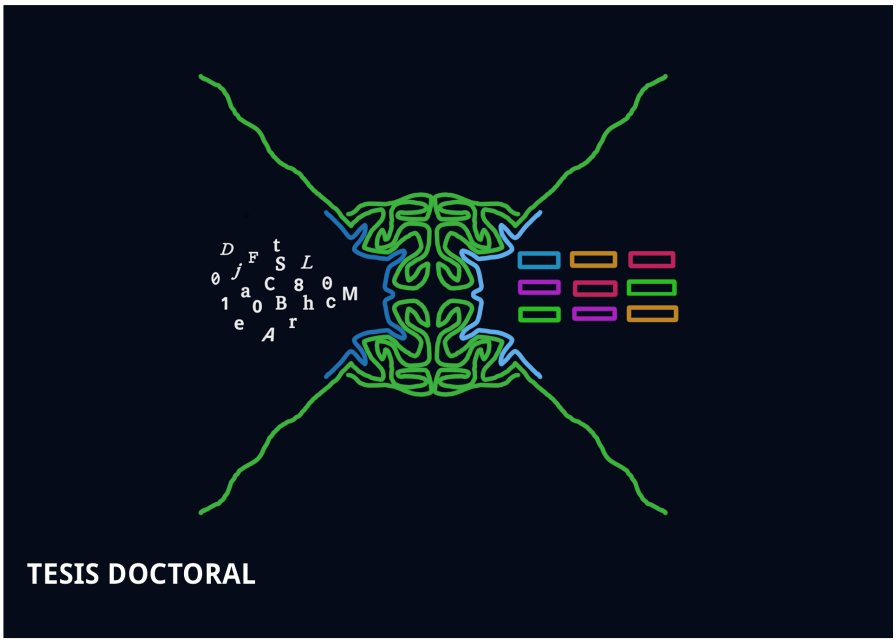


Nuevos métodos híbridos de computación flexible para clasificación multietiqueta

Francisco Charte Ojeda

Directores de tesis
Dr. D. Antonio Jesús Rivera Rivas
Dra. Dª María José del Jesus Díaz
Dr. D. Francisco Herrera Triguero

Abril 2015

Programa Oficial de Doctorado en
Tecnologías de la Información y la Comunicación

Universidad de Granada

Nuevos métodos híbridos de computación flexible para clasificación multietiqueta

Francisco Charte Ojeda
<fcharte@ugr.es>

Tesis para la obtención del título de doctor
por la Universidad de Granada

Programa Oficial de Doctorado en
Tecnologías de la Información y la Comunicación

Departamento de Ciencias de la Computación e Inteligencia Artificial
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Directores de tesis

Dr. D. Antonio Jesús Rivera Rivas
Grupo de Sistemas Inteligentes y Minería de Datos, Universidad de Jaén

Dra. Dª María José del Jesus Díaz
Grupo de Sistemas Inteligentes y Minería de Datos, Universidad de Jaén

Dr. D. Francisco Herrera Triguero
Grupo de Soft Computing y Sistemas de Información Inteligentes, Universidad de Granada

Granada, abril de 2015

Tesis doctoral subvencionada por el programa predoctoral de Formación del Profesorado Universitario (FPU) del Ministerio de Educación (Ref. **AP2010-0068**) según convocatoria publicada en BOE de 24 de enero de 2011 y resolución publicada en BOE de 20 de diciembre de 2011.

Lo que sabemos es una gota de agua, lo que ignoramos es el océano.
(Isaac Newton)

Resumen

La presente tesis aborda el estudio de nuevas técnicas de tratamiento de los datos con el objetivo de mejorar el funcionamiento de los sistemas de clasificación multietiqueta. La motivación de este trabajo está en el cada vez mayor número de campos de aplicación de dicho tipo de clasificación, a raíz de la necesidad de etiquetar documentos de todo tipo: textos, imágenes, vídeos, música, etc., y su utilidad en otros campos como la medicina y la genética, especialmente la predicción de funciones de proteínas.

En el desarrollo de la tesis se siguen fundamentalmente dos estrategias: aprovechar la información de correlación entre etiquetas a fin de reducir la dimensionalidad del espacio de salida, por una parte, y analizar las características específicas de los conjuntos de datos multietiqueta a fin de proponer algoritmos de preprocesamiento a medida, para reducir el desequilibrio entre etiquetas y mejorar el rendimiento de los clasificadores. El trabajo en estas dos vías ha llevado al diseño y desarrollo de múltiples algoritmos recogidos en la tesis, cuya finalidad se resume a continuación:

- LI-MLC: Es un método en el que se hibrida un algoritmo de minería de reglas de asociación con métodos de clasificación multietiqueta existentes, reduciendo la dimensionalidad del espacio de etiquetas a fin de mejorar el rendimiento y la eficiencia.
- LP-ROS/LP-RUS: Métodos de remuestreo aleatorio basados en la técnica de transformación para conjuntos de datos multietiqueta conocida como LP (Label Powerset).
- ML-ROS/ML-RUS: Métodos de remuestreo aleatorio basados en el análisis individual de la frecuencia de aparición de cada etiqueta en el conjunto de datos.
- MLSMOTE: Algoritmo de generación de instancias sintéticas para conjuntos de datos multietiqueta.
- MLeNN: Algoritmo de eliminación de instancias basado en la regla del vecino más cercano.
- REMEDIAL: Método de preprocesamiento que permite mejorar el rendimiento en clasificación mediante la separación de etiquetas con desbalanceo.

Además de los citados algoritmos, junto con su correspondiente experimentación, la tesis también propone múltiples medidas de caracterización para conjuntos de datos multietiqueta y el análisis justificado sobre su utilidad y aplicación. Toda esta información puede obtenerse fácilmente gracias al paquete `mldr` para el entorno R, desarrollado asimismo como parte de la presente tesis.

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos

Δ	Diferencia simétrica entre los operandos
$[\![expr]\!]$	Devuelve 1 si *expr* es cierta
$\mid D \mid$	Número de instancias que contiene D
$\mid L \mid$	Número total de etiquetas que contiene L
$\mid X \mid$	Número de atributos en X
C	Clasificador multietiqueta
D	Conjunto de datos multietiqueta
f	Número de atributos de entrada
L	Conjunto de las etiquetas presentes en D
X	Conjunto de los atributos de entrada en D
X^j	Espacio de los posibles valores del atributo j-ésimo
Y_i	Subconjunto de L presente en la *i-ésima* instancia de D
Z_i	Subconjunto de L predicho por un clasificador para la *i-ésima* instancia de D
FN	Número de falsos negativos
FP	Número de falsos positivos
TN	Número de verdaderos negativos
TP	Número de verdaderos positivos

Agradecimientos

Al escribir estas líneas, las últimas tras finalizar la redacción de la presente memoria de tesis que es el resumen de varios años de trabajo, resulta casi inevitable mirar atrás y tratar de analizar y valorar lo conseguido, lo que se quería conseguir y lo que queda por hacer.

El esfuerzo puesto en el desarrollo de los trabajos que se describen aquí ha sido enorme y, en contra de lo que cabría suponer, no ha sido un sacrificio individual, detrás está el trabajo de un grupo de personas sin las cuales se me antoja imposible haber llegado hasta aquí. Nunca podré agradecer lo suficiente el apoyo incondicional, la confianza y el empuje continuo que me han entregado mis directores de tesis, Antonio, María José y Paco. Gracias.

Hago extensible este agradecimiento a mis compañeros, miembros del grupo de investigación SIMIDAT de la Universidad de Jaén al que pertenezco, por estar siempre ahí a mi disposición.

Por último quiero expresar mi gratitud a María Jesús, David y Alejandro, mi familia, cuyo soporte diario me eleva para llegar cada vez más alto. Gracias, sabed que una buena parte de este trabajo es vuestro pues vosotros me habéis dado las fuerzas para realizarlo.

Francisco Charte Ojeda

Granada, abril 2015.

Introducción

El volumen de información generada diariamente por usuarios, instituciones y empresas se incrementa siguiendo una progresión exponencial, publicándose cada día miles de entradas en foros, blogs y redes sociales en las que se incluyen textos, imágenes, vídeos, música, etc. Categorizar toda esa información demanda el uso de sistemas automáticos de clasificación, dado que el coste de hacerlo manualmente resulta difícilmente abordable.

En este contexto, en el que la información fluye de manera constante desde todo tipo de dispositivos hacia servidores en los que va acumulándose en enormes bases de datos, resulta esencial contar con procedimientos que faciliten la integración, normalización y análisis de esos datos a fin de extraer a partir de ellos información útil. De esta forma las empresas podrán utilizarla para sugerir a los usuarios temas que pueden resultarle de interés, las entidades bancarias para determinar si una transacción es fraudulenta o legítima, las redes sociales para etiquetar automáticamente imágenes y textos, etc.

Las técnicas de minería de datos, como parte del proceso de descubrimiento de conocimiento útil en bases de datos, cobran cada vez más importancia, permitiendo sacar provecho de esos grandes volúmenes de datos que antes eran meros registros históricos y que actualmente se han convertido en un gran valor para todo tipo de entidades. Estas precisan de herramientas para explorar la estructura de los datos, efectuar distintos tipos de análisis sobre ellos a fin de determinar qué operaciones es necesario aplicar, decidir los algoritmos de aprendizaje a emplear según el objetivo que se persigue y, finalmente, interpretar los resultados

obtenidos a partir de ellos. Para dar forma a dichas herramientas es preciso resolver problemas abiertos en distintos campos, aplicando métodos existentes o definiendo otros nuevos según las necesidades.

En las siguientes secciones se detalla el tipo de problemas para los que se plantean soluciones en la presente tesis, así como la motivación y los objetivos específicos detrás de la misma y su estructura.

Planteamiento del problema

En el entorno que acaba de esbozarse es necesario considerar un conjunto de tareas, que serán descritas con detalle en el Capítulo 1, entre las que destacan el preprocesamiento y la clasificación. Los métodos de preprocesamiento tienen el objetivo de introducir cambios en los datos a fin de optimizar el funcionamiento de algoritmos de minería de datos como los de clasificación. Existe, por tanto, una estrecha relación entre ambas tareas.

Los conjuntos de datos, obtenidos a partir de las bases de datos a las que se hacía referencia antes, habitualmente presentan deficiencias o características propias que pueden afectar negativamente a los algoritmos que, mediante técnicas automáticas, tienen el objetivo de aprender modelos que permitan describir la estructura de los datos o bien predecir patrones futuros en función de los atributos conocidos de patrones pasados. Las deficiencias son, en general, fáciles de corregir, existiendo métodos bien conocidos para la normalización de valores, imputación de valores perdidos, eliminación de ruido y otras anomalías. No puede decirse lo mismo para las características específicas de ciertos conjuntos de datos, como es el caso de los de tipo multietiqueta.

Una base de datos multietiqueta se caracteriza por asociar a un cierto conjunto de características o atributos de entrada un conjunto de clases, categorías o etiquetas. Este problema, que será formalmente definido en el Capítulo 1, se presenta en infinidad de situaciones actualmente, sobre todo a la hora de clasificar textos (un mismo artículo puede asociarse a varios temas), imágenes (una

fotografía que contiene varios objetos), música (la misma pieza puede clasificarse en varios estilos) y, en general, todo tipo de contenidos multimedia presentes en las redes sociales. Esta tarea, la clasificación multietiqueta, es una disciplina relativamente reciente frente a la clasificación tradicional, en la que cada patrón está asociado a una sola categoría.

Determinar las características de un conjunto de datos multietiqueta requiere el uso de métricas específicas. Lo mismo ocurre con las técnicas de preprocesamiento de los datos, así como con los algoritmos de aprendizaje. Todos ellos han de tomar en consideración la naturaleza multietiqueta de los datos que, en muchas ocasiones, se resuelve simplificando el problema mediante transformaciones que permiten operar como si fuesen binarios o multiclase. En este escenario, por la relativa juventud del problema en cuestión, existen muchas líneas de trabajo abiertas.

Motivación

El móvil tras la presente tesis ha sido desde el principio dar respuesta a algunas de las cuestiones abiertas en el campo de la clasificación multietiqueta, un campo con mucho trabajo por hacer y con amplias posibilidades de desarrollo. Algunas de las tareas analizadas fueron las siguientes:

- Los conjuntos de datos multietiqueta cuentan con características específicas derivadas de la presencia de múltiples etiquetas asociadas a cada patrón. A pesar de ello, las métricas disponibles para evaluar dichas características son escasas por el momento. Nos planteamos aportar nuevas métricas que faciliten la caracterización de este tipo de conjuntos de datos.

- El hecho de que cada patrón de un conjunto de datos multietiqueta esté asociado a múltiples etiquetas, en ocasiones cientos o incluso miles, genera un nuevo problema que no existía en clasificación tradicional: la alta dimen-

sionalidad en el espacio de salida. Nos propusimos estudiar dicho problema y las formas en que podría abordarse.

- La distribución de las etiquetas presentes en los conjuntos de datos no es homogénea, más bien al contrario, existiendo etiquetas muy raras, que aparecen en pocos patrones, y otras muy comunes, con una alta frecuencia de aparición. Esta problemática, conocida como desbalanceo, se agrava en el caso multietiqueta por la gran cantidad de clases diferentes, frente a un problema binario o multiclase, así como por la existencia de interacciones entre las etiquetas. Esta situación ha sido la motivación del desarrollo de buena parte de la presente tesis.

- Para tareas de clasificación tradicional es posible encontrar multitud de herramientas y algoritmos que, incluso sin un gran conocimiento del problema, permiten operar de forma interactiva e incluso visual sobre los datos. La situación en el caso multietiqueta es muy distinta, con una ausencia casi total de este tipo de herramientas, dirigidas sobre todo al análisis exploratorio de los datos y su estudio. Disponer de una herramienta de este tipo sin duda facilitaría en gran medida el desarrollo de trabajos como los recogidos en la presente tesis.

- Una buena parte de los algoritmos de clasificación multietiqueta basan su funcionamiento en la simplificación del problema original, reduciéndolo a múltiples tareas de clasificación binaria o multiclase. También existen propuestas diseñadas para aprovechar algunas de las características específicas de estos conjuntos de datos, como es la información de correlación entre etiquetas. Es probable que el uso de información adicional, relativa a las características de los datos, durante el diseño del clasificador pueda mejorar su rendimiento, siendo este un tema que demandaría un proceso de análisis y experimentación exhaustiva.

A partir de este conjunto de planteamientos se definieron los objetivos concretos que se querían abordar en la tesis, enumerados en el siguiente apartado.

Objetivos

El objetivo principal de esta tesis es el desarrollo de métodos que, mediante hibridación de distintas técnicas de minería de datos y de algoritmos de preprocesamiento, contribuyan a mejorar el rendimiento predictivo de los modelos de clasificación. Para ello es preciso mejorar el conocimiento de las características de los mismos, así como de los métodos ya existentes. De esta forma, el mencionado objetivo principal se ha dividido en los siguientes objetivos secundarios:

- **Revisión del estado del arte**: El primer paso ha de ser necesariamente el estudio del estado del arte en relación al problema que se pretende abordar, analizando sus aplicaciones, la naturaleza de los conjuntos de datos, las métricas empleadas para describirlos, los enfoques seguidos para solventarlos y las métricas específicas para evaluar los resultados producidos por los algoritmos de cada uno de esos enfoques. El Capítulo 1 de la tesis recoge en su mayor parte esa revisión del estado del arte.

- **Métricas de caracterización**: Como podrá apreciarse a partir de la revisión a la que hace referencia el punto anterior, las métricas de caracterización para conjuntos de datos multietiqueta son escasas y, en su mayor parte, aportan únicamente algunos promedios de conteos de etiquetas en las muestras. El segundo objetivo que se estableció fue el de analizar la información aportada por este tipo de conjuntos de datos, tratando de definir donde fuese posible nuevas métricas que facilitasen la interpretación de dicha información. Esta ha sido una tarea constante a lo largo del desarrollo de la tesis, como podrá comprobarse por las nuevas métricas introducidas en los capítulos 2, 3 y 4.

- **Reducción de la dimensionalidad en el espacio de salida**: Por la manera en que suele abordarse la clasificación multietiqueta, utilizando múltiples clasificadores binarios o combinaciones de etiquetas (enfoques que se describirán en el Capítulo 1), tener un gran conjunto de etiquetas conlleva la construcción de modelos de clasificación más complejos y que requieren

más recursos. Por ello otro de los objetivos de la tesis fue proponer un método que permitiese reducir esa dimensionalidad, a fin de mejorar la eficiencia y eficacia de los clasificadores.

- **Análisis y reducción del desbalanceo**: Como se apuntaba en la sección previa, la presencia de desbalanceo y las características especiales de este problema en los conjuntos de datos multietiqueta ha movido una buena parte del trabajo de esta tesis. Por una parte se persigue analizar y caracterizar el desbalanceo presente en estos conjuntos de datos, un tema pendiente en la bibliografía especializada. Por otra, recurriendo a técnicas de preprocesamiento adaptadas y otras totalmente específicas, a la medida del problema en cuestión, se quiere contribuir a mejorar el rendimiento predictivo de los clasificadores multietiqueta reduciendo el nivel de desbalanceo.
- **Herramienta exploratoria de conjuntos de datos multietiqueta**: Por interés propio y también por la utilidad que podría tener para cualquiera interesado en trabajar con conjuntos de datos multietiqueta, nos propusimos desarrollar una herramienta que de una forma sencilla, desde una línea de comandos interactiva o bien a través de una interfaz gráfica de usuario, facilitase la exploración de este tipo de datos. Un software de este tipo podría, además, servir como base para la experimentación y el desarrollo de nuevos algoritmos, tanto de preprocesamiento como de clasificación.

Estos objetivos se han distribuido en la presente memoria de tesis acorde a la estructura descrita en la sección siguiente.

Estructura de la tesis

Además de esta introducción, la presente memoria de tesis consta de seis capítulos con la siguiente estructura:

1. **Fundamentos**: En el primer capítulo se sitúa el problema abordado en esta tesis en el contexto general del proceso de descubrimiento de conocimiento

a partir de bases de datos, se define la tarea de clasificación multietiqueta, se describen las características esenciales de este tipo de conjuntos de datos y se enumeran las aplicaciones que tienen, así como las propuestas de clasificadores multietiqueta existentes, las métricas de evaluación y otras tareas relacionadas. En la parte final del capítulo se detalla la problemática específica objeto de estudio de la tesis.

2. **Tratamiento de la dimensionalidad en el espacio de etiquetas**: El segundo capítulo se concentra en el análisis de los problemas que plantea la existencia de alta dimensionalidad en el espacio de salida, describiendo las vías que se han seguido hasta el momento para abordarlo. A continuación se propone un nuevo algoritmo, denominado LI-MLC, que mediante la hibridación de un algoritmo de minería de reglas de asociación y un clasificador multietiqueta consigue mejorar la eficiencia en clasificación de este último. La propuesta es validada experimentalmente usando un amplio grupo de conjuntos de datos multietiqueta.

3. **Desbalanceo en conjuntos de datos multietiqueta**: En el tercer capítulo se describen los aspectos específicos que exhibe el desequilibro entre clases en conjuntos de datos multietiqueta, proponiéndose un conjunto de métricas para caracterizar esta problemática. A continuación se presentan cuatro algoritmos de preprocesamiento que mediante técnicas aleatorias persiguen reequilibrar la distribución de las etiquetas, eliminando instancias (LP-RUS y ML-RUS) y agregando nuevas instancias (LP-ROS y ML-ROS). Todos ellos son analizados experimentalmente y comparados estadísticamente.

4. **Técnicas de remuestreo heurístico**: Si en el anterior las propuestas se basaban en selección aleatoria de muestras, en el cuarto capítulo se presentan tres algoritmos de tipo heurístico para el preprocesamiento de conjuntos de datos con el objetivo de reducir su nivel de desbalanceo, junto con una nueva métrica diseñada para evaluar el grado de concurrencia entre etiquetas muy frecuentes y poco frecuentes en los mismos patrones. Primero se

propone MLSMOTE, un algoritmo pensado para generar instancias sintéticas teniendo en cuenta las características específicas de estos conjuntos de datos. A continuación se presenta MLeNN, un algoritmo de eliminación de instancias informado, basado en datos de los vecinos más cercanos. La propuesta final, el algoritmo REMEDIAL, se basa en una nueva métrica y permite separar las etiquetas muy frecuentes de las poco frecuentes, mejorando el rendimiento predictivo sin provocar pérdida de información. Los tres algoritmos son evaluados empíricamente y comparados con otras propuestas existentes.

5. **Análisis exploratorio de conjuntos de datos multietiqueta**: En el quinto capítulo se presenta una herramienta, creada durante el desarrollo de la presente tesis, cuyo objetivo es facilitar la exploración y manipulación de conjuntos de datos multietiquetas. Se trata de un paquete para el conocido entorno/lenguaje R, facilitando la funcionalidad tanto a través de la línea de comandos como mediante una interfaz gráfica de usuario. El capítulo describe cada una de las funcionalidades del paquete y muestra cómo usar parte de ellas mediante ejemplos demostrativos. El citado paquete está disponible en el repositorio oficial de R, CRAN, y ha sido instalado por cientos de usuarios de dicho software.

6. **Conclusiones y futuras líneas de investigación**: El capítulo final de esta memoria de tesis facilita un resumen de los objetivos que se propusieron y las metas alcanzadas, así como una lista de las publicaciones efectuadas y una enumeración de las líneas de investigación que quedan abiertas para trabajo futuro.

Capítulo 1

Fundamentos

La clasificación de patrones es una de las tareas más importantes en el campo del aprendizaje automático supervisado. La clasificación multietiqueta es una generalización de la clasificación binaria y multiclase, al no imponer a priori un límite en el número de elementos que puede contener el conjunto de atributos de salida. Este tipo de clasificación se aplica en campos como la categorización de textos (Katakis et al. [2008]), el etiquetado de música (Wieczorkowska et al. [2006]), la catalogación de imágenes (Boutell et al. [2004]) o la predicción de funciones de proteínas (Diplaris et al. [2005]), entre otros.

El objetivo de este capítulo es introducir los fundamentos sobre los que se apoya la presente tesis, encontrándose dividido en cuatro secciones. Las dos primeras sitúan el área general de trabajo, el proceso de KDD en general y la clasificación multietiqueta en particular, presentando las características principales de estas bases de datos y haciendo una revisión de los métodos descritos en la literatura para abordar este problema. En la tercera sección se enumeran tareas relacionadas con la clasificación multietiqueta, mientras que en la cuarta se centra el foco en los aspectos concretos de dicho problema que se abordan en esta tesis.

1.1. Contexto

Comencemos situando adecuadamente el contexto del problema a abordar, empezando por el ámbito más global, que nos permitirá obtener una visión general, para ir paulatinamente estrechando el foco hasta situarnos en el punto concreto de interés.

1.1.1. El proceso de KDD

La disponibilidad de grandes volúmenes de datos en empresas e instituciones, recopilados a partir de la actividad diaria de millones de usuarios que queda registrada en todo tipo de sistemas informáticos, ha incrementado paulatinamente el valor del proceso conocido genéricamente como KDD (*Knowledge Discovery in Databases*) o de extracción de conocimiento desde bases de datos (Fayyad et al. [1996]). Las bases de datos, orientadas fundamentalmente al procesamiento en línea de transacciones (OLTP, *On-Line Transaction Processing*) por parte de aplicaciones de propósito general, son el origen del proceso de KDD. El objetivo final es convertir esos datos en conocimiento útil de alguna clase, ya sea de apoyo a la toma de decisiones, descripción de la información o predicción de datos futuros.

El proceso de extracción de conocimiento se compone de múltiples fases, representadas esquemáticamente en la Figura 1.1. El punto de comienzo, en el dominio en que se encuentra el problema que se intenta resolver, es la comprensión de dicho dominio a fin de poder especificar cuáles son los objetivos que quieren alcanzarse. Los datos necesarios para la consecución de dichos objetivos pueden residir en fuente heterogéneas, ser inconsistentes y contener información no relevante. Por ello en las dos fases siguientes se procede a integrar y unificar esos datos, limpiándolos de errores y seleccionando aquellos que se consideran relevantes. Usualmente también es preciso aplicar otras tareas de preprocesamiento y transformación, a fin de preparar los datos para la fase fundamental que es la de *minería de datos*. Es en esta fase del proceso en la que se utilizan los datos como entrada para algoritmos cuyo objetivo es extraer el conocimiento útil y no

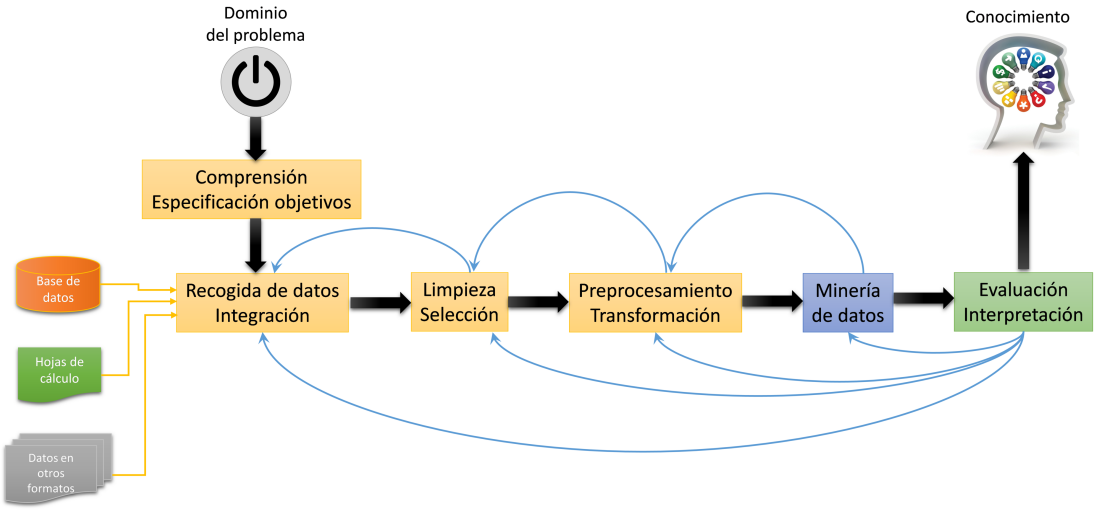


Figura 1.1: Fases del proceso de extracción de conocimiento.

trivial, ajustando parámetros y elaborando modelos que los describen o servirán para hacer predicciones futuras. El resultado de esta fase ha de ser evaluado e interpretado. Dependiendo de las conclusiones obtenidas es posible tener que volver uno o varios pasos atrás en el proceso, por ejemplo aplicando distintos métodos de limpieza, selección o transformación, a fin de alcanzar el conocimiento que sirva finalmente para alcanzar los objetivos establecidos al inicio.

Como se aprecia en la Figura 1.1, este proceso es iterativo y cada paso puede requerir la vuelta atrás para repetir la fase previa. Asimismo, tras la evaluación puede ser preciso volver a cualquiera de los pasos anteriores dependiendo de las conclusiones que se obtengan. De todo este proceso nos interesan especialmente las fases correspondientes al preprocesamiento y transformación de los datos y la minería de datos en sí misma.

1.1.2. Preprocesamiento y transformación

A pesar de que un conjunto de datos haya sido adecuadamente preparado, mediante la integración, limpieza y selección que permitan contar con informa-

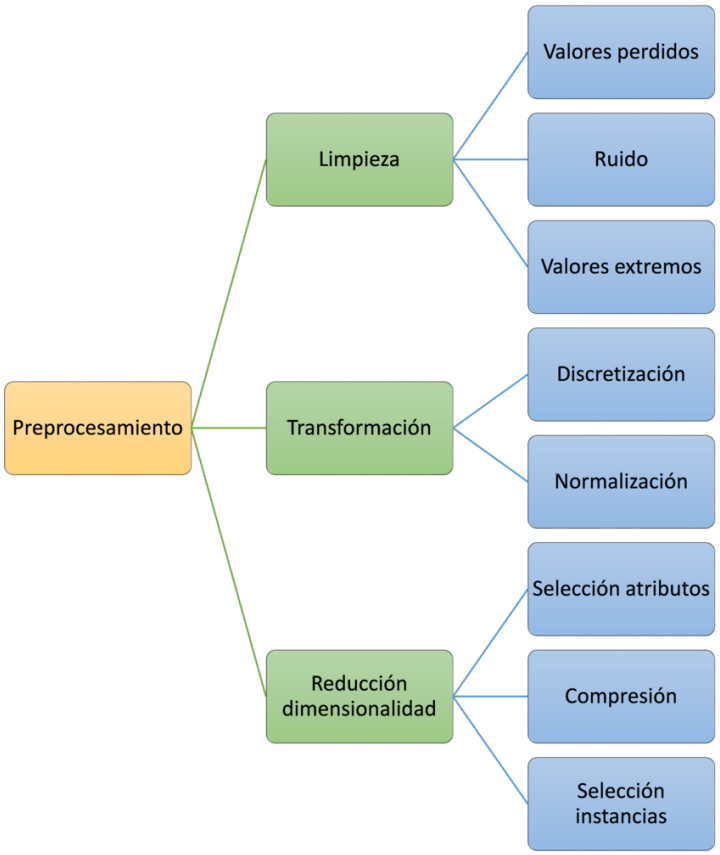


Figura 1.2: Taxonomía de tareas de preprocesamiento.

ción sin errores y relevante, es probable que en él aún permanezcan problemas que pueden influir de manera determinante en los algoritmos de minería de datos. Por ello es importante determinar y, en su caso, aplicar los métodos de preprocesamiento y transformación (Zhang et al. [2003], García et al. [2015]) adecuados a fin de mejorar el rendimiento obtenido en las fases posteriores. La Figura 1.2 representa esquemáticamente distintos tipos de tareas de preprocesamiento.

