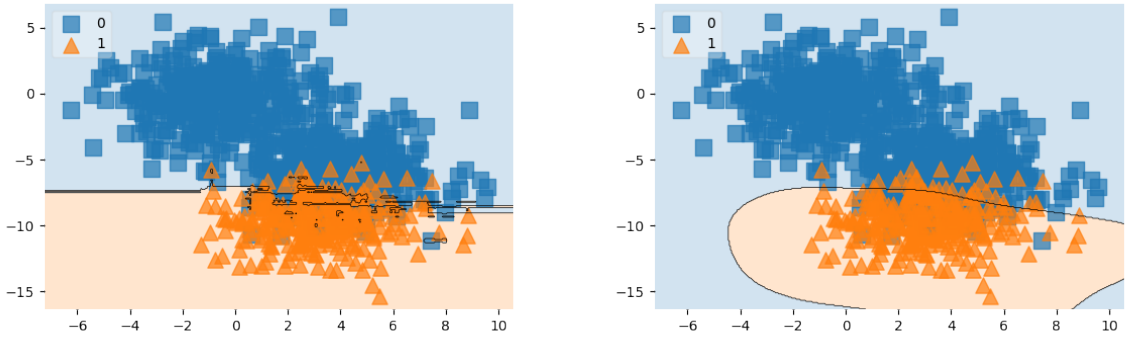
Entrada: $(\mathcal{L}, \mathcal{H}, \lambda, n, m, S, BaseModel)$ ▷ S es el método de muestreo
1: $\mathbb{L} \leftarrow \emptyset$ ▷ inicialización del conjunto de aprendices limitados
2: **Para** $k \in [1, \ldots, m]$ **ejecuta**:
3: $\quad \mathcal{L}_k = s(\mathcal{L}, n)$
4: $\quad test = \mathcal{L} - \mathcal{L}_k^{(S)}$
5: $\quad L^* = \infty$
6: $\quad$ **Para** $h_j \in \mathcal{H}$ **ejecuta**:
7: $\quad\quad l(h_j) \leftarrow FitModel(h_j, \mathcal{L}_k)$ ▷ ajuste j-ésimo aprendiz limitado
8: $\quad\quad$ **Si** $L_{mo}(\lambda, l(h_j)) < L^*$ **entonces:**
9: $\quad\quad\quad L^* \leftarrow L_{mo}(\lambda, l(h_j))$
10: $\quad\quad\quad \mathbb{L}_k \leftarrow l(h_j)$ ▷ actualización i-ésimo aprendiz limitado
Salida: $F(x) = argmax \sum_{k=1}^{m} \mathbb{L}_k(x)$ ▷ salida según la regla de voto mayoritario

los valores recomendados anteriormente y el conjunto de hiperparámetros $\mathcal{H}$ es lo suficientemente grande para la muestra $\mathcal{L}_k^{(S)}$, entonces un subconjunto de los aprendices estarán sobreeajustados. Entre estos aprendices sobreeajustados, se selecciona el aprendiz limitado con mínimo sobreeajuste según la Definición 8. Obsérvese que el error de entrenamiento del aprendiz limitado es menor que el error de test y se evita el subajuste porque ambos errores se minimizan. Por lo tanto, es evidente que el aprendiz con mínimo sobreeajuste supera al aprendiz ingenuo. El análisis de la cota de error teórica correspondiente se proporciona en el Apéndice A. Este error se puede estimar en términos de la precisión de los aprendices limitados y la correlación entre ellos. Para calcular la función objetivo con mínimo sobreeajuste, se calcula el $Error_{train}$ en la muestra $\mathcal{L}_k^{(S)}$ y el $Error_{test}$ en el conjunto $\mathcal{L} - \mathcal{L}_k^{(S)}$. El enfoque *OOB* para medir el error proporciona una forma precisa de evaluar el nivel de sobreeajuste de cada aprendiz. Téngase en cuenta que se podría utilizar cualquier métrica de evaluación para calcular ambos errores. Sin embargo, en algunos problemas de clasificación, la exactitud fomenta el sobreeajuste en los aprendices frente a otras medidas de evaluación como la puntuación *F1* o el *MCC* (Grandini *et al.*, 2020).

Como se mencionó anteriormente, se necesita un método de muestreo para obtener cada muestra $\mathcal{L}_k$. Dado sus similitudes con *Bagging*, la primera opción para el muestreo es *bootstrapping*. Sin embargo, *bootstrap* no genera suficiente diversidad para la condición de mínimo sobreeajuste. Aquí, el objetivo principal es garantizar la diversidad de la muestra para asegurar que se combinen funciones de decisión suficientemente diferentes. Por lo tanto, modificar la proporción de muestreo en cada clase ayudará a alcanzar la diversidad en las funciones de decisión de los aprendices limitados.

6.2.1. *Weighted Random Bootstrap*

WRAB es un procedimiento de muestreo que consiste en añadir una capa de pesos aleatorios que desequilibran el tamaño de la muestra entre las diferentes clases en un problema de clasificación. Consideremos un tamaño de muestra n y un conjun-



(a) *Función de decisión estimada con* RF **(b)** *Función de decisión estimada con* SVM

Figura 6.2: *Distribución de datos para el ejemplo con 666 puntos para la clase azul y 333 puntos en la clase naranja. Se muestra la diferencia entre la superficie de decisión óptima para* RF *y* SVM.

to de aprendizaje $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ donde $\boldsymbol{x_i} \in \mathbb{R}^r$ y $y_i \in \mathbb{N}^m$. El procedimiento extrae S muestras de $\mathcal{L}$. Cada muestra *WRAB* se denota por

$$\bigcup_{j=1}^{m} \mathcal{L}^{*b} = \{(\boldsymbol{x}_i^{*b}, y_i^{*b}) : i = 1, 2, \ldots, n_j\} \tag{6.2}$$

donde $b = 1, 2, \ldots, S$ y $n_j \in [0, n]$. Cada n_j asegura que $\sum_{j=1}^{m} w_j n_j = n$ donde $\sum_{j=1}^{m} w_j = 1$ y $w_j \sim U(0, 1)$. Hay que tener en cuenta que algunas muestras podrían aparecer varias veces porque cada muestra *WRAB* se obtiene mediante muestreo con reemplazo.

Por lo tanto, el marco *MOE* tiene cuatro hiperparámetros: el número de muestras para cada aprendiz base, el tamaño de estas muestras, el grado de sobreeajuste y el método de muestreo (*WRAB* o *bootstrap*). Los dos primeros parámetros son ampliamente conocidos en conjuntos generativos, mientras que los dos últimos son hiperparámetros propios de *MOE*.

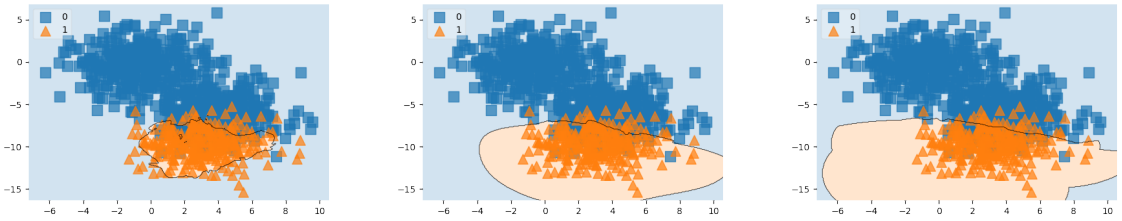
6.3. Experimentos

En esta sección, se estudia y evalúa el marco propuesto frente a varias alternativas. Primero, se discute el rendimiento y los hiperparámetros de *MOE* en un conjunto de datos artificial. Luego, se utilizan veinticinco conjuntos de datos de clasificación binaria, con una variedad de observaciones, número de características y equilibrio entre las clases, para evaluar la propuesta. Finalmente, se resumen las lecciones aprendidas. El código fuente y los datos para reproducir los experimentos se pueden encontrar en `https://github.com/URJCDSLab/moe`.

6.3.1. Análisis de los hiperparámetros de sobreeajuste en un escenario controlado

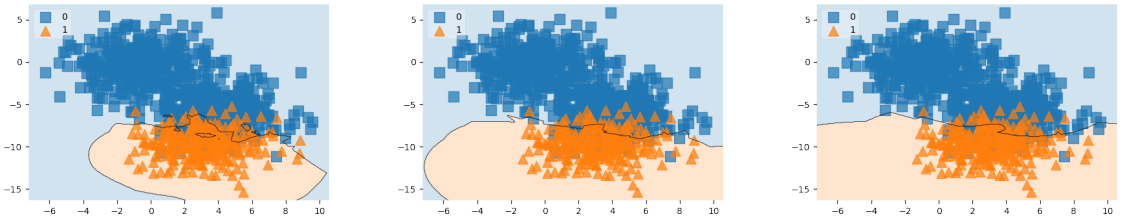
Se considera un problema de clasificación binaria con tres distribuciones normales bivariantes cuyos centros se colocan aleatoriamente en $(-0{,}65, -0{,}1)$, $(2{,}96, -9{,}52)$ y $(4{,}05, -6{,}06)$ en la región cuadrada $[-10, 10] \times [-10, 10] \in \mathbb{R}^2$ con una desviación estándar igual a 2. Los dos primeros pertenecen a la clase 0 y el último pertenece a la clase 1. Son independientes entre sí y contienen 333 puntos cada uno. En este ejemplo, se consideran *RF* y *SVM* como modelos base de *ML*. Ambos modelos presentan errores de clasificación similares (alrededor del 10 %), obtenidos con validación cruzada de 10 particiones mediante la suma de falsos positivos y falsos negativos en 10 particiones de test. Las funciones de decisión tomadas como referencia, estimadas con *RF* y *SVM*, se presentan en la Figura 6.2. En este caso, para el *MOE*, los aprendices limitados utilizarán *SVM* con un kernel Gaussiano. Además, el espacio de búsqueda para minimizar la función de sobreajuste mínima está conformado por el parámetro de regularización $C \in \{1, 10, 100, 1000\}$, y el coeficiente del kernel $\gamma \in \{0{,}01, 0{,}1, 1, 10\}$. Los resultados para los tres valores diferentes del parámetro de sobreajuste λ se muestran en la Figura 6.3. Para cada ejemplo, se construyen 10 aprendices limitados utilizando *bootstrap* donde el tamaño de la muestra es igual al 10 % del conjunto de aprendizaje completo. Una definición adecuada de la función objetivo en la definición del modelo con sobreajuste mínimo es esencial para lograr un alto rendimiento. Un valor bajo de λ implica un método extremadamente sobreajustado como se muestra en la figura 6.3a. En este caso, la tasa de error es del 15 % (50 % peor de lo esperado). El valor más alto del parámetro λ logra la tasa de error esperada del 10 %, pero como se muestra en la figura 6.3c, la función de decisión correspondiente es bastante similar al algoritmo base. De hecho, esta situación no necesariamente sería incorrecta, ya que la función de decisión de *SVM* es más adecuada para una de las clases, en oposición a la función de decisión de *RF*. Como se muestra en la figura 6.2, ambas funciones de decisión (*RF* y *SVM*) son adecuadas para este problema. Una solución intermedia se puede encontrar en la figura 6.3 para el mayor valor de λ. En este ejemplo, la distribución es más densa cerca de la media en cada una de las tres nubes normales bivariantes. Por lo tanto, valores bajos de λ combinados con muestreo *bootstrap* y un algoritmo base estable conducen a una menor diversidad de los aprendices limitados. Por lo tanto, para aumentar la diversidad de las funciones de decisión locales para cada aprendiz, se incrementa el parámetro λ. La figura 6.3 muestra la alta flexibilidad del ensamblado en base al parámetro λ.

La tabla 6.1 presenta los errores de entrenamiento y test del esquema *MOE* para diferentes métodos de muestreo y diferentes valores del parámetro λ. Además, también está disponible el valor mínimo de la función de pérdida de sobreajuste. Los resultados de esta tabla revelan que, para algunas elecciones de λ, el sobreajuste del



(a) $\lambda = 1$ **(b)** $\lambda = 3$ **(c)** $\lambda = 5$

Figura 6.3: *Las tres diferentes superficies de decisión para diferentes valores del parámetro de sobreajuste en el marco* MOE *con muestreo* bootstrap.



(a) $\lambda = 1$ **(b)** $\lambda = 3$ **(c)** $\lambda = 5$

Figura 6.4: *Las tres diferentes superficies de decisión para diferentes valores del parámetro de sobreajuste en el marco* MOE *con muestreo de* WRAB.

Hiperparámetros	Error entrenamiento	Error test	L_{mo}
bootstrap, $\lambda = 1$	0.00 (0.00)	0.19 (0.03)	0.04 (0.01)
bootstrap, $\lambda = 3$	0.04 (0.02)	0.13 (0.04)	0.07 (0.03)
bootstrap, $\lambda = 5$	0.07 (0.03)	0.12 (0.03)	0.08 (0.03)
WRAB, $\lambda = 1$	0.02 (0.03)	0.20 (0.03)	0.05 (0.02)
WRAB, $\lambda = 3$	0.02 (0.03)	0.16 (0.04)	0.05 (0.02)
WRAB, $\lambda = 5$	0.05 (0.03)	0.16 (0.04)	0.10 (0.03)

Tabla 6.1: *Detalle de los errores para el marco* MOE *con diferentes configuraciones de metodología de muestreo y parámetro de control de sobreajuste. Se muestra la media (y la desviación estándar) de los aprendices limitados para cada configuración.*

muestreo *WRAB* es mayor que el logrado con la metodología *bootstrap*. Por lo tanto, el *WRAB* proporciona más diversidad entre las muestras y mantiene el sobreajuste mínimo así como el muestreo *bootstrap*.

6.3.2. Análisis del rendimiento de las variaciones de MOE

A continuación, se evalúa el marco *MOE* en una batería de conjuntos de datos de clasificación binaria. Los conjuntos de datos se han obtenido del repositorio UCI Dua y Graff (2017), Penn Machine Learning Benchmarks Olson *et al.* (2017) y LIBSVM Data Chang (2008). Un resumen de sus principales características se presenta en la tabla

Conjuntos	N. observaciones	N. características	Proporción clase minoritaria
appendicitis	106	8	0.198
australian	690	15	0.445
backache	180	32	0.139
banknote	1372	5	0.445
breastcancer	569	31	0.373
bupa	345	6	0.490
cleve	303	14	0.455
colon-cancer	62	2001	0.355
diabetes	768	9	0.349
flare	1066	11	0.171
fourclass	862	3	0.356
german_numer	1000	25	0.300
haberman	306	4	0.265
heart	270	14	0.444
ilpd	579	11	0.285
ionosphere	351	35	0.359
kr_vs_kp	3196	37	0.478
liver-disorders	145	6	0.379
lupus	87	4	0.402
mammographic	830	5	0.486
mushroom	8124	23	0.482
r2	116	10	0.448
svmguide1	3089	5	0.353
svmguide3	1243	23	0.238
transfusion	748	5	0.238

Tabla 6.2: *Detalle de los conjuntos de datos de clasificación binaria reales. El número de observaciones resulta después de eliminar aquellas que tienen valores faltantes.*

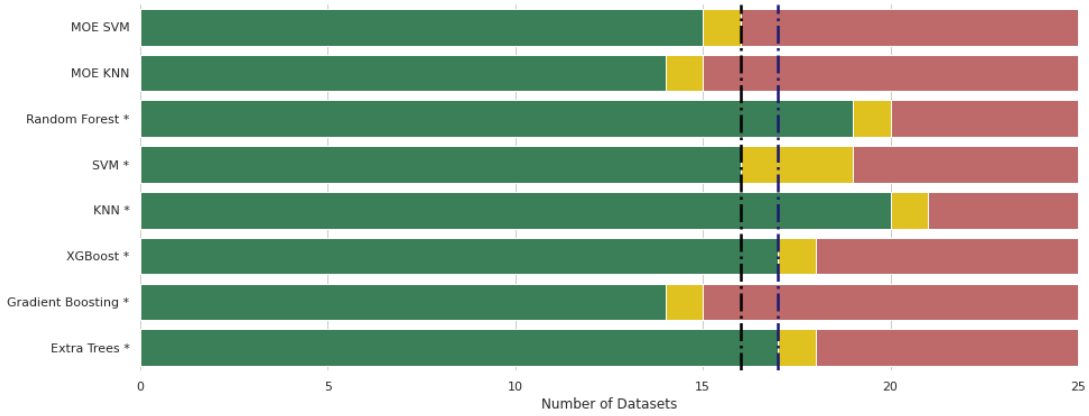
6.2. En este experimento, comparamos los métodos *RF*, *SVM* con kernel de función de base radial, *kNN*, *XGBoost*, *GB* y *ET* con tres variaciones de *MOE* utilizando *DT*, *kNN* y *SVM* como aprendices base. La calidad predictiva de los modelos se evalúa utilizando la medida *MCC*, que es una de las métricas de evaluación más populares en la comunidad *ML* Redondo *et al.* (2020), especialmente para conjuntos de datos desequilibrados. Tenga en cuenta que la proporción de la clase minoritaria en algunos conjuntos de datos es menor al 0,30. Para facilitar la interpretabilidad, *MCC* se ha reescalado para tomar valores entre 0 y 1. Los hiperparámetros se optimizan utilizando búsqueda exhaustiva con validación cruzada de 10 particiones. La lista de hiperparámetros se detalla en la tabla 6.3 (si un parámetro no se menciona, entonces se selecciona el valor predeterminado).

