# Clasificación Jerárquica Multiclase

Daniel Andrés Silva Palacios

Tesis Doctoral


UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Supervisores
María José Ramirez Quintana
Cèsar Ferri Ramirez

España
04/2021

# Resumen

La sociedad moderna se ha visto afectada por los acelerados avances de la tecnología. La aplicación de la inteligencia artificial se puede encontrar en todas partes, desde la televisión inteligente hasta los coches autónomos. Una tarea esencial del aprendizaje automático es la clasificación. A pesar de la cantidad de técnicas y algoritmos de clasificación que existen, es un campo que sigue siendo relevante por todas sus aplicaciones. Así, frente a la clasificación tradicional multiclase en la que a cada instancia se le asigna una única etiqueta de clase, se han propuesto otros métodos como la clasificación jerárquica y la clasificación multietiqueta. Esta tesis tiene como objetivo resolver la clasificación multiclase mediante una descomposición jerárquica. Asimismo, se exploran diferentes métodos de extender la aproximación definida para su aplicación en contextos cambiantes.

La clasificación jerárquica es una tarea de aprendizaje automático en la que el problema de clasificación original se divide en pequeños subproblemas. Esta división se realiza teniendo en cuenta una estructura jerárquica que representa las relaciones entre las clases objetivo. Como resultado el clasificador jerárquico es a su vez una estructura (un árbol o un grafo) compuesta por clasificadores de base. Hasta ahora, en la literatura, la clasificación jerárquica se ha aplicado a dominios jerárquicos, independientemente que la estructura jerárquica sea proporcionada explícitamente o se asume implícita (en cuyo caso se hace necesario inferir primero dicha estructura jerárquica). La clasificación jerárquica ha demostrado un mejor rendimiento en dominios jerárquicos en comparación con la clasificación plana (que no tiene en cuenta la estructura jerárquica del dominio). En esta tesis, proponemos resolver los problemas de clasificación

multiclase descomponiéndolo jerárquicamente de acuerdo a una jerarquía de clases inferida por un clasificador plano. Planteamos dos escenarios dependiendo del tipo de clasificador usado en la jerarquía de clasificadores: clasificadores duros (crisp) y clasificadores suaves (soft).

Por otra parte, un problema de clasificación puede sufrir cambios una vez los modelos han sido entrenados. Un cambio frecuente es la aparición de una nueva clase objetivo. Dado que los clasificadores no han sido entrenados con datos pertenecientes a la nueva clase, no podrán encontrar predicciones correctas para las nuevas instancias, lo que afectará negativamente en el rendimiento de los clasificadores. Este problema se puede resolver mediante dos alternativas: el reentrenamiento de todo el modelo o la adaptación del modelo para responder a esta nueva situación. Como parte del estudio de los algoritmos de clasificación jerárquica se presentan varios métodos para adaptar el modelo a los cambios en las clases objetivo.

Los métodos y aproximaciones definidas en la tesis se han evaluado experimentalmente con una amplia colección de conjuntos de datos que presentan diferentes características, usando diferentes técnicas de aprendizaje para generar los clasificadores de base. En general, los resultados muestran que los métodos propuestos pueden ser una alternativa a métodos tradicionales y otras técnicas presentadas en la literatura para abordar las situaciones específicas planteadas.

# Resum

La societat moderna s'ha vist afectada pels accelerats avenços de la tecnologia. L'aplicació de la intel·ligència artificial es pot trobar a tot arreu, des de la televisió intel·ligent fins als cotxes autònoms. Una tasca essencial de l'aprenentatge automàtic és la classificació. Tot i la quantitat de tècniques i algoritmes de classificació que existeixen, és un camp que segueix sent rellevant per totes les seves aplicacions. Així, enfront de la classificació tradicional multiclase en la qual a cada instància se li assigna una única etiqueta de classe, s'han proposat altres mètodes com la classificació jeràrquica i la classificació multietiqueta. Aquesta tesi té com a objectiu resoldre la classificació multiclase mitjançant una descomposició jeràrquica. Així mateix, s'exploren diferents mètodes d'estendre l'aproximació definida per a la seva aplicació en contextos canviants.

La classificació jeràrquica és una tasca d'aprenentatge automàtic en la qual el problema de classificació original es divideix en petits subproblemes. Aquesta divisió es realitza tenint en compte una estructura jeràrquica que representa les relacions entre les classes objectiu. Com a resultat el classificador jeràrquic és al seu torn una estructura (un arbre o un graf) composta per classificadors de base. Fins ara, en la literatura, la classificació jeràrquica s'ha aplicat a dominis jeràrquics, independentment que l'estructura jeràrquica sigui proporcionada explícitament o s'assumeix implícita (en aquest cas es fa necessari inferir primer aquesta estructura jeràrquica). La classificació jeràrquica ha demostrat un millor rendiment en dominis jeràrquics en comparació amb la classificació plana (que no té en compte l'estructura jeràrquica de l'domini). En aquesta tesi, proposem resoldre els problemes de classificació multiclasse

descomponent jeràrquicament d'acord a una jerarquia de classes inferida per un classificador pla. Plantegem dos escenaris depenent de el tipus de classificador usat en la jerarquia de classificadors: classificadors durs (crisp) i classificadors suaus (soft).

D'altra banda, un problema de classificació pot patir canvis una vegada els models han estat entrenats. Un canvi freqüent és l'aparició d'una nova classe objectiu. Atès que els classificadors no han estat entrenats amb dades pertanyents a la nova classe, no podran trobar prediccions correctes per a les noves instàncies, el que afectarà negativament en el rendiment dels classificadors. Aquest problema es pot resoldre mitjançant dues alternatives: el reentrenament de tot el model o l'adaptació de el model per respondre a aquesta nova situació. Com a part de l'estudi dels algoritmes de classificació jeràrquica es presenten diversos mètodes per adaptar el model als canvis en les classes objectiu.

Els mètodes i aproximacions definides en la tesi s'han avaluat experimentalment amb una àmplia col·lecció de conjunts de dades que presenten diferents característiques, usant diferents tècniques d'aprenentatge per generar els classificadors de base. En general, els resultats mostren que els mètodes proposats poden ser una alternativa a mètodes tradicionals i altres tècniques presentades en la literatura per abordar les situacions específiques plantejades.

# Abstract

The modern society has been affected by rapid advances in technology. The application of artificial intelligence can be found everywhere, from intelligent television to autonomous cars. An essential task of machine learning is classification. Despite the number of classification techniques and algorithms that exist, it is a field that remains relevant for all its applications. Thus, as opposed to the traditional multiclass classification in which each instance is assigned a single class label, other methods such as hierarchical classification and multi-label classification have been proposed. This thesis aims to solve multiclass classification by means of a hierarchical decomposition. Also, different methods of extending the defined approach are explored for application in changing contexts.

Hierarchical classification is an automatic learning task in which the original classification problem is divided into small sub-problems. This division is made taking into account a hierarchical structure that represents the relationships between the target classes. As a result the hierarchical classifier is itself a structure (a tree or a graph) composed of base classifiers. Up to now, in the literature, hierarchical classification has been applied to hierarchical domains, regardless of whether the hierarchical structure is explicitly provided or assumed to be implicit (in which case it becomes necessary to first infer the hierarchical structure). Hierarchical classification has demonstrated better performance in hierarchical domains compared to flat classification (which does not take into account the hierarchical structure of the domain). In this thesis, we propose to solve the problems of multiclass classification by breaking it down hierarchically according to a class hierarchy inferred by a plane

classifier. We propose two scenarios depending on the type of classifier used in the classifier hierarchy: hard classifiers (crisp) and soft classifiers (soft).

On the other hand, a classification problem may change once the models have been trained. A frequent change is the appearance of a new target class. Since the existing classifiers have not been trained with data belonging to the new class, they will not be able to find correct predictions for the new instances, which will negatively affect the performance of the classifiers. This problem can be solved by two alternatives: retraining the entire model or adapting the model to respond to this new situation. As part of the study of hierarchical classification algorithms, several methods are presented to adapt the model to changes in target classes.

The methods and approaches defined in the thesis have been evaluated experimentally with a large collection of data sets that have different characteristics, using different learning techniques to generate the base classifiers. In general, the results show that the proposed methods can be an alternative to traditional methods and other techniques presented in the literature to address the specific situations raised.

# Dedicatoria

*A mi esposa, por ser mi apoyo en cada paso del camino, me diste la energía y la motivación para seguir adelante.*

*A mi madre María, por todos los consejos que me diste.*

*A mi hijo Mathias.*

# Agradecimientos

Agradezco a la Universidad Politécnica de Valencia por darme la oportunidad de ser parte del programa de doctorado. Y principalmente a mis tutores por su guia, esfuerzo y dedicación.

Gracias a todos los profesores de los cursos transversales.

Agradezco especialmente a todos los compañeros y amigos que estuvieron allí dispuestos a darme consejos y escucharon mis ideas.

A mi familia por todo su apoyo.

Gracias a Dios por darme la fortaleza para continuar y el ánimo disfrutar el proceso.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1 Motivación

Los algoritmos de aprendizaje automático son responsables de la gran mayoría de los avances recientes en inteligencia artificial. Un claro ejemplo son los sistemas de recomendación aplicados a plataformas populares como YouTube, Spotify y Netflix, que utilizan el aprendizaje automático para analizar los datos de millones de usuarios con el fin de ofrecerles productos personalizados. Los motores de búsqueda como Google, medios sociales como Facebook y Twitter, y asistentes de voz como Siri y Alexa, también incorporan mecanismos de aprendizaje, que les permiten responder según las características e interacciones que tengan con el usuario.

Los algoritmos de aprendizaje automático están presentes en muchas de las actividades que realizamos en nuestra vida cotidiana. Cuando viajamos en coche, posiblemente estemos usando un sistema automático que controla la navegación y, según la situación del tráfico, nos ofrece alternativas para llegar a nuestro destino. Cuando navegamos por internet o cuando usamos redes sociales, es probable que los contenidos que aparecen en nuestra pantalla sean exclusivamente preparados para un tipo de usuario que tenga gustos y preferencias similares a las nuestras. Estos sistemas aprenden y buscan patrones para así identificarnos como posibles compradores de sus productos y servicios.

Existen dos tipos básicos de algoritmos de aprendizaje automático: supervisados y no supervisados. Los primeros aprenden a nombrar nuevas instancias aprendiendo de ejemplos etiquetados previamente, por lo que se conocen tanto las entradas como la salidas, mientras que los segundos buscan patrones para ordenar objetos similares que no han sido nombrados previamente, de forma que se pueda describir su estructura.

Esta tesis pretende realizar una investigación centrada en un aprendizaje supervisado para mejorar la calidad de la clasificación, la cual tiene como objetivo aprender modelos capaces de convertir un conjunto de ejemplos de entrenamiento previamente clasificados en conocimiento. Ejemplos de una tarea de clasificación son: identificar una acción en un vídeo, reconocer un objeto en una fotografía o clasificar un correo electrónico como no deseado.

Las tareas podrían ser sencillas o complejas y desafiantes, igual que las presentes al realizar aplicaciones prácticas a situaciones reales. Por ejemplo: la detección de cáncer por imágenes (visión por ordenador), identificar las regiones de codificación genética (biología), anticipar los movimientos en el mercado de valores (finanzas) y la predicción, diagnóstico o cura de nuevas enfermedades (atención sanitaria).

Existe una amplia variedad de técnicas adecuadas para construir modelos que realicen la clasificación. Normalmente cada uno de ellos trabaja en función de los objetivos y requerimientos de una tarea de clasificación específica. Algunos algoritmos del aprendizaje automático generan clasificadores que pueden explicar sus predicciones, mientras que otros son cajas negras cerradas imposibles de descifrar. A veces es difícil seleccionar una técnica de clasificación para resolver un problema concreto por la gran variedad de técnicas existentes.

En algunas tareas de clasificación, el ejemplo puede pertenecer a varias clases a la vez, lo cual se conoce como clasificación multietiqueta. La clasificación multietiqueta representa una corriente de investigación con importantes aplicaciones prácticas; sin embargo, en este estudio centraremos el análisis en problemas de clasificación asumiendo que cada ejemplo se etiqueta con una y solo una clase.

El hecho de asignar una clase a una instancia supone que se ha aprendido a distinguir entre un conjunto de dos o más clases. Generalmente al resolver un problema de clasificación, se utilizan técnicas que no tienen en cuenta las posibles relaciones que existen entre las clases. Los autores Silla y Freitas [SF11] recopilan un conjunto de técnicas de clasificación que utilizan las relaciones predefinidas entre las clases de un problema de clasificación cuando

estas forman una estructura jerárquica. Los algoritmos de aprendizaje automático utilizados para la clasificación que usan una jerarquía predefinida son conocidos como clasificadores jerárquicos.

Existen múltiples estudios que mencionan las mejoras en el rendimiento obtenido al utilizar las jerarquías disponibles en problemas con un claro dominio jerárquico (existe una relación jerárquica o taxonomía que relaciona las posibles clases del dominio). Sin embargo, con frecuencia se pueden encontrar conjuntos de datos que no tienen un dominio jerárquico y, por lo tanto, no tienen una jerarquía definida. Basándonos en eso, nos preguntamos si es posible usar técnicas de inferencia jerárquica en estos casos.

En este estudio, en lugar de analizar problemas de clasificación donde las jerarquías de clases han sido previamente definidas, nos centraremos en el análisis de los mecanismos que nos permiten inferir automáticamente estas relaciones y aplicarlas a cualquier problema de clasificación bajo dominios no jerárquicos. Para esto revisaremos formas de inferir una jerarquía de clases a partir de los datos; en concreto, utilizaremos un valor de distancia calculado a partir de la matriz de confusión obtenida como resultado de resolver el problema de clasificación con las técnicas tradicionales. Con esta información realizaremos un proceso de *clustering* con el objetivo de extraer una jerarquía formada con las clases y analizaremos algunos algoritmos para podar la jerarquía obtenida y reducir así la complejidad del proceso. La inferencia de jerarquías nos permite crear un árbol de clasificadores que es aplicable al problema, de la misma forma que lo hacen los clasificadores jerárquicos. Además, presentaremos una alternativa a los métodos propuestos utilizando técnicas de clasificación capaces de realizar estimaciones de probabilidad.

Es frecuente encontrar situaciones en las que las características de los datos cambian con el tiempo. En este escenario un modelo se vuelve obsoleto y paulatinamente reduce su rendimiento. Por ejemplo, el sistema de recomendación que predice artículos de interés para los usuarios de un supermercado debe aprender de forma incremental para adaptarse a las nuevas necesidades e intereses del cliente. Una forma de resolver este problema es el reentrenamiento del modelo para su posterior despliegue. Este proceso puede volverse muy complejo, podría tardar mucho tiempo o no preservar adecuadamente conocimiento relevante previamente aprendido. Otra posible solución es adaptar el modelo según los cambios dados en el contexto. Existen varias estrategias de adaptación incremental que tienen como objetivo modificar el modelo para que responda a cambios que se han producido en los datos.

En este trabajo de investigación hemos analizado posibles estrategias para adaptar el modelo de clasificación jerárquico en contextos cambiantes. En este punto, nos centramos en escenarios donde las clases se aprenden de forma incremental y proponemos formas de adaptar clasificadores jerárquicos utilizando la estructura jerárquica inferida.

Encontramos que la descomposición de un problema de clasificación a través del uso de una jerarquía inferida de los datos mejora la calidad del modelo, reduce su complejidad y permite la modificación del modelo incrementalmente para adaptarlo en escenarios donde los contextos son dinámicos.

En los siguientes capítulos de esta tesis realizaremos un análisis profundo de los métodos y técnicas de clasificación jerárquica multiclase. Además, se presentarán propuestas que responden a los desafíos actuales y generan preguntas relevantes para futuras investigaciones.

## 1.2 Objetivos

Los objetivos generales de esta tesis son:

- G1. El primer objetivo general de esta tesis es inferir y explotar las relaciones entre las etiquetas de clases para mejorar la calidad de la clasificación en un problema de dominio no jerárquico y en el que no se tiene una jerarquía definida.
- G2. El segundo objetivo general es evaluar el uso de estimaciones de probabilidad para la creación y aplicación de clasificadores jerárquicos multiclase con el fin de obtener jerarquías que permitan mejorar el proceso de clasificación y, por lo tanto, aumentar el rendimiento de los modelos en problemas no jerárquicos.
- G3. El tercer objetivo es evaluar el uso de jerarquías de clases en escenarios que presentan contextos cambiantes, en particular, adaptar los clasificadores jerárquicos multiclase en situaciones en las que aparecen nuevas etiquetas de clase no vistas durante el entrenamiento del modelo.

De los estos objetivos generales se desprenden los siguientes objetivos específicos:

- G1.1. Analizar los fundamentos, conceptos básicos y técnicas del aprendizaje automático enfocadas en un aprendizaje supervisado donde se utilicen las relaciones entre clases para mejorar técnicas de clasificación.

- G1.2. Evaluar las estrategias que tienen en cuenta las relaciones entre clases para la inferencia de jerarquias partiendo de datos.
- G2.1. Evaluar metodologías existentes que permitan utilizar las jerarquías de clases en problemas de clasificación.
- G2.2. Proponer un método que permita utilizar las jerarquías de clases en problemas de clasificación donde las técnicas de clasificación tienen como salida estimaciones de probabilidad.
- G3.1. Analizar las técnicas de adaptación existentes utilizadas para realizar una clasificación en un contexto en que las clases son aprendidas incrementalmente.
- G3.2. Proponer y evaluar estrategias de adaptación que permitan el aprendizaje incremental utilizando modelos de clasificación jerárquicos multiclase.

## 1.3 Contribuciones Principales

En esta sección resumimos las contribuciones de este trabajo. El capítulo 4 describe nuestra primera contribución. Proponemos un método para inferir jerarquías de clase a partir de los datos y un algoritmo de poda capaz de reducir su tamaño. Ejecutamos experimentos para evaluar el rendimiento de las técnicas de clasificación jerárquica multiclase. Los principales hallazgos se compilan en un artículo aceptado en la *Conferencia Internacional de Ciencias Computacionales* (ICCS 2017). El artículo se titula "Improving Performance of Multiclass Classification by Inducing Class Hierarchies" [SFR17].

En el capítulo 5, describimos la segunda contribución. Consiste en la propuesta de un método de clasificación jerárquica que utiliza estimaciones de probabilidad. Evaluamos experimentalmente y analizamos los resultados. Los resultados los publicamos en un artículo titulado "Probabilistic class hierarchies for multiclass classification" en el *Journal of Computational Science* [SFR18b].

Finalmente, en el capítulo 6 presentamos métodos de adaptación de modelos jerárquicos multiclase que responden a la aparición de una nueva clase objetivo. Estos métodos fueron evaluados experimentalmente y analizados en varios escenarios. Estas contribuciones han sido presentadas en el artículo titulado "Adapting Hierarchical Multiclass Classification to Changes in the Target Concept", el cual fue presentado en el *Congreso de la Asociación Española de*

*Inteligencia Artificial* [SFR18a]. En los siguientes capítulos de este documento, revisaremos en profundidad cada contribución.

El autor y los tutores participaron directamente en la concepción, planificación e investigación de los diferentes temas presentados en esta tesis. Además, contribuyeron en la planificación de los experimentos, el análisis de los resultados y la preparación de comunicaciones.

## 1.4 Esquema

Esta tesis se divide en siete capítulos. En este capítulo, hemos presentado las motivaciones de la investigación, los objetivos principales, las preguntas de la investigación y las contribuciones.

El capítulo 2 introduce conceptos básicos del aprendizaje automático, incluyendo las modalidades de aprendizaje y la descripción de las tareas de agrupamiento y clasificación. Luego, se describe cómo realizar una correcta evaluación de los clasificadores y se introducen los principales tests estadísticos para determinar la relevancia de los resultados obtenidos en la evaluación. Finalmente, revisamos las principales técnicas utilizadas en la clasificación.

En el capítulo 3 se presentan algunas de las estrategias utilizadas para mejorar la clasificación multiclase, tales como los *ensembles*, la descomposición de problemas de clasificación y la clasificación jerárquica.

En el capítulo 4 presentamos un método para la inferencia de jerarquías de clase basado en la similitud de los datos de entrenamiento. Planteamos un proceso de poda de nodos de la jerarquía que permiten reducir su tamaño. Ejecutamos múltiples experimentos con varios conjuntos de datos con la finalidad de evaluar el rendimiento del método. La evaluación y el análisis de los resultados se presentan al final del capítulo.

En el capítulo 5 se proponen métodos clasificación jerárquica multiclase que usan estimaciones de probabilidad. Evaluamos su rendimiento a través de una serie de experimentos utilizando múltiples conjuntos de datos. En la última sección de este capítulo se presentan los resultados, el análisis y las conclusiones.

El capítulo 6 presenta el problema del aprendizaje incremental de clases y los métodos disponibles. Hemos desarrollado varios métodos para la adaptación modelos jerárquicos multiclase para responder a la aparición de nuevas etiquetas de clase. Evaluamos experimentalmente las propuestas utilizando

un conjunto de bases de datos. Se incluyen los resultados obtenidos con los métodos propuestos y se discuten las ventajas y desventajas de cada método.

Finalmente, el capítulo 7 presenta las principales conclusiones a las que hemos llegado.

# Capítulo 2

# Conceptos básicos sobre aprendizaje automático

La inteligencia artificial es la responsable de muchos de los avances tecnológicos recientes. Estos han pasado desde que el ser humano vio por primera vez a un ordenador como un mecanismo capaz de procesar grandes cantidades de información y realizar cálculos complejos con relativa facilidad. Desarrollar algoritmos y métodos para obtener la llamada «inteligencia artificial» fue algo imaginado mucho antes de que existieran los ordenadores que usamos hoy en día. Ya en 1950 Alan Turing definía el llamado test de Turing, en el cual se evaluaba la capacidad de una máquina para imitar un comportamiento inteligente parecido al del ser humano.

Desde entonces, se han usado algoritmos para crear modelos capaces de realizar tareas complejas y se ha establecido lo que hoy es conocido como «aprendizaje automático». En el artículo "Introducción al aprendizaje automático" [Cap17], el autor define *aprendizaje* como un proceso a través del cual se adquieren o modifican habilidades, destrezas, conocimientos, conductas o valores como resultado de la experiencia, la instrucción, el razonamiento y la observación. Un factor importante para crear modelos robustos es la capacidad de proveer al algoritmo la información adecuada y la experiencia necesaria para propiciar el proceso de aprendizaje. Algunas aplicaciones prácticas usaron téc-

nicas de aprendizaje automático para solucionar problemas en empresas donde la toma de decisiones se basaba en el análisis de datos históricos para realizar predicciones. El término empleado para denominar el campo que se ocupa del descubrimiento de patrones relevantes para la toma de decisiones empresariales fue «minería de datos».

Los años 80 estuvieron marcados por el nacimiento de sistemas expertos basados en reglas. Estos fueron rápidamente adoptados por el sector corporativo, lo que generó un nuevo interés en aprendizaje automático. El aumento de la potencia de cálculo, junto con la gran abundancia de datos disponibles, han vuelto a lanzar el campo del aprendizaje automático en los últimos años. Numerosas empresas están transformando sus modelos de negocio con el fin de explotar sus datos históricos y están incorporando técnicas de aprendizaje automático en sus procesos, productos y servicios para obtener ventajas sobre la competencia, lo que ha dado espacio para un desarrollo sin igual del aprendizaje automático, en el que convergen ciencias como las matemáticas, la estadística y las ciencias de la computación. La popularización de la minería de datos y sus capacidades de transformar la forma de hacer negocios ha motivado la aparición de nuevos métodos de aprendizaje.

En este capítulo revisaremos las modalidades de aprendizaje y los problemas de clasificación en los que se intenta predecir cuál es la etiqueta de clase (de entre un conjunto de clases prefijadas) a asignar a un objeto. Presentaremos un ejemplo hipotético donde definiremos formalmente qué es una tarea de clasificación, las técnicas de validación y la evaluación de modelos de clasificación. Asimismo, introduciremos varios de los principales algoritmos utilizados en clasificación y analizaremos varias técnicas para comparar el rendimiento entre modelos.

## 2.1 Modalidades de aprendizaje

El aprendizaje automático contempla dos modalidades: supervisado y no supervisado. En el primero, el aprendizaje se realiza, como su nombre indica, bajo supervisión, lo que significa que los datos usados para el aprendizaje están etiquetados (conocemos el valor real de su etiqueta de clase). Por ejemplo, las etiquetas de una imagen fotográfica pueden ser los objetos que contiene, tales como personas, animales, plantas y objetos inanimados; en el contexto de una noticia escrita, la etiqueta puede pertenecer a un tema político, deportes o actualidad. Por lo general, las etiquetas son definidas por los humanos y son más difíciles de conseguir que los datos sin etiquetar.

Hay dos tipos principales de problemas en el aprendizaje supervisado. La mayor diferencia entre ellos se basa en el tipo de objetos que intentan predecir. En el caso de que la predicción sea una clase categórica, el problema se conoce como *clasificación*, mientras que, si predecimos un valor numérico, hablamos de *regresión*.

En el caso de un aprendizaje no supervisado, se ingresa al algoritmo un conjunto de datos de entrada que no han sido etiquetados o clasificados. Se busca encontrar relaciones entre los datos de entrada basándose en los patrones, las similitudes o, incluso, las diferencias que pueden separar y ordenar los datos. Dado que el modelo recibe datos sin etiquetar, no hay ningún error o señal de supervisión para evaluar la posible solución. El método más común de aprendizaje no supervisado es el agrupamiento o *clustering*.

Existen otros enfoques en cuanto a las formas de aprendizaje, como el aprendizaje semi-supervisado [Sug16], en el que se usan datos etiquetados y no etiquetados para el aprendizaje. Otro tipo de enfoque es conocido como aprendizaje por refuerzo, introducido en [SB18] y en el cual se intenta maximizar la recompensa de los resultados. Por ejemplo, si alguien quiere aprender a conducir un coche, se le recompensará cada vez que realice un adelantamiento correcto.

En la siguiente sección, analizaremos en profundidad las tareas de *clustering* y de clasificación, dos puntos clave en la investigación realizada.

## 2.2 Clustering

El *clustering* o análisis de agrupación es un tipo de aprendizaje no supervisado. La entrada del proceso es un conjunto de datos no etiquetados y la salida dos o más *clusters* o grupos. Cada ejemplo usado en el aprendizaje está descrito por un vector de atributos. Por ejemplo, si una entidad financiera requiere agrupar un conjunto de clientes para ofrecer opciones de crédito, podríamos considerar los atributos salario y edad. El conjunto de ejemplos corresponde a cada uno de los clientes registrados en su base de datos.

La figura 2.1 muestra un dominio simple con ejemplos descritos con dos atributos: salario y edad. Se puede observar que la mayoría de los ejemplos podrían reunirse en dos o tres grupos. La identificación visual es sencilla para dos dimensiones, pero, cuando se trabaja con más de tres, no se puede visualizar ni ver los posibles *clusters*. En este caso, se pueden aplicar algoritmos de *clustering* para realizar el análisis.

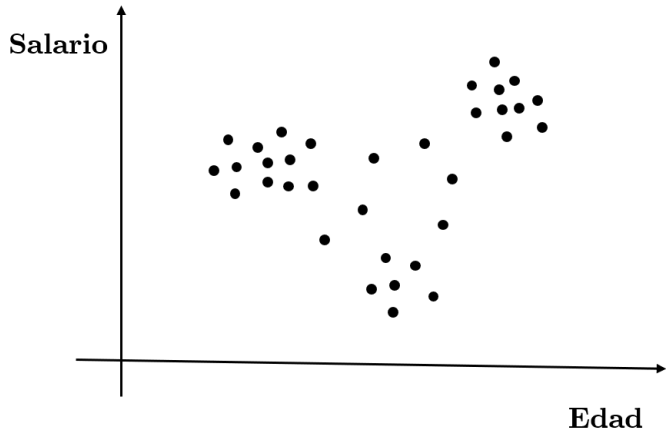



**Figura 2.1:** Un dominio bidimensional con grupos de ejemplos.

Consideremos que $\mathcal{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ es nuestro conjunto de datos donde $n$ es el número de puntos de datos. Se define un *cluster-k* de $\mathcal{X}$ como la partición de $\mathcal{X}$ en $k$ conjuntos (*clusters*), donde cada *cluster* $Cl_1, \ldots, Cl_k$ obedece a las siguientes condiciones:

- $Cl_i \neq \emptyset, i = 1, \ldots, k$
- $\cup_{i=1}^{k} Cl_i = \mathcal{X}$
- $Cl_i \cap Cl_j = \emptyset, i \neq j, i, j = 1, \ldots, k$

Existen dos tipos principales de aproximaciones para realizar agrupaciones: particionales o jerárquicos. A continuación, se revisarán brevemente ambos tipos de técnicas de agrupación.

*Clustering particional*

Uno de los algoritmos particionales más conocidos es *k-means*, en el cual $k$ representa el número de *clusters* a formar, siendo este un parámetro de entrada del algoritmo. El algoritmo funciona de la siguiente forma. Inicialmente, se seleccionan $k$ puntos aleatorios del conjunto de datos los cuales constituyen los centroides de los *clusters*. Luego, se asigna a cada *cluster* los puntos que se encuentran más cercanos a su centroide. Finalmente, se actualiza la ubicación de los centroides usando la media de los datos de cada *cluster*. Este proceso se

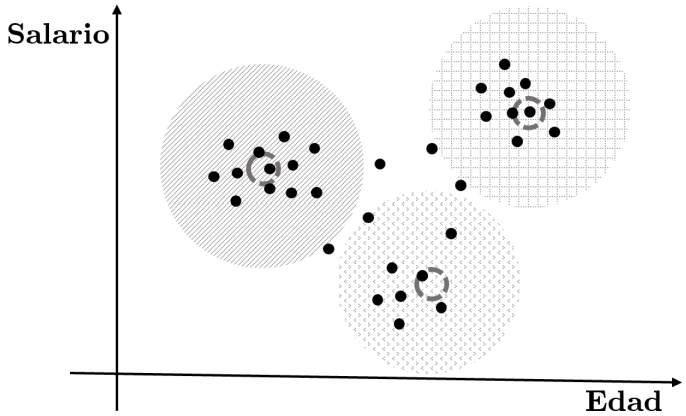



**Figura 2.2:** Ejemplo de k-means Clustering.

repite hasta que los centroides dejen de moverse. La figura 2.2 muestra los tres centroides del ejemplo inicial. La principal debilidad de este método es que no es determinista (sólo es determinista respecto a los parámetros de entrada); además, los puntos que se encuentran equidistantes entre dos centroides podrían incluirse a cualquiera de ambos *clusters*, pero la ventaja es que siempre converge a una solución óptima local [BF98].

Cuando se realiza un proceso de *clustering* es muy importante definir cómo computar distancias entre instancias. Si consideramos dos puntos $p_x$ y $p_y$, descritos mediante los vectores de $l$ componentes $\vec{x}$ e $\vec{y}$, respectivamente, existen varias formas de calcular su distancia. Por ejemplo:

- Distancia de Manhattan
$$d_{manhattan}(p_x, p_y) = \sum_{i=1}^{l} \mid x_i - y_i \mid$$

- Distancia de Euclídea
$$d_{euclid}(p_x, p_y) = \sqrt{\sum_{i=1}^{l} (x_i - y_i)^2}$$

- Distancia de Minkowski
$$d_{minko}(p_x, p_y) = \left( \sum_{i=1}^{l} \mid x_i - y_i \mid^p \right)^{1/p}$$

Obsérvese que, para $p = 1$, la expresión anterior coincide con la distancia de Manhattan y, para $p = 2$, coincide con la distancia Euclídea.

*Clustering Jerárquico*

Un algoritmo de agrupación jerárquica, en lugar de producir un conjunto de *clusters* independientes, produce una jerarquía de agrupaciones anidadas, las cuales están organizadas como un árbol [And14]. Cada nodo (*cluster*) en el árbol (excepto las hojas) es la unión de sus hijos (*subclusters*), y la raíz del árbol es el *cluster* que contiene todos los objetos. Un *cluster* jerárquico puede verse como una secuencia de *clusters* particionales y un *cluster* particional puede obtenerse tomando cualquier miembro de esa secuencia, por ejemplo, al cortar el árbol jerárquico en un nivel particular.

Hay dos tipos de aproximaciones en los algoritmos de agrupación jerárquico: el aglomerativo y el divisivo. El aglomerativo comienza con $n$ *clusters*, cada uno de los cuales contiene un único elemento (*singleton*) y, en cada paso, se unen los pares de *clusters* más cercanos.

Por otro lado, en los algoritmos divisorios, la agrupación inicial consiste en un solo conjunto y, en cada paso, se divide el *cluster* hasta que se tiene un *singleton* de los puntos remanentes individuales. En el primer caso es necesario definir la noción de proximidad entre *clusters* y en el segundo se debe decidir qué *cluster* dividir en cada paso y cómo realizarlo.

La visualización gráfica de una agrupación jerárquica se denomina «dendrograma». La figura 2.3 muestra un ejemplo de dendrograma usando el conjunto de datos de Mtcars [HP81]. Una ventaja del uso de dendrogramas es que nos permite ver los datos agrupados a varios niveles de granularidad.

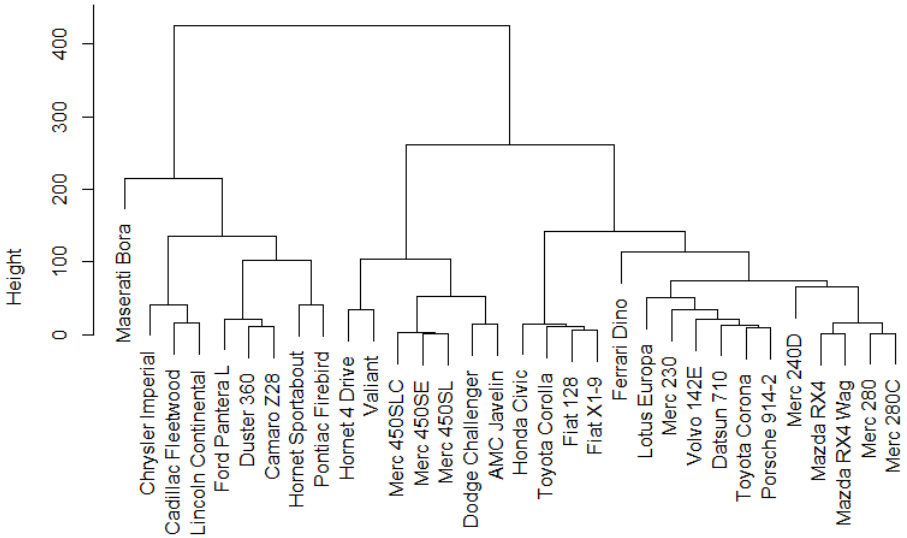



**Figura 2.3:** Ejemplo de dendrograma del conjunto de datos de Mtcars [HP81]. Este fue tomado de la revista estadounidense *Motor Trend* de 1974 y comprende el consumo de combustible y 10 aspectos del diseño y rendimiento de 32 automóviles (modelos 1973-74).

La fusión de dos *clusters* $Cl_1$ y $Cl_2$ depende de la estrategia de enlazado que se aplique. Algunas de las técnicas de enlazado más populares son:

- *Single Link.* También denominado «vecino más cercano», define la distancia entre dos *clusters* como la distancia entre los elementos más cercanos de cada *cluster*.

$$D(Cl_1, Cl_2) = \min_{x_1 \in Cl_1, x_2 \in Cl_2} d(x_1, x_2) \tag{2.1}$$

- *Complete Link.* Define la distancia entre dos *clusters* como la distancia entre los dos elementos más lejanos de cada *cluster*.

$$D(Cl_1, Cl_2) = \max_{x_1 \in Cl_1, x_2 \in Cl_2} d(x_1, x_2) \tag{2.2}$$

- *Simple Average.* La distancia entre dos grupos es el promedio de los valores de distancia de cada elemento del grupo 1 con cada elemento

del grupo 2. De esta manera, ambos grupos tienen influencia sobre el resultado final y se ven menos afectados por los valores atípicos.

$$D(Cl_1, Cl_2) = \frac{1}{|Cl_1|}\frac{1}{|Cl_2|}\sum_{x_1 \in Cl_1}\sum_{x_2 \in Cl_2} d(x_1, x_2) \tag{2.3}$$

- *Centroid-Linkage.* La distancia entre dos grupos es la distancia entre sus centroides (medios).

$$D(Cl_1, Cl_2) = d\left(\left(\frac{1}{|Cl_1|}\sum_{x \in Cl_1}\vec{x}\right), \left(\frac{1}{|Cl_2|}\sum_{x \in Cl_2}\vec{x}\right)\right) \tag{2.4}$$

## 2.3 Clasificación

Comencemos esta sección introduciendo algunos conceptos básicos y terminologías relacionadas con la clasificación a través de un ejemplo. Imaginemos un problema de aprendizaje en el que somos turistas y hemos llegado a una pequeña isla de vacaciones. Pronto descubrimos que un importante ingrediente de la dieta local es el mango. Sin embargo, nunca lo hemos degustado y, por lo tanto, debemos aprender a predecir si el mango del mercado es dulce o no con solo mirarlo.

Lo primero es decidir en qué características del mango basar la predicción. En base a nuestra experiencia previa, decidimos usar dos características: el color, que va desde el verde oscuro o naranja hasta el rojo, y la suavidad, que va desde duro como una roca hasta pulposo. Con lo único que contamos para encontrar las reglas de predicción es una única experiencia previa, en la cual, después de examinar el color y la suavidad, descubrimos si era dulce o no. Analicemos esta tarea como una demostración de las consideraciones involucradas en los problemas de aprendizaje [SB14].

El modelo formal de aprendizaje se puede descomponer en un conjunto de entradas y salidas. Un algoritmo general de aprendizaje tiene como entradas los siguientes elementos:

- **Conjunto dominio**: un conjunto arbitrario, $\mathcal{X}$, que representa el conjunto de objetos que queremos etiquetar. Por ejemplo, en el problema de aprendizaje mencionado anteriormente, el conjunto dominio será el conjunto de todos los mangos. Usualmente, estos puntos de dominio serán

representados por un vector de valores o atributos. Nos referiremos a estos puntos de dominio como instancias.

- **Conjunto de etiquetas**: en nuestro ejemplo, restringimos las etiquetas a un conjunto de dos características que debemos predecir $\{dulce, no-dulce\}$. Denotamos $\mathcal{Y}$ como el conjunto de posibles etiquetas.
- **Datos de entrenamiento**: es un conjunto $\mathcal{S} = \{(x_1, y_1) \dots (x_n, y_n)\}$, es decir, una secuencia finita de pares en $\mathcal{X} \times \mathcal{Y}$ (secuencia de puntos del dominio etiquetados) para ser usado como entrada al algoritmo de aprendizaje. En el ejemplo es el conjunto de mangos que previamente han sido probados, su color, suavidad y dulzor. Tales ejemplos etiquetados son llamados ejemplos de entrenamiento o instancias. Cada una de las características que componen una instancia se conoce como atributo.

Un problema de clasificación es un aprendizaje inductivo que consiste en aprender un clasificador a partir de un conjunto de entrenamiento. Un clasificador, también conocido como predictor o modelo de clasificación, es una función $h : \mathcal{X} \to \mathcal{Y}$, usada para predecir la etiqueta de una nueva instancia. En el contexto del ejemplo, es la función que precede a si un mango seleccionado aleatoriamente del mercado es dulce o no.

La clasificación puede ser de varios tipos, dependiendo del número de clases que conforman el problema de clasificación. Se denomina «binario» cuando es capaz de predecir una de dos posibles clases. Por ejemplo, solamente se puede clasificar un mango como dulce o no. Otro ejemplo típico de clasificador binario es clasificar el correo electrónico como deseado y no deseado. Por otro lado, se denomina «multiclase» al problema de clasificar una instancia en una entre más de dos posibles clases.

Se dice que el clasificador es multietiqueta si es posible que, dentro del problema de clasificación, se puedan asignar dos o más etiquetas de clase a una instancia. Una posible aplicación, por ejemplo, es la categorización de documentos por tema. Un documento, en el caso de un libro, podría pertenecer a varias categorías (por ejemplo: acción y aventura, historia, romance y ciencias humanas). Las posibles salidas del algoritmo de aprendizaje pueden ser una o más de las etiquetas $\mathcal{Y} = \{acción, aventura, historia, romance, ciencias\}$. Al contrario, si el predictor solamente asigna una etiqueta para cada instancia, es conocido como un «clasificador de etiqueta única». En el ejemplo de la predicción del mango, se puede decir que es un clasificador binario y de etiqueta única. Por simplicidad, y porque es el tema central de estudio de esta tesis, denominaremos «clasificador» a un modelo que estima una clase de entre un

conjunto de dos o más clases y que puede asignar solamente una etiqueta por instancia al realizar la predicción.

## 2.4 Evaluación de algoritmos de clasificación

La calidad de un modelo de clasificación depende de la capacidad que tiene el clasificador para generalizar el conocimiento contenido en los datos. Para evaluar un modelo de clasificación se pueden utilizar varios esquemas y medidas de rendimiento, lo que permite seleccionar el mejor clasificador ante un problema de clasificación específico. Después de realizar la evaluación, los resultados de las métricas obtenidas de varios modelos de clasificación son comparados utilizando una o varias pruebas estadísticas, las cuales indicarán si en realidad existen diferencias significativas entre los modelos evaluados.

A continuación, presentamos las métricas usadas para la evaluación del modelo de clasificación, los esquemas de validación más utilizados y las pruebas estadísticas que permiten medir la significatividad estadística de los resultados obtenidos.

### *2.4.1 Medidas de rendimiento del clasificador*

Una vez aplicada una estrategia de validación del modelo, es necesario evaluar la calidad del clasificador. Para ello debemos definir una serie de métricas que nos permitirán cuantificar el rendimiento del clasificador. En [FHM09] se analizan varias métricas usadas para calcular el rendimiento de los clasificadores en escenarios específicos. Una herramienta muy utilizada para mostrar los resultados de una clasificación multiclase es a través de una matriz de confusión, en la que los valores predichos se pueden comparar con los valores reales.

*Matriz de confusión*

Consideramos que tenemos un conjunto finito $\mathcal{D} = \{x_1, x_2, x_3 \ldots, x_n\}$ de $n$ instancias, con $\mathcal{D} \subset \mathcal{X}$. Sabemos que la clase real de cada instancia $x \in \mathcal{D}$ es $r(x)$ y la clase predicha $c(x)$ corresponde a la predicción del clasificador $c$ sobre la instancia $x$. Los valores de $c(x)$ y de $r(x)$ son tomados del conjunto de etiquetas $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$. La matriz de confusión $M$ es una matriz cuadrada, en la cual las dimensiones son el número de clases $m$. $M_{ij}$ denota cuántas veces una instancia de la clase $y_i$ fue clasificada como de clase $y_j$.

Usando el ejemplo presentado al inicio de este capítulo, la tabla 2.1 muestra los resultados de la clasificación (dulce/no-dulce) de seis mangos. En la tabla, P hace referencia a que el mango es dulce (predicción positiva), mientras que N es que no lo es (predicción negativa).

| **x** | **r(x)** | **c(x)** |
|---|---|---|
| $x_1$ | P | N |
| $x_2$ | N | N |
| $x_3$ | P | P |
| $x_4$ | N | P |
| $x_5$ | P | P |
| $x_6$ | N | N |

**Tabla 2.1:** Ejemplo de resultados obtenidos por un clasificador binario. La columna r(x) representa los valores reales, mientras que c(x) representa los valores predichos.

En la tabla 2.2, se muestra la matriz de confusión correspondiente a los resultados presentados en la tabla 2.1. Esta matriz de confusión muestra los resultados de un problema de clasificación en el que se clasifican dos clases. En este caso, las instancias positivas correctamente clasificadas se denominan «verdaderos positivos» (*True Positive*, TP) y las instancias negativas clasificadas correctamente se denominan «verdaderos negativos» (*True Negative*, TN), mientras que las instancias positivas clasificadas incorrectamente se denominan «falsos negativos» (*False Negative*, FN) y las instancias negativas clasificadas erróneamente se denominan «falsos positivos» (*False Positive*, FP).

| | | Predicho | |
|---|---|---|---|
| | | *P* | *N* |
| Actual | *P* | 2 (TP) | 1 (FN) |
| | *N* | 1 (FP) | 2 (TN) |

**Tabla 2.2:** Ejemplo de matriz de confusión de un clasificador binario.

Tomando en cuenta la matriz de confusión, se pueden definir varias métricas:

**Tasa de aciertos o *accuracy***. Es la métrica más conocida, se define como la ratio de las instancias clasificadas correctamente y nos da la idea general del rendimiento del clasificador. Para un clasificador binario, el cálculo del *accuracy* es:

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.5}$$

En el caso de un clasificador multiclase, el valor de *accuracy* es:

$$accMultiClass = \frac{\sum_{i=1}^{m} M_{i,i}}{\sum_{i=1}^{m} \sum_{j=1}^{m} M_{ij}} \tag{2.6}$$

donde $M_{i,i}$ indica el número de muestras clasificadas correctamente de la clase $y_i$.

**Tasa de error.** Esta métrica se define como la ratio de las instancias clasificadas incorrectamente. El error se define como $1 - accuracy$.

**Sensibilidad.** También conocida como ratio de verdaderos positivos, relaciona los positivos predichos con el número de verdaderos positivos. La sensibilidad se define como: $TP/(TP + FN)$.

**Precisión.** Son los verdaderos positivos comparados con el número total de instancias predichas como positivas. La precisión se define como: $TP/(TP + FP)$.

**Especificidad.** También conocida como la ratio de verdaderos negativos, se define como el número de instancias negativas clasificadas correctamente dividido por el número de instancias negativas verdaderas. La especificidad se define como: $TN/(FP + TN)$.

**Falsa alarma.** También conocida como la ratio de falsos positivos, se define como el número de instancias falsas positivas comparado con el número de instancias negativas. La falsa alarma se define como: $FP/(TN + FP)$.

El principal inconveniente al aplicar una gran cantidad de medidas de rendimiento en la evaluación de clasificadores multiclase es que el grado de complejidad en el análisis de los resultados crece con el número de clases o el número de métricas usadas por cada clase. Por lo tanto, usaremos el *accuracy* como medida del rendimiento para evaluar el desempeño de los clasificadores.

### 2.4.2 Esquemas de evaluación

Los esquemas de evaluación nos ayudan a definir qué instancias son utilizadas para el entrenamiento y cuáles para las pruebas. La estrategia más básica sería utilizar el conjunto completo de datos tanto para el entrenamiento como para las pruebas. Al ser los mismos datos los resultados terminan siendo optimistas, por lo que la recomendación es usar datos diferentes para entrenar y para evaluar los clasificadores.

Si seleccionamos equivocadamente los datos usados para el entrenamiento podría ocurrir que las instancias seleccionadas reduzcan el poder del clasificador, ya que las seleccionadas para las pruebas podrían contener conocimiento relevante para el clasificador. En ese caso la pregunta que surge es: ¿cuál es la mejor estrategia para partir el conjunto de datos en instancias de entrenamiento y prueba y que, al mismo tiempo, dicha partición no impacte sobre el rendimiento del clasificador?

Existen varias alternativas para la selección de instancias de entrenamiento y de prueba. A continuación, se comentan las más importantes.

- **Método de retención.** Con este método se selecciona un porcentaje aleatorio de instancias para entrenar el modelo y otro para las pruebas. Generalmente, las instancias de entrenamiento son mayores que el número de instancias seleccionadas para las pruebas. Es posible que el método, al ser aleatorio, seleccione únicamente muestras relevantes o irrelevantes para distinguir ciertas clases. Este problema se puede solucionar utilizando un muestreo estratificado, en el que cada clase se muestrea de manera independientemente. Con esto se busca mantener la misma frecuencia de clases en las instancias de entrenamiento y en las de las pruebas.

- **Método de validación cruzada.** En este método, el conjunto de datos es dividido en k subconjuntos disjuntos. Uno de los k subconjuntos es usado para las pruebas y los otros (k  1) se usan para el entrenamiento. Este proceso es repetido para cada uno de los k subconjuntos de forma que todos ellos son usados una vez como conjunto de prueba. Finalmente, se promedia el error obtenido en cada iteración. La ventaja de este método es que todos los ejemplos del conjunto de instancias disponibles tienen la oportunidad de ser tratados como instancias de prueba. Otra forma de realizar la validación cruzada (especialmente recomendada cuando el conjunto de datos es muy pequeño) es la estrategia de «dejar uno fuera» (*leave one out*), donde k es igual al número de ejemplos de entrenamiento, y por lo tanto cada segmento de test contiene exactamente un ejemplo.

- **Método *bootstrapping*.** Este método realiza un muestreo con reemplazo para la creación del conjunto de instancias de entrenamiento. En un escenario típico, $n$ instancias son seleccionadas aleatoriamente. Ya que es un muestreo con remplazo, la probabilidad de que una instancias no sea seleccionada después de $n$ muestras es $(1 - 1/n)^n \approx 1/e$, siendo $e$ la base del logaritmo natural. Para el conjunto de test se usan las instancias no seleccionadas (*out-of-bag*). Este proceso se puede repetir varias veces, tal que por cada iteración de *bootstrapping* se genera una nueva muestra bootstrap, se entrena el modelo con ella y se evalúa con las instancias *out-of-bag*.

De los esquemas de validación vistos, una de las técnicas más usadas y potentes es la validación cruzada. Esta técnica es especialmente útil en los casos en los que no tenemos disponibles muchos datos de entrenamiento. Además, asegura que cada dato aparezca exactamente una vez en el conjunto de datos de prueba, por lo que ninguno se escapa ni del entrenamiento ni de la evaluación.

## 2.5 Comparando clasificadores

La gran variedad de algoritmos de clasificación y sus diferentes versiones nos obligan a conocer sus características y naturalezas para seleccionar el mejor clasificador para un problema específico. Con este objetivo, se han definido diferentes criterios para medir y evaluar el rendimiento de un clasificador. Para asegurarnos que existen diferencias entre los resultados obtenidos en la evaluación de dos o más clasificadores es necesario aplicar pruebas de significatividad estadística.

En general, las pruebas estadísticas asumen que las observaciones son independientes y parten de la hipótesis nula de que no existe una diferencia significativa en el rendimiento de los clasificadores evaluados. Además, es necesario establecer un nivel de significatividad fija para evaluar la hipótesis. Si al realizar la comparativa se acepta la hipótesis nula, quiere decir que no existe suficiente evidencia para decir que un clasificador es mejor que otro. Este hecho no garantiza que no haya diferencias, sino que este resultado puede ser causado por otros factores, como no utilizar suficientes datos en el análisis o que la probabilidad de rechazar correctamente la hipótesis nula de la prueba estadística utilizada es muy baja. En el caso contrario, si se rechaza la hipótesis nula quiere decir que sí existen diferencias y que estas son significativas.

Existen dos tipos de pruebas aplicables cuando realizamos una comparativa entre clasificadores: los *test* paramétricos y los no paramétricos. La diferencia

entre ellos es que, en los primeros, las observaciones deben cumplir con los supuestos relacionados con la distribución de las medidas de rendimiento seleccionadas. Sin embargo, en la mayoría de las comparaciones los supuestos no se cumplen, por esta razón no consideramos los *test* **t** y ANOVA [Fis56] para realizar comparaciones entre clasificadores y, en su lugar, usamos pruebas no paramétricas, aunque sean menos eficientes.

En el resto de las secciones adoptamos la terminología presentada en [JS11], en la cual las pruebas y sus hipótesis son presentadas con respecto a las medias de las muestras, donde las medias representan el promedio de las medidas de rendimiento de un clasificador, que son las salidas (muestras) del algoritmo sobre un conjunto de datos. La medida de rendimiento de un clasificador $f$ se denota con $pm(f)$ y puede ser cualquiera de las medidas de rendimiento estudiadas (por ejemplo, la tasa de aciertos).

En las siguientes secciones nos centraremos en revisar las pruebas no paramétricas aplicadas en los resultados de los experimentos de esta investigación.

### *2.5.1 Test Wilcoxon Signed-Rank*

El *test* de Wilcoxon [Wil92], también conocido como prueba $T$ de Wilcoxon es útil cuando se requiere comparar el desempeño de dos modelos sobre varios conjuntos de datos. Si consideramos dos clasificadores $f_1$ y $f_2$ con sus medidas de rendimiento $pm(f_1)$ y $pm(f_2)$, el procedimiento es el siguiente:

- Para cada prueba $i, i \in \{1, 2, \ldots, n\}$, calculamos la diferencia entre las medidas de rendimiento de los dos clasificadores $d_i = pm_i(f_2) - pm_i(f_1)$, donde el subíndice $i$ denota la cantidad correspondiente a la i-ésima prueba.

- Asignamos un rango de orden a todos los valores absolutos de $d_i$. Es decir, $|d_i|$. La función $rank()$ representa el rango $d_i$ dentro de todos los $d$. En el caso de un empate, asignamos rangos promedio a cada empatado $d_i$.

- A continuación, se calcula la suma de rangos:

$$W_{s1} = \sum_{i=1}^{n} I(d_i > 0) rank(d_i), \tag{2.7}$$

$$W_{s2} = \sum_{i=1}^{n} I(d_i < 0) rank(d_i), \tag{2.8}$$

- Asumiendo que hay un número $r$ de diferencias cuyos valores son cero, hay dos enfoques para tratar esta cuestión. El primero es ignorar estas diferencias, en cuyo caso $n$ toma el nuevo valor de $n - r$ y se calcula de la manera mostrada anteriormente. Otro enfoque es dividir los rangos de estas $r$ entre $W_{s1}$ y $W_{s2}$ por igual. Si $r$ es impar, simplemente ignoramos uno de los valores con diferencias de cero. Por lo tanto, en este caso, $n$ conserva su valor original (excepto cuando $r$ es impar, donde $n = n - 1$). Los $W_{s1}$ y $W_{s2}$ se definen como:

$$W_{s1} = \sum_{i=1}^{n} I(d_i > 0) rank(d_i) + \frac{1}{2} \sum_{i=1}^{n} I(d_i = 0) rank(d_i),$$
$$W_{s2} = \sum_{i=1}^{n} I(d_i < 0) rank(d_i) + \frac{1}{2} \sum_{i=1}^{n} I(d_i = 0) rank(d_i),$$

- Posteriormente se realiza el calculo de la estadística $T_{wilcox}$ como $T_{wilcox} = min(W_{s1}, W_{s2})$. En el caso de $n$ más pequeñas ($n \leq 25$), los valores críticos exactos de $T$ pueden ser consultados en la tabla de valores críticos de $T_{wilcox}$ para verificar si la hipótesis nula puede ser rechazada. En el caso de las $n$ más grandes, la distribución de $T_{wilcox}$ puede ser aproximada normalmente.