Con este tipo de técnicas es posible abordar tareas como la reducción de la dimensionalidad en el espacio de entrada mediante la selección de características, el tratamiento de variables correlacionadas mediante la creación de características, la discretización de valores en las variables, la reducción del número de instancias

en conjuntos de datos muy grandes a través de la selección de instancias, el reequilibrio en la distribución de clases mediante técnicas de remuestreo, etc. A continuación se describen someramente algunas de las tareas más comunes:

- **Limpieza de datos**: Bajo esta denominación genérica se engloban varias tareas, como son la gestión de valores perdidos, el tratamiento del ruido y la detección de valores extremos. **Los valores perdidos** (Grzymala-Busse and Grzymala-Busse [2010]) en un conjunto de datos pueden darse por distintas razones, por ejemplo de manera intencionada cuando una persona deja huecos en un formulario de entrada porque no quiere responder a ciertas cuestiones, hasta fallos de transcripción en las fases previas de KDD, por ejemplo en la fase de integración. El proceso de limpieza puede consistir en eliminar las instancias con valores perdidos o bien en realizar una imputación de los mismos, usando para ello el valor promedio del atributo afectado, la moda, o estimándolo a partir de algún modelo de predicción, por señalar algunos ejemplos. **El ruido** en los datos viene dado por valores en los atributos que son claramente incorrectos, presentando variaciones aleatorias no explicables. Al igual que los valores perdidos, es posible eliminar el ruido mediante distintas técnicas, incluyendo la eliminación de las instancias en que aparece, la sustitución por el valor más común entre sus vecinos o el cálculo mediante regresión. **Los valores extremos** (*outliers*) se caracterizan por presentar en sus atributos valores fuera de la norma, pero sin poder considerarse ruido sino posiblemente instancias de interés especial. Existen múltiples métodos estadísticos (Ben-Gal [2005]) para la detección y el tratamiento de valores extremos.

- **Transformación**: Cuando se opera con información cuantitativa es habitual aplicar ciertas transformaciones a los datos de entrada a fin de mejorar el comportamiento posterior de los algoritmos de minería de datos. Entre estas transformaciones las dos más usuales son la **normalización**, consistente en llevar los valores de distintos atributos a una escala común, y la **discretización**, cuya finalidad es convertir datos cuantitativos en cualitati-

vos (Yang et al. [2005]). Si bien las dos mencionadas son las más habituales, también caben en esta categoría otras técnicas como la agregación de datos o la construcción de nuevos atributos a partir de los existentes. La transformación de los datos puede ser también necesaria en casos en los que no es posible determinar algoritmos de minería de datos apropiados para su estructura original o, a pesar de sí hallarlos, es más eficiente o se obtiene mejor rendimiento tras aplicar a los datos una conversión determinada. Como se explicará más adelante, en la Sección 1.5.1, los métodos de transformación juegan un papel determinante en el área de la clasificación multietiqueta.

- **Reducción de la dimensionalidad**: Construir modelos a partir de conjuntos de datos con un gran número de variables y/o de instancias tiene un alto coste computacional, de ahí que las técnicas de reducción de dimensionalidad (Chizi and Maimon [2005]) estén entre los algoritmos de preprocesamiento de uso más habitual. La reducción de la dimensionalidad en el espacio de entrada (número de atributos) habitualmente permite tanto reducir el coste computacional de los modelos generados más adelante (más eficiencia) como incrementar su rendimiento (mejor poder predictivo). La **selección de características** (Vergara and Estévez [2014]) se apoya fundamentalmente en la eliminación de atributos redundantes, por ejemplo al ser combinaciones lineales de otros, e irrelevantes, aquellos que por su poca varianza no aportan información útil. La mayoría de algoritmos de selección operan como filtros o como envoltorios (*wrappers*) (Langley et al. [1994]). Los primeros recurren a heurísticas para determinar qué atributos pueden ser excluidos del conjunto de datos, mientras que los segundos inducen internamente un modelo de aprendizaje para realizar esa misma tarea. Los métodos de **selección de instancias y remuestreo** (Tang et al. [2014]) permiten reducir el número de muestras de datos, por ejemplo escogiendo aquellas más representativas aplicando técnicas de agrupamiento (*clustering*) entre otras, o bien reequilibrar la distribución de los datos a través de la eliminación o generación de instancias, según los casos.

En capítulos posteriores se profundizará en varios de los métodos de transformación mencionados, como las técnicas de remuestreo y las de reducción de dimensionalidad, al ser estos una de las bases fundamentales de las propuestas desarrolladas en la presente tesis.

1.1.3. Minería de datos

El fin último de las distintas fases de preparación de los datos es poder aplicar algoritmos de minería de datos o *data mining* (Fayyad et al. [1996]). Estos pueden clasificarse en distintas categorías, dependiendo de si los datos de entrada están o no etiquetados o según el objetivo que se persiga de esta fase. Atendiendo al primer criterio podemos distinguir entre:

- **Aprendizaje supervisado**: Los datos de entrada han sido manualmente etiquetados por expertos en el dominio del problema, por lo que los algoritmos pueden utilizar dicha información para inferir conocimiento que les permita reconocer otros patrones que no han sido etiquetados.
- **Aprendizaje no supervisado**: Los datos facilitados como entrada no están etiquetados, siendo preciso recurrir a métodos de aprendizaje alternativos basados exclusivamente en la información de cada patrón.
- **Aprendizaje semi-supervisado**: Es una técnica en la que se combinan muestras de datos etiquetadas y no etiquetadas (Zhu and Goldberg [2009]), usando las primeras para inducir un modelo y las segundas para refinarlo tras ser procesadas por dicho modelo o bien para mejorar la delimitación de las fronteras impuestas por el modelo.

Atendiendo al segundo de los criterios se diferencia entre:

- **Minería de datos descriptiva**: Su objetivo es describir la estructura de los datos, las relaciones existentes entre ellos u obtener algún otro tipo de información que aporte conocimiento útil no trivial. Este tipo de minería de

datos suele estar vinculado a métodos de aprendizaje no supervisados, como es el caso de los algoritmos de agrupamiento (*clustering*) o descubrimiento de reglas de asociación.

- **Minería de datos predictiva**: Su objetivo es generar, a partir de los datos de entrada, un modelo que permita etiquetar patrones similares en el futuro o predecir el valor de salida que se obtendría a partir de unos valores de entrada. Este tipo de minería de datos suele estar vinculada a métodos de aprendizaje supervisados, como es el caso de la clasificación o la regresión.

Tareas de minería de datos

El objetivo de un algoritmo de minería de datos se determinará en función de la tarea que se necesite abordar. Las más notables son las siguientes:

- **Clasificación**: El objetivo fundamental es predecir la clase de un patrón (Kotsiantis [2007]), de tipo categórico (discreto y sin orden) a partir de la información de patrones previamente etiquetados. Dependiendo de la técnica utilizada se generará un modelo, usando los patrones etiquetados que sirva como predictor para los no etiquetados, o bien se utilizará directamente la información de los primeros para clasificar los segundos, sin que medie un modelo.

- **Regresión**: En este caso el objetivo también es predecir un valor de salida a partir de un patrón de entrada, pero dicho valor será numérico (Draper et al. [1966]) y, habitualmente, continuo, en lugar de categórico. Se cuenta asimismo con una base de datos de patrones previamente anotada, con los valores que corresponden a cada patrón, lo cual permite generar un modelo a partir de ella.

- **Tratamiento de series temporales**: Las series temporales (Fu [2011]) se caracterizan porque los patrones de datos tienen una relación cronológica, siendo precisos algoritmos específicos para agrupar, clasificar, segmentar y

extraer reglas a partir de ellas. Además existen tareas propias para este tipo de bases de datos, como son la indexación (Faloutsos et al. [1994]), que hace posible la consulta a partir de muestras totales o parciales, o la predicción de series futuras (*forecasting*, De Gooijer and Hyndman [2006]).

- **Agrupamiento**: Esta tarea (Rokach [2010]) consiste en determinar el grado de similitud entre una base de datos de patrones no etiquetados, a fin de agruparlos según un cierto criterio. Además de para describir la estructura de los datos, esta técnica también se utiliza con otros fines como la detección de patrones raros (*outliers*) o la mejora de las fronteras obtenidas a partir de un modelo de clasificación.

- **Asociación**: El objetivo es descubrir asociaciones entre los datos con un enfoque no supervisado, generalmente con el objetivo de obtener un conjunto de reglas de asociación (Höppner [2010]) que ofrezcan una visión fácilmente interpretable de la estructura de los datos. También es una tarea que puede aplicarse a la reducción de dimensionalidad en el espacio de entrada, descubriendo asociaciones que permitan eliminar atributos innecesarios, por ejemplo mediante un PCA (*Principal Component Analysis*, Jolliffe [1986]).

Técnicas de minería de datos

Las tareas que acaban de enumerarse pueden ser abordadas recurriendo a un enorme conjunto de algoritmos distintos. Un buen número de ellos pueden agruparse en las siguientes categorías de técnicas de minería de datos:

- **Árboles**: Los algoritmos que inducen árboles (Quinlan [1986]) a partir de los patrones de entrada se han aplicado a tareas de clasificación, regresión y descubrimiento de reglas, entre otras. Algunos de los algoritmos más conocidos son C4.5 (Quinlan [1993]), probablemente el más usado en clasificación; CART (*Classification and Regression Tree*, Breiman et al. [1984]) y FP-Growth (Han et al. [2004]), usado para la minería de reglas de asociación.

- **Máquinas de vectores soporte**: Conocido genéricamente como SVM (*Support Vector Machine*) o SVN (*Support Vector Network*), este algoritmo se basa en la maximización de la distancia a una frontera de división (Cortes and Vapnik [1995]). Utilizándose inicialmente como una técnica de clasificación binaria, se han desarrollado algoritmos SVM tanto para clasificación multiclase como para regresión.

- **Vecinos cercanos**: Las técnicas kNN (*k Nearest Neighbors*) se caracterizan por ser perezosas, no construyéndose un modelo a partir de los patrones de entrada. Al procesar un nuevo patrón se utiliza una medida de similitud para determinar cuáles son los patrones conocidos más cercanos. Sus aplicaciones son múltiples, incluyendo algoritmos de clasificación (Cover and Hart [1967]), regresión (Altman [1992]), agrupamiento de datos (Everitt et al. [2011]), reducción de datos y selección de instancias (Hart [1968]), entre otras.

- **Minería de elementos frecuentes**: Determinar la frecuencia de elementos en una base de datos (Goethals [2005]) es una técnica básica para multitud de tareas, incluyendo el descubrimiento de reglas de asociación, obtención de correlaciones, agrupamiento de datos y clasificación. El primer algoritmo basado en esta técnica fue Apriori (Agrawal et al. [1994]), al que han seguido muchos otros.

- **Computación flexible**: Bajo esta denominación genérica (*Soft Computing*) se engloban técnicas basadas en la observación e imitación de procesos que pueden encontrarse en la naturaleza. Es una categoría amplia en la que podemos distinguir los siguientes grupos:

  - **Técnicas estadísticas**: Las técnicas estadísticas basadas en el análisis de probabilidades e inferencia bayesiana tienen múltiples aplicaciones en minería de datos. El teorema de Bayes se aplica directamente en la construcción de clasificadores probabilísticos (*Naive Bayes*). La regresión lineal y regresión logística también tienen su base en modelos estadísticos probabilísticos. Los modelos gráficos probabilísticos, como

las redes bayesianas (Cheng and Greiner [1999]), fueron considerados técnicas punteras en su momento.

- **Redes neuronales**: Las redes neuronales o ANN (*Artificial Neural Networks*) basan su funcionamiento en la estructura de conexiones y mecanismo de activación del cerebro humano, tratando de reproducir su modo de aprendizaje. Las ANN se aplican a problemas de clasificación, reducción de dimensionalidad y agrupamiento, entre otras tareas. Algunos de los algoritmos más empleados son el perceptrón multicapa o MLP (*Multi-Layer Perceptron*, McClelland et al. [1986]), las ANN con funciones de base radial o RBFN (*Radial Basis Function Network*, Broomhead and Lowe [1988]) y las redes auto organizativas o SOM (*Self-Organizing Map*, Kohonen [1982]).
- **Técnicas evolutivas y bioinspiradas**: Las técnicas basadas en la teoría de la evolución e inspiradas en el comportamiento de organismos biológicos son usadas habitualmente en problemas de optimización, existiendo algoritmos que usan estas técnicas para clasificación, regresión y agrupamiento, entre otras tareas. Los algoritmos evolutivos (Bäck et al. [2000], Eiben and Smith [2003]) se engloban fundamentalmente en dos grandes ramas conocidas como algoritmos genéticos y programación genética, y emulan el proceso de cruce, reproducción y mutación de los organismos vivos, representando una vía de exploración estocástica del espacio de soluciones. Los algoritmos evolutivos son una de las técnicas bioinspiradas que ha tenido mayor desarrollo. Otros métodos bioinspirados muy conocidos son los basados en colonias de hormigas o ACO (*Ant Colony Optimization*, Dorigo et al. [1999]) y los de nubes de partículas o Swarm (Kennedy [2010]).
- **Lógica difusa**: La lógica difusa (Zadeh [1965]) trata de modelar mediante algoritmos la forma de cognición humana, en la que para muchos conceptos no existe una línea de división nítida y absoluta, sino grados más o menos fuertes de pertenencia. Este tipo de técnica se ha aplicado a prácticamente todas las tareas de minería de datos, incluyendo

clasificación, agrupamiento y regresión entre ellos. Existen árboles de clasificación y regresión difusos (Fuzzy-CART, Jang [1994]), sistemas neuro-difusos para obtención de reglas (Mitra and Hayashi [2000]), algoritmos de agrupamiento difuso como C-Means (Bezdek [1981]), etc.

De las distintas tareas de minería de datos antes enumeradas aquí nos interesa específicamente la clasificación, a la que se dedica la siguiente sección.

1.1.4. Clasificación

La clasificación es una de las tareas abordadas en la fase de minería de datos, concretamente una tarea predictiva que habitualmente se desarrolla sobre métodos de aprendizaje supervisado (Kotsiantis [2007]). Su finalidad es aprender de patrones, previamente etiquetados, un modelo que permita predecir la clase a asignar a patrones futuros no etiquetados. También existen algoritmos de clasificación capaces de realizar esta tarea sin necesidad de construir previamente un modelo, como ocurre con los basados en la información de los vecinos más cercanos o kNN (*k Nearest Neighbors*) (Aha [1997]).

Los conjuntos de datos para clasificación tradicional cuentan con un conjunto de atributos de entrada, las variables predictoras, y un único atributo de salida, la variable a predecir. Dependiendo del número de valores que pueda tomar esta última variable, a la que se denomina habitualmente *clase*, se distingue entre **clasificación binaria**, en la que cada patrón únicamente puede pertenecer a una de dos clases posibles, y **clasificación multiclase**, en la que cada patrón únicamente puede pertenecer a una de un conjunto limitado de clases. Por tanto la clasificación binaria puede ser vista como una especialización de la clasificación multiclase, en la que el conjunto posible de clases cuenta solamente con dos elementos.

Mediante algoritmos de aprendizaje, habitualmente supervisados, se usan los atributos de entrada para inducir (Michalski [1983]) un modelo predictivo capaz de clasificar automáticamente patrones no etiquetados. Existen modelos espe-

cialmente idóneos para clasificación binaria, como es el caso de las máquinas de vectores soporte (SVM, *Support Vector Machines*, Burges [1998]), así como modelos con la capacidad de tratar dos o más clases, como son los modelos basados en árboles (Murthy [1998]) o en redes neuronales (Zhang [2000]). Asimismo, y como se apuntaba al inicio del apartado, se dispone de algoritmos de clasificación que no precisan la generación de un modelo previo (Aha [1997]), como es el caso de los métodos basados en instancias o kNN y todos sus derivados.

1.2. Clasificación multietiqueta

Avanzando un paso más en el estrechamiento del foco, que nos ha llevado desde el proceso completo de KDD hasta una tarea específica de minería de datos como es la clasificación, llegamos a un tipo específico de problema dentro de esa familia: la clasificación multietiqueta (Tsoumakas et al. [2010]). Esta se diferencia de la clasificación tradicional en que cada patrón de entrada tiene asociados múltiples atributos de salida, no solo uno, a los que se denomina genéricamente *etiquetas*. Cada una de estas etiquetas puede estar presente o no, por lo que en la práctica los valores posibles para cada una son solamente dos. Lo que se espera de los modelos de clasificación multietiqueta es que, dado un conjunto de atributos de entrada, predigan el conjunto de etiquetas (*labelset*) que le corresponde.

La Figura 1.3 es una representación conceptual de la diferencia existente entre la clasificación tradicional, en la parte superior, y la clasificación multietiqueta. En el primer caso cada patrón o muestra solamente pertenecerá a una clase, por lo que es posible asignarles un color concreto. En el caso multietiqueta cada patrón mostraría una combinación de colores, dependiendo de las etiquetas que tuviese asociadas.

En las siguientes secciones se describen las características fundamentales que tienen las bases de datos multietiqueta, los enfoques que se han seguido para diseñar clasificadores multietiqueta y las formas en que se evalúa el rendimiento de dichos clasificadores.

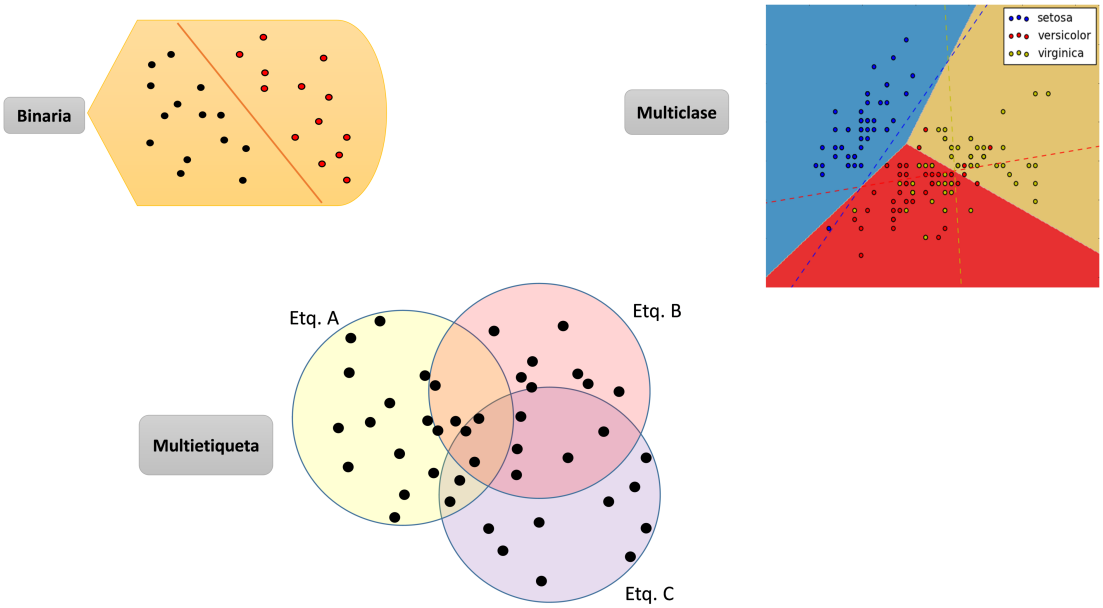


Figura 1.3: Diferencia conceptual entre los distintos tipos de clasificación.

1.3. Características de los conjuntos de datos multietiqueta

Todo proceso de extracción de conocimiento a partir de bases de datos verá su rendimiento influido fundamentalmente por dos aspectos: la calidad y características de los datos de entrada, factor en el que puede influir decisivamente el preprocesamiento aplicado a los mismos, y los propios algoritmos de minería de datos. Esta es la razón de que la caracterización de los conjuntos de datos sea importante. Las medidas de caracterización son útiles a fin de tomar la mejor decisión posible en cuanto a qué algoritmo ha de ser aplicado o diseñado dependiendo de la naturaleza exhibida por los datos.

En el contexto de los conjuntos de datos multietiqueta en la literatura se ha atendido especialmente a las características que describen cómo se distribuyen las

etiquetas, por una parte, y cómo lo hacen los conjuntos de etiquetas o *labelsets*, por otra.

Asumiendo que D es un conjunto de datos multietiqueta (MLD), denominaremos L al conjunto de todas las etiquetas que pueden aparecer en las instancias de D. $\mid D \mid$ denota el número de instancias en D y $\mid L \mid$ el número de etiquetas en L. Al subconjunto de L que aparece en la *i-ésima* instancia de D lo notaremos como Y_i. Este sería el *labelset* asociado a D_i. Un mismo *labelset* puede aparecer en múltiples instancias de D. El número de *labelsets* distintos que pueden generarse a partir de L es $2^{|L|}$. El número de *labelsets* diferentes que pueden aparecer en D estará limitado por la expresión $min(\mid D \mid,\ 2^{|L|})$ que, en la práctica, suele estar limitado superiormente por $\mid D \mid$.

1.3.1. Distribución de las etiquetas

La forma en que se distribuyen las etiquetas en L en las instancias de D es considerada una de las características fundamentales de cualquier MLD. Esta medida, conocida genéricamente como cardinalidad ($Card$) e introducida en Tsoumakas and Katakis [2007], se calcula según la Ecuación 1.1. La cardinalidad de un MLD permite conocer cuántas de las etiquetas de L están presentes en cada instancia de D en promedio. Un MLD con alta cardinalidad suele estar asociado a un conjunto L de gran tamaño, aunque lo inverso no es siempre cierto: hay MLDs que tienen asociado un gran conjunto de etiquetas a pesar de lo cual su cardinalidad es baja.

$$Card = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \mid Y_i \mid. \tag{1.1}$$

Esta medida se expresa en unidad concreta: etiquetas. Su interpretación no es fácil por sí sola, de ahí que se acompañe siempre de una segunda medida, sin dimensión, conocida como densidad ($Dens$) y presentada también en Tsoumakas and Katakis [2007], calculada según se indica en la Ecuación 1.2. Al usar el car-

dinal del conjunto L como divisor se obtiene una medida relativa de la dispersión de etiquetas. Una densidad alta denota que en el *labelset* de cada instancia hay una buena representación de los elementos de L. En contraposición, una densidad baja indicaría mayor dispersión, al aparecer en cada instancia un pequeño subconjunto de todas las etiquetas existentes en L.

$$Dens = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \frac{\mid Y_i \mid}{\mid L \mid}. \tag{1.2}$$

Suele asumirse que el aprendizaje a partir de MLDs con baja densidad resulta más difícil que cuando la densidad es alta, y que una densidad baja influye más en los resultados que una alta cardinalidad según Bernardini et al. [2014].

1.3.2. Distribución de los conjuntos de etiquetas

Las etiquetas de L se agrupan en cada instancia de D formando *labelsets* (Tsoumakas et al. [2010]). El número de *labelsets* diferentes está acotado, como se apuntaba anteriormente, por la expresión $min(\mid D \mid,\ 2^{|L|})$. Cuanto mayor sea el número de *labelsets* distintos en D, el número de combinaciones únicas de L, tanto más difícil será el aprendizaje a partir del MLD. A esta medida se la conoce como diversidad de etiquetas (Div), medida que también puede ser normalizada dividiéndola entre el número total de instancias en D.

Por otra parte, la frecuencia de aparición de los *labelsets* también es un factor a tomar en consideración. A pesar de que el número total de *labelsets* no sea muy alto, la existencia de muchas combinaciones que aparecen una sola vez, asociadas a una única instancia de D, también puede representar un obstáculo adicional en el proceso de aprendizaje.

El análisis de los *labelsets* puede aportar información valiosa sobre la forma en que se relacionan las etiquetas entre sí, obteniendo concurrencias y correlaciones independientes respecto al conjunto de atributos de entrada.

1.3.3. Otras características de los MLD

Otras medidas básicas de caracterización de los MLD son el número de instancias, el número absoluto de atributos de entrada y de etiquetas y las relaciones entre dichas medidas y otras como el número de combinaciones de etiquetas (métrica también conocida como *diversidad*). En la Tabla 1.1 se muestran estas características para algunos de los MLD conocidos. Algunos indicadores destacables a la vista de estos datos serían:

Tabla 1.1: Características básicas de algunos MLD

Nombre	**#Instancias**	**#Atributos**	**#Etiquetas**	**Card**	**#Labelsets**
bibtex	7395	1836	159	2.40	2856
bookmarks	87856	2150	208	2.03	18716
cal500	502	68	174	26.04	502
corel5k	5000	499	374	3.52	3175
corel16k	13811	500	161	2.87	4937
delicious	16105	500	983	19.02	15806
emotions	593	72	6	1.87	27
enron	1702	1001	53	3.38	753
EUR-Lex	19348	5000	3993	5.31	16467
genbase	662	1186	27	1.25	32
mediamill	43907	120	101	4.38	6555
medical	978	1449	45	1.25	94
rcv1*	6000	47236	101	2.88	1028
scene	2407	294	6	1.07	15
tmc2007	28596	49060	22	2.16	1341
yeast	2417	103	14	4.24	198

- El alto número de atributos de salida (etiquetas) con que cuentan varios MLD implicará para la mayoría de los algoritmos que se describirán después una gran complejidad, al tener que operar con cientos o miles de clasificadores binarios o bien un número de combinaciones de etiquetas enorme.

- Hay casos en los que el número de etiquetas supera al número de atributos predictivos, en ocasiones, como ocurre con `delicious` o `cal500`, incluso duplicando o triplicando las salidas a las entradas.

- En general el número de atributos de entrada es alto en los MLD, habiendo casos en los que se tienen más variables que muestras datos. Un caso extremo es `tmc2007`, con casi el doble de atributos que de muestras de datos.

- El número de combinaciones diferentes de etiquetas presentes en el MLD suele incrementarse con la cardinalidad y el número total de etiquetas. En determinados casos, como ocurre con `cal500`, `delicious` o `EUR-Lex`, hay casi tantas combinaciones de etiquetas como muestras de datos.

Si bien las medidas enumeradas aquí y en los dos apartados previos son las referenciadas asiduamente en la literatura, y por tanto son las más estudiadas, a partir del análisis de las frecuencias de aparición de las etiquetas y de los *labelsets* es posible deducir medidas de caracterización adicionales relacionadas con la dispersión, el desbalanceo en la distribución de etiquetas, la concurrencia entre etiquetas frecuentes y no frecuentes, la asimetría y curtosis en la distribución de etiquetas, etc. Varias de estas características son parte del objetivo de estudio de la presente tesis y serán descritas con mayor detalle en el Capítulo 3 (métricas para evaluar el desbalanceo en conjuntos de datos multietiqueta) y el Capítulo 4 (métrica para analizar la concurrencia entre etiquetas desbalanceadas).

1.4. Aplicaciones multietiqueta y sus conjuntos de datos

La mayor parte de las publicaciones de nuevas propuestas para clasificación multietiqueta apoyan su experimentación en una colección de conjuntos de datos bien conocidos, usando generalmente un pequeño subconjunto de los mismos. Esta colección es una representación de los tipos de aplicación para los que se

utilizan los clasificadores multietiqueta: etiquetado de textos, categorización de audio, imágenes estáticas y vídeo, medicina y biología, etc. A continuación se enumeran varios de ellos agrupados según tipo de aplicación.

1.4.1. Categorización de textos

La necesidad de incluir textos en dos o más categorías está en las raíces de la clasificación multietiqueta, por lo que no es extraño que un buen número de los MLD existentes estén asociados a este tipo de aplicación. Por ejemplo:

- **bibtex**: Introducido en Katakis et al. [2008], este MLD contiene meta-datos de entradas bibliográficas. Se usan como atributos el título de los artículos, nombres de los autores y de la revista, fecha de publicación, etc. En total se obtienen 1836 características. A cada documento se le asignan etiquetas con el objetivo de clasificarlos automáticamente en categorías relevantes, existiendo un total de 159 etiquetas distintas.
- **enron**: Introducido en Klimt and Yang [2004], se trata de un subconjunto de correos electrónicos etiquetados. Cada documento está descrito por 753 características y hay un total de 53 categorías a las que puede ser asociado: reuniones, asuntos legales, informes de viaje, imagen de empresa, etc.
- **EUR-Lex**: Introducido en Mencia and Fürnkranz [2008], está compuesto por textos legales de la Unión Europea en representación TF/IDF. De todos los términos se tomaron los 5 000 más frecuentes como atributos de entrada, con 3 993 etiquetas con relaciones jerárquicas.
- **medical**: Introducido en Crammer et al. [2007]. Se cuenta con 1449 características extraídas de un texto en el que se describen los síntomas de distintos pacientes, siendo el objetivo asociarle diagnósticos posibles a partir de un conjunto de 45 posibilidades no excluyentes.
- **rcv1v2**: Introducido en Lewis et al. [2004], se trata de un corpus de 800 000 textos de noticias etiquetados manualmente en conceptos organizados en

una jerarquía, representados por 126 códigos de tema de los cuales los editores usaron 103. Habitualmente se utilizan subconjuntos de este MLD con solo una parte de las instancias.

- **tmc2007**: Introducido en Srivastava and Zane-Ulman [2005]. En este caso los documentos de entrada son informes de fallos ocurridos durante vuelos, a partir de los cuales se han extraído 49 060 características. El objetivo es asociar a cada uno las etiquetas, de un conjunto de 22, que identifican dichos fallos.

1.4.2. Etiquetado de recursos multimedia

El etiquetado automático de imágenes en redes sociales, de secuencias de audio para identificar estilos musicales o especies de pájaros, y de vídeos son los problemas que dieron lugar a los siguientes MLD:

- **birds**: Introducido en Briggs et al. [2012]. Se trata en realidad de un problema multi-instancia multietiqueta, en el que se trata de predecir simultáneamente la especie de varios pájaros, entre 19 posibilidades no excluyentes, a partir de 260 características extraídas de cortes de 10 segundos de sonido.

- **cal500**: Introducido en Turnbull et al. [2008], el objetivo es facilitar la anotación y recuperación de piezas musicales y efectos sonoros. Cada sonido está descrito por 68 características, existiendo 174 etiquetas posibles. Es un conjunto de datos en el que el número de salidas prácticamente triplica al de atributos de entrada, con la peculiaridad adicional de que cada una de sus 502 muestras tiene asociado un *labelset* único.

- **corel**: Existen distintos conjuntos extraídos de esta base de datos de imágenes, como las usadas en Duygulu et al. [2002] (`corel5k`) y en Barnard et al. [2003] (`corel16k`). En ambos casos se cuenta con el mismo conjunto de características de entrada, obtenidas a partir de los atributos de las

imágenes, pero el número de etiquetas a asignar a cada imagen es de 374 en el primer caso y de una media[1] de 161 en el segundo.

- **emotions**: Introducido en Wieczorkowska et al. [2006], el objetivo es etiquetar piezas musicales con las emociones que generan. Cada pieza está descrita por 72 características, existiendo 6 emociones no excluyentes entre sí. Junto con el `scene`, este posiblemente sea el conjunto de datos multietiqueta más utilizado. Ambos cuentan con solo 6 etiquetas y un número reducido de instancias.