La media y la desviación estándar de la puntuación *MCC* en las particiones de test de la validación cruzada de 10 particiones para la mejor combinación de hiperparámetros se presentan en la tabla 6.4. Los resultados globales muestran que los métodos *MOE-DT* y *MOE-KNN* fueron los mejores en 8 de 25 conjuntos de datos,

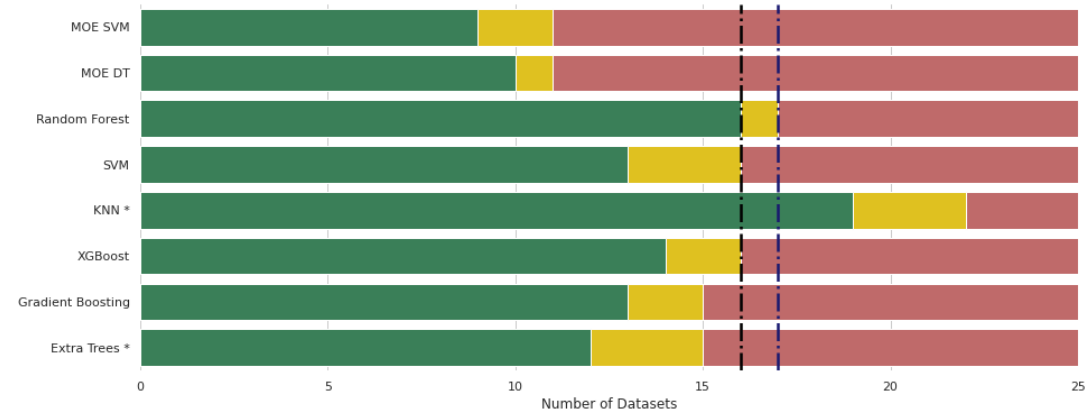
Clasificador	Hiperparámetros	Valores
MOE	Wrab	True, False
	λ	1, 3, 5
	P. muestra	0.10, 0.30, 0.50
	N. aprendices	10, 20, 30
KNN	k	1, 3, ..., 11
SVM	C	1, 10, 100, 1000
	gamma	0.0001, 0.001, 0.01, 0.1, 1, 10
Random Forest	Max. características	None, sqrt, log2
	N. estimadores	100, 300, 500
XGBoost	Max. profundidad	3, 5, 7
	Eta	0.1, 0.2, 0.3
	N. estimadores	100, 300, 500
Extra Trees	Max. características	None, sqrt, log2
	N. estimadores	100, 300, 500
Gradient Boosting	Max. profundidad	3, 5, 7
	Tasa de aprendizaje	0.1, 0.2, 0.3
	N. estimadores	100, 300, 500
	Max. caracterítiscas	None, sqrt, log2

Tabla 6.3: *Métodos de* ML *utilizados y sus valores de hiperparámetros seleccionados.*

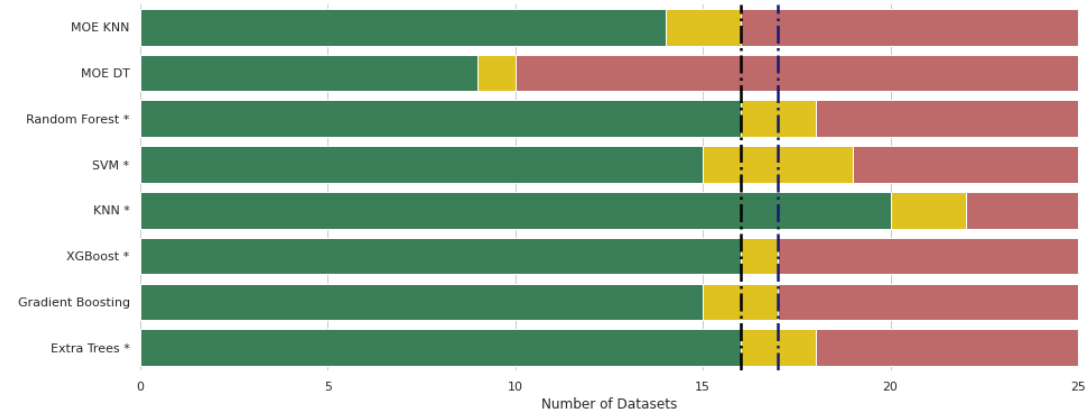
siendo el *MOE-DT* el mejor en términos de la media de *MCC*. Téngase en cuenta que la desviación estándar en todos los métodos es bastante similar, por lo que podemos comparar las medidas de rendimiento medias en general. El Sign Test y el Wilcoxon Signed Rank Test Demšar (2006) se utilizan para la comparación pareada de conjuntos de datos globalmente para evaluar el rendimiento individual de cada variante *MOE*. Con respecto al Sign Test, la hipótesis nula es que la variante *MOE* que se estudia es equivalente al método con el que se compara, y un rechazo de la hipótesis nula es que la variante *MOE* supera al método alternativo. Para rechazar la hipótesis nula, el número de victorias debe ser mayor o igual al valor crítico. En el caso de empates, estos se dividen entre los dos métodos. La Figura 6.5 (basada en D. V. Oliveira *et al.* (2017)) presenta el rendimiento de las variaciones *MOE* en términos de victorias, empates y derrotas.



(a) MOE-DT.



(b) MOE-KNN.



(c) MOE-SVM.

Figura 6.5: *Rendimiento de cada variante* MOE *en términos de victorias (verde), empates (amarillo) y derrotas (rojo). Las líneas verticales discontinuas representan los valores críticos,* 16 *y* 17 *para dos niveles de confianza diferentes,* $\alpha = 0{,}05, 0{,}10$*, respectivamente. Los métodos marcados con * representan aquellos donde la variante* MOE *correspondiente ha superado significativamente a ellos utilizando el Wilcoxon Signed Rank Test* ($\alpha = 0{,}05$)*.*

	MOE SVM	**MOE KNN**	**MOE DT**	**Random Forest**	**SVM**	**KNN**	**XGBoost**	**Gradient Boosting**	**Extra Trees**
appendicitis	0.788 (0.177)	0.810 (0.146)	**0.818 (0.161)**	0.796 (0.150)	0.810 (0.146)	0.798 (0.135)	0.791 (0.148)	0.801 (0.157)	0.796 (0.150)
australian	0.871 (0.027)	0.862 (0.048)	0.881 (0.031)	0.871 (0.034)	0.871 (0.029)	0.843 (0.061)	**0.885 (0.022)**	0.876 (0.032)	0.865 (0.039)
backache	0.500 (0.000)	**0.653 (0.123)**	0.626 (0.139)	0.575 (0.163)	0.600 (0.147)	0.583 (0.143)	0.607 (0.154)	0.607 (0.170)	0.609 (0.162)
banknote	**1.000 (0.000)**	0.999 (0.002)	0.992 (0.008)	0.993 (0.005)	**1.000 (0.000)**	0.999 (0.003)	0.999 (0.003)	0.999 (0.003)	0.999 (0.002)
breastcancer	**0.982 (0.018)**	0.971 (0.017)	0.961 (0.024)	0.963 (0.019)	0.978 (0.018)	0.970 (0.019)	0.972 (0.021)	0.974 (0.012)	0.974 (0.009)
bupa	0.656 (0.109)	**0.661 (0.049)**	0.640 (0.069)	0.591 (0.101)	0.645 (0.091)	0.612 (0.110)	0.607 (0.094)	0.606 (0.093)	0.583 (0.080)
cleve	0.830 (0.067)	**0.838 (0.078)**	0.822 (0.046)	0.827 (0.060)	0.813 (0.063)	0.832 (0.073)	0.834 (0.034)	0.830 (0.043)	0.832 (0.070)
colon-cancer	0.876 (0.086)	**0.913 (0.092)**	0.886 (0.082)	0.877 (0.086)	0.862 (0.076)	0.775 (0.151)	0.859 (0.077)	0.877 (0.086)	0.887 (0.076)
diabetes	0.748 (0.034)	0.749 (0.033)	**0.751 (0.045)**	0.747 (0.033)	0.746 (0.041)	0.707 (0.034)	0.731 (0.037)	0.746 (0.046)	0.747 (0.036)
flare	0.639 (0.058)	**0.662 (0.045)**	0.658 (0.055)	0.617 (0.068)	0.613 (0.085)	0.611 (0.062)	0.632 (0.069)	0.633 (0.058)	0.586 (0.066)
fourclass	**1.000 (0.000)**	**1.000 (0.000)**	0.995 (0.008)	0.996 (0.006)	**1.000 (0.000)**	**1.000 (0.000)**	0.991 (0.010)	0.996 (0.008)	**1.000 (0.000)**
german_numer	0.719 (0.032)	0.688 (0.038)	**0.720 (0.044)**	0.716 (0.033)	0.713 (0.027)	0.633 (0.042)	0.715 (0.050)	0.715 (0.048)	0.692 (0.046)
haberman	0.648 (0.123)	0.632 (0.102)	**0.653 (0.119)**	0.583 (0.078)	0.595 (0.089)	0.625 (0.089)	0.580 (0.110)	0.576 (0.091)	0.571 (0.059)
heart	**0.868 (0.056)**	0.861 (0.066)	0.847 (0.049)	0.837 (0.064)	0.854 (0.059)	0.834 (0.070)	0.814 (0.055)	0.823 (0.052)	0.846 (0.058)
ilpd	0.662 (0.062)	0.643 (0.063)	**0.665 (0.040)**	0.628 (0.049)	0.626 (0.055)	0.625 (0.086)	0.622 (0.068)	0.655 (0.073)	0.658 (0.041)
ionosphere	0.954 (0.042)	0.899 (0.059)	0.941 (0.051)	0.929 (0.039)	**0.955 (0.056)**	0.870 (0.050)	0.924 (0.052)	0.943 (0.041)	0.939 (0.039)
kr_vs_kp	0.993 (0.005)	0.947 (0.014)	0.994 (0.003)	0.995 (0.005)	0.994 (0.004)	0.953 (0.014)	0.995 (0.004)	**0.997 (0.004)**	0.996 (0.004)
liver-disorders	0.748 (0.058)	0.752 (0.055)	0.780 (0.105)	0.758 (0.097)	0.724 (0.084)	0.715 (0.133)	0.777 (0.102)	**0.785 (0.088)**	0.757 (0.054)
lupus	0.770 (0.123)	0.757 (0.190)	**0.780 (0.116)**	0.744 (0.198)	0.743 (0.126)	0.742 (0.178)	0.712 (0.160)	0.716 (0.152)	0.709 (0.208)
mammographic	0.815 (0.040)	0.814 (0.032)	**0.821 (0.035)**	0.789 (0.023)	0.810 (0.036)	0.802 (0.028)	0.813 (0.035)	0.807 (0.037)	0.756 (0.042)
mushroom	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**
r2	**0.798 (0.133)**	0.749 (0.104)	0.781 (0.117)	0.765 (0.105)	0.770 (0.150)	0.722 (0.170)	0.756 (0.105)	0.785 (0.087)	0.784 (0.133)
svmguide1	0.964 (0.011)	0.959 (0.009)	0.968 (0.008)	0.966 (0.010)	0.968 (0.009)	0.960 (0.007)	**0.969 (0.010)**	0.971 (0.011)	0.966 (0.008)
svmguide3	0.764 (0.054)	0.694 (0.056)	0.775 (0.049)	0.766 (0.060)	0.767 (0.043)	0.708 (0.056)	**0.796 (0.051)**	0.785 (0.048)	0.758 (0.050)
transfusion	0.684 (0.051)	**0.686 (0.061)**	0.683 (0.056)	0.626 (0.074)	0.676 (0.048)	0.662 (0.046)	0.644 (0.043)	0.651 (0.043)	0.602 (0.078)
Mean	0.811 (0.055)	0.808 (0.059)	**0.818 (0.058)**	0.798 (0.062)	0.805 (0.059)	0.783 (0.070)	0.801 (0.061)	0.806 (0.059)	0.796 (0.060)

Tabla 6.4: *La media de la* MCC *escalada, con la desviación estándar entre paréntesis, calculada en cada partición de test de la validación cruzada de los nueve métodos comparados. Los mejores resultados se presentan en negrita para cada conjunto de datos.*

En particular, el *MOE-DT* supera a *RF*, *SVM*, *kNN*, *XGBoost* y *ET* con confianza estadística utilizando $\alpha = 0{,}05$, y no se garantiza que sea mejor que el *MOE-SVM*, el *MOE-KNN* y *GB*. El *MOE-KNN* supera a *kNN* con confianza estadística utilizando $\alpha = 0{,}05$ y, *RF* con confianza estadística utilizando $\alpha = 0{,}10$. Sin embargo, no se garantiza que sea mejor que el *MOE-SVM*, el *MOE-DT*, *SVM*, *XGBoost*, *GB* y *ET*. Finalmente, el *MOE-SVM* supera a *RF*, *SVM*, *kNN* y *ET* con confianza estadística utilizando $\alpha = 0{,}05$ y, *XGBoost* y *GB* con confianza estadística utilizando $\alpha = 0{,}10$, y no se garantiza que sea mejor que el *MOE-KNN*, el *MOE-DT*. Finalmente, utilizando el Wilcoxon Signed Rank Test, con $\alpha = 0{,}05$, se confirma que el *MOE-DT* supera a los demás métodos, con la excepción del *MOE-KNN* y el *MOE-SVM*. El *MOE-SVM* supera a los demás métodos, con la excepción del *MOE-KNN*, el *MOE-SVM* y *GB*. Además, el *MOE-SVM* solo supera a *kNN*.

Globalmente, las variaciones *MOE* tienen un buen rendimiento en los experimentos. Este hecho confirma la correcta construcción de los aprendices limitados en términos de precisión y diversidad. El *MOE-SVM* y el *MOE-KNN* superan a la versión única de los algoritmos en casi todos los ejemplos, y el *MOE-DT* supera al *RF* que también utiliza aprendices limitados de acuerdo con la Definición 7. Además, el *MOE-DT* gana de manera más clara sobre el resto. Esa aparente ventaja del *MOE-DT* podría ser producida por la tendencia al sobreajuste de los *DT*s, junto con su inestabilidad. Por lo tanto, los *DT*s proporcionan tanto diversidad como precisión entre todos los aprendices limitados. Esas apreciaciones se derivan de los mejores hiperparámetros seleccionados para cada versión diferente de *MOE*, que se pueden encontrar en la tabla 6.5.

Hiperparámetros	VAlor	MOE-SVM	MOE-DT	MOE-kNN
Wrab	False	8	9	11
	True	17	16	14
λ	1	6	15	7
	3	6	5	5
	5	13	5	13
N. Aprendices	10	5	3	6
	20	9	11	9
	30	11	11	10
P. Sample	0.10	10	7	10
	0.20	8	7	5
	0.50	7	11	10

Tabla 6.5: *Frecuencia de los mejores hiperparámetros para* MOE.

Nótese que la muestra *WRAB* se selecciona el 64 % del total de ocasiones en el *MOE-DT*, en comparación con el 68 % y el 56 % en el *MOE-SVM* y el *MOE-kNN*, respectivamente. Por lo tanto, para un modelo estable, la muestra *WRAB* es una excelente mejora en la diversidad deseada, pero también funciona bien con modelos inestables como los *DT*s. Los valores seleccionados de λ presentan dos tendencias. El *MOE-kNN* y el *MOE-SVM* muestran una predilección por el valor más alto. El ensamble de *SVM*s y *kNN*s mínimamente sobreajustados se benefician de menos sobreajuste en general. Sin embargo, el *MOE-DT* presenta una inclinación por el valor más bajo. No hay un valor óptimo claro para las tres variantes *MOE* entre 20 y 30, pero estos son preferidos sobre 10. Tenga en cuenta que, a medida que aumenta la complejidad del problema, se necesitan más aprendices para cubrir todo el conjunto de aprendizaje. Finalmente, el *MOE-kNN* tiende a no seleccionar el 0,20, sin diferencias entre el 0,10 y el 0,30. El *MOE-DT* selecciona ligeramente más el 0,30, y el *MOE-SVM* selecciona ligeramente más el 0,10, en ningún caso es una diferencia muy sustancial.