- La hipótesis nula se rechaza cuando el valor $T_{wilcox}$ es más pequeño que los valores críticos para $n$ o con la prueba de significación apropiada considerando las tablas respectivas.

### *2.5.2 Test de Friedman*

El *test* de Friedman [Fri37] realiza un análisis basado en los rangos de cada clasificador sobre cada conjunto de datos y no explícitamente sobre la medida de rendimiento, y reporta si hay una diferencia significativa entre los rankings promedios de dos o más modelos. Si consideramos $n$ conjuntos de datos y $k$ clasificadores para evaluar, entonces el proceso sigue los siguientes pasos:

- Para cada conjunto de datos por separado, los clasificadores se ordenan en sentido descendente de acuerdo a la medida de rendimiento $pm$, desde

el clasificador de mejor rendimiento hasta el peor. Es decir, si para el conjunto de datos $S_i$, el clasificador $f_j$ satisface que $pm_{S_i}(f_j) > pm_{S_i}(f'_j)$, $\forall j'$, $j' \in \{1, 2, \ldots, k\}$, $j \neq j'$, entonces es rankeado en primera posición. En el caso de un empate justo después del rango $r$, se asigna un rango de $[(r+1) + (r+2) + \cdots + (r+d)]/d$ a cada uno de los clasificadores empatados.

- Consideremos $R_{ij}$ como el rango de el clasificador $f_j$ sobre el conjunto de datos $S_i$.
- Calculamos las siguientes cantidades:
  - El rango medio del clasificador $f_j$ sobre todos los conjuntos de datos es:
$$\overline{R}_{.j} = \frac{1}{n} \sum_{i=1}^{n} R_{ij}$$
  - El rango medio global:
$$\overline{R} = \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} R_{ij}$$
  - La suma de los cuadrados totales que denota la variación en los rangos:
$$SS_{Total} = n \sum_{j=1}^{k} (\overline{R}_{.j} - \overline{R})^2$$
  - La suma de los cuadrados del error que denota el error de la variación:
$$SS_{Error} = \frac{1}{n(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} (R_{ij} - \overline{R})^2$$
- La prueba estadística conocida como el «estadístico de Friedman» se calcula con:
$$\chi_F^2 = \frac{SS_{Total}}{SS_{Error}}$$
- De acuerdo con la hipótesis nula, que establece que todos los clasificadores son equivalentes en su desempeño y, por lo tanto, sus rangos promedio

$R_{.j}$ deberían ser iguales, el $\chi_F^2$ sigue una distribución de $\chi^2$ con $k-1$ grados de libertad para $n$ grandes ($> 15$) y k ($> 5$):

- En el caso de valores grandes de $n$ y $k$, $\chi_F^2$ puede ser buscada en la tabla de la distribución $\chi^2$. Si el valor de $p$, que es este caso viene dado por $P(\chi_{k-1}^2 \leq \chi_F^2)$, es menor que el valor crítico para el deseado nivel de significación, entonces se rechaza la hipótesis nula.
- En el caso de $n$ y $k$ pequeños, la aproximación de $\chi^2$ es imprecisa y se aconseja buscar el valor de $p$ en las tablas $\chi_F^2$.

Existen algunas alternativas al test de Friedman descritas y evaluadas con múltiples clasificadores en [Dem06]. Una variación es el test de Friedman de rangos alineados [JL62], en el que se calcula el rendimiento medio alcanzado por cada modelo en cada conjunto de datos, llamado valor de localización. A continuación, se calculan las diferencias entre el rendimiento obtenido por cada clasificador con respecto al valor de localización. Este paso se repite para todos los clasificadores y conjuntos de datos. Las diferencias resultantes, llamadas observaciones alineadas, se ordenan de forma relativa unas con otras. Los rangos de orden asignados a las observaciones alineadas se denominan *rankings* alineados.

Otro test alternativo a Friedman para realizar múltiples comparaciones entre clasificadores es el test de Quade [Qua79]. Esta prueba tiene en cuenta el hecho de que algunos problemas son más difíciles que otros. Por tanto, los rangos de orden calculados sobre cada problema se podrían escalar en función de las diferencias observadas en los rendimientos de los algoritmos, y se obtendrá como resultado un análisis de ranking ponderado de la muestra de resultados.

### *2.5.3 Test de Iman y Davenport*

El test de Iman y Davenport [ID80] es menos exigente y riguroso que el test de Friedman pero igualmente fiable. Se basa en una distribución $F$ con $k-1$ y $(k-1)(n-1)$ grados de libertad. Si el valor del estadístico $F_F$ es mayor que el valor crítico de $F$, se rechaza la hipótesis nula.

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1) - \chi_F^2}$$

### 2.5.4 Pruebas Post Hoc

Hasta ahora las pruebas presentadas evalúan si las diferencias en el rendimiento de los clasificadores usados con los conjuntos de datos son estadísticamente significativas. Sin embargo, para realizar un análisis más profundo se utilizan las pruebas *post hoc*, que ayudan a realizar un análisis por pares de modelos y distinguir cuales son realmente diferentes. Algunos de los procedimientos más conocidos son Bonferroni-Dunn [Dun61], el procedimiento de Li [Li08], el procedimiento de Hommel [Hom88] y el procedimiento Holm [Hol79]. En este estudio hemos usado el procedimiento de Holm, ya que no es demasiado conservador como Bonferroni, y por otro lado, el test de Li se usa normalmente como complemento al método de Holm.

## 2.6 Técnicas usadas en clasificación

Existen varias técnicas de aprendizaje utilizadas para resolver problemas de clasificación y cada una tiene características propias que permiten resolver diferentes problemas de clasificación. A continuación, se presentan brevemente algunas de las principales técnicas más utilizadas en clasificación.

### 2.6.1 Clasificadores bayesianos

Los modelos probabilísticos usan la inferencia estadística para encontrar la clase asignada a una instancia. La salida del algoritmo, en lugar de ser una etiqueta de clase, es la probabilidad *a posteriori* de que una instancia dada sea miembro de cada una de las posibles clases. Después de obtener la probabilidad *a posteriori*, se usa la teoría de la decisión para determinar la clase de cada nueva instancia. Por otro lado, la probabilidad *a priori* de una clase se puede calcular fácilmente como la fracción de las instancias de entrenamiento que pertenecen a cada clase.

Los clasificadores probabilísticos más populares son los clasificadores bayesianos. En ellos, para calcular la probabilidad *a posteriori*, se determina la probabilidad condicional de la clase y la probabilidad *a priori* de la clase por separado y luego se aplica el teorema de Bayes. El teorema de Bayes fue presentado después de su muerte en 1763, y expresa la probabilidad de un evento aleatorio $A$ dado $B$ en términos de la distribución de probabilidad condicional del evento $B$ dado $A$ y la distribución de probabilidad de solo $A$ [Bay91].

Consideremos una instancia $x$ descrita mediante $l$ diferentes atributos, donde los valores son $At = \{at_1, at_2, \ldots, at_l\}$ respectivamente. Se desea determinar la probabilidad *a posteriori* de que la clase $\mathcal{Y}(x)$ de la instancia $x$ sea $i$, dado los valores de los atributos, es decir, $P(\mathcal{Y}(x) = i|at_1, \ldots, at_l)$. Con este objetivo es posible usar la regla de Bayes de la siguiente forma:

$$P(\mathcal{Y}(x) = i|at_1, \ldots, at_l) = P(\mathcal{Y}(x) = i) * \frac{P(at_1 \ldots, at_l|\mathcal{Y}(x) = i)}{P(at_1, \ldots, at_l)} \tag{2.9}$$

Ya que el denominador es constante para todas las instancias y solamente se necesita determinar la clase con la máxima probabilidad *a posteriori* se puede aproximar la expresión a la siguiente:

$$P(\mathcal{Y}(x) = i|at_1, \ldots, at_l) \approx P(\mathcal{Y}(T) = i) * P(at_1 \ldots, at_l|\mathcal{Y}(x) = i) \tag{2.10}$$

La idea es que la expresión sea evaluada fácilmente a partir de los datos. Para esto se asume que el efecto de una característica es independiente de las otras características. Se conoce como Naïve Bayes (NB) al clasificador que mantiene esta suposición de independencia y realiza esta simplificación. Especialmente en la ecuación 2.10, la expresión $P(\mathcal{Y}(x) = i|at_1, \ldots, at_l)$ puede expresarse como el producto de las probabilidades condicionales en función de los atributos.

$$P(at_1, \ldots, at_l|\mathcal{Y}(x) = i) = \prod_{j=1}^{l} P(at_j|\mathcal{Y}(x) = i) \tag{2.11}$$

De esta forma las probabilidades pueden ser estimadas a partir de los datos de entrenamiento. El término $P(at_j|\mathcal{Y}(x) = i)$ se calcula como la fracción de los registros en la parte de los datos de entrenamiento correspondientes a la $i$-ésima clase, que contiene los valores de atributos del $j$-ésimo atributo. Existe la posibilidad de que los datos de entrenamiento contengan pocas o ninguna instancia de cierta clase con un valor específico de atributo. En este caso se debe aplicar un procedimiento de suavizado, ya que, de lo contrario no se podrá obtener la predicción. En el caso de que deseemos obtener la etiqueta de la clase, simplemente se debe tomar la clase con la más alta probabilidad calculada. El clasificador Naïve Bayes presenta buenos resultados, especialmente en la clasificación de textos con un alto número de características, a pesar de que el supuesto de independencia no es real en la mayoría de los problemas

de clasificación. En [ZJY21], se propone un algoritmo llamado Bayes ingenuo ponderado por atributos e instancias, el cual combina la ponderación de atributos con la ponderación de instancias en un marco uniforme. Los pesos de los atributos se incorporan en la fórmula de clasificación de NB, y luego se estiman las probabilidades previas y condicionales utilizando datos de entrenamiento ponderados por instancias.

### 2.6.2 Árboles de decisión

Cuatro generaciones de diferentes tipos de algoritmos de árboles de decisión se han presentado, según Wei-Yin Loh, desde 1963 hasta 2010 (ver más en [Loh11]). Un árbol de decisión es un clasificador que crea una partición jerárquica de los datos. Cada partición se realiza usando un criterio, que puede ser una condición sobre uno (univariante) o muchos atributos (multivariante). Este método trata de, recursivamente, dividir los datos de entrenamiento, de forma que se maximice la discriminación entre las diferentes clases en los nodos. Para ello, el criterio de partición elige en cada nodo un atributo con el que se realiza la partición de los ejemplos, lo que genera tantos nodos hijos como valores tenga dicho atributo en el caso de atributos nominales, o una partición binaria en el caso de atributos numéricos.

El criterio de partición más utilizado es la propiedad estadística llamada ganancia de información, la cual mide la reducción en la impureza, desorden o incertidumbre sobre un conjunto de ejemplos. La impureza de un nodo puede ser medida usando el gini-index y la entropía. En ambos se selecciona el atributo que parte los datos en conjuntos más homogéneos (con respecto a la clase de las instancias). Si consideramos que $p_1, \ldots, p_m$ es la fracción de los registros pertenecientes a cada una de las $m$ clases en un nodo $Nd$, entonces el gini-index $G(Nd)$ se define como: $1 - \sum_{i=1}^{m} p_i^2$.

El valor de G(Nd) está entre 0 y $1-1/m$. Mientras más pequeño sea el valor de $G(Nd)$, menor será la homogeneidad. Si las clases están balanceadas el valor será $1 - 1/m$. Por otro lado, una medida alternativa es la entropía E(Nd): $-\sum_{i=1}^{m} p_i * log(p_i)$.

El valor de la entropía se encuentra entre 0 y 1. Mientras más pequeña sea la entropía, mayor será la homogeneidad en los datos. Una vez calculada la impureza, es posible definir la ganancia de información como la reducción de esta impureza, causada por la partición de los ejemplos de acuerdo a los atributos. Si tomamos en cuenta la entropía, la ganancia de información $Gain(S, At)$ de un atributo $At$, relativo a un conjunto de ejemplos $D$ se define como:

$$Gain(D, At) = E(D) - \sum_{v \in Values(At)} \frac{|D_v|}{|D|} E(D_v) \tag{2.12}$$

donde $S$ es el conjunto de datos de entrenamiento, $At$ un atributo, $Values(At)$ son los posibles valores que puede tomar el atributo $At$ y $D_v$, el subconjunto de $D$ para el cual el atributo $A$ tiene el valor $v$.

Al construir el árbol, las divisiones deben realizarse de forma que minimicen la suma del *gini-index* o la entropía entre dos nodos. El proceso avanza recursivamente hasta que se cumpla el criterio de terminación, el cual puede ser cuando todos los registros pertenezcan a la misma clase y requieran un nivel mínimo de sesgo o pureza, o un número mínimo de registros en el nodo para evitar un sobreajuste.

Un ejemplo de árbol de decisión es el presentado en la figura 2.4. Esta muestra un árbol de decisión relacionado con la clasificación de un mango en dulce o no dulce. El clasificador binario presentado primero evalúa el color del mango y después la condición de suavidad. En este caso, si se cumple la condición de que el color está en el rango entre amarillo y rojo y es blando, entonces se clasifica como dulce.

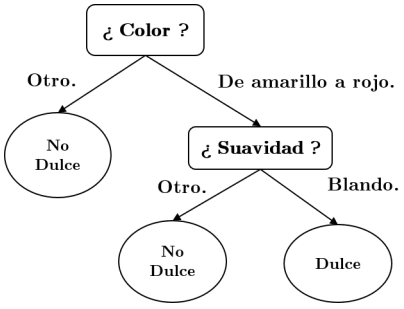



**Figura 2.4:** Ejemplo de árbol de decisión[SB14].

Un problema en la construcción del árbol de decisión es la complejidad de predecir su mejor momento para detener su crecimiento, a fin de evitar un sobreajuste. Por lo tanto, en muchas ocasiones, el árbol de decisión se poda para eliminar los nodos que puedan corresponder al sobreajuste. En general, las estrategias utilizadas para evitar un sobreajuste pueden ser de dos tipos: la pre-poda que detiene el crecimiento del árbol antes de que se llegue al punto de

clasificar perfectamente los datos, y la pos-poda que aplica un procesamiento posterior de poda después de que el árbol sobreajuste los datos.

Hay diferentes formas de podar el árbol de decisiones. Una es usar un principio de longitud mínima de descripción (Minimum Description Length, MDL) para decidir cuándo podar un nodo del árbol [Kon98]. Otro enfoque es mantener una pequeña porción de los datos de entrenamiento durante la fase de crecimiento. Luego se prueba para ver si reemplazar un subárbol con un solo nodo mejora la precisión de clasificación en el conjunto de datos. Si este es el caso, se realiza la poda. Existen estudios como [Qui87; Esp+97; Min89] en los cuales se examinan las ventajas y desventajas de estas y otras estrategias de poda.

En la fase de test, la instancia de prueba sigue la ruta apropiada en el árbol de decisión, en función de la evaluación de los criterios de división en el proceso de decisión jerárquico. La etiqueta de clase del nodo hoja es asignada a la instancia al final del proceso.

Una de las ventajas de los árboles de decisión es la posibilidad de comprender e interpretar los resultados aprendidos. Cada rama del árbol, desde la raíz hasta las hojas, puede expresarse como una regla condicional de la forma «Si *condición* entonces *clase*», donde *condición* es una conjunción de las condiciones en los nodos internos de la rama y *clase* es la etiqueta de clase en la hoja. Además, este método puede trabajar con datos incompletos que contengan atributos irrelevantes, datos con alta dimensionalidad o conjuntos de datos muy grandes.

Las implementaciones del algoritmo de árbol de decisión más conocidas son: ID3 [Qui86], C4.5 [Qui14], C5.0 [PG09] y CART [SC09]. Existen múltiples aplicaciones posibles de los árboles de decisión, por ejemplo para predecir enfermedades cardiovasculares [KG11], también se ha usado el algoritmo ID3 conocido como "J48" para predecir el estado laboral de estudiantes después de su graduación [JT13], otro ejemplo es predecir las tasas de respuesta de las encuestas de satisfacción, actitud y lealtad del consumidor [Han+19].

### *2.6.3 Métodos basados en reglas*

Se puede decir que los métodos basados en reglas son más generales que los árboles de decisión, ya que los segundos, al tener una estructura jerárquica fija, no permiten la superposición de reglas, lo que no es un límite para los métodos basados en reglas.

Durante la fase de entrenamiento, se codifica el conocimiento contenido de las instancias a través de reglas, y en la fase de prueba se determina qué reglas son relevantes para realizar la predicción. El proceso de extraer una regla desde los datos de entrenamiento se conoce como el crecimiento de la regla. El resultado de la clasificación se basa en combinar las reglas relevantes para, finalmente, obtener la etiqueta que será asignada a la instancia de prueba.

Las reglas están formadas por dos componentes: la primera, la parte izquierda, se llama antecedente, y la otra, la parte derecha, se llama consecuente. En el caso del ejemplo que refiere a la clasificación de la dulzura de un mango, las reglas extraídas son:

$$R1 : (Color = Amarillo)\&(Suavidad = Blando) \Rightarrow (Dulzura = 1)$$
$$R2 : () \Rightarrow (Dulzura = 0)$$

En muchos casos, es posible crear reglas que están en conflicto entre ellas en el consecuente de la regla para una instancia de prueba en particular. Por lo tanto, es importante diseñar métodos que puedan determinar efectivamente una solución a estos conflictos. Una opción es aplicar las reglas en el orden en que han sido generadas, evitando aplicar dos reglas a una misma instancia. Si no existe un orden entre las reglas, se aplican todas las posibles y la clase que se le asigne se decide por voto mayoritario.

Se han propuesto varios algoritmos basados en reglas que utilizan distintos métodos de inducción. Entre ellos están los Clasificadores Basados en Asociaciones (CBA) [MLH98], CN2 [CN89] y RIPPER [CS99].

RIPPER y CN2 usan el paradigma de cobertura secuencial, en el que las reglas con alta precisión y cobertura se extraen secuencialmente de los datos de entrenamiento. La idea es crear una regla correspondiente a una clase específica, y luego eliminar todas las instancias de entrenamiento que coincidan (cubran) el antecedente de esta regla. Este enfoque se aplica repetidamente hasta que solo queden instancias de entrenamiento de una clase en particular en los datos. Cuando no se active ninguna regla, se selecciona una clase predeterminada asignada previamente para estos casos.

La especialización de una regla implica la adición de conjunciones sucesivas al lado izquierdo después de seleccionar una clase consecuente particular. Luego de la fase de crecimiento, se entra a la fase de poda de las reglas, la cual es similar a la fase con el mismo nombre utilizada en los árboles de decisión. Las reglas son priorizadas en el mismo orden que se extrajeron de los datos de

entrenamiento. Para una instancia de prueba, se le asigna el valor de clase en el consecuente de la primera regla aplicable o, en caso de que ninguna regla se pueda aplicar, la predeterminada.

La interpretabilidad de los métodos basados en reglas permite que se utilicen en muchas aplicaciones prácticas. Por ejemplo, en [XDP18], se propone un modelo híbrido que combina RIPPER y un método de selección de atributos para predecir el riesgo crediticio en la banca comercial. Otra ejemplo es utilizar la inducción de reglas, para identificar los niveles de gravedad de los trastornos del movimiento en personas que utilizan sistemas de realidad virtual [Pan+18].

### *2.6.4 Aprendizaje basado en instancias*

El aprendizaje basado en instancias se caracteriza por prescindir de la fase de entrenamiento del modelo. La instancia que debemos predecir está directamente relacionada con las instancias de entrenamiento para estimar su valor de clase. Esta estrategia es conocida como «aprendizaje perezoso», porque se espera hasta tener un conocimiento de las instancias de prueba para obtener la predicción; es decir, no se genera un modelo o clasificador a partir de los datos de entrenamiento, sino que éstos se usan directamente para hacer la predicción.

La ventaja de estos métodos radica en que son rápidos en el aprendizaje, ya que, en realidad, no se realiza un aprendizaje como tal, aunque el proceso de predicción es más costoso, porque se deben determinar las instancias locales relevantes y, con esta información, obtener la etiqueta asignada. Por otro lado, cuando trabajamos directamente con los datos, puede producirse un problema del sobreajuste y generar una mayor dificultad respecto a la generalización.

**k-Vecinos más cercanos (*k-Nearest Neighbors*, kNN).**

Fue desarrollado por Thomas Cover en 1967 [CH67], es un algoritmo simple basado en instancias, en el cual, dada una instancia, se buscan sus $k$ vecinos más cercanos entre los datos de entrenamiento. La clase con mayor presencia entre los $k$ vecinos es la clase resultante de la clasificación; por lo tanto, el resultado de la clasificación puede variar dependiendo del valor de $k$. Si $k = 1$, entonces a la instancia se le asigna simplemente la clase de ese único vecino más cercano. La figura 2.5 muestra un problema de clasificación binario con un número de vecinos $k = 3$ y $k = 6$. Se puede observar que para $k = 3$ la instancia (denotada por una estrella) es clasificada como perteneciente a la clase círculo oscuro, mientras que $k = 6$ es clasificada como círculo claro.

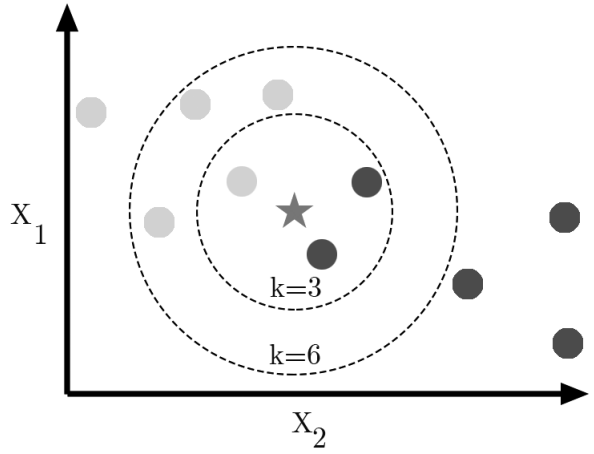



**Figura 2.5:** Ejemplo de modelo *K-Nearest Neighbor* [Com07].

Es posible definir variaciones del algoritmo de aprendizaje básico basado en instancias: por ejemplo, se pueden crear pequeños grupos a partir de las instancias de cada clase y el centroide de cada grupo puede usarse como referencia para encontrar los vecinos más cercanos. Otra forma de aprendizaje basado en instancias es aquella en la que los vecinos más cercanos no se usan directamente para obtener la etiqueta de salida, sino que, en su lugar, se crea un clasificador optimizado localmente utilizando los ejemplos en la vecindad. En este caso, son técnicas de clasificación para crear los clasificadores locales. Se pueden encontrar más detalles sobre los métodos basados en instancias en las obras de [AKA91],[WAM97].

### *2.6.5 Máquina de vectores de soporte*

La máquina de vectores de soporte o *Support vector machine* (SVM), fue desarrollada por Vladimir Vapnik en 1995 [CV95] y está basada en la idea de encontrar el hiperplano que mejor separe los puntos de una clase de la de otra. Se busca que este tenga la máxima distancia (margen) con los puntos que están más cerca de él. De esta forma, las instancias de prueba representadas por su vector de características serán etiquetadas de una clase u otra dependiendo en qué lado del hiperplano se ubiquen. Al vector formado por los puntos más cercanos al hiperplano se le conoce como vector de soporte y se usa para encontrar la ubicación óptima del hiperplano dentro del espacio. La figura 2.6 muestra un ejemplo de problema de clasificación binaria mostrada en dos dimensiones

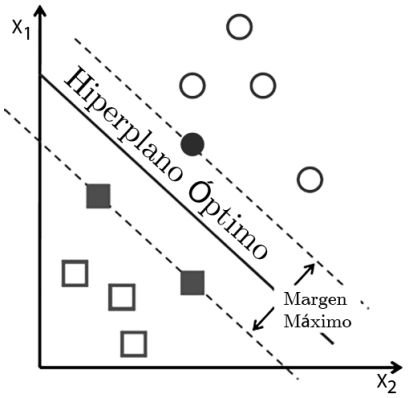



**Figura 2.6:** Ejemplo de máquina de soporte de vectores [Dra19].

y su correspondiente hiperplano encontrado por una máquina de soporte de vectores.

Sin embargo, en ocasiones, las instancias de entrenamiento tienen ruido, lo cual distorsiona la capacidad para realizar buenas predicciones. En ese caso es posible reducir el margen, de forma que permita reducir los errores en la clasificación. En escenarios en los que no es posible la separación lineal frecuentemente se utilizan funciones de Kernel para proyectar la información a un espacio de mayor dimensión, asilo que aumenta las capacidades del algoritmo.

Existen múltiples aplicaciones de SVM. Por ejemplo, los autores en [AGS20] incluyeron SVM en un dispositivo de bajo costo y alto rendimiento, fabricado para mejorar la detección temprana del melanoma en el ámbito de la atención primaria de salud. En [Al-+19], los autores propusieron un sistema de categorización de texto que combina clasificadores binarios entrenados con diferentes conjuntos de dicotomías, donde el número de ejemplos de entrenamiento aumenta exponencialmente cuando se comparan con el conjunto original. Esta propiedad permite entrenar con diferentes datos sin reducir el número de ejemplos o características. Así, es posible componer un conjunto con clasificadores diversos y fuertes. Los autores de [Tao+16], propusieron una SVM sensible a la localidad utilizando un algoritmo de combinación de kernel para resolver problemas de la detección de rostros. La SVM sensible a la localidad se emplea para construir un modelo local en cada región local, que maneja la tarea de clasificación. Luego, se emplean múltiples redes neuronales convolucionales locales para aprender conjuntamente los rasgos faciales.

### 2.6.6 Red neuronal artificial (Artificial neural networks)

Desde la primera aplicación de redes neuronales presentada por Bernard Widrow y Marcian Hoff en 1959, las redes neuronales han tenido un desarrollo constante y han sido clave en la popularización del aprendizaje automático.

Las redes neuronales artificiales simulan la red neural del cerebro humano. Este proceso se simplifica a través de unidades artificiales llamadas neuronas, las cuales están enlazadas formando una red intercomunicada. Una red neuronal se puede describir como un grafo dirigido, formado por neuronas y enlaces.

Una neurona artificial está compuesta por un conjunto de entradas que pueden provenir del exterior o de otras neuronas, un peso sináptico que representa el grado de comunicación entre la neurona $i$ y la neurona $j$, reglas de propagación que integran la información de distintas neuronas y proporcionan el valor del potencial post-sináptico de la neurona, la función de activación que provee el estado de activación actual de la neurona y la función de salida representa la salida actual de la neurona.

La entrada de una neurona es la suma ponderada de las salidas de las neuronas conectadas a ella. Los nodos de salida están asociados con un conjunto de pesos que definen la fuerza de la conexión sináptica entre las neuronas. El peso actúa como excitador en caso de una entrada positiva, mientras que un peso negativo actúa como inhibidor y, si el peso es cero, no existe comunicación entre el par de neuronas.

La topología de una red neuronal describe la organización y disposición de las neuronas dentro de la red. Esta viene dada por el número de capas, la cantidad de neuronas por capa, el grado de conectividad y el tipo de conexión existente entre neuronas. Las redes neuronales pueden organizarse con una capa de entrada, una o varias capas ocultas y la capa de salida.

Un tipo más básico de red neuronal es el *perceptrón*, el cual implementa un mecanismo de entrenamiento que le permite determinar automáticamente los pesos sinápticos que clasifican correctamente a una nueva instancia. El perceptrón está compuesto por dos capas: una de entrada y otra de salida. La primera recibe las entradas del exterior y transmite esta información a las neuronas de salida, que es donde se realiza el procesamiento. En el caso de que la red tenga una capa de entrada, una o varias capas ocultas y una capa de salida, se la conoce como red multicapa. Un ejemplo de esto se muestra en la figura 2.7, en este caso, un número de $n$ neuronas que son parte de la capa de entrada, $m$ neuronas que forman la oculta y una neurona que forma parte de la capa de salida.

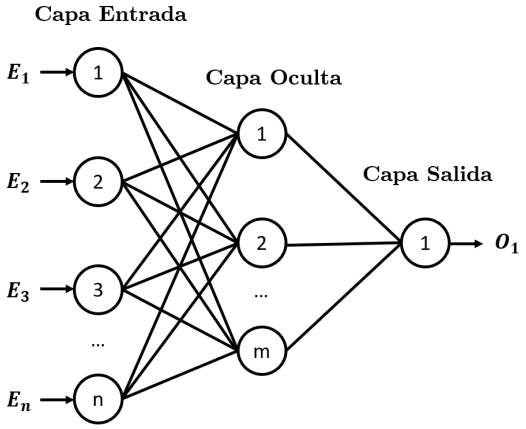



**Figura 2.7:** Ejemplo de red neuronal multicapa. [Com04]

Al entrenar una red neuronal se busca establecer los pesos entre conexiones a partir de la información contenida en los datos de entrenamiento de forma que se minimice el error de clasificación. Por lo general, se comienza con pesos seleccionados aleatoriamente y se van ajustando gradualmente según el error encontrado. El entrenamiento se realiza en dos fases. Primero, la función de activación se aplica repetidamente para propagar las entradas en una dirección hacia delante. Como se conoce la etiqueta real de la instancia de entrenamiento, se puede calcular el valor del error cometido. En la segunda fase, el error es usado para actualizar los pesos de la capa de salida y propagar los pesos hacia atrás al resto de las capas.

Las redes neuronales pueden aproximar bien cualquier función, aunque el precio de esta generalidad es que el entrenamiento puede tomar mucho tiempo, además de ser sensibles al ruido y, a veces, tienden a sobre-ajustarse a los datos de entrenamiento. Si queremos crear un clasificador multiclase, se puede usar una estrategia uno-contra-todos o crear una arquitectura de red en la que el número de nodos de salida corresponda con el número de etiquetas de clase. En este caso cada salida está relacionada con una etiqueta del problema de clasificación.

## 2.7 Resumen

En este capítulo se han revisado los conceptos básicos de clasificación. Se comenzó definiendo el problema de clasificación y sus partes esenciales, y se enmarcó el proceso de creación de un clasificador dentro de una aplicación de aprendizaje automático. También se revisaron los esquemas utilizados para la validación de un clasificador, su evaluación a través de métricas y se presentaron las pruebas estadísticas utilizadas para la comparación de uno o varios modelos de clasificación. Finalmente, se realizó una revisión de las principales técnicas de clasificación desde un punto de vista general. Estos conceptos básicos son fundamentales y serán utilizados en el desarrollo de los siguientes capítulos.

# Capítulo 3

# Mejoras en la clasificación multiclase

En el capítulo anterior revisamos los conceptos básicos de clasificación, las medidas de evaluación, las estrategias para comparar modelos y las principales técnicas de aprendizaje automático usadas para la clasificación. En este capítulo introduciremos algunas técnicas utilizadas para mejorar la precisión en los modelos, analizando, en particular, las estrategias que se usan para descomponer los problemas de clasificación multiclase, los *ensembles* y la clasificación jerárquica.

## 3.1 Estrategias para obtener una mejor clasificación

Uno de los principales objetivos al resolver una tarea de clasificación es poder hacer las mejores predicciones posibles, en las que la noción de «mejor» depende de la medida de evaluación utilizada para valorar la calidad del clasificador. Sin embargo, la obtención de buenas predicciones no siempre es una tarea sencilla y suele complicarse aún más en el caso de los problemas multiclase, ya que en este caso el clasificador tiene que distinguir entre un gran número de clases para hacer las predicciones. Por esta razón, en las últimas

décadas, se han realizado numerosos esfuerzos para mejorar el rendimiento de los clasificadores.

La mayoría de los enfoques propuestos en la literatura se encuadran en uno de estos tres grupos:

A) Diseñar mejores técnicas de aprendizaje. Como ejemplo podemos mencionar la noción de *ensemble learning* [Die00], un término general basado en la idea de utilizar más de un modelo para determinar el resultado previsto.

B) Aplicar algún tipo de transformación sobre los datos de entrenamiento. Los diferentes enfoques basados en la selección de instancias [Olv+10; Bla19] o atributos [DL97; Li+18], y los métodos de sobremuestreo y submuestreo propuestos para tratar conjuntos de datos desbalanceados [Cha+02; MT14; TE93] son ejemplos de propuestas que pertenecen a este grupo.

C) Modificar o ajustar las predicciones dadas por el clasificador. A este grupo pertenecen los enfoques basados en los métodos de selección de umbrales [Yan01] o las técnicas de calibración de clasificadores [Bel+10; CC06].

Todos estos métodos asumen que se cumple la condición de que no existe ninguna relación entre las etiquetas de clase, es decir, que son independientes entre sí.

Sin embargo, en la clasificación multiclase se ha demostrado que el rendimiento de la clasificación puede también mejorarse descomponiendo el problema en una colección (o incluso una jerarquía) de problemas de clasificación más pequeños o menos complejos que el original. Se han explorado dos formas alternativas de descomponer el problema original: la primera se basa en la idea de que un problema se vuelve menos complejo si se reduce su dimensionalidad, mientras que la segunda se apoya en la idea de que un problema multiclase se vuelve más simple si se reduce el número de clases.

A continuación, revisaremos algunas de las estrategias mencionadas para mejorar el rendimiento de los clasificadores. Primero, exploraremos los métodos multiclasificadores (*ensembles*), posteriormente estudiaremos varias formas de descomponer los problemas de clasificación y, finalmente, analizaremos técnicas de clasificación jerárquica.

## 3.2 Aprendizaje por ensembles

El término *ensemble* hace referencia a un sistema formado por múltiples clasificadores que tienen como objetivo tomar mejores decisiones que las tomadas con un solo clasificador [OM99]. Las decisiones basadas en *ensembles* son parte de nuestra vida cotidiana, por ejemplo, en el sistema de justicia, la decisión de culpabilidad o inocencia la realiza un panel de jueces o miembros de un jurado. En un *ensemble*, cada clasificador representa a un experto que aporta criterios propios a la predicción final.

En un problema de aprendizaje, cada modelo hace una predicción diferente para luego combinarse entre todas y obtener una predicción única. Cada modelo se conoce como un modelo débil, que puede ser utilizado como bloque de construcción para diseñar modelos más complejos combinando varios de ellos. La mayoría de las veces, los modelos débiles no funcionan muy bien por sí mismos, ya sea porque tienen un alto *bias* o porque tienen demasiada variabilidad para ser robustos. Por lo tanto, la idea de los métodos *ensemble* es tratar de reducir el *bias* o la varianza de estos predictores débiles combinando varios de ellos para crear un modelo fuerte (o modelo de conjunto) que logre mejores resultados.

Consideremos un problema de clasificación, en el cual se debe asignar una clase a una instancia nueva usando un *ensemble* de $n$ clasificadores, $\mathcal{S}_w$, $w \in \{1, 2, \ldots, n\}$. Según la teoría de los sistemas de clasificación múltiple, si los errores cometidos por un clasificador $\mathcal{S}_i$ son diferentes a los errores cometidos por otro clasificador $\mathcal{S}_j$, entonces es posible explotar estas diferencias para obtener una clasificación más precisa y confiable de la instancia nueva. Tomemos, por ejemplo, un caso en el que tengamos tres clasificadores binarios y pretendamos realizar una clasificación de las instancias presentadas en la figura 3.1. Los puntos blancos representan los ejemplos incorrectamente clasificados, mientras que los puntos oscuros representan las instancias correctamente clasificadas; como se puede observar, cada clasificador cometió errores en diferentes instancias. Este problema se corregiría realizando una votación simple.

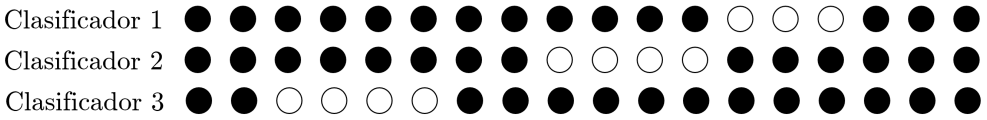



**Figura 3.1:** Ejemplo de error de clasificación por un *ensemble*. [Kub17]

El uso de *ensembles* para obtener modelos robustos ha sido aplicado exitosamente por participantes ganadores de muchas competencias de minería de

datos, como son los concursos *Netflix Prize* y KDD [KM]. A pesar de que el objetivo de los *ensembles* es mejorar el rendimiento, también existen otros usos, como la selección de atributos, la detección de valores faltantes, el aprendizaje incremental [GS09] y la fusión de datos.

El proceso de crear un *ensemble* se realiza en dos fases. Primero se construyen los clasificadores base y luego se combinan las predicciones realizadas [Die00]. Una forma de crear los clasificadores es seleccionar las instancias de entrenamiento que servirán para su creación. Se debe considerar que el *ensemble* resultante debe estar compuesto por clasificadores que no estén altamente correlacionados, ya que, en este caso, los clasificadores harán predicciones similares y no mejorará la clasificación final. Con el objetivo de obtener una colección de clasificadores de base diversos, se pueden aplicar las siguientes técnicas:

- Extraer diferentes subconjuntos de ejemplos o de atributos y entrenar un clasificador por cada subconjunto.
- Aplicar diferentes algoritmos sobre el conjunto de entrenamiento.
- Incluir aleatoriedad en el proceso de aprendizaje de un algoritmo en particular o utilizar diferentes parámetros en los modelos de clasificación.

Las estrategias para combinar predicciones pueden ser de dos tipos: sin pesos y con pesos. La primera basa su predicción en el voto de la mayoría. Esta estrategia cuenta el número de votos de cada etiqueta predicha por cada clasificador y selecciona la más alta como la etiqueta asignada a la instancia. Por otro lado, una combinación con pesos asigna pesos a cada clasificador. Los pesos más altos se asignan a los clasificadores con mayor precisión (o mayor confianza en sus predicciones), de esta forma, la decisión final será más próxima a la correcta. A continuación, se presentan varias de las estrategias usadas para el entrenamiento de los miembros de un *ensemble*.

- ***Bagging*** [Bre96]. Este método permite combinar varios modelos de clasificación, en los cuales cada uno se entrena con diferentes conjuntos de entrenamiento. Estos grupos de instancias se forman haciendo un muestreo con repetición del conjunto de datos de entrenamiento. Para obtener la decisión final es posible realizar conteos con voto mayoritario o, en el caso que se usen clasificadores probabilísticos, se puede usar una media del valor de probabilidad por clase. Por ejemplo, el algoritmo *Random Forest* [Bre01], utiliza *bagging* para que crear un conjunto de árboles de decisión, en este caso, cada modelo es entrenado con datos ligeramente

distintos. La predicción de una nueva instancia se obtiene agregando las predicciones de todos los árboles individuales.

- ***Boosting*** [Bre+98]. Es una técnica que consiste en ajustar secuencialmente múltiples modelos débiles de forma adaptativa. Cada modelo se ajusta secuencialmente dando más importancia a las observaciones que fueron mal clasificadas previamente; por lo tanto, cada clasificador aprende de los errores anteriores. Esto es posible gracias a la asignación de mayores pesos a muestras mal clasificadas y un menor peso a muestras correctamente clasificadas.

  Un método que utiliza el principio de boosting es *Gradient Boosting* [Fri01], el cual genera un conjunto de árboles de decisión secuencialmente. La predicción se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo. La flexibilidad de este algoritmo hace posible aplicarlo a regresión o clasificación. XGBoost [CG16] es una implementación de *Gradient Boosting* que tiene ventajas como que permite una ejecución en paralelo, entrenamiento incremental y es aplicable a grandes bases de datos.

- ***Staking*** [Wol92]. Este método apila modelos de clasificación; igual que una red neuronal, las entradas de un modelo son las salidas de otros. La idea de la acumulación es aprender varios modelos débiles diferentes y combinarlos entrenando un meta modelo para producir un resultado basado en las múltiples predicciones devueltas por estos modelos débiles. Por lo tanto, necesitamos definir dos cosas para construir nuestro modelo de apilamiento: los $n$ predictores que queremos combinar y el meta modelo que los combina.

## 3.3 Descomposición de problemas de clasificación

Descomponer un problema puede ser una buena forma de fragmentar un problema complejo en subproblemas más pequeños, comprensibles y manejables [RM09]. Las principales ventajas de descomponer un problema de clasificación son:

- Se puede mejorar el *accuracy* de los métodos de clasificación. Esto puede ser explicado por el equilibro total que se logra entre el *bias* y la varianza, ya que se crea un conjunto de submodelos de menor complejidad, en el cual es posible fijar el nivel de ambos. La mejora en el rendimiento

también se puede explicar por la capacidad de explotar las habilidades especializadas de cada componente.

- La posibilidad de aplicar algoritmos de aprendizaje sobre grandes conjuntos de datos que tienen muchos atributos o una gran cantidad de instancias. Además, la descomposición reduce el volumen de los datos que se procesan a la vez.
- Un claro efecto de su uso es que el problema puede ser visto y mejorado desde el punto de vista de cada uno de sus componentes y no es necesario tratarlo como un todo.
- Es posible aumentar la comprensibilidad del modelo, ya que, en lugar de un modelo complicado, se crean varios modelos más simples.
- La modularidad de la solución permite una mayor facilidad para el mantenimiento del modelo global. Por ejemplo, en el caso de un aprendizaje incremental se puede aplicar un proceso de adaptación de los modelos focalizando en aquellos que son afectados por los nuevos datos de entrenamiento.

A continuación, revisaremos las principales estrategias de descomposición utilizadas para resolver problemas de clasificación multiclase.

## 3.4 Descomposición de multiclase a binario

Una aplicación práctica de la descomposición es posibilitar la transformación de un problema multiclase en varios subproblemas binarios. De esta forma, es posible utilizar algoritmos pensados para trabajar únicamente con problemas de clasificación binarios en problemas multiclase. Uno de estos casos es la máquina de soporte de vectores [MGC09], que inicialmente fue presentado como un algoritmo de clasificación binario y luego fue exitosamente aplicado a problemas multiclase. Esta transformación se puede reproducir con cualquier algoritmo de clasificación binario.

Revisaremos algunas de las estrategias para descomponer un problema de clasificación multiclase en múltiples subproblemas binarios.

| Clase | Clasificadores | | | | |
|---|---|---|---|---|---|
| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
| $c_1$ | + | - | - | - | - |
| $c_2$ | - | + | - | - | - |
| $c_3$ | - | - | + | - | - |
| $c_4$ | - | - | - | + | - |
| $c_5$ | - | - | - | - | + |

**Tabla 3.1:** Ejemplo de la estrategia uno contra todos

### *3.4.1 Uno contra todos (One versus all, OVA)*

Este método crea un clasificador binario por cada etiqueta del problema de clasificación bajo el criterio de que cada clasificador distinga entre una clase específica y el resto. Como se menciona en [RK04], este esquema ha sido propuesto numerosas veces por varios investigadores. Si consideramos un problema de clasificación con un número $m$ de clases $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$, entonces se entrenarán un número $m$ de clasificadores $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$. En particular, el clasificador $f_1$ tendrá como ejemplos positivos las instancias que son de la clase $y_1$ y como negativos los ejemplos correspondientes a todas las clases que no son $y_1$; por lo tanto, el conjunto de instancias de entrenamiento serán las instancias pertenecientes a las clases $\{y_2, y_3, \ldots, y_m\}$.

Tomamos como ejemplo un problema de clasificación de cinco clases y la función $f_i$ como el $i$-ésimo clasificador que decide la clase $y_i$ , donde $(+)$ corresponde a la etiqueta de clase positiva y $(-)$ corresponde a la etiqueta de clase negativa. La tabla 3.1 ejemplifica el método «Uno contra todos»: las columnas de la tabla son los clasificadores binarios entrenados con las clases positivas y negativas; para obtener la predicción de la instancia $r$, se aplican todos los clasificadores a la nueva instancia. Finalmente, la clase asignada a la instancia corresponde a la clase positiva asignada por uno de los clasificadores.

Como se podría pensar, el problema aparece cuando varios clasificadores, o bien ninguno de ellos, predicen una clase positiva para una instancia. En este caso, es posible aplicar un método *post-hoc*. En el caso de los clasificadores que obtienen probabilidades de pertenencia, se seleccionará el que tenga la mayor probabilidad.

| Clase | Clasificadores | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{23}$ | $f_{24}$ | $f_{25}$ | $f_{34}$ | $f_{35}$ | $f_{45}$ |
| $c_1$ | + | + | + | + | 0 | 0 | 0 | 0 | 0 | 0 |
| $c_2$ | - | 0 | 0 | 0 | + | + | + | 0 | 0 | 0 |
| $c_3$ | 0 | - | 0 | 0 | - | 0 | | + | + | 0 |
| $c_4$ | 0 | 0 | - | 0 | 0 | - | 0 | - | 0 | + |
| $c_5$ | 0 | 0 | 0 | - | 0 | 0 | - | 0 | - | - |

**Tabla 3.2:** Ejemplo de la estrategia Uno contra uno

### *3.4.2 Uno contra uno (One versus one,OVO)*

En el caso de que no se quiera construir un clasificador binario por cada clase, se puede considerar construir un clasificador binario para cada par de clases $r$ y $s$. A este método se lo conoce como «uno contra uno». Por lo tanto, para un conjunto de $m$ clases se construyen $m(m-1)/2$ clasificadores binarios.

La tabla 3.2 muestra un ejemplo de clasificación multiclase para $m = 5$, en donde el valor de 0 corresponde a las clases que no se han tomado en cuenta para la construcción de los modelos. La predicción se realiza por votación de todos los $m(m-1)/2$ clasificadores binarios. En [AGT07], los autores realizaron una comparación entre OVA y OVO, y encontraron que no existían diferencias significativas entre ambas estrategias.

### *3.4.3 Grafo acíclico dirigido*

Esta estrategia tiene como objetivo mejorar la estrategia OVO y evitar la aplicación de todos los clasificadores al realizar las predicciones. La idea es ordenar los clasificadores como una estructura de grafo, en la cual el orden del árbol es arbitrario y es posible usar un orden alfabético o uno numérico. Primero se aplica un clasificador, se descarta una clase a la vez y se repite hasta que después de $m-1$ ejecuciones solamente quede una clase. Esta estrategia es conocida como «*Directed Acyclic Graph Support Vector Machine*»(DAGSVM) y fue presentada por [PCS99]. La principal ventaja de este método es la rapidez en la predicción de nuevos ejemplos respecto a las anteriores propuestas. La figura 3.2 muestra un ejemplo de construcción de un grafo acíclico dirigido para un problema de decisión de cinco clases.

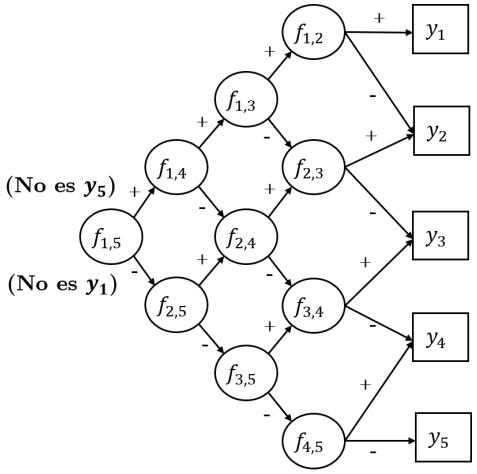



**Figura 3.2:** Ejemplo de estrategia de grafos directos acíclicos.

Una forma alternativa de descomponer un problema cuando el dominio es jerárquico es usar esta jerarquía para construir una jerarquía de clasificadores. Esta aproximación se analiza en la siguiente sección.

## 3.5 Clasificación jerárquica

La clasificación jerárquica es una metodología de clasificación que, a diferencia de otras aproximaciones, se centra en explotar las relaciones jerárquicas preestablecidas en el dominio objetivo, es decir, entre las etiquetas de clase: tomando en cuenta las relaciones, se realiza una descomposición del problema de clasificación en subproblemas de menor complejidad. En lugar de entrenar un único clasificador multiclase, se entrena un conjunto de clasificadores interconectados siguiendo la estructura jerárquica definida por la jerarquía de clases. Un componente indispensable es la existencia de una jerárquica predefinida que relaciona las posibles etiquetas de salida $\mathcal{Y}$ del problema de clasificación.

Podemos encontrar ejemplos de problemas en los que sus clases están jerárquicamente relacionadas en muchas aplicaciones reales, como: la clasificación de documentos o de sonidos, y el reconocimiento de imágenes. Si analizamos el problema de clasificar noticias, estas pueden organizarse en diferentes niveles de categorías. El periódico y sus expertos establecen una estructura jerárquica organizada a través de una taxonomía de temas. Una noticia podría pertenecer a temas como deportes, política, actualidad y entretenimiento; a su vez la

categoría "deportes" se podría subdividir en: atléticos, de pelota y de combate. Y si bajamos aún más la jerarquía, la categorías de "deportes de pelota" podrían ser: baloncesto, fútbol y tenis.

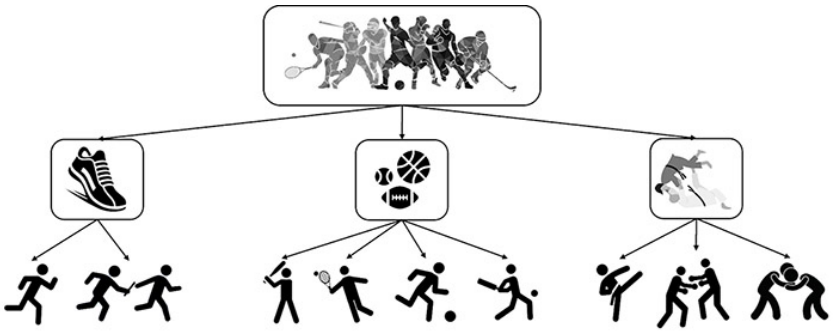


**Figura 3.3:** Ejemplo de jerarquía de clases para deportes.

La figura 3.3, muestra una jerarquía para la clase deportes. En este caso, la clase que se ubica encima del árbol (nodo raíz) es deportes. Esta clase agrupa todos los deportes existentes y de ella se desprenden tres nodos hijos, que corresponden a los deportes atléticos, de pelota y combate. Cada una de estas subcategorías agrupa los deportes individuales según estas tres categorías. Si la clase hija es «balompié» y está conectada con una clase padre «deportes de pelota» entonces se puede decir que la clase «balompié» es también de la clase «deportes de pelota».

Nótese que las categorías internas de la jerarquía de clases podrían haberse creado siguiendo criterios diferentes. Asimismo, las categorías podrían haberse definido en función del terreno en el que se realiza el deporte, por ejemplo, deportes de agua, tierra o aire. La definición de la jerarquía de clases no es un proceso trivial, ya que requiere de conocimiento del dominio para lograr que la estructura represente correctamente las relaciones entre clases del problema de clasificación.

De la misma forma que en algunas metodologías de *ensembles*, a cada clasificador se le asigna una porción del problema de clasificación y, por lo tanto, se reduce la complejidad del problema porque no se aborda el problema de clasificación completo, sino únicamente una sección específica. Para cada sección jerárquica del problema se crean clasificadores locales especializados. A diferencia de los *ensembles*, los clasificadores jerárquicos usan la jerarquía para crear clasificadores que ayuden a separar las clases en función de sus niveles

con el objetivo de que la combinación de estos modelos produzca un modelo robusto.

En [SF11], los autores dividen los clasificadores jerárquicos en: locales, globales y planos. Nosotros, en esta revisión nos centraremos únicamente en los clasificadores locales, debido a que son los más utilizados, dado que los globales no tienen una metodología estándar definida para su aplicación y los de tipo plano o *flat* simplemente transforman un problema de clasificación jerárquico en un problema multiclase simple. En esta investigación llamaremos clasificador *plano* a un clasificador que resuelve un problema de clasificación multiclase de etiqueta simple, es decir, que no es jerárquico y, por lo tanto, no utiliza una jerarquía de clase definida.

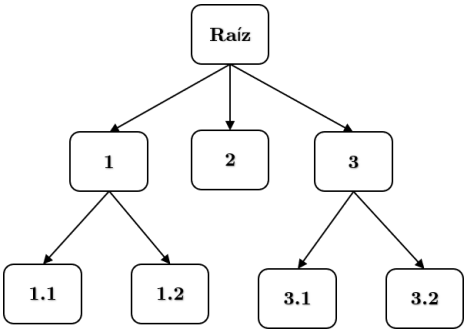
### *3.5.1 Jerarquía de clases*

La jerarquía de clases es la base de un clasificador jerárquico. Generalmente, dicha jerarquía es realizada previamente por un experto en el dominio. No obstante, hay ocasiones en las cuales la jerarquía de clases no es conocida *a priori*, por lo que se hace necesario inferirla (esta opción será revisada en el capítulo 4).

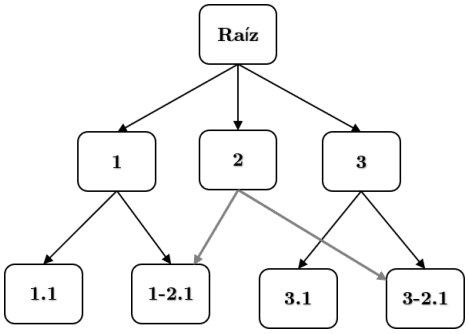
Una jerarquía de clases se puede representar con un grafo conectado, dirigido y sin ciclos (*Directed acyclic graph*, DAG), donde cada nodo representa una clase y los enlaces muestran una relación de pertenencia (la clase $A$ pertenece a la clase $B$, que conecta una clase hijo con su clase padre. Esta relación es conocida como «IS-A» ($\prec$). Si consideramos que $\mathcal{Y}$ es un conjunto finito que enumera todas las etiquetas de la clase de dominio y que $y_j, y_i, y_k$ son tres clases distintas dentro de la jerarquía, entonces una relación «IS-A» está definida con las siguientes propiedades [SF11]:

- El único elemento más alto en la jerarquía es el *nodo raíz*.
- $\forall y_i, y_j \in \mathcal{Y}$, si $y_i \prec y_j$ entonces $y_j \not\prec y_i$.
- $\forall y_i \in \mathcal{Y}, y_i \not\prec y_i$.
- $\forall y_i, y_j, y_k \in \mathcal{Y}$,$y_i \prec y_j$ y $y_j \prec y_k$ implica que $y_i \prec y_k$.

Los árboles son una clase específica de DAG en la que cada nodo solamente puede tener un único nodo padre dentro de la estructura. Los nodos que se encuentran al final de cada rama del árbol se denominan «nodos hojas», mientras que el resto de los nodos son los «nodos internos». DAG y árboles son las principales formas de representar una jerarquía de clases.




**(a)** Árbol.




**(b)** Grafo dirigido acíclico (DAG).

**Figura 3.4:** Formas de representar una jerarquía de clases.

La figura 3.4 muestra ejemplos de ambas representaciones. Los árboles y los DAG tienen muchas similitudes, como que son dirigidos, conectados, comienzan en un nodo raíz y no tienen ciclos. Sin embargo, en un DAG existen múltiples caminos posibles entre el nodo raíz y una hoja; por lo tanto, un objeto puede pertenecer a múltiples clases de un mismo nivel, como se puede observar en la figura 3.4b.