- **mediamill**: Introducido en Snoek et al. [2006] como parte de una competición, el objetivo es detectar 101 conceptos semánticos, las etiquetas, en una colección de secuencias de vídeo de las que se han extraído 120 características.

- **scene**: Introducido en Boutell et al. [2004], el problema corresponde a la categoría de categorización de imágenes. Concretamente trata de etiquetar distintas escenas asignándoles los contextos relevantes. De cada imagen se han extraído 294 características y hay 6 etiquetas posibles.

1.4.3. Genética/Biología

- **genbase**: Introducido en Diplaris et al. [2005], es una base de datos en la que se aporta más de un millar características diferentes por cada gen posible, siendo el objetivo asignarles las funciones biológicas que le corresponden. Estas son en total 27 y cada gen puede expresar varias de ellas.

- **yeast**: Introducido en Elisseeff and Weston [2001], es similar a genbase. El problema es asociar genes con una o más de las funciones biológicas que les corresponden. Existen 14 funciones distintas, que son las etiquetas, a predecir a partir de 198 características extraídas de cada gen.

[1] El conjunto corel16k está formado por 16 subconjuntos con ligeras diferencias en el número de instancias y de etiquetas.

1.4.4. Otras aplicaciones

Además de las enumeradas en los anteriores apartados, en las que se ha hecho referencia a conjuntos de datos multietiqueta que están a disposición pública, la clasificación multietiqueta se ha utilizado para muchas otras aplicaciones empleando conjuntos de datos privados o generados *ad hoc* para una necesidad concreta. En Sobol-Shikler and Robinson [2010] se analizan expresiones no verbales del habla para predecir estados de ánimo en personas que pueden darse de forma simultánea. Los autores de Cong and Tong [2008] proponen un método para clasificar automáticamente expedientes de patentes en las categorías que les corresponden. El método propuesto en Chidlovskii [2010] aborda un problema de clasificación multietiqueta supervisado, partiendo de apenas un 20 % de páginas etiquetadas de Wikipedia y con el objetivo de etiquetar automáticamente otros contenidos. La finalidad de la propuesta de Sriram et al. [2010] es mejorar la búsqueda en Twitter, etiquetando la información entrante en cinco categorías no excluyentes que permiten al usuario obtener únicamente aquello en que está interesado: noticias, opiniones, eventos, etc. En Chen et al. [2015] se utiliza un modelo de hipergrafos multietiqueta que, a partir de las características de múltiples objetivos en movimiento en secuencias de vídeo, predice conceptos semánticos de alto nivel a fin de detectar eventos complejos, en los que participan varios actores.

Si bien la finalidad concreta de cada propuesta difiere, prácticamente todas ellas pueden agruparse en las categorías antes mencionadas: categorización de texto, etiquetado de recursos multimedia y genética/biología.

1.5. Clasificadores multietiqueta

El proceso de aprendizaje para la obtención de un modelo de clasificación multietiqueta es análogo al de los clasificadores tradicionales, siguiendo un esquema supervisado la mayoría de las veces. La diferencia esencial estriba en la necesidad de que ese modelo genere múltiples salidas, tantas como etiquetas existan en L, en lugar de una única clase.

En definitiva, se trata de determinar una función $C : X_i \rightarrow Y_i$, $X_i \in X^1 \times X^2 \times \cdots \times X^f, Y_i \subseteq L$, siendo C el clasificador, X_i los atributos de entrada de la instancia *i-ésima*, Y_i el conjunto de etiquetas asociado a dicha instancia, L el conjunto total de etiquetas, f el número de atributos de entrada y X^j el espacio de los posibles valores para el atributo j-ésimo, $j \in \{1..f\}$

Hay dos enfoques fundamentales a la hora de desarrollar algoritmos de clasificación multietiqueta. El primero de ellos se basa en técnicas de transformación que hacen posible solventar el problema apoyándose en algoritmos de clasificación tradicionales. En el segundo se aboga por la adaptación de dichos algoritmos a fin de que sean capaces de operar con múltiples salidas. A partir del primer enfoque surge un tercer grupo: el de los *ensembles* o multi-clasificadores.

La clasificación multietiqueta puede enfocarse como un caso particular de un problema diferente: *label ranking*. En un problema de clasificación multiclase se crearía el ranking y se tomaría únicamente la primera etiqueta, la que está en lo alto del ranking (Tsoumakas et al. [2010]). En un problema multietiqueta se tomarían aquellas que superan un cierto umbral. Los algoritmos de ranking multietiqueta se limitan a crear una lista ordenada por una cierta medida de las posibles etiquetas que puede tener una muestra, pero sin facilitar un punto que indique a partir de cuál de ellas debería cortarse, es decir, qué etiquetas deberían tomarse y cuáles no. Dado que están ordenadas según la citada medida, basta con conocer el punto de corte o umbral. Ciertos algoritmos de clasificación multietiqueta actúan sobre un ranking previamente generado ajustando el umbral de corte.

En las siguientes subsecciones se describen múltiples propuestas agrupadas en distintas categorías.

1.5.1. Métodos basados en técnicas de transformación

También conocidos como métodos de *transformación del problema*, los métodos de clasificación multietiqueta basados en transformaciones tienen el objetivo

de convertir el conjunto de datos original, de naturaleza multietiqueta, en uno o más conjuntos de datos con una sola etiqueta que puedan ser tratados con modelos de clasificación tradicionales. Para ello se ha de llevar a cabo un trabajo de preprocesamiento de los conjuntos de datos, obteniendo como resultado un nuevo conjunto o bien un grupo de ellos, según los casos, siendo cada uno de ellos binario respecto a la clase (solamente se ha de predecir si cada muestra pertenece o no a la clase) o bien multiclase (existen varias clases posibles, de las cuales cada muestra pertenece a una). En Barot and Panchal [2014] puede encontrarse una revisión reciente de métodos de transformación para clasificación multietiqueta.

Algunas propuestas descritas en Tsoumakas et al. [2010] son:

- Generar uno o múltiples conjuntos de tipo multiclase a partir de los datos multietiqueta, usando para ello alguna de las técnicas denominadas genéricamente *Ranking Via Single Label*. Las tres alternativas básicas son:
  - Descartar todas aquellas instancias de datos que tengan asociada más de una etiqueta, usando el resto para entrenar el clasificador. El resultado de esta transformación sería la obtención de un nuevo conjunto de datos de tipo multiclase, con una sola etiqueta por muestra, y con igual o menor número de instancias que el original.
  - Desdoblar las instancias que tienen asociadas múltiples etiquetas en varias instancias con una sola, copiando los atributos de entrada en cada copia. Opcionalmente se le puede asignar a cada nueva instancia un peso obtenido a partir de la frecuencia de la etiqueta a la que representa, equilibrando la importancia de los patrones más raros frente a los comunes. El resultado es un conjunto multiclase con una sola etiqueta por muestra y un número de instancias superior al original.
  - Seleccionar en las muestras con varias etiquetas solamente una de ellas y descartar el resto. La selección puede ser aleatoria o bien estar basada en una heurística. El resultado es un conjunto de datos multiclase con el mismo número de muestras que el original.

- Considerar el subconjunto de etiquetas asociado a cada muestra como una etiqueta única (Boutell et al. [2004]), a la que se asignaría una nueva denominación que podría obtenerse de la combinación de las originales. Produce, como las dos anteriores, un único conjunto de tipo multiclase.

- Separar las etiquetas existentes en el dataset mediante técnicas de tipo OVA (*One-Versus-All*) u OVO (*One-Versus-One*) a fin de poder usar clasificadores binarios (Godbole and Sarawagi [2004]) o generar un ranking a partir de clasificadores centrados en el aprendizaje de pares de etiquetas (Crammer and Singer [2003], Hüllermeier et al. [2008] y Fürnkranz et al. [2008]). Este tipo de transformación producirá, a partir del conjunto de datos origen, tantos conjuntos de datos distintos como etiquetas haya en uso en las muestras de datos.

Las propuestas existentes son múltiples[2] tal y como se detalla en Tsoumakas et al. [2010], pero entre ellas destacan dos técnicas de transformación predominantes, BR y LP, que sirven a su vez como base para muchas otras. Algunas de ellas son:

- **BR**: Fue introducido en Godbole and Sarawagi [2004] con el nombre *ensemble of binary classifiers*, siendo actualmente más conocido como BR (*Binary Relevance*). Esta técnica transforma el MLD de origen en múltiples conjuntos de datos binarios, tantos como etiquetas existan en L. En el dataset binario correspondiente a cada etiqueta las instancias en que está presente son positivas, apareciendo como negativas el resto. Se trata por tanto de un enfoque uno-contra-todos u OVA. Al clasificar nuevas instancias, estas han de pasar por todos los clasificadores binarios y, finalmente, mediante la unión de las predicciones individuales, se obtiene el *labelset* a predecir. Se trata de un método sencillo que permite utilizar como clasificador subyacente cualquier algoritmo de clasificación binaria. El mayor inconveniente

[2]Diferentes métodos de transformación que generan conjuntos de clasificadores (multiclasificadores o *ensembles*) no se han incluido aquí, siendo descritos en el apartado 1.5.3.

que suele achacarse a este enfoque es que descarta por completo las relaciones entre etiquetas, una información que podría ser potencialmente útil para mejorar los resultados de clasificación. También implica un incremento lineal en el tiempo de ejecución en relación al número de etiquetas total que exista en L. A pesar de su aparente sencillez, la clasificación multietiqueta basada en BR ha probado su eficacia en multitud de contextos como se demuestra en Luaces et al. [2012].

- **2BR**: Para paliar el problema fundamental de BR, no tomar en consideración las dependencias entre etiquetas, en Tsoumakas et al. [2009] se introduce el algoritmo 2BR, basado en una técnica de *stacking* que consiste en usar BR en dos niveles, aprendiendo un modelo base y un meta-modelo que toma como entrada las salidas del modelo base, incluyendo un proceso de poda en el que explícitamente se toma en consideración el coeficiente de correlación entre etiquetas.

- **BR+**: Introducido en Alvares-Cherman et al. [2012], el objetivo es el mismo que en 2BR: mejorar el rendimiento de BR incluyendo información de otras etiquetas. En este caso el conjunto de atributos de entrada usado para entrenar el clasificador de cada etiqueta se amplía, agregando la información del resto de las etiquetas. Al procesar instancias de test, sin embargo, se afronta un problema nuevo: el espacio de entrada de la muestra es diferente al de cada uno de los modelos binarios generados anteriormente. Lo que se hace es procesar la muestra con un clasificador BR estándar, usando las salidas para extender el espacio de entrada de la muestra y entregarla a los clasificadores que operan con el espacio de entrada aumentado.

- **LP**: Fue introducido en Boutell et al. [2004] con el nombre *MODEL-n*, haciendo la n final énfasis en que cada combinación distinta de etiquetas se convierte en una nueva clase. Actualmente es conocido como LP (*Label Powerset*). Este método utiliza el *labelset* de cada instancia en D como identificador de clase, transformando el problema de clasificación multietiqueta en uno de clasificación multiclase. El número de clases obtenidas será

el número de distintas combinaciones de etiquetas que aparezcan en D, por lo que su número es potencialmente muy alto como se indicó anteriormente. A diferencia de BR, LP sí toma en consideración la relación que pudiera existir entre las etiquetas al generar una nueva clase por cada combinación presente en D. El principal problema es que el número de clases distintas puede llegar a ser intratable, al igualar el número de instancias del conjunto de datos. También ha de tenerse en cuenta que un clasificador LP es necesariamente incompleto, ya que únicamente puede predecir *labelsets* que estén presentes en el conjunto de entrenamiento.

- **PS**: Para intentar paliar el principal problema de LP, la explosión en el número de combinaciones de etiquetas, en Read et al. [2008] se presenta PS (*Pruned Sets*), introduciendo el concepto de *pruned sets* (conjuntos de etiquetas con poda), aplicándose una poda para eliminar aquellos *labelsets* que no llegan a un cierto umbral de frecuencia y cuyos subconjuntos pueden ser introducidos con posterioridad siguiendo distintas estrategias.

Propuestas como el algoritmo ChiDep, introducido en Tenenboim-Chekina et al. [2010], tratan de combinar lo mejor de BR y LP separando para ello las etiquetas mediante un test de independencia χ^2, entre grupos de etiquetas independientes y grupos de etiquetas dependientes, procesándose los primeros con BR y los segundos con LP.

Cada método de transformación ha de contar también con una técnica que le permita construir los *labelset* resultantes a partir de las predicciones devueltas por los clasificadores tradicionales, generando así la salida final del clasificador multietiqueta. En el caso de BR basta con unir las predicciones individuales de cada clasificador binario, mientras que con LP el identificador de clase obtenido a partir del clasificador multiclase es el *labelset* a predecir.

Un importante efecto secundario de muchos de estos métodos de transformación estriba en el problema de desbalanceo que se genera. Para cada clasificador obtenido mediante el enfoque BR se consideran como positivos únicamente las muestras en que aparece la etiqueta relevante, siendo negativas todas las demás

(enfoque OVA). Por regla general el número de muestras negativas será mucho mayor que el de positivas, existiendo un importante desequilibrio.

Los métodos de clasificación multietiqueta basados en transformación BR precisan un clasificador binario subyacente que facilite las predicciones, tarea para la que suele recurrirse sobre todo a algoritmos basados árboles y a máquinas de vectores soporte. No obstante, y dado que una de las primeras aplicaciones de este tipo de clasificación fue la categorización automática de textos, también es habitual la incorporación de algoritmos como AdaBoost, dando origen al algoritmo BoosTexter (Schapire and Singer [2000].)

1.5.2. Métodos basados en adaptación de algoritmos

Durante años la tarea de clasificación automática tradicional ha sido afrontada mediante múltiples familias de algoritmos que, a lo largo del tiempo, se han probado y refinado paulatinamente. Estos algoritmos representan un importante pilar sobre el que desarrollar otros más específicos, incluyendo aquellos dirigidos a procesar MLDs.

Ciertos algoritmos, como es el caso de las SVM, por su naturaleza son difícilmente adaptables al contexto multietiqueta. Por el contrario otros, como los árboles, las redes neuronales y los métodos basados en instancias, pueden ser extendidos de forma relativamente sencilla. Esta es la razón de que muchas de las propuestas de adaptación de algoritmos para clasificación multietiqueta estén basados en estas tres familias de métodos.

Métodos basados en árboles

Los árboles de decisión (DT, *Decision Trees*) son uno de los mecanismos más sencillos de clasificación, a pesar de lo cual ciertos algoritmos de este tipo, como es el caso del conocido C4.5 (Quinlan [1993]), obtienen resultados que les permiten competir con otros clasificadores. Algunas de las propuestas de clasificación multietiqueta basadas en árboles son las siguientes:

- **ML-C4.5**: Con el objetivo de clasificar genes de acuerdo a su función, un mismo gen puede influir en varias funciones y, por tanto, estar asociado a más de una clase (etiqueta), en Clare and King [2001] se propone un algoritmo basado en C4.5 modificado para operar con varias etiquetas. La adaptación tiene dos puntos clave: las hojas del árbol ahora pueden almacenar un conjunto de clases, en lugar de una sola, y la medida de entropía original se ha modificado para tomar en consideración la probabilidad (frecuencia relativa) de que la muestra procesada no pertenezca a una cierta clase. De esta forma el particionado de cada conjunto de muestras (a medida que va generándose el árbol) de acuerdo con un cierto atributo se calcula como suma ponderada de la entropía para cada subconjunto posible. En dicha ponderación si un cierto elemento aparece dos veces en un subconjunto, porque pertenezca a más de una clase, entonces se suma esas dos veces. El hecho de cada hoja se corresponda con un conjunto de clases influye tanto en el proceso de etiquetado de los nodos del árbol como en el posterior proceso de poda propio del algoritmo C4.5.

- **ADTBoost.MH**: Los ADT (*Alternate Decision Tree*) son una generalización de los árboles de decisión tradicionales introducida en Freund and Mason [1999] y que propone una alternativa al uso de técnicas de *boosting* para mejorar la precisión de los clasificadores basados en árboles. Basada en ADTs, y extendiendo el modelo AdaBoost.MH (Schapire and Singer [1999]), en De Comité et al. [2003] se propone ADTBoost.MH, un tipo de ADT adaptado para incorporar internamente una descomposición de los ejemplos multietiqueta mediante técnicas OVA. Por su estructura, ADTBoost.MH podría etiquetarse también como un método multiclasificador.

- **ML-TREE**: Una de las propuestas más recientes basada en árboles es el algoritmo ML-TREE (Wu et al. [2014]). El algoritmo induce un árbol con una estructura jerárquica, usando en cada nodo el enfoque OVA con clasificadores de tipo SVM para ir dividiendo el conjunto de datos a medida que se generan nuevos hijos, en un proceso recursivo. La frecuencia con que dos etiquetas aparecen conjuntamente en las hojas que van generándose se usa

como un indicador de la relevancia de relación entre ellas, definiendo así un modelo automático de descubrimiento de las dependencias entre etiquetas.

- **LaCova**: Introducido en Al-Otaibi et al. [2014], este algoritmo genera un árbol alternando el uso de las transformaciones BR y LP en función de las dependencias entre etiquetas representadas en cada nodo, para ello se utiliza una matriz de covarianzas de las etiquetas. De esta forma el árbol va dividiéndose recursivamente de forma horizontal, tomando como criterio una de las características de entrada, o bien verticalmente, separando etiquetas.

Métodos basados en redes neuronales

Las redes neuronales artificiales o ANN (*Artificial Neuronal Networks*) en general, y las RBFNs (*Radial Basis Function Networks*) en particular, han demostrado su efectividad en problemas de clasificación, regresión y predicción de series temporales, por lo que su adaptación para abordar de manera satisfactoria la clasificación multietiqueta es uno de los temas que más publicaciones ha ocupado en los últimos años. Algunas de las propuestas al respecto son las siguientes:

- **BP-MLL**: Partiendo de uno de los modelos de ANN más básicos, como es el perceptrón, y el algoritmo de aprendizaje más popular para este tipo de modelos: *back-propagation*, en Zhang [2006] se propone una adaptación para clasificación multietiqueta que está considerada la primera aplicación de las ANN a este campo. El aspecto clave de BP-MLL estriba en la introducción de una función propia para el cálculo del error cometido adaptada al hecho de que cada muestra tiene asociadas varias etiquetas, concretamente penalizando las predicciones en que se colocan en el ranking etiquetas que no corresponden con la muestra procesada. La capa de entrada tiene tantas neuronas como atributos de entrada y la de salida tantas como etiquetas existan. La capa oculta se compone de neuronas con función sigmoidal y su número se establece también en función del número de etiquetas.

- **I-BP-MLL**: A fin de determinar el conjunto de etiquetas predichas para la muestra procesada, el algoritmo BP-MLL usa un parámetro que actúa como umbral de corte para el ranking que se obtiene a la salida de la ANN. La dificultad que plantea el ajuste de dicho parámetro es resuelta en Grodzicki et al. [2008], incorporándolo a la función de cálculo del error de forma que el umbral va adaptándose automáticamente durante el proceso de aprendizaje. En realidad se genera un umbral personalizado para cada una de las etiquetas, en lugar de recurrir a uno general como en BP-MLL. Este hecho contribuye a que la versión modificada mejore al algoritmo original.

- **ML-RBF**: En Zhang [2009] se propone este algoritmo para el diseño de RBFNs especializado en el tratamiento de conjuntos de datos multietiqueta. Tomando las muestras asociadas a cada etiqueta, ejecuta el algoritmo K-medias tantas veces como etiquetas distintas haya para realizar un clustering que servirá para obtener los centros de las RBFs. Un parámetro α determina el número de clusters por etiqueta. El número de neuronas en la capa interna es igual o superior al de etiquetas en el conjunto de datos, dependiendo de dicho parámetro. El entrenamiento se completa ajustando los pesos hacia las neuronas de salida (una por etiqueta) mediante el método SVD (*Singular Value Decomposition*) minimizando una función de suma de los cuadrados del error cometido. La activación de todas las neuronas está fijada a 1 y existe un sesgo (*bias*) por cada etiqueta que se aplica a todas las neuronas asociadas. El habitual parámetro σ se calcula en una ecuación en la que intervienen la distancia media entre cada par de vectores prototipo y un parámetro de escalado μ. El ajuste de este parámetro y de α afecta de manera fundamental al algoritmo.

- **CCA-ELM**: Introducido en Kongsorot and Horata [2014], este algoritmo propone una metodología para adaptar una ELM (*Extreme Learning Machine*), un tipo de red neuronal con una sola capa oculta que se caracteriza por su rapidez en aprendizaje, a fin de que pueda procesar datos multietiqueta. Para ello se recurre a un CCA (*Canonical Correlation Analysis*) para determinar correlaciones entre los atributos de entrada y las etiquetas,

generando un nuevo espacio en el que se combinan las entradas y las etiquetas. En el paso final la predicción facilitada por la ELM ha de ser devuelta al espacio original, mediante la transformación inversa.

- **FPSO-MLRBF y FSVD-MLRBF**: Estas dos propuestas, variantes del algoritmo ML-RBF, son presentadas en Agrawal et al. [2014] como hibridaciones de distintas técnicas que persiguen mejorar el rendimiento base de ML-RBF. En el primer caso se utiliza un PSO (*Particle Swarm Optimization*) difuso y en el segundo un *k-means* difuso junto con SVD (*Single Value Decomposition*). En ambos casos a fin de determinar los nodos que existirán en la capa oculta de la ANN y optimizar los pesos entre esta y la capa de entrada.

Métodos basados en SVM

Las máquinas de vectores soporte o SVM han sido aplicadas tradicionalmente a problemas de clasificación binaria pero, como ha ocurrido con otras técnicas, a lo largo del tiempo se han incorporado a las mismas extensiones y mejoras que han permitido su uso en otras áreas, como es la de la clasificación multietiqueta. A continuación se describen varias de las aplicaciones de SVM en este campo:

- **MODEL-x**: El objetivo de los autores de Boutell et al. [2004] era el etiquetado de escenas naturales en las que podían aparecer múltiples objetos y, en consecuencia, ser asociadas a más de una clase: urbana, playa, campo, montaña, playa+campo, campo+montaña, playa+urbana, etc. Para ello optaron por una SVM dado su buen comportamiento en la clasificación de imágenes. Para adecuar la SVM al problema multietiqueta se define un modelo al que los autores denominan MODEL-x, consistente en usar varias veces los datos para entrenar un clasificador por etiqueta. El aspecto más interesante de la propuesta es precisamente el método de entrenamiento empleado, denominado *cross-training*, que radica básicamente en que las muestras que pertenecen a varias clases se tratan como positivas al entre-

nar cada modelo individual, no como negativas como ocurriría al considerar la combinación de sus etiquetas como una clase diferente.

- **Rank-SVM**: En Elisseeff and Weston [2001] también se expone una técnica basada en SVM que los autores denominan Rank-SVM. Es la solución que proponen para aquellos casos en los que existen correlaciones entre las etiquetas asociadas a cada muestra, caso en el que, como apuntan, los clasificadores binarios no ofrecen los mejores resultados. Por ello definen una medida que les permite obtener un ranking de etiquetas minimizando una métrica denominada AHL, una aproximación lineal de la métrica *Hamming Loss* (véase Sección 1.6), considerando un modelo SVM en el que una serie de hiperplanos separan unas clases de otras. Este sistema de ranking se complementa con una función predictora que ajusta el umbral, permitiendo así ofrecer un subconjunto de etiquetas para cada muestra procesada. En la experimentación, la conclusión es que a pesar de que Rank-SVM toma en consideración la correlación entre etiquetas, sus resultados no son mejores que la del enfoque binario cuando los datos tienen unas ciertas características: un gran número de atributos de entrada y pocas instancias para entrenar.

- **Rank-CVM**: Basada en Rank-SVM, la propuesta de Xu [2013] es una adaptación en la que se trata de reducir la complejidad computacional del clasificador. Para ello se opta por utilizar una variante de las SVM tradicionales, conocida como CVM (*Core Vector Machine*), cuya solución analítica es inmediata y mucho más eficiente. El resultado es un clasificador multietiqueta con el mismo rendimiento que Rank-SVM pero un orden de magnitud más rápido.

- **SCRank-SVM**: La propuesta hecha en Wang et al. [2014] también parte del algoritmo Rank-SVM, siendo su objetivo mejorar tanto el rendimiento en clasificación como en complejidad computacional. Para ello los autores proponen reformular el cálculo de frontera de decisión de la SVM, eliminando uno de los parámetros usados habitualmente y simplificando así las

restricciones para maximizar el margen, de ahí la denominación SCRank-SVM (*Simplified Constraint Rank-SVM*).

Métodos basados en instancias

A diferencia de los algoritmos de aprendizaje inductivos, los basados en instancias no construyen un modelo a través de un proceso previo de entrenamiento, sino que operan de manera perezosa seleccionando los k vecinos más cercanos (kNN, *k Nearest Neighbors*) en el momento en que se ha de procesar una nueva muestra. Existen varias adaptaciones de esta técnica al campo de la clasificación multietiqueta, entre ellas:

- **ML-kNN**: En Zhang and Zhou [2007] se describe el algoritmo ML-kNN (*Multi-Label kNN*), en el que para cada instancia a clasificar se toman los k vecinos más cercanos, se genera un ranking de las etiquetas que tienen asignadas y se usa dicha información para calcular la probabilidad de aparición de cada etiqueta en la muestra a clasificar. Utilizan como medida de distancia entre vecinos la distancia euclídea, lo cual demuestra que ésta es una buena medida dados los resultados obtenidos. El cálculo de la pertenencia o no de una muestra a una etiqueta se realiza con técnicas bayesianas, calculando una probabilidad a priori, determinada por la frecuencia de cada etiqueta en toda la población de entrenamiento, y una probabilidad a posteriori, calculada con las etiquetas de la vecindad de la muestra a clasificar.

- **IBLR-ML**: Basándose en ML-kNN, en Cheng and Hüllermeier [2009] proponen dos algoritmos similares que mejoran al anterior aplicando métodos bayesianos. La propuesta parte de que el algoritmo ML-kNN funciona según el modelo BR, creando un clasificador binario para cada etiqueta sin tener en cuenta las correlaciones entre estas. El algoritmo que proponen considera las etiquetas asociadas a los vecinos más cercanos de la instancia a clasificar como características adicionales de dicha instancia. Con esta información calculan unas probabilidades a priori y obtienen una ecuación de regresión.

A partir de ahí plantean dos variantes del algoritmo: IBLR-ML (*Instance-based Logistic Regression for Multi-label Classification*) e IBLR-ML+. Para el correcto funcionamiento del algoritmo propuesto, sin embargo, es necesario ajustar previamente un parámetro α que determina el peso de esos atributos adicionales en el cálculo de la probabilidad a posteriori, mediante un proceso de adaptación que exige la exploración de todo el dataset para aplicar un método de estimación de parámetros estadísticos.

- **kNNc**: Introducido en Calvo-Zaragoza et al. [2014], este algoritmo funciona en dos fases en las que se combina la selección de prototipos con la clasificación basada en instancias o kNN. En la primera se emplea un conjunto reducido obtenido mediante la selección de prototipos para determinar el conjunto de clases más cercanas a las presentes en la instancia que va a clasificarse. En la segunda se utiliza el conjunto de datos original completo, pero limitando la predicción a las clases determinadas en el paso previo.

- **BRkNN, LPkNN, BRkNN-new**: Estos algoritmos son combinaciones de BR y LP con kNN. Los dos primeros se describen en Spyromitros et al. [2008] y son fundamentalmente la suma de una transformación BR o LP y el uso de kNN para determinar los vecinos más cercanos y, a partir de ellos, generar el conjunto de etiquetas a predecir. El tercero (Genga et al. [2014]) es una mejora de BRkNN en el que se utiliza información de concurrencia de etiquetas para mejorar la predicción.

Métodos probabilísticos

- **CML y CMLF**: En Ghamrawi and McCallum [2005] se destaca el hecho de que los métodos basados en clasificadores binarios no tienen en cuenta las correlaciones existentes entre las etiquetas (asumen que son independientes), proponiéndose un método que se apoya precisamente en esa información para mejorar la precisión de los resultados. Para recoger las correlaciones entre etiquetas se usa un CRF (*Conditional Random Field*), un modelo probabilístico de tipo discriminativo, usado habitualmente en segmentación

de textos, que asocia una probabilidad a una etiqueta dependiendo de las observaciones realizadas. Con él se representan las dependencias entre los valores de salida. Para cada par de etiquetas, respecto a una instancia dada, se contemplan cuatro posibles estados: no se le asocia ninguna, se le asocian las dos, se le asocia la primera o se le asocia la segunda. Se analizan dos modelos distintos: CML (*Collective Multi-Label*) y CMLF (*Collective Multi-Label with Features*). El primero mantiene parámetros de correlación para cada pareja de etiquetas, mientras que el segundo examina ternas del tipo *atributo-etiqueta1-etiqueta2*. El modelo CMLF se basa en que la correlación entre dos etiquetas dadas viene determinada por la presencia de una cierta característica en las muestras. El modelo CML mantiene un parámetro de correlación adicional entre las dos etiquetas, mientas que CMLF lleva dicho parámetro a cada terna característica-pareja de etiquetas. El algoritmo que usan descarta aquellas palabras (que son los atributos que se usan para etiquetar los textos) cuya frecuencia de aparición es menor a un cierto límite inferior. Con el resto se crea el modelo CMLF para establecer la correlación entre la presencia de una palabra y la asignación de una pareja de etiquetas al documento. Los resultados experimentales sobre el corpus de texto de Reuters prueba que el método propuesto mejora la precisión del uso de clasificadores binarios por un margen muy importante.

- **PMM1 y PMM2**: Modelos generativos probabilísticos de clasificación multietiqueta introducidos en Ueda and Saito [2002]. Se proponen dos variantes de PMM (*Probabilistic Mixture Model*), siendo PMM2 más flexible que PMM1 al utilizarse un enfoque distinto de aproximación de los parámetros de funcionamiento del algoritmo. El objetivo, como en muchas otras propuestas, es la clasificación automática de textos, en este caso estimando las probabilidades de las etiquetas a partir de las de los términos que aparecen en los documentos.

- **PCC**: Los autores de Cheng et al. [2010] proponen una extensión de CC en la que se recurre a un método bayesiano para componer las cadenas de clasificadores de forma óptima. Con este fin se modelan las dependencias entre

etiquetas calculando la distribución conjunta completa de todas ellas, determinando cuál es el encadenamiento óptimo. Aunque los resultados de PCC (*Probabilistic Classifier Chains*) mejoran a los de CC, lo hace a expensas de un coste computacional mucho más elevado.