6.4. Lecciones aprendidas

Esta sección sintetiza las principales lecciones aprendidas de la propuesta en los casos de estudio.

Una de las fortalezas de los ensamblados generativos basados en muestreo radica en la diversidad de las diferentes regiones de decisión generadas por los modelos base ajustados en cada muestra. Si se utilizan suficientes modelos base en una variedad de muestras, cubriendo todo el conjunto de aprendizaje con una alta diversidad, la combinación resultante de predicciones superará el resultado individual. Cada modelo debe garantizar una precisión suficiente.

La diversidad de las regiones de decisión se podría alcanzar utilizando técnicas de remuestreo. Sin embargo, en algoritmos estables, las perturbaciones en el conjunto de aprendizaje generadas utilizando métodos de remuestreo como *Bootstrap* no son

suficientes para cambiar significativamente las funciones de decisión resultantes. Es posible generar muestras más diversas en problemas de clasificación al muestrear aleatoriamente sobre las etiquetas, generando así funciones de decisión más diversas.

En este capitulo se demuestra que el método de muestreo *WRAB* es mejor que el método de muestreo *bootstrap* en el marco de *MOE* para problemas de clasificación binaria. Se realiza una prueba de Wilcoxon para evaluar rigurosamente las diferencias entre *WRAB* y *bootstrap* en 25 conjuntos de datos. Para probar la hipótesis nula de que no hay diferencias en el rendimiento, aplicamos la prueba de dos colas, lo que nos da un $p - valor = 0{,}0092$. Por lo tanto, se puede rechazar la hipótesis nula a un nivel de confianza del 95 %, concluyendo que hay una diferencia en el rendimiento entre los dos métodos. Para confirmar que la mediana de las diferencias se puede asumir positiva, utilizamos la prueba de una cola, obteniendo un $p - valor = 0{,}0046$. Esto muestra que se puede rechazar la hipótesis nula de que la mediana es negativa a un nivel de confianza del 95 % a favor de la alternativa de que la mediana es mayor que cero. En resumen, se puede concluir que el método de muestreo *WRAB* obtiene mejores resultados que el método de muestreo *bootstrap* en este contexto.

En resumen, se recomienda seguir los resultados experimentales y utilizar el método de muestreo *WRAB* y 30 aprendices como valores predeterminados para los hiperparámetros del *MOE*. En la mayoría de los casos, estos valores funcionan bien. Como punto de partida, se debe establecer la proporción de la muestra en $0,10$ para el *MOE-SVM* y en $0,50$ para el *MOE-kNN* y el *MOE-DT*. El parámetro de control del sobreajuste se debe establecer en 1 para el *MOE-DT* y en 5 para el *MOE-SVM* y el *MOE-kNN*. En general, se recomienda modificar el número de aprendices y la proporción de la muestra a valores más bajos antes de cambiar los valores recomendados de λ o *WRAB*.

El mínimo sobreajuste de los clasificadores proporciona un buen rendimiento en el ensamblado, siempre que las muestras cubran gran parte del conjunto de aprendizaje. Las perturbaciones obtenidas por métodos de remuestreo tienen mayor impacto en la función de decisión de un clasificador mínimamente sobreajustado que en uno bien ajustado. Por lo tanto, la función de decisión variará más en un clasificador mínimamente sobreajustado que en uno bien ajustado. Además, el tamaño de la muestra será más pequeño que en otros conjuntos como *RF*, debido a que la diversidad entre los clasificadores es mayor y los clasificadores son suficientemente precisos.

Capítulo 7

Selección de observaciones y complejidad

Hasta el momento hemos obtenido ventajas del muestreo en el entorno incremental y combinado. Sin embargo, una buena selección de muestra en un problema de clasificación estático de un único modelo de *ML*, también puede culminar con una mejora global del rendimiento del modelo. Mediante la identificación de factores importantes como valores atípicos, errores y relaciones entre variables o reconocer datos de influencia, el conjunto de aprendizaje $\mathcal{L}$ se puede filtrar para obtener $S \subseteq \mathcal{L}$ tal que $\textit{rendimiento}(S) \geq \textit{rendimiento}(\mathcal{L})$. Este proceso se llama selección de observaciones (*IS*) y consiste en filtrar el ruido y los datos redundantes (Blachnik, 2019). Sin embargo, en la práctica, *IS* es un compromiso entre maximizar el rendimiento y reducir los datos almacenados (Leyva *et al.*, 2015). Cabe destacar que cuanto más simple es la clasificación de los datos supervisados, más manejable es el problema de *IS*.

A través de la *IS*, se reduce el conjunto de aprendizaje, lo que es útil para disminuir la carga computacional del proceso de aprendizaje algorítmico, y por tanto, incurre en general en una disminución del tiempo de ejecución. Esto es especialmente relevante en clasificadores basados en instancias, ya que para clasificar una sola instancia, estos clasificadores utilizan todo el conjunto de aprendizaje. Los métodos de *IS* pueden comenzar con un subconjunto vacío $S = \emptyset$, o con el conjunto de aprendizaje completo, $S = \mathcal{L}$. La primera aproximación engloba los métodos incrementales y son aquellos que incluyen observaciones durante el proceso de selección. La segunda aproximación engloba los métodos decrementales, que son aquellos que eliminan observaciones a lo largo de la selección.

Basándonos en una taxonomía clásica (Olvera-López *et al.*, 2010), al igual que en la selección de características, se pueden categorizar los métodos de *IS* en dos grupos, dependiendo de si se usa directamente un clasificador o no:

- Envoltura: El criterio de selección se basa en el rendimiento obtenido por un clasificador (comúnmente, se descartan aquellas observaciones que no contribuyen a la mejora del rendimiento del modelo).
- Filtro: El criterio de selección utiliza una función de selección que no se basa en un clasificador.

En particular, los métodos de filtro se basan en diferenciar las observaciones interiores o de las observaciones límite. Las observaciones límite son aquellas cuyo vecino más cercano pertenece a una clase distinta. Las observaciones interiores son aquellas que no son observaciones límite. Para definir la frontera de decisión de un clasificador, las observaciones límite ofrecen información muy útil para definir la frontera que separa las distintas regiones correspondientes a cada clase (Wilson y Martinez, 2000). Consideramos que las observaciones límite son, a priori, más difíciles de clasificar que las observaciones interiores. Podemos decir que son observaciones más complejas en el espacio de características donde las observamos.

Empleamos el término complejidad de una forma natural, tanto en la selección de observaciones como en el ámbito del aprendizaje incremental, para identificar la dificultad de clasificar un conjunto de datos. En los últimos años, se han desarrollado nuevas técnicas para comprender esta complejidad asociada al problema de clasificación y determinar lo desafiante que es encontrar una solución que lo resuelva mediante modelos de *ML*. Estas técnicas se engloban bajo el concepto de medidas de complejidad (Ho y Basu, 2002). Algunas de estas medidas se construyen a partir del nivel de observación, proporcionando un valor de complejidad para cada una de ellas. Estos valores se agregan posteriormente para obtener un valor de complejidad para todo el conjunto de datos. Dado que tanto la complejidad como las medidas de complejidad son conceptos fundamentales, su uso en *IS* se ha abordado desde su origen (Leyva *et al.*, 2014).

Una de las medidas de complejidad más utilizadas es el *kDN* por su fácil implementación y buen rendimiento. El *kDN* de una observación se define como el porcentaje de sus k vecinos más cercanos que pertenecen a otras clases (Smith *et al.*, 2014). Por ejemplo, ha sido aplicado con éxito como un filtro de ruido en un esquema en línea (G. Oliveira *et al.*, 2021). De manera similar, en (Walmsley *et al.*, 2022), los autores filtran las observaciones con un valor *kDN* alto, ya que consideran que éstas son probablemente observaciones atípicas o ruidosas y pueden ser descartadas con mayor seguridad. Otros trabajos utilizan el *kDN* como guía para muestreo informado para mejorar el rendimiento de los modelos de clasificación (Sleeman IV y Krawczyk, 2019).

El *kDN* depende del número de vecinos k. En general se recomienda fijar $k = 5$ (Leyva *et al.*, 2014; Arruda *et al.*, 2020). Por lo tanto, para este valor de k, el valor de

complejidad de cada observación solo puede tomar seis valores fijos posibles. Esto es una limitación en diferentes situaciones como el filtrado de ruido o la ordenación de datos según la complejidad. En consecuencia, para lograr una medida más suave el parámetro k tendría que aumentarse, alejándose de los parámetros recomendados. De hecho, la falta de suavidad resultante del valor de k es una de las críticas del *kDN* (Lancho *et al.*, 2022).

7.1. Medidas de complejidad

Las medidas de complejidad caracterizan la dificultad de un problema de clasificación supervisada. Dado que el rendimiento de la clasificación depende de los datos, estas medidas exploran el conjunto de datos analizando factores que dificultan el rendimiento de los clasificadores, como la forma de la frontera de decisión (por ejemplo, su falta de linealidad), la densidad de las clases, el grado de superposición entre las clases, etc. Esta información de complejidad se ha demostrado muy útil para el proceso de clasificación posterior. Las medidas de complejidad han sido implementadas con éxito para una amplia variedad de propósitos como la selección de características (Sarbazi-Azad *et al.*, 2020; Okimoto y Lorena, 2019), problemas desequilibrados (Barella *et al.*, 2021; X. Zhang *et al.*, 2019), ruido en las etiquetas (Garcia *et al.*, 2015), Big Data (Maillo *et al.*, 2020), recomendación de clasificadores (Song *et al.*, 2012; Brun *et al.*, 2018), etc.

A partir del trabajo original de Ho y Basu (Ho y Basu, 2002), se han propuesto varias medidas de complejidad (Smith *et al.*, 2014; Lancho *et al.*, 2021; Pascual-Triana *et al.*, 2021). Una recompilación de la mayoría de ellas se puede encontrar en (Lorena *et al.*, 2019), donde las medidas de complejidad se clasifican en seis grandes grupos:

- Medidas basadas en características (centradas en la superposición de características entre clases)
- Medidas de linealidad (para determinar la separabilidad lineal de las clases)
- Medidas de vecindario (para analizar la distribución y la superposición de las clases utilizando la distancia entre puntos)
- Medidas de red (para explorar los datos modelados como un gráfico)
- Medidas de dimensionalidad (para relacionar el número de muestras con el número de dimensiones)
- Medidas de desequilibrio de clase (para evaluar el equilibrio entre el tamaño de las clases).

Las medidas de complejidad se presentaron originalmente a nivel de conjunto de datos (Ho y Basu, 2002), ofreciendo un valor de complejidad único para todo el conjunto de datos. La perspectiva por observación de las medidas de complejidad se abordó por primera vez en (Smith *et al.*, 2014). Los autores introdujeron el concepto de *dificultad de instancia*, que es la probabilidad de que una observación sea clasificada incorrectamente, y propusieron un conjunto de medidas de complejidad, llamadas *medidas de dificultad*, cuyo objetivo es entender por qué algunas observaciones son más difíciles de clasificar. Estas medidas se definen a nivel de observación y se pueden promediar también para obtener el nivel de conjunto de datos. Una de estas *medidas de dificultad* es el *kDN* que también se clasifica como medida de vecindario. Además, algunas de las medidas de complejidad originales se han adaptado al nivel de observación (Arruda *et al.*, 2020). Recientemente, se ha abordado un análisis multinivel de la complejidad de los datos, cubriendo el nivel de observación, la clase y el conjunto de datos con una nueva medida de complejidad propuesta llamada "medida de hostilidad" (Lancho *et al.*, 2022).

Antes de abordar el nivel de observación, desde un enfoque global del conjunto de datos, las medidas de complejidad se aplicaban en tareas como *IS* (Leyva *et al.*, 2014; Cummins y Bridge, 2011) o para evaluar si merece la pena aplicar un filtro de ruido a las observaciones o no (Sáez *et al.*, 2013). Sin embargo, la perspectiva por observación de la complejidad de los datos ha fomentado su uso en las tareas directamente relacionadas con *IS* empleándose como filtro de ruido o para crear mecanismos de muestreo de datos. Por ejemplo, en (Lancho *et al.*, 2022), filtran un 10 % y un 50 % de los puntos más complejos, reduciendo el error. En (T. Zhou *et al.*, 2020), se emplea la complejidad de los datos para el aprendizaje por programación. Presentan el concepto de "dificultad de observación dinámica" para analizar la evolución del rendimiento de una red neuronal profunda en muestras a lo largo del proceso de aprendizaje. En (Chongomweru y Kasem, 2021), proponen un nuevo método de ensamblado para datos desequilibrados en los que las observaciones se seleccionan en función de la "dificultad de observación". Para cada muestra *bootstrap*, se consideran todas las observaciones de la clase minoritaria y se define la probabilidad de que cada observación mayoritaria sea seleccionada por su nivel de dificultad. Este procedimiento asegura muestras *bootstrap* diversas en cuanto a la complejidad. Además, basándose en estos valores de dificultad y para garantizar que los clasificadores aprendan de muestras con diferentes grados de complejidad, en (Kabir *et al.*, 2018) se mide la probabilidad de muestreo de cada observación.

Entre estas medidas de complejidad, se puede destacar el *kDN* debido a su rendimiento satisfactorio en diferentes investigaciones. Por ejemplo, se ha utilizado para construir pesos para seleccionar observaciones de la clase minoritaria en la construcción de un ensamblado jerárquico (Xie *et al.*, 2022). Utilizando la información proporcionada por el *kDN* y el error de cada observación, presentan dos estrate-

gias para seleccionar muestras minoritarias: asumiendo que las observaciones más complejas ofrecen más información o asumiendo que las observaciones más fáciles contienen más información. En (G. Oliveira *et al.*, 2021), presentan un enfoque para tratar tanto los cambios de concepto reales como virtuales, donde se utiliza el *kDN* como un filtro de ruido en dos pasos diferentes. Consideran que las observaciones con un *kDN* mayor de 0,8 (usando $k = 5$) no deben ser consideradas ya que el 80 % de sus vecinos no son de su misma clase y, por lo tanto, son observaciones ruidosas que potencialmente pueden dañar al sistema. En (Walmsley *et al.*, 2022), realizaron un análisis de cómo el ruido afecta el rendimiento de los algoritmos de selección dinámica. Utilizaron la *kDN* para evitar observaciones ruidosas dando más probabilidad de ser seleccionadas a las observaciones con un valor más bajo de *kDN*.

Capítulo 8

Muestreo mediante vecindarios dinámicos ponderados

En esta capítulo presentamos la medida de complejidad *DDN*, que estima la complejidad de cada observación según la proporción de observaciones de otras clases que pertenecen a su vecindario. La medida *DDN* se diferencia de *kDN* en el cálculo del vecindario, que se basa en el clasificador *NCN*, presentado en (Chaudhuri, 1996). Se calculan de manera dinámica, dependiendo de la distribución de datos. La idea principal es que el vecindario es más grande en áreas de baja densidad y más pequeño en áreas de alta densidad. Además, el valor de complejidad de cada observación se pondera en función de la distancia entre los vecinos y la observación bajo evaluación (Aceña *et al.*, 2019). Los vecindarios dinámicos y el paso de ponderación suavizan los valores de la medida de complejidad al mismo tiempo que preservan el excelente comportamiento en el conjunto de datos. *DDN* supera las limitaciones de *kDN* y preserva sus resultados satisfactorios. En particular, este capítulo se centra en el análisis de la medida *DDN* y su uso para abordar el problema de *IS* contemplando como base el *kDN*.