### *3.5.2 Entrenamiento de un clasificador jerárquico*

Un clasificador jerárquico consiste en un conjunto de clasificadores locales especializados que se organizan siguiendo la forma de la jerarquía que interconecta las clases. Existen varias estrategias usadas en la creación de los clasificadores tomando en cuenta la jerarquía de clases.

A continuación, mostramos diferentes tipos de estrategias para la creación de clasificadores jerárquicos.

- **Clasificador local por nodo.** Esta estrategia crea un clasificador binario para cada nodo de la jerarquía, excepto para el nodo raíz. Cada uno predice si una instancia pertenece o no a su clase correspondiente. La desventaja de este enfoque es que podrían aparecer incoherencias entre los niveles de la jerarquía, puesto que un clasificador padre podría afirmar que la instancia no pertenece a la clase, mientras que su nodo descendente señale lo contrario.

- **Clasificador local por nodo padre.** Un clasificador local por nodo padre (*Local Classifier per Parent Node*, LCPN) entrena un clasificador multiclase por cada nodo padre en la jerarquía. Los nodos padres son los nodos que están enlazados con nodos en niveles inferiores de la jerarquía; por lo tanto, todos los nodos son padres excepto los nodos hojas.

- **Clasificador local por nivel.** La estrategia de crear clasificadores locales por nivel (*Local Classifier per Level, LCL*) es la menos utilizada en la literatura y consiste en entrenar un clasificador multiclase para cada nivel de la jerarquía. La desventaja de este método es que pueden producirse inconsistencias en las predicciones realizadas entre varios niveles de la jerarquía. Si consideramos una jerarquía de clases con $n$ niveles, se crearían $\{f_1, f_2, \ldots, f_n\}$ clasificadores diferentes.

Los clasificadores jerárquicos locales se han utilizado en muchas aplicaciones reales y se han presentado estudios prácticos que ponen de relieve las mejoras del rendimiento en comparación con los clasificadores que no utilizan jerarquías en su clasificación. Una limitación en el desarrollo de este tipo de clasificadores es el tiempo adicional necesario para la construcción del modelo (etapa de entrenamiento).

**Ejemplo 3.5.1: Estrategias de creación de clasificadores jerárquicos**

*Consideremos un problema de clasificación en el cual es necesario clasificar una nueva instancia entre varias clases de deportes cuya estructura jerárquica se muestra en la figura 3.5. Esta jerarquía es una versión simplificada del problema de clasificación presentado en [Fer16]. La estructura agrupa los diferentes deportes en función de su tipo: con bicicleta, a pie o sobre el mar. En este ejemplo los deportes a pie también se subdividen en caminar y correr. Las clases en las hojas son los deportes más específicos.*

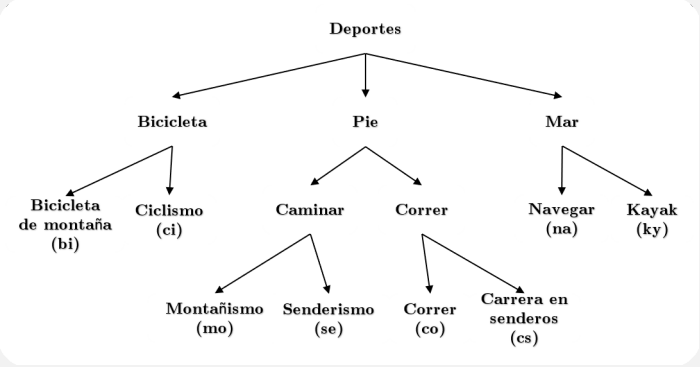
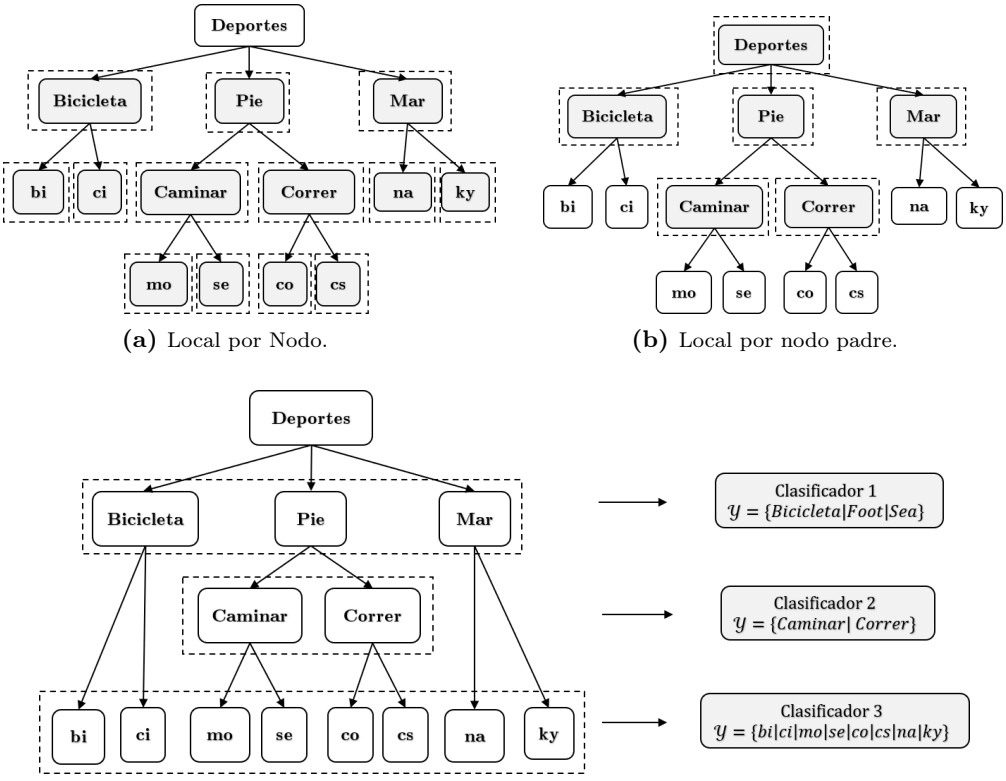



**Figura 3.5:** Ejemplo de estructura jerárquica de deportes.

Crear un modelo de clasificación jerárquico que distinga entre diferentes clases de deportes respetando la jerarquía establecida consiste en entrenar un conjunto de clasificadores locales siguiendo una de las estrategias mencionadas en esta sección.

La figura 3.6 muestra los clasificadores resultantes despues de aplicar las estrategias presentadas y cada una de ellas construye un número diferente de clasificadores. La estrategia local por nodo (figura 3.6a) construye el mayor número de clasificadores. La estrategia LCPN (figura 3.6b) reduce el número de clasificadores que hay que construir, porque no se entrenan clasificadores en las hojas del árbol. Finalmente, la estrategia que construye menos clasificadores es LPL (figura 3.6c), porque construye un clasificador por cada uno de los 3 niveles de la jerarquía.

*Fin del ejemplo 4.3.1* ■




**(a)** Local por Nodo.

**(b)** Local por nodo padre.

**(c)** Local por nivel.

**Figura 3.6:** Ejemplo de creación de clasificadores jerárquicos. Los nodos marcados con cuadrados de líneas discontinuas representan los clasificadores creados con cada estrategia.

| **Símbolos utilizados** | |
|---|---|
| $\mathfrak{T}_r$ | El conjunto de todas las instancias de entrenamiento. |
| $\mathfrak{T}_r^+(y_j)$ | El conjunto de instancias de entrenamiento positivas de $y_j$. |
| $\mathfrak{T}_r^-(y_j)$ | El conjunto de instancias de entrenamiento negativas de $y_j$. |
| $\downarrow(y_j)$ | El conjunto de las clase hijas de $y_j$. |
| $\Downarrow(y_j)$ | El conjunto de clase descendientes de $y_j$. |
| $\leftrightarrow(y_j)$ | El conjunto de clase con el mismo padre que $y_j$. |
| $*(y_j)$ | Denota los ejemplos donde la clase más específica conocida es $y_j$. |

**Tabla 3.3:** Notación para instancias de entrenamiento positivas y negativas.

El proceso de entrenamiento requiere seleccionar el conjunto de datos utilizado para el entrenamiento de cada clasificador local. Se puede definir el conjunto de ejemplos positivos y negativos usando varios criterios, que dependen del tipo de clasificador jerárquico utilizado.

A continuación, se muestran las políticas de selección de ejemplos para el entrenamiento de clasificadores, en el caso de que se utilice clasificador jerárquico por nodo padre. La notación utilizada se muestra en la tabla 3.3 y fue propuesta por [FS07].

- **Política hermana [FS07].**

$$\mathfrak{T}_r^+(y_j) = *(y_j) \cup \Downarrow (y_j) \; and \; \mathfrak{T}_r^-(y_j) = \leftrightarrow (y_j) \cup \Downarrow (\leftrightarrow (y_j)). \tag{3.1}$$

- **Política hermana exclusiva [CM07].**

$$\mathfrak{T}_r^+(y_j) = *(y_j) \; and \; \mathfrak{T}_r^-(y_j) = \leftrightarrow (y_j) \tag{3.2}$$

La política comúnmente utilizada es la «política hermana», en la cual las instancias positivas son las que pertenecen a la clase $*(y_j)$ y todos sus descendientes $\Downarrow (y_j)$. Esta se muestra como la mejor alternativa, según el estudio realizado por los autores utilizando varias bases de datos, por que se utiliza una menor cantidad de datos y obtiene el mismo valor de *accuracy* que otras políticas. Los clasificadores locales pueden ser entrenados con cualquier algoritmo de clasificación y se pueden aplicar técnicas diferentes para diferentes nodos.

De los ejemplos de clasificadores jerárquicos presentados en esta sección, se puede observar que en los niveles superiores de la jerarquía hay una menor complejidad que en los niveles inferiores. En los niveles superiores, las clases agrupan otras descendientes en superclases. El proceso de clasificación es más sencillo que realizar la clasificación directamente contra todas las etiquetas del problema. Por otro lado, mientras más nos acercamos a las hojas de la jerarquía de clasificadores, el problema de clasificación es más especializado, porque en esas posiciones las clases tienen características similares y podría producirse confusión entre ellas.

En general, el proceso utilizado para construir el modelo con clasificadores locales es mayor que la construcción de un clasificador plano único, porque el número de clasificadores que se necesita crear es igual al número de nodos internos más el nodo raíz. Además, cuando el modelo se utiliza para hacer

predicciones, tarda más tiempo, debido a las múltiples evaluaciones de los modelos locales.

Una vez entrenado el conjunto de predictores que conforman el modelo de clasificación jerárquico, es posible usarlo para realizar predicciones de instancias nuevas. Dado que el modelo está compuesto por un árbol de clasificadores, la predicción se hace mediante una estrategia de arriba a abajo o *Top-down* similar a la realizada con un árbol de decisión. El procedimiento comienza desde la parte superior del árbol aplicando el clasificador ubicado en el nodo raíz, luego se baja siguiendo la trayectoria dictada por las predicciones realizadas por los clasificadores y se continúa hasta alcanzar una hoja cuya etiqueta de clase es asignada a la instancia. Los clasificadores jerárquicos que están obligados a llegar hasta el final de la rama y solamente son capaces de asignar a cada instancia la etiqueta de clase de la hoja alcanzada se dice que son de tipo mandatorio. Por otro lado, también es posible que el problema de clasificación habilite la posibilidad de realizar una predicción en un nodo interno, que representa una clase intermedia.

## 3.6 Resumen

En este capítulo hemos revisado distintas formas de mejorar la clasificación multiclase. Así, nos enfocamos en analizar varias estrategias de *ensembles*. A continuación, revisamos las principales estrategias utilizadas para la descomposición de un problema de clasificación y, finalmente, resumimos los conceptos más importantes relacionados con la clasificación jerárquica y sus características para mejorar el rendimiento de la clasificación.

Los métodos que incluyen una descomposición del problema tienen varias ventajas interesantes, una de las cuales es la posibilidad de aplicarlos con cualquier algoritmo de clasificación básico. El proceso de entrenamiento es más sencillo debido a que los clasificadores locales son menos complejos y posibilita la creación de los clasificadores de forma distribuida en un entorno de procesamiento en paralelo, ya que es posible un entrenamiento simultáneo de múltiples clasificadores. En el siguiente capítulo, se explorará una metodología para mejorar el rendimiento de un clasificador, aplicando una estrategia automática de descomposición jerárquica del problema de clasificación multiclase.

# Capítulo 4

# Clasificación jerárquica multiclase

Construir un modelo de clasificación multiclase robusto y preciso es un problema que tiene un alto grado de dificultad, en especial cuando el número de clases es elevado. En este capítulo nos centraremos en la alternativa de dividir jerárquicamente el problema de clasificación multiclase en una colección de subproblemas más simples.

Estrategias como OVA u OVO (secciones 3.4.1 y 3.4.2), descritas en el capítulo 3, no tienen en cuenta las relaciones entre clases tal y como lo hace la clasificación jerárquica (sección 3.5). Sin embargo, al requerir la existencia de una jerarquía de clases, la clasificación jerárquica ha sido aplicada únicamente en casos donde dicha jerarquía existe.

En este capítulo analizaremos el caso en el que la jerarquía de clases no está definida por el dominio, sino que es posible obtenerla a través de un proceso de inferencia guiado por las predicciones de un clasificador plano. Esto nos permite descomponer el problema de clasificación multiclase en una jerarquía de problemas de clasificación binarios.

## 4.1 Clasificación jerárquica multiclase

El procedimiento de clasificación para un problema multiclase en el que se tiene en cuenta las relaciones entre clases se realiza a través de dos pasos:

- **Inferencia de la jerarquía de clases**: se obtiene una representación jerárquica de las relaciones entre clases utilizando los datos de entrenamiento.
- **Construcción del modelo multiclase jerárquico**: se realiza la construcción de una estructura en forma de árbol que tiene como nodos los clasificadores locales de acuerdo con la jerarquía de clases inferida.

Revisemos primero la literatura para conocer qué métodos se han propuesto para inferir una jerarquía de clases en dominios jerárquicos en los que no se dispone *a priori* de esta jerarquía.

## 4.2 Trabajos previos sobre inferencia de jerarquías de clases

La definición de la jerarquía es un paso prioritario en un problema de clasificación jerárquica. Como fue mencionado en el capítulo anterior, obtener buenas jerarquías es un proceso difícil, que implica gastar mucho tiempo y recursos. Por esta razón, la jerarquías son comúnmente definidas por un experto en el dominio. Sin embargo, existen ocasiones en las que la cantidad de datos es enorme o el número de conceptos que hay que incluir es muy alto. Por ejemplo, en biología, crear un árbol con las miles de interacciones de las proteínas es una tarea imposible de realizar manualmente. Para estos casos, la única opción es automatizar este procedimiento.

Con el fin de implementar un proceso que obtenga automáticamente una jerarquía de clases, es necesario inferir conocimiento de las clases y buscar las relaciones entre ellas. En la literatura se han presentado varias formas para obtener estas jerarquías de clases:

- Podemos generarlas a partir de los datos, simplemente aplicando un algoritmo de *clustering* que use las distancias entre instancias. En [LZO07] se presenta un método para aprender automáticamente una jerarquía para la clasificación de documentos. El enfoque consiste en aplicar una proyección lineal discriminante para transformar los documentos representados como vectores en un espacio de baja dimensión y, luego, definir la similitud entre dos clases como la distancia entre sus centroides.

La figura 4.1 muestra la jerarquía obtenida de un conjunto de datos con 20 clases de noticias. Este conjunto de datos fue creado por Ken Lang en 2008[1].

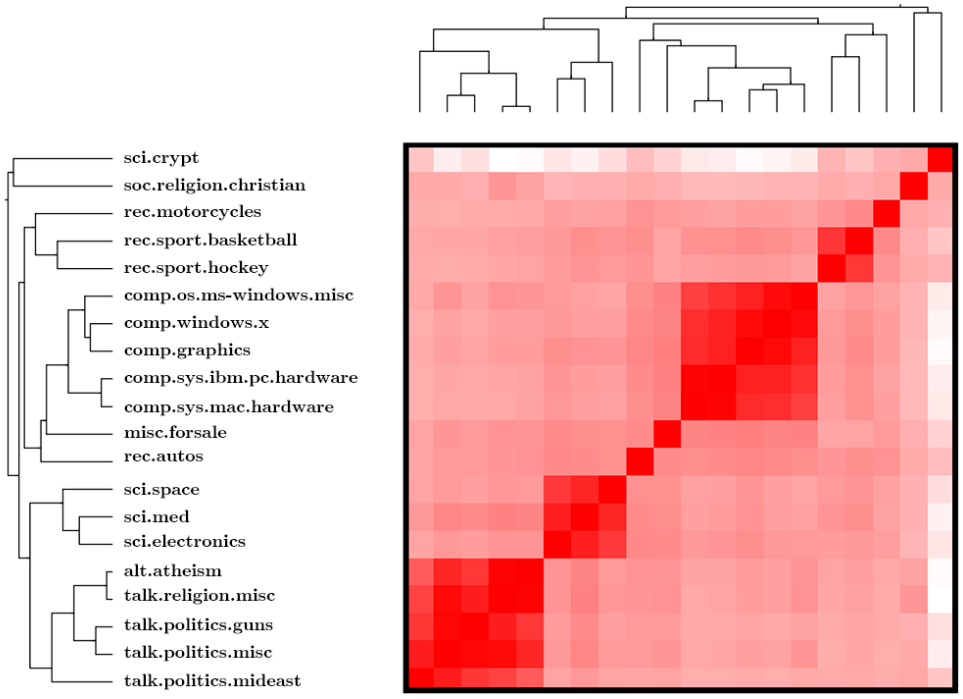



**Figura 4.1:** Ejemplo de jerarquía de clases obtenida aplicando *clustering* [LZO07].

- En [SF07] se presenta un enfoque similar para el aprendizaje de ontologías. Los autores utilizaron un sistema de recomendación en el cual, a partir de una matriz usuario-artículo que describe las calificaciones de los artículos dadas por los usuarios, establecieron una colección de ontologías utilizando varias funciones de distancia y aplicando tanto el *clustering* jerárquico aglomerativo como el divisivo (detalles sobre métodos de *clustering* se puede encontrar en la sección 2.2).

- Una forma alternativa de inferir la jerarquía de clases es determinar la similitud entre las clases utilizando las predicciones dadas por un clasificador, en muchos casos construido especialmente para este fin. En el marco de la clasificación jerárquica multietiqueta, en [CMO05], se utiliza una red neuronal ARTMAP para generar una jerarquía de relaciones

[1] http://qwone.com/ jason/20Newsgroups/

entre clases. En este caso, partimos de la clasificación multiclase de una imagen en función de los pixeles que la componen. La lista de predicciones determina una lista de reglas $r_1 \Rightarrow r_2$ que definen las relaciones entre pares de clases predichas. La creación de las reglas se realiza utilizando el algoritmo «Apriori» presentado en [AS+94]. Después, estas reglas se usan para asignar las etiquetas de clase: el antecedente $r_1$ es la clase asignada a un nivel inferior y el consecuente $r_2$ es la clase asignada al nivel superior. La figura 4.2 muestra un ejemplo de la representación gráfica obtenida de una imagen satelital de la ciudad de Monterrey.

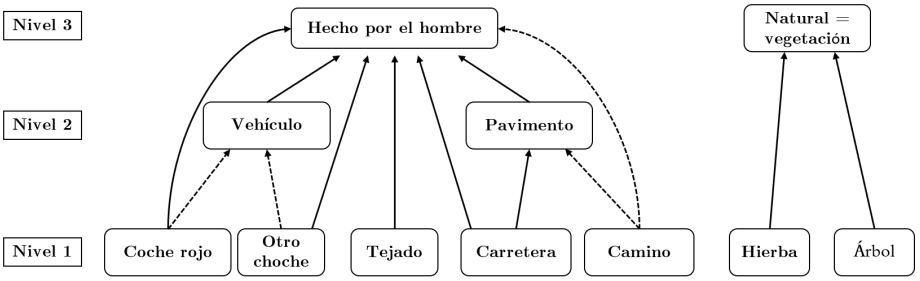



**Figura 4.2:** Ejemplo de jerarquía inferida por ARTMAP [CMO05].

- Una extensión del método anterior se presenta en [BBS11], donde, en lugar de usar una red neuronal del tipo ARTMAP, se usa una red ML-ARTMAP, la cual es una extensión multietiqueta de *Fuzzy Adaptative Resonance Associative Map* ARAM [Tan95]. Al igual que en el método anterior, se utiliza el algoritmo «Apriori» para aprender las reglas de asociación que extraen la jerarquía de clases de las predicciones dadas por el clasificador multietiqueta. Este método se diferencia del proceso anterior en que (1) en lugar de obtener un grafo dirigido acíclico obtiene un árbol y (2) no incluye un mecanismo para encontrar clases equivalentes, porque supone que todas las clases son distintas.

- En el campo de la clasificación multiclase no jerárquica, en [GSC02], se presenta un enfoque para inferir las jerarquías de clases, utilizando un conjunto de documentos de texto. Este enfoque se basa en que una matriz de confusión puede dar una idea de la similitud entre clases dentro de un «espacio de confusión». El proceso comienza entrenando un clasificador NB, el cual es evaluado a partir de su matriz de confusión, de tal manera que cada fila de la matriz es considerada como el vector de componentes que representa a una clase. A continuación, se realiza una normalización para obtener valores en un rango entre cero y uno, y se calcula la distancia

entre pares de clases utilizando una medida de distancia, como puede ser la distancia euclídea o de Kullback Leibler. Finalmente, se construye una matriz de distancias para generar un dendrograma aplicando un método de *clustering* jerárquico aglomerativo.

La propuesta que nosotros hacemos se diferencia de los primeros métodos descritos en que las relaciones entre clases se obtienen partiendo de la matriz de confusión en lugar de utilizar directamente las instancias de entrenamiento. En comparación con el método [GSC02], en el que se utilizan únicamente las filas de la matriz de confusión para calcular la distancia entre clases, nosotros proponemos utilizar filas y columnas, porque todas ellas tienen información sobre los errores de clasificación entre dos clases dadas.

## 4.3 Proceso de inferencia de la jerarquía de clases

El método que proponemos para la inferencia de jerarquías de clase se basa en extraer las relaciones entre clases partiendo de los datos de entrenamiento. Este procedimiento se puede dividir en varios pasos: primero, el proceso crea un clasificador plano con las instancias de entrenamiento, luego se utiliza el clasificador para realizar predicciones sobre las propias instancias de entrenamiento siguiendo la aproximación presentada en [God02]. A partir de la matriz de confusión que se obtiene se calcula la matriz de distancia usando una noción apropiada de similitud. Después, se realiza un *clustering* jerárquico para obtener una representación gráfica de la relación entre clases y, finalmente, derivamos una jerarquía de clases a partir del dendrograma. La figura 4.3 muestra todos los pasos del proceso de inferencia de la jerarquía de clases.

### *4.3.1 Construcción de un clasificador plano*

Para entrenar el clasificador plano se dispone de un conjunto de instancias de entrenamiento correctamente etiquetadas. Este clasificador realiza predicciones sobre los propios datos de entrenamiento para obtener una matriz de confusión $M$, la cual refleja la capacidad del clasificador para distinguir entre las diferentes clases. Así, cuanto mayor es el valor de una celda $M_{i,j}, i \neq j$, mayor es la dificultad que tiene el clasificador para distinguir las instancias de la clase $i$ de las de $j$ y, por lo tanto, más equivocaciones comete.

En general, en cualquier matriz de confusión $M$ podemos observar los siguientes hechos: (1) los errores de clasificación no suelen estar distribuidos uniformemente, lo que indica que es más difícil para el clasificador distinguir unas

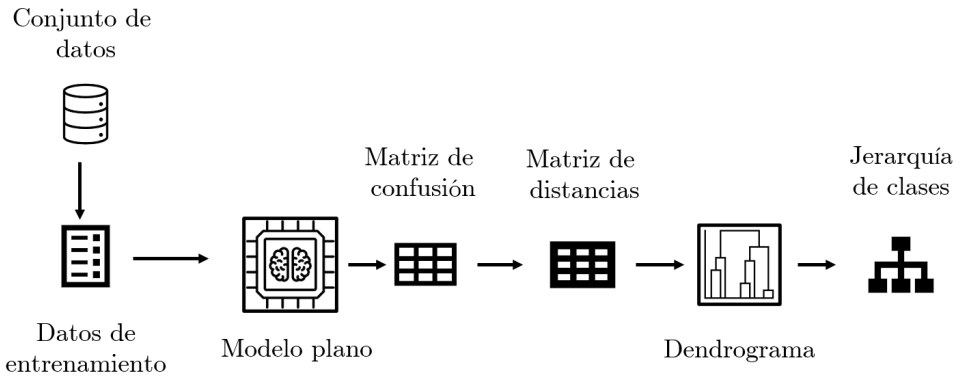



**Figura 4.3:** Proceso de inferencia de la jerarquía de clases.

clases de otras y (2) $M$ suele ser asimétrica, lo que significa que, para medir realmente el grado de confusión entre dos clases $i$ y $j$, debemos tener en cuenta todos los errores que el clasificador comete en relación con ambas clases, es decir, $M_{i,j}$ y $M_{j,i}$.

El método de inferencia que proponemos está basado en este razonamiento. Básicamente, la idea es la siguiente: cuanto mayor es la confusión entre clases, más similares son y, por lo tanto, menor será la distancia entre ellas. Por el contrario, si no existe confusión entre las clases, esto quiere decir que son clases diferentes y, por tanto, la distancia será mayor.

A continuación, definimos una nueva métrica para calcular la distancia o, alternativamente, la similitud entre las clases, usando la matriz de confusión, la cual tiene en cuenta la asimetría de esta matriz.

### *4.3.2 Cálculo de la similitud entre las clases*

Antes de hablar de la métrica propuesta, es necesario definir los conceptos de espacio métrico, métrica y semimétrica [GS91]. Un *espacio métrico* es un par $(E, d)$, donde $E$ es un conjunto no vacío (a sus elementos se les denomina «puntos») y $d$ es una función $d : E \times E \to \mathbb{R}$, llamada métrica o distancia, que posee las siguientes propiedades:

1. Separación: $d(o_1, o_2) = 0$ si y solo si $o_1 = o_2$.
2. Simetría: $\forall o_1, o_2 \in E, d(o_1, o_2) = d(o_2, o_1)$.

3. Desigualdad triangular: $\forall o_1, o_2, o_3 \in E, d(o_1, o_3) \leq d(o_1, o_2) + d(o_2, o_3)$.

4. No negatividad: $\forall o_1, o_2 \in E, d(o_1, o_2) \geq 0$.

La última propiedad se deriva de las tres anteriores.

Una semimétrica sobre $E$ es una función $d : E \times E \rightarrow \mathbb{R}$ que satisface los axiomas 1, 2 y 4, pero no necesariamente la desigualdad triangular. Como veremos, la función que definimos derivada de la matriz de confusión es una semimétrica.

Sea $M$ una matriz de confusión $m \times m$ (siendo $m$ el número de etiquetas de clase). De acuerdo con la definición de matriz de confusión, las filas de $M$ representan las instancias en las clases reales, mientras que las columnas representan las instancias en las clases predichas. Para que todos los valores de la matriz estén en el mismo rango [0.1], efectuamos una normalización por filas. Formalmente, la matriz de confusión normalizada que denotamos como $\overline{M}$, se obtiene aplicando la siguiente transformación a cada elemento $M_{ij}$:

$$\overline{M_{ij}} = \frac{M_{ij}}{\sum_{j=1}^{m} M_{ij}} \tag{4.1}$$

Como hemos mencionado anteriormente, en la matriz de confusión una clase $i$ puede ser confundida con una clase $j$ y, de la misma forma, una clase $j$ puede ser confundida con una clase $i$; por lo tanto, existen dos valores en $\overline{M}$ que representan la confusión entre las clases $i$ y $j$: $\overline{M_{ij}}$ y $\overline{M_{ji}}$. Debemos promediar ambos valores para conocer el grado de confusión entre estas clases. Llamaremos *superposición* al grado de confusión entre clases, el cual se define como:

$$superposición(i, j) = \begin{cases} \frac{\overline{M}_{i,j} + \overline{M}_{j,i}}{2} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{4.2}$$

Como la matriz está normalizada, el valor de superposición entre dos clases varía entre 0 (clases completamente distinguibles) y 1 (clases completamente indistinguibles). En particular, cada clase es indistinguible de ella misma. Dicho de otra forma, la superposición es una medida de similitud entre clases, por lo que denominamos «matriz de similitud» a la matriz resultante de aplicar la función definida por la ecuación 4.2. Nótese que la matriz de similitud es simétrica.

Finalmente, considerando que la distancia entre clases representa cuán diferentes son estas clases, este concepto de distancia es, en cierto modo, el opuesto a la similitud. Por lo tanto, podemos derivar una función de distancia a partir de la función de superposición. Dado que todos los valores están normalizados entre 0 y 1, la distancia entre dos clases $i$ y $j$, basada en superposición, se denota como $d_S$ y se define como sigue:

$$d_S(i,j) = 1 - superposición(i,j) \tag{4.3}$$

El siguiente ejemplo muestra cómo se calcula la distancia $d_S$. También sirve para ilustrar que esta distancia no cumple la condición de la desigualdad triangular, por lo que se trata de una semimétrica. Además, con el fin de realizar una comparativa, el ejemplo incluye también la derivación de una matriz de distancias basada en la distancia Euclídea.

**Ejemplo 4.3.1: Cálculo de distancias entre clases basadas en la matriz de confusión**

*Sea un problema de clasificación con 7 etiquetas de clase $\mathcal{Y} = \{a, b, c, d, e, f, g\}$ y supongamos que la matriz de confusión de un clasificador plano es la presentada en la tabla 4.1. Como mencionamos anteriormente, una matriz de confusión nos permite evaluar el rendimiento de un modelo de clasificación. En la diagonal principal se muestran los aciertos y en el resto de las celdas de la matriz, los errores de clasificación. Como se puede ver en el área central, el número de aciertos en la diagonal principal es mayor.*

| | | **Predicho** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| | **a** | **5** | 4 | 2 | 3 | 0 | 0 | 0 |
| | **b** | 2 | **15** | 8 | 11 | 7 | 2 | 0 |
| | **c** | 1 | 10 | **20** | 23 | 12 | 3 | 0 |
| **Real** | **d** | 0 | 7 | 11 | **30** | 16 | 5 | 5 |
| | **e** | 0 | 8 | 22 | 15 | **25** | 10 | 3 |
| | **f** | 0 | 0 | 2 | 9 | 10 | **10** | 5 |
| | **g** | 0 | 0 | 0 | 0 | 1 | 1 | **8** |

**Tabla 4.1:** Ejemplo de matriz de confusión. Los aciertos están destacados con negrita.

Vamos a calcular dos matrices de distancias a partir de la matriz de confusión: (1) nuestra distancia $d_S$ basada en superposición y (2) la matriz que se obtiene aplicando

la distancia Euclídea a las filas de la matriz de confusión (tal y como se propone en [GSC02]).

1. **Cómputo de** $d_S$. El primer paso es normalizar la matriz de confusión aplicando la ecuación 4.1, cuyo resultado puede verse en la tabla 4.2.

| | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
|---|---|---|---|---|---|---|---|
| **a** | 0.357 | 0.286 | 0.143 | 0.214 | 0.000 | 0.000 | 0.000 |
| **b** | 0.044 | 0.333 | 0.178 | 0.244 | 0.156 | 0.044 | 0.000 |
| **c** | 0.014 | 0.145 | 0.290 | 0.333 | 0.174 | 0.043 | 0.000 |
| **d** | 0.000 | 0.095 | 0.149 | 0.405 | 0.202 | 0.068 | 0.068 |
| **e** | 0.000 | 0.096 | 0.265 | 0.181 | 0.301 | 0.120 | 0.036 |
| **f** | 0.000 | 0.000 | 0.056 | 0.250 | 0.278 | 0.278 | 0.139 |
| **g** | 0.000 | 0.000 | 0.000 | 0.000 | 0.100 | 0.100 | 0.800 |

**Tabla 4.2:** Matriz de confusión normalizada.

A continuación, aplicamos la función de superposición (ecuación 4.2), cuyo resultado es la matriz de similitud que se muestra en la tabla 4.3.

| | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
|---|---|---|---|---|---|---|---|
| **a** | | | | | | | |
| **b** | 0.165 | | | | | | |
| **c** | 0.079 | 0.161 | | | | | |
| **d** | 0.107 | 0.170 | 0.241 | | | | |
| **e** | 0.000 | 0.126 | 0.219 | 0.198 | | | |
| **f** | 0.000 | 0.022 | 0.050 | 0.159 | 0.199 | | |
| **g** | 0.000 | 0.000 | 0.000 | 0.034 | 0.068 | 0.119 | |

**Tabla 4.3:** Matriz de similitud.

Finalmente, construimos la matriz de distancia (tabla 4.4) restando a 1 los valores de la matriz de similitud.

Se puede observar que la clase $g$ es la más distante a las clases $a$, $b$ y $c$ (distancias iguales a 1). Por otra parte, los pares de clases más cercanos son las clases $(c, d)$ y $(c, e)$, ya que tienen el valor de distancia menor entre los posibles pares de clases.

| | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
|---|---|---|---|---|---|---|---|
| **a** | | | | | | | |
| **b** | 0.83 | | | | | | |
| **c** | 0.92 | 0.84 | | | | | |
| **d** | 0.89 | 0.83 | 0.76 | | | | |
| **e** | 1.00 | 0.87 | 0.781 | 0.802 | | | |
| **f** | 1.00 | 0.98 | 0.95 | 0.84 | 0.80 | | |
| **g** | 1.00 | 1.00 | 1.00 | 0.97 | 0.93 | 0.88 | |

**Tabla 4.4:** Matriz de distancia.

Es fácil comprobar que $d_S$ satisface las condiciones de no negatividad, simetría e separación de los idénticos, pero, para algunos pares de clases $d_S$ puede no cumplir la propiedad de desigualdad triangular. Por ejemplo, para las clases $a$ y $f$, ya que $d_S(a, f) = 1 < d_S(a, d) + d_S(d, f) = 0.89 + 0.84 = 1.73$.

2. **Usando la distancia euclídea**. Una forma directa de derivar una matriz de distancia a partir de una de confusión es aplicar la distancia Euclídea ($d_E$) entre las clases, considerando que cada clase viene descrita por una fila de la matriz de confusión normalizada.

   Por ejemplo, usando la matriz de confusión del ejemplo (tabla 4.2) la distancia Euclídea entre las clases $a$ y $b$ es:

$$\begin{aligned}
\vec{a} &= \langle 0.36, 0.29, 0.14, 0.21, 0.00, 0.00, 0.00 \rangle \\
\vec{b} &= \langle 0.04, 0.33, 0.18, 0.24, 0.16, 0.04, 0.00 \rangle \\
d_E(\vec{a}, \vec{b}) &= \sqrt{(0.36 - 0.04)^2 + (0.29 - 0.33)^2 + (0.14 - 0.18)^2 + } \\
&\quad \overline{(0.21 - 0.24)^2 + (0 - 0.16)^2 + (0 - 0.04)^2 + (0 - 0)^2} \\
&= \sqrt{0.1024 + 0.0016 + 0.0016 + 0.0009 - 0.0256 + 0.0016 + 0} \\
&= 0.3656
\end{aligned}$$

   Este procedimiento se aplica de forma iterativa para cada posible par de clases, para así poder construir la matriz de distancia que se muestra en tabla 4.5. En este caso, la clase $g$ se encuentra más distante de todas las demás y los pares de clases más cercanas son $(c, d)$ y $(c, e)$, tal y como sucede al usar $d_S$. No obstante, si comparamos las dos tablas, observamos que los valores de distancia entre clases son más altos en $d_S$ que en $d_E$, lo que significa que captura mejor las diferencias entre clases. Este hecho se verá reflejado en los resultados experimentales.

| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| **a** | | | | | | | |
| **b** | 0.36 | | | | | | |
| **c** | 0.45 | 0.24 | | | | | |
| **d** | 0.51 | 0.31 | 0.19 | | | | |
| **e** | 0.53 | 0.31 | 0.22 | 0.27 | | | |
| **f** | 0.63 | 0.47 | 0.41 | 0.31 | 0.31 | | |
| **g** | 0.97 | 0.92 | 0.93 | 0.86 | 0.86 | 0.75 | |

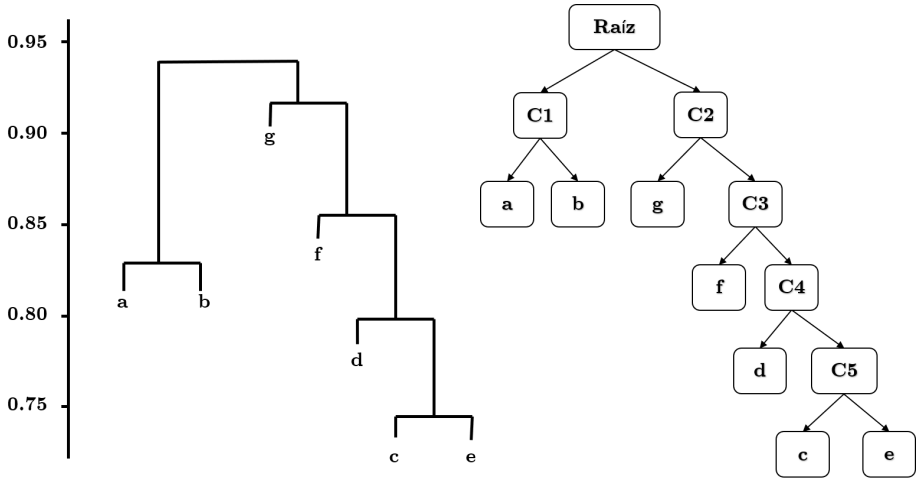
**Tabla 4.5:** Matriz de distancia calculada con la distancia euclídea.

*Fin del ejemplo 4.3.1* ■

### *4.3.3 Obtención de la jerarquía de clases*

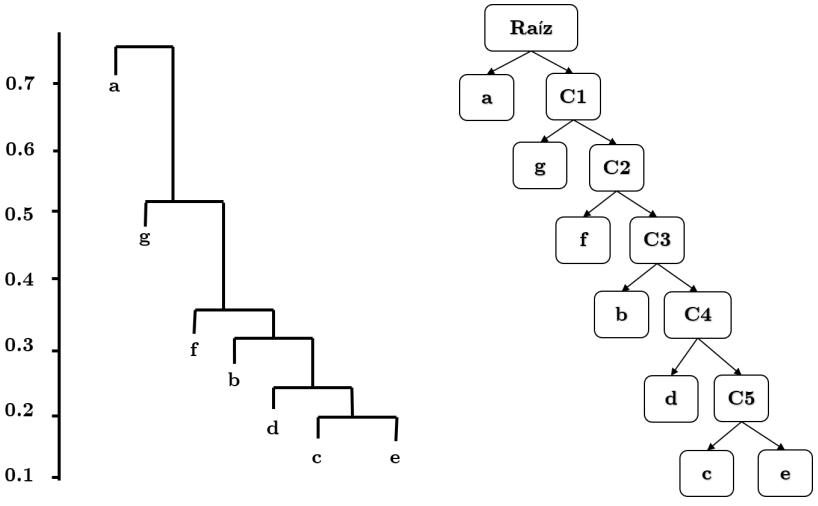
Para generar la jerarquía de clases aplicamos un algoritmo de *clustering* jerárquico aglomerativo (ver sección 2.2) usando $d_S$, lo que nos permite obtener un dendrograma.

Como el algoritmo de agrupación jerárquico aglomerativo agrega los grupos de dos en dos, el dendrograma es un árbol binario que convertimos en una jerarquía de clases, ya que las hojas del árbol corresponden a las etiquetas de clase, mientras que el resto de los nodos internos representan clases ficticias generadas automáticamente. Estas clases ficticias representan la agrupación de todas las etiquetas de clase de sus subárboles hijos. De esta forma, el nodo ubicado en la raíz representa una superclase que incluye todas las clases del problema de clasificación. La figura 4.4 representa el dendrograma y la jerarquía de clases del ejemplo 4.3.1 usando la distancia $d_S$. En esta jerarquía, las clases ficticias son $C1$, $C2$, $C3$, $C4$ y $C5$, donde por ejemplo la clase $C1$ representa el agrupamiento de las clases $a$ y $b$, y la clase $C3$ representa el agrupamiento de las clases $f$, $d$, $c$ y $e$. La figura 4.5 muestra el dendrograma y la correspondiente jerarquía de clases del ejemplo 4.3.1 usando la distancia Euclídea $d_E$. Como puede observarse, cambiando la distancia de base usada por el *clustering* jerárquico, se obtienen diferentes dendrogramas y, por lo tanto, diferentes jerarquías de clase.




**(a)** Dendrograma. **(b)** Jerarquía de clases.

**Figura 4.4:** Resultado del proceso de inferencia utilizando la semimétrica de similitud. (a) El dendrograma y (b) la jerarquía de clases.




**(a)** Dendrograma. **(b)** Jerarquía de clases.

**Figura 4.5:** Resultado del proceso de inferencia utilizando el cálculo distancia de euclídea. (a) El dendrograma y (b) la jerarquía de clases.

## 4.4 Construcción del modelo jerárquico multiclase

La segunda parte del proceso de clasificación jerárquico multiclase consiste en realizar la construcción del árbol de clasificadores tomando en cuenta la jerarquía de clases inferida.

La estrategia que proponemos para la construcción del modelo de clasificación jerárquico multiclase sigue la aproximación «clasificador local por nodo padre» (LCPN) presentado en la sección 3.5.2. Esta estrategia ha sido seleccionada porque, además de ser una de las más populares al realizar la clasificación jerárquica, no presenta las desventajas que tienen los otros métodos jerárquicos locales. Concretamente, si se aplicara la estrategia «clasificador local por nodo» (LCN) sería posible que una instancia pudiera recorrer la jerarquía de clasificadores por varios caminos, lo que traería como consecuencia que la instancia fuera asignada a varias clases al final del recorrido. Por otro lado, si se aplicara una estrategia «local por nivel» (LCL), podría suceder que una instancia fuera asignada a una clase de nivel inferior pero no a sus ancestros de niveles superiores.

El proceso de creación del modelo jerárquico se realiza recorriendo la jerarquía de clases desde la raíz hasta las hojas y entrenando un clasificador por cada nodo interno (o nodo padre) de la jerarquía. Para entrenar el clasificador correspondiente a un nodo interno de la jerarquía de clases $C_i$, se seleccionan del conjunto de entrenamiento todas las instancias pertenecientes a los niveles inferiores (descendientes de $C_i$), siguiendo la *política hermana* presentada en la sección 3.5.2. Este proceso se repite para todos los nodos padre en la jerarquía.

Una vez construida la jerarquía de clasificadores, para clasificar una nueva instancia se atraviesa la jerarquía de manera descendente aplicando los clasificadores desde la raíz hasta que se alcanza una hoja, de la misma forma que en el proceso *Top-down* utilizado y comentado en la clasificación jerárquica. Por ejemplo, consideremos un problema de clasificación con 5 etiquetas de clase: $\mathcal{Y} = \{a, b, g, f, d\}$. La Figura 4.6 muestra el árbol de clasificadores entrenado, donde los círculos representan las clases y los rectángulos, los clasificadores. Supongamos que una instancia nueva es clasificada como de clase $g$. Para ello, los clasificadores aplicados han sido $f_0$ y $f_2$ dado que la clase $g$ es descendiente de ambos.

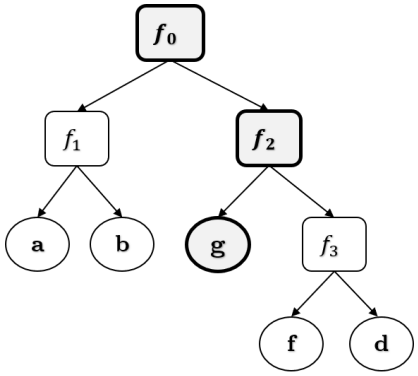



**Figura 4.6:** Ejemplo de aplicación del modelo con la estrategia *Top-down*.

Dado que la jerarquía de clases es un árbol binario con tantas hojas como etiquetas de clase, para un problema con $m$ clases tenemos que entrenar $(m-1)$ clasificadores locales. Sin embargo, la cantidad de clasificadores locales que hay que aplicar para clasificar una nueva instancia varía desde $(m/2)$, cuando la jerarquía de clases (y el árbol de clasificadores) está balanceada, hasta $(m-1)$, cuando la jerarquía de clases es una cadena (como en la figura 4.5(b)). A continuación, presentamos un procedimiento para simplificar la jerarquía de clases con el fin de reducir el número de nodos internos y, por tanto, el número de clasificadores locales que tenemos que entrenar.

## 4.5 Optimización de la jerarquía de clasificadores mediante la poda de la jerarquía de clases

Como acabamos de mencionar, una reducción del número de nodos a través de la poda de la jerarquía de clases permitirá reducir tanto su tamaño como el del modelo jerárquico y, consecuentemente, también la complejidad de este.

El método de poda propuesto recorre la jerarquía de clases en una búsqueda en profundidad, verificando que no existan dos nodos contiguos que estén a la misma distancia de enlazado; de lo contrario, los nodos se fusionarán.

En la implementación del procedimiento se ha representado el nodo de un árbol a través de un objeto de tipo *clase* llamado *Nodo* (ver código 1). Una instancia de este objeto tiene las siguientes propiedades: una etiqueta que representa la clase, la altura del nodo dentro del dendrograma, una referencia al nodo padre

y sus nodos hijos. Un árbol o jerarquía de clases está representada por un conjunto de nodos relacionados.

El algoritmo 1 muestra el método de poda propuesto. La variable de entrada *jerarquíaClases* representa la jerarquía de clases obtenida del dendrograma, la cual está formada por múltiples objetos de tipo Nodo. El proceso de poda comienza con una función *Principal* que realiza un recorrido recursivo a través del árbol, por cada nodo realiza una llamada a la función *ProcesoPoda*, que compara las alturas entre cada nodo padre e hijo posible de árbol, y, finalmente, la función *UnirNodos* contrae una rama si la altura del nodo padre y el nodo hijo son iguales. Al finalizar el proceso de poda, se obtiene la jerarquía de clases podada.

En el agrupamiento jerárquico, la idea de utilizar un proceso posterior para reducir el tamaño de la jerarquía ha sido aplicado en [DA98], donde se fija un límite de elementos que puede formar un *cluster*; por lo tanto, los puntos con un tamaño inferior al límite (por ejemplo 5 elementos) son integrados con los *clusters* de mayor dimensión.

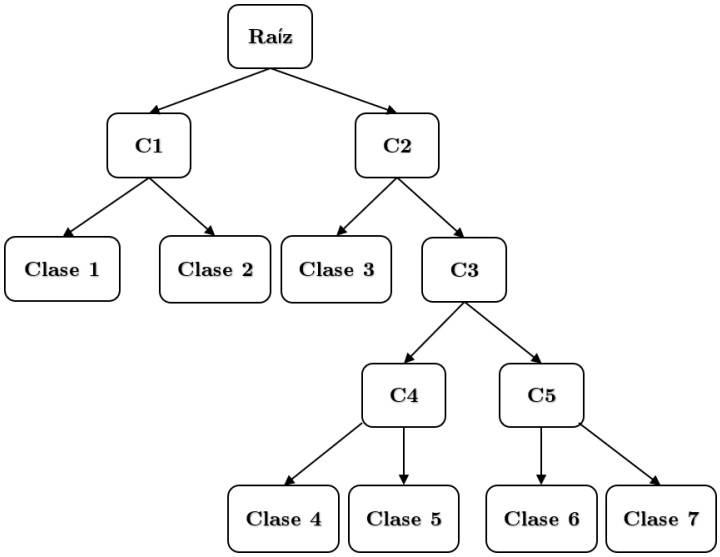
```
Clase Nodo{

//Propiedades
   Caracteres etiqueta;
   Entero altura;
   Nodo nodoPadre;
   Lista<Nodo> nodosHijos = {};

//Métodos
   Nodo obtenerPadre(){
      retorna nodoPadre;
   }

   Lista<Nodo> obtenerHijos(){
      retorna nodosHijos;}

   Entero obtenerAltura(){
      retorna altura;
   }

   void quitarHijo(Nodo nodoIn){
      nodosHijos.remove(nodoIn);
   }

   void agregarNodoHijo(Nodo nodoIn){
      nodosHijos.add(nodoIn);
   }
   ...
}
```

**Código 1**. Definición de la clase nodo.

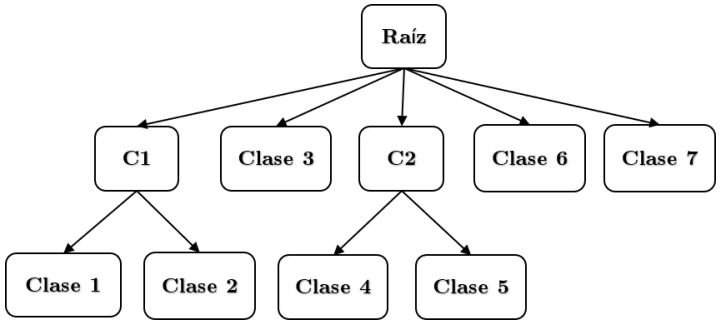
**Algoritmo 1:** Algoritmo de poda

**Entrada:** Nodo jerarquíaClases
**Salida:** Nodo jerarquíaPodada

```
Función Principal (Nodo jerarquiaClases) es
    subArbolPodado = ProcesoPoda(jerarquíaClases);
    hijos = subArbolPodado.obtenerHijos();
    para cada hijo en hijos hacer
        hijo = Principal(hijo);
    return subArbolPodado;

Función ProcesoPoda(Nodo arbolIn) es
    Nodo subarbolPodado;
    hijos = arbolIn.obtenerHijos();
    para cada hijo en hijos hacer
        nodoAltura = arbolIn.obtenerAltura();
        hijoAltura = hijo.obtenerAltura();
        si (nodoAltura == hijoAltura) entonces
            subarbolPodado = UnirNodos(arbolIn,hijo);
    return subarbolPodado;

Función UnirNodos(Nodo nodoPadre,Nodo nodoHijo) es
    nodoPadre.quitarHijo(nodoHijo);
    decendientes = obtenerHijos(nodoHijo);
    para cada descendiente en decendientes hacer
        nodoPadre.agregarNodoHijo(descendiente);
    return nodPoadre;
```




**(a)** Jerarquía de clase original.




**(b)** Jerarquía de clase luego del proceso de poda.

**Figura 4.7:** Ejemplo del proceso de poda de nodos de la jerarquía aplicado al conjunto de datos Zoo del repositorio UCI [Aha87]: (a) la jerarquía de clases dada por el dendrograma y (b) la jerarquía de clases después de aplicar la poda.

La figura 4.7 muestra la jerarquía original y la resultante del proceso de poda aplicado al conjunto de datos Zoo del repositorio UCI [Aha87]. Este conjunto de datos fue creado por Richard Forsyth [For90] y contiene instancias de entrenamiento de un conjunto de 7 diferentes tipos de animales. Como puede verse los nodos *Raíz*, $C2$, $C3$ y $C5$ de la figura 4.7a se han fusionado en el nodo raíz de la jerarquía (figura 4.7b). Como consecuencia, el número de

clasificadores locales que deben aprenderse se reduce de seis a tres. También se puede observar que el criterio de fusión de los nodos que están a la misma distancia de enlazado se podría modificar fácilmente para permitir la fusión de otros nodos cuya distancia de enlazado no supere un determinado umbral, es decir, los nodos cuya distancia de enlazado sea inferior a un mínimo fijado podrían fusionarse en un único nodo. Exploraremos esta idea en la siguiente sección.

## 4.6 Estrategia de poda de nodos basada en un umbral

En esta sección exploramos una variante del método de poda basada en un umbral específico definido sobre la distancia de enlazado. Formalmente, dados dos *clusters* (nodos del dendrograma) contiguos[2], $i$ e $(i+1)$, si sus distancias de enlazado no exceden un cierto umbral $\theta$, es decir, $\mid linkage(i) - linkage(i+1) \mid < \theta$, entonces los *clusters* se fusionan, donde $linkage(A)$ es la distancia de enlazado a la que se ha creado el *cluster* $A$. El algoritmo 2 presenta este procedimiento de poda.

El umbral $\theta$ puede ser elegido por el usuario, depender del dominio o ser estimado. Proponemos determinar el umbral a partir de las distancias de enlazado usadas por el algoritmo de *clustering* jerárquico para realizar la agrupación. Más específicamente, para un dendrograma dado que consiste en agrupaciones de $n$ *clusters*, el umbral $\theta$ se determina como:

$$\theta = \alpha \cdot maximum_{k \in [1..(n-1)]} \mid linkage(k) - linkage(k+1) \mid \qquad (4.4)$$

donde $\alpha \in [0..1]$ es el parámetro que indica el nivel de poda. Un valor $\alpha = 0$ significa que solo los *clusters* a la misma distancia de enlace se unirán en el proceso de poda, mientras que $\alpha = 1$ es el valor máximo posible que colapsa todos los *clusters* en la raíz del dendrograma. De hecho, dependiendo de la forma del dendrograma, la jerarquía de clases puede colapsar en la raíz a un valor de $\alpha < 1$.

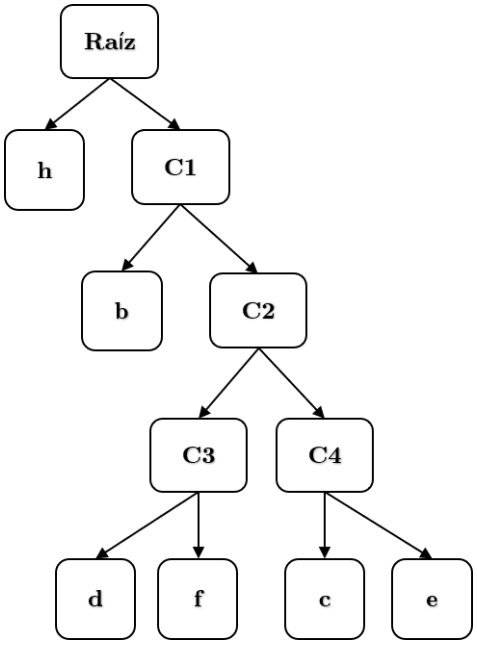
La figura 4.8 muestra el proceso de poda basado en umbral aplicado al conjunto de datos *Flare*, tomado del repositorio UCI [Aha87]. Este conjunto de datos es usado para predecir uno de siete tipos de llamaradas solares. Cada atributo cuenta el número de llamaradas solares de una cierta clase que ocurren en un período de 24 horas. Como puede observarse, para $\alpha = 0.1$ (figura 4.8b), los

[2]En aras de la simplicidad, usamos el orden de creación de los *clusters* para denotarlos.