- **TNBCC**: El mayor inconveniente de PCC y otros métodos basados en cadenas de clasificadores es la complejidad computacional, lo cual limita en la práctica el número de etiquetas con que pueden ser utilizados dichos algoritmos. En Sucar et al. [2014] se proponen varias extensiones basadas en un enfoque distinto, denominado BCC (*Bayesian Chain Classifier*), definiéndose el algoritmo TNBCC (*Tree Naïve BCC*). En este se utilizan redes bayesianas para modelar las dependencias entre etiquetas, reduciendo el número de combinaciones a considerar en las cadenas de clasificadores.

- **CRBM**: Introducido en Li et al. [2015], el algoritmo CRBM (Conditional Restricted Boltzmann Machine) utiliza una RBM para capturar las dependencias entre etiquetas a fin de construir un modelo capaz de operar sobre MLDs con conjuntos de etiquetas incompletos. Las RBM (Smolensky [1986]) han demostrado su eficacia en la generación de características de alto nivel a partir de los atributos de entrada de un conjunto de datos, apareciendo como un componente habitual en técnicas de *deep learning*. Una RBM es un modelo gráfico probabilístico, un grafo con dos capas, la de entrada y una capa oculta en la que se modelan las relaciones entre los atributos de entrada. En el algoritmo CRBM esta técnica se utiliza disponiendo las etiquetas de salida en la capa de entrada esperando obtener en la capa oculta un modelo de dependencia que permita predecir etiquetas ausentes en las muestras de datos.

Otros métodos

- **HG**: Los autores de Sun et al. [2008] persiguen el objetivo de capturar en el modelo las relaciones existentes entre las etiquetas, para lo cual recurren a un algoritmo basado en hipergrafos, en el que cada etiqueta se representa

por una hiperarista y que da lugar a un problema computacionalmente difícil de resolver pero que, bajo ciertas premisas, puede simplificarse y ser abordado con algoritmos pensados para el problema de los mínimos cuadrados.

- **CLAC**: Uno de los problemas que plantea la clasificación multietiqueta es que el número de subconjuntos distintos de etiquetas es potencialmente de $2^{|L|}$, generando un problema secundario aún más importante: que el número de muestras que comparten un mismo subconjunto (una misma clase global) puede ser muy reducido. Es lo que en Veloso et al. [2007] denominan *disjuncts* que, de ser ignorados dada su poca representatividad, pueden inducir un modelo poco preciso ya que en un conjunto de datos de gran tamaño puede haber múltiples casos así. Los autores proponen un algoritmo de clasificación perezoso, que en el momento en que llega una instancia a clasificar toma como referencia los atributos de esa muestra y se filtra el conjunto de datos descartando atributos e instancias sin relación. El resultado es la reducción del conjunto de entrenamiento y, en consecuencia, que los *disjuncts* adquieran más relevancia. Además el clasificador propuesto, denominado CLAC (*Correlated Lazy Associative Classifier*), mantiene información sobre la correlación entre etiquetas usando un mecanismo similar al del modelo CC. De manera iterativa, con un algoritmo tipo *greedy*, se va incorporando a las muestras información sobre las etiquetas que tienen asociadas.

- **GACC**: (*Genetic Algorithm for ordering Classifier Chains*) Gonçalves et al. [2013] utiliza un algoritmo genético para optimizar el orden en un clasificador de tipo CC, en lugar de recurrir a un ensemble como ECC que, por la introducción aleatoria de orden, dificulta la interpretación del modelo obtenido

- **MuLAM**: Los algoritmos basados en colonias de hormigas (Dorigo et al. [1999]) suelen aplicarse a problemas de optimización (ACO, *Ant Colony Optimization*), pero existen métodos que, como Ant-Miner (Parpinelli et al.

[2002]), permiten la extracción de reglas que permiten usarlo en problemas de clasificación. En Chan and Freitas [2006], basándose en Ant-Miner, crean un algoritmo llamado MuLAM (*Multi-Label Ant Miner*) capaz de trabajar con problemas de clasificación multietiqueta.

- **ML-KMPSO**: La propuesta de Liang et al. [2013] es un método en el que se combinan los algoritmos kNN y MPSO (*Michigan Particle Swarm Optimization*). Tras calcular la probabilidad a priori de las etiquetas en el conjunto de datos, se recurre a MPSO para optimizar la selección de los vecinos más cercanos a la instancia a clasificar, obteniendo un grupo de partículas expertas que serán las que determinen el conjunto de etiquetas a predecir.

- **GEP-MLC**: En Ávila et al. [2009] los autores proponen un clasificador multietiqueta basado en funciones discriminantes optimizadas mediante GEP (*Gene Expression Programming*, Ferreira [2002]), una modalidad de programación genética especialmente adecuada para problemas de regresión que también ha sido aplicada anteriormente en clasificación tradicional. El algoritmo aprende una o más funciones discriminantes para cada etiqueta, por lo que internamente sería equivalente a una transformación de tipo BR en la que cada clasificador es una función optimizada mediante GEP.

- **MLC-ACL**: Introducido en R. Alazaidah and Al-Radaideh [2015], este método es una combinación de transformación y adaptación de algoritmo. El procedimiento consta de tres fases: en la primera se convierte en MLD en un conjunto de datos con una sola etiqueta, siguiendo el criterio de la etiqueta menos frecuente; en la segunda se aplica un algoritmo de clasificación para una sola etiqueta basado en reglas, y en el tercero, que es un paso iterativo, se emplea el algoritmo Apriori para detectar correlaciones entre etiquetas y así convertir las reglas de clasificación en un clasificador multietiqueta.

1.5.3. Métodos basados en multi-clasificadores

Uno de los enfoques más utilizados a la hora de abordar un problema de clasificación multietiqueta, por su demostrada efectividad, es el uso de multi-clasificadores (*ensembles*) o conjuntos de clasificadores. Varios de ellos están basados parcialmente en los métodos de transformación BR y LP, anteriormente descritos. De hecho el propio método BR podría incluirse en esta categoría, ya que recurre a un conjunto de clasificadores binarios para resolver la tarea en cuestión. Varias de las aportaciones más destacables en este grupo son:

- **RPC**: El método propuesto en Hüllermeier et al. [2008] introduce una etiqueta ficticia que representa una medida nula o punto cero cuyo objetivo es facilitar la calibración del clasificador, de forma que las etiquetas por encima de ella serían las predichas finalmente y las que estén por debajo se descartan. El método, llamado RPC (*Ranking by Pairwise Comparison*), usa un modelo binario para cada par de etiquetas que determina si una está por encima de la otra o viceversa (en el ranking). El resultado de todos los modelos genera el ranking global tomando el de cada pareja como un voto. Aplican esta técnica sobre un algoritmo previo, denominado MLPC (*Pairwise Multilabel Perceptron*), obteniendo una adaptación llamada CMLPC (*Calibrated MLPC*) que se propone como mejora del RPC.

- **CC/ECC**: En Read et al. [2011] se propone un algoritmo denominado CC *Classifier Chains* en el que un conjunto de clasificadores binarios se encadenan de forma que las predicciones de cada uno se acumulan como entradas para el siguiente. Siendo L el conjunto de etiquetas, se crean $|L|$ clasificadores de forma que el primero predice usando los atributos de x_i, el segundo agrega a ese vector de atributos uno adicional que será 0 o 1, según la predicción del primero para su etiqueta, y así sucesivamente hasta que el clasificador $|L| - simo$ se entrene con los atributos de la muestra y todas las predicciones previas. La adición de la predicción de clasificadores previos en la cadena como información adicional permite usar la información de correlación entre etiquetas, mejorando así la precisión de los resultados.

Este encadenamiento, sin embargo, impide el funcionamiento en paralelo de los clasificadores. Es fácil darse cuenta de que el funcionamiento de esta cadena de clasificadores binarios se verá influido por el orden en que se tomen las etiquetas, ya que para el entrenamiento del primero no se dispone de información de los demás, mientras que el último tiene información de todos los demás. Por ello el segundo método, denominado ECC (*Ensemble Classifier Chain*), genera un conjunto de cadenas de clasificadores tomando aleatoriamente el orden de las etiquetas y un subconjunto de los datos. Cada cadena del *ensemble* facilitará unos resultados que son tomados como votos para obtener el conjunto de etiquetas asociado a la muestra a clasificar.

- **CLR**: Otro método de clasificación multietiqueta basado en un conjunto de clasificadores binarios es CLR (*Calibrated Label Ranking*, Fürnkranz et al. [2008]). En este caso se opta por un enfoque uno-contra-uno (OVO, *One-Versus-One*) en lugar de OVA, por lo que se genera un clasificador binario por cada pareja posible de etiquetas. Como resultado se genera un ranking de etiquetas potencialmente relevantes para cada instancia, aplicando un sistema de voto para generar dicho ranking. El problema de complejidad lineal de BR se convierte, por tanto, en un problema cuadrático, al existir $\mid L \mid (\mid L \mid -1)/2$ clasificadores binarios. A estos se suma un clasificador adicional por cada etiqueta en los que se introduce una etiqueta artificial, cuyo objetivo es facilitar el ajuste del umbral de corte. Dicha etiqueta representa un punto artificial de corte entre etiquetas relevantes y no relevantes.

- **EPS**: Basándose en la técnica de *pruned sets*, consistente en una transformación LP y posterior poda de combinaciones no frecuentes, en Read et al. [2008] también se propone el algoritmo EPS (*Ensemble of Pruned Sets*). Este consiste básicamente en la generación de m clasificadores PS y la definición de un mecanismo de voto que, a diferencia de LP y PS de forma aislada, es capaz de predecir combinaciones de etiquetas que no aparecen originalmente en los datos.

- **RAkEL**: La agrupación de clasificadores de tipo LP es la base de la propuesta hecha en Tsoumakas and Vlahavas [2007] con el algoritmo RAkEL (Random k-Labelsets). El método consiste en utilizar un conjunto de clasificadores de tipo LP, entrenando cada uno de ellos con un pequeño subconjunto de las etiquetas existentes. De esta manera se evitan los problemas que plantea la generación de un único clasificador con todas las combinaciones posibles de etiquetas y, al tiempo, se tiene en cuenta la información de correlación entre etiquetas. Se ejecutan m iteraciones tomando en cada una, y de manera aleatoria, un subconjunto de k etiquetas de L, sin reemplazamiento y entrenando un clasificador. Si $k = 1$ y $m = |L|$ se tiene el modelo binario BR. Si $k = |L|$ y $m = 1$ tenemos el modelo LP. BR y LP, por tanto, podrían verse como casos específicos del algoritmo RAkEL, por lo que puede decirse que los autores lo que han hecho es generalizar y parametrizar los dos métodos de transformación más usuales. A la hora de clasificar una instancia, esta es procesada por los distintos clasificadores obteniendo para cada etiqueta una serie de predicciones que son promediadas. Si el valor final es superior a un umbral dado, en el rango $[0, 1]$, entonces se asigna la etiqueta a la muestra.

- **HOMER**: Sobre la misma base que RAkEL, HOMER recurre a una estrategia jerárquica en la que, mediante un algoritmo de agrupamiento (*clustering*), el conjunto de inicial de muestras de entrenamiento se divide recursivamente en varios grupos, generando un árbol de clasificadores cada uno de ellos para un subconjunto cada vez más reducido de etiquetas. Durante el procesamiento de nuevas instancias estas entran por la raíz del árbol, en el que está el clasificador global para todas las etiquetas, y de ahí va transfiriéndose recursivamente a aquellos hijos activados por la predicción hecha en el nivel previo. De esta forma se consigue un mejor rendimiento para MLDs con un gran número de etiquetas.

- **RF-PCT**: Introducido en Kocev et al. [2007], este método utiliza como base árboles de decisión multi-objetivo, con capacidad para predecir varias salidas simultáneamente. Estos son una variante de los PCT (*Predictive*

Clustering Trees). El multiclasificador se forma utilizando el enfoque *random forest*, de ahí la denominación RF-PCT (*Random Forest of Predictive Clustering Trees*). Como es habitual al operar con RF, se recurre al *bagging* para generar diversidad en los clasificadores obtenidos, seleccionando un subconjunto de atributos de entrada diferente para cada árbol.

- **CDE**: En Tenenboim-Chekina et al. [2010] los autores de del algoritmo ChiDep presentan también CDE (*ChiDep Ensemble*), un multiclasificador cuyo clasificador base es ChiDep. El proceso comienza generando aleatoriamente un gran conjunto de particiones de labelsets, a continuación se calcula el χ^2 para cada uno de ellas y las primeras m se utilizan para generar los clasificadores del *ensemble*.

- **EML**: Un tipo particularmente interesante de multi-clasificador es el EML (*Ensemble of Multilabel Learners*) propuesto en Tahir et al. [2012a], al estar formado por un conjunto heterogéneo de clasificadores multietiqueta. En EML se combinan las salidas de ECC, MLkNN, IBLR-ML, RAkEL y CLR utilizando distintas técnicas de voto, uniendo así las ventajas de cada uno de esos clasificadores por separado. El principal problema de este enfoque estriba en el rendimiento, ya que varios de los clasificadores del *ensemble* son, a su vez, *ensembles* de clasificadores más simples.

- **CT**: La propuesta de Read et al. [2015] parte también de la idea básica de las cadenas de clasificadores, como ECC o BCC, construyendo un modelo al que denominan CT (*Classifier Trellis*). El algoritmo establece una estructura apriori, el enrejado (*trellis*), para el sistema multiclasificador, lo cual reduce la complejidad computacional al no tener que descubrirla a partir de los datos. Sobre esa estructura se introducen cambios únicamente en el orden en que las etiquetas aparecen en ella, recurriendo para ello a una heurística simple basada en la frecuencia entre pares de etiquetas. El objetivo final es obtener el buen rendimiento de las cadenas de clasificadores pero rompiendo la limitación de estas en cuanto a su escalabilidad.

Tomando como base *ensembles* de clasificadores binarios de tipo BR, RPC o CLR se han construido otros en los que la aportación se centra en ajustar el sistema de voto por el cual se obtiene la predicción final. Es el caso, por ejemplo, de DLVM (*Dual Layer Voting Method*, Madjarov et al. [2011]) y de QCLR (*QWeighted CLR*, Mencía et al. [2010]).

Muchos de los multiclasificadores usan sistemas de voto a partir de los que se genera un ranking de etiquetas, siendo preciso aplicar un umbral de corte, que por defecto suele ser 0.5, para determinar qué etiquetas son relevantes. En Quevedo et al. [2012] los autores proponen distintas estrategias para establecer dicho umbral, tomando en consideración la probabilidad a posteriori de todas las etiquetas y la relación entre estas, definiendo una formulación a medida para distintas funciones de pérdida.

En los últimos años han sido publicadas varias revisiones y comparaciones sobre métodos de clasificación multietiqueta. En Madjarov et al. [2012] se efectúa una comparación experimental entre múltiples clasificadores a fin de determinar su rendimiento. El trabajo de Zhang and Zhou [2014] se concentra en el análisis de ocho algoritmos concretos, mientras que en Gibaja and Ventura [2014] se aborda una enumeración mucho más extensa, no solo de clasificadores sino también de otras técnicas relacionadas con este campo.

Tabla 1.2: Clasificadores multietiqueta

Categoría	**Subcategoría**	**Nombre**	**Referencia**
Transformación	Binaria OVA		
		BoosTexter	Schapire and Singer [2000]
		BR	Godbole and Sarawagi [2004]
		2BR	Tsoumakas et al. [2009]
		CC	Read et al. [2011]
		BR+	Alvares-Cherman et al. [2012]
	Binaria OVO		
		MMP	Crammer and Singer [2003]
		RPC	Hüllermeier et al. [2008]
		CLR	Fürnkranz et al. [2008]

Tabla 1.2: Clasificadores multietiqueta

Categoría	Subcategoría	Nombre	Referencia
	Comb. etiquetas		
		LP	Boutell et al. [2004]
		PS	Read et al. [2008]
	Mixtos		
		ChiDep	Tenenboim-Chekina et al. [2010]
Adaptación	Árboles		
		ML-C4.5	Clare and King [2001]
		ADTBoost.MH	De Comité et al. [2003]
		ML-TREE	Wu et al. [2014]
		LaCova	Al-Otaibi et al. [2014]
	SVM		
		Rank-SVM	Elisseeff and Weston [2001]
		Model-x	Boutell et al. [2004]
		Rank-CVM	Xu [2013]
		SCRank-SVM	Wang et al. [2014]
	Redes neuronales		
		BP-MLL	Zhang [2006]
		I-BP-MLL	Grodzicki et al. [2008]
		ML-RBF	Zhang [2009]
		CCA-ELM	Kongsorot and Horata [2014]
		FPSO-MLRBF	Agrawal et al. [2014]
		FSVD-MLRBF	Agrawal et al. [2014]
	Instancias		
		ML-kNN	Zhang and Zhou [2007]
		BRkNN	Spyromitros et al. [2008]
		LPkNN	Spyromitros et al. [2008]
		IBLR-ML	Cheng and Hüllermeier [2009]
		kNNc	Calvo-Zaragoza et al. [2014]
		BRkNN-new	Genga et al. [2014]
	Probabilísticos		
		PMM1/PMM2	Ueda and Saito [2002]
		CML/CMLF	Ghamrawi and McCallum [2005]

Tabla 1.2: Clasificadores multietiqueta

Categoría	Subcategoría	Nombre	Referencia
Adaptación		PCC	Cheng et al. [2010]
		TNBCC	Sucar et al. [2014]
		CRBM	Li et al. [2015]
	Otros		
		MuLAM	Chan and Freitas [2006]
		CLAC	Veloso et al. [2007]
		HG	Sun et al. [2008]
		GEP-MLC	Ávila et al. [2009]
		GACC	Gonçalves et al. [2013]
		ML-KMPSO	Liang et al. [2013]
		MLC-ACL	R. Alazaidah and Al-Radaideh [2015]
Multiclasificadores		RF-PCT	Kocev et al. [2007]
		RAkEL	Tsoumakas and Vlahavas [2007]
		EPS	Read et al. [2008]
		RPC	Hüllermeier et al. [2008]
		CLR	Fürnkranz et al. [2008]
		HOMER	Tsoumakas et al. [2008]
		QCLR	Mencía et al. [2010]
		CDE	Tenenboim-Chekina et al. [2010]
		DLVM	Madjarov et al. [2011]
		ECC	Read et al. [2011]
		EML	Tahir et al. [2012a]
		CT	Read et al. [2015]

1.6. Métricas de evaluación

Evaluar el rendimiento de un clasificador tradicional es una tarea relativamente simple, ya que las predicciones hechas por el mismo únicamente pueden ser correctas, la salida es igual a la clase de la instancia evaluada, o incorrectas, en caso contrario. Este tipo de evaluación también puede aplicarse sobre

clasificadores multietiqueta, de forma que únicamente se consideren correctas las predicciones en las que aparecen todas las etiquetas relevantes y solamente ellas. Este, no obstante, suele considerarse un enfoque excesivamente estricto. Por esta razón fue necesario diseñar nuevas métricas de evaluación del rendimiento en clasificación, a la medida de este nuevo problema.

Actualmente el número de métricas de evaluación disponibles supera la veintena. Estas pueden ser agrupadas según distintos criterios:

- **Basadas en muestras vs basadas en etiquetas**: Las métricas basadas en muestras (Schapire and Singer [2000], Godbole and Sarawagi [2004] y Ghamrawi and McCallum [2005]) determinan el rendimiento del clasificador evaluando cada instancia por separado y promediando los valores de todas las instancias. En contraposición, las métricas basadas en etiquetas (Tsoumakas and Vlahavas [2007]) evalúan la medida en cada etiqueta por separado, tras lo cual puede obtenerse el promedio según dos criterios Tsoumakas et al. [2010]:
  - *Macro-averaging*: La métrica se calcula por separado para cada una de las etiquetas y después se promedia para obtener la evaluación global.
  - *Micro-averaging*: Se acumulan los resultados de cada etiqueta y después se calcula la métrica en cuestión.
- **Basadas en bipartición vs basadas en ranking**: La salida de un clasificador puede tomar la forma de una partición binaria, en la que se indica únicamente qué etiquetas son relevantes y cuáles no, o bien forma de ranking, aportando un grado de relevancia para cada etiqueta. Aplicando un umbral de corte cualquier ranking puede convertirse en una bipartición. Las métricas basadas en bipartición utilizan el conteo de aciertos y fallos para evaluar el rendimiento, mientras que las basadas en ranking utilizan los grados de relevancia para facilitar una medida equivalente.

La Figura 1.4 representa esquemáticamente esta taxonomía de métricas de evaluación, desde las más genéricas, en la parte superior, a las métricas concretas,

en la inferior. En los siguientes apartados de esta sección se describe cada una de ellas. En la Tabla 1.3 se especifica para cada una, la categoría y subcategoría a la que pertenecen, si han de maximizarse o minimizarse y el número de la ecuación correspondiente.

Figura 1.4: Taxonomía de métricas de evaluación multietiqueta.

1.6.1. Métricas basadas en ejemplos

Este tipo de métricas reciben como entrada la predicción completa para cada una de las muestras del conjunto a evaluar. Esa predicción puede ser una partición binaria o un ranking. Las métricas basadas en ejemplos y que toman como entrada una bipartición son las enumeradas a continuación por orden alfabético:

Tabla 1.3: Métricas de rendimiento en clasificación multietiqueta

Categoría	Subcategoría	Max|Min	Nombre	Ecuación
Basadas en ejemplos	Bipartición	Max	Accuracy	1.3
		Max	F-measure	1.4
		Min	Hamming Loss	1.5
		Max	Precision	1.6
		Max	Recall	1.7
		Max	Subset Accuracy	1.8
	Ranking	Max	Average Precision	1.9
		Min	Coverage	1.10
		Min	One Error	1.11
		Min	Ranking Loss	1.12
Basadas en etiquetas	Bipartición	Max	MacroMet	1.13
		Max	MicroMet	1.14
	Ranking	Max	MacroAUC	1.15
		Max	MicroAUC	1.16

- **Accuracy**: Se define (véase la Ecuación 1.3) como la proporción de etiquetas predichas correctamente con respecto al total de etiquetas, tanto reales como predichas, para cada instancia. La medida global se obtiene promediando el resultado de todas las instancias. Es considerada una de las medidas mas intuitivas para evaluar el rendimiento de un clasificador multietiqueta.

$$Accuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}. \tag{1.3}$$

- **F-measure**: Como se aprecia en la Ecuación 1.4, esta métrica se construye como la media armónica de *Precision* (Ecuación 1.6) y *Recall* (Ecuación 1.7.

Su objetivo es ofrecer una visión del rendimiento del clasificador balanceada entre precisión y sensibilidad.

$$F\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{1.4}$$

- **Hamming Loss**: Es probablemente la medida más utilizada en la literatura especializada. Mediante el cálculo de la diferencia simétrica obtiene un conteo de errores, tanto falsos positivos como falsos negativos, para cada instancia. El número de diferencias es dividido entre el número total de etiquetas, obteniendo una proporción que se agrega para todo el dataset y, finalmente, se promedia, tal y como se aprecia en la Ecuación 1.5. Esta métrica es complementaria de *Accuracy*, por lo que $HammingLoss = 1 - Accuracy$.

$$HammingLoss = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|}. \tag{1.5}$$

- **Precision**: La precisión del clasificador determina la proporción de etiquetas predichas que son realmente relevantes para una muestra, calculándose según la Ecuación 1.6. Como las demás medidas basadas en ejemplos, el valor individual de cada muestra se agrega y promedia para todo el conjunto de datos.

$$Precision = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|}. \tag{1.6}$$

- **Recall**: También conocida como *sensibilidad* y *exhaustividad*, esta medida suele utilizarse conjuntamente con la anterior a fin de determinar qué proporción de etiquetas relevantes han sido predichas por el clasificador. El cálculo se efectúa según la Ecuación 1.7.

$$Recall = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i|}. \tag{1.7}$$

- **Subset Accuracy**: También conocida como *0/1 Subset Accuracy* y *Classification Accuracy*, es la métrica más estricta ya que únicamente contabiliza (véase la Ecuación 1.8) como correctos los casos en que hay una igualdad completa entre el conjunto de etiquetas real y el predicho por el clasificador.

$$SubsetAccuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}[\![Y_i = Z_i]\!]. \tag{1.8}$$

Cuando la salida entregada por el clasificador no es una partición binaria, sino un ranking de etiquetas o un conjunto de probabilidades de pertenencia, pueden utilizarse las siguientes medidas:

- **Average Precision**: Esta medida determina para cada una de las etiquetas relevantes de la instancia la proporción de etiquetas que están delante de ella en el ranking. Para ello, en la Ecuación 1.9 *rank(*x_i, y*)* se define como una función que para la instancia x_i y la etiqueta relevante y perteneciente a Y_i, cuya posición es conocida, facilita el valor que en el ranking Z_i ocupa esa posición. El objetivo es determinar cuántas posiciones en promedio hay que moverse en el ranking para encontrar una etiqueta relevante. La media de ese número de posiciones para todo el conjunto de datos es el valor final.

$$AveragePrecision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\{y'|rank(x_i,y') \leq rank(x_i,y), y' \in Y_i\}|}{rank(x_i,y)}. \tag{1.9}$$

- **Coverage**: Esta métrica, definida según la Ecuación 1.10, obtiene para una instancia los pasos que hay que avanzar en el ranking devuelto por

el clasificador hasta encontrar todas las etiquetas relevantes. Ese valor se promedia para todas las instancias.

$$Coverage = \frac{1}{|D|}\sum_{i=1}^{|D|} \operatorname{argmax}_{y \in Y_i} \langle rank(x_i, y) \rangle - 1. \tag{1.10}$$

- **One Error**: El objetivo de esta medida es determinar cuántas veces la etiqueta que ocupa el primer puesto del ranking facilitado por el clasificador no forma parte del conjunto real de etiquetas de la instancias. Para ello, según se indica en la Ecuación 1.11, se obtiene el índice de la etiqueta cuyo valor de ranking es máximo en Z_i, sumando 1 en caso de que no forme parte de Y_i.

$$OneError = \frac{1}{|D|}\sum_{i=1}^{|D|} [\![[\operatorname{argmax}_{y \in Z_i} \langle rank(x_i, y) \rangle \notin Y_i]]\!]. \tag{1.11}$$

- **Ranking Loss o RL**: Mediante esta métrica se compara cada una de las parejas posibles de etiquetas relevantes y no relevantes para una instancia dada, contándose como errores los casos en que una etiqueta no relevante queda en el ranking por delante de una que no lo es. Dicho valor, como el resto de medidas basadas en ejemplos, se agrega para todo el conjunto de datos y se promedia, tal y como se muestra en la Ecuación 1.12.

$$RL = \frac{1}{|D|}\sum_{i=1}^{|D|} \frac{1}{|Y_i||\overline{Y_i}|} |y_a, y_b : rank(x_i, y_a) > rank(x_i, y_b), (y_a, y_b) \in Y_i \times \overline{Y_i}|. \tag{1.12}$$

1.6.2. Métricas basadas en etiquetas

Las métricas de evaluación del rendimiento en clasificación multietiqueta basadas en ejemplos dan el mismo peso a cada una de las muestras existentes en el conjunto de datos. Las basadas en etiquetas pueden promediarse según dos

enfoques distintos, a los que se conoce como *macro* y *micro*. La fórmulas genéricas, que permiten calcular cualquier métrica basada en bipartición con estos dos enfoques, son las mostradas en la Ecuación 1.13 y la Ecuación 1.14. La función *evalMet()* sería la métrica a evaluar: *Accuracy*[3], *Precision*, *Recall* o *F-measure*.

$$MacroMet = \frac{1}{|L|}\sum_{l=1}^{|L|} evalMet(TP_l, FP_l, TN_l, FN_l). \tag{1.13}$$

$$MicroMet = evalMet(\sum_{l=1}^{|L|} TP_l, \sum_{l=1}^{|L|} FP_l, \sum_{l=1}^{|L|} TN_l, \sum_{l=1}^{|L|} FN_l). \tag{1.14}$$

El enfoque *macro* agrega los aciertos y fallos para cada una de las etiquetas, evalúa la métrica correspondiente y después obtiene el promedio. De esta forma otorga el mismo peso a todas las etiquetas, análogamente a como las medidas basadas en ejemplos dan el mismo peso a todas las muestras. En contraposición, el enfoque de evaluación *micro* efectúa un único cálculo de la métrica con la suma de aciertos y fallos de todas las etiquetas, ofreciendo por tanto una visión más global del rendimiento al no ponderarse por muestras o etiquetas, sino efectuando la evaluación sobre el volumen total de predicciones hechas por el clasificador.

Además de las métricas que se calculan a partir de predicciones en forma de biparticiones, también existen medidas basadas en etiquetas que operan sobre rankings. El área bajo la curva ROC (*Receiver Operating Characteristic*), conocida genéricamente como AUC (*Area Under the Curve*), puede calcularse según el enfoque *macro* atendiendo a la Ecuación 1.15 o bien con el enfoque *micro* mediante la Ecuación 1.16.

[3]En la práctica *MacroAccuracy* y *MicroAccuracy* no se calculan ya que, por su formulación, devuelven idénticos valores que la métrica *Accuracy* basada en ejemplos.

$$MacroAUC = \frac{1}{|L|}\sum_{l=1}^{|L|}\frac{|\{x', x'' : rank(x', y_l) \geq rank(x'', y_l), (x', x'') \in X_l \times \overline{X_l}\}|}{|X_l||\overline{X_l}|},$$
$$X_l = \{x_i | y_l \in Y_i\}, \overline{X_l} = \{x_i | y_l \notin Y_i\}. \tag{1.15}$$

$$MicroAUC = \frac{|\{x', x'', y', y'' : rank(x', y') \geq rank(x'', y''), (x', y') \in S^+, (x'', y'') \in S^-\}|}{|S^+||S^-|},$$
$$S^+ = \{(x_i, y) | y \in Y_i\}, S^- = \{(x_i, y) | y \notin Y_i\}. \tag{1.16}$$

Aparte de las mencionadas aquí, es posible encontrar métricas de evaluación adicionales, en general más especializadas. Un ejemplo de ello son las específicas para clasificación multietiqueta jerárquica, de las cuales la más usada es *Hierarchical Loss* (Cesa-Bianchi et al. [2006]), una extensión de *Hamming Loss* en la que se considera el nivel de la jerarquía en que se produce el error.