8.1. *Dynamic Disagreeing Neighbors*

En esta sección se presenta la medida de complejidad *DDN*. Esta medida proporciona información valiosa sobre la complejidad de los datos en tres niveles: observación, clase y conjunto de datos. Se evalúa inicialmente para el nivel de observación y sus valores también se agrupan para obtener un valor de complejidad para la perspectiva de clase y conjunto de datos. El objetivo de la medida propuesta es analizar cómo cada punto es afectado negativamente por su vecindario desde un punto de vista de clasificación, es decir, hasta qué punto el entorno de un punto va a dificultar

su clasificación. Las principales ventajas de *DDN* son: (1) los vecindarios dinámicos, cuya definición depende de la densidad de los puntos, y (2) la ponderación dependiente de la distancia en el cálculo del valor de complejidad de un punto, los puntos más cercanos tendrán más peso. Esto garantiza un vecindario representativo para cada observación y calibra correctamente la influencia de las observaciones. En particular, la construcción de los vecindarios *DDN*, derivados del clasificador *NCN*, se basa en los k vecinos soporte y se describe formalmente a continuación.

Por simplicidad, consideramos $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ la matriz de observaciones de entrada e $\boldsymbol{Y} = \{y_i\}_{i=1}^n$ el vector de etiquetas correspondiente, donde $y_i \in \mathcal{G}$ y $\mathcal{G}$ es el conjunto de clases. En estos términos, los k vecinos soporte $S_k = \{\boldsymbol{x}_j\}_{j=1}^k$ de una observación $\boldsymbol{x_i} \in \boldsymbol{X}$ son aquellos que minimizan la expresión:

$$d\left(\boldsymbol{x}_i, \frac{\sum_{j=1}^k \boldsymbol{x}_j}{k}\right) \tag{8.1}$$

siendo $d(\cdot,\cdot)$ una función de distancia, típicamente la distancia Euclídea.

La observación $\boldsymbol{x}_j \in S_k$ siendo la más alejada de $\boldsymbol{x_i}$ proporciona el radio r que define el vecindario:

$$N_k(\boldsymbol{x}_i) = \{\{\boldsymbol{x}_j\}_{j=1}^m \mid d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq r \wedge \boldsymbol{x}_j \in \boldsymbol{X} \setminus \{\boldsymbol{x}_i\}\} \tag{8.2}$$

Este tipo de vecindarios contienen $m \geq k$ observaciones y su radio varía dependiendo de la densidad. En áreas de alta densidad, se espera que m tome valores más cercanos a k, es decir, el radio es pequeño. Por el contrario, en áreas de baja densidad, el radio tendrá valores más altos y es habitual que $m > k$.

Este efecto adaptativo se ilustra en la figura 8.1. Aquellos puntos en áreas más densas tienen vecindarios más pequeños y los puntos en zonas con más dispersión presentan vecindarios más grandes. Esta flexibilidad permite que la medida de complejidad *DDN* capte la distribución de clases de la zona que afecta a cada punto.

Una vez que hemos determinado el vecindario de una observación $\boldsymbol{x}_i$, se puede estimar su valor de complejidad. En este cálculo, el peso de cada observación en su vecindario $N_k(\boldsymbol{x}_i)$ dependerá de su distancia a $\boldsymbol{x}_i$. Esta ponderación por distancia asegura que las observaciones más alejadas no contribuyen más que las más cercanas en el cálculo de complejidad en el caso de vecindarios con gran radio. Así, la influencia de cada vecino en la complejidad disminuye con la distancia, lo que proporciona información más precisa.

La complejidad de una observación, $c(\boldsymbol{x}_i)$, es la proporción de puntos en su vecindario con una etiqueta diferente a y_i, calculada ponderando la distancia con la función *softmax*.

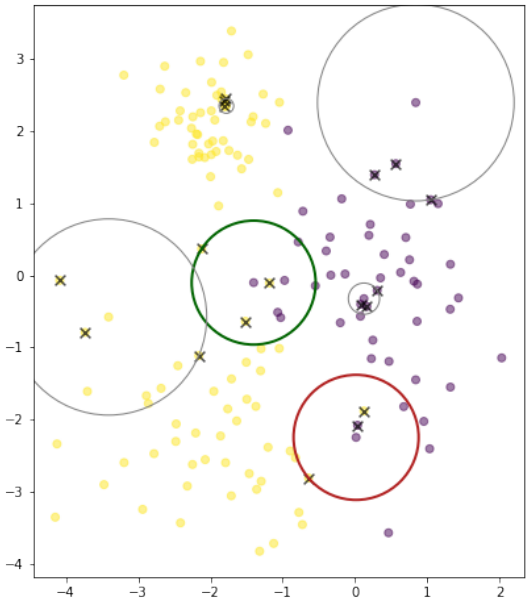


Figura 8.1: *Ejemplos de seis vecindarios de diferentes tamaños según la distribución de datos. Una circunferencia los simboliza y las circunferencias coloreadas son los vecindarios con observaciones que pertenecen a las dos clases. Los tres vecinos de soporte se marcan con una × negra. Finalmente, el color de los puntos representa las dos etiquetas de un problema de clasificación binaria.*

$$c(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in N_k(\boldsymbol{x}_i)} \frac{e^{-\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2}}{\sum_{\boldsymbol{x}_j \in N_k(\boldsymbol{x}_i)} e^{-\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2}} \cdot I_{\{y_i \neq y_j\}} \tag{8.3}$$

siendo $\|\cdot\|$ la norma l^2 y I la función indicatriz que toma el valor 1 cuando su argumento es verdadero y 0 en caso contrario. Por ejemplo, en la figura 8.1 la observación que genera el vecindario verde tiene una complejidad de 0,41 en comparación con el 0,375 que obtendría sin ponderar. El valor de complejidad de 0,41 refleja mejor que, aunque el punto esté cerca de los puntos de su clase, está claramente en el límite entre las clases. Del mismo modo, la observación que genera el vecindario rojo tiene una complejidad de 0,46 en comparación con el 0,5 que obtendría sin utilizar el *softmax*.

El algoritmo 8.1 proporciona el pseudocódigo para calcular la medida de complejidad *DDN* para cada observación de $\boldsymbol{X}$. El cálculo de la complejidad del conjunto de datos y los niveles de clase es sencillo mediante el promedio de la complejidad de observación en cada caso. Por lo tanto, la complejidad de la clase G, denotada por c_G, se puede obtener promediando las complejidades correspondientes a esa clase:

Algoritmo 8.1 Dynamic Disagreeing Neighbors

Entrada: $(\boldsymbol{X}, \boldsymbol{Y}, k)$
for all $\boldsymbol{x}_i \in \boldsymbol{X}$ **ejecuta**:
  $C \leftarrow \emptyset$
  $Q \leftarrow \emptyset$
  $\boldsymbol{q}_1 \leftarrow findNN(\boldsymbol{X}, \boldsymbol{x}_i)$ ▷ $\boldsymbol{q}_1$ es el vecino más cercano
  $Q \leftarrow \boldsymbol{q}_1$
  $\boldsymbol{X}_{aux} \leftarrow \boldsymbol{X} \setminus \{\boldsymbol{q}_1\}$
  $r \leftarrow d(\boldsymbol{q}_1, \boldsymbol{x}_i)$
  $s \leftarrow 1$
  while $s \leq k$ **ejecuta**:
    $s \leftarrow s + 1$
    $d_{min} \leftarrow \infty$
    for all $\boldsymbol{x}_j \in \boldsymbol{X}_{aux}$ **ejecuta**:
      $\overline{\boldsymbol{x}} \leftarrow computeCentroid(Q \cup \{\boldsymbol{x}_j\})$
      Si $d(\overline{\boldsymbol{x}}, \boldsymbol{x}_i) \leq d_{min}$ **entonces:**
        $\boldsymbol{q}_s \leftarrow \boldsymbol{x}_j$
        $d_{min} \leftarrow d(\overline{\boldsymbol{x}}, \boldsymbol{x}_i)$ ▷ actualización de la distancia mínima
    $Q \leftarrow Q \cup \{\boldsymbol{q}_s\}$
    $r \leftarrow max(r, d(\boldsymbol{q}_s, \boldsymbol{x}_i))$ ▷ actualización del radio cuando se requiere
    $\boldsymbol{X}_{aux} \leftarrow \boldsymbol{X}_{aux} \setminus \{\boldsymbol{q}_s\}$
  $N \leftarrow computeNeighborhood(\boldsymbol{x}_i, r)$ ▷ Cálculo del vecindario - ecuación (8.2)
  $c(\boldsymbol{x}_i) \leftarrow computeComplexity(N, \boldsymbol{x}_i, y_i)$ ▷ Cálculo de la complejidad - ecuación (8.3)
  $C \leftarrow C \cup \{c(\boldsymbol{x}_i)\}$
Salida: C ▷ C es el conjunto de las complejidades

$$c_G = \frac{\sum_{i=1}^{n} c(\boldsymbol{x}_i)}{n_G} \cdot I_{[y_i=G]} \tag{8.4}$$

donde n_G es la cardinalidad de la clase G. De manera similar, la complejidad a nivel de conjunto de datos, c^*, se puede encontrar promediando la complejidad de todas las observaciones:

$$c^* = \frac{\sum_{i=1}^{n} c(\boldsymbol{x}_i)}{n} \tag{8.5}$$

Por definición, los valores de complejidad que proporciona *DDN* en el nivel de observación toman valores en el intervalo $[0, 1]$. Por lo tanto, en sus otros dos niveles de definición, que son el promedio del nivel de observación, la medida de complejidad de *DDN* también está entre 0 y 1. Este intervalo facilita naturalmente la interpretación de la medida.

8.2. Experimentos

En esta sección evaluamos la medida de complejidad *DDN* propuesta y su validez para *IS*. A lo largo de los experimentos, se comparará con la medida de complejidad *kDN*. La sección comienza con un análisis gráfico del comportamiento de la medida en datos artificiales. Más tarde, se realiza un análisis de estabilidad para diferentes valores de k en conjuntos de datos reales. Finalmente, se aborda el problema de *IS*. El código fuente y los datos para reproducir los experimentos se pueden encontrar en

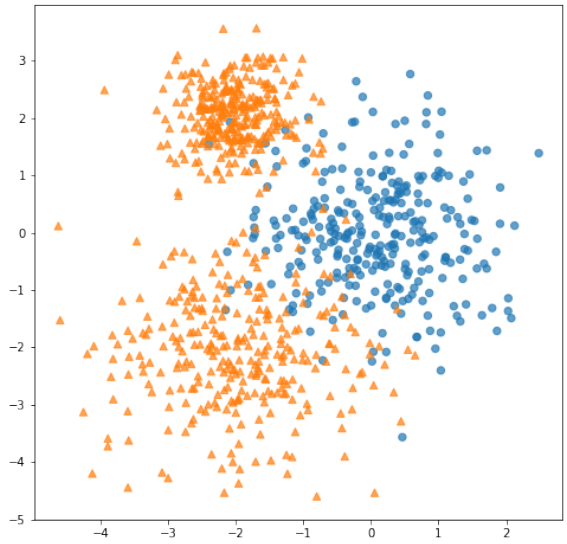


Figura 8.2: *Distribución de datos para el análisis gráfico con* 600 *observaciones para la clase de triángulo naranja y* 300 *para la clase de círculo azul.*

clase	medida	complejidad
0	*DDN*	0,0658
	kDN	0,0663
1	*DDN*	0,1418
	kDN	0,1460

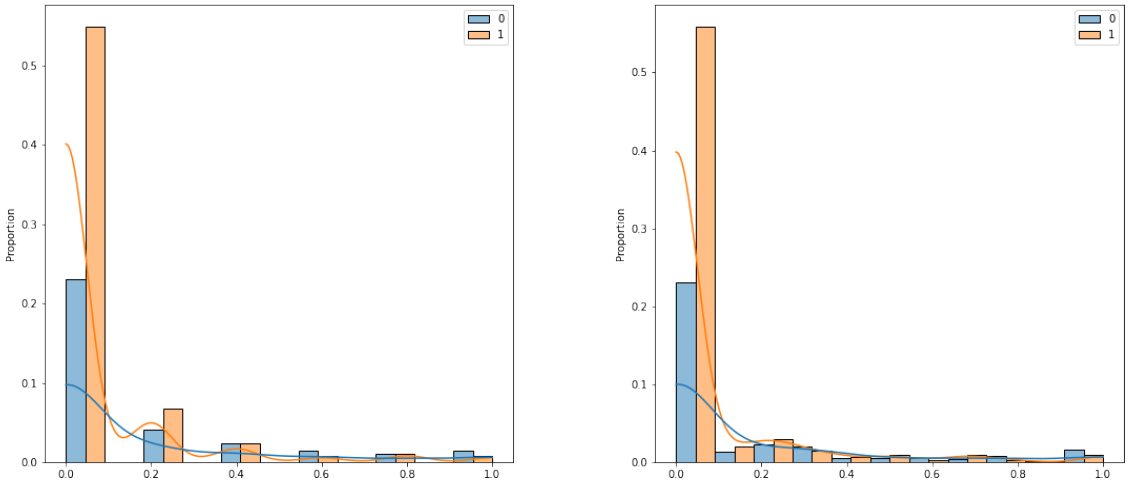
Tabla 8.1: *Una comparación de los valores de complejidad por clase.*

`https://github.com/URJCDSLab/dynamic_disagreeing_neighbors`.

8.2.1. Análisis gráfico en conjuntos artificiales

Esta sección se dedica a la comparación de *DDN* y *kDN* tanto gráficamente como analíticamente en datos artificiales. El experimento evaluará la complejidad a los tres niveles posibles: conjunto de datos, clase y observación. Téngase en cuenta que *kDN* solo está definido originalmente en los niveles de observación y conjunto de datos, pero su nivel de clase también se aborda aquí para analizar adecuadamente ambas medidas. El parámetro k se establece en 5 para *kDN*. Para *DDN*, se han probado $k = 3, 5$ y 7 ofreciendo resultados similares en todos los casos, por tanto se muestra la comparativa con $k = 3$.

En este problema de clasificación binaria tenemos tres distribuciones normales bivariantes, como se muestra en la figura 8.2. Este conjunto de datos artificial presenta diferentes densidades y superposiciones entre las dos clases. A nivel de conjunto

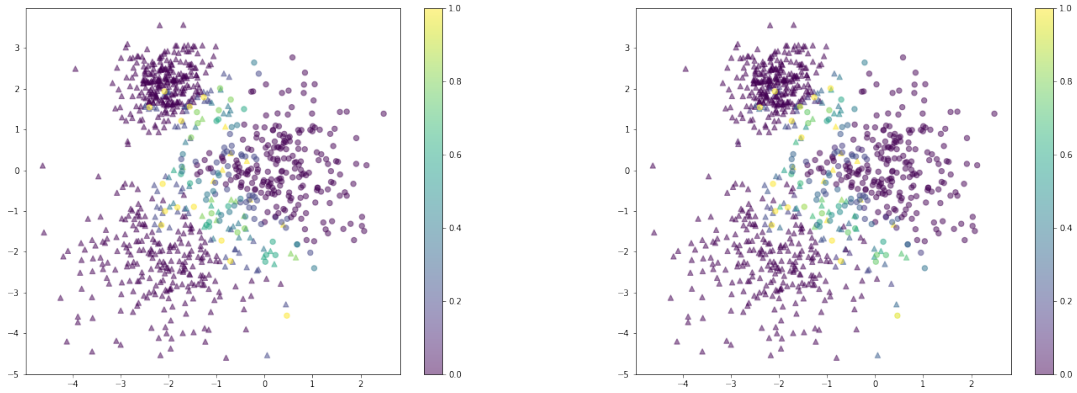


(a) kDN **(b)** DDN

Figura 8.3: *Los dos histogramas de las medidas de complejidad en el nivel de clase, calculados para* kDN *y* DDN.

de datos, *kDN* es 0,0929, y *DDN* es 0,0911. Los dos valores son muy similares, como se espera en este conjunto de datos simple. Las complejidades también son muy similares a nivel de clase, como se puede comprobar en la tabla 8.1. Las similitudes entre las dos medidas se pueden ver al examinar los valores de complejidad asociados al conjunto de datos y las clases, que son ligeramente más bajos para *DDN*. Sin embargo, un análisis más profundo de la complejidad por clase revela diferencias aparentes entre las dos medidas. Como se muestra en la figura 8.3, los valores proporcionados por *DDN* son mucho más suaves que los obtenidos por *kDN*, aunque el valor medio es similar tanto a nivel de conjunto de datos como a nivel de clase. Esto significa que la información recogida por *DDN* se puede considerar globalmente de calidad similar a la de *kDN* en este caso, pero punto a punto, proporciona una información más rica. La complejidad a nivel de observación se detalla en la figura 8.4, donde el gradiente de color representa la diferente complejidad de cada punto. En particular, prestando especial atención a la frontera entre las dos clases, se ve un gradiente más suave para *DDN* en comparación con un gradiente más abrupto para *kDN*. Además, la tabla 8.2 detalla la distribución por deciles para los valores por encima de 0 para ambas métricas, donde se comprueba de nuevo las diferencias entre ambas métricas de complejidad a nivel de observación.