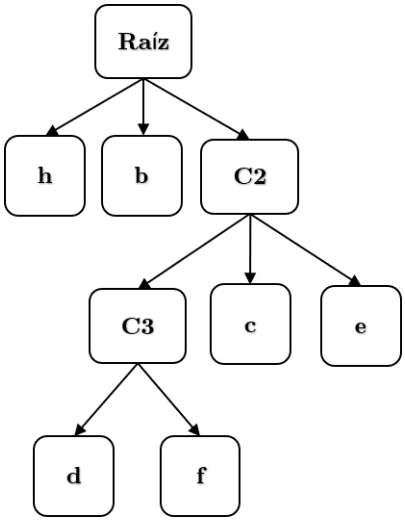
**Algoritmo 2:** Poda de la jerarquía de clases con umbral.

**Entrada:** Nodo jerarquíaClases
**Salida:** Nodo jerarquíaPodada

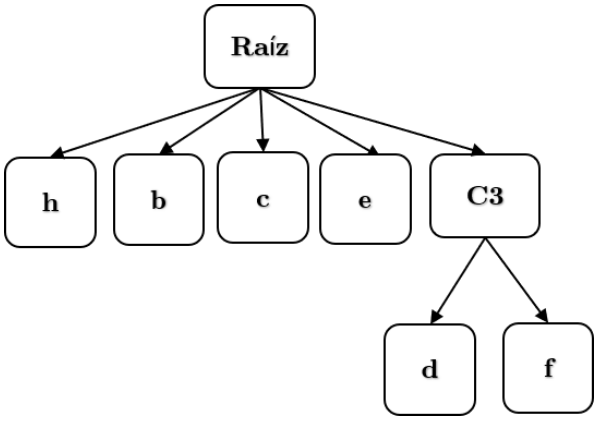
```
Función Principal (Nodo jerarquiaClases,double umbral) es
    subArbolPodado = ProcesoPoda(jerarquíaClases,umbral);
    hijos = subArbolPodado.obtenerHijos();
    para cada hijo en hijos hacer
        hijo = Principal(hijo,umbral);
    return subArbolPodado;

Función ProcesoPoda(Nodo arbolIn, double umbral) es
    Nodo subarbolPodado;
    hijos = arbolIn.obtenerHijos();
    para cada hijo en hijos hacer
        nodoAltura = arbolIn.obtenerAltura();
        hijoAltura = hijo.obtenerAltura();
        distancia = nodoAltura-hijoAltura;
        si (distancia<=umbral) entonces
            subarbolPodado = UnirNodos(arbolIn,hijo);
    return subarbolPodado;

Función UnirNodos(Nodo nodoPadre,Nodo nodoHijo) es
    nodoPadre.quitarHijo(nodoHijo);
    decendientes = obtenerHijos(nodoHijo);
    para cada descendiente en decendientes hacer
        nodoPadre.agregarNodoHijo(descendiente);
    return nodoPadre;
```




**(a)** Jerarquía de clase original.




**(b)** Jerarquía comprimida con $\alpha = 0.1$.




**(c)** Jerarquía comprimida con $\alpha = 0.4$.

**Figura 4.8:** Ejemplo del proceso de poda con umbral de la jerarquía aplicado al conjunto de datos *Flare* con diferentes valores del parámetro de poda $\alpha$.

nodos raíz y $C1$ de la jerarquía original (figura 4.8a) se fusionan en un nodo, que es la raíz de la jerarquía resultante. De la misma manera, los nodos $C2$ y $C4$ (figura 4.8a) se fusionan en un nuevo nodo $C2$ (figura 4.8b). Asimismo, para $\alpha = 0.4$ (figura 4.8c), los nodos raíz, $C1$, $C2$, y $C3$ de la jerarquía original se fusionan en el nodo raíz. Como consecuencia, el número de clasificadores planos que hay que aprender se ha reducido de cinco a tres para $\alpha = 0.1$ y de cinco a dos para $\alpha = 0.4$.

## 4.7 Evaluación de los métodos jerárquicos multiclase

En esta sección evaluamos experimentalmente el desempeño de nuestra propuesta para resolver problemas multiclase. Los experimentos se llevaron a cabo sobre 20 conjuntos de datos diferentes (tabla 4.6) extraídos de los repositorios públicos UCI [Aha87] y LIBSVM2 [Fan11]. Todos los conjuntos de datos son, *a priori*, multivariantes, multiclase y no jerárquicos.

El criterio que seguimos para seleccionarlos fue incluir conjuntos de datos de diferente tamaño y número de clases. Hemos preprocesado los conjuntos de datos eliminando las instancias con valores faltantes y columnas innecesarias o que tengan un mismo valor para todas las instancias del conjunto de datos, además aplicamos un submuestreo a los conjuntos de datos 10 y 15 (para hacer frente al desequilibrio de clases) y un muestreo estratificado a los conjuntos de datos 19 y 20 (para reducir su tamaño). Una descripción detallada del preprocesado de cada conjunto de datos se puede consultar en el apéndice A.

Para analizar la idoneidad de las jerarquías generadas por nuestro método, incluimos, además de la semimétrica $d_S$ que hemos definido, dos métodos existentes para inferir jerarquías de clases. En primer lugar, el método basado en instancias propuesto en [LZO07], que calcula la distancia entre los centroides de cada clase y que denominamos $d_C$. El segundo método calcula las similitudes aplicando la distancia euclídea $d_E$ sobre la matriz de confusión normalizada y generada por el clasificador plano, tal y como en [GSC02]. Utilizamos la distancia de enlace completa (Ecuación 2.2) en el algoritmo de *clustering* jerárquico aglomerativo aplicado para inferir las jerarquías de clases (utilizando $d_S$ y $d_E$). La razón de esta elección es que la distancia de enlace simple produce un dendrograma con un mayor número de encadenamientos, mientras que la distancia de enlace promedio requiere un mayor número de operaciones para su cálculo.

| Id | Conjunto de datos | Número | | | (Min , Max) |
|---|---|---|---|---|---|
| | | **Inst.** | **Attr.** | **Clases** | |
| 1 | Arrhytmia | 416 | 330 | 7 | (4% , 59%) |
| 2 | Covtype | 2100 | 54 | 7 | (14% , 14%) |
| 3 | Dermatology | 358 | 34 | 6 | (6% , 31%) |
| 4 | Flare | 1066 | 19 | 6 | (4% , 31%) |
| 5 | Forest | 523 | 27 | 4 | (16% , 30%) |
| 6 | Glass | 214 | 9 | 6 | (4% , 33%) |
| 7 | Letters | 2600 | 16 | 26 | (4% , 4%) |
| 8 | Nuclear | 552 | 80 | 8 | (8% , 16%) |
| 9 | Pendigits | 7494 | 16 | 10 | (10%, 10%) |
| 10 | Reuters | 1000 | 58 | 10 | (10% , 10%) |
| 11 | Satimage | 1795 | 36 | 6 | (9% , 25%) |
| 12 | Segmentation | 2310 | 18 | 7 | (14% , 14%) |
| 13 | Sports | 8000 | 13 | 10 | (10% , 10%) |
| 14 | TrafficLight | 300 | 10 | 6 | (15% , 19% |
| 15 | Usps | 3000 | 256 | 10 | (10% , 10%) |
| 16 | Vertebral | 310 | 6 | 4 | (19% , 48%) |
| 17 | Yeast | 1484 | 8 | 10 | (0.3% , 31%) |
| 18 | Zoo | 101 | 16 | 7 | (4% , 20%) |
| 19 | 10News | 3700 | 45 | 10 | (10% , 10%) |
| 20 | 20News | 6660 | 48 | 20 | (5% , 5%) |

**Tabla 4.6:** Información sobre el conjunto de datos utilizados en los experimentos: número de instancias, atributos y clases. La última columna muestra los porcentajes de instancias que son de la clase minoritaria y mayoritaria.

Además, para analizar si la clasificación multiclase puede mejorarse mediante el uso de jerarquías de clases, comparamos los resultados obtenidos mediante las jerarquías con los obtenidos por el clasificador plano, entrenado en primer lugar para inducir la jerarquía.

En los experimentos aplicamos once técnicas de clasificación, hemos utilizado la misma técnica para la creación del clasificador plano y los clasificadores locales que forman el modelo jerárquico. Las implementaciones se encuentran en las librerías *Caret* [Kuh+14], rpart, e1071 y C50 del lenguaje R [R C15]. En concreto, utilizamos los siguientes algoritmos de clasificación: dos árboles de decisión (J48 y C50), un aprendiz de reglas proposicionales (JRIP), Naive Bayes (NB), vecinos más próximos (KNN), una lista de decisión (PART), un árbol de partición recursivo (RPART), una red neuronal (NNET), el ensemble

Random Forest (RF), el método de análisis discriminante flexible (FDA) y una máquina de vectores de soporte (SVM) con kernel radial. Se han usado hiperparámetros fijos para cada modelo y todos los datasets. En el apéndice B se puede encontrar información adicional sobre los parámetros utilizados en cada una de las técnicas de clasificación.

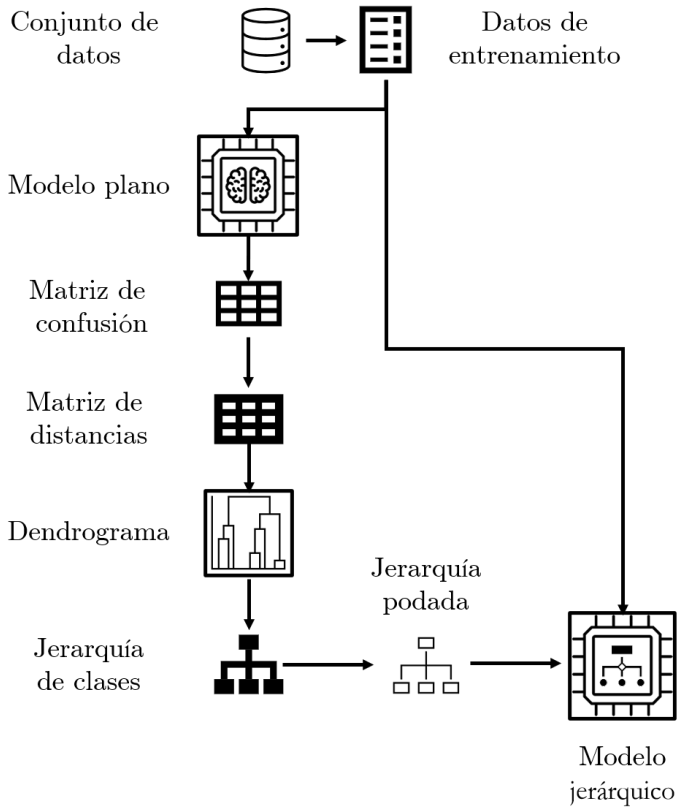



**Figura 4.9:** Esquema de descomposición jerárquica de los problemas multiclase guiado por la inferencia de la jerarquía de clases.

Los experimentos que realizamos siguen el esquema que se muestra en la figura 4.9. Primero se construye un clasificador plano y se infiere la jerarquía de clases como se explica en la sección 4.3. A continuación, se poda la jerarquía de clases para reducir el número de clasificadores planos que debemos aprender y, finalmente, se genera el árbol de clasificadores según la jerarquía de clases podada.

| | | Conjuntos de datos | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **J48** | **F** | 0.714 | 0.695 | 0.947 | 0.732 | 0.849 | 0.720 | 0.736 | 1.000 | 0.962 | - |
| | **E** | **0.716** | 0.708 | 0.947 | 0.738 | 0.874 | 0.685 | **0.720** | 0.996 | 0.956 | - |
| | **S** | 0.709 | 0.707 | 0.949 | **0.744** | 0.866 | 0.714 | **0.720** | **1.000** | **0.961** | - |
| | **C** | 0.680 | **0.711** | **0.955** | 0.728 | **0.877** | **0.739** | 0.703 | 0.989 | 0.954 | - |
| **JRIP** | **F** | 0.758 | 0.649 | 0.947 | 0.686 | 0.847 | 0.676 | 0.687 | 0.899 | 0.952 | - |
| | **E** | 0.730 | 0.686 | 0.961 | **0.742** | 0.877 | **0.704** | 0.708 | 0.955 | 0.956 | - |
| | **S** | **0.738** | **0.692** | 0.955 | 0.732 | **0.881** | 0.684 | 0.715 | 0.960 | 0.959 | - |
| | **C** | **0.738** | 0.676 | **0.964** | 0.710 | 0.871 | 0.671 | **0.744** | **0.975** | **0.960** | - |
| **KNN** | **F** | 0.637 | 0.643 | 0.875 | 0.720 | 0.893 | 0.669 | 0.800 | 0.995 | 0.993 | 0.507 |
| | **E** | **0.654** | **0.661** | 0.855 | 0.730 | 0.894 | 0.621 | 0.766 | **0.996** | **0.993** | 0.450 |
| | **S** | **0.654** | 0.658 | **0.858** | **0.733** | 0.891 | 0.612 | **0.779** | **0.996** | **0.993** | **0.471** |
| | **C** | **0.654** | 0.649 | **0.858** | 0.707 | **0.896** | **0.622** | 0.721 | **0.996** | **0.993** | 0.444 |
| **NB** | **F** | 0.589 | 0.218 | 0.916 | 0.593 | 0.870 | 0.571 | 0.674 | 0.975 | 0.883 | 0.213 |
| | **E** | **0.589** | 0.165 | **0.802** | 0.548 | 0.852 | 0.559 | 0.581 | 0.886 | 0.805 | 0.186 |
| | **S** | **0.589** | 0.146 | 0.771 | 0.551 | 0.852 | 0.583 | 0.535 | **0.982** | 0.793 | 0.197 |
| | **C** | **0.589** | **0.240** | 0.785 | **0.590** | **0.860** | **0.599** | **0.707** | 0.879 | **0.864** | **0.223** |
| **NNET** | **F** | 0.587 | 0.283 | 0.972 | 0.761 | 0.876 | 0.703 | 0.362 | 1.000 | 0.463 | 0.537 |
| | **E** | 0.644 | **0.508** | 0.958 | **0.759** | 0.898 | 0.684 | **0.770** | **1.000** | 0.920 | 0.569 |
| | **S** | 0.661 | 0.503 | 0.964 | **0.759** | **0.917** | **0.703** | 0.763 | **1.000** | 0.913 | **0.578** |
| | **C** | **0.685** | 0.454 | **0.966** | 0.751 | 0.906 | 0.694 | 0.720 | **1.000** | **0.921** | 0.559 |
| **PART** | **F** | 0.710 | 0.707 | 0.939 | 0.723 | 0.845 | 0.711 | 0.724 | 1.000 | 0.970 | - |
| | **E** | 0.678 | 0.697 | 0.950 | 0.726 | 0.861 | 0.678 | 0.727 | 0.996 | 0.960 | - |
| | **S** | 0.664 | **0.704** | **0.955** | 0.733 | **0.870** | 0.693 | 0.708 | **1.000** | **0.966** | - |
| | **C** | **0.709** | 0.695 | 0.944 | **0.741** | 0.866 | **0.711** | **0.730** | 0.998 | 0.960 | - |
| **RPART** | **F** | 0.738 | 0.625 | 0.941 | 0.739 | 0.868 | 0.700 | 0.463 | 1,000 | 0.836 | 0.505 |
| | **E** | 0.764 | 0.666 | 0.924 | 0.729 | **0.868** | **0.708** | 0.645 | 0.984 | 0.901 | 0.492 |
| | **S** | **0.769** | 0.667 | 0.927 | **0.736** | 0.866 | 0.680 | 0.636 | **1.000** | **0.906** | **0.508** |
| | **C** | 0.757 | **0.669** | **0.941** | 0.723 | 0.858 | 0.695 | **0.646** | 0.962 | 0.882 | 0.500 |
| **RF** | **F** | 0.800 | 0.772 | 0.978 | 0.738 | 0.883 | 0.818 | 0.875 | 1.000 | 0.991 | 0.635 |
| | **E** | **0.805** | **0.775** | 0.978 | 0.750 | **0.891** | 0.816 | 0.849 | 0.998 | **0.991** | 0.557 |
| | **S** | 0.800 | 0.774 | **0.983** | **0.751** | 0.889 | **0.831** | **0.880** | **1.000** | **0.991** | 0.572 |
| | **C** | 0.793 | 0.769 | 0.980 | 0.735 | 0.889 | 0.797 | 0.822 | **1.000** | 0.988 | **0.578** |
| **SVM** | **F** | 0.589 | 0.684 | 0.970 | 0.756 | 0.895 | 0.714 | 0.834 | 0.998 | 0.995 | 0.569 |
| | **E** | **0.589** | 0.650 | 0.937 | **0.749** | **0.898** | 0.659 | 0.753 | **0.998** | 0.994 | 0.477 |
| | **S** | **0.589** | 0.690 | **0.945** | 0.744 | **0.898** | 0.666 | 0.803 | **0.998** | **0.996** | **0.503** |
| | **C** | **0.589** | **0.702** | 0.925 | 0.735 | 0.896 | **0.680** | **0.819** | 0.996 | 0.994 | 0.460 |
| **C50** | **F** | 0.719 | 0.708 | 0.941 | 0.728 | 0.855 | 0.713 | 0.725 | 1.000 | 0.968 | 0.561 |
| | **E** | **0.713** | 0.709 | 0.944 | 0.736 | 0.864 | 0.694 | 0.697 | 0.995 | 0.955 | 0.519 |
| | **S** | 0.709 | **0.713** | 0.947 | **0.741** | 0.864 | **0.731** | **0.742** | **1.000** | **0.959** | 0.515 |
| | **C** | 0.706 | 0.712 | **0.955** | 0.734 | **0.877** | 0.700 | 0.718 | 0.996 | 0.956 | **0.530** |
| **FDA** | **F** | 0.755 | 0.684 | 0.953 | 0.755 | 0.896 | 0.691 | 0.642 | 1.000 | 0.874 | 0.495 |
| | **E** | **0.774** | 0.699 | 0.947 | 0.749 | 0.877 | 0.684 | 0.680 | 0.998 | 0.939 | 0.495 |
| | **S** | 0.772 | **0.703** | 0.944 | 0.754 | 0.881 | 0.694 | **0.737** | **1.000** | **0.955** | **0.505** |
| | **C** | 0.757 | **0.703** | **0.958** | **0.758** | **0.889** | **0.727** | **0.737** | 0.998 | 0.927 | 0.484 |

**Tabla 4.7:** (a) Valores de *accuracy* obtenidos por las diferentes técnicas de clasificación, utilizando los conjuntos de datos del 1 al 10 de la tabla 4.6. Para cada técnica de clasificación y conjunto de datos el mejor resultado está subrayado y, además, el mejor método que utiliza una jerarquía está resaltado en negrita.

| | | Conjunto de Datos | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **Promedio** |
| **J48** | **F** | 0.826 | 0.871 | 0.624 | 0.740 | 0.844 | 0.516 | 0.605 | 0.925 | - | - | 0.783 |
| | **E** | 0.811 | **0.957** | 0.620 | **0.747** | 0.848 | **0.581** | 0.573 | **0.943** | - | - | 0.789 |
| | **S** | **0.817** | 0.956 | **0.628** | 0.724 | 0.852 | 0.563 | 0.581 | 0.933 | - | - | **0.790** |
| | **C** | 0.816 | **0.957** | 0.620 | 0.711 | **0.861** | 0.560 | **0.587** | 0.929 | - | - | 0.787 |
| **JRIP** | **F** | 0.827 | 0.838 | 0.569 | 0.720 | 0.842 | 0.577 | 0.587 | 0.869 | - | - | 0.761 |
| | **E** | **0.839** | **0.957** | **0.615** | 0.713 | 0.857 | **0.568** | **0.587** | 0.902 | - | - | 0.786 |
| | **S** | 0.826 | **0.957** | 0.608 | 0.743 | 0.863 | 0.561 | 0.574 | **0.931** | - | - | **0.787** |
| | **C** | 0.820 | 0.950 | 0.608 | **0.760** | **0.866** | 0.540 | 0.577 | **0.931** | - | - | 0.786 |
| **KNN** | **F** | 0.871 | 0.790 | 0.524 | 0.325 | 0.954 | 0.394 | 0.588 | 0.861 | 0.246 | 0.172 | 0.673 |
| | **E** | **0.875** | 0.935 | **0.523** | 0.347 | 0.954 | 0.406 | 0.579 | 0.812 | 0.214 | 0.137 | 0.670 |
| | **S** | 0.874 | 0.935 | 0.522 | 0.334 | **0.955** | **0.418** | 0.578 | **0.824** | **0.241** | **0.166** | **0.675** |
| | **C** | **0.875** | **0.938** | 0.520 | **0.361** | 0.954 | 0.400 | **0.581** | 0.812 | 0.228 | 0.150 | 0.668 |
| **NB** | **F** | 0.822 | 0.824 | 0.571 | 0.543 | 0.738 | 0.544 | 0.424 | 0.882 | 0.185 | 0.143 | 0.609 |
| | **E** | 0.750 | 0.871 | 0.511 | 0.452 | 0.632 | 0.527 | 0.350 | 0.599 | **0.171** | **0.108** | 0.547 |
| | **S** | 0.749 | 0.871 | 0.520 | 0.371 | 0.684 | **0.532** | 0.344 | 0.645 | 0.142 | 0.078 | 0.547 |
| | **C** | **0.795** | **0.878** | **0.538** | **0.482** | **0.694** | 0.531 | **0.362** | **0.796** | 0.157 | 0.101 | **0.584** |
| **NNET** | **F** | 0.843 | 0.605 | 0.229 | 0.659 | 0.672 | 0.521 | 0.596 | 0.941 | 0.264 | 0.163 | 0.602 |
| | **E** | **0.876** | **0.937** | 0.491 | 0.634 | **0.935** | 0.532 | 0.569 | 0.953 | 0.264 | 0.181 | 0.704 |
| | **S** | 0.870 | 0.933 | **0.517** | **0.707** | 0.931 | **0.566** | 0.579 | **0.963** | **0.281** | **0.192** | **0.715** |
| | **C** | 0.870 | 0.913 | 0.508 | 0.560 | 0.931 | 0.558 | **0.587** | 0.954 | 0.273 | 0.176 | 0.699 |
| **PART** | **F** | 0.824 | 0.881 | 0.618 | 0.753 | 0.878 | 0.526 | 0.555 | 0.925 | - | - | 0.782 |
| | **E** | 0.812 | 0.951 | **0.592** | 0.694 | 0.868 | 0.571 | 0.555 | **0.953** | - | - | 0.781 |
| | **S** | **0.824** | 0.954 | 0.588 | **0.700** | 0.861 | **0.579** | **0.571** | 0.943 | - | - | 0.783 |
| | **C** | 0.819 | **0.962** | 0.590 | 0.693 | **0.873** | **0.579** | 0.551 | 0.911 | - | - | **0.784** |
| **RPART** | **F** | 0.793 | 0.917 | 0.559 | 0.746 | 0.760 | 0.492 | 0.569 | 0.873 | 0.194 | 0.103 | 0.671 |
| | **E** | **0.803** | **0.954** | 0.589 | 0.707 | 0.814 | 0.427 | 0.579 | **0.866** | 0.228 | 0.131 | 0.689 |
| | **S** | 0.798 | 0.948 | 0.584 | 0.724 | 0.830 | **0.450** | 0.583 | **0.866** | **0.229** | **0.141** | **0.692** |
| | **C** | 0.799 | 0.932 | **0.596** | **0.736** | **0.840** | **0.450** | **0.584** | 0.832 | 0.193 | 0.123 | 0.686 |
| **RF** | **F** | 0.882 | 0.974 | 0.710 | 0.807 | 0.943 | 0.339 | 0.627 | 0.973 | 0.338 | 0.259 | 0.767 |
| | **E** | 0.876 | **0.976** | **0.708** | **0.807** | 0.900 | 0.329 | 0.604 | **0.982** | 0.265 | 0.163 | 0.751 |
| | **S** | 0.875 | 0.975 | 0.707 | 0.803 | **0.938** | **0.339** | 0.608 | 0.973 | 0.260 | 0.164 | **0.756** |
| | **C** | **0.877** | 0.972 | 0.700 | 0.790 | 0.926 | 0.335 | **0.616** | 0.981 | **0.289** | **0.193** | 0.752 |
| **SVM** | **F** | 0.870 | 0.939 | 0.626 | 0.571 | 0.969 | 0.473 | 0.596 | 0.951 | 0.327 | 0.230 | 0.728 |
| | **E** | 0.872 | **0.958** | 0.607 | 0.670 | 0.956 | 0.460 | **0.584** | **0.972** | 0.151 | 0.089 | 0.701 |
| | **S** | **0.873** | **0.958** | **0.616** | **0.677** | 0.957 | **0.477** | 0.569 | **0.972** | **0.272** | **0.185** | **0.719** |
| | **C** | 0.871 | 0.934 | 0.612 | 0.571 | **0.965** | **0.477** | 0.555 | 0.962 | 0.238 | 0.157 | 0.707 |
| **C50** | **F** | 0.822 | 0.963 | 0.622 | 0.769 | 0.851 | 0.510 | 0.560 | 0.943 | 0.213 | 0.133 | 0.715 |
| | **E** | 0.815 | 0.960 | 0.620 | 0.746 | 0.851 | **0.598** | 0.562 | **0.943** | 0.200 | 0.116 | 0.712 |
| | **S** | **0.822** | 0.954 | **0.625** | 0.769 | 0.817 | 0.596 | 0.577 | 0.925 | 0.221 | **0.148** | **0.719** |
| | **C** | 0.820 | **0.961** | 0.620 | **0.786** | **0.862** | 0.571 | **0.592** | 0.909 | **0.224** | 0.139 | 0.718 |
| **FDA** | **F** | 0.842 | 0.948 | 0.577 | 0.715 | 0.811 | 0.513 | 0.592 | 0.933 | 0.261 | 0.172 | 0.705 |
| | **E** | **0.853** | **0.966** | 0.604 | **0.801** | 0.884 | **0.548** | 0.578 | 0.933 | 0.226 | 0.144 | 0.719 |
| | **S** | 0.851 | 0.963 | 0.599 | 0.787 | **0.896** | **0.548** | **0.586** | **0.953** | **0.275** | **0.191** | **0.730** |
| | **C** | 0.848 | 0.956 | **0.606** | 0.707 | 0.895 | **0.548** | **0.586** | 0.942 | 0.257 | 0.169 | 0.723 |

**Tabla 4.7:** (b) Valores de *accuracy* obtenidos por las diferentes técnicas de clasificación utilizando los conjuntos de datos del 11 al 20 de la tabla 4.6. Para cada técnica de clasificación y conjunto de datos el mejor resultado está subrayado y, además, el mejor método que utiliza una jerarquía está resaltado en negrita. La última columna de la tabla muestra los resultados obtenidos, al promediar los valores de todos los conjuntos de datos.

Usamos una validación cruzada de 10 pliegues para el proceso completo (desde la generación del clasificador plano inicial hasta la construcción de la jerarquía de clasificadores) y la tasa de acierto (o *accuracy*) como medida de evaluación.

### *4.7.1 Análisis de resultados*

La tabla 4.7 muestra los resultados de *accuracy* obtenidos por las diferentes técnicas de clasificación. En esta la segunda columna representa la metodología aplicada en la construcción de la jerarquía (Centroide=$C$, Euclídea=$E$, Semimétrica=$S$), así como la opción no jerárquica correspondiente a la clasificación plana ($F$). Para mejorar la legibilidad de la tabla, la hemos dividido en dos: (a) muestra los resultados de los datasets 1 al 10 y (b) muestra los resultados de los datasets 11 al 20. La tabla completa, incluidos los valores de la desviación estándar, pueden consultarse en el apéndice C (tabla C.1). No se ha podido obtener los resultados completos de *accuracy* para los conjuntos de datos Reuters, 10News y 20News aplicando las técnicas PART, JRIP y J48, debido a que estos métodos requieren largos tiempos de procesamiento. El tiempo de procesamiento requerido pueden verse influido por factores como: el número clases del problema de clasificación, el número de nodos que componen la jerarquía de clases generada y la velocidad de ejecución de las librerías utilizadas para crear los clasificadores locales.

En general, los resultados de la tabla 4.7 (a y b) muestran que los métodos que utilizan una jerarquía de clases mejoran, en términos de *accuracy*, la clasificación plana para la mayoría de las técnicas, excepto para los métodos NB, SVM y RF, en los cuales el método plano obtiene los valores más altos en la tasa de acierto comparado con los que utilizan jerarquías.

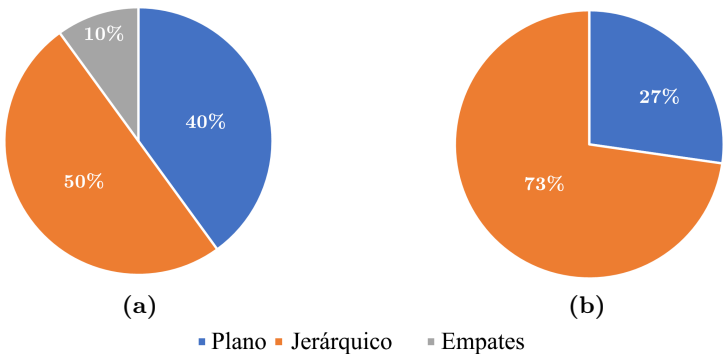



**Figura 4.10:** Visualización de los porcentajes en los que un clasificador obtiene el mejor *accuracy* según: (a) los conjuntos de datos y (b) las técnicas de clasificación.

La figura 4.10 muestra los porcentajes resultantes al comparar los métodos jerárquicos con el plano. Si analizamos los resultados agrupados por conjunto de datos (figura 4.10a), los clasificadores jerárquicos obtienen, un 50% de las veces, un mejor *accuracy*, mientras que el plano lo hace un 40%. Lo mismo ocurre si analizamos los resultados por técnicas de clasificación (figura 4.10b), donde los porcentajes son del 73% para los métodos jerárquicos y 27% para el plano. En consecuencia, se podría establecer que, en general, el uso de jerarquías de clases es útil para mejorar la calidad de la clasificación en contraposición a aplicar directamente una técnica plana tradicional.

En la siguiente subsección, y con el objetivo de analizar con un mayor nivel de detalle los resultados de la tabla 4.7, agregamos los resultados por conjunto de datos y técnicas de clasificación.

*Análisis de resultados agregados por conjuntos de datos y técnicas de clasificación*

La tabla 4.8 muestra los promedios agregados por conjuntos de datos, donde se excluyen los resultados de los datasets Reuters, 10NewsGroup y 20NewsGroups porque no fue posible completar sus experimentos, puesto que estos datasets son de tamaño considerable.

Al analizar los métodos jerárquicos encontramos que, en relación con la forma en la que inducen las jerarquías de clases, los métodos que obtienen los mejores resultados son los que utilizan distancia entre centroides, seguidos por los que usan nuestra semimétrica. Concretamente, el uso de la semimétrica obtiene los mejores resultados para los conjuntos Flare, Nuclear, TrafficLight y Vertebral. En la última posición se encuentra el método que utiliza la distancia euclídea, que destaca con el conjunto de datos Segmentation. Por otro lado, el clasificador plano obtiene el mayor *accuracy* promedio en 5 de los 17 conjuntos de datos.

La tabla 4.9 muestra los promedios agregados y la tabla 4.10 muestra las diferencias relativas por técnicas de clasificación, calculadas como $df = 1 - (Accuracy/MejorAccuracyxConjunto)$, donde $MejorAccuracyxConjunto$ es el valor más alto de accuracy por conjunto de datos. Se han excluido del análisis los métodos PART, JRIP y J48, porque no hemos podido obtener los resultados con todos los conjuntos de datos. La razón es que estos conjuntos de datos tienen una combinación de número de atributos y clases que los hacen complejos y lentos de aprender con estas técnicas de aprendizaje.

| **Datos** | **Plano** | **Jerárquico** | | |
|---|---|---|---|---|
| | **F** | **E** | **S** | **C** |
| Arrhytmia | 0.6905 | 0.6960 | 0.6958 | **0.6961** |
| Covtype | 0.6062 | 0.6295 | 0.6325 | **0.6345** |
| Dermatology | 0.9435 | 0.9275 | 0.9271 | **0.9301** |
| Flare | 0.7210 | 0.7233 | **0.7253** | 0.7193 |
| Forest | 0.8706 | 0.8776 | 0.8795 | **0.8805** |
| Glass | 0.6987 | 0.6811 | 0.6901 | **0.6941** |
| Letters | 0.6838 | 0.7178 | 0.7289 | **0.7334** |
| Nuclear | 0.9879 | 0.9820 | **0.9942** | 0.9808 |
| Pendigits | 0.8988 | 0.9427 | 0.9447 | **0.9454** |
| Satimage | 0.8384 | 0.8347 | 0.8345 | **0.8373** |
| Segmentation | 0.8682 | **0.9475** | 0.9458 | 0.9412 |
| Sports | 0.5663 | 0.5891 | 0.5922 | **0.5925** |
| TrafficLight | 0.6680 | 0.6653 | **0.6672** | 0.6506 |
| Usps | 0.8420 | 0.8635 | 0.8713 | **0.8788** |
| Vertebral | 0.4914 | 0.5043 | **0.5117** | 0.5045 |
| Yeast | 0.5726 | 0.5564 | 0.5591 | **0.5616** |
| Zoo | 0.9160 | 0.8962 | 0.9025 | **0.9054** |

**Tabla 4.8:** Valores promedio de *accuracy* por conjuntos de datos obtenidos con los métodos jerárquicos y plano. Para cada método de clasificación y conjunto de datos el mejor resultado está subrayado, y el mejor método que utiliza una jerarquía está resaltado en negrita.

| **Técnicas** | **Plano** | **Jerárquico** | | |
|---|---|---|---|---|
| | **F** | **E** | **S** | **C** |
| J48 | 0.7827 | 0.7894 | **0.7896** | 0.7869 |
| JRIP | 0.7606 | 0.7857 | **0.7870** | 0.7859 |
| KNN | 0.7372 | 0.7412 | **0.7420** | 0.7375 |
| NB | 0.6845 | 0.6164 | 0.6187 | **0.6582** |
| NNET | 0.6514 | 0.7687 | **0.7794** | 0.7634 |
| PART | 0.7817 | 0.7805 | 0.7831 | **0.7842** |
| RPART | 0.7423 | 0.7605 | **0.7629** | 0.7589 |
| RF | 0.8300 | 0.8256 | **0.8304** | 0.8218 |
| SVM | 0.7900 | 0.7827 | **0.7899** | 0.7814 |
| C50 | 0.7881 | 0.7884 | **0.7936** | 0.7926 |
| FDA | 0.7754 | 0.7949 | **0.8014** | 0.7966 |

**Tabla 4.9:** Valores promedio de *accuracy* por técnica de clasificación obtenidos con métodos jerárquicos y plano. Para cada método de clasificación el mejor resultado está subrayado, y el mejor método que utiliza una jerarquía está resaltado en negrita.

| **Técnicas** | **Plano** | **Jerárquico** | | |
|---|---|---|---|---|
| | **F** | **E** | **S** | **C** |
| **J48** | 0.0842 | **0.0760** | **0.0760** | 0.0795 |
| **JRIP** | 0.1082 | 0.0801 | **0.0795** | 0.0817 |
| **KNN** | 0.1451 | 0.1406 | **0.1396** | 0.1453 |
| **NB** | 0.2067 | 0.2846 | 0.2835 | **0.2370** |
| **NNET** | 0.2383 | 0.1064 | **0.0920** | 0.1115 |
| **PART** | 0.0867 | 0.0884 | 0.0847 | **0.0834** |
| **RPART** | 0.1318 | 0.1130 | **0.1095** | 0.1132 |
| **RF** | 0.0341 | 0.0397 | **0.0341** | 0.0443 |
| **SVM** | 0.0811 | 0.0903 | **0.0816** | 0.0920 |
| **C50** | 0.0801 | 0.0770 | **0.0700** | 0.0719 |
| **FDA** | 0.0934 | 0.0703 | **0.0632** | 0.0686 |

**Tabla 4.10:** Valores promedio de diferencias relativas por técnica de clasificación obtenidos con métodos jerárquicos y plano. Para cada método de clasificación el mejor resultado está subrayado, y el mejor método que utiliza una jerarquía está resaltado en negrita.

Si comparamos los resultados obtenidos con los métodos jerárquicos, nuestra función semimétrica $d_S$ obtiene, en promedio, los mejores resultados para 9 de las 11 técnicas de clasificación utilizadas, excepto para el NB o PART donde la distancia entre los centroides $d_C$ obtiene mejores valores de *accuracy* promedios. Por otro lado, cuando utilizamos el método plano en la clasificación, en general se obtienen los mejores resultados promedios cuando se usan las técnicas NB y SVM. El método que usa la distancia euclídea no obtiene, en promedio, los mejores resultados con ninguna de las técnicas utilizadas.

No cabe duda de que los métodos jerárquicos propuestos son capaces de mejorar el rendimiento obtenido con los métodos planos tradicionales. Sin embargo, esta mejora depende tanto de las características del problema de clasificación como de las técnicas de clasificación utilizadas. Tanto el método que utiliza distancia entre centroides como el que utiliza la semimétrica destacan al obtener los mejores resultados. En la siguiente sección realizamos un análisis estadístico de los resultados para establecer las diferencias entre los métodos de clasificación estudiados.

*Análisis estadístico de los resultados*

Con objeto de comprobar si las diferencias entre los resultados obtenidos por los métodos de clasificación evaluados son estadísticamente significativos, hemos aplicado el test estadístico no paramétrico de Friedman (ver subsección 2.5.2) que nos permite comparar múltiples algoritmos y dominios.

La hipótesis de partida sostiene que las diferencias estadísticas entre los resultados obtenidos al experimentar con el método propuesto y los métodos de referencia son significativas, mientras que, por el contrario, la hipótesis nula sostiene que tienen un rendimiento similar.

En primer lugar, se analizaron los resultados agrupados por conjuntos de datos. Aplicamos la prueba estadística de Friedman a los resultados, tomando en cuenta los conjunto de datos (tabla 4.8). Al calcular el valor estadístico de Friedman se obtuvo $\chi_F^2 = 5.6823$ y un valor-$p$ =0.128129 y observamos que el valor-$p$ no es inferior a 0.05; por lo tanto, no es posible rechazar la hipótesis nula que sostiene que los métodos tienen un rendimiento similar.

En segundo lugar, analizamos los resultados en función de las técnicas de clasificación (tabla 4.9). Encontramos que existen diferencias estadísticamente significativas según el test de Friedman con un valor-$p$ =0.00618 al usar *accuracy* promedios y un valor-$p$ =0.018 al analizar las diferencias relativas. Con el fin de detallar las diferencias entre métodos hemos aplicado el procedimiento posterior de Holm. Observamos que los pares con diferencias estadísticamente significativas y valores-$p$ inferiores a 0.05 son el método que usó distancia euclídea con el que usa semimétrica y el método que no usa jerarquía (plano) con el método que usa la semimétrica. Al analizar las diferencias relativas, se confirman estos resultados.

Según el test de ranking de Friedman que ordena los métodos del mejor al peor, en primer lugar se encuentra el método que usa la semimétrica, seguido del método de centroides, el método con distancia euclídea y, en última posición, se ubica el método plano.

A continuación, analizamos el efecto que tiene el tamaño de los conjuntos de datos (en términos de número de clases e instancias) sobre el rendimiento de los métodos. Para ello, primero hemos agrupado los conjuntos de datos en tres categorías, según el número de clases ($NumClass$): pequeño ($NumClass =< 6$), mediano ($6 < NumClass < 10$) y grande ($10 <$ NumClass). Observamos que, para el grupo con un número de clases pequeño, no existen diferencias significativas entre los métodos, según Friedman, con un valor-$p$ = 0.6921.

Al evaluar el grupo de datos con un número de clases mediano, encontramos que tampoco existen diferencias significativas entre los métodos con un valor-$p$=0.5163. Finalmente, al analizar los resultados del grupo con un alto número de clases, se encontró que el valor estadístico de Friedman es $\chi_F^2 = 7.35$ con un valor-$p$ =0.0615, dado que el valor p es muy cercano a 0.05, se aplicó el test de Iman y Davenport (ver sección 2.5.3), que es menos riguroso pero igualmente fiable. Según este test se obtuvo un estadístico de Iman de 3.090 y un valor-$p$ de 0.049 lo que permite rechazar la hipótesis nula y aceptar la alternativa que sostiene que hay diferencias entre los métodos evaluados. Sin diferencias significativas entre ellos están los métodos de semimétrica y centroides.

Agrupando los conjuntos de datos según el número de instancias, vemos que para los conjuntos de datos de tamaño pequeño ($NumInst < 526$) y medio ($527 < NumInst < 2101$) todos los métodos funcionan de manera similar y, por lo tanto, no se encontraron diferencias significativas entre los métodos; para los conjuntos de datos de tamaño grandes ($NumInst > 2100$), los métodos que utilizan una jerarquía de clases son mejores que los planos, en concreto el método basado en centroides y el método que usa la semimétrica ocupan la primera y segunda posición del orden de Friedman.

### *4.7.2 Análisis del efecto del parámetro de poda sobre la clasificación*

En esta sección analizaremos el efecto que tiene el parámetro $\alpha$ sobre la clasificación multiclase jerárquica. Para ello, realizamos un primer análisis para determinar si existe evidencia que nos permita asumir que usar un valor de $\alpha$ distinto de cero supone una mejora en la clasificación. En este primer experi-

| $\alpha$ | **Accuracy** |
|---|---|
| 0 | 0.813 |
| 0.1 | 0.813 |
| 0.2 | 0.809 |
| 0.3 | 0.808 |
| 0.4 | 0.804 |
| 0.5 | 0.803 |

**Tabla 4.11:** Valores promedio de *accuracy* globales agrupandos por técnicas de clasificación y conjuntos de datos, y valores $\alpha$ entre $(0 - 0.5)$. Se utilizaron los conjuntos de datos Dermatología, Forest, Glass, Sports y Usps.

mento se seleccionaron cinco conjuntos de datos (Dermatología, Forest, Glass, Sports y Usps) extraídos de la tabla 4.6 y seis técnicas de clasificación (SVM, J48, PART, RF, NNET, NB). Se estableció el valor de $\alpha$ entre 0 y 0.5 con incrementos de 0.1. Los promedio globales de *accuracy* (agrupandos por técnica de aprendizaje y conjuntos de datos) se muestran en la tabla 4.11. Como puede observarse, en este caso los parámetros que obtienen el mejor valor de *accuracy* son $\alpha = 0$ y $\alpha = 0.1$.

| **Técnicas** | $\alpha$ | **Conjuntos de Datos** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **SVM** | **0** | **0.589** | **0.690** | **0.945** | 0.744 | 0.898 | 0.666 | 0.803 | **0.996** |
| | **0.1** | **0.589** | 0.677 | 0.942 | **0.747** | **0.902** | **0.685** | **0.832** | **0.996** |
| **J48** | **0** | 0.709 | **0.707** | 0.949 | **0.744** | **0.866** | **0.714** | 0.720 | **0.961** |
| | **0.1** | **0.736** | 0.701 | **0.950** | 0.734 | 0.864 | 0.702 | **0.732** | **0.961** |
| **RPART** | **0** | **0.769** | **0.667** | 0.927 | 0.736 | **0.866** | **0.680** | **0.636** | **0.906** |
| | **0.1** | 0.759 | 0.646 | **0.944** | **0.740** | **0.866** | 0.665 | 0.477 | 0.885 |
| **RF** | **0** | **0.800** | **0.774** | **0.983** | **0.751** | 0.889 | **0.831** | **0.880** | **0.991** |
| | **0.1** | 0.781 | 0.759 | 0.975 | 0.742 | **0.891** | 0.824 | 0.878 | **0.991** |
| **NNET** | **0** | **0.661** | **0.503** | **0.964** | **0.759** | **0.917** | **0.703** | **0.763** | **0.913** |
| | **0.1** | 0.592 | 0.327 | 0.955 | 0.753 | 0.891 | 0.685 | 0.301 | 0.817 |
| **NB** | **0** | **0.589** | 0.146 | 0.771 | **0.551** | 0.852 | 0.583 | 0.535 | 0.793 |
| | **0.1** | **0.589** | **0.165** | **0.930** | 0.524 | **0.862** | **0.606** | **0.670** | **0.840** |
| **Técnicas** | $\alpha$ | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| **SVM** | **0** | **0.873** | **0.958** | 0.616 | 0.677 | 0.957 | 0.826 | **0.972** | **0.814** |
| | **0.1** | 0.872 | 0.700 | **0.628** | **0.701** | **0.969** | **0.848** | **0.972** | 0.804 |
| **J48** | **0** | 0.817 | 0.956 | **0.628** | 0.724 | **0.852** | **0.810** | 0.933 | 0.806 |
| | **0.1** | **0.820** | **0.964** | 0.618 | **0.739** | 0.850 | 0.806 | **0.943** | **0.808** |
| **RPART** | **0** | **0.798** | **0.948** | **0.584** | 0.724 | **0.830** | 0.816 | 0.866 | **0.784** |
| | **0.1** | 0.797 | **0.948** | 0.583 | **0.727** | 0.770 | **0.819** | **0.893** | 0.768 |
| **RF** | **0** | 0.875 | 0.975 | 0.707 | 0.803 | 0.938 | 0.845 | **0.973** | **0.868** |
| | **0.1** | **0.880** | **0.981** | **0.710** | **0.814** | **0.953** | **0.848** | 0.972 | 0.867 |
| **NNET** | **0** | **0.870** | **0.933** | **0.517** | **0.707** | **0.931** | **0.797** | **0.963** | **0.793** |
| | **0.1** | 0.851 | 0.923 | 0.364 | 0.673 | 0.739 | 0.787 | 0.944 | 0.707 |
| **NB** | **0** | 0.749 | 0.871 | 0.520 | 0.371 | 0.684 | 0.810 | 0.645 | 0.631 |
| | **0.1** | **0.813** | **0.891** | **0.557** | **0.431** | **0.725** | **0.813** | **0.893** | **0.687** |

**Tabla 4.12:** Valores de *accuracy* promedio usando 15 conjuntos de datos de la tabla 4.6 y valores de $\alpha = 0$ y $\alpha = 0.1$.

Realizamos un segundo experimento usando 15 conjuntos de datos extraídos de la tabla 4.6. Los valores del parámetro $\alpha$ usados fueron 0 y 0.1. Los resultados de la tabla 4.12 muestran que, para cuatro de las seis técnicas de aprendizaje, el parámetro de poda $\alpha = 0$ supera los resultados obtenidos con $\alpha = 0.1$, excepto

para los método NB y J48, que obtienen un mejor rendimiento promedio con $\alpha = 0.1$.

| Dataset | $\alpha$ | |
|---|---|---|
| | **0** | **0.1** |
| **Arrhytmia** | **0.686** | 0.674 |
| **Covtype** | **0.581** | 0.546 |
| **Dermatology** | 0.923 | **0.949** |
| **Flare** | **0.714** | 0.707 |
| **Forest** | **0.881** | 0.879 |
| **Glass** | **0.696** | 0.695 |
| **Letters** | **0.723** | 0.648 |
| **Pendigits** | **0.927** | 0.915 |
| **Satimage** | 0.83 | **0.839** |
| **Segmentation** | **0.94** | 0.901 |
| **Sports** | **0.595** | 0.577 |
| **TrafficLight** | 0.668 | **0.681** |
| **Usps** | **0.865** | 0.834 |
| **Vertebral** | 0.817 | **0.82** |
| **Zoo** | 0.892 | **0.936** |
| **Promedio** | **0.783** | 0.773 |

**Tabla 4.13:** Valores promedio de *accuracy* agrupando los resultados por conjuntos de datos y valores $\alpha$.

La tabla 4.13 muestra los resultados de *accuracy* agrupados por conjuntos de datos. El mejor valor según estos resultados es $\alpha = 0$. Para confirmar estos resultados hemos aplicado el test de Wilcoxon en los resultados de *accuracy*. Se obtuvo un valor estadístico de Wilcoxon de 35 y un valor crítico de 15, lo cual confirma la hipótesis nula que afirma que no existen diferencias estadísticamente significativas entre estos resultados.

Por otro lado, se aplicó el test estadístico de Wilcoxon para encontrar diferencias significativas entre los pares de métodos con un valor $\alpha = 0$ y $\alpha = 0.1$ y se encontró que únicamente existen diferencias entre los métodos NB y NNET: en el primero el aumento en $\alpha$ mejora la clasificación, mientras que en el segundo la empeora

1 Resumiendo, de manera general parece que aumentar el parámetro de poda $\alpha$ empeora la clasificación, ya que, al unir nodos, se diluyen las diferencias entre las clases detectadas por el clasificador plano.

## 4.8 Resumen

En este capítulo hemos propuesto un método que utiliza la jerarquía de clases con el objetivo de mejorar la tasa de acierto en los problemas de clasificación multiclase. En situaciones donde existe un alto número de clases, los métodos tradicionales no pueden discernir correctamente las nuevas observaciones, dado el alto número de posibilidades. La propuesta consiste en construir un modelo jerárquico de clasificadores descomponiendo el problema multiclase original en varios problemas de clasificación más sencillos.

El método se basa en la inferencia de una jerarquía de clases a partir de la matriz de confusión de un clasificador plano. Usando la matriz de confusión, se define una semimétrica, la cual mide la similitud entre clases y sirve para construir una matriz de distancias.

A continuación, se usa un algoritmo de agrupamiento jerárquico para inferir la jerarquía de clases, que se usa para construir una jerarquía de clasificadores siguiendo el método LCPN (un clasificador binario por cada nodo interno de la jerarquía). También introducimos un procedimiento de poda de la jerarquía de clases, el cual nos permite reducir la complejidad de la clasificación jerárquica al reducir el número de clasificadores locales. Finalmente, propusimos una variante del método de poda usando un umbral que permite adaptar la poda según los requerimientos del problema de clasificación.

Los experimentos con veinte conjuntos de datos multiclase muestran que, a nivel general, las técnicas jerárquicas son capaces de mejorar la tasa de acierto con respecto a la aproximación básica plana en los problemas multiclase no jerárquicos, el cual era el primer objetivo que nos planteamos en este trabajo de investigación. Además, el método propuesto para inferir la jerarquía de clases, basado en la distancia semimétrica es competitivo comparado con el método basado en centroides y supera a otras propuestas como las basadas en la distancia euclídea.

Con respecto al método de poda presentado, hemos encontrado que los mejores valores de $\alpha$ son entre 0 y 0.1. Valores mayores de $\alpha$ empeoran la tasa de acierto en la clasificación.

Estos primeros resultados plantean nuevas cuestiones sobre el uso de jerarquías en los problemas de clasificación multiclase. Hasta ahora hemos utilizado las técnicas de clasificación de forma no probabilística, analizando el resultado de la clasificación como perteneciente o no a una clase específica. Nos preguntamos: ¿Es posible utilizar clasificadores suaves (*soft*), donde el resultado no

es una etiqueta de clase, sino una probabilidad de pertenecer a una clase?, ¿podría esto mejorar el rendimiento del proceso de clasificación en problemas multiclase? Y, si es así, ¿qué problemas o ventajas aparecerían como resultado de su aplicación?. En el próximo capítulo se analizará en profundidad la aplicación de los clasificadores jerárquicos desde un punto de vista de los estimadores de probabilidad.

# Capítulo 5

# Clasificación jerárquica multiclase usando estimadores de probabilidad

En este capítulo continuamos con el estudio de los clasificadores jerárquicos multiclase, utilizando para ello predictores capaces de retornar una probabilidad de pertenencia por cada una de las clases con el objetivo de obtener una mejor calidad en la clasificación. Esta propiedad permite explorar variantes al procedimiento de creación y aplicación de los modelos de clasificación jerárquica multiclase presentado en el capítulo 4.

En particular, se proponen formas alternativas de inferir la jerarquía de clases, de podar el árbol jerárquico y de aplicar el modelo a nuevos datos. De la misma forma que en el capítulo anterior, las propuestas son evaluadas experimentalmente usando una colección de datos y varias técnicas de clasificación.

## 5.1 Clasificadores suaves y discretos

En muchos problemas prácticos de clasificación es útil saber no solo las predicciones dadas por el modelo, sino también la confianza que el clasificador tiene en ellas. Por ejemplo, supongamos que, un banco utiliza un clasificador para tomar decisiones sobre la concesión o no de préstamos. Imaginemos que el clasificador predice la clase «sí» para dos clientes, $p_1$ y $p_2$, lo que significa que el préstamo se concede. Sin embargo, ¿son ambas decisiones igualmente arriesgadas para el banco? ¿Cambiaría el banco su decisión si supiera que el clasificador está 90% seguro al predecir que $p_1$ es de clase «sí» pero solo un 55% para $p_2$?

Según las salidas que puede tener un clasificador, éstos pueden ser de dos tipos: los clasificadores *crisp* o discretos, que devuelven una etiqueta de clase (de entre las posibles), y los clasificadores *soft* (suaves o estimadores de probabilidad), que estiman la probabilidad de pertenencia a cada una de las posibles clases realizando luego la clasificación basada en estas probabilidades estimadas.

Formalmente, dada una instancia $x \in \mathcal{X}$ e $y_i \in \mathcal{Y}$, $1 \leq i \leq m$, cada una de las posibles etiquetas de clase, un clasificador suave estima para $x$ el siguiente vector de probabilidades $\vec{p}(x)$:

$$\vec{p}(x) = \langle P(y_1 \mid x), P(y_2 \mid x) \ldots, P(y_m \mid x) \rangle \tag{5.1}$$

### *5.1.1 Clasificación jerárquica usando clasificadores suaves*

Siguiendo la línea del capítulo anterior, presentamos un método de clasificación jerárquica multiclase que utiliza clasificadores suaves y que consta de los mismos pasos vistos para el caso "*crisp*": inferencia de la jerarquía de clases, construcción del modelo y aplicación de este.

El motivo de usar clasificadores suaves es aprovechar la información adicional que supone conocer para cada instancia la probabilidad de pertenecer a cada una de las clases (y no solo la etiqueta de clase que finalmente se le asignará), así, podemos saber cuál es el nivel de confusión entre las clases (cuanto más parecidas sean las probabilidades mayor será la confusión del clasificador al intentar determinar cuál es la clase final a asignar).