1.7. Tareas relacionadas

La tarea de clasificación multietiqueta está relacionada con varias otras áreas que no son objeto de estudio de esta tesis, por lo que no se abordará su estudio en mayor profundidad. Las más destacables son:

- **Ranking multietiqueta**: El objetivo de esta tarea es obtener una función que a partir de las características de entrada de cada instancia genere un ranking de relevancia que para la misma tienen las etiquetas de un conjunto predefinido. A partir de un ranking multietiqueta es posible, aplicando un umbral de corte, generar una bipartición que representaría la clasificación multietiqueta para la muestra dada (Brinker et al. [2006]). La conversión inversa no siempre es posible, ya que muchos algoritmos producen directa-

mente la citada bipartición, no dando a unas etiquetas más relevancia que a otras.

- **Clasificación multietiqueta jerárquica**: En esta variante de la clasificación multietiqueta (Cerri and de Carvalho [2010]) el conjunto de etiquetas no es lineal, sino que tiene una estructura de árbol (un solo padre por cada nodo) o de grafo (múltiples padres por cada nodo), según los casos. Los algoritmos pueden utilizar esta información relativa a la estructura jerárquica de las etiquetas para ajustar su funcionamiento, de forma que si la predicción en un cierto nodo niega la relevancia de una cierta etiqueta esta tampoco será relevante para ninguno de los nodos hijo, mientras que una etiqueta relevante automáticamente se propagaría a todos sus ascendientes.

- **Clasificación de *streams* multietiqueta**: Las secuencias de datos multietiqueta (Qu et al. [2009]) son flujos continuos de muestras que demandan un aprendizaje incremental y en tiempo real, habitualmente procesados por dispositivos con limitaciones de rendimiento (memoria y potencia de procesamiento). Un algoritmo de clasificación multietiqueta para este tipo de datos ha de ir adaptando su modelo interno de forma continua, ya que las nuevas muestras pueden inducir derivas distintas en los conceptos (*concept drift*) a lo largo del tiempo, debiendo ser capaz de clasificar instancias no etiquetadas en cualquier instante.

- **Clasificación multietiqueta y multi-instancia**: Un conjunto de datos multietiqueta y multi-instancia se caracteriza porque cada patrón se compone de múltiples partes, representadas por varias muestras relativas todas al mismo concepto. Un correo electrónico, por ejemplo, consta del cuerpo del mensaje, las direcciones de los destinatarios, el tema y otra información alojada en la cabecera. Ese conjunto de instancias sería el que se utilizaría en un clasificador multietiqueta y multi-instancia (Zhou and Zhang [2006]) para predecir el conjunto de etiquetas relevante.

- **Clasificación multidimensional**: La clasificación multietiqueta puede considerase una tarea de clasificación multiobjetivo en la que los objetivos son

binarios: las etiquetas pertenecen o no a una instancia dada. Dentro de esa misma categoría genérica también se encuentra la clasificación multidimensional (Zaragoza et al. [2011]), en la que los objetivos no son binarios sino que pueden tomar uno de un conjunto finito de valores. Cuando dichos valores son numéricos se habla de regresión con múltiples salidas.

También quedan fuera del contexto de estudio de la presente tesis otros problemas relacionados con la clasificación multietiqueta, entre ellos la selección y generación de características, la reducción de dimensionalidad en el espacio de atributos, las estrategias de particionamiento para conjuntos de datos multietiqueta, etc.

1.8. Problemática específica objeto de estudio

En los últimos años el número de publicaciones relativas a clasificación multietiqueta ha ido creciendo de manera importante (Gibaja and Ventura [2014]), alcanzando las 700 propuestas entre 2009 y 2012. La Figura 1.5, generada con el buscador Scopus, muestra el rápido crecimiento experimentado en dichos años. Esto deja patente la gran cantidad de problemas que continúan abiertos en este campo, cuyo estudio es relativamente reciente en relación con otras técnicas de aprendizaje y minería de datos.

De ese gran abanico de problemas abiertos en el área, esta tesis se centra en tres aspectos concretos: la dimensionalidad en el espacio de etiquetas, el desequilibrio en la distribución de las etiquetas y el aprovechamiento de las relaciones existentes entre estas. En los apartados siguiente se presenta cada uno de ellos como introducción a los temas tratados en capítulos siguientes.

1.8.1. Dimensionalidad en el espacio de etiquetas

Habitualmente cuando se habla de alta dimensionalidad implícitamente se está haciendo referencia a conjuntos de datos que cuentan con un gran número

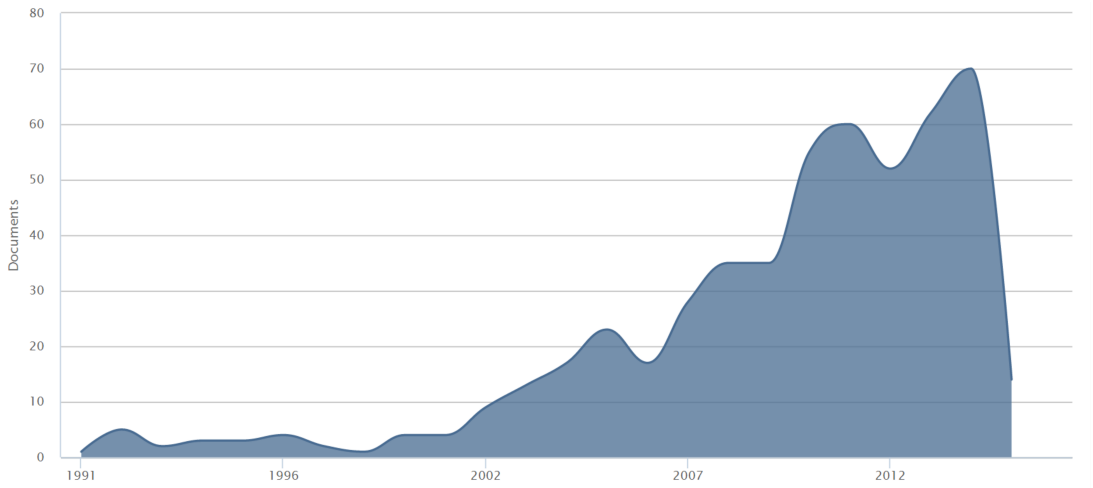


Figura 1.5: Número de publicaciones con el término *multietiqueta* en título o *abstract*.

de atributos o características de entrada. En el caso multietiqueta debe tenerse en cuenta no solamente ese factor, sino también la existencia de un gran número de etiquetas, factor ante el que ciertos métodos de transformación, como es el caso de LP, resultan casi impracticables. En el caso de la clasificación multietiqueta, por tanto, la conocida maldición de la dimensionalidad (*curse of dimensionality*, problema que dificulta el aprendizaje de forma notable) se presenta tanto en el espacio de entrada como en el de salida.

Una alta dimensionalidad en el espacio de etiquetas está directamente relacionada con el tamaño del conjunto L, más que con la cardinalidad o la densidad de etiquetas. Estas medidas, no obstante, influyen en la manera en que es posible abordar dicho problema. Si $\mid L \mid >> Card(D)$ esto implica una gran dispersión en el espacio de etiquetas, por lo que el número total de variables de salida podría reducirse aprovechando dicha característica. Sin embargo esta circunstancia no siempre se cumple, debiendo buscarse otras vías si se quiere reducir la dimensionalidad en el espacio de etiquetas.

En el capítulo 2 se describe el método desarrollado para este fin, hibridando un algoritmo de minería de reglas de asociación con un método que, actuando

a modo de filtro, permite utilizar cualquier clasificador multietiqueta subyacente sobre un espacio de etiquetas reducido.

1.8.2. Desequilibrio en la distribución de etiquetas

El aprendizaje a partir de conjuntos de datos desbalanceados es un problema bien conocido en el contexto de la clasificación tradicional, existiendo infinidad de propuestas sobre cómo abordarlo. Es un hecho generalmente aceptado que la mayoría de los conjuntos de datos multietiqueta presentan esta problemática, de forma que algunas etiquetas están raramente representadas en D mientras que otras resultan ser muy frecuentes.

Las aportaciones sobre cómo afrontar el aprendizaje multietiqueta ante la presencia de desbalanceo son, a pesar de lo dicho anteriormente, relativamente escasas. Por dicha razón la mayor parte del trabajo desarrollado en la presente tesis, al que se dedican los capítulos 3 y 4, se ocupa de este tema. En una primera fase se estudia cómo determinar el nivel de desbalanceo de los conjuntos de datos multietiqueta, así como la forma de reequilibrar la distribución de etiquetas mediante técnicas de remuestreo aleatorio. En la segunda parte se proponen algoritmos heurísticos que, aprovechando las medidas previamente introducidas, optimicen el remuestreo.

1.8.3. Caracterización de conjuntos de datos multietiqueta

A fin de poder abordar el tratamiento de un cierto problema en los datos es esencial entender las características de estos, razón por la que, como parte de este estudio, se han definido y propuesto distintas métricas relacionadas con la distribución de las etiquetas, el desequilibrio de estas o las relaciones existentes entre ellas, habiéndose presentado también vías específicas para la visualización de algunas de estas características. Estas métricas han servido como base para el desarrollo de varios de los algoritmos propuestos en los capítulos siguientes.

Todas las medidas propuestas están recogidas en distintas publicaciones, de forma que cualquier investigador puede utilizarlas a su conveniencia. Asimismo se ha efectuado una implementación de referencia de las mismas en una herramienta de análisis exploratorio de conjuntos de datos multietiqueta para el conocido entorno/lenguaje R. En el capítulo 5 se describe dicho paquete software, denominado `mldr`, incluyendo la interfaz gráfica de usuario que facilita el acceso a toda la información a aquellos no familiarizados con R.

Capítulo 2

Tratamiento de la dimensionalidad en el espacio de etiquetas

En este capítulo se describe la propuesta del algoritmo LI-MLC (*Label Inference for MultiLabel Classification*), cuyo objetivo es abordar el problema de la alta dimensionalidad en el espacio de etiquetas a fin de reducir la complejidad computacional y mejorar el rendimiento predictivo en clasificación multietiqueta. Para ello se combina el uso de reglas de asociación con el de algoritmos de clasificación multietiqueta existentes, obteniendo así un método híbrido.

El capítulo comienza introduciendo el contexto general de trabajo y haciendo una presentación de alto nivel de la propuesta. A continuación se facilitan los fundamentos relativos a los mecanismos sobre los que se asienta LI-MLC: las reglas de asociación, la obtención de información de dependencia entre etiquetas y la reducción de la dimensionalidad. Después se facilitarán los detalles sobre el diseño del algoritmo LI-MLC y la experimentación realizada para validar su funcionamiento. El capítulo finaliza con un resumen de conclusiones obtenidas a partir de este trabajo.

2.1. Introducción

En el apartado 1.1.2 del Capítulo 1 se introducía el preprocesamiento dentro del proceso de KDD. El propósito del preprocesamiento de datos puede ir dirigido a obtener unos mejores resultados en las métricas de evaluación, es quizá el objetivo más obvio, pero también es interesante su aplicación para optimizar el rendimiento computacional del clasificador, un aspecto que cobra mayor importancia cuando se emplean múltiples clasificadores y el número de etiquetas es grande, así como del incremento de simplicidad de los modelos obtenidos. El tiempo de ejecución necesario para procesar un conjunto de datos relativamente grande con ciertos algoritmos, como los basados en cadenas y *ensembles* de clasificadores binarios (véase sección 1.5.3 del capítulo previo), puede provocar que la tarea sea prácticamente inviable.

Dependiendo de la parte del conjunto de datos sobre la que se actúe, el preprocesamiento puede aplicarse en tres espacios diferenciados o bien en una combinación de estos. Denominando $|D|$ al número de instancias y $|X|$ al de atributos, dichos espacios serían:

- **Espacio de atributos**: El conjunto de datos se interpreta como un grupo de $|X|$ columnas (las características de cada patrón), cada una de ellas con $|X^j|$ valores, de las que quedarían en principio excluidas las correspondientes a las etiquetas.

- **Espacio de instancias**: El conjunto de datos se interpreta como un grupo de $|D|$ filas (las muestras de datos), cada una de ellas con $|X|$ valores en los que pueden estar incluidos o no los atributos correspondientes a las etiquetas.

- **Espacio de etiquetas**: Solamente se toman del conjunto de datos las $|L|$ columnas correspondientes a las etiquetas y se trabaja exclusivamente con los valores que toman estas en cada una de las muestras.

Los problemas a los que puede aplicarse el trabajo de preprocesamiento en cada uno de estos espacios son lógicamente distintos. En los dos primeros, el de atributos e instancias, el enfoque más obvio pasa por la aplicación de algoritmos de selección de características e instancias. El tercero representa un espacio nuevo al considerar únicamente las etiquetas y solo tiene sentido cuando la cardinalidad del conjunto de datos es superior a 1, es decir, en problemas multietiqueta, no binarios o multi-clase.

2.1.1. Preprocesamiento en el espacio de etiquetas

Hay técnicas de preprocesamiento que, como algunos algoritmos de selección de características y de instancias, emplean la clase asociada a cada muestra para analizar las correlaciones existentes entre entradas y salidas, determinando qué atributos son los que aportan mayor información útil para el modelo a construir. En clasificación multietiqueta este tipo de técnicas encuentran un obstáculo adicional: el conjunto de etiquetas de cada muestra puede llegar a ser incluso mayor que el de características de entrada pero, a pesar de ello, resultar de vital importancia para dichas tareas.

Algunos de los problemas que pueden presentarse en el espacio de etiquetas son, en cierta manera, análogos a los afrontados en el espacio de atributos: alta dimensionalidad (muchas etiquetas diferentes), redundancia, etiquetas perdidas, etc. Otros, por el contrario, no existían hasta ahora en el espacio de salida, como puede ser una cardinalidad elevada que dé lugar a multitud de combinaciones distintas. Se abre, por tanto, un nuevo espacio para las tareas de preprocesamiento que no existía hasta el momento: el que actúa exclusivamente sobre las etiquetas.

La cantidad total de etiquetas y la cardinalidad son aspectos que influyen directamente en los métodos de transformación (BR y LP) y también afectan, en mayor o menor medida, a los de adaptación. Cuando se recurre a la combinación BR+clasificador binario, por ejemplo, es preciso entrenar tantos clasificadores como etiquetas distintas haya en cada dataset. Reducir el número de etiquetas mejoraría la eficiencia del clasificador y también afectará al rendimiento en cla-

sificación. Cuando la cardinalidad es alta, la combinación LP+clasificador multiclase se encuentra con el obstáculo del gran número de combinaciones posibles que, como se apuntaba en el capítulo anterior, provoca dispersión y desbalanceo. La reducción de la cardinalidad relajaría la explosión combinatoria y, teóricamente, mejoraría el clasificador.

Analicemos en qué forma influiría la alta dimensionalidad en el espacio de etiquetas en los diferentes tipos de algoritmos de clasificación multietiqueta:

- **Algoritmos adaptados**: Con pocas excepciones, entre las que se contarían las propuestas basadas en el enfoque kNN, los algoritmos adaptados para operar con MLD han de construir un modelo que, en esencia, es una representación de las correlaciones existentes entre los atributos de entradas y las etiquetas de salida. Cuanto mayor sea el número de etiquetas tanto mayor será la complejidad del modelo, incrementándose el tiempo necesario para entrenarlo. En general, los modelos más simples tienden a ser más eficientes.

- **Clasificadores basados en BR**: Los clasificadores binarios basados en la transformación BR, así como los que están basados en binarización como es el caso de CC o ECC, han de entrenar múltiples clasificadores, generalmente uno por etiqueta. En consecuencia, al trabajar con MLD que tienen cientos de etiquetas sería preciso inducir ese mismo número de clasificadores, un proceso con una gran demanda computacional en tiempo y memoria. Cuantas más etiquetas haya mayor es la probabilidad de que existan relaciones entre ellas y, en general, este es un aspecto no tomado en consideración por este tipo de métodos.

- **Métodos combinatorios**: Combinar las etiquetas activas en cada una de las muestras, usando el resultado como identificador de clase, es una manera fácil de trabajar con datos multietiqueta recurriendo a clasificadores multiclase. Es el enfoque usado por la transformación LP y algunos otros algoritmos basados en él, pero es una opción difícilmente aplicable cuando se tiene un gran número de etiquetas por la generación exponencial de

combinaciones de etiquetas. Habitualmente también se sufre de una alta dispersión, disponiendo de pocas instancias asociadas a cada clase.

- **Multiclasificadores**: CC, ECC, RAkEL, HOMER y muchos otros métodos de clasificación multietiqueta basados en *ensembles* operan internamente con una colección de clasificadores binarios o multiclase. En consecuencia, esta opción sufre de las mismas debilidades apuntadas antes para BR y LP.

Como regla general, la reducción del espacio de salida (el número total de etiquetas) contribuirá también a disminuir el tiempo y la memoria necesarios para entrenar cualquier clasificador y generará modelos más simples que, habitualmente, funcionarán mejor. Esta es la razón de que existan múltiples propuestas (véase la Sección 2.5) cuyo objetivo es trabajar en esta dirección. La mayor parte de ellas se fundamentan en la idea de comprimir el espacio de etiquetas, proyectándolo en un espacio de menor dimensionalidad de manera que el problema original se transforma en otro tipo de tarea, como puede ser regresión o clasificación binaria. Una vez resuelto ese problema intermedio es necesario invertir la transformación de compresión, obteniendo así la predicción multietiqueta.

Una vía alternativa a la transformación del espacio de etiquetas sería la selección de algunas de ellas basándose en la información de dependencia. La correlación entre etiquetas puede ser usada para distintos fines, como se detallará la Sección 2.3. Esta idea es uno de los pilares del método cuya propuesta se introduce a continuación.

2.1.2. Presentación de la propuesta

A diferencia de lo que ocurre con los métodos de selección de características, que pueden derivar en la eliminación de atributos del conjunto de datos antes de que este sea entregado al algoritmo que entrena el clasificador, una hipotética selección de etiquetas no puede hacer exactamente lo mismo. Las etiquetas que se eliminen del conjunto original han de recuperarse con posterioridad, ya que de lo contrario sencillamente desaparecerían de los resultados de la clasificación.

Se trataría, en consecuencia, de una ocultación temporal de ciertas etiquetas, de forma que el clasificador se entrene con un conjunto reducido de ellas obteniendo así un modelo más simple (más fácil de interpretar), más eficiente (menor complejidad computacional) y, probablemente, más eficaz (mayor rendimiento en clasificación). Finalizado el entrenamiento, el clasificador ofrecería para cada muestra de test una predicción de etiquetas entre las que no estarían visibles las que se habían ocultado. La presencia o no de estas en el resultado final quedaría en manos de un proceso de postprocesamiento.

La metodología que se propone aquí se basa en la extracción de asociaciones (implicaciones) fuertes entre etiquetas que nos permitan reducir el número de etiquetas antes de aprender el clasificador multietiqueta. Una vez obtenido el clasificador, la información de dependencia de etiquetas debe añadirse a la predicción. Para ello se utilizan reglas de asociación en dos fases: preprocesamiento y postprocesamiento. La primera se iniciará con una selección de etiquetas que, básicamente, se guiará por los pasos siguientes:

- Aplicar sobre las etiquetas del MLD un algoritmo de obtención de reglas de asociación adecuado para grandes conjuntos de datos, en previsión de la necesidad de tratar con conjuntos de datos que tengan un gran número de etiquetas e instancias.
- Ordenar las reglas obtenidas según su confianza o alguna otra medida de interés y quedarse con aquellas que estén por encima de un cierto umbral.
- Para cada una de las reglas existentes en la lista (que es necesario conservar hasta el final) recorrer el consecuente, formado por una o más etiquetas que es posible deducir a partir del antecedente, eliminando del conjunto de etiquetas original aquellas que forman parte del consecuente.

Terminada esta fase se tiene un MLD con un conjunto de etiquetas reducido, por una parte, y una serie de reglas de asociación, por otra. Ese MLD reducido se entrega al algoritmo de clasificación multietiqueta para llevar a cabo el entrenamiento.

El postprocesamiento se llevará a cabo tras la predicción hecha por el clasificador para cada muestra de test. En ese instante se habrán de aplicar las reglas de asociación, guardadas tras el preprocesamiento, con el objetivo de recuperar las etiquetas que es posible deducir de las reglas. De esta forma se completará la predicción del conjunto de etiquetas que corresponde a cada muestra procesada.

2.2. Reducción de dimensionalidad

La dificultad que impone el aprendizaje a partir de un espacio de entrada de alta dimensionalidad es bien conocida. Incluso se ha designado a este hecho con una denominación que la resume perfectamente: la *maldición de la dimensionalidad*. Se trata, por tanto, de un problema profundamente estudiado desde hace tiempo en el campo de la minería de datos, siendo multitud las propuestas existentes para reducir dicha maldición en la medida de lo posible.

Existen métodos no supervisados, como PCA (*Principal Component Analysis*, Jolliffe [2005]) y LSI (*Latent Semantic Indexing*, Dumais et al. [1995]), que pueden aplicarse directamente a conjuntos de datos multietiqueta ya que no se precisa conocimiento alguno sobre las variables de salida, únicamente se usan las de entrada.

Los enfoques supervisados, que utilizan también la información del espacio de salida a fin de determinar qué atributos de entrada aportan más información útil, han de ser adaptados, como es el caso de la propuesta hecha en Park and Lee [2008] en la que se presenta un método LDA (*Linear Discriminant Analysis*) multietiqueta.

Los métodos citados permiten eliminar del espacio de entrada aquellos atributos que no aportan información significativa para el proceso de aprendizaje y, al tiempo, reducen la dimensionalidad disminuyendo el número de variables que es preciso utilizar para generar el modelo. Este, sin embargo, es un enfoque que difícilmente puede aplicarse de manera directa al espacio de salida, siendo este

un aspecto mucho menos estudiado dado que los conjuntos de datos tradicionales únicamente cuentan con una salida posible.

El conjunto de etiquetas de un MLD representa el espacio de respuestas que se espera de un clasificador, por lo que no pueden eliminarse sin más una o más variables del mismo. Cualquier método que pretenda reducir la dimensionalidad en el espacio de etiquetas ha de hacer su trabajo en dos partes:

1. Analizar qué etiquetas pueden eliminarse del conjunto de datos de entrenamiento antes de proceder con el entrenamiento del clasificador.

2. Reconstrucción de la respuesta a partir de la predicción obtenida del clasificador una vez entrenado.

Hasta ahora la mayor parte de las propuestas existentes (véase la Sección 2.5) se basan en técnicas de compresión/descompresión del espacio de etiquetas. La propuesta hecha aquí se basa en el análisis de la información de dependencia entre etiquetas.

2.3. Información de dependencia entre etiquetas

Una pieza esencial de la propuesta realizada en el presente capítulo es el mecanismo de obtención de información sobre dependencia entre etiquetas. El objetivo es que, a fin de reducir la dimensionalidad en el espacio de etiquetas, sea posible eliminar del conjunto utilizado para clasificación aquellas que puedan ser inferidas por otras vías de manera más eficiente y con un determinado nivel de confianza.

Asumiendo que se trabaje con datos reales y limpios (sin ruido, inconsistencias, etc.), el hecho de que dos o más etiquetas aparezcan juntas de manera muy frecuente sugiere la existencia de algún nivel de correlación entre ellas. La certeza y fuerza de esa dependencia es algo a analizar, ya que es una información potencialmente de utilidad para cualquier algoritmo de clasificación multietiqueta.

Es importante destacar que a la hora de obtener información de dependencia entre etiquetas (Dembczynski et al. [2010]) pueden seguirse dos caminos distintos:

- **Dependencia condicional**: Es la dependencia que se da atendiendo a una instancia concreta del conjunto de datos (los atributos de entrada influyen). Las cadenas de clasificadores de Read et al. [2011] y las cadenas probabilísticas basadas en las anteriores (Cheng et al. [2010]) utilizan este tipo de dependencia.

- **Dependencia incondicional**: Es la dependencia global entre etiquetas sin que influyan los atributos de muestras concretas, tal y como se usa en los métodos discriminantes de Godbole and Sarawagi [2004].

La que nos interesa aquí es fundamentalmente la dependencia incondicional, expresada como un modelo global de correlaciones o coocurrencias entre etiquetas.

Como se indica en Zhang and Zhou [2014], uno de los desafíos que tiene el aprendizaje multietiqueta es la reducción del espacio de salida aprovechando las correlaciones entre etiquetas. Los autores de dicho trabajo clasifican las estrategias usadas por los algoritmos multietiqueta para obtener dichas correlaciones en tres categorías: 1) primer grado, 2) segundo grado y 3) alto nivel (*high order*). El orden depende del número de etiquetas cuya dependencia se analiza: solo una, dos, o más de dos, respectivamente.

Aquí proponemos otra manera de agrupar los algoritmos, en función del método que utilizan para obtener la información de dependencia entre etiquetas. Nos podemos encontrar con los siguientes casos:

- **Ignorarla**: Algunos métodos de clasificación, quizá el exponente más notable sea BR, ignoran por completo la presencia de información de dependencia entre etiquetas, operando con clasificadores binarios totalmente independientes entre sí.

- **Implícitamente**: Ciertos métodos, como los basados en la combinación de etiquetas LP, incorporan implícitamente la información de dependencia

al usar como identificador de clase cada combinación posible. Existen, no obstante, enfoques más sofisticados para aprovechar dicha información.

- **Explícitamente**: Hay múltiples métodos, entre ellos CC, ECC Read et al. [2011], 2BR Tsoumakas et al. [2009] y BR+ Alvares Cherman et al. [2010], que extienden el espacio de características de cada clasificador binario utilizando como nuevos atributos de entrada las salidas de los demás clasificadores, mediante la composición de cadenas de clasificadores o el apilado (*stacking*) de los mismos. De esta forma se incorpora explícitamente la información sobre las relaciones existentes entre las etiquetas.

- **Mediante modelos estadísticos**: Las propuestas presentadas en Zhang and Zhang [2010], Guo and Gu [2011], Tenenboim-Chekina et al. [2010] se apoyan en métodos estadísticos, como las redes bayesianas, que representan explícitamente las dependencias entre etiquetas, o bien los tests de independencia χ^2 que asignan una puntuación de dependencia a cada pareja de etiquetas.

- **Mediante agrupamiento**: Otra vía para obtener datos de dependencia entre etiquetas consiste en agrupar las instancias y recuperar información de las etiquetas que aparecen en cada grupo. Este es el enfoque usado en HOMER (Tsoumakas et al. [2008]).

- **Otros métodos**: Además de los mencionados en los grupos previos, hay otros enfoques de extracción de información de dependencia entre etiquetas. RAkEL (Tsoumakas and Vlahavas [2007]) utiliza un *ensemble* de clasificadores entrenados con pequeños subconjuntos de etiquetas, usando LP para capturar las dependencias. PLST (Tai and Lin [2010]) usa una solución geométrica, recurriendo a proyecciones de un hipercubo en dimensiones menores a fin de obtener correlaciones entre etiquetas. En Park and Fürnkranz [2008] se utilizan reglas de asociación con el objetivo de definir restricciones durante la clasificación basadas en la dependencia entre etiquetas.

Algunas de las propuestas citadas utilizan internamente el conocimiento obtenido de dependencia entre etiquetas a fin de mejorar el proceso de aprendizaje, otros lo usan para dividir los datos de entrenamiento y construir *ensembles*, y otros la aprovechan para establecer restricciones sobre los resultados de clasificación. Ninguna de ellas se fija como objetivo reducir el espacio de etiquetas desde un principio, antes de iniciar la fase de aprendizaje.

La efectividad de los métodos de reducción del espacio de etiquetas (Weston et al. [2002], Hsu et al. [2009], Zhou et al. [2012]) y la utilidad de la información de dependencia entre etiquetas (Read et al. [2011], Zhang and Zhang [2010], Guo and Gu [2011], Tenenboim-Chekina et al. [2010]) son ambas técnicas probadas y demostradas. El trabajo aquí presentado parte de la hipótesis de que es posible usar la información de dependencia entre etiquetas para eliminar algunas etiquetas, reduciendo el espacio de salida sin recurrir a esquemas de compresión o proyección. El objetivo del algoritmo LI-MLC propuesto en este capítulo es combinar esas dos técnicas para afrontar el problema de la alta dimensionalidad en el espacio de etiquetas, apoyándose para ello en el uso de algoritmos de minería de reglas de asociación como herramienta para obtener correlaciones fuertes.

2.4. Reglas de asociación

Como se ha mencionado, en este estudio se propone una metodología de reducción de la dimensionalidad en el espacio de etiquetas para clasificación multietiqueta que utiliza reglas de asociación, por lo que en esta sección se introduce este tipo de herramienta de representación de conocimiento, la tarea de minería asociada y los algoritmos más relevantes.

Las reglas de asociación (Hipp et al. [2000]) describen coocurrencias entre elementos que aportan conocimiento sobre el sistema subyacente a un conjunto de datos y pueden ser interpretadas como implicaciones, de forma que la aparición de unos ciertos elementos predice la ocurrencia de otro. El ejemplo de uso más habitual es el del supermercado que quiere analizar la cesta de la compra para

determinar qué productos son adquiridos conjuntamente por sus clientes: el caso típico, mencionado de manera reiterada en la bibliografía sobre este tema, son los pañales y la cerveza, ya que las personas con hijos pequeños no suelen poder salir durante el fin de semana y, en consecuencia, se proveen anticipadamente de ambos productos. A partir de esa información el comerciante decide colocar cerca del primer producto el segundo, para inducir a su compra y evitar que los padres se olviden de la cerveza cuando van a por los pañales o viceversa, o bien aplicar cualquier otra política o técnica de marketing en base al conocimiento de que ambos productos suelen adquirirse de forma conjunta.

Al proceso de extracción de reglas de asociación a partir de una base de datos o conjunto de transacciones es a lo que se conoce como minería de reglas de asociación, existiendo múltiples algoritmos para llevarlo a cabo. En Hipp et al. [2000] se hace una revisión de los más comunes, muchos de ellos basados en el más conocido: el algoritmo Apriori introducido por Agrawal et al. [1994].

Aunque las reglas de asociación suelen emplearse como una herramienta descriptiva, que permite obtener conocimiento sobre unos hechos, también han sido utilizadas con otros fines como son la clasificación (Ma et al. [1998]) mediante los denominados sistemas CARs (*Classification Association Rules*). La minería de reglas para clasificación se basa en la obtención de un conjunto reducido de reglas que tienen un cierto valor mínimo de soporte y confianza, de manera que su evaluación puede servir para determinar la clase de las muestras de datos a partir del valor de sus atributos.