Por lo tanto, se puede concluir que los valores globales entre las dos medidas de complejidad son comparables. Sin embargo, a nivel de observación, *DDN* ofrece una ventaja sobre *kDN* ya que proporciona una mayor variabilidad en el nivel inferior.



(a) kDN **(b)** DDN

Figura 8.4: *Análisis visual de las medidas de complejidad en el nivel de observación calculadas para* kDN *y* DDN. *El color amarillo indica observaciones más complejas. Las dos clases se representan con dos formas geométricas diferentes: triángulos y círculos.*

Measure	**0 %**	**10 %**	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**	**90 %**	**100 %**
DDN	0,02	0,14	0,19	0,22	0,25	0,32	0,37	0,51	0,68	1,00	1,00
kDN	0,20	0,20	0,20	0,20	0,20	0,40	0,40	0,40	0,60	0,84	1,00

Tabla 8.2: *Distribución por deciles de las dos medidas de complejidad.*

Este hecho permite que *DDN* funcione mejor que *kDN* localmente, principalmente cuando la medida de complejidad se utiliza en métodos *IS*, como se mostrará en el último experimento.

8.2.2. Análisis de correlación con el rendimiento en clasificación

En esta sección se evalúa la correlación entre el rendimiento esperado y las medidas de complejidad en una serie de conjuntos de datos de clasificación binaria. Se espera que las medidas de complejidad estén correlacionadas con el error de clasificación. Los conjuntos de datos se han obtenido del repositorio UCI (Dua y Graff, 2017), Penn Machine Learning Benchmarks (Olson *et al.*, 2017) y LIBSVM Data (Chang, 2008). La tabla 8.3 presenta un resumen de sus principales características. Los conjuntos se dividen en entrenamiento y test (80/20), emplearemos el conjunto de test para medir el rendimiento. Se realiza una búsqueda de hiperparámetros con validación cruzada de 5 particiones y se selecciona el mejor de entre los modelos *SVM*, *kNN*, *RF* y *GB*. Finalmente, el rendimiento esperado será el rendimiento obtenido en el conjunto de test del mejor de los cuatro modelos. El rendimiento del modelo se evalúa utilizando la medida *MCC* ya que es una de las pocas métricas que ofrecen

Conjunto	N. observaciones	N. características	Proporción clase minoritaria
a9a	32561	123	0,241
appendicitis	106	7	0,198
australian	690	14	0,445
backache	180	31	0,139
banknote	1372	4	0,445
breastcancer	569	30	0,373
bupa	345	5	0,490
cleve	303	13	0,455
cod-rna	59535	8	0,333
colon-cancer	62	2000	0,355
diabetes	768	8	0,349
flare	1066	10	0,171
fourclass	862	2	0,356
german_numer	1000	24	0,300
haberman	306	3	0,265
heart	270	13	0,444
housevotes84	232	16	0,466
ilpd	579	10	0,285
ionosphere	351	34	0,359
kr_vs_kp	3196	36	0,478
liver-disorders	145	5	0,379
mammographic	830	4	0,486
mushroom	8124	22	0,482
r2	116	9	0,448
sonar	208	60	0,466
splice	1000	60	0,483
svmguide1	3089	4	0,353
svmguide3	1243	22	0,238
transfusion	748	4	0,238
w1a	2477	300	0,029
w2a	3470	300	0,031
w3a	4912	300	0,029
w4a	7366	300	0,029
w5a	9888	300	0,028
w6a	17188	300	0,031
w7a	24692	300	0,030
w8a	49749	300	0,030

Tabla 8.3: *Detalle de los conjuntos de datos reales de clasificación binaria. El número de observaciones resulta después de eliminar aquellas que tienen valores faltantes.*

resultados comparables y de calidad tanto en datos equilibrados como en datos desequilibrados (Chicco y Jurman, 2020). Téngase en cuenta que hay problemas desde equilibrados hasta extremadamente desequilibrados en la muestra de conjuntos de datos seleccionados. En algunos casos, la proporción de la clase minoritaria es inferior al 3 %. Por razones de interpretabilidad, reescalamos *MCC* para tomar valores entre 0 y 1, el rango habitual de la mayoría de medidas de rendimiento.

Nivel de complejidad	k	3	4	5	6	7
conjunto	*DDN*	**0.69**	0,66	0,65	0,65	0,64
	kDN	**0.69**	0,69	0,68	0,68	0,66
clase de mayor complejidad	*DDN*	**0.74**	0,73	0,73	0,73	0,74
	kDN	0,72	0,71	0,70	0,67	0,65
clase de menor complejidad	*DDN*	0,66	0,63	0,63	0,62	0,56
	kDN	**0.68**	0,67	0,66	0,65	0,64

Tabla 8.4: *Correlación de Spearman entre el rendimiento y la complejidad complementaria para diferentes valores de k. Los valores más altos por nivel de complejidad están en negrita.*

Calcularemos la correlación de Spearman entre el rendimiento esperado y la complejidad complementaria (1 - complejidad) para evaluar si los valores de complejidad corresponden al error esperado. La tabla 8.4 muestra la correlación de Spearman para diferentes valores de k a nivel de conjunto de datos y a nivel de clase. La complejidad medida en términos de *kDN* muestra la mayor correlación para $k = 3$ a nivel de conjunto de datos y a nivel de clase para la clase menos compleja. Además, los valores son muy estables, sobre todo a nivel conjunto, para los diferentes k evaluados. Cabe destacar que se logra la mayor correlación con la complejidad de la clase más compleja. En cuanto a *DDN*, muestra la mayor correlación para $k = 3$ y, similar a *kDN*, los valores de correlación son muy estables para los diferentes k probados. Respecto la clase más compleja, se mantiene un valor casi constante para los distintos k, más estable que para *kDN*. Además, para la clase más compleja *DDN* siempre mantiene una correlación más alta que *kDN*.

Estos resultados son consistentes con el estado del arte. En particular, ratifican a $k = 5$ como el valor por defecto para *kDN* dada su estabilidad en los diferentes valores de k evaluados. Además, para ambas medidas, se obtienen las mayores correlaciones con la clase más compleja. Esto sugiere que la complejidad de un problema de clasificación puede estar más relacionada con la complejidad de la clase más compleja que con la complejidad a nivel conjunto de datos, según el trabajo reciente de (Barella *et al.*, 2021). Destaquemos de nuevo que, en el caso de la correlación con la clase de mayor complejidad, *DDN* supera a *kDN* debido a su gran estabilidad a lo largo de los valores de k y a sus valores de correlación más altos en comparación con *kDN*. Dado que *DDN* ofrece una alta correlación con el rendimiento esperado, se puede concluir que proporciona información útil antes del modelo de clasificación a la hora de estimar el error esperado de un conjunto de datos.

8.2.3. Selección de observaciones

En este experimento, se aborda el problema de *IS* mediante las medidas *kDN* y *DDN*. Es decir, se va a evaluar y comparar el rendimiento de ambas medidas de complejidad en la tarea de *IS*. Como se observó en las secciones anteriores y según lo que se conoce hasta el momento, utilizaremos $k = 5$ para *kDN* y $k = 3$ para el caso de

DDN.

Las medidas de complejidad a nivel de observación permiten clasificar el conjunto completo según la complejidad individual. Se puede seleccionar un conjunto de observaciones eligiendo un umbral adecuado que iguale o mejore el rendimiento en comparación con el uso de todo el conjunto de datos. En este trabajo, se construye un heurística de muestreo (*sampling heuristic* (*SH*)) para seleccionar el mejor umbral entre un conjunto de umbrales dados $\{\delta_1, \delta_2, \ldots, \delta_t\}$.

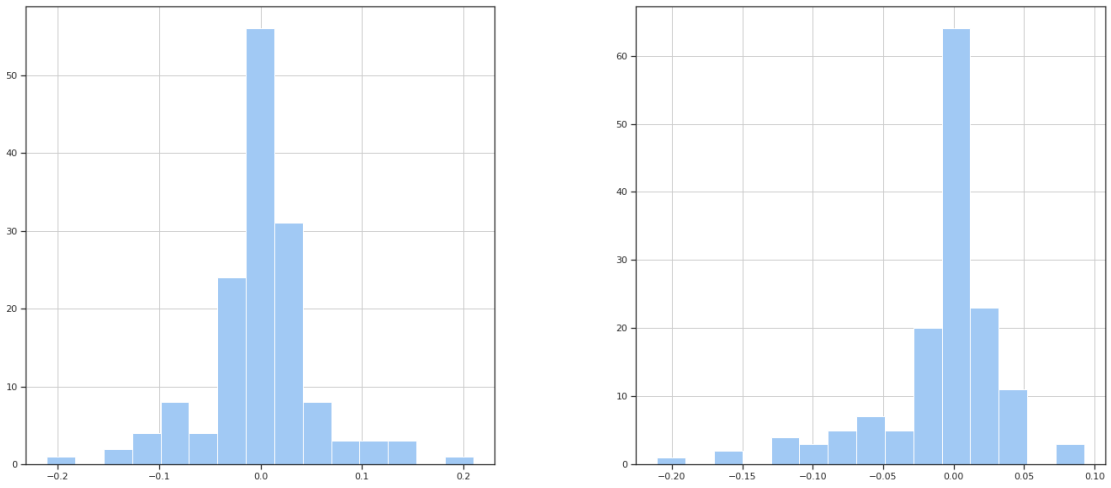
En términos generales, el *SH* funciona de la siguiente manera: dado un umbral δ_i, se descartan las observaciones del conjunto de entrenamiento cuya complejidad sea mayor a δ_i. Luego, se evalúa el rendimiento de un modelo *ML* mediante una validación cruzada estratificada de 3 particiones. Se repite el procedimiento para cada umbral y se selecciona el umbral que mejor funcione. Finalmente, dado el mejor umbral δ^*, se divide el conjunto de entrenamiento por δ^* y se evalúa el resultado en el conjunto de test.

Con el fin de obtener una muestra lo más pequeña posible mientras se asegura que recolecta la información necesaria, se toman todas las observaciones que cumplen con $c(\boldsymbol{x}_i) < \delta_i$ con las siguientes excepciones:

- Proporción de ceros: las observaciones con complejidad cero pueden tener mucho peso a medida que se eliminan las observaciones más complejas. Por lo tanto, de manera análoga a (Aceña *et al.*, 2022b), se elige el mínimo entre el número existente de ceros y una muestra aleatoria de ceros del mismo tamaño que las observaciones con complejidad distinta de cero. Con este paso reequilibramos el conjunto, lo cual puede ser muy importante dependiendo del modelo que entrenemos.

- Baja diferencia entre la complejidad de las clases: si las complejidades de ambas clases son bajas, implica que muchos ceros también contienen mucha información. En este caso, es aconsejable aumentar la proporción de ceros. Para este experimento, se han establecido tres veces más ceros que observaciones con complejidad distinta de cero cuando la diferencia de complejidad entre las clases es menor a l_c. La intención aquí es garantizar que no se da más importancia a los datos extremadamente complejos de lo necesario.

- Mucha diferencia entre la complejidad de las clases: una clase puede ser mucho más compleja que la otra (una diferencia mayor a h_c), lo que suele ocurrir en problemas desequilibrados. En este caso se toma la clase más compleja en su totalidad, incluso para los datos cuyo complejidad excede el umbral delta. En este caso, se intenta garantizar que la información de la clase compleja no se pierda.

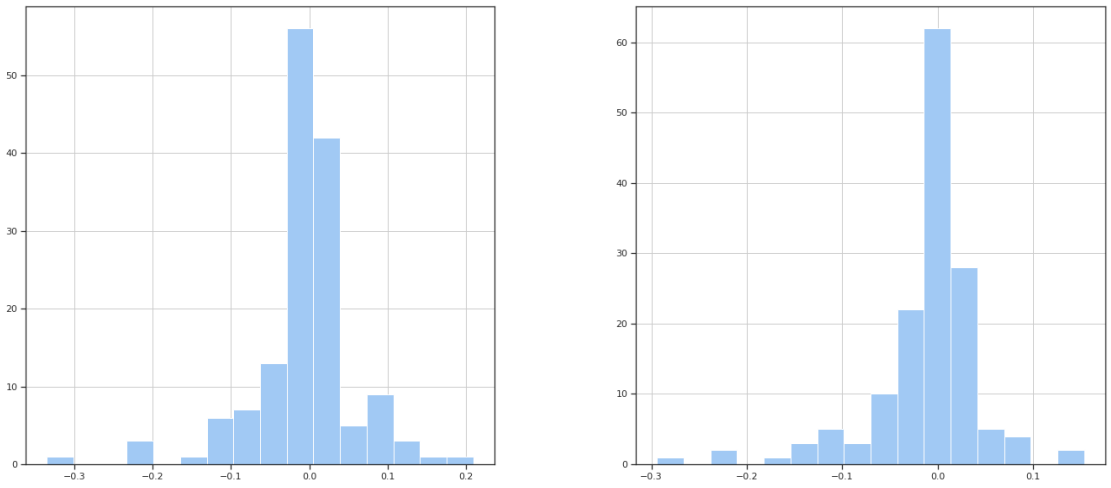


(a) kDN **(b)** DDN

Figura 8.5: *Los dos histogramas de la diferencia del mejor rendimiento para el conjunto de datos completo y el mejor rendimiento para la muestra de* IS *con búsqueda de hiperparámetros globales, calculados para* kDN *y* DDN. *Los valores negativos implican un mejor rendimiento para la muestra de* IS.

Los umbrales para la diferencia de complejidad se han establecido experimentalmente en $l_c = 0{,}15$ cuando la diferencia es pequeña y $h_c = 0{,}25$ en el caso de una gran diferencia entre la complejidad de las clases. En cuanto a la selección de los deltas, se eligen de manera diferente para *DDN* y *kDN* debido a los diversos valores que puede tomar cada medida de complejidad. En el caso de *kDN*, se evalúan todas las posibilidades y se elige la mejor dado que disponemos de únicamente cinco casos. Sin embargo, para *DDN* no es necesario, ni computacionalmente eficiente, evaluar todos los valores posibles. Para ello se sigue un esquema de búsqueda de tipo exploración-explotación. Primero, se toman cinco valores aleatorios de los posibles valores entre 0,05 y 0,95, con un paso de 0,05. Luego, se evalúan estos cinco valores y se selecciona el mejor. Finalmente, de todos los valores posibles, se evalúan los dos más cercanos al que ha resultado mejor. El proceso se repite hasta que no se logra una mejora adicional.

El *SH* que acabamos de definir es un algoritmo para encontrar el umbral óptimo de complejidad, considerando todas las excepciones anteriores, y se utiliza para resolver el problema de *IS* en cuatro modelos *ML*: *SVM*, *kNN*, *RF* y *GB*. En primer lugar, vamos a probar unos hiperparámetros fijos para todos los umbrales. Los hiperparámetros óptimos se encuentran mediante una búsqueda exhaustiva de validación cruzada de 5 particiones en el conjunto de entrenamiento completo para cada modelo.



(a) kDN **(b)** DDN

Figura 8.6: *Los dos histogramas de la diferencia del mejor rendimiento para el conjunto de datos completo y el mejor rendimiento para la muestra de* IS *con la búsqueda de hiperparámetros específicos, calculados para* kDN *y* DDN. *Los valores negativos implican un mejor rendimiento para la muestra de* IS.

La figura 8.5 muestra el histograma de la diferencia en el rendimiento de test entre el mejor resultado entrenado con el conjunto de datos completo y la muestra seleccionada con el *SH* para las dos medidas de complejidad. Para *kDN*, la distribución es menos asimétrica y contiene una mayor proporción de valores positivos que en el caso de *DDN*. Lo que implica un mejor rendimiento de *DDN* en este caso.

Otra alternativa a considerar es seleccionar hiperparámetros específicos del modelo *ML* para cada umbral posible, lo que es mucho más costoso computacionalmente pero puede proporcionar mejores resultados en algunos casos. La figura 8.6 muestra los histogramas de la brecha de rendimiento, incluida la selección de hiperparámetros para los modelos por cada δ_i. En este caso, el histograma correspondiente al *kDN* tiene un mínimo ligeramente inferior. Sin embargo, *DDN* tiene una asimetría a la izquierda mucho mayor, un valor máximo más pequeño y un número menor de valores positivos.