### 5.1.2 Inferencia de jerarquía de clases usando clasificadores suaves

El proceso de inferencia de la jerarquía de clases propuesto en la sección 4.3 primero determinar la distancia entre clases, la cual se usa para representar la relación entre clases mediante un dendrograma (estructura jerárquica). En el capítulo anterior propusimos calcular las distancias a partir de la matriz de confusión de un clasificador *crisp* entrenado para tal efecto. En esta ocasión proponemos usar un clasificador suave y construir su matriz de confusión utilizando las probabilidades estimadas para cada una de las instancias de entrenamiento. De esta forma, la matriz de confusión se obtiene sumando los vectores de probabilidad estimados de todas las instancias de entrenamiento de acuerdo a su clase real (ecuación 5.2).

$$M_{i,j} = \sum_{\langle x,i\rangle \in \mathcal{X},} P(\mathcal{Y}(x) = j \mid x) \tag{5.2}$$

**Ejemplo 5.1.1: Construcción de matriz de confusión suave**

*Consideremos un problema de clasificación donde se debe distinguir entre las clases* $\mathcal{Y} = \{a, b, c\}$. *Supongamos que se ha creado un clasificador suave que se aplica a tres instancias nuevas* $(e_1, e_2, e_3)$, *dando como resultado tres vectores de probabilidad* $E = \{\vec{p}(e_1), \vec{p}(e_2), \vec{p}(e_3)\}$.

La tabla 5.1 ilustra el proceso de creación de la matriz de confusión a través de las predicciones dadas para estas tres instancias cuyas clases reales son $a$, $b$, y $a$, respectivamente. El proceso de alimentación de la matriz de confusión se realiza instancia a instancia. Los vectores de probabilidad de las instancias $e_1$ y $e_2$ se colocan directamente en las filas 1 y 2 ya que sus clases reales son $a$ y $b$, respectivamente (tablas 5.1 (a) y (b)). Al considerar la instancia $e_3$, como su clase real es $a$ su vector de probabilidades se suma a los valores de la primera fila ( tabla 5.1 (c)).

| | | **Predicho** | | |
|---|---|---|---|---|
| | | a | b | c |
| **Real** | a | 0.7 | 0.1 | 0.2 |
| | b | | | |
| | c | | | |

**(a)** $\vec{p}(e_1) =< 0.7, 0.1, 0.2 >$
$clase\_real(e_1) = a$

| | | **Predicho** | | |
|---|---|---|---|---|
| | | a | b | c |
| **Real** | a | 0.7 | 0.1 | 0.2 |
| | b | 0.4 | 0.6 | 0 |
| | c | | | |

**(b)** $\vec{p}(e_2) =< 0.4, 0.6, 0 >$
$clase\_real(e_2) = b$

| | | **Predicho** | | |
|---|---|---|---|---|
| | | a | b | c |
| **Real** | a | 1.5 | 0.1 | 0.4 |
| | b | 0.4 | 0.6 | 0 |
| | c | | | |

**(c)** $\vec{p}(e_3) =< 0.8, 0, 0.2 >$
$clase\_real(e_3) = a$

**Tabla 5.1:** Ejemplo de creación de la matriz de confusión utilizando las predicciones dadas por un clasificador suave para tres ejemplos $e_1$, $e_2$, y $e_3$.

*Fin del ejemplo 5.1.1* ■

A continuación, podemos derivar la matriz de distancias usando la distancia construida a partir de la semimétrica definida en el apartado 4.3.2, ya que en los experimentos presentó resultados competitivos de tasa de acierto para la mayoría de las técnicas de clasificación. Una vez obtenida la matriz de distancia, la jerarquía de clases se construye aplicando el algoritmo de agrupación jerárquica aglomerativa. Finalmente para obtener el clasificador jerárquico multiclase se debe construir el conjunto de clasificadores locales tal y como se realizó en la sección 4.4. Veamos un ejemplo de construcción de la matriz de distancias.

**Ejemplo 5.1.2: Inferencia de jerarquía de clases con métodos suaves**

*Consideremos el conjunto de datos para la segmentación de imágenes compartido por el grupo de visión de la Universidad de Massachusetts [Bro90] a través del repositorio UCI [Aha87]. El objetivo de este conjunto de datos es clasificar imágenes de exteriores de 3x3 pixeles. El número de instancias es 2310 y las posibles clases son* $\mathcal{Y} = \{brickface, sky, foliage, cement, window, path, grass\}$.

La matriz de confusión obtenida siguiendo el procedimiento propuesto que utiliza clasificadores suaves se muestra en la tabla 5.2 (a). Por el contrario, la tabla 5.2 (b) muestra la matriz de confusión obtenida mediante el procedimiento de conteo de errores que comúnmente se utiliza para construirla. Como se puede observar, en la primera forma de construir la matriz usando probabilidades condicionales (decisiones "suaves"), los errores son distribuidos entre cada una de clases dependiendo del nivel

de confusión, mientras que usando clasificadores que hacen decisiones "discretas" la matriz de confusión refleja el hecho de acertar o no en la clasificación.

A continuación, la tabla 5.3 (a y b) muestra las matrices de distancia obtenidas a partir de las dos matrices de confusión ("suave" y "discreta", respectivamente).

Finalmente, la figura 5.1 muestra las jerarquías de clase y dendrogramas obtenidos a partir de las matrices de distancias anteriores. Nótese que, desde el punto de vista de la similitud entre clases, la estimación de los valores de las clases o de las probabilidades de pertenencia (incluso para los mismos datos) no es equivalente (es decir, la función $d_S$ calculada es diferente) y, por lo tanto, conduce a diferentes jerarquías de clases.

| | **Predicho** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Real** | **Grass** | **Sky** | **Window** | **Cement** | **Path** | **Brickface** | **Foliage** |
| **Grass** | 293.936 | 0.274 | 0.526 | 0.365 | 0.234 | 0.283 | 0.681 |
| **Sky** | 0.184 | 294.159 | 0.214 | 0.457 | 0.290 | 0.162 | 0.367 |
| **Window** | 0.657 | 0.391 | 228.065 | 21.232 | 0.812 | 5.436 | 40.045 |
| **Cement** | 0.571 | 0.996 | 21.067 | 269.339 | 3.102 | 2.189 | 4.964 |
| **Path** | 0.567 | 0.360 | 0.295 | 2.069 | 291.593 | 0.244 | 0.450 |
| **Brickface** | 0.272 | 0.223 | 3.635 | 1.432 | 0.409 | 287.547 | 1.321 |
| **Foliage** | 0.813 | 0.597 | 43.197 | 2.105 | 0.559 | 1.139 | 249.172 |

**(a)** Matriz de confusión usando técnicas suaves.

| | **Predicho** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Real** | **Grass** | **Sky** | **Window** | **Cement** | **Path** | **Brickface** | **Foliage** |
| **Grass** | 297 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Sky** | 0 | 297 | 0 | 0 | 0 | 0 | 0 |
| **Window** | 0 | 0 | 231 | 13 | 0 | 0 | 53 |
| **Cement** | 0 | 0 | 14 | 281 | 2 | 0 | 0 |
| **Path** | 0 | 0 | 0 | 0 | 297 | 0 | 0 |
| **Brickface** | 0 | 0 | 6 | 0 | 0 | 291 | 0 |
| **Foliage** | 0 | 0 | 10 | 2 | 0 | 0 | 285 |

**(b)** Matriz de confusión usando técnicas crisp.

**Tabla 5.2:** Ejemplo de creación matriz de confusión usando el conjunto de datos *Segmentation*.

| | **Grass** | **Sky** | **Window** | **Cement** | **Path** | **Brickface** | **Foliage** |
|---|---|---|---|---|---|---|---|
| **Grass** | | | | | | | |
| **Sky** | 0.999 | | | | | | |
| **Window** | 0.998 | 0.998 | | | | | |
| **Cement** | 0.998 | 0.997 | 0.929 | | | | |
| **Path** | 0.998 | 0.998 | 0.998 | 0.991 | | | |
| **Brickface** | 0.999 | 0.999 | 0.984 | 0.993 | 0.998 | | |
| **Foliage** | 0.997 | 0.998 | 0.859 | 0.988 | 0.998 | 0.995 | |

**(a)** Matriz de distancias usando técnicas suaves.

| | **Grass** | **Sky** | **Window** | **Cement** | **Path** | **Brickface** | **Foliage** |
|---|---|---|---|---|---|---|---|
| **Grass** | | | | | | | |
| **Sky** | 1 | | | | | | |
| **Window** | 1 | 1 | | | | | |
| **Cement** | 1 | 1 | 0.955 | | | | |
| **Path** | 1 | 1 | 1 | 0.997 | | | |
| **Brickface** | 1 | 1 | 0.990 | 1 | 1 | | |
| **Foliage** | 1 | 1 | 0.894 | 0.997 | 1 | 1 | |

**(b)** Matriz de distancias usando técnicas discretas.

**Tabla 5.3:** Ejemplo de creación de la matriz de distancias usando el conjunto de datos *Segmentation*.

*Fin del ejemplo 5.1.2* ■

Una vez inferida la jerarquía el siguiente paso es construir un árbol de clasificadores utilizando la estrategia LCPN (tal y como se realizó en el capítulo 4). Con el objetivo de reducir el número de clasificadores locales a entrenar en este capítulo aplicamos la variante del método de poda de la jerarquía de clases presentada en la sección 4.5 con un valor $\alpha = 0$.

Podar la jerarquía de clases es el paso previo a la creación del conjunto de clasificadores locales. La estrategia de entrenamiento de los clasificadores locales es la misma presentada en el capítulo 4 por lo que no será explicada aquí. En su lugar, la siguiente sección analizará el método de aplicación del modelo para clasificar nuevas instancias.

## 5.2 Método de predicción Top-down & Bottom-up

La estrategia utilizada hasta ahora para realizar las predicciones usando el conjunto de clasificadores ordenados jerárquicamente ha sido la estrategia *Top-down*. En este escenario, para clasificar una nueva instancia, el árbol de clasificadores se recorre de manera descendente aplicando los clasificadores desde la raíz hasta que se alcanza una hoja.

Si bien es posible aplicar el método *Top-down* presentado en la sección 4.4 con clasificadores suaves, el contar con las probabilidades de pertenencia para cada una de las clases (como respuesta de los clasificadores locales) hace posible incorporar esta información durante la aplicación de los modelos con el objetivo de refinar el resultado final de la clasificación.

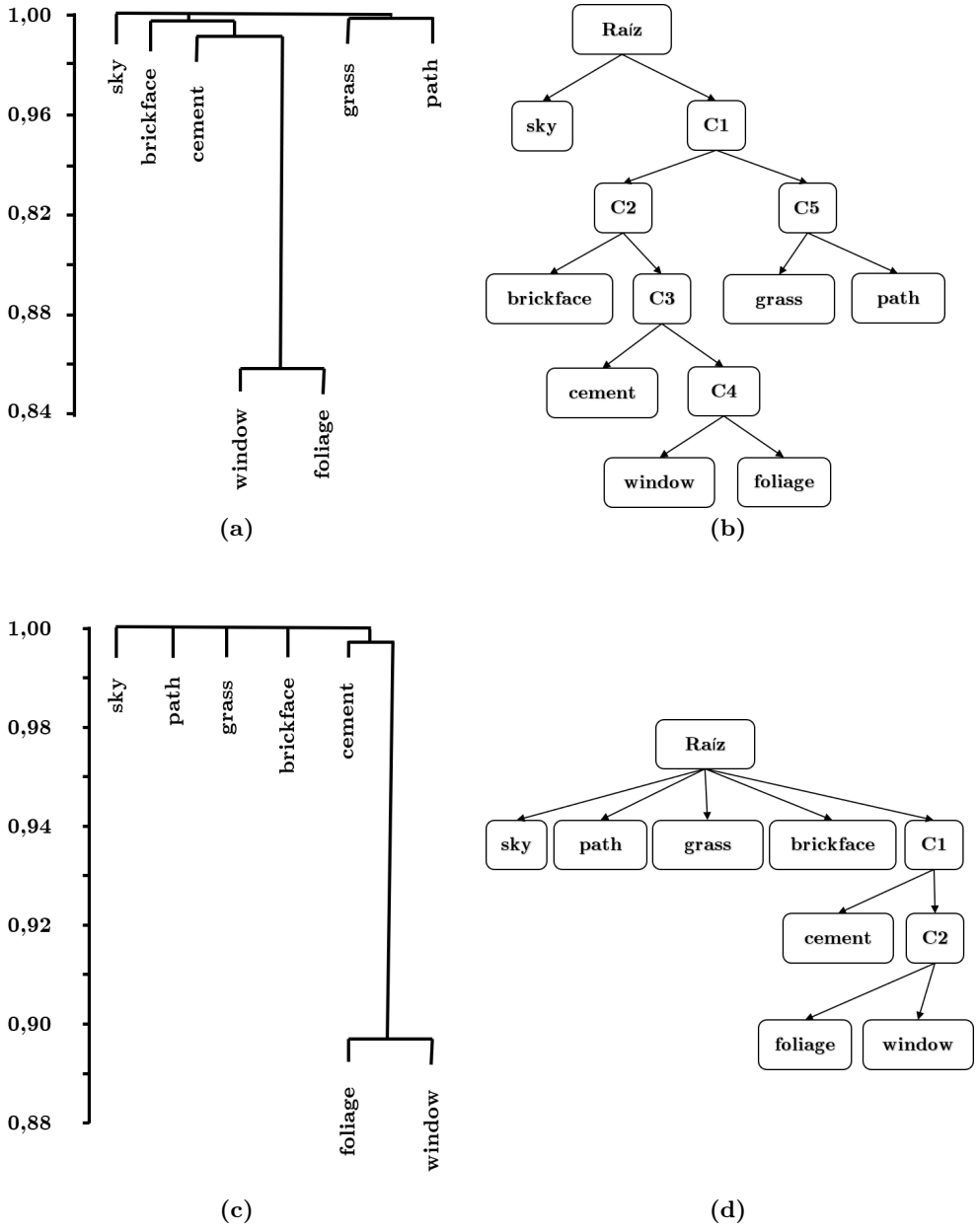



**Figura 5.1:** Dendrogramas (izquierda) y jerarquías de clases (derecha) inducidas mediante el uso de la matriz de confusión generada por un clasificador suave (arriba) y un clasificador discreto (abajo) para el conjunto de datos Segmentation [Bro90] del repositorio UCI [Aha87].

En esta sección presentamos el método *Top-down & Bottom-up* como una alternativa al método *Top-down*. Dada una nueva instancia $x$ a clasificar, se atraviesa el árbol de clasificadores de arriba hacia abajo aplicando todos los clasificadores. En cada uno de ellos estimamos las probabilidades de pertenencia a las clases que representan sus nodos descendientes. Una vez alcanzadas las hojas, en cada una de ellas se construye un vector de probabilidades condicionales $\langle P(y_1 \mid x), \ldots, P(y_n \mid x)\rangle$. Ya que las hojas representan las clases, el vector en una hoja $y_i$ tiene todos sus componentes iguales a cero, excepto el componente $P(y_i \mid x) = 1$. Una vez generados los vectores de probabilidad en las hojas, estos se propagan de abajo hacia arriba, recorriendo el árbol de forma *bottom-up*. La figura 5.2 muestra un ejemplo de los posibles caminos y sus vectores de probabilidades para un problema de clasificación entre seis clases, el número de caminos posibles es igual al número de clases.

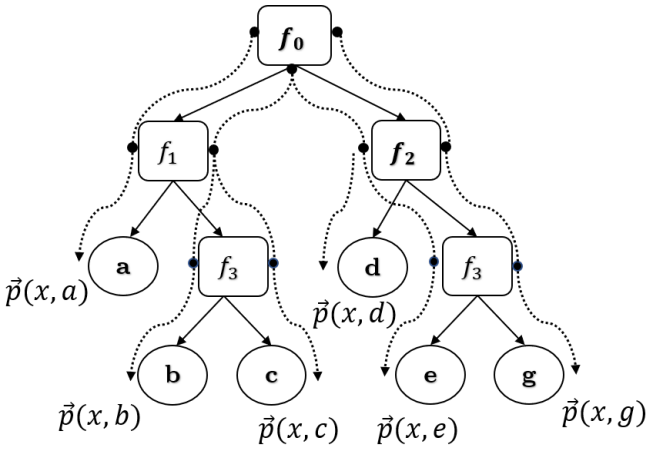



**Figura 5.2:** Ejemplo de posibles caminos y vectores de probabilidad para un problema de clasificación entre seis clases $\{a, b, c, d, e, g\}$.

Durante el proceso *bottom-up*, el cómputo de los vectores de probabilidad en los nodos internos se realiza de la siguiente forma. Dado un nodo $\nu$ que no es una hoja, denotamos el conjunto de sus nodos hijos como $Chl(\nu)$. Entonces, el vector de probabilidad de un ejemplo $x$ en el nodo $\nu$, $\vec{p}(x, \nu)$, se obtiene como:

$$\vec{p}(x,\nu) = \sum_{\mu \in Chl(\nu)} (P(\mu \mid x) \times \vec{p}(x,\mu)) \tag{5.3}$$

Finalmente, la clase pronosticada para $x$ es el componente de más alto valor en el vector $\vec{p}(x, raíz)$.

## 5.3 Experimentos

El propósito de este estudio es evaluar experimentalmente el efecto de usar clasificadores suaves al construir clasificadores jerárquicos multiclase. En particular se analiza si el procedimiento propuesto mejora la tasa de acierto en comparación con el uso de métodos discretos.

Para ello, compararemos las propuestas basadas en el uso de clasificadores suaves con nuestras propuestas basadas en clasificadores discretos (presentadas en el capítulo 4). En concreto, los experimentos se centran en evaluar los siguientes métodos:

- **C-TD.** Es el procedimiento original, incluye la construcción de la matriz de confusión utilizando un clasificador plano discreto y para la aplicación del modelo la técnica *Top-down*.
- **C-TDBU.** Incluye la creación de la matriz utilizando un clasificador plano discreto y la técnica de aplicación *Top-down & Bottom-up*.
- **P-TD.** Usa un clasificador plano suave para la construcción de la matriz de confusión, mientras que usa la técnica tradicional *Top-down* al realizar las predicciones.
- **P-TDBU.** Este método construye la matriz de confusión utilizando las probabilidades predichas por un clasificador suave y aplica el modelo con la estrategia *Top-down & Bottom-up*.

### *5.3.1 Configuración de los experimentos*

El esquema seguido para la construcción del modelo de clasificación jerárquico multiclase es el mismo que el descrito en la sección 4.7.

Para estos experimentos se seleccionaron 15 conjuntos de datos que se presentaron en la tabla 4.6 de la sección 4.7. No fue posible usar la totalidad de los conjuntos de datos porque algunos necesitaban un tiempo excesivo de procesamiento o requerían una extensiva cantidad de memoria para ejecutarlos. La tabla 5.4, muestra el conjunto de datos seleccionados y sus características, ade-

más el preprocesamiento de los datos es el mismo que en el capítulo 4 (descrito en el apéndice A).

| **Datos** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **Id** | **Arrhytmia** | **Covtype** | **Dermatology** | **Flare** | **Forest** | **Glass** | **Letters** | **Pendigits** |
| **NumInst** | 416 | 2100 | 358 | 1066 | 523 | 214 | 2600 | 7494 |
| **NumAtr** | 330 | 54 | 34 | 19 | 27 | 9 | 16 | 16 |
| **NumClase** | 7 | 7 | 6 | 6 | 4 | 6 | 26 | 10 |

| **Datos** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|
| **Id** | **SatImage** | **Segmentation** | **Sports** | **TrafficLight** | **Usps** | **Vertebral** | **Zoo** |
| **NumInst** | 1795 | 3000 | 8000 | 300 | 3000 | 310 | 101 |
| **NumAtr** | 36 | 18 | 13 | 10 | 256 | 6 | 16 |
| **NumClase** | 6 | 7 | 10 | 6 | 10 | 4 | 7 |

**Tabla 5.4:** Información sobre los conjuntos de datos utilizados en los experimentos: número de instancias, atributos y clases.

En esta ocasión seleccionamos únicamente siete técnicas de clasificación ya que a nuestro criterio son las más representativas de las estudiadas en las sección 2.6. En concreto, utilizamos dos árboles de decisión (C50, RPART), naive bayes (NB), red neuronal (NNET), $randomforest$ (RF), $k$-vecinos más cercanos (KNN) y la máquina de vectores de soporte (SVM).

El apéndice B contiene información adicional sobre los parámetros usados en cada una de las técnicas de clasificación. El proceso completo se realizó mediante validación cruzada de diez *folds* y usamos la tasa de acierto como medida de la eficiencia en la clasificación.

### *5.3.2 Evaluación de métodos suaves en la construcción de clasificadores jerárquicos multiclase*

La tabla 5.5 muestra los resultados obtenidos en los experimentos. Para mejorar la legibilidad de la tabla, la hemos partido en dos: (a) muestra los resultados de aplicando las técnicas SVM, RPART, RF, NNET y NB, y (b) contiene los resultados obtenidos con las técnicas C50 y KNN. En ambas el mejor resultado para cada técnica de clasificación y conjunto de datos está marcado en negrita. Una tabla que detallada con los valores de diferencias relativas se muestran en el apéndice C (tabla C.3). En general, si analizamos los valores promedios por dataset y por técnica, y contamos el número de veces que se logra un mayor valor de *accuracy* por cada técnica, entonces el método C-TD obtiene el 21% de las veces el valor más alto.

Por otro lado, si nos enfocamos en los métodos que sí incluyen estimaciones de probabilidad, el método P-TDBU obtiene un porcentaje del 29%, C-TDBU

| | | Conjunto de Datos | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **SVM** | **C-TD** | **0.589** | 0.690 | **0.942** | 0.755 | 0.898 | 0.685 | 0.938 | 0.995 |
| | **C-TDBU** | **0.589** | **0.692** | 0.939 | 0.758 | **0.902** | **0.687** | **0.942** | **0.996** |
| | **P-TD** | **0.589** | 0.690 | 0.934 | 0.757 | 0.898 | 0.675 | 0.923 | 0.995 |
| | **P-TDBU** | **0.589** | **0.692** | 0.931 | **0.759** | 0.898 | 0.680 | 0.930 | 0.995 |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | 0.896 | 0.958 | 0.616 | 0.674 | 0.963 | 0.826 | **0.972** | 0.826 |
| | **C-TDBU** | 0.897 | **0.962** | 0.623 | 0.710 | **0.967** | 0.826 | 0.953 | **0.830** |
| | **P-TD** | **0.898** | 0.960 | 0.616 | 0.680 | 0.961 | 0.826 | 0.952 | 0.824 |
| | **P-TDBU** | **0.898** | **0.962** | **0.625** | **0.713** | 0.964 | **0.832** | 0.954 | 0.828 |
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **RPART** | **C-TD** | 0.767 | 0.652 | **0.942** | 0.745 | **0.866** | 0.651 | 0.592 | 0.885 |
| | **C-TDBU** | 0.767 | 0.652 | **0.942** | 0.745 | **0.866** | 0.660 | 0.592 | 0.885 |
| | **P-TD** | **0.769** | **0.655** | 0.936 | 0.741 | **0.866** | 0.656 | **0.690** | **0.886** |
| | **P-TDBU** | **0.769** | **0.655** | 0.936 | **0.747** | **0.866** | **0.665** | **0.690** | **0.886** |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | **0.847** | 0.948 | 0.581 | **0.727** | **0.826** | **0.816** | **0.866** | 0.781 |
| | **C-TDBU** | **0.847** | 0.948 | 0.581 | **0.727** | **0.826** | **0.816** | **0.866** | 0.781 |
| | **P-TD** | 0.845 | **0.955** | 0.582 | 0.724 | 0.821 | **0.816** | **0.866** | 0.787 |
| | **P-TDBU** | 0.845 | **0.955** | **0.583** | 0.724 | 0.821 | **0.816** | **0.866** | **0.788** |
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **RF** | **C-TD** | 0.779 | 0.750 | 0.972 | 0.750 | 0.889 | 0.805 | **0.968** | **0.992** |
| | **C-TDBU** | **0.795** | 0.757 | 0.969 | **0.753** | 0.887 | 0.811 | **0.968** | **0.992** |
| | **P-TD** | 0.747 | 0.753 | **0.978** | 0.745 | **0.897** | **0.812** | 0.949 | 0.988 |
| | **P-TDBU** | 0.767 | **0.760** | **0.978** | **0.753** | 0.891 | 0.811 | 0.959 | 0.990 |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | **0.913** | 0.979 | 0.715 | **0.814** | **0.954** | 0.832 | **0.972** | 0.872 |
| | **C-TDBU** | **0.913** | 0.980 | 0.715 | 0.804 | 0.953 | 0.842 | **0.972** | **0.874** |
| | **P-TD** | 0.905 | **0.982** | 0.710 | 0.797 | 0.937 | **0.855** | 0.953 | 0.867 |
| | **P-TDBU** | 0.907 | 0.981 | **0.716** | 0.793 | 0.946 | 0.852 | **0.972** | 0.872 |
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **NNET** | **C-TD** | 0.637 | 0.695 | **0.981** | 0.757 | 0.894 | 0.674 | 0.416 | 0.864 |
| | **C-TDBU** | 0.622 | **0.705** | 0.975 | **0.758** | 0.896 | 0.695 | 0.544 | 0.841 |
| | **P-TD** | 0.629 | 0.697 | 0.975 | 0.750 | **0.906** | 0.684 | 0.795 | **0.919** |
| | **P-TDBU** | **0.662** | 0.697 | 0.972 | 0.751 | 0.896 | **0.703** | **0.807** | 0.905 |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | 0.893 | 0.969 | 0.635 | 0.804 | 0.916 | 0.774 | 0.943 | 0.790 |
| | **C-TDBU** | 0.888 | 0.970 | 0.640 | 0.809 | 0.912 | 0.784 | 0.953 | 0.800 |
| | **P-TD** | **0.901** | 0.969 | 0.633 | **0.820** | **0.925** | 0.806 | **0.972** | 0.825 |
| | **P-TDBU** | 0.891 | **0.971** | **0.642** | 0.810 | 0.922 | **0.810** | 0.959 | **0.827** |
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **NB** | **C-TD** | **0.087** | 0.643 | **0.766** | 0.521 | **0.856** | 0.400 | 0.474 | **0.769** |
| | **C-TDBU** | **0.087** | 0.643 | **0.766** | 0.521 | **0.856** | 0.400 | 0.486 | **0.769** |
| | **P-TD** | 0.065 | 0.646 | 0.735 | **0.541** | **0.856** | **0.440** | 0.494 | 0.761 |
| | **P-TDBU** | 0.065 | **0.647** | 0.735 | **0.541** | **0.856** | 0.417 | **0.517** | 0.762 |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | **0.699** | 0.713 | 0.384 | 0.377 | **0.668** | **0.810** | **0.878** | 0.603 |
| | **C-TDBU** | **0.699** | 0.713 | **0.386** | 0.377 | **0.668** | **0.810** | **0.878** | **0.604** |
| | **P-TD** | **0.699** | **0.714** | 0.382 | **0.414** | 0.660 | **0.810** | 0.814 | 0.602 |
| | **P-TDBU** | **0.699** | **0.714** | 0.385 | 0.408 | 0.660 | **0.810** | 0.814 | 0.602 |

**Tabla 5.5:** (a) Valores de *accuracy* obtenidos por los métodos multiclase jerárquicos suaves usando las técnicas SVM, RPART, RF, NNET y NB para la generación de la jerarquía de clasificadores. La técnica que obtiene el mejor resultado para cada método y conjunto de datos se resalta en negrita.

| | | Conjunto de Datos | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **C50** | **C-TD** | 0.694 | **0.689** | **0.950** | 0.743 | **0.864** | **0.727** | **0.871** | 0.957 |
| | **C-TDBU** | 0.694 | **0.689** | **0.950** | **0.747** | **0.864** | **0.727** | **0.871** | 0.957 |
| | **P-TD** | **0.714** | 0.687 | 0.947 | 0.736 | **0.864** | 0.707 | 0.868 | **0.959** |
| | **P-TDBU** | **0.714** | 0.687 | 0.947 | 0.739 | **0.864** | 0.707 | 0.869 | **0.959** |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | 0.853 | 0.971 | 0.622 | 0.747 | 0.848 | **0.816** | **0.816** | 0.811 |
| | **C-TDBU** | **0.854** | 0.971 | **0.623** | 0.747 | 0.849 | **0.816** | **0.816** | 0.812 |
| | **P-TD** | 0.853 | **0.973** | 0.622 | **0.767** | 0.857 | **0.816** | **0.816** | 0.812 |
| | **P-TDBU** | 0.853 | **0.973** | **0.623** | **0.767** | **0.858** | **0.816** | **0.816** | **0.813** |
| **Técnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **KNN** | **C-TD** | **0.647** | 0.666 | 0.844 | 0.732 | 0.892 | 0.622 | **0.953** | **0.993** |
| | **C-TDBU** | 0.642 | 0.661 | 0.861 | **0.743** | 0.893 | 0.631 | **0.953** | **0.993** |
| | **P-TD** | 0.642 | **0.672** | 0.852 | 0.732 | **0.894** | 0.646 | 0.952 | **0.993** |
| | **P-TDBU** | 0.645 | 0.670 | **0.877** | 0.741 | 0.889 | **0.654** | 0.952 | **0.993** |
| | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| | **C-TD** | 0.895 | **0.942** | 0.601 | 0.563 | 0.954 | **0.839** | 0.833 | 0.798 |
| | **C-TDBU** | 0.893 | **0.942** | 0.605 | 0.567 | **0.955** | 0.832 | 0.834 | 0.800 |
| | **P-TD** | **0.897** | 0.939 | 0.601 | 0.557 | 0.953 | **0.839** | **0.853** | 0.801 |
| | **P-TDBU** | 0.891 | 0.939 | **0.606** | **0.581** | 0.953 | **0.839** | **0.853** | **0.806** |

**Tabla 5.5:** (b) Valores de *accuracy* obtenidos por los métodos multiclase jerárquicos usando las técnicas C50 y KNN para la generación de la jerarquía de clasificadores. La técnica que obtiene el mejor resultado para cada método y conjunto de datos se resalta en negrita.

un 27% y P-TD un 23%. Estos resultados muestran que las variaciones del método obtienen una mejoría al realizar la clasificación.

La tabla 5.6 muestra los resultados del *accuracy* agrupados por conjuntos de datos. Desde estos puntos de vista, en promedio, la técnica que obtiene los valores más altos en 9 de los 15 conjuntos de datos utilizados es P-TDBU con un valor de *accuracy* promedio de 0.7907. El segundo mejor promedio lo obtuvo el método P-TD con 0.7885, y el tercer lugar le corresponde al método C-TDBU con 0.7858.

Concretamente, para los conjuntos de datos Forest, Pendigits y Satimage el método P-TD que utiliza probabilidades al construir la matriz de confusión obtiene los valores más altos. Por otro lado, en los conjuntos de datos Dermatology y Usps el método C-TDBU, que utiliza únicamente las probabilidades al realizar las predicciones obtiene los mejores valores de *accuracy*. Finalmente, solo para el conjunto de datos Zoo, el más pequeño de todos ya que solo contiene 101 instancias, el método C-TD obtiene los mejores resultados, sin embargo, este tiene el peor *accuracy* promedio.

| Datos | C-TD | C-TDBU | P-TD | P-TDBU |
|---|---|---|---|---|
| **Arrytmia** | 0.6000 | 0.5995 | 0.5936 | **0.6016** |
| **Covtype** | 0.6835 | 0.6856 | 0.6857 | **0.6869** |
| **Dermatology** | 0.9138 | **0.9146** | 0.9081 | 0.9109 |
| **Flare** | 0.7147 | 0.7179 | 0.7146 | **0.7187** |
| **Forest** | 0.8799 | 0.8806 | **0.8830** | 0.8800 |
| **Glass** | 0.6520 | 0.6587 | 0.6600 | **0.6624** |
| **Letters** | 0.7446 | 0.7652 | 0.8101 | **0.8178** |
| **Pendigits** | 0.9222 | 0.9190 | **0.9287** | 0.9271 |
| **SatImage** | 0.8566 | 0.8559 | **0.8568** | 0.8549 |
| **Segmentat** | 0.9258 | 0.9265 | 0.9275 | **0.9279** |
| **Sports** | 0.5934 | 0.5961 | 0.5922 | **0.5972** |
| **Trafficlight** | 0.6723 | 0.6773 | 0.6798 | **0.6852** |
| **Usps** | 0.8756 | **0.8758** | 0.8735 | 0.8749 |
| **Vertebral** | 0.8162 | 0.8180 | 0.8241 | **0.8250** |
| **Zoo** | **0.8971** | 0.8960 | 0.8894 | 0.8905 |

**Tabla 5.6:** Valores de *accuracy* promedio por conjunto de datos obtenidos por los métodos multiclase jerárquicos. Para cada método y conjunto de datos el mejor resultado está resaltado en negrita.

La tabla 5.7 muestra los resultados agregados por técnicas de clasificación. El promedio global afirma que el método P-TDBU es el que obtiene el mejor valor global, seguido por P-TD y C-TDBU, de la misma forma que al analizarlo por conjuntos de datos. Si tomamos en cuenta los valores de tasa de aciertos obtenidos, los métodos P-TDBU y C-TDBU, en promedio, obtienen los más altos resultados en cuatro y tres técnicas respectivamente del total de siete técnicas utilizadas en el estudio. De igual forma que en el capítulo 4, con el objetivo de reducir el sesgo en el análisis de resultados se calcularon los promedios de diferencias relativas de *accuracy* que confirman lo observado en las tablas promedio (tabla 5.8 ).

Para estimar el nivel de significatividad estadística entre las diferencias que existen entre los métodos propuestos se ha aplicado el test no paramétrico de Friedman (ver sección 2.5.2), tal y como se hizo en el capítulo 4. Esta prueba permite evaluar más de dos clasificadores sobre diferentes problemas de clasificación y conjuntos de datos. Dado que la prueba de Friedman evalúa si existen diferencias significativas entre el conjunto de métodos en general, para obtener un detalle aun mayor hemos incluido los resultados obtenidos al aplicar

| Técnica | C-TD | C-TDBU | P-TD | P-TDBU |
|---|---|---|---|---|
| **SVM** | 0.8265 | **0.8295** | 0.8236 | 0.8281 |
| **RPART** | 0.7807 | 0.7813 | 0.7872 | **0.7883** |
| **RF** | 0.8723 | **0.8741** | 0.8672 | 0.8717 |
| **NNET** | 0.7902 | 0.7995 | 0.8254 | **0.8266** |
| **NB** | 0.6030 | **0.6039** | 0.6021 | 0.6020 |
| **C50** | 0.8112 | 0.8117 | 0.8124 | **0.8128** |
| **KNN** | 0.7984 | 0.8003 | 0.8015 | **0.8055** |

**Tabla 5.7:** Valores de *accuracy* promedios por técnicas de clasificación obtenidos por los métodos multiclase jerárquicos. Para cada método y técnica, el mejor resultado está resaltado en negrita.

| Técnicas | C-TD | C-TDBU | P-TD | P-TDBU |
|---|---|---|---|---|
| **SVM** | 0.066 | **0.062** | 0.069 | 0.064 |
| **RPART** | 0.112 | 0.112 | 0.106 | **0.104** |
| **RF** | 0.009 | **0.007** | 0.015 | 0.010 |
| **NNET** | 0.101 | 0.091 | 0.064 | **0.062** |
| **NB** | 0.323 | **0.322** | 0.323 | 0.323 |
| **C50** | 0.079 | 0.079 | 0.078 | **0.077** |
| **KNN** | 0.097 | 0.095 | 0.094 | **0.089** |

**Tabla 5.8:** Valores de diferencias relativas por técnicas de clasificación obtenidos por los métodos multiclase jerárquicos. Para cada método y técnica, el mejor resultado está resaltado en negrita.

| Técnicas | Ranking |
|---|---|
| C-TD | 3.0667 |
| P-TD | 2.6 |
| C-TDBU | 2.5333 |
| P-TDBU | 1.8 |

**Tabla 5.9:** Ranking promedio de los algoritmos aplicando el test de Friedman. El valor más bajo es el mejor método mientras que el valor más alto es el peor método.

| Pares de Técnicas | | | Valores-$p$ |
|---|---|---|---|
| **C-TD** | **vs.** | **P-TDBU** | 0.043257 |
| **P-TD** | **vs.** | **P-TDBU** | 0.44843 |
| **C-TDBU** | **vs.** | **P-TDBU** | 0.47918 |
| **C-TD** | **vs.** | **C-TDBU** | 0.773697 |
| **C-TD** | **vs.** | **P-TD** | 0.773697 |
| **C-TDBU** | **vs.** | **P-TD** | 0.887537 |

**Tabla 5.10:** Comparativa entre pares de métodos de clasificación y sus valores-$p$ con un nivel de confianza del 95% aplicando el método de Holm.

el proceso posterior de Holm [Hol79], así podemos obtener una comparación entre pares de métodos.

La tabla 5.9 muestra el ranking de los métodos propuestos siguiendo el test de Friedman. Con un valor-$p$ obtenido a partir del cálculo del estadístico de Iman y Davenport de 0.055423, se han encontrado diferencias significativas entre los métodos propuestos. Concretamente los métodos que ocupan las primeras posiciones del listado son P-TDBU seguido de C-TDBU y P-TD, mientras que en última posición se encuentra C-TD. Similares resultados se obtuvieron al analizar las diferencias relativas.

La tabla 5.10 muestra los valores-$p$ con un nivel de confianza del 95%. Según estos resultados únicamente se han encontrado diferencias significativas entre los métodos C-TD y P-TDBU tanto al analizar los accuracy promedio y las diferencias relativas, lo cual corrobora la hipótesis planteada al principio del capítulo de que los métodos suaves podrían mejorar la calidad de la clasificación multiclase jerárquica en relación a la aproximación que solo usa clasificadores discretos. Resumiendo, construir la matriz de confusión usando estimaciones de probabilidad y aplicar los modelos siguiendo un procedimiento *Top-down & Bottom-up* permite obtener una mejora significativa en los resultados.

## 5.4 Resumen

En este capítulo nos centramos en el desarrollo de una variante del método de clasificación multiclase jerárquica presentado en el capítulo anterior y que utiliza estimaciones de probabilidad obtenidas al usar clasificadores suaves. Además, hemos introducido una modificación en la estrategia original de la

aplicación del modelo que combina las estimaciones de probabilidad de los clasificadores locales desde las hojas hasta la raíz del árbol.

Los experimentos realizados muestran la validez de nuestra propuesta ya que el uso de estimaciones de probabilidad permite mejorar la tasa de acierto en relación a los resultados obtenidos con el enfoque discreto original.

# Capítulo 6

# Adaptación del modelo jerárquico multiclase en entornos cambiantes

El entorno tradicional del aprendizaje supervisado supone que el contexto en el que tiene lugar el proceso de aprendizaje, desde el entrenamiento del modelo hasta su despliegue, no cambia. Esta concepción estacionaria contrasta con lo que sucede en escenarios más realistas, donde es habitual que el contexto de entrenamiento difiera del de despliegue. El reconocimiento de objetos y la venta minorista son casos ilustrativos de contextos de aprendizaje cambiantes. Así, es bastante común que, por ejemplo, un modelo de recomendación entrenado para sugerir un plan de inversiones automático en la bolsa tenga la necesidad de adaptarse con nuevos conocimientos al seguir de cerca los cambios en los mercados.

¿Cómo se puede caracterizar el contexto? En [Her+16], los autores presentan la siguiente definición general: el contexto es toda la información relacionada con los datos, tales como su distribución, representación, calidad, funciones de utilidad y tipo de tarea. Los autores introducen una taxonomía de los posibles cambios de contexto que incluye, entre otros, los cambios en la distribución

conjunta de entradas y salidas cuando pasamos del entrenamiento al test (que se denomina más comúnmente como «dataset shift»).

Otro cambio relevante en el contexto de un problema de clasificación se produce cuando el conjunto de posibles clases que hay que predecir varía por una novedad, es decir, cuando, en el conjunto de test, aparece una nueva clase que no estaba presente durante la etapa de entrenamiento. Se trata de un problema difícil que suele producirse en áreas como la visión por ordenador, la categorización de textos o el aprendizaje basado en flujos continuos de datos (*data streams*).

La principal consecuencia de la aparición de un nuevo concepto es la reducción de la calidad del modelo y que como resultado, la tasa de acierto se reduce porque las instancias del test que pertenecen a la nueva clase son mal clasificadas y, por lo tanto, es necesario aplicar alguna estrategia para adaptar el modelo al nuevo contexto sin que afecte su rendimiento. El aprendizaje incremental [CP01; Gir00; FHR01] es un paradigma de aprendizaje automático en el que los datos de entrada se utilizan continuamente para ajustar el conocimiento aprendido, es decir, para entrenar aún más el modelo. De acuerdo con este paradigma, el aprendizaje, cuando aparecen nuevas etiquetas de clase puede realizarse siguiendo los siguientes enfoques:

- reentrenar el modelo utilizando datos nuevos e históricos (si se dispone de ellos), proceso que puede también combinarse con ponderaciones para dar más importancia a los datos nuevos o con un preprocesamiento para hacer frente al desbalance entre las clases (ya que tendremos menos datos de la clase nueva al no estar presente en los datos históricos)
- modificar el modelo (si es posible) para abarcar también la nueva situación. Este último enfoque se denomina «reframing »[Her+16].

En una tarea de clasificación, el reentrenamiento implica descartar el clasificador existente y volver a entrenar uno nuevo. Como consecuencia de ello, los conocimientos adquiridos previamente se pierden. Este problema es conocido comúnmente como «olvido catastrófico», término mencionado en [MBB13].

Una complicación adicional es determinar la cantidad adecuada de datos antiguos y nuevos que se han de utilizar. Además, muchas veces no es posible utilizar los datos originales en el reentrenamiento porque se han perdido, corrompido o descartado por falta de recursos para ser almacenados. Otro problema que tiene el reentrenamiento es que puede ser costoso desde el punto de vista computacional (si el modelo requiere ser reentrenado muchas veces).

Como contrapartida, los métodos que adaptan los clasificadores existentes (como alternativa al reentrenamiento) son en un principio más eficientes, ya que no vuelven a generar el modelo completo. La complejidad del proceso de adaptación depende de la técnica de aprendizaje y de las características individuales del problema de aprendizaje que se ha de resolver. Una forma general de adaptar un modelo es determinar qué partes deben permanecer sin cambios (es decir, cómo reutilizar los viejos conocimientos) y qué partes deben actualizarse o añadirse (es decir, cómo incorporar las novedades).

Como se ha estudiado en los capítulos anteriores, un problema de clasificación multiclase grande y complejo puede descomponerse en subproblemas pequeños y simples mediante el uso de los métodos de clasificación jerárquica multiclase presentados en los capítulos 4 y 5. La modularidad que aporta la descomposición jerárquica hace que el clasificador jerárquico sea más fácil de actualizar. Concretamente, ante la aparición de una nueva clase, simplemente, se tendría que cambiar solo aquellas partes concretas del modelo jerárquico que se ven afectados por la nueva clase.

En este capítulo revisaremos los conceptos básicos de la clasificación incremental y, en particular nos centraremos en cómo aplicarlos a problemas en los que el conjunto de clases se amplía por la aparición de una nueva etiqueta de clase durante el despliegue del modelo. Adicionalmente, proponemos algunos métodos para adaptar los modelos de la clasificación jerárquicos multiclase, los cuales serán evaluados experimentalmente utilizando varios conjuntos de datos.

## 6.1 Aprendizaje incremental

En un escenario típico, el conjunto de datos de entrenamiento utilizado para aprender un clasificador está completo y disponible; sin embargo, hay situaciones en las que el conjunto de datos de entrenamiento se va proporcionando incrementalmente y, por lo tanto, el clasificador debe aprender constantemente de la nueva información. Cuando esto ocurre, decimos que estamos ante un problema de aprendizaje incremental (*Incremental Learning*, IL). El proceso de construcción del modelo se realiza en iteraciones sucesivas, es decir, cada vez que surgen nuevos ejemplos se actualiza el modelo.

Las estrategias típicas de aprendizaje no son aplicables a escenarios incrementales porque es común que las características de los datos de entrenamiento cambien a lo largo del tiempo. Por ejemplo, la nuevas instancias podrían tener diferentes atributos o podrían aparecer nuevas clases.

Según [Pol+01], un algoritmo de aprendizaje incremental debería cumplir los siguientes criterios:

- Debe ser capaz de aprender información adicional a partir de nuevos datos.
- No debe requerir el acceso a los datos originales utilizados para entrenar al clasificador existente.
- Debe preservar los conocimientos adquiridos previamente (es decir, no sufrir un olvido catastrófico).
- Debe ser capaz de acomodar nuevas clases que puedan introducirse con los nuevos datos.

Un aprendizaje incremental puede realizarse adaptando de forma secuencial el modelo, bien con cada una de la instancias nuevas a medida que aparecen, bien con los ejemplos de entrenamiento agrupados en paquetes o lotes de instancias y adaptando el modelo cada vez que un lote esté accesible. En el primer caso, el entrenamiento se realiza de forma continua, utilizando una instancia nueva a la vez y sin necesidad de almacenarla, por lo que este método incremental se conoce como aprendizaje en línea de una sola pasada (*One-pass online learning*). Por otro lado, cuando el aprendizaje incremental se realiza usando grupos de paquetes, recibe el nombre de aprendizaje incremental por lotes (*batch incremental learning*).

En este capítulo presentamos cómo adaptar nuestra propuesta jerárquica multiclase para realizar un aprendizaje incremental por lotes al detectar nuevas clases.

### *6.1.1 Aprendizaje incremental de clases*

El aprendizaje incremental de clases (*Class Incremental Learning*, CIL) tiene como objetivo adaptar el clasificador cuando aparecen nuevas clases durante la etapa de despliegue del modelo, de forma que este pueda ser usado para clasificar instancias tanto de la nueva clase como de las clases originales con las que fue entrenado el modelo.

La aparición de nuevas clases es un fenómeno que puede ocurrir en muchas circunstancias del mundo real, como por ejemplo: una nueva categoría de artículo en una revista, un nuevo tipo de imagen en una tarea de reconocimiento o un nuevo tipo de ciberataque detectado.

En los problemas CIL se distinguen dos fases:

- **La detección de nuevas clases emergentes**. Es el proceso de detectar la aparición de una nueva clase a través de la inspección de las nuevas instancias.
- **La adaptación del modelo**. Se refiere a la actualización de un modelo antiguo con la nueva información pero preservando el conocimiento previo.

## 6.2 Estrategias utilizadas en CIL

Existen varias formas de abordar un problema CIL. El método más popular se basa en la construcción de *ensembles* (ver sección 3.2) porque pueden ser fácilmente actualizados al añadir nuevos componentes. A continuación se describen algunos de los enfoques relevantes utilizados para resolver un problema CIL:

- Presentado por [MTP09], *Learn++.NC* es un algoritmo de aprendizaje de *ensembles* incremental, en el que cada clasificador se entrena con un subconjunto diferente de instancias. Cuando se observan instancias de nuevas clases, el *ensemble* aprende un nuevo clasificador y lo añade al conjunto. Luego, al realizar las predicciones, se define un mecanismo de votación ponderada que decide cuál o cuáles clasificadores del conjunto se utilizarán, así como el peso de cada predicción individual.
- Otro enfoque CIL basado en *ensembles* se presenta en [Dew+13]. El método incorpora un mecanismo de olvido utilizado para eliminar un modelo del *ensemble* cuando existen otros con mejor precisión.
- En [Cha16] se presenta otro método basado en crear un *ensemble* de clasificadores. Las nuevas clases son identificadas analizando los valores atípicos presentes en las instancias de entrenamiento. Una vez identificado un grupo de valores atípicos, se aplica un algoritmo de *clustering* para organizarlos por grupos de clases y, a continuación, se entrena un nuevo modelo por cada clase nueva identificada. Cabe destacar que los conjuntos de instancias utilizadas para el entrenamiento de cada modelo son almacenados, lo que permite su uso en el reentrenamiento de modelos desactualizados.
- En [DYZ14], los autores proponen una plataforma llamada *Learning with Augmented Class with Unlabel data* (LACU) que se basa en la idea de que

los datos no etiquetados pueden proporcionar información útil durante el entrenamiento para generar modelos que funcionen mejor para las clases vistas y no vistas. El método combina el principio de maximización de margen de la técnica SVM con el de la técnica de separación de baja densidad de los algoritmos semisupervisados.

- En [ZSX06], se propone un método CIL basado en una máquina de vectores soporte para mejorar la precisión de una tarea de categorización de texto. Dado un problema con $n$ clases, se construye $n-1$ clasificadores SVM binarios con el conjunto de instancias de entrenamiento. La estrategia utilizada para crear los clasificadores es OVA (ver sección 3.4.1). Cuando se detecta una clase nueva, se realiza un proceso de selección de atributos y luego se aprende un nuevo SVM binario que reconozca los nuevos ejemplos del resto de instancias, en este caso, las instancias negativas representan las clases conocidas, mientras que las positivas la clase nueva.

- En [Che+19], se presenta un método CIL híbrido para el reconocimiento de objetos colocados sobre la mano. Consiste en aprender automáticamente nuevos conceptos añadiendo hiperplanos de clasificación al modelo SVM multiclase. Teniendo en cuenta que un modelo SVM distingue entre dos clases, se construyen varios clasificadores de la misma forma que un SVM aplicado a problemas multiclase. Para realizar el aprendizaje incremental, se agrega un nuevo clasificador que distinga entre la nueva y todas las clases aprendidas (usando una estrategia OVA). Además, se ajustan los hiperplanos de los modelos antiguos de forma que se minimice el error de predicción.

- En [MTZ17] se presenta una estrategia para crear un *ensemble* basada en la clase (*Class Base Ensemble*, CBE). Este método entrena un conjunto de clasificadores binarios por cada clase individual del problema de clasificación. En un escenario incremental de clase, se adapta el modelo entrenando un clasificador por cada clase nueva, que luego se incluye en el conjunto. Se sigue una estrategia OVA para crear cada nuevo clasificador, lo que significa que las instancias de la nueva clase se toman como positivas y todas las demás instancias son instancias de entrenamiento negativas.

La mayoría de las aproximaciones presentadas usan solamente los datos nuevos para la actualización. Frecuentemente, se realiza la comparativa con estrategias de reentrenamiento, donde el modelo completo es descartado y se utilizan los nuevos datos para crear el modelo. Además, algunos de los enfoques men-

cionados asumen que las instancias de una nueva clase se identifican mediante mecanismos existentes y se centran en la forma de actualizar los modelos. En este estudio mantenemos este supuesto, con el objetivo de centrarnos en los posibles mecanismos de adaptación de los modelos jerárquicos multiclase.

## 6.3 Adaptación de clasificadores jerárquicos multiclase

Las propiedades de modularidad y especialización de sus componentes permiten que los clasificadores jerárquicos multiclase puedan adaptarse de forma rápida y sencilla para resolver problemas CIL. Como se presentó en los capítulos 4 y 5, la clasificación jerárquica multiclase explota las relaciones entre clases al descomponer jerárquicamente el problema en varios subproblemas de menor complejidad. En base a esto, la adaptación del modelo jerárquico ante la aparición de una nueva clase se centra en la modificación de algunos de los modelos componentes de la jerarquía:

Sea $D$ un conjunto de entrenamiento; $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$, el conjunto de clases; $H_{\mathcal{Y}}$ la jerarquía de clases inferida por un clasificador plano entrenado con $D$; y sea $M$, el modelo jerárquico multiclase generado aplicando el método HCM (capítulo 4). Y sea $u$ la clase emergente tal que $D$ no incluye instancias de la clase $u$. Durante la fase de despliegue, se dispone de un conjunto de instancias $\Upsilon$ en el que se ha identificado el grupo de instancias que son de la clase $u$. Dichas instancias se usarán para la actualización del modelo.

A continuación, presentamos varios algoritmos CIL específicamente diseñados para modificar incrementalmente el modelo $M$.

### *6.3.1 Aprendizaje incremental de clase basado en similitud (AIBS)*

Esta estrategia basa su mecanismo de adaptación en el grado de similitud entre la nueva clase y el conjunto de las clases originales. La idea es obtener en la jerarquía de clases la ubicación de la clase más parecida a la nueva. Con esta información cambiamos la jerarquía de clases y generamos un nuevo nodo interno que contiene como hijos estas dos clases. En consecuencia, el árbol de clasificadores es modificado con un nuevo clasificador local tal que la jerarquía resultante contendrá un camino que permite la predicción de la clase emergente.

La estrategia AIBS consta de los siguientes pasos:

- **Detección de la clase original más similar a la clase emergente.** Sea $u$ la nueva clase emergente detectada en el conjunto de datos $\Upsilon$ y sea $\Upsilon_u \subseteq \Upsilon$ las instancias que pertenecen a la nueva clase $u$. Como se mencionó anteriormente, asumimos que existe un mecanismo que detecta las instancias $\Upsilon_u$; por lo tanto, podemos proceder considerando que todas las instancias en $\Upsilon_u$ están etiquetadas.

  El modelo $M$ se aplica para predecir las instancias en $\Upsilon_u$. Lógicamente, $M$ clasifica de manera errónea todas las instancias en $\Upsilon_u$, ya que la clase $u$ no ha sido usada para entrenar $M$. El vector $\Phi_u = \langle \Phi_{u1}, \ldots, \Phi_{um} \rangle$ representa las predicciones de $M$ para $\Upsilon_u$; donde $\Phi_{ui}, 1 \leq i \leq m$ denota el número de instancias en $\Upsilon_u$ que se clasifican erróneamente como de la clase $y_i$ ($\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$).

  Por consiguiente, una manera de descubrir cuál es la clase en $\mathcal{Y}$ más similar a $u$ es examinar $\Phi_u$ buscando el componente de mayor valor, ya que cuanto más alto es el valor de $\Phi_{ui}$, más similares son las clases $u$ e $i$ de acuerdo con el modelo $M$. Entonces, la clase $s$ más similar a una clase emergente $u$ se define como $s = argmax_{y_i \in \mathcal{Y}}(\Phi_{ui})$.

- **Adaptación de la jerarquía de clases.** El siguiente paso consiste en adaptar la jerarquía de clases $H_{\mathcal{Y}}$ para incluir la clase objetivo $u$. Esto se hace convirtiendo la hoja de la clase $s$ en un nodo interno con dos hijos hoja: uno etiquetado con la clase $s$ y el otro con la clase $u$. Denominaremos a la jerarquía de clases adaptada $H^*_{\mathcal{Y}}$.