Nuestro interés en las reglas de asociación estriba, tal y como se explicará más adelante, en su uso como herramienta para ocultar al clasificador multietiqueta aquellas etiquetas que puedan ser deducidas a partir de la aparición de otras con un cierto nivel de confianza. En el siguiente apartado se introducen los conceptos generales sobre minería de reglas de asociación y en el posterior se detalla el algoritmo elegido para su extracción.

2.4.1. Conceptos generales y medidas

Sea $I = x_1, \ldots, x_n$ un conjunto de objetos o elementos distintos, a los que llamaremos genéricamente ítems. En el ejemplo del supermercado cada ítem sería un producto a la venta. Se denomina *k-itemset*, o sencillamente itemset, a un subconjunto $X \subseteq I, k = |X|$. k puede tomar valores entre 1 y n, el número de ítems existentes, y para un cierto valor de k existirán múltiples combinaciones diferentes, por lo que existirá un determinado número de itemsets distintos pero del mismo tamaño, salvo para $k = n$ caso en el que solamente habría uno posible.

Sea $D = T_1, \ldots, T_n$ un conjunto de subconjuntos de I, de forma que cada T_i es un itemset de un cierto tamaño al que se llama transacción. En el ejemplo del supermercado D sería la base de datos en la que se registran las ventas y cada transacción la venta realizada a un cliente, compuesta de varios ítems.

Tomando un itemset $X \subseteq I$, se dice que una cierta transacción T da soporte a dicho itemset si se cumple que $X \subseteq T$, es decir, si todos los ítems contenidos en X están también en T. A la fracción de transacciones en D que dan soporte a un cierto itemset X es a lo que se denomina soporte o cobertura de X, expresado según se indica en la Ecuación 2.1.

$$supp(X) = \frac{\mid \{T \in D | X \subseteq T\} \mid}{\mid D \mid}. \tag{2.1}$$

A los itemsets cuyo soporte supera un cierto valor umbral, normalmente fijado de antemano a la ejecución del algoritmo de minería de reglas de asociación, se les llama **itemsets frecuentes**. El primer paso de todo algoritmo de extracción de reglas será la obtención de una lista con los itemsets frecuentes que existan en la base de datos.

Una regla de asociación se define como una implicación del tipo $X \Rightarrow Y$ en la que tanto X (antecedente) como Y (consecuente) son itemsets: $X, Y \subseteq I$ y se cumple que $X \cap Y = \emptyset$. El soporte de una regla se obtiene a partir del soporte de antecedente y consecuente, tal y como se muestra en la Ecuación 2.2.

$$supp(X \Rightarrow Y) = supp(X \cup Y). \tag{2.2}$$

Lo habitual es buscar reglas que tengan un soporte relativamente alto, ya que aquellas con un soporte bajo pueden haber aparecido por simple casualidad y no ser representativas de un comportamiento colectivo. Es un aspecto en el que también influirá el tamaño del conjunto de transacciones con que se trabaje.

El soporte de una regla indica la frecuencia con que aparece en la base de transacciones la unión de los itemsets que forman antecedente y consecuente, pero no nos dice nada sobre la precisión de la implicación, es decir, la frecuencia con que al presentarse el antecedente lo hace también el consecuente. Para medir este factor se recurre a la confianza, medida como se muestra en la Ecuación 2.4

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}. \tag{2.3}$$

Lo que ofrece esta medida es una estimación de la probabilidad condicional $P(Y/X)$ y el objetivo es que esta sea lo más alta posible. Una regla con una alta confianza indica que existe una correlación entre antecedente y consecuente. Si $conf(X \Rightarrow Y) = 1$ ello indicará que en todas las transacciones en que aparece X también lo hace Y y que, por lo tanto, podría asumirse que la primera implica la presencia de la segunda. Por el contrario, una confianza baja puede indicar que no existe relación alguna entre esos itemsets más allá de la casual.

Ha de tenerse en cuenta que las reglas de asociación no son entidades reversibles: $X \Rightarrow Y \neq Y \Rightarrow X$. En ambos casos el soporte sería el mismo, pero no así la confianza:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \neq \frac{supp(X \cup Y)}{supp(Y)} = conf(Y \Rightarrow X). \tag{2.4}$$

Si bien la confianza es la medida más empleada a la hora de seleccionar reglas de asociación generadas por un algoritmo de minería de reglas, existen medidas alternativas como puede ser el interés de una regla (también conocida como *lift*) introducida en Brin et al. [1997] y definida como se muestra en la Ecuación 2.5.

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)supp(Y)}. \tag{2.5}$$

Al calcular la confianza de una regla de asociación podría darse el caso de que $supp(X \cup Y)/supp(X) = supp(Y)$ en caso de que X e Y sean itemsets independientes (desde un punto de vista estadístico). El interés, al tomar en consideración tanto el soporte de X como el de Y en el denominador, asume esa independencia pero, tal y como indican los autores, es una medida que indica grado de coocurrencia, pero no de implicación, ya que es completamente simétrica y se cumple que $lift(X \Rightarrow Y) = lift(Y \Rightarrow X)$.

Aunque la métrica más común para medir la calidad de reglas de asociación es *Conf*, su uso conjunto con *Supp* es conocido como el marco confianza-soporte (Silverstein et al. [1998]), la discusión relativa a reglas e implicación en Brin et al. [1997] establece que en ocasiones puede obtenerse un valor alto de *Conf* sin que exista una implicación del antecedente respecto al consecuente, lo cual es una debilidad para los fines del trabajo desarrollado aquí. Hay disponibles otras medidas para evaluar reglas de asociación, entre ellas la medida convicción (*Conv*) definida en Brin et al. [1997] a la que corresponde la Ecuación 2.6. Esta medida se ajusta mejor a las necesidades del método que está proponiéndose, ya que se trata de una medida direccional. Un alto valor en esta métrica indica, sin que importe cuál sea el soporte de la regla, que existe una implicación entre antecedente y consecuente.

$$Conv(X \Rightarrow Y) = \frac{1 - Supp(Y)}{1 - Conf(X \Rightarrow Y)}. \tag{2.6}$$

2.4.2. Minería de reglas de asociación

La extracción de reglas de asociación consiste en encontrar todas aquellas reglas que tienen un soporte y confianza mínimos a partir de una base de transacciones. El problema es que el espacio de búsqueda crece exponencialmente respecto a $|I|$ (número de ítems distintos), por lo que puede llegar a ser inabordable. El proceso se divide en dos partes diferenciadas tal y como se indica en Agrawal et al. [1994]:

1. Obtener el conjunto F formado por todos los itemsets frecuentes, asociando a cada uno de ellos el soporte que le corresponda.

2. Para cada uno de los itemsets en F se calcula la confianza de todas las reglas del tipo $X \backslash Y \Rightarrow Y,\ Y \subseteq X,\ X \neq Y \neq \emptyset$ y se conservan aquellas que alcanzan la confianza mínima preestablecida (o el valor mínimo para la medida alternativa, si no se emplea la confianza).

La generación de itemsets que tienen un mínimo soporte (frecuentes), fijado de antemano, es un problema combinatorio. Partiendo de los *1-itemset*, conjuntos formados solamente por un ítem, hay que calcular el soporte de cada uno y, a continuación, formar todas las combinaciones posibles de *2-itemset*, repitiendo el proceso hasta generar todos los *k-itemset* posibles.

El algoritmo más conocido para minería de reglas de asociación a partir de una base de datos o dataset es Apriori (basado en el principio del mismo nombre) introducido en Agrawal et al. [1994], que reduce en cierta medida el problema combinatorio usando la técnica de generación de itemsets candidatos. Esta se basa en una sencilla premisa: si un cierto *k-itemset* no es frecuente (no alcanza el soporte mínimo), tampoco lo serán aquellos *k+1-itemsets* en los que esté contenido. A pesar de ello su uso no es apropiado con grandes bases de datos o itemsets compuestos de muchos elementos dada su complejidad computacional y el hecho de que recorre la colección de transacciones múltiples veces.

De los distintos algoritmos descritos en la bibliografía especializada como alternativa a Apriori se ha optado, por su rendimiento, por el algoritmo FP-Growth (Han et al. [2004]). En el apartado siguiente se resume su funcionamiento.

2.4.3. El algoritmo FP-Growth

Frente al algoritmo Apriori clásico, FP-Growth es un método en el que se prescinde de la generación de itemsets candidatos, es decir, se elimina la parte más costosa de la extracción de reglas. Los patrones o itemsets frecuentes se producen a partir de una estructura de datos: el FP-Tree. Dicha estructura de datos, y el propio algoritmo en sí, agregan algo de complejidad a la implementación (respecto a Apriori) pero el resultado es mucho más eficiente.

La primera fase del algoritmo es la encargada de construir la estructura de datos, denominada FP-Tree, para lo cual da únicamente dos pasadas sobre la base de datos que aloja las transacciones:

1. Durante la primera pasada se obtiene un recuento de la aparición de cada uno de los valores (itemsets de longitud 1) existentes en la BDD (Base de datos) de transacciones, al final de la cual se eliminan aquellos que no llegan al soporte mínimo (fijado de antemano) y se genera una lista ordenada por soporte de mayor a menor.

2. Apoyándose en la citada lista ordenada, una segunda pasada sobre la BDD procede a generar el FP-Tree paso a paso con cada transacción procesada. Partiendo de un nodo raíz nulo, por cada transacción leída se hace lo siguiente:

    - Se toma el elemento con mayor soporte de los que almacena la transacción y, partiendo de la raíz, se busca en el primer nivel del árbol. Si no está se añade y se inicializa su contador a 1, si está se incrementa el contador asociado.

- Asumiendo que el nodo del elemento anterior es la nueva raíz, se toma el siguiente elemento con mayor soporte en la transacción y se repite la operación. De esta forma se va creando una ruta descendente entre los nodos de los diferentes niveles.
- Procesados todos los elementos de la transacción se lee la siguiente y se reinicia el proceso.

Concluida la segunda pasada se tiene en memoria el FP-Tree con todos los enlaces entre nodos padres e hijos, formando un árbol *n-ario* que no estará completo hasta que se agreguen enlaces entre apariciones de un mismo elemento en diferentes ramas del árbol. Dichos vínculos facilitarán la navegación por el árbol. Lo que se hace es recorrer este, agregando a una tabla cada elemento y la referencia a los nodos en que aparece, después se recorre la tabla y por cada entrada con más de un nodo se vinculan estos entre sí.

El FP-Tree es una estructura compacta, normalmente mucho más reducida que los datos originales, y servirá para extraer los itemsets frecuentes que, a la postre, es lo que perseguimos. Lo interesante es que ese proceso ya no requerirá recorrer la BDD de transacciones ni generar candidatos (como en Apriori).

El procedimiento de extracción de patrones frecuentes a partir del FP-Tree es realmente el cometido del algoritmo FP-Growth. Basándose en la técnica del *divide y vencerás*, este algoritmo parte de las hojas del FP-Tree y va generando subárboles para cada conjunto de hojas que compartan el mismo elemento, sirviéndose de los enlaces padre-hijo y también de los vínculos (añadidos previamente) entre elementos que aparecen en distintas ramas.

Cada subárbol representa un conjunto de rutas o caminos que comparten un mismo elemento en la hoja pero con distintos prefijos (si hay más de un camino). En un proceso recursivo, se corta el nivel inferior de esos subárboles y se repite la operación anterior obteniendo todos los árboles posibles para cada elemento hasta llegar a la raíz. A partir de los prefijos, y tras un recuento interno que permite ordenar los elementos según su frecuencia, se obtienen árboles condicionales para el elemento que se había cortado.

El último paso del algoritmo es la obtención a partir de cada subárbol de los itemsets frecuentes, usando para ello los enlaces padre-hijo entre los nodos y los contadores asociados. Básicamente se toma el elemento de la hoja como itemset frecuente de longitud 1, a continuación se le pone como prefijo las combinaciones de elementos de los nodos padre que tenga para los itemset de longitud 2 y así sucesivamente.

2.5. Trabajos previos

Antes de proceder a la descripción del algoritmo LI-MLC, en esta sección se enumeran las propuestas más importantes publicadas hasta ahora en relación con la reducción de la dimensionalidad en el espacio de etiquetas. El objetivo es facilitar un análisis breve de la relación entre estas investigaciones previas y la propuesta que se presenta aquí.

- **Selección de subconjuntos de etiquetas**: La descomposición del conjunto de etiquetas en múltiples subconjuntos más pequeños es un enfoque utilizado en varios algoritmos, como RAkEL (Tsoumakas and Vlahavas [2007]) y HOMER (Tsoumakas et al. [2008]). Ambos utilizan subconjuntos de etiquetas para entrenar varios clasificadores multiclase, apoyándose para ello en la transformación LP. RAkEL compone los subconjuntos aleatoriamente, mientras que HOMER recurre a un agrupamiento jerárquico a fin de obtener subconjuntos de etiquetas relacionadas. Al final se usan todas las etiquetas para entrenar algún clasificador, dado que el espacio de etiquetas no se reduce sino que es dividido en partes. El clasificador subyacente utilizado por estos métodos es un multiclase, no un clasificador multietiqueta. En contraposición, LI-MLC reduce el espacio de etiquetas antes de que el clasificador comience a hacer su trabajo, por lo que habrá etiquetas que no participen en el proceso. Asimismo LI-MLC no transforma el problema original en uno distinto, dado que no impone el uso de un tipo específi-

co de clasificador. Es posible utilizar cualquier algoritmo de clasificación multietiqueta sobre los datos una vez reducidos.

- **Poda de conjuntos de etiquetas poco frecuentes (PS)**: En Read et al. [2008] también se propone un método basado en la transformación LP. El espacio de etiquetas es reducido mediante la poda de aquellos conjuntos de etiquetas cuya frecuencia de aparición no supera un cierto umbral. Estos conjuntos son después descompuestos en otros más simples, con menos etiquetas, reintroduciéndose aquellos que superan el citado umbral. De esta manera se capturan de forma implícita solo las relaciones más importantes entre las etiquetas, mejorando la capacidad de generalización del clasificador. En el mismo trabajo también se propone un *ensemble* basado en dicha idea (EPS). Como ocurre con RAkEL y HOMER, el clasificador subyacente es de tipo multiclase, no multietiqueta. En consecuencia, no es un método de aplicación general independiente del problema que se aborda o el algoritmo de clasificación elegido, como en el caso de LI-MLC. Además LI-MLC obtiene las dependencias entre etiquetas mediante reglas de asociación, midiendo las correlaciones entre ellas, mientras que PS y EPS obtienen dicha información implícitamente, como parte de los conjuntos de etiquetas que aparecen en el conjunto de datos.

- **KDE (*Kernel Dependency Estimation*)**: Esta técnica, propuesta en Weston et al. [2002], no está diseñada específicamente para reducir el espacio de etiquetas de un MLD, sino que es una vía de propósito general para buscar dependencia entre un conjunto de entradas y un conjunto de salidas, siendo la clasificación multietiqueta una de sus potenciales aplicaciones. Mediante un análisis de componentes principales del espacio de etiquetas se obtiene un conjunto de proyecciones independientes, pudiendo reducirse el número de salidas seleccionando, en esta fase, solamente las representaciones más significativas (aquellas que tienen mayores *eigenvalues*). La asociación entre las entradas y esta representación de las salidas puede ser aprendida de forma independiente, por ejemplo utilizando un algoritmo de regresión. El paso final, con las predicciones independientes obtenidas de los regresores,

consiste en aplicar una función para buscar la preimagen de la proyección y obtener así el conjunto final de salidas. Para este pueden utilizarse distintas funciones, dependiendo de la tarea concreta que esté abordándose. Para un problema de clasificación los autores proponen buscar la solución entre un conjunto de candidatos adquiridos desde el conjunto de datos de entrenamiento. El enfoque de LI-MLC es totalmente diferente. El espacio de etiquetas original es usado para aprender las correlaciones entre las etiquetas, en lugar de ser transformado en representaciones independientes. Solo las etiquetas que puedan ser inferidas con un cierto nivel de confianza son eliminadas, dando como resultado un nuevo MLD. En consecuencia el problema sigue siendo de clasificación multietiqueta, por lo que puede ser afrontado con algoritmos para tal fin. Lo más importante es que en la fase final LI-MLC obtiene predicciones multietiqueta, que solo hay que complementar con las etiquetas inferidas a partir de las reglas. No hay necesidad de recurrir a algoritmos para buscar la preimagen de una proyección.

- **CS (*Multilabel Prediction Via Compressed Sensing*)**: La propuesta hecha en Hsu et al. [2009] está basada en una técnica muy probada de compresión llamada CS (*Compressed Sensing*), en la que se establece que la complejidad de un modelo con k etiquetas puede ser reducido al entrenamiento de $\mathcal{O}(\log(k))$ modelos más simples. Existe, no obstante, una premisa que es necesario cumplir: el MLD debe presentar un nivel significativo de dispersión. Se trata, por tanto, de un enfoque adecuado para MLD que tienen un gran conjunto de etiquetas distintas, pero en cuyas instancias solo aparece un pequeño subconjunto de las mismas. Esta es la naturaleza de los dos conjuntos de datos usados por los autores en su experimentación. El procedimiento seguido es similar al descrito para KDE. La fase de compresión se lleva a cabo mediante proyecciones aleatorias del espacio de etiquetas binario original, obteniendo una representación en un espacio real (no binario) de menor dimensionalidad. Hecho esto, dichas proyecciones son usadas para entrenar un conjunto de modelos de regresión. Finalmente la clasificación de los nuevos patrones se lleva a cabo con este conjunto de re-

gresores, cuyas salidas han de ser descomprimidas para obtener las etiquetas a predecir. La única similitud entre este enfoque y LI-MLC se basa en la existencia de una fase de preprocesamiento, que reduce el espacio de salida, y un postprocesamiento que está a cargo de la reconstrucción. LI-MLC puede aplicarse sin asumir la existencia de dispersión en el espacio de etiqueta. De hecho, si el espacio de etiquetas es muy disperso la extracción de reglas de asociación puede resultar más difícil como se explicará más adelante. También ha de tenerse en cuenta que lo que se propone en Hsu et al. [2009] es un algoritmo de clasificación multietiqueta basado en compresión y regresión, sin posibilidad de usar otro tipo de clasificador subyacente, mientras que LI-MLC, como ya se ha indicado, actúa como un envoltorio en torno a cualquier clasificador multietiqueta, teniendo por tanto un mayor campo de aplicación.

- **CL (*Compressed Labeling on Distilled Label Sets*)**: Basándose en la idea de CS, en Zhou et al. [2012] se propone un método que es una variación para la compresión del espacio de etiquetas manteniendo el uso de algoritmos de clasificación para realizar las predicciones, en lugar de recurrir a la regresión. En esta propuesta hay dos elementos clave: 1) el método para extraer del MLD los subconjuntos de etiquetas más frecuentes, denominados DL (*distilled label sets*), y 2) la transformación del espacio real de menor dimensionalidad obtenido por las proyecciones aleatorias en un espacio binario. Esto último se consigue utilizando los signos de las proyecciones aleatorias. En cuanto al primero, aplica un enfoque de agrupamiento recursivo sobre el espacio de etiquetas. Una vez que se ha obtenido el espacio de etiquetas comprimido, se usa un clasificador binario para predecir cada etiqueta de forma independiente. Estas predicciones son después complementadas usando la información de correlación almacenada en los DL. Los pasos seguidos por CL son los mismos que en LI-MLC. Primero se obtiene información de dependencia de las etiquetas. Para ello CL utiliza los DL, mientras que LI-MLC genera un conjunto de reglas de asociación. Segundo, se comprime el espacio de etiquetas. En CL este paso se lleva a cabo me-

diante proyecciones aleatorias, mientras que LI-MLC lo completa eliminando las etiquetas que pueden ser inferidas a partir de las reglas previamente obtenidas. Tercero, las instancias con el espacio de etiquetas reducido son entregadas a un algoritmo de clasificación para obtener las predicciones iniciales. CL puede usar cualquier clasificador binario para esta tarea, recuperando predicciones independientes para cada etiqueta. LI-MLC puede usar cualquier clasificador multietiqueta, obteniendo predicciones conjuntas. Finalmente, esas predicciones iniciales son complementadas y se devuelve el conjunto de etiquetas final para cada muestra. En CL este proceso se apoya en los DL y la aplicación de tests estadísticos, mientras que en LI-MLC se realiza con la inferencia de las reglas de asociación.

Una de las mayores diferencias entre LI-MLC y las propuestas que acaban de enumerarse radica en que todas ellas transforman el problema de clasificación multietiqueta original en otro tipo de problema: clasificación multiclase, regresión o clasificación binaria, mientras que LI-MLC preserva la naturaleza multietiqueta original. Además LI-MLC no depende de un mecanismo de descomposición del objetivo global en múltiples subobjetivos más simples que pueden ser entrenados individualmente y cuyas salidas es preciso recombinar después. Una vez que se han extraído las etiquetas señaladas por las reglas de asociación, solo es preciso entrenar un clasificador. Las salidas de este son complementadas con la inferencia de las reglas, sin necesidad de fusionar salidas de varios predictores ni buscar preimágenes con algún algoritmo de descompresión.

2.6. Inferencia de etiquetas para clasificación multietiqueta

En esta sección del capítulo se facilitan los detalles de diseño y funcionamiento del algoritmo LI-MLC, nuestra propuesta para reducir la dimensionalidad en el espacio de etiquetas. En el primer apartado se describe detalladamente el algo-

ritmo. En el segundo se estudia en qué casos sería aplicable y qué medidas de caracterización de los MLD podrían servir como guía para tomar dicha decisión.

2.6.1. El algoritmo LI-MLC

El funcionamiento de LI-MLC se divide en tres etapas: 1) preprocesamiento, 2) entrenamiento del clasificador y 3) postprocesamiento. Estas pueden distinguirse claramente en el pseudo-código mostrado en el Algoritmo 1. El objetivo de cada una de estas fases el siguiente:

- **Preprocesamiento**: Toma como entrada la partición de entrenamiento del MLD, generando como salida una versión reducida de la misma, con menos etiquetas, y un conjunto de reglas de asociación con la forma mostrada en la Ecuación 2.7.
- **Entrenamiento del clasificador**: La partición de entrenamiento reducida es entonces usada para entrenar cualquier clasificador multietiqueta.
- **Postprocesamiento**: Una vez que se ha obtenido la predicción hecha por el clasificador para cualquier muestra, la evaluación de las reglas de asociación infiere las etiquetas que es preciso agregar para obtener la predicción final.

$$L_i, ..., L_j \rightarrow L_k, ..., L_m \quad L_x \in L. \tag{2.7}$$

En la fase de preprocesamiento LI-MLC toma cada etiqueta como un *item* o elemento de una transacción, y el conjunto de etiquetas asociado a cada muestra se interpreta como la transacción completa. Las instancias X_i del conjunto de entrenamiento son procesadas (líneas 11-14) extrayendo los correspondientes *labelsets* L_i, y usando cada uno de ellos como una transacción. La base de transacciones T generada a partir de esos datos es entregada al algoritmo FP-Growth, obteniendo un conjunto R de reglas de asociación que ordenamos de mayor a menor según la medida *Conviction* a fin de aplicar primero la regla más fuerte.

Algorithm 1 Pseudo-código de LI-MLC

procedure LI-MLC(X) ▷ Dataset a procesar
  $DTra \leftarrow trainingPartition(X)$
  $DTst \leftarrow testPartition(X)$
  $DTra', Rules \leftarrow preprocessing(DTra)$
  $Classifier \leftarrow obtainClassifier(DTra')$
  $P \leftarrow postprocessing(Classifier, DTst, Rules)$
  $Evaluate(P)$
end procedure
procedure PREPROCESSING($DTra$)
  $T \leftarrow \emptyset$ ▷ Conjunto de transacciones
  for each instance X_i in $DTra$ **do**
    $L_i \leftarrow labelsOf(X_i)$
    $T \leftarrow T \cup L_i$
  end for
  $R \leftarrow FPGrowth(T)$
  $R \leftarrow orderByConviction(R)$
  ▷ Eliminar reglas no aplicables
  for each R_i in R **do** ▷ De mayor a menor conviction
    $LC_i \leftarrow labelInConsequent(R_i)$
    $deleteRemainRulesWhoseConseqHas(LC_i)$
    if $isOnlyLabelInSomeSample(LC_i)$ **then**
      $deleteRule(R_i)$
    end if
  end for
  ▷ R tiene el conjunto final de reglas a aplicar
  $C \leftarrow labelsInConsequent(R)$
  $DTra \leftarrow DTra - C$ ▷ Partición de entrenamiento reducida
  return $DTra, R$
end procedure
procedure POSTPROCESSING($Classifier, DTst, R$)
  $P \leftarrow \emptyset$ ▷ Conjunto de predicciones
  for each instance X_i in $DTst$ **do**
    $P_i \leftarrow Classifier(X_i)$
    for each R_i in R **do**
      $P_i \leftarrow P_i \cup applyRule(R_i)$
    end for
  end for
  return P
end procedure

Teniendo el conjunto de reglas de asociación R a su disposición, LI-MLC toma las etiquetas que aparecen en el consecuente de cada regla obteniendo el conjunto de etiquetas a eliminar. Dado que la misma etiqueta podría aparecer en el consecuente de más de una regla, la cardinalidad del conjunto de etiquetas podría ser inferior a la del conjunto de reglas. En esta situación únicamente la regla con el valor más alto de *Conviction* será utilizada, descartándose de R el resto (línea 20). Además, hay casos en que una etiqueta no puede ser eliminada por ser la única presente en algunas instancias del conjunto de datos, lo cual reduce el número de etiquetas que es posible eliminar (líneas 21-23) en ciertos casos.

Tomando el conjunto final C de etiquetas a eliminar como referencia, el método genera una versión de la partición de entrenamiento sin ellas (línea 27). Este es el conjunto reducido de entrenamiento, con menos atributos de salida que el original, entregado como entrada a los algoritmos de clasificación multietiqueta elegidos. La complejidad de este proceso depende del algoritmo de minería de reglas de asociación utilizado, pero es importante destacar que el preprocesamiento solo es necesario llevarlo a cabo una vez por partición de datos dado que el algoritmo de minería de reglas de asociación usado es determinístico. En consecuencia, una vez generada la partición reducida de entrenamiento esta podría ser empleada para entrenar múltiples clasificadores.

La fase de postprocesamiento obtiene las predicciones P_i, hechas por el clasificador multietiqueta, para cada muestra de datos (línea 33). Esta predicción está incompleta, la predicción final debe reflejar las dependencias entre etiquetas extraídas anteriormente. Para ello LI-MLC evalúa el conjunto de reglas R, añadiendo las etiquetas representadas por cada consecuente. Este es un proceso iterativo (líneas 34-36) en el que las reglas son tomadas en el mismo orden que se siguió en la fase de preprocesamiento, activando la etiqueta del consecuente en cada paso si la muestra que se procesa tiene activas las etiquetas indicadas en el antecedente de la regla.

De esta forma se genera la predicción final que es devuelta como resultado. La complejidad de este proceso es lineal con respecto al número de reglas, siempre muy inferior al número de muestras de datos o atributos de entrada.

2.6.2. Aplicación de LI-MLC a conjuntos multietiqueta

Como bien es sabido, no existe un algoritmo que sea capaz de generar el mejor resultado en todos los casos posibles. Las características de los conjuntos de datos usados influyen en el comportamiento de los algoritmos. Por ello es importante estudiar apriori dichas características a fin de, en la medida de lo posible, extraer pautas que ayuden a decidir cuándo debe aplicarse y cuando no.

La metodología descrita en el apartado previo, el algoritmo LI-MLC, será útil si las etiquetas de un MLD no tienen una distribución excesivamente dispersa. Obviamente, si la métrica *Card* es muy baja será casi imposible obtener correlaciones entre etiquetas, dado que el número de muestras con dos o más etiquetas no resultará suficiente (poco soporte). No obstante, la citada métrica no nos servirá por sí sola para determinar si hay solo unas pocas o bien muchas muestras conteniendo dos o más etiquetas. Esta métrica es un promedio del número de etiquetas en todo el conjunto de datos, sin ninguna información sobre cómo están distribuidas dichas etiquetas en las distintas instancias. Tampoco la métrica *Dens* será de utilidad, ya que es simplemente *Card* dividida entre el número total de etiquetas. Es necesario caracterizar los MLD con métricas adicionales.

La dispersión de las etiquetas en un MLD podemos conocerla calculando el coeficiente de variación (CV), según la Ecuación 2.8. En esta, σ denota la desviación estándar y μ la media. Esta es una métrica utilizada habitualmente en problemas de desbalanceo (Cieslak and Chawla [2008]). Si un MLD tiene un valor de *Card* no muy bajo (hay más de una etiqueta por muestra) y su CV es relativamente bajo (las etiquetas están distribuidas uniformemente), la probabilidad de encontrar correlaciones será mayor. Con valores altos de CV (alta dispersión) podría ser difícil obtener reglas, incluso cuando *Card* es también alto.

$$CV = \frac{\sigma}{\mu}. \tag{2.8}$$

Otra posibilidad para determinar la distribución de las etiquetas la encontramos en dos métricas estadísticas básicas (Groeneveld and Meeden [1984]): la curtosis (KR) o apuntamiento, expresada como se indica en la Ecuación 2.9, y el grado de asimetría o *skewness* (SK), obtenido según la Ecuación 2.10. En este contexto, x_i ha de interpretarse como una muestra de datos. KR es un indicador del nivel de concentración de la distribución en torno a un cierto valor, mientras que SK lo es de dónde se encuentra dicho valor.

$$KR = E\left(\frac{x_i - \mu}{\sigma}\right)^4 - 3. \tag{2.9}$$

$$SK = E\left(\frac{x_i - \mu}{\sigma}\right)^3. \tag{2.10}$$

Cuanto más alto sea el valor de KR tanto más apuntada estará la distribución de la variable y, en ese caso, tanto más concentrado estará el número de etiquetas por muestra en torno a un mismo valor. Por otra parte, SK es una medida de asimetría y puede ser negativa o positiva. Cuando la distribución es apuntada (alto KR) un SK positivo denotará que la mayor parte de los valores están por debajo de la media (se reparten a su izquierda). Por el contrario, un valor negativo implicaría lo opuesto y la mayoría de los valores estaría por encima de la media (repartidos a su derecha). Un valor de SK en torno a cero sugiere una distribución simétrica.