En general, *DDN* funciona mejor bajo ambos métodos de selección de hiperparámetros (global o específico). Sin embargo, no se puede afirmar que uno siempre sea mejor que el otro. Por lo tanto, para una mayor profundidad del análisis combinamos ambas aproximaciones: se seleccionan los hiperparámetros globales o específicos que mejor funcionen en el entrenamiento para cada umbral. Añadimos otra condición para que en el caso de obtener un rendimiento de 1 con un conjunto de

modelo	medida	0 %	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
diferencia de rendimiento	*DDN*	−0,29	−0,08	−0,04	−0,02	−0,01	0,00	0,00	0,01	0,01	0,02	0,09
	kDN	−0,34	−0,08	−0,03	−0,01	0,00	0,00	0,00	0,01	0,02	0,04	0,21
proporción de muestra	*DDN*	0,10	0,57	0,70	0,75	0,80	0,86	0,92	0,97	0,99	1,00	1,00
	kDN	0,08	0,13	0,17	0,48	0,61	0,65	0,78	0,86	0,94	0,99	1,00
umbral	*DDN*	0,10	0,25	0,30	0,41	0,49	0,65	0,75	0,85	0,90	0,95	1,00
	kDN	0,20	0,20	0,20	0,40	0,40	0,60	0,60	0,60	0,80	0,80	1,00

Tabla 8.5: *Comparativa de métricas globales de soluciones combinadas:* kDN *y* DDN *en términos de rendimiento, proporción de muestra y umbral de corte para cada medida de complejidad.*

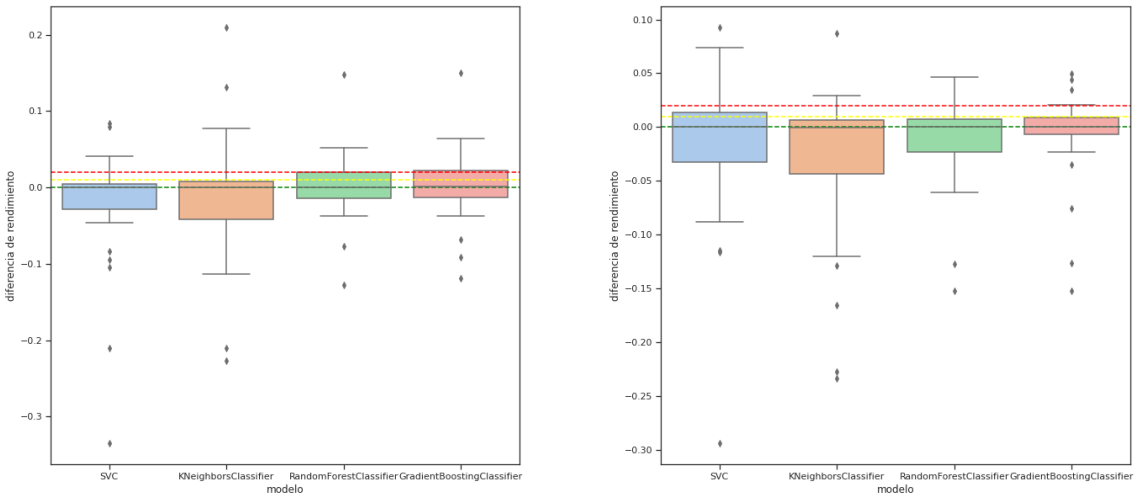
datos completo, no se realice muestreo, asegurando así un alto rendimiento. La tabla 8.5 muestra la distribución por deciles de las métricas de rendimiento para analizar el método combinado para las dos medidas de complejidad. Se puede ver que *DDN* mejora el rendimiento de *kDN* excepto por el valor mínimo obtenido. En general, esto se debe a que *DDN* utiliza una muestra más grande, probablemente debido a la mayor suavidad de sus valores. Esta suavidad también se observa en la variabilidad de los valores tomados para los umbrales de corte.

Por último, en la figura 8.7 podemos observar el comportamiento de cada uno de los modelos en términos de la diferencia de rendimiento del entrenamiento con el conjunto completo frente a la mejor opción de muestreo resultante de comparar la selección de hiperparámetros global y específica. Se aprecia claramente que el *GB* es el modelo menos sensible al ruido, dado que es el que menos mejora para ambas medidas de complejidad. Por el contrario, tanto el *SVM* como el *kNN* son los modelos que se ven más beneficiados del submuestreo, y son aparentemente, los más sensibles al ruido. El *RF* parece que se ve beneficiado del muestreo con *DDN*, pero no sucede lo mismo con el muestreo empleando *kDN*. En términos generales *DDN* como medida de filtrado tiene mejor rendimiento que *kDN* en cada uno de los modelos de manera independiente, aunque siendo menos evidente en el caso del *SVM* si nos fijamos únicamente en la diferencia positiva.

8.3. Lecciones aprendidas

En este capítulo se ha analizado *kDN* a nivel de clase, un estudio que no está presente habitualmente en el estado del arte. Se confirma el buen rendimiento que ofrece esta medida a todos los niveles posibles y confirmando la gran relación que tiene con el error de clasificación. Además, se confirma la dificultad que presenta la elección del k a la hora de representar fielmente la complejidad de las observaciones de manera independiente.

Utilizando como referencia *kDN*, tanto en conjuntos simulados como reales, se garantiza que *DDN* es una medida de complejidad valiosa y robusta. Además, se con-



(a) kDN **(b)** DDN

Figura 8.7: *Dos gráficos de caja y bigotes para las dos medidas de complejidad, donde se observa la diferencia de rendimiento entre usar el conjunto completo y el obtenido para el método combinado para cada modelo. Las líneas punteadas verde, amarilla y roja representan las diferencias positivas* 0, 0,01 *y* 0,02*, respectivamente. Los valores negativos implican un mejor rendimiento para la muestra de* IS.

firma que tanto la definición de los vecindarios como ponderar las observaciones en función de la distancia han sido elecciones acertadas. Esto es gracias a emplear como referencia *kDN* y al comparar la medida propuesta con el error de clasificación, siendo esto último la forma más fiel de evaluar una medida de complejidad.

Respecto del empleo de *DDN* para el problema de *IS*, la heurística definida proporciona soluciones para los diversos métodos de selección de hiperparámetros disponibles, de modo que en más del 60 % de los casos, mejora o iguala el resultado que se puede obtener al considerar el conjunto de datos completo. Además, se ponen de manifiesto las ventajas de una medida de complejidad suave frente a una medida de complejidad no abrupta para *IS*. *DDN* ofrece valores de complejidad más ricos y variados, lo que permite una selección más precisa y efectiva de observaciones.

Finalmente, parece claro que hay observaciones muy complejas que podemos considerar ruido y por tanto eliminarlas. También, hemos visto que si hay una clase mucho más compleja que la otra es probable que casi todas las observaciones contenidas en esa clase sean muy relevantes para definir la frontera de decisión del problema de clasificación. Además, se pone de manifiesto que establecer un método general, que sirva para cualquier modelo de *ML*, para muestrear sobre el conjunto de entrenamiento es muy complejo. Por tanto, lo ideal es tener en cuenta los mecanismos de aprendizaje de cada modelo para aprovecharlos incluyendo la información

adicional que puede extraerse del análisis de complejidad.

Capítulo 9

Conclusiones

En esta tesis, hemos investigado el problema de clasificación desde el enfoque del aprendizaje estadístico, centrándonos en las técnicas de muestreo para identificar las observaciones más relevantes en cada escenario y explotar esta información para mejorar el rendimiento de los modelos en múltiples niveles. La realización de esta investigación ha sido posible gracias a la financiación proporcionada por las ayudas para la realización de Doctorados Industriales de la Comunidad de Madrid. Esto significa que, además de ser un trabajo de investigación académica, los resultados presentados aquí se han trasladado de manera inmediata al entorno empresarial. En particular, se ha conseguido desarrollar un modelo de propensión de compra satisfactorio en un esquema incremental de actualización continua donde se optimiza el almacenamiento de datos. Además, la identificación de clientes clave se convierte en información valiosa que conlleva una mejor gestión de los recursos de la compañía y la mejora directa del proceso de toma de decisiones. Por otra parte, desde la perspectiva académica hemos tratado de resolver las siguientes cuestiones satisfactoriamente:

1. ¿Puede mejorarse el rendimiento de un clasificador en un entorno incremental realizando un bajo muestreo recurrente?
2. En un problema de clasificación desequilibrado, ¿ambas clases aportan la misma información?
3. ¿Es posible identificar las observaciones más relevantes en un problema de clasificación independientemente de clasificador empleado?
4. ¿Se pueden ensamblar algoritmos de *Machine Learning* (*ML*) estables de manera satisfactoria?
5. ¿Es el sobreajuste de los *Decision Tree*s (*DT*s) parte de la explicación del buen funcionamiento de los ensamblados generativos?

El muestreo es el tema central de la presente tesis. En particular, el muestreo en el aprendizaje incremental se aborda con éxito en el capítulo 4, donde se establece el marco teórico necesario y su aplicación práctica. Podemos asegurar que reducir la cantidad de datos que obtenemos a lo largo del tiempo es factible y, además, muy beneficioso en términos de almacenamiento a largo plazo. La dificultad radica en ser capaces de identificar qué datos contienen la información relevante para resolver el problema de clasificación con éxito. Los resultados empíricos nos llevan a concluir que no todos los datos contienen esta información. Además, en el caso de la clasificación desequilibrada, la clase menos representada tiende a disponer de información más valiosa. Esta información se puede evaluar analizando la complejidad de los datos, como se aborda en el capítulo 8, donde se analiza la relevancia de cada observación de manera agnóstica al algoritmo empleado para resolver el problema. De esta manera, llegamos a la conclusión de que cada clase puede tener una relevancia muy distinta a la hora de determinar la complejidad de un problema de clasificación. Además, encontramos que la clase más compleja representa mejor la complejidad global del problema. En otras palabras, ciertas observaciones disponen de información acerca del problema mucho más relevante que otras.

Por otra parte, el muestreo en el aprendizaje ensamblado es la pieza fundamental de los ensamblados generativos. Sin embargo, en el pasado solo parecía funcionar con algoritmos inestables. Esto se debe a que el uso de métodos de remuestreo clásicos no modifica lo suficiente el conjunto de datos. En otras palabras, la información subyacente en el conjunto sigue siendo demasiado similar a la original. En el capítulo 6, abordamos el muestreo en los ensamblados para garantizar la diversidad en los aprendices resultantes de cada muestra. Sabemos que no todos los datos contienen la misma información, por lo que dibujar muestras que contengan información diferente fomenta que los aprendices generen fronteras de decisión lo suficientemente diversas entre sí. Además, el uso del sobreajuste de manera local en cada una de estas muestras evita el sobreajuste global para cualquier algoritmo base en el contexto de ensamblados generativos, confirmando empíricamente la intuición que se desprende del bien conocido algoritmo original del *Random Forest* (*RF*).

A continuación, se presentan las principales contribuciones de esta investigación, junto con posibles líneas de investigación futura. También se incluye un listado de publicaciones en revistas científicas y congresos internacionales que se han derivado de esta investigación, así como de aquellos en los que se ha colaborado de manera paralela.

9.1. Principales contribuciones

Las principales contribuciones de esta investigación se centran en la identificación de observaciones clave en distintos paradigmas de la predicción de etiquetas en el aprendizaje supervisado, como son, el aprendizaje estático, aprendizaje incremental y aprendizaje combinado. Se han logrado tres grandes contribuciones a estas áreas, dando lugar a varios artículos científicos (Aceña *et al.*, 2019, 2022b,a). Estos aportes se corresponden con los capítulos 4, 6 y 8 específicamente.

A continuación, se resumen las principales conclusiones de cada una de las tres contribuciones, que son:

- La introducción del concepto de subconjunto soporte y el diseño de una metodología de reentrenamiento para *Support Vector Machine*s (*SVM*s).
- La definición de aprendicees limitados y la introducción de un nuevo marco *Minimally Overfitted Ensemble* (*MOE*) de aprendizaje combinado basado en el mínimo sobreajuste, el muestreo *Weighted RAndom Bootstrap* (*WRAB*).
- El desarrollo de la nueva medida de complejidad *Dynamic Disagreeing Neighbors* (*DDN*) y su empleo para resolver el problema de *Instance Selection* (*IS*).

Subconjuntos soporte

En (Aceña *et al.*, 2022b), presentamos por primera vez el concepto subconjunto soporte. Un subconjunto de soporte es un subconjunto del conjunto de entrenamiento que, junto con los nuevos datos, produce la misma frontera de decisión que si el modelo se entrenara con el conjunto de datos completo. El concepto de subconjunto soporte ha sido respaldado por resultados teóricos que garantizan su existencia. De particular interés es el mínimo subconjunto soporte, los subconjuntos de tamaño mínimo entre todos los subconjuntos soporte. Además, hemos propuesto un método para estimar un subconjunto soporte mínimo para *SVM* y, basándonos en esta estimación, una nueva metodología de reentrenamiento de *SVM* que implica un tiempo computacional y coste de memoria más bajos que el reentrenamiento con el conjunto de datos completo.

Los resultados experimentales muestran que el método logra una puntuación F_1 similar en escenarios comunes de reentrenamiento, al tiempo que reduce el número de vectores soporte y la cantidad de datos utilizados. Además, estos escenarios de reentrenamiento incluyen problemas complejos como el cambio de distribución y los problemas de clasificación desequilibrados. En estos problemas, el método reduce drásticamente el tiempo de entrenamiento y la cantidad de datos requeridos. Esto

implica que el método es capaz de identificar correctamente las instancias relevantes. Sin embargo, cuando el tamaño del conjunto de datos es grande, el reentrenamiento basado en subconjuntos soporte puede tardar más que el reentrenamiento completo en la primera iteración de una secuencia. Además, cuando el tamaño del conjunto de datos es pequeño, esta situación se puede reproducir en todas las iteraciones. Sin embargo, el incremento relativo de tiempo es insignificante en este último caso. Esta brecha de tiempo es una consecuencia directa de la reducción escasa en el número de vectores soporte, lo que aumenta las etapas de evaluación y reentrenamiento. Finalmente, en algunos casos, el rendimiento no alcanza la calidad de *LibSVM* debido probablemente al muestreo aleatorio en la etapa de remuestreo.

***Minimally Overfitted Ensemble* (*MOE*)**

En (Aceña *et al.*, 2022a), se introduce la noción de aprendiz limitado. Un aprendiz limitado es un aprendiz sobreajustado cuya predicción es mejor que la predicción aleatoria. Basado en esta noción, se propone un marco general de aprendizaje combinado basado en remuestreo llamado *MOE*. Este marco, está diseñado para trabajar con cualquier algoritmo de *ML* como base, tanto estable como inestable. La idea clave detrás de *MOE* es una búsqueda del mínimo sobreeajuste en cada muestra. El resultado son aprendices ligeramente sobreeajustados, un tipo específico de aprendices limitados. Además, los conjuntos de datos para entrenar a cada aprendiz individual se generan utilizando un nuevo método de muestreo. *WRAB* añade una capa encima del método de muestreo *bootstrap* que modifica el equilibrio original entre las clases. De este modo, los aprendices resultantes son más diversos, especialmente cuando se trabaja con algoritmos estables. Los aprendices resultantes construidos por el marco *MOE* se combinan utilizando la regla de voto mayoritario.

El rendimiento de la propuesta es evaluado en varios conjuntos de datos reales con tres algoritmos base diferentes: *SVM*, *DT* y *k-Nearest Neighbour* (*kNN*). Las variantes de *MOE* obtuvieron un mejor rendimiento que los respectivos algoritmos en su versión individual. El mejor rendimiento general ha sido alcanzado por el *MOE-DT*. Además, el análisis de hiperparámetros confirma que la inestabilidad y la tendencia a sobreajustar del algoritmo base beneficia el rendimiento general del marco *MOE*. En algoritmos base estables, el *WRAB* se demuestra eficiente para evitar la limitación de falta de variabilidad, generando aprendices individuales diversos utilizando algoritmos base estables como *SVM* con kernel Gaussiano y superando su versión individual. Además, la propuesta supera ligeramente al método de ensamblado de aprendices limitados de referencia, el *RF*.