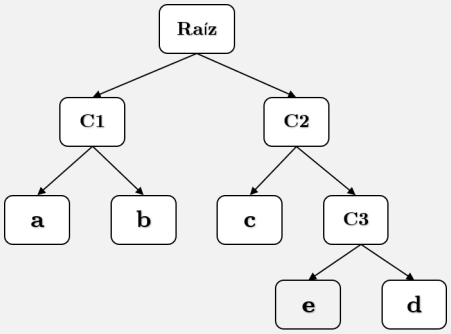
- **Adaptación de la jerarquía de clasificadores.** Una vez que la jerarquía de clases ha sido modificada, debemos actualizar correctamente el modelo jerárquico $M$ para que sea consistente con $H^*_{\mathcal{Y}}$. Esto se hace entrenando un clasificador binario correspondiente para el nodo interno añadido para crear $H^*_{\mathcal{Y}}$, y haciendo crecer localmente el árbol $M$ sustituyendo la hoja etiquetada por la clase $s$ (la clase original más parecida) por el clasificador recién creado. Como el nuevo clasificador binario $f_{us}$ tiene que distinguir entre las clases $u$ y $s$ debemos entrenarlo usando instancias solo de estas dos clases.

Obsérvese que, en el método propuesto de adaptación, reutilizamos la mayor parte de los conocimientos adquiridos previamente, debido a que sólo una pequeña parte de la jerarquía de clasificadores se cambia para cubrir la nueva clase.

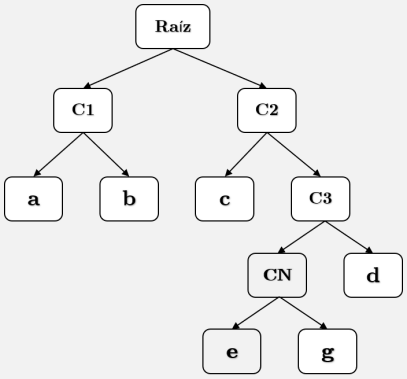
**Ejemplo 6.3.1: Aprendizaje incremental de clase basado en similitud (AIBS)**

*Consideremos un conjunto de datos con cinco clases* $\{a, b, c, d, e\}$ *que usamos para entrenar un modelo jerárquico multiclase* $M$*. Supongamos que se detecta una nueva clase* $g$ *en el conjunto de despliegue. El resultado de aplicar* $M$ *a las instancias de despliegue de la clase* $g$ *es* $\Phi_g = \langle 1, 2, 0, 14, \boldsymbol{22} \rangle$*.*

La clase original más parecida a $g$ es la clase $e$. La figura 6.1a muestra la jerarquía de clases inferida originalmente, con los lugares sombreados de la jerarquía donde se hará la actualización. Por otro lado, la jerarquía de clases actualizada se muestra en la figura 6.1b.




**(a)** Jerarquía de clases original $H_{\mathcal{Y}}$. La hoja sombreada es la clase original más parecida a la clase objetivo $g$.




**(b)** Jerarquía de clases adaptada $H^*_{\mathcal{Y}}$.

**Figura 6.1:** Ejemplo de la adaptación de la jerarquía de clases por la presencia de una nueva etiqueta de clase usando AIBS.

*Fin del ejemplo 6.3.1* ■

### *6.3.2 Aprendizaje incremental de clase basado primero en la clase emergente (CEMP)*

Este método sugiere que la clase nueva debe priorizarse sobre las clases aprendidas originalmente.

Las predicciones se realizan bajo un esquema de uno contra todos, tomando las instancias de la clase nueva como las instancias positivas y las otras como de la clase negativa.

Un clasificador jerárquico multiclase realiza las predicciones bajo una estrategia *Top-down*, la cual recorre por un solo camino toda la jerarquía de clasificadores desde la raíz hasta la hoja. El camino que sigue la instancia depende de las predicciones realizadas por los clasificadores ubicados en los distintos niveles de la jerarquía. Los modelos que están más altos dentro de la jerarquía de clasificadores son aplicados primero, y realizan una distinción entre clases internas que, a su vez, agrupan otras de menor nivel, mientras que los clasificadores cerca de las hojas realizan las predicciones sobre clases más específicas.

Dado que la raíz de la jerarquía de clases representa el grupo de todas las clases del problema de clasificación, para priorizar la nueva clase sobre las demás, se propone introducir un clasificador binario en la parte superior de la jerarquía que separe la clase emergente de las demás.

Para adoptar esta método, se siguen los siguientes pasos:

- Se modifica la jerarquía de clases insertando una nueva clase $CS$ sobre la clase raíz, de forma que contenga dos ramas; una que conecta la clase raíz de las jerarquía de clases previa y la otra, la clase nueva.
- Se entrena un nuevo clasificador binario $f_{CS}$ que distinga entre las clases aprendidas previamente y la nueva. A continuación, se inserta este como raíz del nuevo árbol de clasificadores.

Esta estrategia es conveniente para problemas de clasificación en los cuales las nuevas instancias son más importantes que las clases aprendidas previamente. Este es el caso, por ejemplo, de un sistema de identificación de ataques informáticos, en el que una nueva categoría de virus o ataque debería ser priorizado, porque, en este caso particular, los atacantes buscan nuevas estrategias que exploten vulnerabilidades no contempladas previamente.

**Ejemplo 6.3.2: Aprendizaje incremental de clase basado primero en la clase emergente**

*Consideremos el problema de clasificación presentado en el Ejemplo 6.3.1.*

La adaptación se realiza incluyendo dos clases dentro de la jerarquía, la primera, llamada $CS$, agrupa todas las clases del problema y divide la raíz del árbol entre la

clase nueva $g$ y las antiguas. La figura 6.2 muestra la jerarquía adaptada. Por lo tanto, se realiza el entrenamiento de un clasificador adicional $f_{CS}$, que distinga entre la clase nueva y las demás.

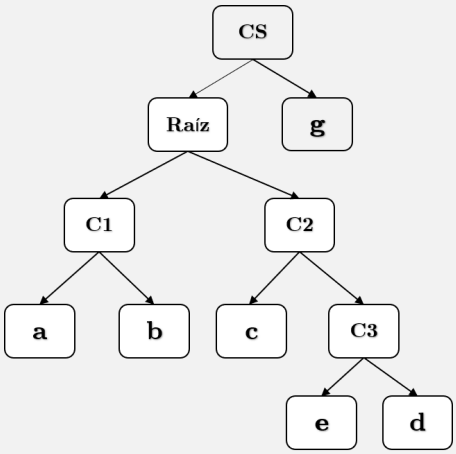



**Figura 6.2:** Ejemplo de la adaptación incremental de clase basada primero en la clase emergente. Jerarquía de clases adaptada $H^*_{\mathcal{Y}}$.

*Fin del ejemplo 6.3.2* ■

### *6.3.3 Proceso de adaptación jerárquica incremental (PAJI)*

Hasta ahora, hemos supuesto que, del conjunto de datos $\Upsilon$ usados en la actualización del modelo jerárquico, únicamente hemos identificado las instancias de las clases nuevas. Sin embargo, en un escenario de clasificación incremental, es posible que se conozcan las etiquetas de todas las instancias usadas para la actualización.

La estrategia PAJI permite la actualización incremental de todo el modelo de clasificación jerárquico multiclase a partir de un paquete de instancias $\Upsilon$ que incluye las clases originales y la nueva clase emergente. El procedimiento de adaptación se puede dividir en los siguientes pasos:

- Primero, se adapta la jerarquía de clases $H_{\mathcal{Y}}$ para incluir la nueva clase objetivo $u$. A continuación, se procede como en el método AIBS: se

convierte la hoja de la clase más similar $s$ en un nodo interno con dos hijos (la hoja de la clase $s$ y la hoja de la clase $u$), y el modelo de clasificación jerárquico $M$ se adapta al incorporar un nuevo clasificador que distingue entre $s$ y $u$.

- En segundo lugar, por cada nodo de la jerarquía, se entrena un nuevo clasificador $f_i^*$ utilizando el conjunto de instancias $\Upsilon$. De esta forma cada nodo de la jerarquía de clasificadores consta de dos clasificadores. Esta estrategia nos permite modificar el modelo y ser capaces de hacer frente a las instancias de la nueva clase mientras se mejora la potencia de clasificación de los clasificadores ya construidos.

La estrategia para el entrenamiento de cada uno de los clasificadores adicionales es la estrategia conocida como «política hermana» [FS07], que fue revisada en la sección 3.5.2.

Representamos el conjunto de las instancias de entrenamiento positivas y negativas de la clase $y_j$ como $\mathfrak{T}_r^+(y_j)$ y $\mathfrak{T}_r^-(y_j)$, $\Downarrow(y_j)$ como el conjunto de clases descendientes de $y_j$, $\leftrightarrow(y_j)$ el conjunto de clases con el mismo padre que $y_j$ y $*(y_j)$ son los ejemplos donde la clase más específica es $y_j$. Entonces, cada clasificador binario adicional se entrena con el siguiente conjunto de datos:

$$\mathfrak{T}_r^+(y_j) = *(y_j) \cup \Downarrow(y_j)$$
$$\mathfrak{T}_r^-(y_j) = \leftrightarrow(y_j) \cup \Downarrow(\leftrightarrow(y_j))$$

En otras palabras, las instancias positivas pertenecen a la clase $*(y_j)$ y todos sus descendientes $\Downarrow(y_j)$, mientras que las negativas son la unión de las instancias que son de clases hermanas con sus descendientes. Los clasificadores locales pueden entrenarse con cualquier algoritmo de clasificación dependiendo de cómo se ajusten a los requerimientos del problema de clasificación.

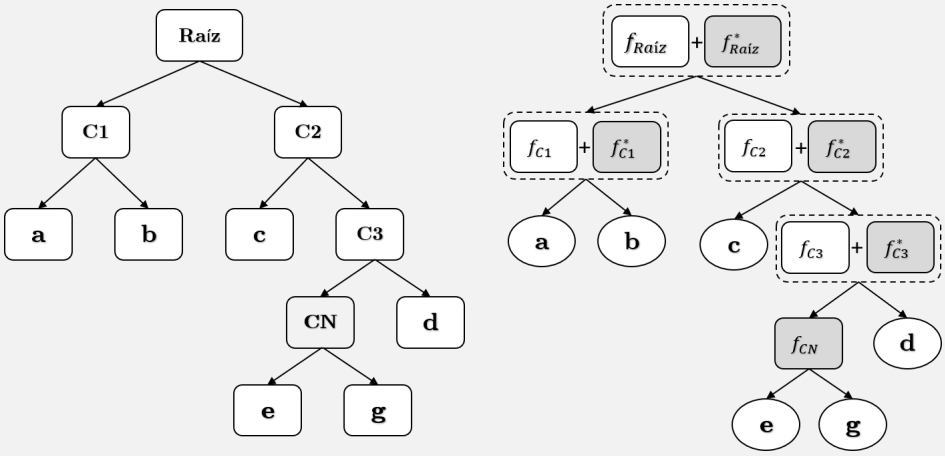
La aplicación del modelo de clasificación jerárquico adaptado incrementalmente sigue una estrategia *Top-down* similar a la que se aplica al usar modelos de clasificación jerárquica. El camino que recorre la instancia desde la cima del árbol hasta la hoja se basa en las predicciones realizadas por los clasificadores internos del árbol. En este caso, existen varios clasificadores por nodo y, por lo tanto, se debe realizar una combinación de las predicciones de estos clasificadores, la cual depende del tipo de clasificadores, discretos o suaves. Si la salida son estimaciones de probabilidad, los autores de [XKS92] plantean una forma sencilla de combinar los clasificadores conocida como *Average Bayes Classifier*. La combinación se realiza aplicando el promedio de las probabilidades obteni-

das por los clasificadores por cada clase del problema de clasificación. Al final, se obtiene una probabilidad única por clase y se asigna a la instancia que tiene la mayor probabilidad de pertenencia.

**Ejemplo 6.3.3: Proceso de adaptación jerárquica incremental**

*Para el problema de clasificación del ejemplo 6.3.1:*

Primero, se realiza el proceso completo de adaptación de la jerarquía de clases para incorporar la nueva clase (ver subsección 6.3.1). A continuación, se modifica el modelo jerárquico inicial de tal manera que los nodos internos queden compuestos por dos clasificadores: $M^* = \{(f_{Raíz}; f^*_{Raíz}), (f_{C1}; f^*_{C1}), (f_{C2}; f^*_{C2}), (f_{C3}, f^*_{C3}), f_{CN}\}$.




**(a)** Jerarquía de clases adaptadas $H_{\mathcal{Y}}$.

**(b)** Árbol de clasificadores jerárquicos adaptada $M^*$.

**Figura 6.3:** Ejemplo de la adaptación múltiple del modelo jerárquico multiclase.

La figura 6.3a, muestra los cambios necesarios para adaptar la jerarquía de clases de forma incremental, mientras que la figura 6.1b, muestra los cambios producidos en la jerarquía de clasificadores. Los clasificadores $f^*_i$ representan los nuevos clasificadores creados con las instancias $\Upsilon$.

*Fin del ejemplo 6.3.3* ■

En un entorno incremental, donde los datos son alimentados en forma de paquetes de instancias, la actualización de los clasificadores es parte de un proceso continuo. Por lo tanto, el número de clasificadores por nodo puede crecer indefinidamente, por lo que es conveniente implementar una estrategia que limite el número de clasificadores por nodo. Por ejemplo, se podría establecer un número máximo o un mecanismo de evaluación y descarte de los clasificadores.

## 6.4 Experimentos

En esta sección evaluamos experimentalmente las ventajas y desventajas que tienen los métodos CIL propuestos en la sección anterior.

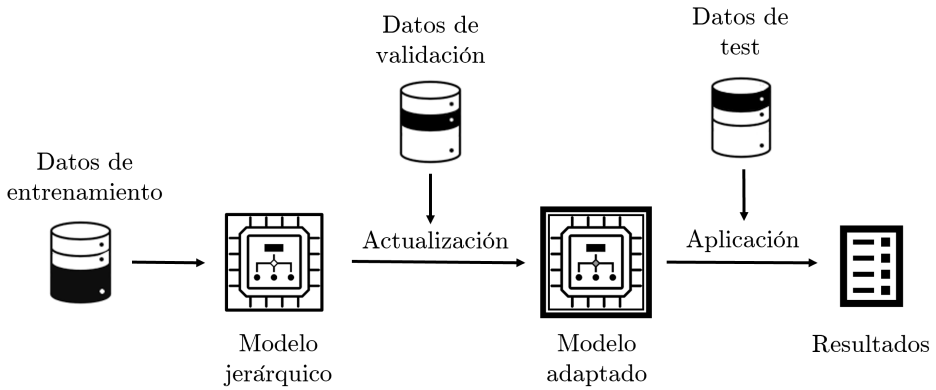



**Figura 6.4:** Proceso de experimentación.

El proceso de experimentación comienza con la creación de un modelo jerárquico multiclase usando los datos de entrenamiento ($D$), luego se realiza la actualización del modelo con los datos de validación ($\Upsilon$) y, finalmente, se evalúa el modelo utilizando los datos de test ($T$). El esquema del proceso completo se muestra en la figura 6.4.

Los métodos que se analizan son: el método de adaptación basado en similitud de clases (AIBS), el método de aprendizaje incremental de clase basado primero en la clase emergente (CEMP) y la estrategia de adaptación jerárquica incremental (PAJI). Además, con el fin de hacer una comparativa con los métodos tradicionales, se ha incluido el método CBE, ya que muchos de los métodos revisados en la Sección 6.2 lo usan como base para adaptar el modelo ante la aparición de una nueva clase [MTP09; ZSX06; Dew+13; Che+19]. Por

otro lado, al igual que nuestro método jerárquico multiclase, CBE no depende de la técnica de clasificación específica usada para el entrenamiento de sus modelos locales, lo que permite la comparación entre las distintas estrategias usando diversas técnicas de aprendizaje. Finalmente, se incluye también, como método de referencia, la aproximación basada en reentrenamiento (REE), en la que se descarta el modelo original y se entrena uno nuevo.

### *6.4.1 Escenarios de Entrenamiento*

Al realizar una actualización incremental del modelo de clasificación, el entrenamiento de los clasificadores nuevos se realiza en función de la disponibilidad de los datos de entrenamiento, lo cual puede establecer los siguientes escenarios:

- **Escenario 1**. Se mantiene un registro del historial de instancias usadas para el entrenamiento de los clasificadores.
- **Escenario 2**. No se conservan las instancias después del entrenamiento de los clasificadores.

El primer escenario considera posible el almacenamiento constante de los datos de entrenamiento, de forma que siempre estén disponibles al realizar el entrenamiento de los modelos futuros. Por el contrario, el segundo escenario limita la disponibilidad de los datos al momento de realizar el entrenamiento de los modelos. Este escenario es compatible con situaciones en las que no es posible un almacenamiento continuo de los datos históricos, como al realizar un entrenamiento *online*. En este estudio nos interesa conocer la capacidad de respuesta que tienen los métodos propuestos bajo esta restricción, a pesar de no utilizar enormes conjuntos de datos.

En el método AIBS, en el primer escenario, el clasificador nuevo se entrenaría con instancias de $D$ y $\Upsilon$ que pertenecen a la clase más similar y $\Upsilon_u$ para la clase nueva, mientras que, en el segundo escenario, únicamente se usarían las instancias $\Upsilon_u$ y $\Upsilon_s$. Nótese que en el segundo escenario es necesario conocer las etiquetas de clase de las instancias pertenecientes a la clase nueva y a la similar del conjunto de datos $\Upsilon$ usado para la adaptación.

Con el método CEMP, en el primer escenario, el clasificador binario se entrenaría con todas las instancias de $D$ y $\Upsilon - \Upsilon_u$ como ejemplos negativos y $\Upsilon_u$ como ejemplos positivos, mientras que, en el segundo escenario, se utilizan las instancias $\Upsilon_u$ como ejemplos positivos y $\Upsilon - \Upsilon_u$ como negativos.

En el método incremental PAJI, las instancias usadas para el entrenamiento de los clasificadores locales se seleccionan según las clases de sus nodos descendientes y la disponibilidad en los datos (tal y como se ha visto en la sección 3.5.2). En el caso del primer escenario, las instancias que se usan son todas las pertenecientes a $D$ y $\Upsilon$, mientras que, en el segundo escenario, solamente se usarían las instancias de $\Upsilon$.

| Id | Datos | NumInst | NumAtr | NumClases |
|---|---|---|---|---|
| **1** | Actinopterygii | 22369 | 15 | 15 |
| **2** | Dermatology | 358 | 34 | 6 |
| **3** | Diamonds | 53940 | 9 | 5 |
| **4** | Forest | 523 | 27 | 4 |
| **5** | Frogs | 7195 | 22 | 8 |
| **6** | Glass | 214 | 9 | 6 |
| **7** | Koppen | 5567 | 25 | 9 |
| **8** | Optdigits | 3823 | 64 | 10 |
| **9** | Pendigits | 7494 | 16 | 10 |
| **10** | RandomRBFGen | 50000 | 10 | 7 |
| **11** | RandomTreeGen | 50000 | 10 | 7 |
| **12** | Satimage | 1795 | 36 | 6 |
| **13** | Segmentation | 2310 | 18 | 7 |
| **14** | Sports | 8000 | 13 | 10 |
| **15** | Texture | 5500 | 40 | 11 |
| **16** | TrafficLight | 300 | 10 | 6 |
| **17** | Vehicle | 846 | 18 | 4 |
| **18** | Vertebral | 310 | 6 | 3 |
| **19** | Vowel | 990 | 13 | 11 |
| **20** | Zoo | 101 | 16 | 7 |

**Tabla 6.1:** Información sobre los conjuntos de datos utilizados en los experimentos de aprendizaje incremental. Se muestra el número de instancias, atributos y clases por cada conjunto de datos.

### 6.4.2 Configuración de los experimentos

La tabla 6.1 muestra los 20 conjuntos de datos usados en los experimentos, de los cuales muchos de ellos ya han sido presentados en la tabla 5.4. Los criterios de selección utilizados dan preferencia a conjuntos de datos con un alto número de instancias y de clases, ya que se evaluará un aprendizaje incremental y se simulará la aparición de una nueva clase. Además, se ha incluido un conjunto de datos sobre la categorización de peces llamado «Actinopterygii» [Par+19], un conjunto de datos sobre la clasificación de los diamantes llamado «Diamonds» y dos bases de datos generadas artificialmente [Bif+10]: «RandomRBFGen», y «RandomTreeGen». Estos conjuntos de datos se incluyeron porque nos interesa analizar el rendimiento de los métodos incrementales propuestos al utilizar conjuntos de datos que tengan una combinación de número de instancias y clases más compleja. En los apéndices A y B puede consultarse información adicional sobre el preprocesamiento de los conjuntos de datos y los parámetros de las técnicas de clasificación utilizados.

Para llevar a cabo los experimentos, dividimos el conjunto de datos en 50% para el entrenamiento ($D$), 25% para la validación ($\Upsilon$) y 25% para la prueba ($T$). Estas divisiones se muestran en la figura 6.5. Primero usamos $D$ para entrenar el modelo original HMC, después se usa $\Upsilon$ para aplicar las estrategias de adaptación y, finalmente, $T$ para la evaluación. Además, para poder simular la aparición de una nueva clase, en cada conjunto de datos eliminamos una clase de $D$, la cual pasa a ser la clase nueva. Repetimos este proceso hasta que todas las clases del conjunto de datos fueron usadas como emergentes.

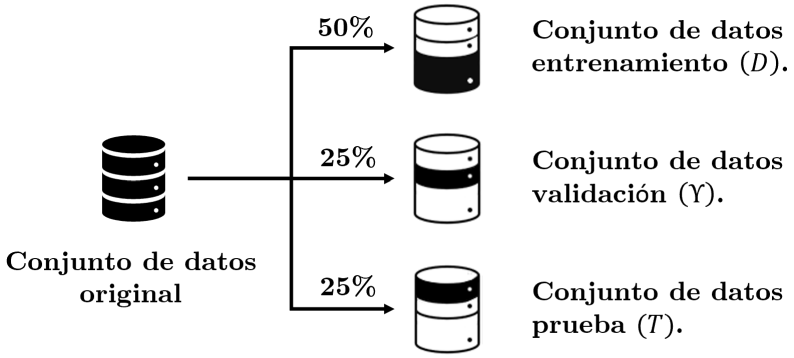



**Figura 6.5:** Partición del conjunto de datos.

Realizamos 10 repeticiones del proceso completo en cada uno de los escenarios planteados. Como medida de evaluación, usamos la tasa de acierto. Los resultados muestran el promedio de todas las ejecuciones.

En las siguientes secciones realizaremos un análisis comparativo de los resultados obtenidos en los experimentos realizados. En los primeros experimentos comparamos el método AIBS con el método de referencia REE (reentrenamiento) en ambos escenarios. A continuación, en la siguiente sección analizaremos todos los métodos CIL propuestos y el método de referencia CBE centrándonos en el segundo escenario.

### *6.4.3 Evaluación del método de adaptación basado en similitud de clases (AIBS) y reentrenamiento (REE)*

La tabla 6.2 muestra los resultados promedios obtenidos agrupados por conjunto de datos. Para estos experimentos se usaron las técnicas C50, KNN, NNET, RF, RPART y SVM. En el apéndice C, la tabla C.4 muestra los resultados detallados de estos experimentos, desglosados por técnica de clasificación y método de adaptación. Se utilizaron únicamente 15 de los conjuntos de datos presentados en la tabla 6.1 con el objetivo de reducir la duración de los experimentos, teniendo en cuenta que se evaluaban dos escenarios. Los conjuntos de datos que no hemos incluido en estos experimentos son aquellos en los que, considerando el número de instancias o el número de atributos resultan ser los más grandes (y, por tanto, complejos): Actipterygii, Diamonds, RandomRBF, RandomTree y el conjunto de datos Koppen.

Como puede observarse, en el escenario 1, REE obtiene mejores resultados de *accuracy* en todos los conjuntos de datos, mientras que, en el escenario 2, el método AIBS es capaz de obtener mejores resultados en 5 de los 15 conjuntos de datos. Igual que en el capítulo anterior, hemos calculado la diferencias relativas mostradas en la tabla 6.4 para confirmar que no hay sesgo en los resultados promedios que acabamos de comentar.

No obstante, si tenemos en cuenta el tiempo necesario para la adaptación del modelo HCM (tabla 6.3), el método REE requiere de mucho más tiempo (al tener que entrenar todo el modelo jerárquico de nuevo), llegando a requerir hasta 47 veces más tiempo que el método AIBS (véase en la tabla 6.3 la columna REE/AIBS que indica el ratio entre los tiempos empleados por ambos métodos). No obstante, esta diferencia de tiempo también depende de la técnica de clasificación utilizada para crear los clasificadores locales. Así, en el escenario

| Datos | Escenario 1 | | Escenario 2 | |
|---|---|---|---|---|
| | **AIBS** | **REE** | **AIBS** | **REE** |
| **Dermatology** | 0.863±0.061 | **0.921±0.030** | 0.864±0.062 | **0.890±0.031** |
| **Forest** | 0.825±0.056 | **0.865±0.032** | 0.830±0.059 | **0.859±0.028** |
| **Frogs** | 0.914±0.067 | **0.963±0.006** | 0.916±0.064 | **0.951±0.006** |
| **Glass** | 0.610±0.078 | **0.655±0.063** | **0.629±0.070** | 0.616±0.057 |
| **Optdigits** | 0.839±0.029 | **0.883±0.024** | 0.830±0.029 | **0.842±0.029** |
| **Pendigits** | 0.891±0.032 | **0.934±0.026** | 0.903±0.029 | **0.918±0.023** |
| **Satimage** | 0.843±0.028 | **0.873±0.013** | 0.846±0.027 | **0.863±0.010** |
| **Segmentation** | 0.918±0.028 | **0.953±0.012** | 0.918±0.028 | **0.935±0.010** |
| **Sports** | 0.589±0.013 | **0.611±0.011** | **0.595±0.013** | 0.594±0.012 |
| **Texture** | 0.924±0.018 | **0.957±0.008** | 0.924±0.019 | **0.940±0.007** |
| **TrafficLight** | 0.656±0.060 | **0.700±0.050** | **0.651±0.061** | 0.639±0.056 |
| **Vehicle** | 0.639±0.056 | **0.697±0.031** | 0.640±0.056 | **0.667±0.037** |
| **Vertebral** | 0.758±0.063 | **0.793±0.041** | 0.766±0.072 | **0.797±0.047** |
| **Vowel** | 0.745±0.033 | **0.824±0.029** | **0.738±0.032** | 0.654±0.032 |
| **Zoo** | 0.821±0.097 | **0.896±0.045** | **0.827±0.074** | 0.805±0.041 |

**Tabla 6.2:** Promedio de la tasa de acierto y desviación estándar obtenido por los métodos AIBS y REE. El mejor resultado para cada conjunto de datos se destaca en negrita. La segunda columna muestra los valores obtenidos en el primer escenario y la tercera en el segundo escenario.

1, RF obtuvo la diferencia más baja y SVM la más alta y, en el escenario 2, la más baja es KNN y la más alta es SVM.

Lógicamente, el escenario 1 es, en promedio, más lento que el escenario 2 ya que este primer escenario considera más datos de entrenamiento.

Con el objetivo de saber si las diferencias son estadísticamente significativas hemos aplicado la prueba de Wilcoxon (ver 2.5.1). Los valores de los rangos de orden para ambos escenarios se muestran en la tabla 6.5. Partimos de la hipótesis nula que sostiene que no existe una diferencia significativa entre los rendimientos de los métodos, mientras que la hipótesis alternativa sostiene que existe un método mejor que otro.

Cuando analizamos el escenario 1, la suma de los rangos de orden (de acuerdo con las ecuaciones 2.7 y 2.8) da como resultado $W_+ = 0$ y $W_- = 120$, siendo 0 el valor del estadístico de Wilcoxon, mientras que su valor crítico es $V = 25$.

| **Técnicas** | **Escenario 1** | | | **Escenario 2** | | |
|---|---|---|---|---|---|---|
| | **AIBS** | **REE** | **REE/AIBS** | **AIBS** | **REE** | **REE/AIBS** |
| **C50** | **0.048** | 0.613 | 12.775 | **0.022** | 0.232 | 10.434 |
| **KNN** | **1.741** | 26.781 | 15.379 | **1.350** | 8.658 | 6.414 |
| **NNET** | **14.970** | 224.582 | 15.002 | **10.297** | 91.294 | 8.866 |
| **RF** | **0.247** | 2.770 | 11.216 | **0.111** | 0.872 | 7.884 |
| **RPART** | **0.026** | 0.380 | 14.432 | **0.015** | 0.166 | 11.350 |
| **SVM** | **0.074** | 3.311 | 47.045 | **0.026** | 0.552 | 21.366 |

**Tabla 6.3:** Tiempo promedio en segundos necesarios para adaptar el modelo aplicando los enfoques de reframing basado en similitud de clases (AIBS) y de reentrenamiento (REE) para los dos escenarios. Los menores tiempos están marcados en negrita. En cada escenario se incluye el ratio entre el tiempo de REE y el de AIBS.

| **Técnicas** | **Escenario 1** | | **Escenario 2** | |
|---|---|---|---|---|
| | **AIBS** | **REE** | **AIBS** | **REE** |
| **C50** | 0.072 | **0.071** | **0.072** | 0.092 |
| **KNN** | 0.137 | **0.122** | **0.143** | 0.164 |
| **NNET** | 0.102 | **0.098** | **0.088** | 0.107 |
| **RF** | 0.008 | **0.003** | 0.006 | **0.003** |
| **RPART** | 0.121 | **0.117** | **0.115** | 0.141 |
| **SVM** | 0.096 | **0.082** | **0.102** | 0.122 |

**Tabla 6.4:** Promedio de las diferencias relativas por técnicas de clasificación aplicando los métodos AIBS y REE. El mejor resultado (más bajo) por cada conjunto de datos se muestra en negrita.

| Datos | Rangos de orden | |
|---|---|---|
| | **Escenario 1** | **Escenario 2** |
| **1** | -12,5 | -10 |
| **2** | -6 | -12 |
| **3** | -11 | -14 |
| **4** | -10 | 4 |
| **5** | -8,5 | -2,5 |
| **6** | -7 | -5 |
| **7** | -2 | -7,5 |
| **8** | -4,5 | -7,5 |
| **9** | -1 | 1 |
| **10** | -3 | -6 |
| **11** | -8,5 | 2,5 |
| **12** | -12,5 | -11 |
| **13** | -4,5 | -13 |
| **14** | -15 | 15 |
| **15** | -14 | 9 |

**Tabla 6.5:** Resultados de la Prueba de Signos con Rangos de Wilcoxon obtenido al evaluar los métodos AIBS y REE.

Por lo tanto, REE es mejor que AIBS en el primer escenario, ya que es posible rechazar la hipótesis nula. Cuando evaluamos las diferencias del escenario 2, encontramos que la suma de rango de orden da como resultado $W_+ = 31.5$ y $W_- = 88.5$. El valor del estadístico de Wilcoxon es 31.5 y, ya que el valor crítico es 25, entonces se acepta la hipótesis nula que afirma que no existen diferencias significativas entre los métodos AIBS y REE.

En consecuencia, en el segundo escenario (un escenario donde no se conservan los datos antiguos), el método de adaptación basado en la similitud de clases es capaz de obtener un mejor rendimiento que la aproximación por reentrenamiento, con un considerable ahorro en el tiempo requerido para la adaptación del modelo

### 6.4.4 Evaluación de los métodos CIL para la adaptación de modelos jerárquicos multiclase

Partiendo de estos resultados iniciales, en la segunda parte hemos ampliado los experimentos para abarcar todos los conjuntos de datos mostrados en la tabla 6.1. Además, en esta sección se pretende comparar los métodos de adaptación CIL propuestos: AIBS, PAJI y CEMP, y el método CIL de referencia que es CBE. No obstante, hemos incluido los resultados obtenidos por REE para extender la comparativa con este método tradicional.

Los resultados de esta segunda serie de experimentos se presentan en cuatro tablas: las tablas 6.6 y 6.7 muestran los valores de *accuracy* agrupados por dataset y por técnica de aprendizaje, respectivamente, y las tablas 6.8 y 6.9 muestran los tiempos requeridos para la adaptación del modelo HCM agrupados por dataset y por técnica de aprendizaje, respectivamente. Los resultados detallados de los experimentos por conjunto de datos, métodos de adaptación y técnica de clasificación se han incluido en el apéndice C (tablas C.5 y C.6). Para comprobar que no se han producido sesgos al calcular los valores promedios de *accuracy* y tiempo, se han incluido en el apéndice C las tablas C.7 y C.8 con los valores de las correspondientes diferencias relativas por técnicas de clasificación. En estos experimentos solamente utilizamos las técnicas C50, RF, RPART y SVM, ya que NNET y KNN requerían largos tiempos para su procesamiento. Decidimos centrarnos en el escenario donde los datos históricos de entrenamiento no estaban disponibles (escenario 2).

A pesar de que el método REE obtenía los mejores resultados en la sección anterior, los resultados de estos experimentos muestran que el método CEMP es capaz de obtener los mejores valores de *accuracy* en casi todos los conjuntos de datos (superando al método REE y al resto de métodos CIL), excepto en los conjuntos de datos Sports, TrafficLight y Vowel en los que las diferencias con el método PAJI son pequeñas. Respecto al método de referencia CBE, se observa que obtiene los resultados de *accuracy* más bajos en casi todos los conjuntos de datos, esto es, sus resultados son peores que los de los métodos PAJI y CEMP.

En cuanto al análisis de los tiempos de adaptación, el enfoque AIBS es el más rápido, pues a ser desde 6 hasta 54 veces más rápido que el método REE. Le siguen en rapidez los métodos CEMP y PAJI, mientras que REE es el más lento de todos los métodos evaluados. Sin embargo, las diferencias de tiempo dependen del algoritmo de clasificación utilizado para crear los clasificadores locales. En particular, RF obtuvo el mejor rendimiento con tiempos de procesamiento razonablemente bajos, seguido por el método SVM.

| Datos | Tasa de aciertos | | | | |
|---|---|---|---|---|---|
| | **REE** | **CBE** | **AIBS** | **PAJI** | **CEMP** |
| **Actinopterygii** | 0.616 | *0.522* | 0.624 | 0.637 | **0.641** |
| **Dermatology** | 0.920 | *0.881* | 0.893 | 0.928 | **0.931** |
| **Diamonds** | 0.741 | *0.672* | 0.725 | 0.736 | **0.747** |
| **Forest** | 0.856 | *0.767* | 0.827 | **0.859** | 0.858 |
| **Frogs** | 0.942 | 0.925 | *0.910* | 0.943 | **0.951** |
| **Glass** | 0.621 | *0.558* | 0.633 | 0.650 | **0.650** |
| **Koppen** | 0.903 | *0.855* | 0.889 | 0.908 | **0.914** |
| **Optdigits** | 0.806 | 0.809 | *0.800* | 0.844 | **0.852** |
| **Pendigits** | 0.937 | 0.924 | *0.912* | 0.940 | **0.948** |
| **RandomRBFGen** | 0.844 | 0.795 | *0.789* | 0.841 | **0.850** |
| **RandomTreeGen** | 0.816 | *0.767* | 0.777 | 0.811 | **0.840** |
| **Satimage** | 0.856 | *0.812* | 0.841 | 0.864 | **0.866** |
| **Segmentation** | 0.938 | *0.889* | 0.924 | 0.942 | **0.946** |
| **Sports** | 0.599 | *0.481* | 0.599 | **0.615** | 0.611 |
| **TrafficLight** | 0.659 | *0.546* | 0.678 | **0.697** | 0.688 |
| **Texture** | 0.922 | *0.893* | 0.909 | 0.930 | **0.934** |
| **Vehicle** | 0.675 | *0.651* | 0.652 | 0.693 | **0.705** |
| **Vertebral** | 0.801 | *0.755* | 0.775 | 0.798 | **0.803** |
| **Vowel** | 0.681 | *0.600* | 0.748 | **0.764** | 0.761 |
| **Zoo** | *0.800* | 0.856 | 0.826 | 0.850 | **0.872** |

**Tabla 6.6:** Promedio de la tasa de acierto por conjunto de datos aplicando los métodos CBE, AIBS, REE, PAJI y CEMP. El mejor resultado (más alto) por cada conjunto de datos se muestra en negrita y el peor resultado (más bajo) se destaca en cursiva.

| Técnicas | Tasa de aciertos | | | | |
|---|---|---|---|---|---|
| | **REE** | **CBE** | **AIBS** | **PAJI** | **CEMP** |
| **C50** | 0.794 | *0.754* | 0.787 | 0.808 | **0.816** |
| **RF** | 0.861 | *0.801* | 0.838 | 0.873 | **0.876** |
| **RPART** | 0.742 | *0.702* | 0.740 | 0.759 | **0.763** |
| **SVM** | 0.789 | *0.734* | 0.781 | 0.810 | **0.818** |

**Tabla 6.7:** Promedio de tasa de acierto de los métodos CBE, AIBS, REE, PAJI y CEMP agrupados por técnica de aprendizaje usada para entrenar los clasificadores locales. El mejor resultado (más alto) por cada conjunto de datos se destaca en negrita y el peor resultado (más bajo) se destaca en cursiva.

| Datos | Tiempo | | | | | $\frac{REE}{AIBS}$ |
|---|---|---|---|---|---|---|
| | **REE** | **CBE** | **AIBS** | **PAJI** | **CEMP** | |
| **Actinopterygii** | *9.021* | 2.222 | **0.166** | 5.231 | 1.215 | 54.3 |
| **Dermatology** | *0.181* | 0.021 | **0.018** | 0.081 | 0.033 | 10.4 |
| **Diamonds** | *39.535* | 17.172 | **5.909** | 25.626 | 13.789 | 6.7 |
| **Forest** | *0.218* | 0.045 | **0.025** | 0.094 | 0.044 | 8.7 |
| **Frogs** | *2.053* | 0.394 | **0.106** | 1.122 | 0.502 | 19.5 |
| **Glass** | *0.115* | 0.009 | **0.008** | 0.046 | 0.013 | 14.9 |
| **Koppen** | *0.889* | 0.217 | **0.053** | 0.483 | 0.199 | 16.9 |
| **Optdigits** | *3.484* | 0.705 | **0.121** | 1.917 | 0.600 | 28.8 |
| **Pendigits** | *1.923* | 0.244 | **0.069** | 0.867 | 0.213 | 28.1 |
| **RandomRBFGen** | *22.079* | 5.998 | **1.401** | 13.637 | 6.154 | 15.8 |
| **RandomTreeGen** | *34.656* | 13.664 | **1.594** | 22.709 | 6.927 | 21.7 |
| **Satimage** | *0.749* | 0.182 | **0.056** | 0.352 | 0.165 | 13.5 |
| **Segmentation** | *0.706* | 0.124 | **0.042** | 0.252 | 0.068 | 16.9 |
| **Sports** | *1.525* | 0.386 | **0.061** | 0.777 | 0.358 | 24.9 |
| **TrafficLight** | *0.138* | 0.019 | **0.009** | 0.057 | 0.016 | 14.9 |
| **Texture** | *1.473* | 0.308 | **0.050** | 0.691 | 0.305 | 29.5 |
| **Vehicle** | *0.234* | 0.048 | **0.038** | 0.125 | 0.063 | 6.2 |
| **Vertebral** | *0.074* | 0.014 | **0.011** | 0.030 | 0.014 | 7.0 |
| **Vowel** | *0.343* | 0.032 | **0.008** | 0.098 | 0.031 | 45.7 |
| **Zoo** | *0.109* | **0.006** | 0.007 | 0.036 | 0.010 | 15.0 |

**Tabla 6.8:** Promedio del tiempo en segundos requerido por los método CBE, AIBS, REE, PAJI y CEMP por conjunto de datos. El mejor resultado de tiempo (más rápido) se destaca en negrita y el peor resultado (más lento) se destaca en cursiva.

| Técnicas | Tiempo | | | | | $\frac{REE}{AIBS}$ |
|---|---|---|---|---|---|---|
| | **REE** | **CBE** | **AIBS** | **PAJI** | **CEMP** | |
| **C50** | *1.190* | 0.222 | **0.064** | 0.637 | 0.187 | 18.5 |
| **RF** | *6.993* | 2.029 | **0.628** | 3.670 | 1.898 | 11.1 |
| **RPART** | *0.657* | 0.111 | **0.032** | 0.326 | 0.093 | 20.6 |
| **SVM** | *15.060* | 5.999 | **1.226** | 10.213 | 3.966 | 12.3 |

**Tabla 6.9:** Promedio del tiempo en segundos requerido por los métodos CBE, AIBS, REE, PAJI y CEMP. agrupado por técnica de aprendizaje usada para entrenar los clasificadores locales. El mejor resultado de tiempo (más rápido) se destaca en negrita y el peor resultado (más lento) se destaca en cursiva.

Es destacable que nuestro método AIBS (la propuesta CIL más básica y sencilla), aun no siendo la mejor alternativa desde el punto de vista de la tasa de acierto, llega a alcanzar un buen compromiso entre rendimiento (en términos de *accuracy*) y tiempo de adaptación.

| Método | Ranking |
|---|---|
| **CBE** | 4.6 |
| **AIBS** | 3.95 |
| **REE** | 3.2 |
| **PAJI** | 2.05 |
| **CEMP** | 1.2 |

**Tabla 6.10:** Rankings promedio obtenido de los métodos de adaptación. Un valor más bajo corresponde a una mejor posición en el listado.

| Hipótesis | Valor-$p$ |
|---|---|
| **CBE vs .CEMP** | 0 |
| **AIBS vs .CEMP** | 0 |
| **CBE vs .PAJI** | 0.000003 |
| **REE vs .CEMP** | 0.000443 |
| **AIBS vs .PAJI** | 0.000868 |
| **CBE vs .REE** | 0.025551 |
| **REE vs .PAJI** | 0.085793 |
| **PAJI vs .CEMP** | 0.267393 |
| **AIBS vs .REE** | 0.267393 |
| **CBE vs .AIBS** | 0.267393 |

**Tabla 6.11:** Valores-$p$ ajustados obtenidos aplicando el método de comparación múltiple de Friedman y en el proceso Post Hoc de Holm. Los valores-$p$ con un valor menor a 0.05 rechazan la hipótesis nula y confirma que hay una diferencia estadísticamente significativa entre los pares de métodos.

Hemos realizado los test estadísticos para comprobar la significatividad de los resultados. La tabla 6.10 muestra las rangos de orden medios obtenidos mediante la aplicación del test de Friedman agrupado por conjunto de datos y método de adaptación, dando un valor $p$ de $5.594E-11$. De acuerdo con los resultados, el mejor método es CEMP, seguido por PAJI y REE, los cuales se ubican en las tres primeras posiciones.

Por otro lado, la tabla 6.11, muestra los valores-$p$ calculados con un nivel del confianza del 95% al realizar el procedimiento posterior de Holm para la comparación múltiple entre pares de métodos. Estos resultados destacan al

método CEMP como el que obtiene los mejores resultados, con diferencias significativas en comparación con los otros métodos, excepto con el método PAJI, donde no se encontraron diferencias significativas.

## 6.5 Resumen

En este capítulo hemos propuesto y evaluado tres estrategias diferentes para realizar la adaptación de los clasificadores jerárquicos multiclase ante la aparición de una clase emergente.

En primer lugar se ha propuesto el método llamado adaptación incremental basado en la similitud (AIBS) con la idea de hacer frente al problema de aprendizaje de una nueva clase. AIBS consiste en modificar la jerarquía en la ubicación del nodo más similar a la nueva clase. Este cambio requiere el entrenamiento de un nuevo clasificador para adaptar el modelo. La ventaja de este método es que requiere un cambio mínimo en la jerarquía de clases y, por lo tanto en el modelo, lo que permite se rápida adaptación.

El segundo método propuesto es CEMP, el cual modifica la raíz de la jerarquía de clases. Requiere la creación de un clasificador nuevo que distinga entre las clases aprendidas previamente y la nueva. La ventaja de este procedimiento es su simplicidad, ya que solamente requiere construir un clasificador que se ubicará en la parte superior del árbol de clasificación. Otra característica de este método es la importancia que otorga a la nueva clase sobre las previamente aprendidas.

El tercer método propuesto es el método de adaptación incremental PAJI que se centra en realizar una adaptación completa del modelo, para lo cual se entrenan continuamente todos los clasificadores del árbol que forman el modelo jerárquico. La ventaja de este procedimiento es que realiza una adaptación incremental completa de todos los nodos de la jerarquía, así como del árbol de clasificadores. La desventaja de este método es el tiempo que requiere para realizar la adaptación en cada paso del proceso.

Según los resultados obtenidos, los tres métodos son capaces de actualizar el modelo de clasificación preservando los conocimientos anteriores e incorporando la capacidad de etiquetar las nuevas clases emergentes. Se destaca el método CEMP por su capacidad de obtener la mejor tasa de aciertos seguido por el método PAJI, que tiene un rendimiento similar pero realiza una adaptación más lenta. El método AIBS es el más rápido de todos; a pesar de no obtener los mejores resultados de *accuracy*, tiene un buen desempeño. Por otro lado,

los métodos usados como referencia no destacan, ya que REE es el más lento y CBE es el que obtiene los peores resultados de tasa de aciertos.

A pesar de que hemos estudiado y evaluado la adaptación incremental de clasificadores jerárquicos multiclase, la aplicación de estos en otros escenarios como son el aprendizaje de flujos de datos continuos (data streams) es un área que aún necesita mayor investigación y desarrollo. Sin embargo, los resultados obtenidos arrojan luz sobre cómo podrían aplicarse estos métodos en este tipo de escenarios.

# Capítulo 7

# Conclusiones generales

Los métodos de clasificación jerárquicos presentados en este trabajo de investigación explotan las relaciones entre clases con el objetivo de mejorar la precisión en problemas de clasificación multiclase. En dominios jerárquicos, la clasificación jerárquica se ha aplicado exitosamente y ha logrado una mejora en la calidad de la clasificación al explotar el orden jerárquico preestablecido en el dominio. Este éxito nos motivó a plantearnos la posibilidad de aplicar estas estrategias en dominios no jerárquicos. La idea es que en situaciones donde existe un gran número de clases, los métodos tradicionales encuentran difícil discernir correctamente las nuevas observaciones dada la gran cantidad de posibilidades. La propuesta desarrollada en esta tesis consiste en construir clasificadores especializados para las clases que presentan más errores, es decir, construir una cadena de clasificadores especializados para problemas más simples.

Para ello, nos planteamos como primer objetivo general usar las predicciones dadas por un clasificador plano para explorar las relaciones entre clases. De esta forma, hemos definido una noción de similitud (o, alternativamente, de distancia) entre clases a partir de la matriz de confusión del clasificador plano. Esta función de distancia es usada para inferir la jerarquía de clases aplicando un algoritmo de agrupamiento jerárquico aglomerativo.

El segundo objetivo general que nos planteamos fue el desarrollo de un clasificador jerárquico para resolver problemas multiclase (en dominios no jerárquicos) usando la jerarquía de clases inferida. Hemos desarrollado dos versiones de nuestra propuesta, una que usa clasificadores discretos y otra que usa clasificadores suaves.

Finalmente, para el tercer objetivo planteado hemos desarrollado, analizado y evaluado varias estrategias para la adaptación de los clasificadores jerárquicos multiclase en situaciones donde las clases del problema son aprendidas de forma incremental.

A continuación, se describen los principales hallazgos encontrados y contribuciones realizadas en esta tesis.

- Definición de una semimétrica para calcular la distancia entre clases a partir de una matriz de confusión. Dicha semimétrica se basa en la idea de que, para determinar el nivel de confusión entre clases para un clasificador plano, se deben considerar los errores que el clasificador comete al clasificar instancias de una clase como pertenecientes a otra y viceversa.

- El modelo jerárquico multiclase (HMC) se obtiene entrenando un clasificador local por cada nodo interno de la jerarquía de clases. La evaluación experimental realizada muestra que la nueva técnica propuesta es capaz de mejorar la precisión de la clasificación con respecto al enfoque plano básico y a otras aproximaciones para construir la jerarquía de clasificadores propuestas en la literatura.

- También hemos presentado un método para comprimir la jerarquía de clases que ayuda a reducir la complejidad del modelo jerárquico derivado, ya que reduce el número de clasificadores locales que lo componen. Hemos demostrado experimentalmente que niveles altos de poda reducen e incluso empeoran la clasificación, ya que la fusión de nodos hace que clases en un principio diferentes se comporten como indistinguibles.

- Hemos propuesto una forma alternativa de generar la matriz de confusión usando las probabilidades de pertenencia a una clase estimadas por un clasificador suave.

- Asimismo, hemos definido un método alternativo de aplicar el clasificador jerárquico multiclase para el caso en el que los clasificadores locales son estimadores de probabilidad. Este método se basa en combinar probabilísticamente las clases en las hojas de la jerarquía, propagando estas probabilidades hasta alcanzar la raíz de esta.

- Hemos definido tres métodos para adaptar el modelo jerárquico multiclase ante la aparición en la etapa de despliegue del modelo de nuevas etiquetas de clase. Todos ellos se basan en usar la estructura jerárquica del modelo así como la noción de similitud entre clases. Esta afinidad nos permite determinar fácilmente qué partes de la jerarquía deben permanecer sin cambios y qué partes deben actualizarse para tratar con la clase nueva. De esta manera, podemos mantener la mayor parte del modelo original y, al mismo tiempo, podemos incorporar los conocimientos recién adquiridos de la nueva clase. El primer método reemplaza la hoja del árbol cuya clase es la más similar a la nueva por un clasificador local que distingue entre ambas clases. El segundo método añade una nueva raíz en la estructura jerárquica del modelo capaz de distinguir entre la nueva clase y las clases originales. El tercer método añade un clasificador nuevo por cada nodo interno de la jerarquía.

- Hemos analizado experimentalmente los tres métodos adaptativos en dos escenarios: (1) se dispone de los datos de entrenamiento inicial y (2) sólo se dispone de los datos del despliegue. Los resultados muestran que nuestras propuestas para adaptar el modelo jerárquico pueden obtener un buen rendimiento con menos tiempo de cálculo que la alternativa de descartar el modelo original y reentrenarlo desde cero, especialmente en el segundo escenario.

Los métodos de clasificación jerárquica multiclase pueden ser una herramienta para hacer frente a muchos de los desafíos que persisten en el área del aprendizaje automático. En particular, los métodos presentados en esta investigación podrían ser extendidos a problemas de clasificación multietiqueta donde se puede asignar varias clases a una instancia, o ser modificados para trabajar con flujos de datos o *stream learning*. También sería interesante investigar más detalladamente la relación entre el número de clases y el rendimiento obtenido por nuestro clasificador jerárquico. Otro aspecto que se podría estudiar es el efecto del desequilibrio entre clases en el rendimiento de nuestro método.

# Apéndices

## Apéndice A

# Conjuntos de datos

En esta sección se presentan las características y los pasos de preprocesamiento que hemos aplicado a cada uno de los conjuntos de datos o datasets utilizados en este trabajo de investigación. Todos los conjuntos de datos son multivariantes y han sido utilizados para realizar tareas de clasificación.

## A.1 Conjunto de datos Arrhytmia

**Objetivo de la tarea de clasificación**:

El objetivo era distinguir entre la presencia o ausencia de arritmia cardíaca y clasificarla en uno de los 7 tipos.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 452 |
| **Nro. de Atributos:** | 279 |
| **Tipos de Datos:** | Categóricos, integrales y reales. |
| **Valores Nulos:** | Sí |

**Preprocesamiento:** Para procesar este conjunto de datos, seguimos los siguientes pasos:

- Reemplazamos los valores ausentes con promedios de las columnas V11, V12, V13 y V15.
- Quitamos la variable V14 por que tiene casi todos los valores perdidos.
- Quitamos las columnas que tienen una desviación estándar de cero, ya que no aportan información a la predicción.
- Creamos columnas *dummies* para representar valores categóricos.
- Descartamos las columnas con valores repetidos V275, 265, 157, 152, 146, 144, 142, 140, 133, 132, 84, 70, 68,84.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 416
- Nro. de Atributos: 330
- Nro. de Clases: 7
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 4% y 59%

## A.2 Conjunto de datos Covtype

**Objetivo de la tarea de clasificación**:

El objetivo era predecir el tipo de zona forestal a partir de datos cartográficos.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 581012 |
| **Nro. de Atributos:** | 54 |
| **Tipos de Datos:** | Categóricos, integrales. |
| **Valores Nulos:** | No |

**Preprocesamiento:**
Para procesar este conjunto de datos, seguimos los siguientes pasos:

- Seleccionamos aleatoriamente 300 instancias de cada clase.
- Descartamos las columnas con valores repetidos V21, V22, V23, V29, V39, V50, V42.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 2100
- Nro. de Atributos: 54
- Nro. de Clases: 7
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 14% y 14%.

## A.3 Conjunto de datos Dermatology

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era distinguir entre los diferentes tipos de enfermedades eritematoescamosas. Los valores de las instancias han sido determinadas mediante un análisis de las muestras al microscopio.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 366 |
| **Nro. de Atributos:** | 32 |
| **Tipos de Datos:** | Categóricos, integrales. |
| **Valores Nulos:** | Sí |

**Preprocesamiento:**

- Descartamos los registros que tenían datos incompletos.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 358
- Nro. de Atributos: 34
- Nro. de Clases: 6
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 6% y 31%.

## A.4 Conjunto de datos Flare

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir la aparición de erupciones solares a partir de información de la últimas 24 horas.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 1389 |
| **Nro. de Atributos:** | 10 |
| **Tipos de Datos:** | Categóricos |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Tomamos como clase objetivo la columna V1 (clase Zurich modificada).

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 1066
- Nro. de Atributos: 19
- Nro. de Clases: 6
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 4% y 31%.

## A.5 Conjunto de datos Forest

**Objetivo de la tarea de clasificación**:

El objetivo era predecir el tipo de bosque usando imágenes espectrales obtenidas a partir de fotografías satelitales. Este dataset contiene datos de un estudio de teledetección que cartografió diferentes tipos de bosques.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 326 |
| **Nro. de Atributos:** | 27 |
| **Tipos de Datos:** | Enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:** Realizamos los siguientes pasos:

- Unificamos los conjuntos de datos de entrenamiento y de prueba en un dataset de entrenamiento.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 523
- Nro. de Atributos: 27
- Nro. de Clases: 4
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 16% y 30%.

## A.6 Conjunto de datos Glass

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era determinar el tipo de vidrio, basándonos en los datos de una investigación de criminología.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 214 |
| **Nro. de Atributos:** | 10 |
| **Tipos de Datos:** | Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Descartamos la columna V1 por ser un identificador de la instancia que no es útil para realizar predicciones.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 214
- Nro. de Atributos: 9
- Nro. de Clases: 6
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 4% y 33%.

## A.7 Conjunto de datos Letters

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era identificar las letras escritas a mano a partir de datos obtenidos de imágenes digitales.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 20000 |
| **Nro. de Atributos:** | 16 |
| **Tipos de Datos:** | Entero |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Realizamos un submuestreo aleatorio de un total de 100 instancias por cada clase.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 2600
- Nro. de Atributos: 16
- Nro. de Clases: 26
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 4% y 4%.