En la Figura 2.1 se ha representado en un histograma la distribución de etiquetas en cuatro MLD. Dos de ellos `scene` y `bibtex`, tienen un valor alto de KR y un valor SK por encima de cero, por lo que la mayoría de las muestras tienen una sola etiqueta asociada. En el caso de `emotions` y `corel5k` el valor de KR es cercano a cero, lo que significa que la distribución es más uniforme. En el prime-

ro, SK también está cercano a cero, por lo que se deduce que la mayoría de las muestras tienen el número de etiquetas indicado por *Card*. El segundo muestra un valor negativo para SK y, por tanto, el número de etiquetas por instancia estará por encima de la media para muchas de las muestras. Es fácil ver que la obtención de correlaciones entre etiquetas resultará más difícil en el caso de `scene` y `bibtex` que en el de `emotions` y `corel5k`.

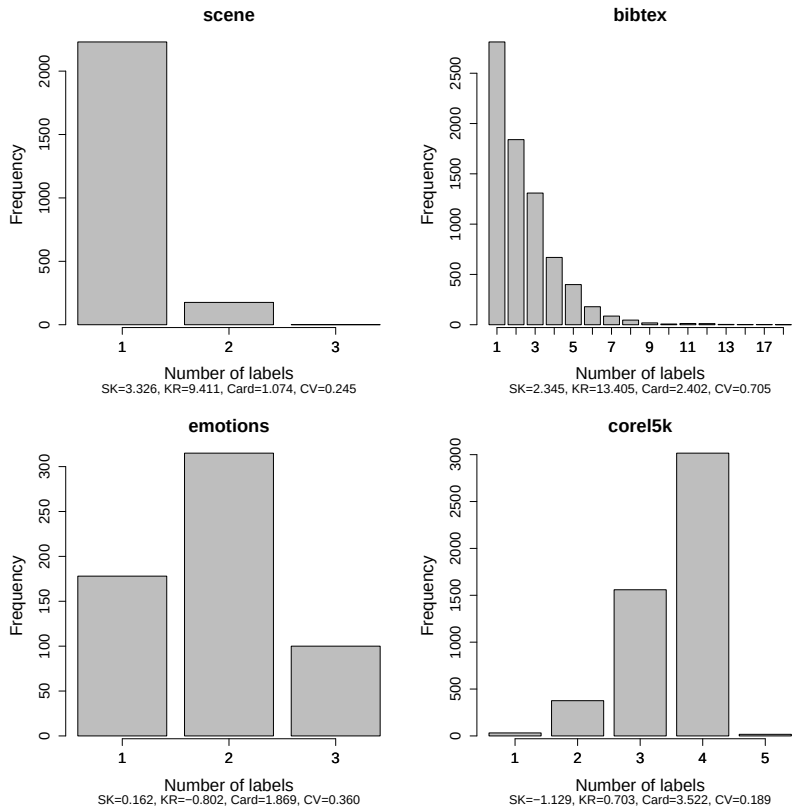


Figura 2.1: Histograma de la distribución de etiquetas por muestra.

Otra forma de ver la distribución de las etiquetas por instancia en un conjunto de datos sería la mostrada en la Figura 2.2. Cada fila (eje Y) es una muestra y las columnas (eje X) representan las diferentes etiquetas. La presencia de una etiqueta en una cierta muestra se denota como una línea cruzando ambos ejes.

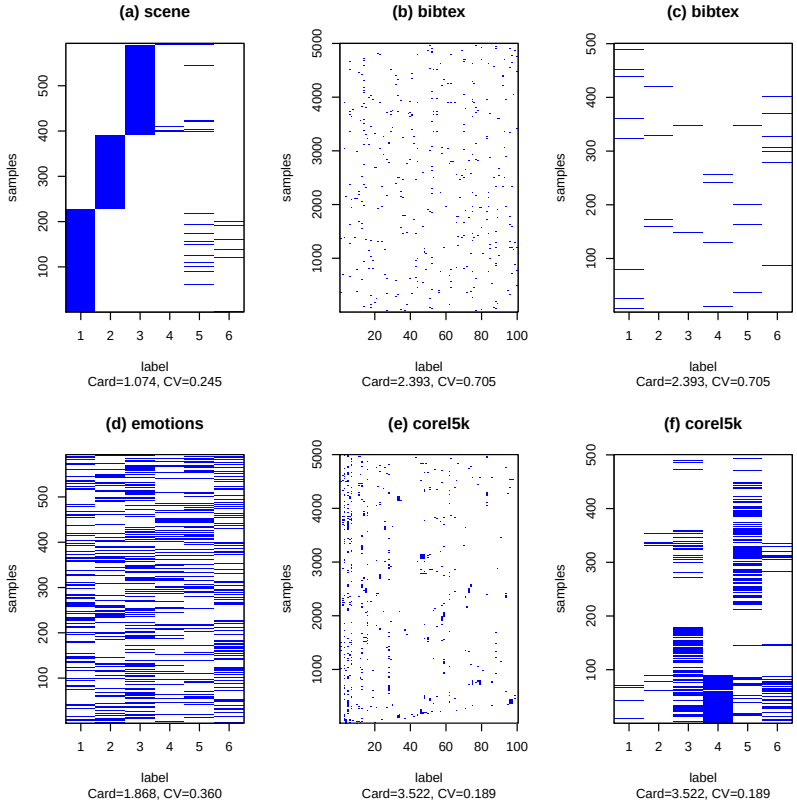


Figura 2.2: Mapa de la distribución de etiquetas por muestra.

La Figura 2.2(a) y la Figura 2.2(d) corresponden a dos conjuntos de datos muy parecidos, con el mismo número total de etiquetas y casi el mismo número de instancias, pero en el caso de la Figura 2.2(d) con un valor de *Card* algo más alto. Esto es suficiente para tener muchas muestras con dos o más etiquetas, mientras que en la Figura 2.2(a) queda patente que cada muestras tiene solo una etiqueta con pocas excepciones. La Figura 2.2(b) y la Figura 2.2(e) muestran dos conjuntos con más instancias, etiquetas y un valor de *Card* más alto. La diferencia principal entre ellos es la dispersión: en la Figura 2.2(e) tenemos $CV = 0{,}189$ contra $CV = 0{,}705$ en la Figura 2.2(b). El efecto de este factor puede observarse mejor en las Figuras 2.2(c) y 2.2(f), que son una versión aumentada de una porción de las anteriores.

La conclusión que podríamos obtener de este análisis es que el uso combinado de varias métricas, como *Card*, *CV*, *SK* y *KR*, aporta información útil para caracterizar un MLD, conociendo cuál será la dificultad a afrontar a la hora de trabajar con él. En general, un valor de *Card* muy bajo (cercano a 1.0) indicará que la mayoría de las muestras tienen asociada solo una etiqueta, por lo que será prácticamente imposible extraer información de dependencia entre ellas. Para valores de *Card* por encima de 1.0, la presencia de un alto *CV*, *SK* o *KR* caracteriza un MLD difícil de tratar.

2.7. Experimentación y validación

LI-MLC, el algoritmo propuesto aquí para reducir la dimensionalidad en el espacio de etiquetas, ha sido probado usando 16 MLD y siete algoritmos de clasificación multietiqueta. La minería de reglas de asociación se ha llevado a cabo con el algoritmo FP-Growth, antes descrito, que precisa dos parámetros de entrada: los valores de umbral para *Conv* y *Supp*. El umbral de *Conv* se ha fijado a un valor mínimo de 1.25, tal y como se recomienda en Brin et al. [1997]. A fin de obtener suficientes reglas de asociación, el umbral para *Supp* se ha fijado en 0.025, bajo la hipótesis de que incluso aquellas asociaciones entre etiquetas no muy frecuentes pero fuertes (con alto *Conv*) pueden resultar de utilidad.

En los apartados de esta sección se enumeran los MLD y algoritmos usados en la experimentación, se detallan los datos obtenidos y se describe el análisis efectuado sobre ellos.

2.7.1. Conjuntos de datos

La Tabla 2.1 muestra los MLD usados en la experimentación y sus características básicas: número de instancias, de atributos y de etiquetas y la métrica *Card*. Siete de ellos (`bibtex`, `bookmarks` (Katakis et al. [2008]), `delicious` (Tsoumakas et al. [2008]), `enron` (Klimt and Yang [2004]), `llog`, `medical` (Crammer

et al. [2007]) y `slashdot`) provienen del dominio de la clasificación de textos, tres (`corel5k` (Duygulu et al. [2002]), `corel16k` (Barnard et al. [2003]) y `scene` (Boutell et al. [2004])) son del dominio de etiquetado de imágenes, dos (`cal500` (Turnbull et al. [2008]) y `emotions` (Wieczorkowska et al. [2006])) provienen del dominio de etiquetado de música, dos más (`genbase` (Diplaris et al. [2005]) y `yeast` (Elisseeff and Weston [2001])) son del dominio de la biología, y los últimos dos (`imdb` y `mediamill` (Snoek et al. [2006])) son del dominio de etiquetado de vídeos/películas. El número de etiquetas de estos MLD está en el rango [6, 983] y su *Card* está en el rango [1,074, 26,044]. Todos estos conjuntos de datos pueden encontrarse en los repositorios de MULAN Tsoumakas et al. y MEKA Read and Reutemann.

Tabla 2.1: Características básicas de los conjuntos de datos.

MLD	**#Instancias**	**#Atributos**	**#Etiquetas**	**Card**
bibtex	7395	1836	159	2.402
bookmarks	87856	2150	208	2.028
cal500	502	68	174	26.044
corel5k	5000	499	374	3.522
corel16k[a]	13766	500	161	2.867
delicious	16105	500	983	19.02
emotions	593	72	6	1.868
enron	1702	753	53	3.378
genbase	662	1186	27	1.252
imdb[b]	12000	1001	28	1.905
llog	1460	1004	75	1.180
mediamill	43907	120	101	4.376
medical	978	1449	45	1.245
scene	2407	294	6	1.074
slashdot	3782	1079	22	1.181
yeast	2417	198	14	4.237

[a] Corel16k es un MLD con 10 subconjuntos. En la tabla se muestran los valores promedio.

[b] Tomando un 10 % aleatorio de las 120 000 instancias del MLD original.

Sobre estos conjuntos de datos se ha aplicado una validación cruzada con la configuración habitual usando 10 particiones. La misma configuración se ha usado para ejecutar cada uno de los algoritmos de clasificación sin aplicar LI-MLC.

2.7.2. Algoritmos de clasificación

Para probar el funcionamiento de LI-MLC se ha hecho una selección de algoritmos de clasificación multietiqueta con representación de métodos basados en transformación, adaptación de algoritmos y *ensembles*. Mencionar que los métodos basados en técnicas de transformación hacen su trabajo tras la fase de preprocesamiento de LI-MLC descrita anteriormente, operando ya sobre el conjunto de datos reducido del que se habrán eliminado las etiquetas que pueden inferirse a partir de las reglas de asociación.

Los métodos elegidos, todos ellos usados con sus parámetros por defecto, son los siguientes: BR Godbole and Sarawagi [2004], LP Boutell et al. [2004], CLR Fürnkranz et al. [2008], CC Read et al. [2011], IBLR-ML Cheng and Hüllermeier [2009], RAkEL Tsoumakas and Vlahavas [2007] y HOMER Tsoumakas et al. [2008]. C4.5 ha sido utilizado como clasificador binario y multiclase subyacente allí donde es necesario. Para IBLR-ML el número de vecinos se ha fijado a 10. Para HOMER, el número de grupos (*clusters*) se ha fijado al mínimo entre 4 y el número de etiquetas existentes en el MLD.

2.7.3. Métricas de evaluación de rendimiento

Existe un gran abanico de métricas de evaluación del rendimiento en clasificación multietiqueta (véase la Sección 1.6 del capítulo previo). Una de las más utilizadas es Hamming Loss (*HL*) (véase Ecuación 1.5). Esta métrica tiende rápidamente a cero cuando se utiliza con algunos MLD, concretamente aquellos en los que únicamente un pequeño subconjunto de etiquetas aparece en las muestras de datos pero el conjunto total de etiquetas es grande. En este contexto, usando el mismo clasificador y cometiendo idéntico número de errores en clasificación, el

rendimiento parecerá cada vez peor a medida que el número de etiquetas total sea menor. No obstante el rendimiento real del clasificador sería el mismo.

A pesar de que *HL* es una métrica popular, el anterior razonamiento demanda su uso conjunto con otras medidas más fuertes. En Zhou et al. [2012] se realiza una evaluación de la forzaleza de diferentes criterios de evaluación de clasificadores multietiqueta, encontrándose que *HL* es la más débil y menos representativa, mientras que *F-measure* es una de las más fuertes. Por esta razón se ha elegido aquí utilizar *F-measure* (véase la Ecuación 1.4) como segunda métrica de evaluación del rendimiento.

2.7.4. Tests estadísticos

A fin de analizar si las diferencias entre los resultados obtenidos son significativas o no, es habitual utilizar tests estadísticos. En este trabajo el interés está en comparar los valores de las métricas de rendimiento en clasificación sobre varios MLD en pares: un conjunto de resultados base generados por los clasificadores multietiqueta y un segundo grupo con esos mismos clasificadores usando la metodología LI-MLC. Para esta tarea es posible utilizar el T-test paramétrico o el test de Wilcoxon no paramétrico.

En Demšar [2006] y Luengo et al. [2009] se muestra que los experimentos de clasificación que siguen un esquema de validación cruzada con 10 particiones (10-fcv) no cumplen los requisitos necesarios de normalidad, heterocedasticidad e independencia indispensables para aplicar tests de tipo paramétrico. Por esta razón, para realizar las comparaciones antes indicadas, hemos utilizado el test de Wilcoxon (Sheskin [2003]) de tipo no paramétrico.

Para cada algoritmo, tomando las métricas de rendimiento de la versión en la que se usa LI-MLC como referencia, se ha aplicado este test y se han obtenido los *p-value* exactos. El mismo método se ha usado también para comparar los tiempos de entrenamiento de cada clasificador. Los tests se han ejecutado utilizando el módulo estadístico del software KEEL Alcala-Fdez et al. [2011].

2.7.5. Caracterización de los conjuntos de datos

Aplicando las medidas propuestas en la Sección 2.6.2 se ha procedido a analizar la dimensionalidad en el espacio de etiquetas de los conjuntos de datos usados en la experimentación. La Tabla 2.2 muestra las medidas para *Card*, *Dens*, *CV*, *SK* y *KR*, así como el número medio de reglas obtenido para cada MLD.

Tabla 2.2: Medidas de caracterización.

MLD	Card	Dens	CV	SK	KR	#Reglas
bibtex	2.402	0.015	0.704	2.353	13.384	0.0
bookmarks	2.028	0.010	0.904	4.197	36.084	0.0
cal500	26.044	0.150	0.221	0.518	0.398	6.4
corel5k	3.522	0.009	0.189	-1.129	0.703	5.0
corel16k	2.867	0.018	0.316	-0.344	-0.677	4.0
delicious	19.020	0.019	0.267	-1.206	0.988	8.0
emotions	1.868	0.311	0.360	0.162	-0.802	3.8
enron	3.378	0.064	0.454	0.652	1.062	13.8
genbase	1.252	0.046	0.555	3.501	14.597	5.7
imdb	1.905	0.068	0.630	1.669	3.492	3.0
llog	1.180	0.016	0.679	1.159	2.503	0.0
mediamill	4.376	0.043	0.533	0.440	0.291	8.0
medical	1.245	0.028	0.371	1.612	1.581	0.0
scene	1.074	0.179	0.245	3.326	9.411	0.0
slashdot	1.181	0.054	0.351	2.147	3.836	0.0
yeast	4.237	0.303	0.371	0.380	0.150	10.8

Hay seis casos en los que no se han obtenido reglas a partir de los datos: 1) `bibtex`, 2) `bookmarks`, 3) `llog`, 4) `medical`, 5) `scene` y 6) `slashdot`. Cuatro de ellos (`llog`, `medical`, `scene` y `slashdot`) tienen un valor de *Card* extremadamente bajo, en el rango [1,074, 1,245] lo que ya explica por sí mismo la imposibilidad de obtener regla alguna. `bibtex` y `bookmarks` tienen valores de *Card* ligeramente superiores, 2.402 y 2.028, pero comparten valores relativamente altos para *CV*, *SK* y *KR*. Estos denotan que solo unas pocas instancias contienen dos o más etiquetas, mientras que el resto solo tienen una. En contraposición, conjuntos

de datos como `emotions`, con un valor para *Card* de 1.868 pero también bajos valores en las demás métricas, han permitido la extracción de algunas reglas.

El análisis previo facilita una guía general de uso, pero también hay excepciones. Por ejemplo, el valor de *Card* para `medical` es similar al de `genbase` y el *CV* de este último es mayor, indicando una mayor dispersión, pero en el primer caso no se obtienen reglas mientras que en el segundo sí. En este caso hay que considerar que `genbase` tiene casi la mitad de etiquetas (en número total) que `medical`. A pesar de que tengan un valor parecido para *Card*, el número de combinaciones de etiquetas posibles es mucho mayor para `medical` (2^{45}) que para `genbase` (2^{27}).

2.7.6. Resultados de clasificación

Las Tablas 2.3 y 2.4 muestran los valores obtenidos para *HL* y *F-measure* para cada combinación de MLD-algoritmo en dos versiones: el clasificador multietiqueta original y la misma configuración aplicando la metodología propuesta, designada como +**LI-MLC**. Los seis MLD para los que no ha sido posible obtener reglas no aparecen en esta tabla, dado que LI-MLC no es aplicable en dichos casos.

La evaluación de los resultados usando *HL* refleja mejoras en un tercio de los casos. Hay varios empates, así como muchos casos en los que la diferencia en una dirección u otra es mínima, en el rango de unas pocas diezmilésimas. Por ejemplo, con `delicious` LI-MLC pierde en todos los casos pero todas las diferencias están en el rango [0.0001, 0.0004]. La importancia de estas diferencias es dudosa pero, en cualquier caso, la mejora obtenida en tiempo de ejecución es muy significativa. La debilidad de *HL* como métrica de evaluación del rendimiento fueron expuestas anteriormente.

Tabla 2.3: Rendimiento en clasificación con Hamming loss (menor es mejor)

MLD	CC	+LI-MLC	BR	+LI-MLC	CLR	+LI-MLC	HOMER	+LI-MLC	IBLR	+LI-MLC	LP	+LI-MLC	RAkEL	+LI-MLC
cal500	**0.1760**	**0.1760**	**0.1615**	0.1620	**0.1385**	0.1401	**0.1846**	0.1893	**0.2309**	0.2334	**0.1996**	0.2030	0.1615	**0.1611**
corel16k	0.0206	**0.0199**	0.0197	**0.0194**	0.0189	**0.0188**	0.0253	**0.0251**	**0.0191**	**0.0191**	0.0321	**0.0306**	0.0197	**0.0194**
corel5k	0.0099	**0.0098**	0.0098	**0.0097**	0.0095	**0.0094**	0.0132	**0.0128**	**0.0224**	0.0226	0.0168	**0.0162**	0.0098	**0.0097**
delicious	**0.0188**	0.0189	**0.0186**	0.0188	**0.0184**	0.0185	**0.0238**	0.0241	**0.0507**	0.0511	**0.0300**	**0.0300**	**0.0187**	**0.0187**
emotions	**0.2550**	0.2766	**0.2474**	0.2857	**0.2423**	0.3098	**0.2609**	0.2940	**0.1883**	0.2417	**0.2777**	0.2982	**0.2474**	0.2885
enron	**0.0524**	0.0546	**0.0508**	0.0553	**0.0471**	0.0525	**0.0584**	0.0623	**0.0564**	0.0593	0.0717	**0.0702**	**0.0508**	0.0551
genbase	**0.0011**	0.0055	**0.0011**	0.0058	**0.0013**	0.0059	0.0013	**0.0006**	**0.0029**	0.0078	**0.0019**	0.0061	**0.0011**	0.0058
imdb	0.0878	**0.0745**	0.0753	**0.0724**	0.0719	**0.0701**	0.0950	**0.0927**	0.0681	**0.0680**	0.1024	**0.0987**	0.0753	**0.0728**
mediamill	**0.0357**	0.0389	**0.0335**	0.0397	**0.0283**	0.0358	**0.0371**	0.0437	**0.0283**	0.0356	**0.0423**	0.0474	**0.0335**	0.0398
yeast	**0.2682**	0.2795	**0.2454**	0.2720	**0.2202**	0.2590	**0.2555**	0.2783	**0.1934**	0.2397	**0.2779**	0.2939	**0.2449**	0.2731

Tabla 2.4: Rendimiento en clasificación con F-Measure (mayor es mejor)

MLD	CC	+LI-MLC	BR	+LI-MLC	CLR	+LI-MLC	HOMER	+LI-MLC	IBLR	+LI-MLC	LP	+LI-MLC	RAkEL	+LI-MLC
cal500	0.3626	**0.3627**	**0.3375**	0.3364	**0.2888**	0.2644	**0.3972**	0.3942	**0.3194**	0.3065	**0.3287**	0.3111	0.3375	**0.3393**
corel16k	0.5388	**0.5578**	0.5259	**0.5702**	0.5313	**0.5606**	0.4517	**0.4606**	0.5539	**0.6032**	0.4548	**0.4932**	0.5243	**0.5624**
corel5k	0.4859	**0.5125**	0.4697	**0.4884**	0.4511	**0.4812**	0.3894	**0.4013**	0.2876	**0.2889**	0.4031	**0.4305**	0.4697	**0.4921**
delicious	0.3290	**0.3421**	0.3097	**0.3198**	0.2549	**0.2763**	**0.3289**	0.3249	**0.1858**	0.1843	0.2599	**0.2602**	0.3056	**0.3176**
emotions	**0.7438**	0.7309	0.7086	**0.7234**	**0.7186**	0.6903	0.6949	**0.7123**	**0.7839**	0.7690	0.7244	**0.7291**	0.7086	**0.7283**
enron	**0.6268**	0.5977	0.6137	**0.6260**	0.6330	**0.6469**	0.5993	**0.6104**	0.5810	**0.6038**	0.5561	**0.5664**	0.6137	**0.6323**
genbase	**0.9919**	**0.9919**	**0.9919**	0.9915	**0.9914**	0.9905	**0.9917**	**0.9917**	0.9856	**0.9870**	**0.9919**	0.9915	**0.9919**	0.9918
imdb	0.7209	**0.7413**	0.6602	**0.7077**	0.6750	**0.7340**	0.5634	**0.5821**	0.7110	**0.7358**	0.6375	**0.6433**	0.6602	**0.6892**
mediamill	**0.6157**	0.5711	**0.6066**	0.5339	**0.6387**	0.5592	**0.5961**	0.5150	**0.6544**	0.5794	**0.1314**	0.0837	**0.6066**	0.5331
yeast	**0.6389**	0.6339	**0.6166**	0.5890	**0.6587**	0.5901	**0.6156**	0.5901	**0.6987**	0.6279	**0.6256**	0.6089	**0.6171**	0.5890

Tabla 2.5: Rendimiento en términos de tiempo de entrenamiento en segundos (menor es mejor)

MLD	CC	+LI-MLC	BR	+LI-MLC	CLR	+LI-MLC	HOMER	+LI-MLC	IBLR	+LI-MLC	LP	+LI-MLC	RAkEL	+LI-MLC
cal500	**148.68**	152.16	63.29	**54.79**	1257.52	**1154.58**	99.66	**93.16**	261.68	**233.29**	14.92	**15.09**	346.04	**328.47**
corel16k	19642.56	**19022.21**	39980.52	**26154.08**	73505.36	**62752.85**	12139.59	**11088.42**	20226.75	**20108.44**	1351.50	**1240.07**	201454.89	**152089.85**
corel5k	**3915.71**	3954.94	5053.38	**5013.73**	18411.18	**14369.86**	3934.75	**3575.49**	20742.53	**19817.02**	**336.88**	340.73	43754.60	**39703.89**
delicious	375022.66	**358317.20**	256064.79	**236233.64**	248904.86	**212829.72**	408823.09	**408405.82**	176880.45	**126634.57**	3538.60	**3531.68**	1457888.84	**1438587.70**
emotions	5.61	**2.93**	12.01	**3.16**	10.83	**4.06**	6.14	**3.07**	4.31	**3.65**	3.92	**2.64**	24.98	**2.52**
enron	**1512.23**	1623.53	2051.74	**1692.63**	4780.69	**3290.02**	2688.95	**1879.37**	143.97	**116.68**	**115.50**	125.39	13070.84	**10607.92**
genbase	7.68	**5.96**	17.07	**10.69**	26.31	**19.24**	12.00	**8.62**	60.24	**60.20**	4.67	**4.51**	63.93	**51.67**
imdb	**85817.48**	89018.88	**97036.83**	97962.34	158132.40	**141208.82**	37421.90	**34063.70**	**549.09**	558.92	**7967.42**	8416.61	**583141.02**	607884.85
mediamill	51554.52	**39843.20**	28622.91	**26448.25**	91883.01	**68983.46**	9129.91	**6910.66**	**21242.97**	21343.17	22968.71	**19839.37**	116597.17	**112656.82**
yeast	90.35	**78.20**	121.14	**86.01**	336.83	**202.17**	141.80	**100.01**	94.84	**94.25**	105.98	**81.13**	791.22	**588.73**

El uso de *F-measure* para evaluar los resultados ofrece una visión muy distinta. En general las diferencias son mucho mayores, en la escala de las centésimas. La aplicación de LI-MLC genera mejores resultados en 41 de 70 configuraciones. El rendimiento en clasificación se mejora casi siempre para `corel16k`, `corel5k`, `enron` e `imdb`. En contraposición nunca se mejora con dos de los MLD: 1) `mediamill` y 2) `yeast`. Para el resto de los MLD los resultados son intermedios. Excepto para mediamill y yeast, LI-MLC tiene una influencia fundamentalmente positiva en el rendimiento de los clasificadores subyacentes. En el siguiente apartado se evaluará si estas diferencias tienen valor estadístico.

La Tabla 2.5 tiene la misma estructura que las anteriores, pero muestra los tiempos de ejecución de cada configuración medidos en segundos. Este es el tiempo empleado por el entrenamiento del clasificador con el algoritmo base original y con LI-MLC. En este último caso también se ha incluido el tiempo que se tarda en obtener las reglas de asociación.

Como puede observarse, para algunas combinaciones de algoritmo y MLD, el tiempo de entrenamiento se ha reducido a una fracción del original. Los ahorros en algunos casos están en el orden de varias horas. Tomando como referencia el tiempo de los algoritmos base, la Figura 2.3 muestra la ganancia relativa en tiempo de ejecución tras aplicar LI-MLC. Hay que destacar que la mejora en tiempos se consigue, en general, sin dañar el rendimiento en clasificación o incluso mejorándolo en muchos casos. Por ejemplo, el uso de LI-MLC con BR y RAkEL resulta en una reducción del 22 % y el 15 % del tiempo de ejecución, mientras que el rendimiento indicado por *F-measure* para estas configuraciones es mejor en seis de los 10 casos, respectivamente.

2.7.7. Estudio estadístico

A partir de los datos mostrados en las tablas previas es posible obtener una visión general de las mejoras alcanzadas. Para saber si esta progresión es realmente significativa desde un punto de vista estadístico, en la Tabla 2.6 se muestran los

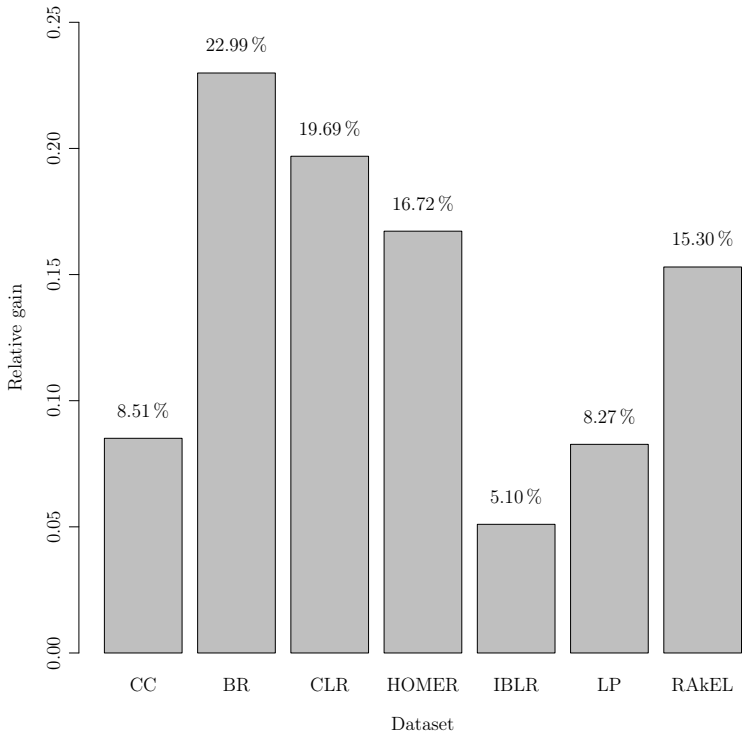


Figura 2.3: Ganancia relativa en tiempo de LI-MLC respecto al clasificador base

p-value obtenidos del test de Wilcoxon tras aplicarlos a los resultados medidos con *HL*, *F-measure* y tiempo de entrenamiento.

Tabla 2.6: p-values exactos devueltos por el test de Wilcoxon

Algoritmo	**HL**	**F-Measure**	**Tiempo ejecución**
BR	≥ 0.2	0.00714	7.25E-5
CC	≥ 0.2	0.09874	0.16880
CLR	≥ 0.2	0.06628	3.81E-6
HOMER	≥ 0.2	0.03850	7.25E-5
IBLR-ML	≥ 0.2	0.05594	0.00202
LP	≥ 0.2	0.01236	0.01597
RAkEL	≥ 0.2	0.00533	6.45E-4

Como puede observarse, la significación estadística de las mejoras de rendimiento en clasificación dependerán de la medida que se escoja:

- Con la métrica *HL* todos los *p-value* están por encima de 0.2, lo que significaría que LI-MLC no aporta beneficio significativo desde un punto de vista estadístico.
- Con la métrica *F-measure*, para CC, CLR e IBLR-ML tenemos que el *p-value* $< 0{,}1$ y para BR, HOMER, LP y RAkEL que el *p-value* $< 0{,}05$. Por tanto, LI-MLC ofrece una mejora significativa estadísticamente con un 90 % y un 95 % de confianza, respectivamente.