***Dynamic Disagreeing Neighbors* (*DDN*)**

En la última contribución presentamos la medida de complejidad *DDN* para los niveles de observación, clase y conjunto de datos. La complejidad de una instancia

es la proporción de observaciones en su vecindario con una etiqueta diferente a esa instancia. Para los niveles agregados de clase y conjunto de datos, se calcula el promedio de complejidad de instancias de cada grupo. Los valores de complejidad con esta medida están contenidos en un rango de 0 a 1. La novedad de esta métrica radica en el cálculo dinámico del vecindario y la ponderación de las observaciones en función de la distancia. Ambos hacen que la medida se adapte a la distribución de clases y considere la densidad alrededor de cada punto. Esta nueva medida de complejidad es conceptualmente similar a la *k-Disagreeing Neighbors* (*kDN*) pero ofrece información más detallada, logrando mayor desagregación en los valores de cada observación. A diferencia de *kDN*, donde los k vecinos más cercanos definen vecindarios y cada punto tiene la misma influencia, en *DDN*, el peso de cada observación depende del tamaño del vecindario y la distancia de la observación observada. *DDN* también tiene un parámetro k que se relaciona con el tamaño del vecindario (número de vecinos de soporte). Como se esperaba, los valores de complejidad están más desagregados a medida que aumenta k en *DDN*.

Se han realizado varias experimentos para evaluar *DDN* y probar su estabilidad, calidad y aplicabilidad al problema *IS*. En estos experimentos, se ha comparado el rendimiento de *DDN* con *kDN* para evaluar sus diferencias y ventajas. Observamos que cuando *DDN* ofrece valores similares a *kDN*, ofrece una ventaja al proporcionar más variabilidad en los valores de complejidad de la observación, lo que lleva a una medida con muchos más niveles de desagregación. Además, ambas medidas se analizan a tres niveles: observación, clase y conjunto de datos. Este análisis muestra que la correlación con el rendimiento esperado es mayor con la complejidad de clase proporcionada por *DDN*. Por último, el experimento final muestra la ventaja de una medida suave como *DDN* en comparación con una medida más abrupta a nivel de observación para realizar *IS*. *DDN* es una medida de complejidad intuitiva y fácil de calcular con resultados muy estables para varios valores de su único parámetro, ofrece resultados de complejidad a nivel de observación que permiten ordenar y filtrar el conjunto de datos según la complejidad de cada observación.

9.2. Preguntas abiertas y oportunidades de mejora

Aunque hemos resuelto muchos problemas en el ámbito de los métodos de muestreo para la mejora del rendimiento en *ML* a lo largo de la investigación, quedan muchos desafíos sin resolver, tanto generales como específicos, en las diferentes contribuciones. Es importante recordar que esta tesis doctoral surge de una colaboración entre universidad y empresa a través de un proyecto de Doctorado Industrial. Por lo tanto, las cuestiones que se exponen a continuación no solo representan una oportunidad de innovación en el ámbito académico, sino que también abren la posibilidad

de nuevos proyectos para entidades privadas, permitiendo así la solución de problemas reales en su día a día. A continuación, se describen las posibles expansiones para cada contribución.

Subconjuntos soporte

El trabajo futuro se centrará en las etapas de muestreo: estimación del subconjunto soporte y remuestreo de los nuevos datos. Evaluaremos un nuevo método para proporcionar una mejor estimación del subconjunto soporte basado en la estimación secuencial o la validación cruzada. Además, podría ser interesante permitir al usuario definir el compromiso entre la calidad del conjunto de datos de reentrenamiento y el uso de recursos computacionales. Con respecto al remuestreo de los nuevos datos, una estrategia de muestreo dinámica que tenga en cuenta la densidad del vecindario de las observaciones reduciría la pérdida de precisión en el entrenamiento con todos los datos. Este muestreo dinámico será beneficioso para adaptarse a nuevas distribuciones y filtrado de datos ruidosos. Se podría utilizar un algoritmo de *clustering* para este propósito. Por lo tanto, el método podría utilizar diferentes estrategias de muestreo dependiendo de la distribución de las observaciones en cada clúster.

***Minimally Overfitted Ensemble* (*MOE*)**

La definición de mínimo sobreajuste necesita un conjunto de modelos de *ML*, lo que aumenta el tiempo de entrenamiento del método y limita los experimentos para grandes conjuntos de datos. Los tiempos de ejecución de las variantes de *MOE* están determinados por la búsqueda exhaustiva de hiperparámetros que se realiza en cada muestra para seleccionar el modelo ligeramente sobreajustado según la Definición 8. Por lo tanto, agregar un gran número de modelos aumenta considerablemente el tiempo de entrenamiento, especialmente cuando se ensamblan modelos muy complejos como *SVM*s. Las mejoras futuras se centrarán en reducir el conjunto de modelos para seleccionar el modelo ligeramente sobreajustado. Por otra parte, el método de muestreo *WRAB* proporciona diversidad en el conjunto cuando el algoritmo base es estable, pero puede no ser suficiente en algunos casos. El tamaño de muestra y el parámetro de sobreajuste podrían ser sensibles al contexto de los datos para garantizar la diversidad de los modelos ligeramente sobreajustados y elegir modelos más o menos estables de aprendizaje automático. Por ejemplo, aumentando el número de muestras en regiones complejas o reduciéndolo cuando se utilizan modelos estables en regiones simples. Una línea de investigación prometedora sería explorar la inclusión dentro del marco de nuevos enfoques para el tratamiento de conjuntos de datos desequilibrados. Finalmente, se estudiará la expansión del marco *MOE* a la clasificación multiclase y la regresión revisando la definición de *Modelo Mínimamente Sobreajustado* y el *WRAB* para estos paradigmas de *ML*.

***Dynamic Disagreeing Neighbors* (*DDN*)**

El trabajo futuro se centra en mejorar la calidad de la información de complejidad proporcionada por *DDN*, que se puede abordar desde diferentes ángulos: remuestreo, evaluación de puntos mal clasificados y regularización. Los métodos de remuestreo se pueden incluir para obtener una medida de complejidad más precisa estimando la distribución empírica de la medida de complejidad. Además, se podría estudiar en base a cada observación los valores proporcionados por *DDN* en relación a las observaciones que distintos modelos de clasificación etiquetan incorrectamente. Finalmente, la solución de remuestreo y el análisis de puntos mal clasificados pueden llevar a un mecanismo de regularización diseñado específicamente para las observaciones que generan más dudas y, por lo tanto, superar al *DDN* actual. Después de realizar estos ajustes, el problema *IS* se podría resolver de manera más precisa con la heurística propuesta ajustando los parámetros de acuerdo a la métrica mejorada.

9.3. Publicaciones realizadas

Contribuciones a revista:

- Aceña, V., de Diego, I. M., Fernández, R. R., & Moguerza, J. M. (2022). Minimally overfitted learners: A general framework for ensemble learning. *Knowledge-Based Systems, 254*, 109669.
- Aceña, V., de Diego, I. M., Fernández, R. R., & Moguerza, J. M. (2023). Support subsets estimation for support vector machines retraining. *Pattern Recognition, 134*, 109117.
- Aceña, V., Lancho, C., de Diego, I. M., Moguerza, J. M., & Dae-Jin L. (2023). Dynamic Disagreeing Neighbors for instance selection. *Applied Intelligence*, (en revisión).

Contribuciones a congreso:

- Aceña, V., Moguerza, J. M., de Diego, I. M., & Fernández, R. R. (2019). Weighted Nearest Centroid Neighbourhood. In *Intelligent Data Engineering and Automated Learning–IDEAL 2019: 20th International Conference, Manchester, UK, November 14–16, 2019, Proceedings, Part I 20* (pp. 94-101). Springer International Publishing.

Apéndice A

Cota superior para el error de generalización en MOE

Minimally Overfitted Ensemble (*MOE*) es una colección de aprendices mínimamente sobreajustados, que es un caso particular de aprendices limitados. Por lo tanto, un *MOE* es un conjunto de aprendices limitados $\{l_k(\boldsymbol{x})\}$ donde $\boldsymbol{x}$ es un vector de entrada y cada aprendiz limitado vota con el mismo peso por la clase más adecuada en la entrada $\boldsymbol{x}$. Cada aprendiz limitado puede ser representado por una función paramétrica $l(\boldsymbol{x}, k)$ donde k es un vector de parámetros definido en términos de las regiones disjuntas del espacio de entrada. Para clasificadores binarios, la función paramétrica tiene la forma:

$$l(\boldsymbol{x}, \theta_k) = \sum_{j=1}^{J} I_{[\boldsymbol{x} \in R_j]} \cdot \hat{y}_j$$

donde R_j define la j-ésima región disjunta del espacio de entrada que es inducida por el clasificador y los y_j representan la etiqueta asignada a la j-ésima región. Por ejemplo, para un *Support Vector Machine* (*SVM*) lineal, el hiperplano separador definido por $sign(\sum_{i \in sv}(\omega^T \boldsymbol{x}_i + b))$ genera dos regiones R_1 y R_2.

Por lo tanto, dada la definición del conjunto en términos de aprendices limitados, el error de generalización para *MOE* es paralelo al análisis de Breiman Breiman (2001) para el *Random Forest.*

Dada una colección de clasificadores $l_1(\boldsymbol{x}), l_2(\boldsymbol{x}), \ldots, l_m(\boldsymbol{x})$ y el conjunto de entrenamiento siguiendo la distribución subyacente de los vectores aleatorios $\boldsymbol{X}, Y$. El

margen de clasificación es:

$$m(\boldsymbol{X}, Y) = P_{\Theta}(l(\boldsymbol{X}, \Theta) = Y) - max_{c \neq Y} P_{\Theta}(l(\boldsymbol{X}, \Theta) = c) \tag{A.1}$$

El error de generalización para un conjunto de votación puede ser limitado mediante la medición de la diversidad y la precisión de los aprendices. La precisión se mide por la fuerza esperada y se define como:

$$PE = P_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y) < 0) \tag{A.2}$$

En el caso binario equilibrado, puede formularse como:

$$m(\boldsymbol{X}, Y) = 2P_{\Theta}(l(\boldsymbol{X}, \Theta) = Y) - 1 \tag{A.3}$$

y cumplir la condición $\mu > 0$ significa que $E_{\boldsymbol{X},Y}(P_{\Theta}(l(\boldsymbol{X}, \Theta) = Y)) > 0{,}5$ y, entonces, se alcanza la condición de un clasificador débil. Los aprendices limitados cumplen esta condición por la Definición 7.

Por lo tanto, si $m(\boldsymbol{X}, Y) > 0$, el conjunto vota por la clase correcta. El error de generalización para un conjunto de aprendices limitados es:

$$PE = P_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y) < 0) \tag{A.4}$$

Aplicando la desigualdad de Chebyshev con $\mu > 0$,

$$PE = \frac{var_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y))}{\mu^2} \tag{A.5}$$

Ahora, vamos a tratar con la expresión para la varianza. Dejemos que

$$\hat{c} = argmax_{c \neq Y} P_{\Theta}(l(\boldsymbol{X}, \Theta) = c) \tag{A.6}$$

sea la clase con más votos incorrectos.

Entonces,

$$m(\boldsymbol{X}, Y) = P_{\Theta}(l(\boldsymbol{X}, \Theta) - P_{\Theta}(l(\boldsymbol{X}, \Theta) = c) = E_{\Theta}(m^*(\boldsymbol{X}, Y, \Theta)) \tag{A.7}$$

donde

$$m^*(\boldsymbol{X}, Y, \theta) = I_{[l(\boldsymbol{x},\theta)=Y]} - I_{[l(\boldsymbol{x},\theta)=\hat{c}]} \tag{A.8}$$

Asumiendo que Θ y Θ' están independientemente e idénticamente distribuidos,

$$\begin{aligned} m(\boldsymbol{X}, Y)^2 &= E_{\Theta}(m^*(\boldsymbol{X}, Y, \Theta))^2 \\ &= E_{\Theta,\Theta'}(m^*(\boldsymbol{X}, Y, \Theta), m^*(\boldsymbol{X}, Y, \Theta')) \end{aligned} \tag{A.9}$$

Por lo tanto,

$$\begin{aligned} var_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y)) &= E_{\Theta,\Theta'}(cov_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta), m^*(\boldsymbol{X}, Y, \Theta'))) \\ &= E_{\Theta,\Theta'}(\rho(\Theta, \Theta')\sigma(\Theta)\sigma(\Theta')) \end{aligned} \tag{A.10}$$

donde,

$$\rho(\theta, \theta') = corr_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta), m^*(\boldsymbol{X}, Y, \theta')) \tag{A.11}$$

es la correlación entre dos aprendices del conjunto, para θ y θ' fijos, y $\sigma(\theta)$ es la raíz cuadrada de

$$\sigma^2(\theta) = var_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta)) \tag{A.12}$$

De (A.10) y la definición de varianza,

$$var_{X,Y}(m(X, Y)) = \overline{\rho} \cdot E_{\Theta}(\sigma(\Theta))^2 \leq \overline{\rho} \cdot E_{\Theta}(\sigma^2(\Theta)) \tag{A.13}$$

donde

$$\overline{\rho} = \frac{E_{\Theta,\Theta'}(\rho(\Theta, \Theta')\sigma(\Theta)\sigma(\Theta'))}{E_{\Theta,\Theta'}(\sigma(\Theta)\sigma(\Theta'))} \tag{A.14}$$

es la correlación promedio entre todos los aprendices tomados a pares.

Ahora, de (A.12)

$$E_{\Theta}(\sigma^2) = E_{\Theta}(E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta')^2) - E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta'))^2) \tag{A.15}$$

De (A.8) sabemos que $m^*(\boldsymbol{X}, Y, \Theta)^2 \leq 1$, por tanto

$$\begin{aligned} E_{\Theta}(E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta'))^2) &\geq (E_{\Theta}(E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta'))))^2 \\ &= (E_{\boldsymbol{X},Y}(E_{\Theta}(m^*(\boldsymbol{X}, Y, \Theta'))))^2 \\ &= (E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta')))2 \\ &= \mu^2 \end{aligned} \tag{A.16}$$

Finalmente,

$$E_{\Theta}(\sigma^2(\Theta)) \geq 1 - \mu^2 \tag{A.17}$$

Usando (A.17) en (A.13), y el resultado en *PE* (A.5), obtenemos la cota superior del error de generalización para un conjunto limitado de aprendices, en términos de

precisión y diversidad, dado por μ y $\overline{\rho}$:

$$PE^* \leq \frac{\overline{\rho}(1-\mu^2)}{\mu^2} \tag{A.18}$$

Referencias

Aceña, V., de Diego, I. M., Fernández, R. R., y Moguerza, J. M. (2022a). Minimally overfitted learners: A general framework for ensemble learning. *Knowledge-Based Systems*, *254*, 109669.

Aceña, V., de Diego, I. M., Fernández, R. R., y Moguerza, J. M. (2022b). Support subsets estimation for support vector machines retraining. *Pattern Recognition*, 109117.

Aceña, V., Moguerza, J. M., Diego, I. M. d., y Fernández, R. R. (2019). Weighted nearest centroid neighbourhood. En *International conference on intelligent data engineering and automated learning* (pp. 94–101).

Arruda, J. L., Prudêncio, R. B., y Lorena, A. C. (2020). Measuring instance hardness using data complexity measures. En *Brazilian conference on intelligent systems* (pp. 483–497).

Artemiou, A., Dong, Y., y Shin, S. J. (2021). Real-time sufficient dimension reduction through principal least squares support vector machines. *Pattern Recognition*, *112*, 107768.

Ayodele, T. O. (2010). Machine learning overview. *New Advances in Machine Learning*, *2*, 9–18.

Barella, V. H., Garcia, L. P., de Souto, M. C., Lorena, A. C., y de Carvalho, A. C. (2021). Assessing the data complexity of imbalanced datasets. *Information Sciences*, *553*, 83–109.

Bauer, E., y Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*(1), 105–139.

Belle, V., y Papantonis, I. (2021). Principles and practice of explainable machine learning. *Frontiers in big Data*, 39.

Bernardo, J., Bayarri, M., Berger, J., Dawid, A., Heckerman, D., Smith, A., y West, M. (2007). Generative or discriminative? getting the best of both worlds. *Bayesian*

statistics, *8*(3), 3–24.

Bifet, A., Gavaldà, R., Holmes, G., y Pfahringer, B. (2018). *Machine learning for data streams: with practical examples in moa*. MIT Press.

Blachnik, M. (2019). Ensembles of instance selection methods: A comparative study. *International Journal of Applied Mathematics and Computer Science*, *29*(1).

Bordes, A., Ertekin, S., Weston, J., Botton, L., y Cristianini, N. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, *6*(9).

Bouaafia, S., Khemiri, R., Sayadi, F. E., y Atri, M. (2020). Fast cu partition-based machine learning approach for reducing hevc complexity. *Journal of Real-Time Image Processing*, *17*(1), 185–196.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Brun, A. L., Britto Jr, A. S., Oliveira, L. S., Enembreck, F., y Sabourin, R. (2018). A framework for dynamic classifier selection oriented by the classification problem difficulty. *Pattern Recognition*, *76*, 175–190.