## A.8 Conjunto de datos Nuclear

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir la ubicación de las proteínas en células eucariotas.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 1484 |
| **Nro. de Atributos:** | 8 |
| **Tipos de Datos:** | Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:** Los pasos realizados fueron los siguientes:

- Descartamos la columna V1 por que refiere al identificador del ratón.
- Eliminamos los registros que contenían datos incompletos.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 552
- Nro. de Atributos: 80
- Nro. de Clases: 8
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 8% y 16%.

## A.9 Conjunto de datos Pendigits

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era reconocer los dígitos escritos por 30 escritores sobre una *tablet* con una resolución de 50 x 50 pixeles.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 10992 |
| **Nro. de Atributos:** | 16 |
| **Tipos de Datos:** | Enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Utilizamos el conjunto de datos de entrenamiento.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 7494
- Nro. de Atributos: 16
- Nro. de Clases: 10
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 10% y 10%.

## A.10 Conjunto de datos Reuters

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era asignar una categoría correcta a una nueva noticia utilizando el conjunto de los documentos que aparecieron en el cable de noticias Reuters en 1987.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 7769 |
| **Nro. de Atributos:** | 5 |
| **Tipos de Datos:** | Categóricos |
| **Valores Nulos:** | No |

**Preprocesamiento:** Los pasos dados fueron los siguientes:

- Transformamos todo el texto en minúsculas.
- Quitamos puntuaciones, números, *stopwords* y espacios en blanco.
- Aplicamos el método de ponderación Tf-idf (*Term frequency – Inverse document frequency*) para la construcción del vector de palabras.
- Utilizamos un nivel de dispersión del 0.97.
- Descartamos instancias y categorías que tenían menos de 100 instancias.
- Realizamos un muestreo aleatorio estratificado de 100 instancias por categoría.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 1000
- Nro. de Atributos: 58
- Nro. de Clases: 10
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 10% y 10%.

## A.11 Conjunto de datos Satimage

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir los diferentes vecindarios usando los valores obtenidos de imágenes multiespectrales satelitales.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 6435 |
| **Nro. de Atributos:** | 36 |
| **Tipos de Datos:** | Enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:** Los pasos realizados fueron los siguientes:

- Utilizamos los datos de entrenamiento.
- Eliminamos los registros que contenían datos incompletos.
- Realizamos un muestreo aleatorio estratificado de 300 instancias por categoría.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 1795
- Nro. de Atributos: 36
- Nro. de Clases: 6
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 9% y 25%.

## A.12 Conjunto de datos Segmentation

**Objetivo de la tarea de clasificación**:

Las instancias de este conjunto de datos fueron extraídas de una base de datos de imágenes de exteriores. El objetivo de la clasificación era predecir los materiales de construcción presentes en las imágenes.

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 2310 |
| **Nro. de Atributos:** | 19 |
| **Tipos de Datos:** | Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:** Realizamos los siguientes pasos:

- Utilizamos los datos de entrenamiento y *test*.
- Aplicamos una transformación de centrado y escalado a los datos de entrenamiento.
- Realizamos un muestreo aleatorio estratificado de 300 instancias por categoría.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 2310
- Nro. de Atributos: 18
- Nro. de Clases: 7
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 14% y 14%.

## A.13 Conjunto de datos Sports

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era identificar la actividad deportiva utilizando los datos obtenidos de un sistema de posicionamiento global (GPS, Global Positioning System).

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 8000 |
| **Nro. de Atributos:** | 10 |
| **Tipos de Datos:** | Reales y enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:** Realizamos los siguientes pasos:

- Descartamos la columna X por ser el identificador de la instancia.
- Aplicamos la transformación de centrado y escalado a los datos de entrenamiento.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 8000
- Nro. de Atributos: 13
- Nro. de Clases: 10
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 10% y 10%.

## A.14 Conjunto de datos TrafficLight

**Objetivo de la tarea de clasificación**:

Las instancias de este conjunto de datos han sido extraídas de una aplicación de semáforos de la Universidad de Ciencias Georges Bonga en Berlín. El objetivo de la clasificación era la identificación de señales de semáforos.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 612 |
| **Nro. de Atributos:** | 10 |
| **Tipos de Datos:** | Reales y enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Eliminamos los registros que contenían datos incompletos.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 300
- Nro. de Atributos: 10
- Nro. de Clases: 6
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 15% y 19%.

## A.15 Conjunto de datos Usps

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era reconocer los caracteres de textos escritos.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 7291 |
| **Nro. de Atributos:** | 10 |
| **Tipos de Datos:** | Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:** Los pasos dados fueron los siguientes:

- Eliminamos los registros que contenían datos incompletos.
- Realizamos un muestreo aleatorio estratificado de 300 instancias por categoría.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 3000
- Nro. de Atributos: 256
- Nro. de Clases: 10
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 10% y 10%.

## A.16 Conjunto de datos Vertebral

**Objetivo de la tarea de clasificación**:

Este conjunto de datos contiene valores de seis características biomecánicas que se utilizan para clasificar a pacientes ortopédicos.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 310 |
| **Nro. de Atributos:** | 7 |
| **Tipos de Datos:** | Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Eliminamos los registros que contenían datos incompletos.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 310
- Nro. de Atributos: 6
- Nro. de Clases: 4
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 19% y 48%.

## A.17 Conjunto de datos Yeast

**Objetivo de la tarea de clasificación**:

Las instancias que forman parte de este conjunto de datos han sido recopiladas de un sistema experto con el objetivo de predecir los lugares de localización de proteínas en bacterias Gram-negativas.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 1484 |
| **Nro. de Atributos:** | 8 |
| **Tipos de Datos:** | Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Eliminamos los registros que contenían datos incompletos.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 1481
- Nro. de Atributos: 8
- Nro. de Clases: 10
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 0.3% y 31%.

## A.18 Conjunto de datos Zoo

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir el tipo de animal según sus características.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 101 |
| **Nro. de Atributos:** | 16 |
| **Tipos de Datos:** | Categóricos y enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:** No realizamos ninguna acción.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 101
- Nro. de Atributos: 16
- Nro. de Clases: 7
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 4% y 20%.

## A.19 Conjunto de datos 20 Newsgroups / 10 Newsgroups

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir la categoría de una noticia escrita.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 20000 |
| **Nro. de Atributos:** | 16 |
| **Tipos de Datos:** | Palabras de textos |
| **Valores Nulos:** | No |

**Preprocesamiento:** Realizamos los siguientes pasos:

- Los transformamos todo el texto en minúsculas.
- Quitamos puntuaciones, números, *stopwords* y espacios en blanco.
- Aplicamos el método de ponderación Tf-idf (*Term frequency – Inverse document frequency*) para la construcción del vector de palabras.
- Utilizamos un nivel de dispersión del 0.97.
- Realizamos un muestreo aleatorio estratificado de 370 instancias por categoría para 10News y 666 instancias para 20News.
- Para el conjunto de datos 20News seleccionamos las 20 categorias, mientras que para 10News se seleccionamos las siguientes 10 categorias: {alt.atheism; comp.os.ms-windows.misc; comp.sys.mac.hardware; comp.windows.x; misc.forsale; rec.autos; rec.sport.baseball; sci.crypt; sci.electronics; sci.space}.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: Para 10News: 3700 y para 20 News 6660
- Nro. de Atributos: 45
- Nro. de Clases: 10 y 20
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 4% y 20%.

## A.20 Conjunto de datos Actinopterygii

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir la categoría de los peces Actinopterygii en función de sus características.

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 22369 |
| **Nro. de Atributos:** | 15 |
| **Tipos de Datos:** | Real |
| **Valores Nulos:** | No |

**Preprocesamiento:** No realizó ninguna actividad.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 22369
- Nro. de Atributos: 15
- Nro. de Clases: 15
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 0.08% y 16%.

## A.21 Conjunto de datos Diamonds

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir la calidad de un diamante según sus características.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 53940 |
| **Nro. de Atributos:** | 9 |
| **Tipos de Datos:** | Categóricos y Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:** No se realizó ninguna actividad.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 53940
- Nro. de Atributos: 9
- Nro. de Clases: 5
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 2% y 39%.

## A.22 Conjunto de datos Koppen

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era predecir el clima de Brasil con mapas de Koppen-Geiger.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 5567 |
| **Nro. de Atributos:** | 30 |
| **Tipos de Datos:** | Categóricos y Reales |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Utilizamos únicamente atributos reales.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 5567
- Nro. de Atributos: 25
- Nro. de Clases: 9
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 3% y 23%.

## A.23 Conjunto de datos Optdigits

**Objetivo de la tarea de clasificación**:

El objetivo de la clasificación era reconocer dígitos escritos a mano a partir de mapas de bits obtenidos de documentos preimpresos.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 5620 |
| **Nro. de Atributos:** | 64 |
| **Tipos de Datos:** | Enteros |
| **Valores Nulos:** | No |

**Preprocesamiento:**

- Utilizamos los datos de entrenamiento.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 3823
- Nro. de Atributos: 64
- Nro. de Clases: 10
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 9.8% y 10.17%.

## A.24 Conjunto de datos RandomRBFGen / RandomTreeGen

**Objetivo de la tarea de clasificación**:

Estos conjuntos de datos generados artificialmente.

**Características**:

| Descripción | Valor |
|---|---|
| **Nro. de Instancias:** | 50000 |
| **Nro. de Atributos:** | 10 |
| **Tipos de Datos:** | Categóricos y reales |
| **Valores Nulos:** | No |

**Preprocesamiento:** No Realizamos ninguna actividad.

**Características del dataset después del procesamiento.**

- Nro. de Instancias: 50000
- Nro. de Atributos: 10
- Nro. de Clases: 7
- Porcentajes de las clases mayoritaria y minoritaria calculados como el Nro. de instancias de las clases mayoritaria y minoritaria sobre el número total de instancias: 6% y 33%.

## Apéndice B

# Parámetros usados por las técnicas de clasificación

En esta sección se especifican los parámetros de las técnicas de clasificación utilizadas en los experimentos realizados.

## B.1 K-nearest neighbors (KNN)

| **Parámetro: *k*** | |
|---|---|
| **Descripción:** | Número de vecinos considerados. |
| **Valor:** | 1 |
| **Parámetro: *l*** | |
| **Descripción:** | Voto mínimo para la decisión definitiva. |
| **Valor:** | 0 |

## B.2 Neural Network (NNET)

| **Parámetro: *Weights*** | |
|---|---|
| **Descripción:** | Pesos para cada ejemplo si el valor no ha sido especificado. |
| **Valor:** | 1 |
| **Parámetro: *Maxit*** | |
| **Descripción:** | Número de iteracciones máximas. |
| **Valor:** | 100 |
| **Parámetro: *Decay*** | |
| **Descripción:** | Parámetro de decaimiento del peso. |
| **Valor:** | 1 |

## B.3 Random Forest (RF)

| **Parámetro: *N-tree*** | |
|---|---|
| **Descripción:** | Número de árboles a crecer. |
| **Valor:** | 500 |

## B.4 Decision Tree (J48)

| **Parámetro:** ***ConfidenceFactor*** | |
|---|---|
| **Descripción:** | El factor de confianza utilizado para la poda. |
| **Valor:** | 0.25 |
| **Parámetro:** ***MinInst*** | |
| **Descripción:** | Establece el número mínimo de instancias por hoja. |
| **Valor:** | 2 |

## B.5 Propositional rule learner (JRIP)

| **Parámetro:** ***BatchSize*** | |
|---|---|
| **Descripción:** | El número preferido de instancias a procesar si se realiza una predicción por lotes. |
| **Valor:** | 100 |
| **Parámetro:** ***CheckErrorRate*** | |
| **Descripción:** | Si la comprobación de la tasa de error es >= 1/2 estará incluida en el criterio de parada. |
| **Valor:** | True |
| **Parámetro:** ***Folds*** | |
| **Descripción:** | Determina la cantidad de datos utilizados para la poda. |
| **Valor:** | 3 |
| **Parámetro: minNo** | |
| **Descripción:** | El peso mínimo total de las instancias en una regla. |
| **Valor:** | 2 |
| **Parámetro:** ***UsePruning*** | |
| **Descripción:** | Especifica si se realiza la poda. |
| **Valor:** | True |

## B.6 Naive Bayes (NB)

| **Parámetro:** ***Laplace*** | |
|---|---|
| **Descripción:** | Doble control positivo de Laplace suavizado. El valor por defecto (0) desactiva el suavizado de Laplace. |
| **Valor:** | 0 |

## B.7 Rule-Base Classifier (RPART)

| **Parámetro:** ***Method*** | |
|---|---|
| **Descripción:** | Las opciones de este parámetro son: "anova", "poisson", "class" o "exp". Si falta el método, la rutina intenta hacer una suposición inteligente. |
| **Valor:** | auto |

## B.8 Support Vector Machines (SVM)

| **Parámetro:** ***Kernel*** | |
|---|---|
| **Descripción:** | El núcleo o kernel utilizado en el entrenamiento y la predicción. |
| **Valor:** | radial |
| **Parámetro:** ***Scale*** | |
| **Descripción:** | Por defecto, los datos se escalan internamente (tanto las variables x e y) a la media cero y a la varianza de la unidad. Los valores del centro y de la escala se devuelven y se usan para predicciones posteriores. |
| **Valor:** | True |
| **Parámetro:** ***Tolerance*** | |
| **Descripción:** | Tolerancia del criterio de terminación. |
| **Valor:** | 0.001 |

## B.9 Decision Trees and Rule-Base Models (C5.0)

| **Parámetro:** ***Trials*** | |
|---|---|
| **Descripción:** | Un número entero que especifica la cantidad de iteraciones de refuerzo. El valor de uno indica que se utiliza un modelo único |
| **Valor:** | 1 |

## B.10 Flexible Discriminant Analysis (FDA)

| **Parámetro:** ***Method*** | |
|---|---|
| **Descripción:** | Método de regresión utilizado en escala óptima. |
| **Valor:** | Análisis discriminante lineal. |

## Apéndice C

# Tablas de resultados

Esta sección contiene las tablas detalladas con los resultados obtenidos de los experimentos realizados en los capítulos.

| Método | | Conjunto de Datos | | | | |
|---|---|---|---|---|---|---|
| | | **Arrythmia** | **Covtype** | **Dermatology** | **Flare** | **Forest** |
| **J48** | **F** | 0.714±0.061 | 0.695±0.026 | 0.947±0.038 | 0.732±0.019 | 0.849±0.029 |
| | **E** | **0.716±0.035** | 0.708±0.031 | 0.947±0.033 | 0.738±0.04 | 0.874±0.044 |
| | **S** | 0.709±0.05 | 0.707±0.032 | 0.949±0.042 | **0.744±0.031** | 0.866±0.041 |
| | **C** | 0.68±0.081 | **0.711±0.034** | **0.955±0.035** | 0.728±0.04 | **0.877±0.045** |
| **JRIP** | **F** | 0.758±0.082 | 0.649±0.043 | 0.947±0.039 | 0.686±0.037 | 0.847±0.053 |
| | **E** | 0.73±0.058 | 0.686±0.033 | 0.961±0.037 | **0.742±0.038** | 0.877±0.045 |
| | **S** | **0.738±0.061** | **0.692±0.034** | 0.955±0.034 | 0.732±0.037 | **0.881±0.043** |
| | **C** | **0.738±0.07** | 0.676±0.031 | **0.964±0.038** | 0.71±0.034 | 0.871±0.03 |
| **KNN** | **F** | 0.637±0.038 | 0.643±0.032 | 0.875±0.047 | 0.72±0.022 | 0.893±0.047 |
| | **E** | **0.654±0.036** | **0.661±0.034** | 0.855±0.034 | 0.73±0.031 | 0.894±0.043 |
| | **S** | **0.654±0.036** | 0.658±0.034 | **0.858±0.036** | **0.733±0.032** | 0.891±0.043 |
| | **C** | **0.654±0.036** | 0.649±0.03 | **0.858±0.085** | 0.707±0.043 | **0.896±0.037** |
| **NB** | **F** | **0.589±0.017** | 0.218±0.024 | 0.916±0.039 | 0.593±0.027 | 0.87±0.04 |
| | **E** | **0.589±0.017** | 0.165±0.029 | **0.802±0.032** | 0.548±0.029 | 0.852±0.041 |
| | **S** | **0.589±0.017** | 0.146±0.028 | 0.771±0.061 | 0.551±0.02 | 0.852±0.044 |
| | **C** | **0.589±0.017** | **0.24±0.031** | 0.785±0.032 | **0.59±0.072** | **0.86±0.039** |
| **NNET** | **F** | 0.587±0.026 | 0.283±0.035 | 0.972±0.023 | 0.761±0.025 | 0.876±0.06 |
| | **E** | 0.644±0.063 | **0.508±0.053** | 0.958±0.024 | **0.759±0.033** | 0.898±0.038 |
| | **S** | 0.661±0.044 | 0.503±0.068 | 0.964±0.03 | **0.759±0.028** | **0.917±0.04** |
| | **C** | **0.685±0.076** | 0.454±0.021 | **0.966±0.018** | 0.751±0.05 | 0.906±0.046 |
| **PART** | **F** | 0.71±0.05 | 0.707±0.022 | 0.939±0.025 | 0.723±0.026 | 0.845±0.041 |
| | **E** | 0.678±0.084 | 0.697±0.04 | 0.95±0.042 | 0.726±0.03 | 0.861±0.048 |
| | **S** | 0.664±0.051 | **0.704±0.03** | **0.955±0.038** | 0.733±0.035 | **0.87±0.052** |
| | **C** | **0.709±0.081** | 0.695±0.015 | 0.944±0.039 | **0.741±0.039** | 0.866±0.046 |
| **RPART** | **F** | 0.738±0.058 | 0.625±0.038 | 0.941±0.037 | 0.739±0.028 | 0.868±0.041 |
| | **E** | 0.764±0.06 | 0.666±0.034 | 0.924±0.041 | 0.729±0.033 | **0.868±0.057** |
| | **S** | **0.769±0.051** | 0.667±0.042 | 0.927±0.035 | **0.736±0.03** | 0.866±0.052 |
| | **C** | 0.757±0.062 | **0.669±0.037** | **0.941±0.04** | 0.723±0.04 | 0.858±0.038 |
| **RF** | **F** | 0.8±0.046 | 0.772±0.02 | 0.978±0.021 | 0.738±0.038 | 0.883±0.041 |
| | **E** | **0.805±0.05** | **0.775±0.027** | 0.978±0.027 | 0.75±0.037 | **0.891±0.04** |
| | **S** | 0.8±0.058 | 0.774±0.023 | **0.983±0.025** | **0.751±0.031** | 0.889±0.039 |
| | **C** | 0.793±0.044 | 0.769±0.026 | 0.98±0.027 | 0.735±0.023 | 0.889±0.039 |
| **SVM** | **F** | **0.589±0.016** | 0.684±0.027 | 0.97±0.027 | 0.756±0.024 | 0.895±0.033 |
| | **E** | **0.589±0.016** | 0.65±0.028 | 0.937±0.034 | **0.749±0.042** | **0.898±0.039** |
| | **S** | **0.589±0.016** | 0.69±0.028 | **0.945±0.029** | 0.744±0.036 | **0.898±0.039** |
| | **C** | **0.589±0.016** | **0.702±0.028** | 0.925±0.037 | 0.735±0.036 | 0.896±0.04 |
| **C50** | **F** | 0.719±0.058 | 0.708±0.032 | 0.941±0.043 | 0.728±0.047 | 0.855±0.048 |
| | **E** | **0.713±0.06** | 0.709±0.007 | 0.944±0.035 | 0.736±0.04 | 0.864±0.061 |
| | **S** | 0.709±0.078 | **0.713±0.015** | 0.947±0.03 | **0.741±0.035** | 0.864±0.054 |
| | **C** | 0.706±0.106 | 0.712±0.034 | **0.955±0.04** | 0.734±0.038 | **0.877±0.045** |
| **FDA** | **F** | 0.755±0.067 | 0.684±0.059 | 0.953±0.024 | 0.755±0.033 | 0.896±0.042 |
| | **E** | **0.774±0.05** | 0.699±0.057 | 0.947±0.036 | 0.749±0.021 | 0.877±0.066 |
| | **S** | 0.772±0.049 | **0.703±0.06** | 0.944±0.032 | 0.754±0.032 | 0.881±0.063 |
| | **C** | 0.757±0.069 | **0.703±0.06** | **0.958±0.027** | **0.758±0.02** | **0.889±0.034** |

**Tabla C.1:** (a) Versión extendida de la tabla 4.7: valores de *accuracy* y desviación estándar obtenidos por las diferentes técnicas de clasificación, utilizando los conjuntos de datos del 1 al 5 de la tabla 4.6. Para cada técnica de clasificación y conjunto de datos el mejor resultado está subrayado; y además, el mejor método que utiliza una jerarquía está resaltado en negrita.

| Método | | Conjunto de Datos | | | | |
|---|---|---|---|---|---|---|
| | | Glass | Letters | Nuclear | Pendigits | Reuters |
| **J48** | **F** | 0.72±0.056 | 0.736±0.03 | 1±0 | 0.962±0.007 | - |
| | **E** | 0.685±0.109 | **0.72±0.027** | 0.996±0.008 | 0.956±0.007 | - |
| | **S** | 0.714±0.091 | **0.72±0.022** | **1±0** | **0.961±0.007** | - |
| | **C** | **0.739±0.103** | 0.703±0.006 | 0.989±0.007 | 0.954±0.007 | - |
| **JRIP** | **F** | 0.676±0.107 | 0.687±0.038 | 0.899±0.078 | 0.952±0.005 | - |
| | **E** | **0.704±0.091** | 0.708±0.017 | 0.955±0.03 | 0.956±0.009 | - |
| | **S** | 0.684±0.091 | 0.715±0.027 | 0.96±0.033 | 0.959±0.005 | - |
| | **C** | 0.671±0.104 | **0.744±0.032** | **0.975±0.032** | **0.96±0.005** | - |
| **KNN** | **F** | 0.669±0.068 | 0.8±0.023 | 0.995±0.012 | 0.993±0.003 | 0.507±0.027 |
| | **E** | 0.621±0.092 | 0.766±0.028 | **0.996±0.026** | **0.993±0.008** | 0.45±0.044 |
| | **S** | 0.612±0.084 | **0.779±0.019** | **0.996±0.011** | **0.993±0.005** | **0.471±0.026** |
| | **C** | **0.622±0.068** | 0.721±0.022 | **0.996±0.013** | **0.993±0.004** | 0.444±0.035 |
| **NB** | **F** | 0.571±0.109 | 0.674±0.015 | 0.975±0.033 | 0.883±0.009 | 0.213±0.028 |
| | **E** | 0.559±0.039 | 0.581±0.03 | 0.886±0.023 | 0.805±0.006 | 0.186±0.044 |
| | **S** | 0.583±0.054 | 0.535±0.036 | **0.982±0.031** | 0.793±0.009 | 0.197±0.026 |
| | **C** | **0.599±0.059** | **0.707±0.02** | 0.879±0.032 | **0.864±0.008** | **0.223±0.031** |
| **NNET** | **F** | 0.703±0.096 | 0.362±0.101 | **1±0** | 0.463±0.181 | 0.537±0.035 |
| | **E** | 0.684±0.096 | **0.77±0.039** | **1±0** | 0.92±0.012 | 0.569±0.044 |
| | **S** | **0.703±0.099** | 0.763±0.03 | **1±0** | 0.913±0.023 | **0.578±0.051** |
| | **C** | 0.694±0.079 | 0.72±0.033 | **1±0** | **0.921±0.012** | 0.559±0.043 |
| **PART** | **F** | 0.711±0.075 | 0.724±0.032 | 1±0 | 0.97±0.007 | - |
| | **E** | 0.678±0.087 | 0.727±0.033 | 0.996±0.008 | 0.96±0.004 | - |
| | **S** | 0.693±0.064 | 0.708±0.038 | **1±0** | **0.966±0.01** | - |
| | **C** | **0.711±0.092** | **0.73±0.032** | 0.998±0.008 | 0.96±0.004 | - |
| **RPART** | **F** | 0.7±0.126 | 0.463±0.009 | 1±0 | 0.836±0.007 | 0.505±0.03 |
| | **E** | **0.708±0.118** | 0.645±0.014 | 0.984±0.006 | 0.901±0.009 | 0.492±0.041 |
| | **S** | 0.68±0.103 | 0.636±0.012 | **1±0** | **0.906±0.017** | **0.508±0.031** |
| | **C** | 0.695±0.099 | **0.646±0.008** | 0.962±0.005 | 0.882±0.01 | 0.5±0.029 |
| **RF** | **F** | 0.818±0.104 | 0.875±0.004 | 1±0 | **0.991±0.004** | 0.635±0.043 |
| | **E** | 0.816±0.054 | 0.849±0.006 | 0.998±0.015 | **0.991±0.003** | 0.557±0.05 |
| | **S** | **0.831±0.048** | **0.88±0.004** | **1±0** | **0.991±0.004** | 0.572±0.041 |
| | **C** | 0.797±0.058 | 0.822±0.006 | **1±0** | 0.988±0.006 | **0.578±0.042** |
| **SVM** | **F** | 0.714±0.076 | 0.834±0.005 | 0.998±0.006 | 0.995±0.002 | 0.569±0.046 |
| | **E** | 0.659±0.074 | 0.753±0.005 | **0.998±0.006** | 0.994±0.003 | 0.477±0.053 |
| | **S** | 0.666±0.078 | 0.803±0.008 | **0.998±0.006** | **0.996±0.002** | **0.503±0.047** |
| | **C** | **0.68±0.074** | **0.819±0.006** | 0.996±0.01 | 0.994±0.001 | 0.46±0.051 |
| **C50** | **F** | 0.713±0.103 | 0.725±0.005 | 1±0 | 0.968±0.011 | 0.561±0.03 |
| | **E** | 0.694±0.097 | 0.697±0.039 | 0.995±0.018 | 0.955±0.007 | 0.519±0.027 |
| | **S** | 0.731±0.11 | 0.742±0.049 | 1±0 | 0.959±0.004 | 0.515±0.04 |
| | **C** | 0.7±0.129 | 0.718±0.005 | 0.996±0.018 | 0.956±0.007 | **0.53±0.028** |
| **FDA** | **F** | 0.691±0.061 | 0.642±0.01 | 1±0 | 0.874±0.008 | 0.495±0.026 |
| | **E** | 0.684±0.086 | 0.68±0.022 | 0.998±0.006 | 0.939±0.014 | 0.495±0.026 |
| | **S** | 0.694±0.093 | **0.737±0.015** | **1±0** | **0.955±0.011** | **0.505±0.028** |
| | **C** | **0.727±0.061** | **0.737±0.01** | 0.998±0.006 | 0.927±0.01 | 0.484±0.024 |

**Tabla C.1**: (b) Versión extendida de la tabla 4.7: valores de *accuracy* y desviación estándar obtenidos por las diferentes técnicas de clasificación, utilizando los conjuntos de datos del 6 al 10 de la tabla 4.6. Para cada técnica de clasificación y conjunto de datos el mejor resultado está subrayado; y además, el mejor método que utiliza una jerarquía está resaltado en negrita.

| Método | | Conjunto de Datos | | | | |
|---|---|---|---|---|---|---|
| | | **Satimage** | **Segmentation** | **Sports** | **TrafficLight** | **Usps** |
| **J48** | **F** | 0.826±0.033 | 0.871±0.078 | 0.624±0.009 | 0.74±0.099 | 0.844±0.021 |
| | **E** | 0.811±0.042 | **0.957±0.069** | 0.62±0.013 | **0.747±0.066** | 0.848±0.024 |
| | **S** | **0.817±0.035** | 0.956±0.047 | **0.628±0.013** | 0.724±0.094 | 0.852±0.023 |
| | **C** | 0.816±0.016 | **0.957±0.014** | 0.62±0.023 | 0.711±0.059 | **0.861±0.021** |
| **JRIP** | **F** | 0.827±0.019 | 0.838±0.064 | 0.569±0.007 | 0.72±0.08 | 0.842±0.02 |
| | **E** | **0.839±0.026** | **0.957±0.071** | **0.615±0.016** | 0.713±0.08 | 0.857±0.015 |
| | **S** | 0.826±0.025 | **0.957±0.071** | 0.608±0.016 | 0.743±0.094 | 0.863±0.023 |
| | **C** | 0.82±0.018 | 0.95±0.014 | 0.608±0.018 | **0.76±0.089** | **0.866±0.018** |
| **KNN** | **F** | 0.871±0.012 | 0.79±0.068 | 0.524±0.018 | 0.325±0.079 | 0.954±0.013 |
| | **E** | **0.875±0.026** | 0.935±0.068 | **0.523±0.018** | 0.347±0.09 | 0.954±0.008 |
| | **S** | 0.874±0.021 | 0.935±0.068 | 0.522±0.012 | 0.334±0.097 | **0.955±0.008** |
| | **C** | **0.875±0.019** | **0.938±0.015** | 0.52±0.021 | **0.361±0.113** | 0.954±0.009 |
| **NB** | **F** | 0.822±0.026 | 0.824±0.071 | 0.571±0.015 | 0.543±0.086 | 0.738±0.016 |
| | **E** | 0.75±0.027 | 0.871±0.078 | 0.511±0.017 | 0.452±0.079 | 0.632±0.035 |
| | **S** | 0.749±0.027 | 0.871±0.088 | 0.52±0.011 | 0.371±0.07 | 0.684±0.037 |
| | **C** | **0.795±0.018** | **0.878±0.022** | **0.538±0.026** | **0.482±0.056** | **0.694±0.021** |
| **NNET** | **F** | 0.843±0.023 | 0.605±0.0138 | 0.229±0.04 | 0.659±0.106 | 0.672±0.047 |
| | **E** | **0.876±0.016** | **0.937±0.084** | 0.491±0.053 | 0.634±0.071 | **0.935±0.013** |
| | **S** | 0.87±0.02 | 0.933±0.067 | **0.517±0.02** | **0.707±0.054** | 0.931±0.014 |
| | **C** | 0.87±0.013 | 0.913±0.012 | 0.508±0.024 | 0.56±0.079 | 0.931±0.02 |
| **PART** | **F** | 0.824±0.028 | 0.881±0.068 | 0.618±0.012 | 0.753±0.085 | 0.878±0.027 |
| | **E** | 0.812±0.025 | 0.951±0.068 | **0.592±0.014** | 0.694±0.094 | 0.868±0.027 |
| | **S** | **0.824±0.024** | 0.954±0.069 | 0.588±0.013 | **0.7±0.064** | 0.861±0.012 |
| | **C** | 0.819±0.017 | **0.962±0.014** | 0.59±0.019 | 0.693±0.072 | **0.873±0.024** |
| **RPART** | **F** | 0.793±0.02 | 0.917±0.013 | 0.559±0.019 | 0.746±0.089 | 0.76±0.03 |
| | **E** | **0.803±0.019** | **0.954±0.019** | 0.589±0.02 | 0.707±0.105 | 0.814±0.03 |
| | **S** | 0.798±0.023 | 0.948±0.017 | 0.584±0.027 | 0.724±0.088 | 0.83±0.026 |
| | **C** | 0.799±0.018 | 0.932±0.019 | **0.596±0.019** | **0.736±0.113** | **0.84±0.016** |
| **RF** | **F** | 0.882±0.014 | 0.974±0.011 | 0.71±0.012 | 0.807±0.068 | 0.943±0.013 |
| | **E** | 0.876±0.014 | **0.976±0.012** | **0.708±0.015** | **0.807±0.073** | 0.9±0.009 |
| | **S** | 0.875±0.015 | 0.975±0.009 | 0.707±0.012 | 0.803±0.082 | **0.938±0.009** |
| | **C** | **0.877±0.011** | 0.972±0.012 | 0.7±0.015 | 0.79±0.081 | 0.926±0.011 |
| **SVM** | **F** | 0.87±0.029 | 0.939±0.015 | 0.626±0.017 | 0.571±0.078 | 0.969±0.005 |
| | **E** | 0.872±0.011 | **0.958±0.009** | 0.607±0.015 | 0.67±0.082 | 0.956±0.011 |
| | **S** | **0.873±0.013** | **0.958±0.009** | **0.616±0.012** | **0.677±0.08** | 0.957±0.015 |
| | **C** | 0.871±0.013 | 0.934±0.01 | 0.612±0.019 | 0.571±0.079 | **0.965±0.013** |
| **C50** | **F** | 0.822±0.013 | 0.963±0.012 | 0.622±0.013 | 0.769±0.078 | 0.851±0.022 |
| | **E** | 0.815±0.022 | 0.96±0.011 | 0.62±0.017 | 0.746±0.085 | 0.851±0.017 |
| | **S** | **0.822±0.017** | 0.954±0.01 | **0.625±0.012** | 0.769±0.079 | 0.817±0.026 |
| | **C** | 0.82±0.022 | **0.961±0.013** | 0.62±0.021 | **0.786±0.067** | **0.862±0.026** |
| **FDA** | **F** | 0.842±0.015 | 0.948±0.018 | 0.577±0.013 | 0.715±0.1 | 0.811±0.018 |
| | **E** | **0.853±0.022** | **0.966±0.002** | 0.604±0.011 | **0.801±0.109** | 0.884±0.017 |
| | **S** | 0.851±0.021 | 0.963±0.002 | 0.599±0.008 | 0.787±0.098 | **0.896±0.084** |
| | **C** | 0.848±0.022 | 0.956±0.002 | **0.606±0.023** | 0.707±0.117 | 0.895±0.016 |

**Tabla C.1**: (c) Versión extendida de la tabla 4.7: valores de *accuracy* y desviación estándar obtenidos por las diferentes técnicas de clasificación, utilizando los conjuntos de datos del 11 al 15 de la tabla 4.6. Para cada técnica de clasificación y conjunto de datos el mejor resultado está subrayado; y además, el mejor método que utiliza una jerarquía está resaltado en negrita.

| Método | | Conjunto de Datos | | | | |
|---|---|---|---|---|---|---|
| | | **Vertebral** | **Yeast** | **Zoo** | **10News** | **20News** |
| **J48** | **F** | 0.516±0.026 | 0.605±0.042 | 0.925±0.086 | - | - |
| | **E** | **0.581±0.036** | 0.573±0.05 | **0.943±0.105** | - | - |
| | **S** | 0.563±0.027 | 0.581±0.043 | 0.933±0.105 | - | - |
| | **C** | 0.56±0.026 | **0.587±0.037** | 0.929±0.071 | - | - |
| **JRIP** | **F** | 0.577±0.032 | **0.587±0.034** | 0.869±0.079 | - | - |
| | **E** | **0.568±0.056** | **0.587±0.039** | 0.902±0.099 | - | - |
| | **S** | 0.561±0.05 | 0.574±0.057 | **0.931±0.068** | - | - |
| | **C** | 0.54±0.045 | 0.577±0.046 | **0.931±0.052** | - | - |
| **KNN** | **F** | 0.394±0.051 | 0.588±0.044 | 0.861±0.08 | 0.246±0.012 | 0.172±0.013 |
| | **E** | 0.406±0.028 | 0.579±0.037 | 0.812±0.064 | 0.214±0.018 | 0.137±0.019 |
| | **S** | **0.418±0.037** | 0.578±0.035 | **0.824±0.105** | **0.241±0.019** | **0.166±0.017** |
| | **C** | 0.4±0.045 | **0.581±0.037** | 0.812±0.071 | 0.228±0.015 | 0.15±0.016 |
| **NB** | **F** | 0.544±0.078 | 0.424±0.054 | 0.882±0.087 | 0.185±0.017 | 0.143±0.014 |
| | **E** | 0.527±0.044 | 0.35±0.038 | 0.599±0.12 | **0.171±0.016** | **0.108±0.008** |
| | **S** | **0.532±0.047** | 0.344±0.037 | 0.645±0.172 | 0.142±0.012 | 0.078±0.011 |
| | **C** | 0.531±0.081 | **0.362±0.048** | **0.796±0.162** | 0.157±0.013 | 0.101±0.011 |
| **NNET** | **F** | 0.521±0.033 | 0.596±0.026 | 0.941±0.067 | 0.264±0.008 | 0.163±0.009 |
| | **E** | 0.532±0.052 | 0.569±0.037 | 0.953±0.061 | 0.264±0.014 | 0.181±0.015 |
| | **S** | **0.566±0.043** | 0.579±0.052 | **0.963±0.061** | **0.281±0.022** | **0.192±0.021** |
| | **C** | 0.558±0.032 | **0.587±0.039** | 0.954±0.052 | 0.273±0.015 | 0.176±0.022 |
| **PART** | **F** | 0.526±0.041 | 0.555±0.044 | 0.925±0.105 | - | - |
| | **E** | 0.571±0.058 | 0.555±0.055 | **0.953±0.105** | - | - |
| | **S** | **0.579±0.05** | **0.571±0.04** | 0.943±0.105 | - | - |
| | **C** | **0.579±0.05** | 0.551±0.036 | 0.911±0.052 | - | - |
| **RPART** | **F** | 0.492±0.049 | 0.569±0.047 | 0.873±0.033 | 0.194±0.012 | 0.103±0.006 |
| | **E** | 0.427±0.06 | 0.579±0.05 | **0.866±0.075** | 0.228±0.014 | 0.131±0.015 |
| | **S** | **0.45±0.06** | 0.583±0.043 | **0.866±0.044** | **0.229±0.016** | **0.141±0.015** |
| | **C** | **0.45±0.06** | **0.584±0.044** | 0.832±0.085 | 0.193±0.011 | 0.123±0.013 |
| **RF** | **F** | 0.339±0.043 | 0.627±0.049 | 0.973±0.048 | 0.338±0.015 | 0.259±0.012 |
| | **E** | 0.329±0.039 | 0.604±0.042 | **0.982±0.048** | 0.265±0.009 | 0.163±0.01 |
| | **S** | **0.339±0.046** | 0.608±0.054 | 0.973±0.048 | 0.26±0.007 | 0.164±0.008 |
| | **C** | 0.335±0.02 | **0.616±0.051** | 0.981±0.04 | **0.289±0.018** | **0.193±0.019** |
| **SVM** | **F** | 0.473±0.034 | 0.596±0.046 | 0.951±0.052 | 0.327±0.009 | 0.23±0.01 |
| | **E** | 0.46±0.04 | **0.584±0.045** | **0.972±0.052** | 0.151±0.014 | 0.089±0.015 |
| | **S** | **0.477±0.04** | 0.569±0.047 | **0.972±0.052** | **0.272±0.012** | **0.185±0.014** |
| | **C** | **0.477±0.04** | 0.555±0.046 | 0.962±0.052 | 0.238±0.018 | 0.157±0.02 |
| **C50** | **F** | 0.51±0.052 | 0.56±0.035 | 0.943±0.086 | 0.213±0.015 | 0.133±0.016 |
| | **E** | **0.598±0.037** | 0.562±0.053 | **0.943±0.065** | 0.2±0.01 | 0.116±0.009 |
| | **S** | 0.596±0.032 | 0.577±0.054 | 0.925±0.063 | 0.221±0.017 | **0.148±0.016** |
| | **C** | 0.571±0.042 | **0.592±0.051** | 0.909±0.093 | **0.224±0.02** | 0.139±0.011 |
| **FDA** | **F** | 0.513±0.037 | 0.592±0.037 | 0.933±0.067 | 0.261±0.019 | 0.172±0.017 |
| | **E** | **0.548±0.04** | 0.578±0.031 | 0.933±0.189 | 0.226±0.011 | 0.144±0.012 |
| | **S** | **0.548±0.04** | **0.586±0.107** | **0.953±0.042** | **0.275±0.022** | **0.191±0.024** |
| | **C** | **0.548±0.04** | **0.586±0.114** | 0.942±0.246 | 0.257±0.017 | 0.169±0.016 |

**Tabla C.1**: (d) Versión extendida de la tabla 4.7: valores de *accuracy* y desviación estándar obtenidos por las diferentes técnicas de clasificación, utilizando los conjuntos de datos del 16 al 20 de la tabla 4.6. Para cada técnica de clasificación y conjunto de datos el mejor resultado está subrayado; y además, el mejor método que utiliza una jerarquía está resaltado en negrita.

| | | Conjunto de Datos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **J48** | **F** | 0.113 | 0.103 | 0.037 | 0.038 | 0.074 | 0.134 | 0.164 | 0.000 | 0.034 |
| | **E** | **0.111** | 0.086 | 0.037 | 0.030 | 0.047 | 0.176 | **0.182** | 0.004 | 0.040 |
| | **S** | 0.119 | 0.088 | 0.035 | **0.022** | 0.056 | 0.141 | **0.182** | **0.000** | **0.035** |
| | **C** | 0.155 | **0.083** | **0.028** | 0.043 | **0.044** | **0.111** | 0.201 | 0.011 | 0.042 |
| **JRIP** | **F** | 0.058 | 0.163 | 0.037 | 0.099 | 0.076 | 0.187 | 0.219 | 0.101 | 0.044 |
| | **E** | 0.093 | 0.115 | 0.022 | **0.025** | 0.044 | **0.153** | 0.195 | 0.045 | 0.040 |
| | **S** | **0.083** | **0.107** | 0.028 | 0.038 | **0.039** | 0.177 | 0.188 | 0.040 | 0.037 |
| | **C** | **0.083** | 0.128 | **0.019** | 0.067 | 0.050 | 0.193 | **0.155** | **0.025** | **0.036** |
| **KNN** | **F** | 0.209 | 0.170 | 0.110 | 0.054 | 0.026 | 0.195 | 0.091 | 0.005 | 0.003 |
| | **E** | **0.188** | **0.147** | 0.130 | 0.041 | 0.025 | 0.253 | 0.130 | **0.004** | **0.003** |
| | **S** | **0.188** | 0.151 | **0.127** | **0.037** | 0.028 | 0.264 | **0.115** | **0.004** | **0.003** |
| | **C** | **0.188** | 0.163 | **0.127** | 0.071 | **0.023** | **0.252** | 0.181 | **0.004** | **0.003** |
| **NB** | **F** | **0.268** | 0.719 | 0.068 | 0.221 | 0.051 | 0.313 | 0.234 | 0.025 | 0.113 |
| | **E** | **0.268** | 0.787 | **0.184** | 0.280 | 0.071 | 0.327 | 0.340 | 0.114 | 0.192 |
| | **S** | **0.268** | 0.812 | 0.216 | 0.276 | 0.071 | 0.298 | 0.392 | **0.018** | 0.204 |
| | **C** | **0.268** | **0.690** | 0.201 | **0.225** | **0.062** | **0.279** | **0.197** | 0.121 | **0.133** |
| **NNET** | **F** | 0.271 | 0.635 | 0.011 | 0.000 | 0.045 | 0.154 | 0.589 | 0.000 | 0.535 |
| | **E** | 0.200 | **0.345** | 0.025 | **0.003** | 0.021 | 0.177 | **0.125** | **0.000** | 0.076 |
| | **S** | 0.179 | 0.351 | 0.019 | **0.003** | **0.000** | **0.154** | 0.133 | **0.000** | 0.083 |
| | **C** | **0.149** | 0.414 | **0.017** | 0.013 | 0.012 | 0.165 | 0.182 | **0.000** | **0.075** |
| **PART** | **F** | 0.118 | 0.088 | 0.045 | 0.050 | 0.079 | 0.144 | 0.177 | 0.000 | 0.026 |
| | **E** | 0.158 | 0.101 | 0.034 | 0.046 | 0.061 | 0.184 | 0.174 | 0.004 | 0.036 |
| | **S** | 0.175 | **0.092** | **0.028** | 0.037 | **0.051** | 0.166 | 0.195 | **0.000** | **0.030** |
| | **C** | **0.119** | 0.103 | 0.040 | **0.026** | 0.056 | **0.144** | **0.170** | 0.002 | 0.036 |
| **RPART** | **F** | 0.083 | 0.194 | 0.043 | 0.029 | 0.053 | 0.158 | 0.474 | 0.000 | 0.161 |
| | **E** | 0.051 | 0.141 | 0.060 | 0.042 | **0.053** | **0.148** | 0.267 | 0.016 | 0.095 |
| | **S** | **0.045** | 0.139 | 0.057 | **0.033** | 0.056 | 0.182 | 0.277 | **0.000** | **0.090** |
| | **C** | 0.060 | **0.137** | **0.043** | 0.050 | 0.064 | 0.164 | **0.266** | 0.038 | 0.114 |
| **RF** | **F** | 0.006 | 0.004 | 0.005 | 0.030 | 0.037 | 0.016 | 0.006 | 0.000 | 0.005 |
| | **E** | **0.000** | **0.000** | 0.005 | 0.014 | **0.028** | 0.018 | 0.035 | 0.002 | **0.005** |
| | **S** | 0.006 | 0.001 | **0.000** | **0.013** | 0.031 | **0.000** | **0.000** | **0.000** | **0.005** |
| | **C** | 0.015 | 0.008 | 0.003 | 0.034 | 0.031 | 0.041 | 0.066 | 0.000 | 0.008 |
| **SVM** | **F** | **0.268** | 0.117 | 0.013 | 0.007 | 0.024 | 0.141 | 0.052 | 0.002 | 0.001 |
| | **E** | **0.268** | 0.161 | 0.047 | **0.016** | **0.021** | 0.207 | 0.144 | **0.002** | 0.002 |
| | **S** | **0.268** | 0.110 | **0.039** | 0.022 | **0.021** | 0.199 | 0.087 | **0.002** | **0.000** |
| | **C** | **0.268** | **0.094** | 0.059 | 0.034 | 0.023 | **0.182** | **0.069** | 0.004 | 0.002 |
| **C50** | **F** | 0.107 | 0.086 | 0.043 | 0.043 | 0.068 | 0.142 | 0.176 | 0.000 | 0.028 |
| | **E** | **0.114** | 0.085 | 0.040 | 0.033 | 0.058 | 0.165 | 0.208 | 0.005 | 0.041 |
| | **S** | 0.119 | **0.080** | 0.037 | **0.026** | 0.058 | **0.120** | **0.157** | **0.000** | **0.037** |
| | **C** | 0.123 | 0.081 | **0.028** | 0.035 | **0.044** | 0.158 | 0.184 | 0.004 | 0.040 |
| **FDA** | **F** | 0.062 | 0.117 | 0.031 | 0.008 | 0.023 | 0.168 | 0.270 | 0.000 | 0.122 |
| | **E** | **0.039** | 0.098 | 0.037 | 0.016 | 0.044 | 0.177 | 0.227 | 0.002 | 0.057 |
| | **S** | 0.041 | **0.093** | 0.040 | 0.009 | 0.039 | 0.165 | **0.163** | **0.000** | **0.041** |
| | **C** | 0.060 | **0.093** | **0.025** | **0.004** | **0.031** | **0.125** | **0.163** | 0.002 | 0.069 |

**Tabla C.2:** (a) Valores de diferencias relativas de accuracy ($df = 1 - (Accuracy/MejorAccuracyxConjunto)$) por conjunto de datos obtenidos con los métodos jerárquicos y plano. Para cada método de clasificación y conjunto de datos (del 1 al 9 de la tabla C.1) el mejor resultado está subrayado, y el mejor método que utiliza una jerarquía está resaltado en negrita.

| | | Conjunto de Datos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **Promedio** |
| **J48** | **F** | 0.063 | 0.108 | 0.121 | 0.083 | 0.129 | 0.137 | 0.035 | 0.058 | 0.084 |
| | **E** | 0.080 | **0.019** | 0.127 | **0.074** | 0.125 | **0.028** | 0.086 | **0.040** | 0.076 |
| | **S** | **0.074** | 0.020 | **0.115** | 0.103 | 0.121 | 0.059 | 0.073 | 0.050 | **0.076** |
| | **C** | 0.075 | **0.019** | 0.127 | 0.119 | **0.111** | 0.064 | **0.064** | 0.054 | 0.079 |
| **JRIP** | **F** | 0.062 | 0.141 | 0.199 | 0.108 | 0.131 | 0.035 | **0.064** | 0.115 | 0.108 |
| | **E** | **0.049** | **0.019** | **0.134** | 0.116 | 0.116 | **0.050** | **0.064** | 0.081 | 0.080 |
| | **S** | 0.063 | **0.019** | 0.144 | 0.079 | 0.109 | 0.062 | 0.085 | **0.052** | **0.079** |
| | **C** | 0.070 | 0.027 | 0.144 | **0.058** | **0.106** | 0.097 | 0.080 | **0.052** | 0.082 |
| **KNN** | **F** | 0.012 | 0.191 | 0.262 | 0.597 | 0.015 | 0.341 | 0.062 | 0.123 | 0.145 |
| | **E** | **0.008** | 0.042 | **0.263** | 0.570 | 0.015 | 0.321 | 0.077 | 0.173 | 0.141 |
| | **S** | 0.009 | 0.042 | 0.265 | 0.586 | **0.014** | **0.301** | 0.078 | **0.161** | **0.140** |
| | **C** | **0.008** | **0.039** | 0.268 | **0.553** | 0.015 | 0.331 | **0.073** | 0.173 | 0.145 |
| **NB** | **F** | 0.068 | 0.156 | 0.196 | 0.327 | 0.238 | 0.090 | 0.324 | 0.102 | 0.207 |
| | **E** | 0.150 | 0.108 | 0.280 | 0.440 | 0.348 | 0.119 | 0.442 | 0.390 | 0.285 |
| | **S** | 0.151 | 0.108 | 0.268 | 0.540 | 0.294 | **0.110** | 0.451 | 0.343 | 0.284 |
| | **C** | **0.099** | **0.100** | **0.242** | **0.403** | **0.284** | 0.112 | **0.423** | **0.189** | **0.237** |
| **NNET** | **F** | 0.044 | 0.380 | 0.677 | 0.183 | 0.307 | 0.129 | 0.049 | 0.042 | 0.238 |
| | **E** | **0.007** | **0.040** | 0.308 | 0.214 | **0.035** | 0.110 | 0.093 | 0.030 | 0.106 |
| | **S** | 0.014 | 0.044 | **0.272** | **0.124** | 0.039 | **0.054** | 0.077 | **0.019** | **0.092** |
| | **C** | 0.014 | 0.065 | 0.285 | 0.306 | 0.039 | 0.067 | **0.064** | 0.029 | 0.111 |
| **PART** | **F** | 0.066 | 0.097 | 0.130 | 0.067 | 0.094 | 0.120 | 0.115 | 0.058 | 0.087 |
| | **E** | 0.079 | 0.026 | **0.166** | 0.140 | 0.104 | 0.045 | 0.115 | **0.030** | 0.088 |
| | **S** | **0.066** | 0.023 | 0.172 | **0.133** | 0.111 | **0.032** | **0.089** | 0.040 | 0.085 |
| | **C** | 0.071 | **0.014** | 0.169 | 0.141 | **0.099** | **0.032** | 0.121 | 0.072 | **0.083** |
| **RPART** | **F** | 0.101 | 0.060 | 0.213 | 0.076 | 0.216 | 0.177 | 0.093 | 0.111 | 0.132 |
| | **E** | **0.090** | **0.023** | 0.170 | 0.124 | 0.160 | 0.286 | 0.077 | **0.118** | 0.113 |
| | **S** | 0.095 | 0.029 | 0.177 | 0.103 | 0.143 | **0.247** | 0.070 | **0.118** | **0.110** |
| | **C** | 0.094 | 0.045 | **0.161** | **0.088** | **0.133** | **0.247** | **0.069** | 0.153 | 0.113 |
| **RF** | **F** | 0.000 | 0.002 | 0.000 | 0.000 | 0.027 | 0.433 | 0.000 | 0.009 | 0.034 |
| | **E** | 0.007 | **0.000** | **0.003** | **0.000** | 0.071 | 0.450 | 0.037 | **0.000** | 0.040 |
| | **S** | 0.008 | 0.001 | 0.004 | 0.005 | **0.032** | **0.433** | 0.030 | 0.009 | **0.034** |
| | **C** | **0.006** | 0.004 | 0.014 | 0.021 | 0.044 | 0.440 | **0.018** | 0.001 | 0.044 |
| **SVM** | **F** | 0.014 | 0.038 | 0.118 | 0.292 | 0.000 | 0.209 | 0.049 | 0.032 | 0.081 |
| | **E** | 0.011 | **0.018** | 0.145 | 0.170 | 0.013 | 0.231 | **0.069** | **0.010** | 0.090 |
| | **S** | **0.010** | **0.018** | **0.132** | **0.161** | 0.012 | **0.202** | 0.093 | **0.010** | **0.082** |
| | **C** | 0.012 | 0.043 | 0.138 | 0.292 | **0.004** | **0.202** | 0.115 | 0.020 | 0.092 |
| **C50** | **F** | **0.068** | 0.013 | 0.124 | 0.047 | 0.122 | 0.147 | 0.107 | 0.040 | 0.080 |
| | **E** | 0.076 | 0.016 | 0.127 | 0.076 | 0.122 | **0.000** | 0.104 | **0.040** | 0.077 |
| | **S** | **0.068** | 0.023 | **0.120** | 0.047 | 0.157 | 0.003 | 0.080 | 0.058 | **0.070** |
| | **C** | 0.070 | **0.015** | 0.127 | **0.026** | **0.110** | 0.045 | **0.056** | 0.074 | 0.072 |
| **FDA** | **F** | 0.045 | 0.029 | 0.187 | 0.114 | 0.163 | 0.142 | 0.056 | 0.050 | 0.093 |
| | **E** | **0.033** | **0.010** | 0.149 | **0.007** | 0.088 | **0.084** | 0.078 | 0.050 | 0.070 |
| | **S** | 0.035 | 0.013 | 0.156 | 0.025 | **0.075** | **0.084** | **0.065** | **0.030** | **0.063** |
| | **C** | 0.039 | 0.020 | **0.146** | 0.124 | 0.076 | **0.084** | **0.065** | 0.041 | 0.069 |

**Tabla C.2:** (b) Valores de *diferencias relativas de accuracy* ($df = 1 - (Accuracy/MejorAccuracyxConjunto)$) por conjuntos de datos obtenidos con los métodos jerárquicos y plano. Para cada método de clasificación y conjunto de datos (del 11 al 18 de la tabla C.1) el mejor resultado está subrayado, y el mejor método que utiliza una jerarquía está resaltado en negrita.

| | | Conjunto de Datos | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Tecnica** | **Método** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **SVM** | **C-TD** | **0,259** | 0,092 | **0,039** | 0,005 | 0,009 | 0,156 | 0,031 | 0,001 |
| | **C-TDBU** | **0,259** | **0,089** | 0,043 | 0,001 | **0,004** | **0,154** | **0,027** | **0,000** |
| | **P-TD** | **0,259** | 0,092 | 0,048 | 0,003 | 0,009 | 0,169 | 0,046 | 0,001 |
| | **P-TDBU** | **0,259** | **0,089** | 0,051 | **0,000** | 0,009 | 0,163 | 0,039 | 0,001 |
| **RPART** | **C-TD** | 0,035 | 0,142 | **0,039** | 0,018 | **0,044** | 0,198 | 0,388 | 0,111 |
| | **C-TDBU** | 0,035 | 0,142 | **0,039** | 0,018 | **0,044** | 0,187 | 0,388 | 0,111 |
| | **P-TD** | **0,033** | **0,138** | 0,046 | 0,024 | **0,044** | 0,192 | **0,287** | **0,110** |
| | **P-TDBU** | **0,033** | **0,138** | 0,046 | **0,016** | **0,044** | **0,181** | **0,287** | **0,110** |
| **RF** | **C-TD** | 0,020 | 0,013 | 0,009 | 0,012 | 0,019 | 0,009 | **0,000** | **0,004** |
| | **C-TDBU** | **0,000** | 0,004 | 0,012 | **0,008** | 0,021 | 0,001 | **0,000** | **0,004** |
| | **P-TD** | 0,060 | 0,009 | **0,003** | 0,018 | **0,010** | **0,000** | 0,020 | 0,008 |
| | **P-TDBU** | 0,035 | **0,000** | **0,003** | **0,008** | 0,016 | 0,001 | 0,009 | 0,006 |
| **NNET** | **C-TD** | 0,199 | 0,086 | **0,000** | 0,003 | 0,013 | 0,170 | 0,570 | 0,132 |
| | **C-TDBU** | 0,217 | **0,072** | 0,006 | **0,001** | 0,010 | 0,145 | 0,438 | 0,156 |
| | **P-TD** | 0,208 | 0,083 | 0,006 | 0,011 | **0,000** | 0,158 | 0,179 | **0,078** |
| | **P-TDBU** | **0,167** | 0,083 | 0,009 | 0,011 | 0,010 | **0,135** | **0,166** | 0,092 |
| **NB** | **C-TD** | **0,891** | 0,154 | **0,219** | 0,314 | **0,055** | 0,507 | 0,510 | **0,228** |
| | **C-TDBU** | **0,891** | 0,154 | **0,219** | 0,314 | **0,055** | 0,507 | 0,498 | **0,228** |
| | **P-TD** | 0,918 | 0,150 | 0,251 | **0,287** | **0,055** | **0,458** | 0,490 | 0,236 |
| | **P-TDBU** | 0,918 | **0,149** | 0,251 | **0,287** | **0,055** | 0,486 | **0,466** | 0,235 |
| **C50** | **C-TD** | 0,127 | **0,093** | **0,031** | 0,021 | **0,046** | **0,105** | **0,100** | 0,039 |
| | **C-TDBU** | 0,127 | **0,093** | **0,031** | **0,016** | **0,046** | **0,105** | **0,100** | 0,039 |
| | **P-TD** | **0,102** | 0,096 | 0,034 | 0,030 | **0,046** | 0,129 | 0,103 | **0,037** |
| | **P-TDBU** | **0,102** | 0,096 | 0,034 | 0,026 | **0,046** | 0,129 | 0,102 | **0,037** |
| **KNN** | **C-TD** | **0,186** | 0,124 | 0,139 | 0,036 | 0,015 | 0,234 | **0,015** | **0,003** |
| | **C-TDBU** | 0,192 | 0,130 | 0,122 | **0,021** | 0,014 | 0,223 | **0,015** | **0,003** |
| | **P-TD** | 0,192 | **0,116** | 0,131 | 0,036 | **0,013** | 0,204 | 0,017 | **0,003** |
| | **P-TDBU** | 0,189 | 0,118 | **0,106** | 0,024 | 0,019 | **0,195** | 0,017 | **0,003** |

**Tabla C.3:** (a) Valores de diferencias relativas ($df = 1 - (Accuracy/MejorAccuracyxConjunto)$) obtenidos por los métodos multiclase jerárquicos para la generación de la jerarquía de clasificadores aplicado a los conjuntos de datos 1 al 8 de la tabla 5.4. La técnica que obtiene el mejor resultado para cada método y conjunto de datos se resalta en negrita.

| | | Conjunto de Datos | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Técnica** | **Método** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Promedio** |
| **SVM** | **C-TD** | 0,019 | 0,024 | 0,140 | 0,178 | 0,004 | 0,034 | **0,000** | 0,066 |
| | **C-TDBU** | 0,018 | **0,020** | 0,130 | 0,134 | **0,000** | 0,034 | 0,020 | **0,062** |
| | **P-TD** | **0,016** | 0,022 | 0,140 | 0,171 | 0,006 | 0,034 | 0,021 | 0,069 |
| | **P-TDBU** | **0,016** | **0,020** | **0,127** | **0,130** | 0,003 | **0,027** | 0,019 | 0,064 |
| **RPART** | **C-TD** | **0,072** | 0,035 | 0,189 | **0,113** | **0,146** | **0,046** | **0,109** | 0,112 |
| | **C-TDBU** | **0,072** | 0,035 | 0,189 | **0,113** | **0,146** | **0,046** | **0,109** | 0,112 |
| | **P-TD** | 0,074 | **0,027** | 0,187 | 0,117 | 0,151 | **0,046** | **0,109** | 0,106 |
| | **P-TDBU** | 0,074 | **0,027** | **0,186** | 0,117 | 0,151 | **0,046** | **0,109** | **0,104** |
| **RF** | **C-TD** | **0,000** | 0,003 | 0,001 | **0,007** | **0,013** | 0,027 | **0,000** | 0,009 |
| | **C-TDBU** | **0,000** | 0,002 | 0,001 | 0,019 | 0,014 | 0,015 | **0,000** | **0,007** |
| | **P-TD** | 0,009 | **0,000** | 0,008 | 0,028 | 0,031 | **0,000** | 0,020 | 0,015 |
| | **P-TDBU** | 0,007 | 0,001 | **0,000** | 0,033 | 0,022 | 0,004 | **0,000** | 0,010 |
| **NNET** | **C-TD** | 0,022 | 0,013 | 0,113 | 0,019 | 0,052 | 0,095 | 0,030 | 0,101 |
| | **C-TDBU** | 0,027 | 0,013 | 0,106 | 0,013 | 0,057 | 0,083 | 0,020 | 0,091 |
| | **P-TD** | **0,013** | 0,013 | 0,117 | **0,000** | **0,043** | 0,057 | **0,000** | 0,064 |
| | **P-TDBU** | 0,024 | **0,011** | **0,103** | 0,012 | 0,046 | **0,053** | 0,014 | **0,062** |
| **NB** | **C-TD** | **0,234** | 0,274 | 0,464 | 0,540 | **0,309** | **0,053** | **0,097** | 0,323 |
| | **C-TDBU** | **0,234** | 0,274 | **0,461** | 0,540 | **0,309** | **0,053** | **0,097** | **0,322** |
| | **P-TD** | **0,234** | **0,273** | 0,466 | **0,495** | 0,317 | **0,053** | 0,163 | 0,323 |
| | **P-TDBU** | **0,234** | **0,273** | 0,462 | 0,502 | 0,317 | **0,053** | 0,163 | 0,323 |
| **C50** | **C-TD** | 0,066 | 0,011 | 0,131 | 0,089 | 0,123 | **0,046** | **0,160** | 0,079 |
| | **C-TDBU** | **0,065** | 0,011 | **0,130** | 0,089 | 0,122 | **0,046** | **0,160** | 0,079 |
| | **P-TD** | 0,066 | **0,009** | 0,131 | **0,064** | 0,114 | **0,046** | **0,160** | 0,078 |
| | **P-TDBU** | 0,066 | **0,009** | **0,130** | **0,064** | **0,113** | **0,046** | **0,160** | **0,077** |
| **KNN** | **C-TD** | 0,020 | **0,041** | 0,161 | 0,313 | 0,013 | **0,019** | 0,143 | 0,097 |
| | **C-TDBU** | 0,022 | **0,041** | 0,155 | 0,308 | **0,012** | 0,027 | 0,142 | 0,095 |
| | **P-TD** | **0,018** | 0,044 | 0,161 | 0,321 | 0,014 | **0,019** | **0,122** | 0,094 |
| | **P-TDBU** | 0,024 | 0,044 | **0,154** | **0,291** | 0,014 | **0,019** | **0,122** | **0,089** |

**Tabla C.3:** (b) Valores de diferencias relativas ($df = 1 - (Accuracy/MejorAccuracyxConjunto)$) obtenidos por los métodos multiclase jerárquicos para la generación de la jerarquía de clasificadores aplicado a los conjuntos de datos 9 al 15 de la tabla 5.4. La técnica que obtiene el mejor resultado para cada método y conjunto de datos se resalta en negrita.

| Escenario 1 | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Datos** | **Métodos** | **Técnicas** | | | | | |
| | | **C50** | **KNN** | **NNET** | **RF** | **RPART** | **SVM** |
| **Dermatology** | **AIBS** | 0.916 | 0.708 | 0.905 | 0.924 | 0.888 | 0.839 |
| | **REE** | **0.948** | **0.802** | **0.961** | **0.972** | **0.919** | **0.921** |
| **Forest** | **AIBS** | 0.81 | 0.826 | 0.844 | 0.827 | 0.82 | 0.821 |
| | **REE** | **0.858** | **0.869** | **0.872** | **0.872** | **0.842** | **0.876** |
| **Frogs** | **AIBS** | 0.894 | 0.939 | 0.915 | 0.932 | 0.859 | 0.945 |
| | **REE** | **0.945** | **0.983** | **0.975** | **0.978** | **0.911** | **0.984** |
| **Glass** | **AIBS** | 0.622 | 0.603 | 0.574 | 0.704 | 0.548 | 0.609 |
| | **REE** | **0.652** | **0.631** | **0.638** | **0.759** | **0.604** | **0.648** |
| **Optdigits** | **AIBS** | 0.828 | 0.923 | 0.853 | 0.918 | 0.792 | 0.717 |
| | **REE** | **0.88** | **0.98** | **0.887** | **0.978** | **0.825** | **0.745** |
| **Pendigits** | **AIBS** | 0.889 | 0.946 | 0.796 | 0.945 | 0.822 | 0.948 |
| | **REE** | **0.951** | **0.99** | **0.809** | **0.989** | **0.87** | **0.994** |
| **Satimage** | **AIBS** | 0.807 | 0.858 | 0.85 | 0.865 | 0.81 | 0.869 |
| | **REE** | **0.842** | **0.888** | **0.884** | **0.903** | **0.833** | **0.889** |
| **Segmentation** | **AIBS** | 0.897 | 0.855 | 0.844 | 0.941 | 0.901 | 0.507 |
| | **REE** | **0.958** | **0.913** | **0.894** | **0.974** | **0.94** | **0.61** |
| **Sports** | **AIBS** | 0.576 | 0.478 | 0.393 | 0.651 | 0.559 | 0.586 |
| | **REE** | **0.601** | **0.499** | **0.402** | **0.687** | **0.571** | **0.602** |
| **Texture** | **AIBS** | 0.89 | 0.945 | 0.965 | 0.932 | 0.848 | 0.966 |
| | **REE** | **0.93** | **0.98** | **0.992** | **0.974** | **0.872** | **0.992** |
| **TrafficLight** | **AIBS** | 0.676 | 0.31 | 0.547 | 0.73 | 0.685 | 0.589 |
| | **REE** | **0.721** | **0.339** | **0.597** | **0.799** | **0.717** | **0.654** |
| **Vehicle** | **AIBS** | 0.65 | 0.551 | 0.658 | 0.662 | 0.633 | 0.678 |
| | **REE** | **0.681** | **0.614** | **0.715** | **0.744** | **0.682** | **0.747** |
| **Vertebral** | **AIBS** | 0.784 | 0.766 | 0.711 | 0.777 | 0.757 | 0.753 |
| | **REE** | **0.804** | **0.788** | **0.771** | **0.813** | **0.789** | **0.794** |
| **Vowel** | **AIBS** | 0.681 | 0.675 | 0.78 | 0.872 | 0.582 | 0.88 |
| | **REE** | **0.758** | **0.772** | **0.827** | **0.937** | **0.626** | **0.941** |
| **Zoo** | **AIBS** | 0.908 | 0.715 | 0.895 | 0.944 | 0.674 | 0.788 |
| | **REE** | **0.932** | **0.832** | **0.94** | **0.964** | **0.818** | **0.888** |

**Tabla C.4:** (a) Tasa de acierto obtenido por los métodos AIBS y REE, técnica de clasificación y conjunto de datos. La configuración de los experimentos se realiza bajo el escenario 1. El mejor resultado para cada conjunto de datos y técnica se destaca en negrita.

| Escenario 2 | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Datos** | **Métodos** | **Técnicas** | | | | | |
| | | **C50** | **KNN** | **NNET** | **RF** | **RPART** | **SVM** |
| **Dermatology** | **AIBS** | 0.919 | **0.703** | 0.907 | 0.93 | **0.89** | 0.833 |
| | **REE** | **0.938** | 0.698 | **0.964** | **0.97** | 0.883 | **0.885** |
| **Forest** | **AIBS** | 0.816 | 0.831 | 0.843 | 0.835 | 0.828 | 0.826 |
| | **REE** | **0.842** | **0.862** | **0.871** | **0.876** | **0.829** | **0.875** |
| **Frogs** | **AIBS** | 0.897 | 0.938 | 0.921 | 0.933 | 0.865 | 0.944 |
| | **REE** | **0.927** | **0.971** | **0.966** | **0.968** | **0.894** | **0.978** |
| **Glass** | **AIBS** | **0.618** | **0.622** | 0.619 | **0.704** | **0.579** | **0.629** |
| | **REE** | 0.602 | 0.598 | **0.624** | 0.702 | 0.575 | 0.595 |
| **Optdigits** | **AIBS** | **0.828** | 0.924 | 0.854 | 0.92 | **0.79** | **0.663** |
| | **REE** | 0.826 | **0.97** | **0.866** | **0.97** | 0.785 | 0.633 |
| **Pendigits** | **AIBS** | 0.904 | 0.945 | **0.825** | 0.946 | 0.844 | 0.955 |
| | **REE** | **0.922** | **0.979** | 0.785 | **0.98** | **0.853** | **0.991** |
| **Satimage** | **AIBS** | 0.809 | 0.862 | 0.85 | 0.869 | 0.811 | 0.874 |
| | **REE** | **0.827** | **0.88** | **0.876** | **0.895** | **0.815** | **0.887** |
| **Segmentation** | **AIBS** | 0.921 | 0.855 | **0.899** | 0.947 | 0.914 | **0.489** |
| | **REE** | **0.943** | **0.877** | 0.89 | **0.963** | **0.917** | 0.426 |
| **Sports** | **AIBS** | **0.579** | **0.484** | **0.414** | **0.656** | 0.565 | 0.594 |
| | **REE** | **0.579** | 0.476 | 0.374 | 0.653 | **0.567** | **0.595** |
| **Texture** | **AIBS** | 0.89 | 0.945 | 0.963 | 0.932 | 0.847 | 0.966 |
| | **REE** | **0.898** | **0.965** | **0.988** | **0.959** | **0.85** | **0.982** |
| **TrafficLight** | **AIBS** | **0.682** | **0.3** | **0.558** | 0.735 | **0.701** | **0.588** |
| | **REE** | 0.666 | 0.259 | 0.511 | **0.758** | 0.668 | 0.568 |
| **Vehicle** | **AIBS** | 0.653 | 0.555 | 0.675 | 0.67 | 0.628 | 0.657 |
| | **REE** | **0.656** | **0.598** | **0.692** | **0.729** | **0.64** | **0.688** |
| **Vertebral** | **AIBS** | 0.782 | 0.771 | 0.733 | 0.795 | 0.77 | 0.746 |
| | **REE** | **0.804** | **0.79** | **0.786** | **0.824** | **0.801** | **0.774** |
| **Vowel** | **AIBS** | **0.674** | **0.527** | **0.775** | **0.871** | **0.575** | **0.87** |
| | **REE** | 0.605 | 0.395 | 0.69 | 0.806 | 0.514 | 0.797 |
| **Zoo** | **AIBS** | **0.883** | **0.778** | 0.885 | 0.938 | **0.674** | **0.803** |
| | **REE** | 0.858 | 0.751 | **0.887** | **0.947** | 0.625 | 0.76 |

**Tabla C.4:** (b) Tasa de acierto obtenido por los métodos AIBS y REE, técnica de clasificación y conjunto de datos. La configuración de los experimentos se realiza bajo el escenario 2. El mejor resultado para cada conjunto de datos y técnica se destaca en negrita.

| **Técnicas** | | **Conjunto de Datos** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **C50** | **REE** | 0.54 | 0.85 | 0.69 | 0.71 | 0.91 | 0.59 | 0.85 | 0.84 | 0.90 | 0.81 |
| | **CBCE** | 0.62 | 0.92 | 0.74 | 0.82 | 0.90 | 0.62 | 0.90 | 0.83 | 0.90 | 0.78 |
| | **AIBS** | 0.59 | **0.94** | 0.76 | 0.84 | 0.93 | 0.60 | 0.91 | 0.83 | 0.92 | 0.83 |
| | **PAJI** | **0.64** | 0.93 | 0.75 | **0.85** | 0.93 | 0.64 | 0.91 | **0.87** | 0.92 | 0.83 |
| | **CEMP** | 0.63 | 0.93 | **0.76** | 0.84 | **0.94** | **0.64** | **0.92** | 0.87 | **0.94** | **0.84** |
| **RF** | **REE** | 0.63 | 0.92 | 0.72 | 0.81 | 0.95 | 0.62 | 0.91 | 0.90 | 0.96 | 0.83 |
| | **CBCE** | 0.73 | 0.93 | 0.74 | 0.84 | 0.93 | 0.70 | 0.91 | 0.92 | 0.95 | 0.83 |
| | **AIBS** | 0.71 | 0.97 | 0.76 | 0.88 | 0.97 | 0.71 | 0.93 | 0.97 | 0.98 | 0.90 |
| | **PAJI** | 0.75 | **0.98** | 0.77 | **0.88** | 0.97 | 0.73 | 0.94 | 0.97 | 0.98 | 0.90 |
| | **CEMP** | **0.75** | 0.97 | **0.77** | **0.88** | **0.97** | **0.73** | **0.94** | **0.98** | **0.99** | **0.91** |
| **RAPART** | **REE** | 0.40 | 0.85 | 0.68 | 0.72 | 0.86 | 0.51 | 0.83 | 0.80 | 0.85 | 0.65 |
| | **CBCE** | 0.49 | 0.89 | 0.72 | 0.83 | 0.87 | 0.58 | 0.87 | 0.79 | 0.84 | 0.70 |
| | **AIBS** | 0.50 | 0.88 | **0.73** | 0.83 | 0.89 | 0.58 | 0.88 | 0.79 | 0.85 | 0.74 |
| | **PAJI** | 0.50 | **0.91** | 0.73 | **0.84** | 0.90 | 0.58 | 0.88 | **0.82** | 0.87 | 0.73 |
| | **CEMP** | **0.51** | 0.91 | 0.73 | 0.84 | **0.91** | **0.59** | **0.89** | 0.82 | **0.88** | **0.74** |
| **SVM** | **REE** | 0.52 | 0.90 | 0.61 | 0.83 | 0.97 | 0.52 | 0.83 | 0.70 | 0.98 | 0.89 |
| | **CBCE** | 0.66 | 0.83 | 0.69 | 0.83 | 0.95 | 0.63 | 0.87 | 0.67 | 0.96 | 0.84 |
| | **AIBS** | 0.66 | 0.89 | 0.71 | 0.88 | 0.98 | 0.60 | 0.89 | 0.64 | 0.99 | 0.91 |
| | **PAJI** | 0.66 | 0.89 | 0.71 | 0.87 | 0.98 | **0.65** | 0.90 | 0.71 | 0.99 | 0.90 |
| | **CEMP** | **0.67** | **0.92** | **0.72** | **0.88** | **0.98** | 0.64 | **0.90** | **0.74** | **0.99** | **0.91** |
| **Técnicas** | | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **C50** | **REE** | 0.90 | 0.78 | 0.88 | 0.50 | 0.62 | 0.87 | 0.63 | 0.75 | 0.61 | 0.85 |
| | **CBCE** | 0.85 | 0.81 | 0.92 | 0.58 | 0.68 | 0.89 | 0.65 | 0.78 | 0.67 | 0.88 |
| | **AIBS** | 0.91 | 0.83 | 0.94 | 0.58 | 0.67 | 0.90 | 0.66 | 0.80 | 0.61 | 0.86 |
| | **PAJI** | 0.89 | **0.85** | 0.94 | **0.60** | **0.70** | 0.91 | 0.68 | 0.80 | 0.66 | 0.88 |
| | **CEMP** | **0.94** | 0.84 | **0.95** | 0.59 | 0.70 | **0.92** | **0.68** | **0.81** | **0.69** | **0.92** |
| **RF** | **REE** | 0.86 | 0.86 | 0.93 | 0.56 | 0.66 | 0.93 | 0.69 | 0.76 | 0.61 | 0.93 |
| | **CBCE** | 0.86 | 0.87 | 0.95 | 0.66 | 0.74 | 0.93 | 0.67 | 0.79 | 0.87 | 0.94 |
| | **AIBS** | 0.91 | 0.90 | 0.96 | 0.65 | 0.76 | 0.96 | 0.73 | **0.83** | 0.81 | 0.95 |
| | **PAJI** | 0.92 | 0.90 | 0.97 | **0.68** | **0.77** | 0.96 | 0.72 | 0.82 | **0.91** | 0.95 |
| | **CEMP** | **0.93** | **0.90** | **0.97** | 0.67 | 0.76 | **0.97** | **0.74** | 0.82 | 0.91 | **0.96** |
| **RAPART** | **REE** | 0.71 | 0.78 | 0.88 | 0.48 | 0.59 | 0.82 | 0.61 | 0.74 | 0.51 | **0.80** |
| | **CBCE** | 0.73 | 0.81 | 0.91 | 0.57 | **0.70** | 0.85 | 0.63 | 0.77 | 0.58 | 0.67 |
| | **AIBS** | 0.78 | 0.82 | 0.92 | 0.57 | 0.67 | 0.85 | 0.64 | **0.80** | 0.51 | 0.63 |
| | **PAJI** | 0.76 | 0.83 | 0.92 | **0.58** | **0.70** | 0.86 | **0.66** | **0.80** | **0.58** | 0.73 |
| | **CEMP** | **0.78** | **0.83** | **0.93** | 0.58 | 0.70 | **0.87** | 0.65 | 0.79 | 0.58 | 0.74 |
| **SVM** | **REE** | 0.61 | 0.83 | 0.87 | 0.39 | 0.31 | 0.96 | 0.67 | 0.76 | 0.68 | 0.85 |
| | **CBCE** | 0.66 | 0.87 | 0.92 | 0.59 | 0.59 | 0.97 | 0.66 | 0.76 | 0.87 | 0.81 |
| | **AIBS** | 0.68 | 0.89 | 0.93 | 0.60 | 0.54 | 0.98 | 0.68 | 0.77 | 0.80 | 0.77 |
| | **PAJI** | 0.68 | 0.89 | **0.94** | **0.60** | **0.62** | 0.98 | 0.71 | 0.78 | **0.90** | 0.84 |
| | **CEMP** | **0.71** | **0.89** | 0.94 | **0.60** | 0.60 | **0.99** | **0.74** | **0.80** | 0.88 | **0.87** |

**Tabla C.5:** Tasa de acierto por conjunto de datos y técnica de clasificación aplicando los métodos de adaptación REE, CBE, AIBS, PAJI y CEMP. El mejor resultado (más alto) por cada conjunto de datos se muestra en negrita .

| Técnicas | | Conjunto de Datos | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **C50** | **REE** | 3.514 | 0.16 | 2.504 | 0.166 | 1.443 | 0.118 | 0.425 | 1.262 | 0.799 | 6.317 |
| | **CBCE** | 0.556 | 0.016 | 0.517 | 0.023 | 0.261 | **0.006** | 0.086 | 0.151 | 0.068 | 1.321 |
| | **AIBS** | **0.088** | **0.015** | **0.26** | **0.019** | **0.067** | 0.007 | **0.03** | **0.05** | **0.024** | **0.35** |
| | **PAJI** | 1.928 | 0.067 | 1.25 | 0.076 | 0.813 | 0.044 | 0.23 | 0.638 | 0.305 | 3.828 |
| | **CEMP** | 0.459 | 0.02 | 0.45 | 0.025 | 0.252 | 0.008 | 0.08 | 0.13 | 0.06 | 1.105 |
| **RF** | **REE** | 10.914 | 0.295 | 36.824 | 0.343 | 4.444 | 0.148 | 2 | 6.677 | 4.635 | 31.289 |
| | **CBCE** | 4.755 | 0.045 | 9.041 | 0.126 | 0.997 | 0.021 | 0.546 | 1.126 | 0.691 | 11.669 |
| | **AIBS** | **0.448** | **0.031** | **4.93** | **0.055** | **0.271** | **0.013** | **0.13** | **0.313** | **0.204** | **3.269** |
| | **PAJI** | 5.576 | 0.146 | 20.278 | 0.184 | 2.3 | 0.062 | 1.06 | 3.248 | 2.119 | 17.445 |
| | **CEMP** | 2.725 | 0.081 | 8.965 | 0.113 | 1.414 | 0.029 | 0.505 | 0.9 | 0.6 | 12.205 |
| **RPART** | **REE** | 1.56 | 0.134 | 1.195 | 0.228 | 0.852 | 0.097 | 0.23 | 0.967 | 0.679 | 3.184 |
| | **CBCE** | 0.312 | **0.011** | 0.288 | 0.013 | 0.094 | **0.004** | 0.036 | 0.061 | 0.036 | 0.712 |
| | **AIBS** | **0.03** | **0.011** | **0.14** | **0.012** | **0.038** | 0.005 | **0.02** | **0.025** | **0.016** | **0.187** |
| | **PAJI** | 0.64 | 0.051 | 0.685 | 0.052 | 0.486 | 0.039 | 0.13 | 0.458 | 0.268 | 1.993 |
| | **CEMP** | 0.17 | 0.012 | 0.25 | 0.016 | 0.118 | 0.006 | 0.03 | 0.05 | 0.03 | 0.729 |
| **SVM** | **REE** | 20.095 | 0.136 | 117.615 | 0.134 | 1.471 | 0.098 | 0.9 | 5.031 | 1.58 | 47.524 |
| | **CBCE** | 3.264 | **0.013** | 58.841 | 0.018 | 0.225 | **0.005** | 0.201 | 1.481 | 0.18 | 10.29 |
| | **AIBS** | **0.099** | **0.013** | **18.305** | **0.014** | **0.046** | 0.006 | **0.03** | **0.096** | **0.03** | **1.798** |
| | **PAJI** | 12.78 | 0.061 | 80.29 | 0.065 | 0.889 | 0.039 | 0.51 | 3.325 | 0.774 | 31.28 |
| | **CEMP** | 1.506 | 0.018 | 45.49 | 0.021 | 0.223 | 0.007 | 0.18 | 1.32 | 0.16 | 10.578 |
| **Técnicas** | | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **C50** | **REE** | 3.413 | 0.523 | 0.555 | 0.815 | 0.121 | 1.025 | 0.164 | 0.068 | 0.3 | 0.117 |
| | **CBCE** | 0.859 | 0.11 | 0.066 | 0.151 | 0.01 | 0.189 | 0.021 | **0.004** | 0.021 | **0.007** |
| | **AIBS** | **0.207** | **0.03** | **0.023** | **0.024** | **0.008** | **0.04** | **0.018** | 0.007 | **0.01** | 0.008 |
| | **PAJI** | 2.025 | 0.234 | 0.162 | 0.412 | 0.049 | 0.465 | 0.078 | 0.023 | 0.08 | 0.036 |
| | **CEMP** | 0.619 | 0.1 | 0.05 | 0.14 | 0.011 | 0.17 | 0.027 | 0.009 | 0.02 | 0.009 |
| **RF** | **REE** | 31.761 | 1.665 | 1.313 | 3.124 | 0.187 | 2.99 | 0.517 | 0.114 | 0.49 | 0.124 |
| | **CBCE** | 8.914 | 0.461 | 0.341 | 0.757 | 0.051 | 0.776 | 0.134 | 0.044 | 0.084 | 0.009 |
| | **AIBS** | **2.149** | **0.154** | **0.113** | **0.175** | **0.016** | **0.13** | **0.104** | **0.022** | **0.02** | **0.008** |
| | **PAJI** | 15.89 | 0.846 | 0.577 | 1.561 | 0.083 | 1.46 | 0.29 | 0.058 | 0.17 | 0.044 |
| | **CEMP** | 8.016 | 0.42 | 0.16 | 0.69 | 0.038 | 0.79 | 0.18 | 0.033 | 0.08 | 0.017 |
| **RPART** | **REE** | 1.325 | 0.34 | 0.425 | 0.564 | 0.13 | 0.695 | 0.111 | 0.055 | 0.28 | 0.095 |
| | **CBCE** | 0.444 | 0.034 | 0.026 | 0.055 | **0.006** | 0.063 | **0.012** | **0.004** | 0.008 | **0.005** |
| | **AIBS** | **0.072** | **0.016** | 0.011 | **0.014** | **0.006** | **0.01** | 0.013 | 0.006 | **0.001** | 0.006 |
| | **PAJI** | 0.809 | 0.119 | 0.095 | 0.225 | 0.047 | 0.25 | 0.054 | 0.018 | 0.06 | 0.034 |
| | **CEMP** | 0.248 | 0.03 | **0.01** | 0.05 | 0.007 | 0.06 | 0.017 | 0.006 | 0.005 | 0.007 |
| **SVM** | **REE** | 102.125 | 0.468 | 0.532 | 1.598 | 0.114 | 1.18 | 0.142 | 0.057 | 0.3 | 0.1 |
| | **CBCE** | 44.438 | 0.124 | 0.061 | 0.582 | **0.007** | 0.203 | 0.025 | **0.005** | 0.016 | **0.004** |
| | **AIBS** | **3.946** | **0.022** | **0.02** | **0.032** | **0.007** | **0.02** | **0.016** | 0.007 | **0.001** | 0.007 |
| | **PAJI** | 72.111 | 0.208 | 0.172 | 0.909 | 0.05 | 0.59 | 0.076 | 0.02 | 0.08 | 0.03 |
| | **CEMP** | 18.825 | 0.11 | 0.05 | 0.55 | 0.008 | 0.2 | 0.029 | 0.008 | 0.019 | 0.008 |

**Tabla C.6:** Tiempos en segundos medidos al aplicar los métodos REE, CBE, AIBS, PAJI y CEMP. El mejor resultado (más bajo) por cada conjunto de datos se muestra en negrita.

| **Técnicas** | **Tasa de acierto (diferencias relativas)** | | | | |
|---|---|---|---|---|---|
| | **REE** | **CBE** | **AIBS** | **PAJI** | **CEMP** |
| **C50** | 0.101 | *0.148* | 0.108 | 0.084 | **0.076** |
| **RF** | 0.023 | *0.094* | 0.049 | 0.009 | **0.005** |
| **RPART** | 0.158 | *0.208* | 0.160 | 0.139 | **0.135** |
| **SVM** | 0.107 | *0.176* | 0.114 | 0.082 | **0.073** |

**Tabla C.7:** Promedio de diferencias relativas (df=1(Accuracy/MejorAccuracyxMetodo)) de accuracy de los métodos CBE, AIBS, REE, PAJI y CEMP agrupados por técnica de aprendizaje. El mejor resultado (más bajo) por cada conjunto de datos se destaca en negrita y el peor resultado (más alto) se destaca en cursiva.

| **Técnicas** | **Tiempo (diferencias relativas)** | | | | |
|---|---|---|---|---|---|
| | **REE** | **CBCE** | **AIBS** | **PAJI** | **CEMP** |
| **C50** | *0.000* | 0.860 | **0.938** | 0.551 | 0.865 |
| **RF** | *0.000* | 0.749 | **0.911** | 0.514 | 0.761 |
| **RPART** | *0.000* | 0.889 | **0.947** | 0.584 | 0.899 |
| **SVM** | *0.000* | 0.814 | **0.945** | 0.497 | 0.833 |

**Tabla C.8:** Promedio de las diferencias relativas de tiempos de adaptación ($df = 1 - (Tiempo/MejorTiempoxMetodo)$) con respecto al tiempo en segundos requerido por los métodos CBE, AIBS, REE, PAJI y CEMP agrupado por técnica de aprendizaje usada para entrenar los clasificadores. El mejor resultado (más alto) se destaca en negrita y el peor resultado (más bajo) se destaca en cursiva.

# Bibliografía


[AGS20] Shereen Afifi, Hamid GholamHosseini y Roopak Sinha. "Dynamic hardware system for cascade SVM classification of melanoma". En: *Neural Computing and Applications* 32.6 (2020), págs. 1777-1788 (vid. pág. 35).

[AGT07] Gidudu Anthony, Hulley Gregg y Marwala Tshilidzi. "Image classification using SVMs: one-against-one vs one-against-all". En: *arXiv preprint arXiv:0711.2914* (2007) (vid. pág. 46).

[Aha87] David Aha. *UCI Machine Learning Repository*. 1987 (vid. págs. 73, 74, 77, 96, 99).

[AKA91] David Aha, Dennis Kibler y Marc Albert. "Instance-based learning algorithms". En: *Machine Learning* 6.1 (1991), págs. 37-66 (vid. pág. 34).

[Al-+19] Bassam Al-Salemi y col. "Multi-label arabic text categorization: A benchmark and baseline comparison of multi-label learning algorithms". En: *Information Processing & Management* 56.1 (2019), págs. 212-227 (vid. pág. 35).

[And14] Michael Anderberg. *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*. Vol. 19. 2014 (vid. pág. 14).

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant y col. "Fast algorithms for mining association rules". En: *Proceedings of the 20th International Conference on Very Large Data Bases*. Vol. 1215. 1994, págs. 487-499 (vid. pág. 60).

[Bay91] Thomas Bayes. "An essay towards solving a problem in the doctrine of chances. 1763." En: *MD Computing: Computers in Medical Practice* 8.3 (1991), pág. 157 (vid. pág. 27).

[BBS11] Florian Brucker, Fernando Benites y Elena Sapozhnikova. "Multilabel classification and extracting predicted class hierarchies". En: *Pattern Recognition* 44.3 (2011), págs. 724-738 (vid. pág. 60).

[Bel+10] Antonio Bella y col. "Calibration of machine learning models". En: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. 2010, págs. 128-146 (vid. pág. 40).

[BF98] Paul Bradley y Usama Fayyad. "Refining initial points for k-means clustering." En: *International Conference on Machine Learning*. Vol. 98. 1998, págs. 91-99 (vid. pág. 13).

[Bif+10] Albert Bifet y col. "MOA: Massive Online Analysis". En: *Journal of Machine Learning Research* 11 (2010), págs. 1601-1604 (vid. pág. 125).

[Bla19] Marcin Blachnik. "Ensembles of instance selection methods: A comparative study". En: *International Journal of Applied Mathematics and Computer Science* 29.1 (2019), págs. 151-168 (vid. pág. 40).

[Bre+98] Leo Breiman y col. "Arcing classifier". En: *The Annals of Statistics* 26.3 (1998), págs. 801-849 (vid. pág. 43).

[Bre01] Leo Breiman. "Random forests". En: *Machine Learning* 45.1 (2001), págs. 5-32 (vid. pág. 42).

[Bre96] Leo Breiman. "Bagging predictors". En: *Machine Learning* 24.2 (1996), págs. 123-140 (vid. pág. 42).

[Bro90] Carla Brodley. *Image segmentation data*. Vision Group, University of Massachusetts. 1990. URL: `https://archive.ics.uci.edu/ml/datasets/Image+Segmentation` (vid. págs. 96, 99).

[Cap17] Fernando Caparrini. *Introducción al Aprendizaje Automático*. Universidad de Sevilla. 2017. URL: `http://www.cs.us.es/~fsancho/?e=75` (vid. pág. 9).

[CC06] Michael Carney y Pádraig Cunningham. "Making good probability estimates for regression". En: *European Conference on Machine Learning*. 2006, págs. 582-589 (vid. pág. 40).

[CG16] Tianqi Chen y Carlos Guestrin. "XGBoost: A scalable tree boosting system". En: *Conference on Knowledge Discovery and Data Mining*. 22. 2016, págs. 785-794 (vid. pág. 43).

[CH67] Thomas Cover y Peter Hart. "Nearest neighbor pattern classification". En: *Transactions on Information Theory* 13.1 (1967), págs. 21-27 (vid. pág. 33).

[Cha+02] Nitesh Chawla y col. "SMOTE: Synthetic minority over-sampling technique". En: *Journal of Artificial Intelligence Research* 16 (2002), págs. 321-357 (vid. pág. 40).

[Cha16] Manoj Chandak. "Role of big-data in classification and novel class detection in data streams". En: *Journal of Big Data* 3.1 (2016), pág. 5 (vid. pág. 113).

[Che+19] Chengpeng Chen y col. "Hybrid incremental learning of new data and new classes for hand-held object recognition". En: *Journal of Visual Communication and Image Representation* 58 (2019), págs. 138-148 (vid. págs. 114, 122).

[CM07] Michelangelo Ceci y Donato Malerba. "Classifying web documents in a hierarchy of categories: a comprehensive study". En: *Journal of Intelligent Information Systems* 28.1 (2007), págs. 37-78 (vid. pág. 54).

[CMO05] Gail Carpenter, Siegfried Martens y Ogi Ogas. "Self-organizing information fusion and hierarchical knowledge discovery: a new framework using ARTMAP neural networks". En: *Neural Networks* 18.3 (2005), págs. 287-295 (vid. págs. 59, 60).

[CN89] Peter Clark y Tim Niblett. "The CN2 induction algorithm". En: *Machine Learning* 3.4 (1989), págs. 261-283 (vid. pág. 32).

[Com04] Wikimedia Commons. *Red Neuronal Artificial*. 2004. URL: `https://commons.wikimedia.org/wiki/File:RedNeuronalArtificial.png` (vid. pág. 37).

[Com07] Wikimedia Commons. *Example of k-nearest neighbour classification*. 2007. URL: `https://commons.wikimedia.org/wiki/File:KnnClassification.svg` (vid. pág. 34).

[CP01] Gert Cauwenberghs y Tomaso Poggio. "Incremental and decremental SVM learning". En: *Advances in Neural Information Processing Systems*. 2001, págs. 409-415 (vid. pág. 110).

[CS99] William Cohen y Yoram Singer. "Context-sensitive learning methods for text categorization". En: *Transactions on Information Systems* 17.2 (1999), págs. 141-173 (vid. pág. 32).

[CV95] Corinna Cortes y Vladimir Vapnik. "Support-vector networks". En: *Machine Learning* 20.3 (1995), págs. 273-297 (vid. pág. 34).

[DA98] Rakesh Dugad y Narendra Ahuja. "Unsupervised multidimensional hierarchical clustering". En: *Acoustics, Speech and Signal Processing*. Vol. 5. IEEE. 1998, págs. 2761-2764 (vid. pág. 71).

[Dem06] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets". En: *Journal of Machine Learning Research* 7 (2006), págs. 1-30 (vid. pág. 26).

[Dew+13] Farid Dewan y col. "An adaptive ensemble classifier for mining concept drifting data streams". En: *Expert Systems with Applications* 40.15 (2013), págs. 5895-5906 (vid. págs. 113, 122).

[Die00] Thomas G. Dietterich. "Ensemble methods in machine learning". En: *Multiple Classifier Systems*. 2000, págs. 1-15 (vid. págs. 40, 42).

[DL97] Manoranjan Dash y Huan Liu. "Feature selection for classification". En: *Intelligent Data Analysis* 1.3 (1997), págs. 131-156 (vid. pág. 40).

[Dra19] Georgios Drakos. *Support Vector Machine vs Logistic Regression*. 2019. URL: `https://cdn-images-1.medium.com/max/800/1*nUpw5agP-Vefm4Uinteq-A.png` (vid. pág. 35).

[Dun61] Olive Dunn. "Multiple comparisons among means". En: *Journal of the American Statistical Association* 56.293 (1961), págs. 52-64 (vid. pág. 27).

[DYZ14] Qing Da, Yang Yu y Zhi-Hua Zhou. "Learning with augmented class by exploiting unlabeled data". En: *Conference on Artificial Intelligence*. 28. 2014 (vid. pág. 113).

[Esp+97] Floriana Esposito y col. "A comparative analysis of methods for pruning decision trees". En: *Transactions on Pattern Analysis and Machine Intelligence* 19.5 (1997), págs. 476-491 (vid. pág. 31).

[Fan11] Rong-En Fan. *LIBSVM Data*. National Taiwan University. 2011. URL: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/` (vid. pág. 77).

[Fer16] Cèsar Ferri. "Identifying the sport activity of GPS tracks". En: *Procedia Computer Science* 80 (2016). International Conference on Computational Science, págs. 301-312 (vid. pág. 52).

[FHM09] Cèsar Ferri, José Hernández y R. Modroiu. "An experimental comparison of performance measures for classification". En: *Pattern Recognition Letters* 30.1 (2009), págs. 27-38 (vid. pág. 18).

[FHR01] Cèsar Ferri, José Hernandez y María José Ramírez. "Incremental learning of functional logic programs". En: *International Sympo-*

*sium on Functional and Logic Programming*. 2001, págs. 233-247 (vid. pág. 110).

[Fis56] Ronald Fisher. "Statistical methods and scientific inference." En: (1956) (vid. pág. 23).

[For90] Richard Forsyth. *Zoo dataset*. 1990. URL: `https://archive.ics.uci.edu/ml/datasets/Zoo` (vid. pág. 73).

[Fri01] Jerome Friedman. "Greedy function approximation: a gradient boosting machine". En: *Annals of Statistics* (2001), págs. 1189-1232 (vid. pág. 43).

[Fri37] Milton Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". En: *Journal of the American Statistical Association* 32.200 (1937), págs. 675-701 (vid. pág. 24).

[FS07] Tiziano Fagni y Fabrizio Sebastiani. "On the selection of negative examples for hierarchical text categorization". En: *Language and Technology Conference*. 2007, págs. 24-28 (vid. págs. 54, 120).

[Gir00] Christophe Giraud. "A note on the utility of incremental learning". En: *Ai Communications* 13.4 (2000), págs. 215-223 (vid. pág. 110).

[God02] Shantanu Godbole. *Exploiting Confusion Matrices for Automatic Generation of Topic Hierarchies and Scaling Up Multi-Way Classifiers*. 2002 (vid. pág. 61).

[GS09] Xin Geng y Kate Smith. "Incremental learning". En: *Encyclopedia of Biometrics*. 2009, págs. 731-735 (vid. pág. 42).

[GS91] Fred Galvin y Shore. "Distance functions and topologies". En: *The American Mathematical Monthly* 98.7 (1991), págs. 620-623 (vid. pág. 62).

[GSC02] Shantanu Godbole, Sunita Sarawagi y Soumen Chakrabarti. "Scaling multi-class support vector machines using inter-class confu-

sion". En: *International Conference on Knowledge Discovery and Data Mining*. 2002, págs. 513-518 (vid. págs. 60, 61, 65, 77).

[Han+19] Jian Han y col. "Using decision tree to predict response rates of consumer satisfaction, attitude, and loyalty surveys". En: *Sustainability* 11.8 (2019) (vid. pág. 31).

[Her+16] José Hernández y col. "Reframing in context: A systematic approach for model reuse in Machine Learning". En: *AI Communications* 29.5 (2016), págs. 551-566 (vid. págs. 109, 110).

[Hol79] Sture Holm. "A simple sequentially rejective multiple test procedure". En: *Scandinavian Journal of Statistics* (1979), págs. 65-70 (vid. págs. 27, 107).

[Hom88] Gerhard Hommel. "A stagewise rejective multiple test procedure based on a modified Bonferroni test". En: *Biometrika* 75.2 (1988), págs. 383-386 (vid. pág. 27).

[HP81] Henderson Harold y Velleman Paul. "Building multiple regression models interactively". En: *Biometrics* 37.2 (1981), págs. 391-411 (vid. págs. 14, 15).

[ID80] Ronald Iman y James Davenport. "Approximations of the critical region of the fbietkan statistic". En: *Communications in Statistics-Theory and Methods* 9.6 (1980), págs. 571-595 (vid. pág. 26).

[JL62] J. L. Hodges Jr. y E. L. Lehmann. "Rank methods for combination of independent experiments in analysis of variance". En: *The Annals of Mathematical Statistics* 33.2 (1962), págs. 482-497 (vid. pág. 26).

[JS11] Nathalie Japkowicz y Mohak Shah. *Evaluating learning algorithms: A classification perspective*. 2011 (vid. pág. 23).

[JT13] Bangsuk Jantawan y Cheng-Fa Tsai. "The application of data mining to build classification model for predicting graduate employment". En: *International Journal of Computer Science and Information Security* 11.10 (2013) (vid. pág. 31).

[KG11] Milan Kumari y Sunila Godara. "Comparative study of data mining classification methods in cardiovascular disease prediction 1". En: *International Journal of Computer Science and Technology* 2.2 (2011) (vid. pág. 31).

[KM] ACM Special Interest Group on Knowledge Discovery y Data Mining. *annual Data Mining and Knowledge Discovery competition.* URL: `https://www.kdd.org/kdd-cup` (vid. pág. 42).

[Kon98] Igor Kononenko. "The minimum description length based decision tree pruning". En: *PRICAI'98: Topics in Artificial Intelligence.* 1998, págs. 228-237 (vid. pág. 31).

[Kub17] Miroslav Kubat. *An introduction to machine learning.* 2017 (vid. pág. 41).

[Kuh+14] Max Kuhn y col. *caret: Classification and Regression Training.* R package version 6.0-30. 2014 (vid. pág. 78).

[Li+18] Jundong Li y col. "Feature Selection: A Data Perspective". En: *ACM Computing Surveys* 50.6 (2018). DOI: `10.1145/3136625` (vid. pág. 40).

[Li08] Jianjun David Li. "A two-step rejection procedure for testing multiple hypotheses". En: *Journal of Statistical Planning and Inference* 138.6 (2008), págs. 1521-1527 (vid. pág. 27).

[Loh11] Wei-Yin Loh. "Classification and regression trees". En: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), págs. 14-23 (vid. pág. 29).

[LZO07] Tao Li, Shenghuo Zhu y Mitsunori Ogihara. "Hierarchical document classification using automatically generated hierarchy". En: *Journal of Intelligent Information Systems* 29.2 (2007), págs. 211-230 (vid. págs. 58, 59, 77).

[MBB13] Martial Mermillod, Aurélia Bugaiska y Patrick Bonin. "The stability-plasticity dilemma: investigating the continuum from catastrophic

forgetting to age-limited learning effects". En: *Frontiers in Psychology*. 2013 (vid. pág. 110).

[MGC09] G. Madzarov, D. Gjorgjevikj e I. Chorbev. "A multi-class SVM classifier utilizing binary decision tree". En: *Informatica* 33.2 (2009), págs. 225-233 (vid. pág. 44).

[Min89] John Mingers. "An empirical comparison of pruning methods for decision tree induction". En: *Machine Learning* 4.2 (1989), págs. 227-243 (vid. pág. 31).

[MLH98] Bing Ma, Bing Liu y Yiming Hsu. "Integrating classification and association rule mining". En: *International Conference on Knowledge Discovery and Data Mining*. 1998 (vid. pág. 32).

[MT14] Giovanna Menardi y Nicola Torelli. "Training and assessing classification rules with imbalanced data". En: *Data Mining and Knowledge Discovery* 28.1 (2014), págs. 92-122 (vid. pág. 40).

[MTP09] Michael Muhlbaier, Apostolos Topalis y Robi Polikar. "Learn $^{++}$.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes". En: *Transactions on Neural Networks* 20.1 (2009), págs. 152-168 (vid. págs. 113, 122).

[MTZ17] Xin Mu, Kai Ting y Zhi Zhou. "Classification under streaming emerging new classes: A solution using completely-random trees". En: *Transactions on Knowledge and Data Engineering* 29.8 (2017), págs. 1605-1618 (vid. pág. 114).

[Olv+10] Arturo Olvera y col. "A review of instance selection methods". En: *Artificial Intelligence Review* 34.2 (2010), págs. 133-143 (vid. pág. 40).

[OM99] David Opitz y Richard Maclin. "Popular ensemble methods: An empirical study". En: *Journal of Artificial Intelligence Research* 11 (1999), págs. 169-198 (vid. pág. 41).

[Pan+18] Evi Septiana Pane y col. "Identifying severity level of cybersickness from EEG signals using CN2 rule induction algorithm". En: *International Conference on Intelligent Informatics and Biomedical Sciences*. Vol. 3. IEEE. 2018, págs. 170-176 (vid. pág. 33).

[Par+19] Antonio Parmezan y col. "Towards hierarchical classification of data streams". En: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. 2019, págs. 314-322 (vid. pág. 125).

[PCS99] John C. Platt, Nello Cristianini y John Shawe-Taylor. "Large margin DAGs for multiclass classification". En: *International Conference on Neural Information Processing Systems*. 1999, págs. 547-553 (vid. pág. 46).

[PG09] Su-lin Pang y Ji-zhang Gong. "C5. 0 classification algorithm and application on individual credit evaluation of banks". En: *Systems Engineering-Theory & Practice* 29.12 (2009), págs. 94-104 (vid. pág. 31).

[Pol+01] Robi Polikar y col. "Learn++: An incremental learning algorithm for supervised neural networks". En: *IEEE Transactions on Systems* 31.4 (2001), págs. 497-508 (vid. pág. 112).

[Qua79] D. Quade. "Using weighted rankings in the analysis of complete blocks with additive block effects". En: *Journal of the American Statistical Association* 74 (1979), págs. 680-683 (vid. pág. 26).

[Qui14] John Ross Quinlan. *C4. 5: Programs for Machine Learning*. 2014 (vid. pág. 31).

[Qui86] John Ross Quinlan. "Induction of decision trees". En: *Machine Learning* 1.1 (1986), págs. 81-106 (vid. pág. 31).

[Qui87] Jhon Ross Quinlan. "Simplifying decision trees". En: *International Journal of Man-Machine Studies* 27.3 (1987), págs. 221-234 (vid. pág. 31).

[R C15] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. 2015 (vid. pág. 78).

[RK04] Ryan Rifkin y Aldebaro Klautau. "In defense of one-vs-all classification". En: *Journal of Machine Learning research* 5 (2004), págs. 101-141 (vid. pág. 45).

[RM09] Lior Rokach y Oded Maimon. "Data mining using decomposition methods". En: *Data Mining and Knowledge Discovery Handbook*. 2009, págs. 981-998 (vid. pág. 43).

[SB14] Shai Shalev y Shai Ben. *Understanding Machine Learning: From Theory to Algorithms*. 2014 (vid. págs. 16, 30).

[SB18] Richard Sutton y Andrew Barto. *Reinforcement learning: An introduction*. 2018 (vid. pág. 11).

[SC09] Dan Steinberg y Phillip Colla. "CART: Classification and regression trees". En: *The Top Ten Algorithms in Datamining* 9 (2009), pág. 179 (vid. pág. 31).

[SF07] Vincent Schickel y Boi Faltings. "Using hierarchical clustering for learning the ontologies used in recommendation systems". En: *International Conference on Knowledge Discovery and Data Mining*. 2007, págs. 599-608 (vid. pág. 59).

[SF11] Carlos Silla y Alex Freitas. "A survey of hierarchical classification across different application domains". En: *Data Mining and Knowledge Discovery* 22.1 (2011), págs. 31-72 (vid. págs. 2, 49).

[SFR17] Daniel Silva, Cèsar Ferri y María José Ramírez. "Improving performance of multiclass classification by inducing class hierarchies". En: *International Conference on Computer Science*. ICCS. 2017, págs. 1692-1701 (vid. pág. 5).

[SFR18a] Daniel Silva, Cèsar Ferri y María José Ramírez. "Adapting hierarchical multiclass classification to changes in the target concept".

En: *Conference of the Spanish Association for Artificial Intelligence*. 2018, págs. 118-127 (vid. pág. 6).

[SFR18b] Daniel Silva, Cèsar Ferri y María José Ramírez. "Probabilistic class hierarchies for multiclass classification". En: *Journal of Computational Science* 26 (2018), págs. 254-263 (vid. pág. 5).

[Sug16] Masashi Sugiyama. "Chapter 33 - Semisupervised learning". En: *Introduction to Statistical Machine Learning*. 2016, págs. 375-390 (vid. pág. 11).

[Tan95] Ah Tan. "Adaptive resonance associative map". En: *Neural Networks* 8.3 (1995), págs. 437-446 (vid. pág. 60).

[Tao+16] Qin Tao y col. "Robust face detection using local CNN and SVM based on kernel combination". En: *Neurocomputing* 211 (2016), págs. 98-105 (vid. pág. 35).

[TE93] Robert Tibshirani y Bradley Efron. "An introduction to the bootstrap". En: *Monographs on Statistics and Applied Probability* 57 (1993), págs. 1-436 (vid. pág. 40).

[WAM97] Dietrich Wettschereck, David Aha y Takao Mohri. "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms". En: *Artificial Intelligence Review* 11.1-5 (1997), págs. 273-314 (vid. pág. 34).

[Wil92] Frank Wilcoxon. "Individual comparisons by ranking methods". En: *Breakthroughs in Statistics*. 1992, págs. 196-202 (vid. pág. 23).

[Wol92] David H. Wolpert. "Stacked generalization". En: *Neural Networks* 5.2 (1992), págs. 241-259 (vid. pág. 43).

[XDP18] Pu Xu, Zhijun Ding y MeiQin Pan. "A hybrid interpretable credit card users default prediction model based on RIPPER". En: *Concurrency and Computation: Practice and Experience* 30.23 (2018), e4445 (vid. pág. 33).

[XKS92] L. Xu, A. Krzyzak y C. Y. Suen. "Methods of combining multiple classifiers and their applications to handwriting recognition". En: *Transactions on Systems, Man, and Cybernetics* 22.3 (1992), págs. 418-435 (vid. pág. 120).

[Yan01] Yiming Yang. "A study of thresholding strategies for text categorization". En: *International Conference on Research and Development in Information Retrieval*. 2001, págs. 137-145 (vid. pág. 40).

[ZJY21] H. Zhang, L. Jiang y L. Yu. "Attribute and instance weighted naive Bayes". En: *Pattern Recognition* 111 (2021) (vid. pág. 29).

[ZSX06] Bo Zhang, Jin Su y Xin Xu. "A class-incremental learning method for multi-class support vector machines in text classification". En: *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*. 2006, págs. 2581-2585 (vid. págs. 114, 122).