Carey, A. N., y Wu, X. (2023). The statistical fairness field guide: perspectives from social and formal sciences. *AI and Ethics*, *3*(1), 1–23.

Cauwenberghs, G., y Poggio, T. (2001). Incremental and decremental support vector machine learning. En *Advances in neural information processing systems* (pp. 409–415).

Chang, C.-C. (2008). Libsvm data: Classification, regression, and multi-label. *http://www. csie. ntu. edu. tw/˜ cjlin/libsvmtools/datasets/*.

Chang, C.-C., y Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 1–27.

Chaudhuri, B. (1996). A new definition of neighborhood of a point in multi-dimensional space. *Pattern Recognition Letters*, *17*(1), 11–17.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., y cols. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4).

Chicco, D., y Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*, 1–13.

Chongomweru, H., y Kasem, A. (2021). A novel ensemble method for classification in imbalanced datasets using split balancing technique based on instance hardness (sbal_ih). *Neural Computing and Applications*, *33*(17), 11233–11254.

Cortes, C., y Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Cover, T. M., y Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Cover, T. M., y Thomas, J. A. (2006). *Elements of information theory*. John Wiley & Sons.

Cummins, L., y Bridge, D. (2011). On dataset complexity for case base maintenance. En *International conference on case-based reasoning* (pp. 47–61).

DeCoste, D., y Wagstaff, K. (2000). Alpha seeding for support vector machines. En *Proceedings of the sixth acm sigkdd international conference on knowledge discovery and data mining* (pp. 345–349).

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*(2), 139–157.

Dua, D., y Graff, C. (2017). *UCI machine learning repository*. Descargado de `http://archive.ics.uci.edu/ml`

D'Addabbo, A., y Maglietta, R. (2015). Parallel selective sampling method for imbalanced and large data classification. *Pattern Recognition Letters*, *62*, 61–67.

Fan, R.-E., Chen, P.-H., y Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of machine learning research*, *6*(Dec), 1889–1918.

Fine, S., y Scheinberg, K. (2002). Incremental learning and selective sampling via parametric optimization framework for svm. En *Advances in neural information processing systems* (pp. 705–711).

Fix, E., y Hodges, J. L. (1951). *Discriminatory analysis* (Report). USAF School of Aviation Medicine, Randolph Field, Texas. Descargado de `https://web.archive.org/web/20200926140733/https://www.academia.edu/44450666/Discriminatory_Analysis` (Nonparametric Discrimination: Consistency Properties (PDF))

Freund, Y., y Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System sciences*, *55*(1), 119–139.

Freund, Y., Schapire, R. E., y cols. (1996). Experiments with a new boosting algorithm. En *Machine learning: Proceedings of the thirteenth international conference* (pp.

148–156).

Gama, J. (2012). A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, *1*(1), 45–55.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., y Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1–37.

Garcia, L. P., de Carvalho, A. C., y Lorena, A. C. (2015). Effect of label noise in the complexity of classification problems. *Neurocomputing*, *160*, 108–119.

Geurts, P., Ernst, D., y Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.

Golub, G. H., Heath, M., y Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, *21*(2), 215–223.

González, S., García, S., Del Ser, J., Rokach, L., y Herrera, F. (2020). A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. *Information Fusion*, *64*, 205–237.

Goodall, D. W. (1967). The distribution of the matching coefficient. *Biometrics*, 647–656.

Grandini, M., Bagli, E., y Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Gu, B., Quan, X., Gu, Y., Sheng, V. S., y Zheng, G. (2018). Chunk incremental learning for cost-sensitive hinge loss support vector machine. *Pattern Recognition*, *83*, 196–208.

Han, S., Kim, H., y Lee, Y.-S. (2020). Double random forest. *Machine Learning*, *109*(8), 1569–1586.

Hand, D., y Christen, P. (2018). A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, *28*, 539–547.

Hansen, L. K., y Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Hassan, M. F., Abdel-Qader, I., y Bazuin, B. (2021). A new method for ensemble combination based on adaptive decision making. *Knowledge-Based Systems*, *233*, 107544.

Hastie, T., Tibshirani, R., Friedman, J. H., y Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Ho, T. K., y Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, *24*(3), 289–

300.

Hoi, S. C., Sahoo, D., Lu, J., y Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing*, *459*, 249–289.

Ij, H. (2018). Statistics versus machine learning. *Nat Methods*, *15*(4), 233.

Izenman, A. J. (2008). *Modern multivariate statistical techniques* (Vol. 1). Springer.

Joachims, T. (1998). *Making large-scale svm learning practical* (Inf. Téc.). Technical Report.

Kabir, A., Ruiz, C., y Alvarez, S. A. (2018). Mixed bagging: A novel ensemble learning framework for supervised classification based on instance hardness. En *2018 ieee international conference on data mining (icdm)* (pp. 1073–1078).

Kao, W.-C., Chung, K.-M., Sun, C.-L., y Lin, C.-J. (2004). Decomposition methods for linear support vector machines. *Neural Computation*, *16*(8), 1689–1704.

Kashef, R. (2021). A boosted svm classifier trained by incremental learning and decremental unlearning approach. *Expert Systems with Applications*, *167*, 114154.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*, 3146–3154.

Khurana, D., Koli, A., Khatter, K., y Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, *82*(3), 3713–3744.

Kumar, S., Mohri, M., y Talwalkar, A. (2012). Sampling methods for the nyström method. *The Journal of Machine Learning Research*, *13*(1), 981–1006.

Lam, L., y Suen, S. (1997). Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *27*(5), 553–568.

Lancho, C., Martín de Diego, I., Cuesta, M., Aceña, V., y M Moguerza, J. (2021). A complexity measure for binary classification problems based on lost points. En *International conference on intelligent data engineering and automated learning* (pp. 137–146).

Lancho, C., Martín De Diego, I., Cuesta, M., Aceña, V., y Moguerza, J. M. (2022). Hostility measure for multi-level study of data complexity. *Applied Intelligence*, 1–24.

Laskov, P., Gehl, C., Krüger, S., y Müller, K.-R. (2006). Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*, *7*(Sep), 1909–1936.

Lázaro, M., Herrera, F., y Figueiras-Vidal, A. R. (2020). Ensembles of cost-diverse ba-

yesian neural learners for imbalanced binary classification. *Information Sciences*, *520*, 31–45.

Lee, M. M., Keerthi, S. S., Ong, C. J., y DeCoste, D. (2004). An efficient method for computing leave-one-out error in support vector machines with gaussian kernels. *IEEE Transactions on Neural Networks*, *15*(3), 750–757.

Leyva, E., González, A., y Perez, R. (2014). A set of complexity measures designed for applying meta-learning to instance selection. *IEEE Transactions on Knowledge and Data Engineering*, *27*(2), 354–367.

Leyva, E., González, A., y Pérez, R. (2015). Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, *48*(4), 1523–1537.

Liu, Y., Yu, X., Huang, J. X., y An, A. (2011). Combining integrated sampling with svm ensembles for learning from imbalanced datasets. *Information Processing & Management*, *47*(4), 617–631.

Loosli, G., Canu, S., y Bottou, L. (2007). Training invariant support vector machines using selective sampling. *Large scale kernel machines*, *2*.

Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C., y Ho, T. K. (2019). How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, *52*(5), 1–34.

Lu, J., Hoi, S. C., Wang, J., Zhao, P., y Liu, Z.-Y. (2016). Large scale online kernel learning. *Journal of Machine Learning Research*, *17*(47), 1.

Ma, L., Song, D., Liao, L., y Wang, J. (2017). Psvm: a preference-enhanced svm model using preference data for classification. *Science China Information Sciences*, *60*(12), 122103.

Maillo, J., Triguero, I., y Herrera, F. (2020). Redundancy and complexity metrics for big data classification: Towards smart data. *IEEE Access*, *8*, 87918–87928.

Malik, M. M. (2020). A hierarchy of limitations in machine learning. *arXiv preprint arXiv:2002.05193*.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, *405*(2), 442–451.

Moerland, T. M., Broekens, J., Plaat, A., Jonker, C. M., y cols. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, *16*(1), 1–118.

Molnar, C. (2022). *Modeling mindsets*. Lulu. com.

Okimoto, L. C., y Lorena, A. C. (2019). Data complexity measures in feature selection. En *2019 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Oliveira, D. V., Cavalcanti, G. D., y Sabourin, R. (2017). Online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, *72*, 44–58.

Oliveira, G., Minku, L. L., y Oliveira, A. L. (2021). Tackling virtual and real concept drifts: An adaptive gaussian mixture model approach. *IEEE Transactions on Knowledge and Data Engineering*.

Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., y Moore, J. H. (2017, 11 de Dec). Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, *10*(36), 1–13. Descargado de `https://doi.org/10.1186/s13040-017-0154-4` doi: 10.1186/s13040-017-0154-4

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., y Kittler, J. (2010). A review of instance selection methods. *Artificial Intelligence Review*, *34*, 133–143.

Osuna, E., Freund, R., y Girosi, F. (1997). An improved training algorithm for support vector machines. En *Neural networks for signal processing vii. proceedings of the 1997 ieee signal processing society workshop* (pp. 276–285).

Partridge, D., y Yates, W. B. (1996). Engineering multiversion neural-net systems. *Neural Computation*, *8*(4), 869–893.

Pascual-Triana, J. D., Charte, D., Andrés Arroyo, M., Fernández, A., y Herrera, F. (2021). Revisiting data complexity metrics based on morphology for overlap and imbalance: snapshot, new overlap number of balls metrics and singular problems prospect. *Knowledge and Information Systems*, *63*(7), 1961–1989.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

Perrone, M. P., y Cooper, L. N. (1992). *When networks disagree: Ensemble methods for hybrid neural networks* (Inf. Téc.). Brown Univ Providence Ri Inst for Brain and Neural Systems.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Rao, R. B., Fung, G., y Rosales, R. (2008). On the dangers of cross-validation. an experimental evaluation. En *Proceedings of the 2008 siam international conference on data mining* (pp. 588–596).

Redondo, A. R., Navarro, J., Fernández, R. R., de Diego, I. M., Moguerza, J. M., y Fernández-Muñoz, J. J. (2020). Unified performance measure for binary classification problems. En *International conference on intelligent data engineering and automated learning* (pp. 104–112).

Richhariya, B., y Tanveer, M. (2020). A reduced universum twin support vector machine for class imbalance learning. *Pattern Recognition*, *102*, 107150.

Rodriguez, J. J., Kuncheva, L. I., y Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1619–1630.

Rokach, L. (2019). *Ensemble learning: Pattern classification using ensemble methods* (Vol. 85). World Scientific.

Sáez, J. A., Luengo, J., y Herrera, F. (2013). Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition*, *46*(1), 355–364.

Sarbazi-Azad, S., Abadeh, M. S., y Mowlaei, M. E. (2020). Using data complexity measures and an evolutionary cultural algorithm for gene selection in microarray data. *Soft Computing Letters*, 100007.

Sasaki, Y., y Fellow, R. (2007). The truth of the f-measure, manchester: Mib-school of computer science. *University of Manchester*, 25.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.

Shalev-Shwartz, S., Singer, Y., Srebro, N., y Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, *127*(1), 3–30.

Shinde, P. P., y Shah, S. (2018). A review of machine learning and deep learning applications. En *2018 fourth international conference on computing communication control and automation (iccubea)* (pp. 1–6).

Sleeman IV, W. C., y Krawczyk, B. (2019). Bagging using instance-level difficulty for multi-class imbalanced big data classification on spark. En *2019 ieee international conference on big data (big data)* (pp. 2484–2493).

Smith, M. R., Martinez, T., y Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine Learning*, *95*(2), 225–256.

Song, Q., Wang, G., y Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. *Pattern recognition*, *45*(7), 2672–2689.

Spicer, J., y Sanborn, A. N. (2019). What does the mind learn? a comparison of human and machine learning representations. *Current opinion in neurobiology*, *55*, 97–102.

Stone, M. (1974). Cross-validation and multinomial prediction. *Biometrika*, *61*(3), 509–515.

Stork, D. G., Duda, R. O., Hart, P. E., y Stork, D. (2001). Pattern classification. *Wiley-Interscience*.

Syed, N. A., Huan, S., Kah, L., y Sung, K. (1999). Incremental learning with support vector machines. En *In proc. the workshop on support vector machines at the international joint conference on articial intelligence (ijcai-99)*.

Tharwat, A. (2021). Classification assessment methods. *Applied computing and informatics*, *17*(1), 168–192.

Ting, K. M., Wells, J. R., Tan, S. C., Teng, S. W., y Webb, G. I. (2011). Feature-subspace aggregating: ensembles for stable and unstable learners. *Machine Learning*, *82*(3), 375–397.

Titsias, M. K., y Likas, A. (2002). Mixture of experts classification using a hierarchical mixture model. *Neural Computation*, *14*(9), 2221–2244.

Torres-Barrán, A., Alaíz, C. M., y Dorronsoro, J. R. (2021). Faster svm training via conjugate smo. *Pattern Recognition*, *111*, 107644.

Valentini, G. (2004). Random aggregated and bagged ensembles of svms: an empirical bias–variance analysis. En *International workshop on multiple classifier systems* (pp. 263–272).

Valentini, G., y Masulli, F. (2002). Ensembles of learning machines. En *Italian workshop on neural nets* (pp. 3–20).

Vapnik, V. (1982). *Estimation of dependences based on empirical data: Springer series in statistics (springer series in statistics)*. Springer-Verlag.

Vygotsky, L. S., y Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Harvard university press.

Wahba, G., y cols. (1999). Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, *6*, 69–87.

Walmsley, F. N., Cavalcanti, G. D., Sabourin, R., y Cruz, R. M. (2022). An investigation into the effects of label noise on dynamic selection algorithms. *Information Fusion*, *80*, 104–120.

Wang, Q., Luo, Z., Huang, J., Feng, Y., y Liu, Z. (2017). A novel ensemble method for imbalanced data learning: Bagging of extrapolation-smote svm. *Computational Intelligence and Neuroscience*, *2017*.

Wang, S.-J., Mathew, A., Chen, Y., Xi, L.-f., Ma, L., y Lee, J. (2009). Empirical analysis

of support vector machine ensemble classifiers. *Expert Systems with Applications*, *36*(3), 6466–6476.

Way, M. J., Scargle, J. D., Ali, K. M., y Srivastava, A. N. (2012). *Advances in machine learning and data mining for astronomy*. CRC Press.

Wen, Z., Li, B., Kotagiri, R., Chen, J., Chen, Y., y Zhang, R. (2017). Improving efficiency of svm k-fold cross-validation by alpha seeding. En *Thirty-first aaai conference on artificial intelligence*.

Wilson, D. R., y Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38*(3), 257–286.

Xie, J., Zhu, M., y Hu, K. (2022). Hierarchical ensemble based imbalance classification. En *International conference on computational science* (pp. 192–204).

Xu, L., Krzyzak, A., y Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(3), 418–435.

Yang, J., Yan, R., y Hauptmann, A. G. (2007). Adapting svm classifiers to data with shifted distributions. En *Seventh ieee international conference on data mining workshops (icdmw 2007)* (pp. 69–76).

Ying, X. (2019). An overview of overfitting and its solutions. En *Journal of physics: Conference series* (Vol. 1168, p. 022022).

Yu, X., Chu, Y., Jiang, F., Guo, Y., y Gong, D. (2018). Svms classification based two-side cross domain collaborative filtering by inferring intrinsic user and item features. *Knowledge-Based Systems*, *141*, 80–91.

Yu, X., Jiang, F., Du, J., y Gong, D. (2019). A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains. *Pattern Recognition*, *94*, 96–109.

Yu, X., Peng, Q., Xu, L., Jiang, F., Du, J., y Gong, D. (2021). A selective ensemble learning based two-sided cross-domain collaborative filtering algorithm. *Information Processing & Management*, *58*(6), 102691.

Yule, G. U. (1912). On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society*, *75*(6), 579–652.

Zhang, X., Li, R., Zhang, B., Yang, Y., Guo, J., y Ji, X. (2019). An instance-based learning recommendation algorithm of imbalance handling methods. *Applied Mathematics and Computation*, *351*, 204–218.

Zhang, Y., Cao, G., Wang, B., y Li, X. (2019). A novel ensemble method for k-nearest neighbor. *Pattern Recognition*, *85*, 13–25.

Zhou, T., Wang, S., y Bilmes, J. (2020). Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, *33*, 8602–8613.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC Press.