Hay que considerar que la información aportada por estas métricas sobre el rendimiento en clasificación difiere significativamente. *HL* es una métrica muy influenciada por el número de etiquetas existentes en el conjunto de datos, parámetro que afecta al resultado sin que importe el número de errores cometidos por el clasificador. *F-measure*, en cambio, es una media armónica de dos métricas, *Precision* y *Recall*, que evalúan parámetros diferentes del clasificador, aportando una visión global más significativa de su rendimiento.

En cuanto al tiempo de ejecución, todos los *p-value* están por debajo de 0.05 con la excepción de CC. En consecuencia, puede afirmarse que LI-MLC reduce significativamente el tiempo necesario para entrenar los clasificadores, con independencia de las mejoras en clasificación.

Resumiendo, LI-MLC ofrece una reducción en el tiempo de ejecución estadísticamente significativa, así como mejoras significativas (con *p-values* muy bajos) en los resultados de clasificación cuando se evalúan con *F-measure*, una métrica considerada más fuerte que HL.

2.8. Conclusiones

Como bien es sabido, todos los clasificadores tienen un cierto ratio de errores, ocurriendo lo mismo con las reglas de asociación. Las etiquetas muy comunes

(aquellas que tienen un valor alto de *Supp*) tienden a generar un sesgo en los clasificadores, generalmente beneficiando a las clases mayoritarias. Este sesgo se reduce cuando LI-MLC oculta estas etiquetas, obteniendo modelos que clasifican mejor instancias en las que aparecen etiquetas menos comunes. En general, la mejora en los resultados de clasificación inducida por LI-MLC es superior al posible error que pudiera introducir la inferencia de las reglas de asociación.

Para algunos métodos de clasificación, como es el caso de LP, la reducción de una sola etiqueta deja en la mitad el número de combinaciones generadas por la transformación del MLD, lo cual resulta en un conjunto de clases mucho más pequeño y, en consecuencia, más fácil de procesar con clasificadores multiclase. Los métodos de tipo *ensemble* basados en LP, como RAkEL y HOMER, se benefician de esta mejora. Dado que para la mayoría de los MLD se obtiene más de una regla, la reducción en el número de combinaciones es mucho más importante, así como la mejora en el tiempo de ejecución.

Entrenar un clasificador individual para cada etiqueta (transformación BR) existente en el MLD parece una buena idea y, de hecho, funciona bien en muchos casos. No obstante, este enfoque no considera en sus predicciones la dependencia entre etiquetas, una información valiosa como se apunta en Alvares Cherman et al. [2010] y Alvares-Cherman et al. [2012]. Las etiquetas de asociación, por el contrario, recogen esa dependencia en aquellos casos en los que hay una implicación fuerte tal y como se explicó antes, prediciendo la presencia de una etiqueta cuando esta depende más de la información de correlación que de la existente entre los atributos de entrada y la propia etiqueta. Muchos algoritmos de clasificación multietiqueta utilizan BR como transformación base, por lo que la mejora aportada por LI-MLC afecta también a sus resultados.

En cuanto al tiempo usado por el algoritmo para construir los modelos, la reducción en el número de etiquetas efectuada por LI-MLC tiene un efecto positivo prácticamente en todos los casos. Los clasificadores se benefician de un conjunto reducido de etiquetas y el tiempo que se tarda en aplicar LI-MLC es muy inferior al que se ahorra construyendo el modelo simplificado. La Tabla 2.5 muestra

que todos los *p-values* están muy por debajo de 0.05 (con la excepción de CC), concluyéndose que hay una diferencia estadísticamente significativa.

Con métodos multietiqueta que emplean clasificadores binarios para cada etiqueta, como es el caso de BR y muchas propuestas de *ensembles*, LI-MLC aporta una mejora en el tiempo de ejecución que es al menos lineal respecto al número de etiquetas eliminadas. Al trabajar con algoritmos basados en la combinación de etiquetas, la mejora podría ser mucho más importante. En general, los métodos de tipo multiclasificador serán los que más se beneficien de la reducción de dimensionalidad del espacio de etiquetas, ya que esto afecta a todos los modelos individuales que forman el *ensemble*.

Además también se ha demostrado cómo es posible utilizar ciertas métricas estadísticas, como *CV*, *SK* y *KR* que, a diferencia de *Card* y *Dens* por sí solas, permiten obtener información útil para saber cuándo es apropiado aplicar la metodología descrita. Sin pérdida de generalidad, estas métricas pueden también ser usadas para caracterizar MLD a fin de obtener una visión general de la distribución de las etiquetas en el conjunto de datos.

2.9. Publicaciones asociadas a este trabajo

Un estudio preliminar sobre la aplicación de reglas de asociación a fin de reducir el espacio de etiquetas fue presentado en HAIS'12 Charte et al. [2012], *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "Improving Multilabel Classifiers via Label Reduction with Association Rules". Volume 7209 of* Lecture Notes in Computer Science*, chapter 18, pages 188–199. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-28930-9. doi: 10.1007/978-3-642-28931-6_18.*

El algoritmo LI-MLC en su implementación final, junto con la experimentación descrita, ha sido publicado en la revista IEEE TNNLS (Charte et al. [2014a]), *F. Charte, A. Rivera, M. J. del Jesus, and F. Herrera. "LI-MLC: A label inference methodology for addressing high dimensionality in the label space for multilabel*

classification. Neural Networks and Learning Systems, IEEE Transactions on, 25 (10): 1842–1854, Oct 2014. ISSN 2162-237X. doi: 10.1109/TNNLS.2013.2296501.

Capítulo 3

Desbalanceo en conjuntos de datos multietiqueta

En este capítulo se introduce el problema del aprendizaje con conjuntos de datos desbalanceados, incluyendo las pecualiaridades en el caso multietiqueta. A continuación se enumeran algunos de los métodos publicados en la bibliografía especializada para tratar este problema en clasificación multietiqueta. Después se presentan las propuestas propias. Por una parte se definen las métricas propuestas para medir el desbalanceo en conjuntos de datos multietiqueta. Por otra, se presentan cuatro algoritmos de remuestreo basados en técnicas aleatorias cuyo objetivo es reequilibrar la distribución de las etiquetas, a fin de contribuir a mejorar el rendimiento en clasificación de los algoritmos. El siguiente paso es describir la experimentación llevada a cabo para validar el funcionamiento de estas propuestas. En la parte final se facilitan las conclusiones obtenidas a partir de este trabajo.

3.1. Introducción

En el apartado 1.1.2 del Capítulo 1 se mencionaba que un algoritmo de preprocesamiento podía servir para reequilibrar la distribución de los datos mediante técnicas de eliminación o de generación de instancias. La aplicación de estas técnicas lógicamente debería estar supeditada a la existencia de un problema previo en los datos, conocido como desbalanceo (*imbalance*) o desequilibrio en la representación de las clases en los conjuntos de datos.

Al hablar genéricamente de desbalanceo nos estamos refiriendo a que las etiquetas de clase asignadas a cada una de las instancias del conjunto de datos no tienen un representación homogénea. Este es un problema profundamente estudiado en clasificación binaria y multiclase (Japkowicz and Stephen [2002], Barandela et al. [2003], Chawla et al. [2004], Sun et al. [2009] Fernández et al. [2013]). Para conocer hasta qué punto un conjunto de datos está desbalanceado se utiliza una medida llamada *ratio de desbalanceo* (IR, *Imbalance Ratio*) introducida en Japkowicz and Stephen [2002]. Tradicionalmente la clasificación con datos desbalanceados se ha abordado mediante técnicas (López et al. [2013]) como el remuestreo, el aprendizaje sensible al coste y las adaptaciones específicas para ciertos algoritmos de aprendizaje.

En el caso multietiqueta es algo comúnmente aceptado que la mayoría de los conjuntos de datos presentan este problema (Tahir et al. [2012a]). Sin embargo, no existen medidas específicas para medir el nivel de desbalanceo de los MLD. En consecuencia la naturaleza desbalanceada de los mismos es algo que se intuye, pero sin una cuantificación numérica del problema.

Hasta la fecha se han publicado algunas propuestas relativas al tratamiento datos multietiqueta con desbalaneo, sobre todo centradas en adaptaciones algorítmicas como las propuestas en Tahir et al. [2012a], Tahir et al. [2012b] y He et al. [2012]. Se trata de soluciones dependientes de un clasificador concreto. Una vía alternativa para el tratamiento del desbalanceo sería la búsqueda de soluciones independientes del clasificador, mediante técnicas de preprocesamiento como los

algoritmos de remuestreo. Este enfoque permitiría el uso de dicha solución con cualquier algoritmo de clasificación multietiqueta de última generación.

Por tanto existe la necesidad de diseñar medidas específicas para medir el nivel de desbalanceo de conjuntos de datos multietiqueta, así como de crear algoritmos de preprocesamiento capaces de reequilibrar la distribución de las etiquetas permitiendo el uso de cualquier clasificador con posterioridad. Esos son los dos fines principales del trabajo al que se dedica el presente capítulo, en el que se hacen las siguientes propuestas:

- Se proponen tres métricas a fin de facilitar la evaluación del grado de desbalanceo presente en conjuntos de datos multietiqueta. Estas medidas se utilizan a continuación para determinar el desbalanceo en varios MLD.
- Se presentan dos algoritmos de remuestreo basados en la transformación LP, evaluando la frecuencia de combinaciones de etiquetas. Uno de ellos lleva a cabo un sobremuestreo u *oversampling* (LP-ROS) de los datos y el otro un bajomuestreo o *undersampling* (LP-RUS).
- Se presentan dos algoritmos de remuestreo que evalúan la frecuencia individual de cada etiqueta, en lugar de combinaciones completas, aislando las instancias en las que aparecen una o más etiquetas minoritarias sirviéndose para ello de las métricas introducidas anteriormente. Uno de los algoritmos es de sobremuestreo (ML-ROS) y el otro de bajomuestreo (ML-RUS).

´La utilidad de estas propuestas será probada experimentalmente y los resultados analizados estadísticamente, proponiéndose también una guía básica de aplicación de los algoritmos citados.

3.2. Desbalanceo en clasificación tradicional

En general, la mayoría de los clasificadores ven reducido su rendimiento cuando el conjunto de datos sobre el que operan presenta desbalanceo. Como se indica

en Japkowicz and Stephen [2002] la razón estriba en su diseño, enfocado a reducir el ratio de error global. Este es un enfoque que tiende a beneficiar a las clase más representada en el conjunto de datos (clase mayoritaria), etiquetando las nuevas instancias con esta clase a expensas de la clase minoritaria. Además una distribución desbalanceada de las clases puede complicar otros problemas habituales, como la presencia de etiquetas con ruido (Khoshgoftaar et al. [2010]).

Según se indica en López et al. [2013], el problema del desbalanceo ha sido afrontado fundamentalmente a través de tres vías distintas:

- **Remuestreo**: Las técnicas de remuestreo de datos siguen el enfoque del preprocesamiento, reequilibrando la distribución de las clases mediante la eliminación (Kotsiantis and Pintelas [2003]) de instancias existentes o la generación (Nitesh et al. [2002]) de nuevas instancias. Las técnicas de remuestreo son soluciones independientes del clasificador para el problema de aprendizaje con datos desbalanceados, si bien también existen propuestas para clasificadores específicos (Lin et al. [2013]), y han demostrado su efectividad como se constata en García et al. [2012].

- **Adaptación de algoritmos**: Se trata de una técnica (Fernández et al. [2013]) dependiente del clasificador, siendo su objetivo modificar algoritmos de clasificación existentes teniendo en cuenta la naturaleza desbalanceada de los conjuntos de datos.

- **Clasificación sensible al coste**: También es una técnica (Provost and Fawcett [2001]) dependiente del algoritmo, al combinar los dos enfoques anteriores mediante la adaptación del clasificador y la aplicación de algún tipo de preprocesamiento a los datos.

El estudio que se presenta aquí se centra en el primero de los enfoques, basado en técnicas de remuestreo de datos. En He and Ma [2013] puede encontrarse una introducción general y detalles adicionales sobre cada una de las opciones mencionadas. En ocasiones es posible utilizar técnicas de preprocesamiento conjuntamente con *ensembles* a fin de afrontar el problema del desbalanceo. En Galar

et al. [2011] se facilita una revisión general de los métodos de *ensembles* existentes. Sobre el uso de estos para afrontar el problema de la clasificación desbalanceada, puede encontrarse una completa revisión en Galar et al. [2012].

La mayor parte de los algoritmos de remuestreo consideran la existencia de una sola clase mayoritaria y una sola clase minoritaria. En consecuencia, las técnicas basadas en *undersampling* eliminan instancias solo de la clase más frecuente, mientras que las de *oversampling* crean instancias solo con la clase menos frecuente. Las técnicas más habituales son el *undersampling* aleatorio (Kotsiantis and Pintelas [2003]) o RUS (*Random Undersampling*), el *oversampling* aleatorio o ROS (*Random Oversampling*), y el *oversampling* heurístico, técnica en la que destaca el conocido algoritmo SMOTE (Nitesh et al. [2002]).

3.3. Desbalanceo en clasificación multietiqueta

La mayor parte de los MLD cuentan con un número total de etiquetas bastante alto, en el rango de varias decenas a algunos cientos en los casos más usuales. También hay algunos casos extremos, con ciertos MLD con menos de diez etiquetas y otros en los que aparecen más de mil.

A pesar de contar con un gran número de etiquetas distintas, en la mayoría de los MLD cada instancia está asociada solo a un pequeño subconjunto de ellas. Este hecho se refleja en la medida *Card*. Como puede verse en la Tabla 3.1, el número medio de etiquetas por instancias (*Card*) está siempre por debajo de 5 salvo en el caso de `cal500`. Intuitivamente puede deducirse que algunas de las etiquetas aparecen en muchas muestras del conjunto de datos, mientras que otras estarán apenas representadas. En general, cuanto mayor sea el número de etiquetas distintas en un MLD tanto mayor será la probabilidad de que algunas de ellas estén infra-representadas o sobre-representadas.

La Figura 3.1 es una representación del porcentaje de muestras en las que aparecen las 40 etiquetas más comunes (o todas las etiquetas, para aquellos MLD que tienen menos de 40) de una serie de conjuntos de datos. Puede observarse

Tabla 3.1: Cardinalidad de algunos MLD.

MLD	Etiquetas	Card
bibtex	159	2.402
cal500	174	26.044
corel5k	374	3.522
corel16k	161	2.867
emotions	6	1.868
enron	53	3.378
genbase	27	1.252
llog	75	1.180
mediamill	101	4.376
scene	6	1.074
slashdot	22	1.181
tmc2007	22	2.158
yeast	14	4.237

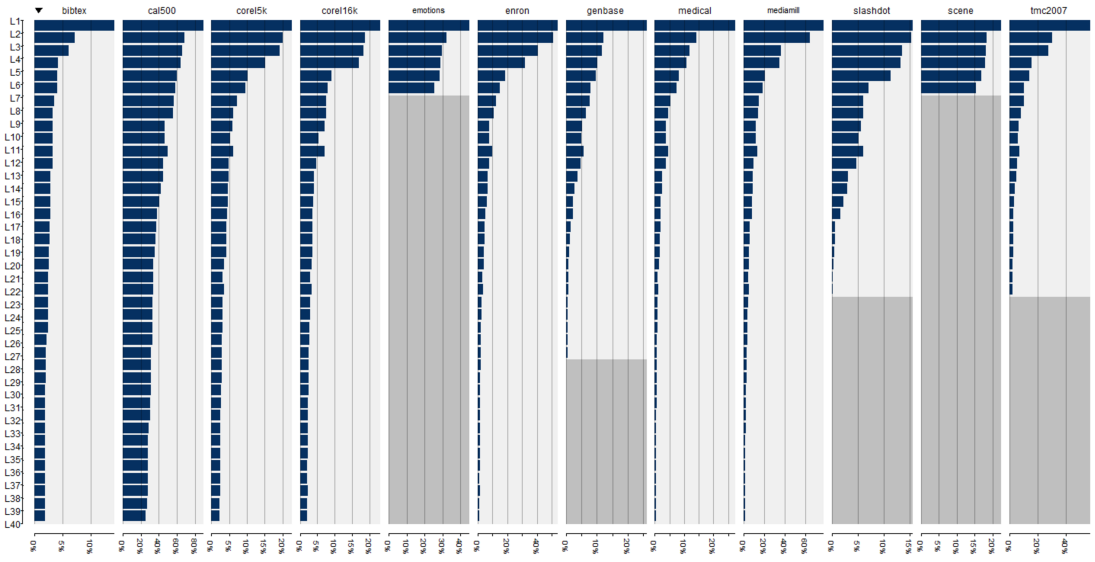
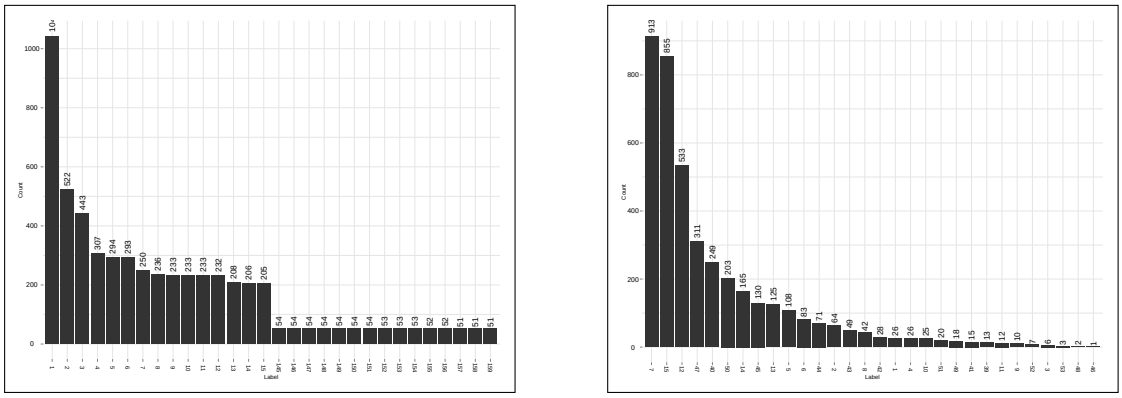


Figura 3.1: Porcentaje de muestras en las que aparece cada etiqueta.

que la mayoría de ellos tienen una o dos etiquetas muy frecuentes, estando el resto mucho menos representadas. En la Figura 3.2 se destacan las características de dos de esos conjuntos de datos, mostrando la diferencia existente entre las 15



(a) bibtex. $Card = 2{,}402$, $Dens = 0{,}015$ (b) enron. $Card = 3{,}378$, $Dens = 0{,}064$

Figura 3.2: Número de muestras en las que aparecen las 15 etiquetas más comunes (parte izquierda de cada gráfica) y las cinco menos comunes (derecha).

etiquetas más comunes y las 15 más raras para cada uno de ellos. Si comparamos la barra situada más próxima a la izquierda de cada gráfica con la que está más a la derecha estaremos visualizando el mayor ratio de desbalanceo existente para cada uno de estos dos MLD. Globalmente puede apreciarse que la mayoría de las etiquetas apenas están presentes en un 5 % de las instancias. Ha de tenerse en cuenta que muchos de los MLD representados en la Figura 3.1 tienen más de 40 etiquetas. Las que no aparecen en dicha figura son mucho menos frecuentes, por lo que es fácil inferir el alto grado de desbalanceo presente en algunos de ellos.

Aunque esta apreciación visual ya nos confirma lo que se suponía, la presencia de desbalanceo en muchos de los MLD, es preciso medir de alguna manera esa realidad. Ese es el objetivo de las métricas propuestas en la Sección 3.4, más adelante en este mismo capítulo.

Otro hecho que también queda patente en las mencionadas figuras es el de que en los MLD no hay una sola clase mayoritaria y una sola minoritaria, sino un grupo de etiquetas mayoritarias y un grupo de etiquetas minoritarias. Por tanto, cualquier algoritmo diseñado para abordar este problema debería tener en cuenta esa realidad, considerando como objetivos varias etiquetas en lugar de solo una.

3.3.1. Aprendizaje con MLD desbalanceados

Hasta el momento el número de publicaciones relativas al tratamiento de desbalanceo en clasificación multietiqueta es relativamente escaso, especialmente si se compara con la cantidad de artículos sobre este mismo problema en clasificación tradicional. La mayoría de las propuestas existentes se agrupan en tres enfoques bien conocidos: adaptaciones de algoritmos, métodos basados en *ensembles* y técnicas de remuestreo.

En los apartados siguientes se describen brevemente las propuestas realizadas ya publicadas, atendiendo a las tres categorías citadas.

Propuestas basadas en adaptación de algoritmos de clasificación multietiqueta

Existen múltiples soluciones propuestas para abordar la clasificación multietiqueta con datos desbalanceados introduciendo adaptaciones en algoritmos existentes, entre ellas:

- En He et al. [2012] los autores afrontan un problema multietiqueta con alto desbalanceo: la predicción de localizaciones de proteínas humanas. El método que proponen, basado en modelos probabilísticos no paramétricos, combina el uso de matrices de covarianza para obtener correlaciones entre etiquetas y el uso de coeficientes ponderados asociados a cada etiqueta a fin de corregir el problema del desbalanceo.

- La propuesta hecha en Li and Shi [2013] es una adaptación del MIMLRBF (Zhang and Wang [2009]), un algoritmo de clasificación multi-instancia y multietiqueta basado en redes neuronales con funciones de base radial (RBFN). El algoritmo propuesto optimiza al original para trabajar con conjuntos de datos desbalanceados a través de dos vías. El número de unidades en la capa oculta no es constante, como en MIMLRBF, sino que se calcula teniendo en cuenta el número de muestras por etiqueta. Además los

pesos de las conexiones entre esta capa oculta y la capa de salida se ajustan aplicando sesgos a medida para cada etiqueta.

- También basada en redes neuronales, en Tepvorachai and Papachristou [2008] se propone un método de enriquecimiento iterativo. Los autores inicializan la ANN aplicando un algoritmo de agrupamiento sobre parte de los datos, tras lo cual se hace un remuestreo sobre cada grupo a fin de equilibrarlos en el espacio euclídeo. Una vez inicializada, el proceso de entrenamiento consiste en un método de enriquecimiento que elimina muestras y agrega otras nuevas para mejorar el equilibrio de la red.

- En Chen et al. [2006] se utiliza una red de tipo *Min-Max-Modular* (Lu and Ito [1999]) para dividir la tarea de clasificación en múltiples tareas más pequeñas. Se usan diferentes estrategias para asegurarse de que el nivel de desbalanceo en dichas subtareas sea menor que en la tarea original. La descomposición del problema original se realiza de forma aleatoria, mediante agrupamiento o utilizando PCA (Jolliffe [2005]). Los subproblemas son siempre de clasificación binaria y son procesados mediante un algoritmo SVM.

Todos estos métodos han sido diseñados para ser dependientes de un algoritmo concreto, siendo sus aplicaciones fundamentalmente específicas del dominio descrito en las respectivas publicaciones.

Propuestas basadas en multiclasificadores

Las técnicas de clasificación basadas en sistemas multiclasificadores han demostrado su eficacia en el campo multietiqueta, con algoritmos como RAkEL (Tsoumakas and Vlahavas [2007]), ECC (Read et al. [2008]) y HOMER (Tsoumakas et al. [2008]) entre sus representantes más destacables. Este mismo enfoque ha sido también aplicado para solventar el problema de desbalanceo, con las siguientes propuestas:

- La propuesta hecha en Tahir et al. [2012a] construye un *ensemble* heterogéneo, llamado EML, usando RAkEL (Tsoumakas and Vlahavas [2007]), ECC (Read et al. [2008]), CLR (Fürnkranz et al. [2008]), MLkNN (Zhang and Zhou [2007]) e IBLR (Cheng and Hüllermeier [2009]) como clasificadores multietiqueta subyacentes. Los autores indican que el uso de *ensembles* es positivo a la hora de trabajar con datos desbalanceados. Algunos de los algoritmos usados en el sistema multiclasificador propuesto son ya *ensembles* de clasificadores en sí mismos. A fin de obtener la predicción final, a partir de las generadas por cada clasificador del *ensemble*, los autores prueban distintos procedimientos, ajustando valores de umbrales y pesos de cada clasificador mediante validación cruzada.

- Diseñado originalmente como un método para abordar el desbalanceo en clasificación tradicional, el algoritmo propuesto en Tahir et al. [2012b] también es aplicable en el campo multietiqueta mediante la creación de *ensembles* de clasificadores binarios. La propuesta, llamada BR-IRUS, basa su estrategia en el entrenamiento de múltiples clasificadores para cada etiqueta, utilizando en cada uno todas las muestras con etiquetas minoritarias pero solo una pequeña porción aleatoria de muestras con etiquetas mayoritarias. De esta manera se obtienen varias fronteras que rodean el espacio de las minoritarias.

La principal desventaja de estas propuestas suele ser su eficiencia, ya que demandan el entrenamiento de un gran número de clasificadores. Este número dependerá de la cantidad de etiquetas del MLD a tratar y, como ya se ha indicado, algunos de ellos cuentan con varios cientos de etiquetas. Además también son soluciones dependientes de un algoritmo hasta cierto punto, ya que no hay libertad para elegir entre cualquier algoritmo de clasificación multietiqueta existente.

Propuestas basadas en remuestreo

En contraposición a los métodos descritos en los apartados previos, los mencionados a continuación pertenecen al grupo de algoritmos independientes del

clasificador. Esto significa que no están vinculados a un algoritmo de clasificación multietiqueta concreto, sino que están basados en el preprocesamiento de los MLD y generan nuevas versiones, más equilibradas, de los mismos. En consecuencia, el rango de aplicaciones de estos métodos es más extenso que el de los dependientes de un clasificador. Las siguientes son las propuestas más importantes publicadas hasta el momento:

- La propuesta hecha en Dendamrongvit and Kubat [2010] es un algoritmo de *undersampling* para clasificación de textos. Los autores siguen el enfoque OVA, generando un conjunto de datos binario individual para cada etiqueta. Todas las muestras que pertenecen a una cierta etiqueta son marcadas como positivas, mientras que el resto aparecen como negativas sin que importe las etiquetas que contengan. El número de instancias de la clase mayoritaria, que habitualmente es la negativa, se reduce con el objetivo de disminuir el sesgo de los clasificadores. La solución propuesta podría ser considerada como un método de transformación, dado que la salida está pensada para su procesamiento con algoritmos de clasificación binarios en lugar de multietiqueta.

- En Giraldo-Forero et al. [2013] los autores analizan diferentes estrategias para la aplicación del algoritmo SMOTE original sobre conjuntos de datos multietiqueta. Para ello definen tres formas de seleccionar las instancias semilla, aquellas que serán tomadas como referencia para localizar los vecinos más cercanos. Solo se toma en consideración una etiqueta minoritaria, lo que cambia es el método utilizado para elegir las muestras que pertenecen a dicha etiqueta. El primer método entrega a SMOTE todas las instancias en las que aparece la etiqueta minoritaria, siguiendo un enfoque OVA como en el método de transformación BR. La segunda estrategia selecciona muestras en las que esté presente únicamente la etiqueta minoritaria, lo cual restringe el conjunto de datos a aquellas muestras que contengan una sola etiqueta. En la tercera estrategia las muestras semilla se procesan en lotes, tantos como combinaciones distintas de etiquetas haya con la etiqueta minoritaria

presente. Esta última estrategia, a la que denominan UG, es la que mejores resultados produce, ya que las dos anteriores en realidad degradan el rendimiento en clasificación.

El mayor déficit de estas propuestas es que, en esencia, ignoran la naturaleza multietiqueta de los datos, al tratar con una única etiqueta minoritaria y descartar las relaciones entre etiquetas. En la Sección 3.5 del presente capítulo y en el capítulo siguiente se presentan propuestas en las que dicha naturaleza es la base del diseño de los métodos de remuestreo.

3.4. Medida del desbalanceo en multietiqueta

En clasificación binaria el nivel de desbalanceo se mide tomando en cuenta únicamente dos clases: una es la clase mayoritaria y la otra la minoritaria. Como ya se ha indicado, hay conjuntos de datos multietiqueta que cuentan con cientos de etiquetas, teniendo muchas de ellas una alta o baja presencia. Por esta razón es importante definir el nivel de desbalanceo en multietiqueta considerando no solo dos etiquetas, sino todas ellas.

Teniendo en cuenta este escenario, aquí proponemos las métricas que se describen en los apartados siguientes para evaluar el nivel de desbalanceo en conjuntos de datos multietiqueta.

3.4.1. Ratio de desbalanceo por etiqueta

Siendo D un conjuntos de datos multietiqueta con un conjunto de etiquetas Y, e Y_i el *i-ésimo labelset*, esta métrica se calcula para la etiqueta y como el ratio entre la etiqueta mayoritaria y la etiqueta y, tal y como se muestra en la Ecuación 3.1.

$$IRLbl(y) = \frac{\underset{y'=Y_1}{\overset{Y_{|Y|}}{\operatorname{argmax}}}(\sum_{i=1}^{|D|} h(y', Y_i))}{\sum_{i=1}^{|D|} h(y, Y_i)},\ h(y, Y_i) = \begin{cases} 1 & y \in Y_i \\ 0 & y \notin Y_i \end{cases}. \tag{3.1}$$

El valor de esta métrica será 1 para la etiqueta más frecuente del MLD y un valor mayor para el resto. Cuanto mayor sea el *IRLbl* más alto será el nivel de desbalanceo de la etiqueta considerada.

3.4.2. Ratio de desbalanceo medio

Esta métrica facilitará un valor que representa el nivel promedio de desbalanceo para todo el MLD, obtenido como se muestra en la Ecuación 3.2.

$$MeanIR = \frac{1}{|Y|} \sum_{y=Y_1}^{Y_{|Y|}} (IRLbl(y)). \tag{3.2}$$

Ha de tenerse en cuenta que MLD con diferentes distribuciones de etiquetas pueden producir el mismo valor para esta métrica. Por esta razón, *MeanIR* debería ser utilizada siempre conjuntamente con la siguiente métrica.

3.4.3. Coeficiente de variación del ratio de desbalanceo

El coeficiente de variación de *IRLbl* se calcula como se indica en la Ecuación 3.3. Su valor indicará si todas las etiquetas sufren de un nivel similar de desbalanceo o, por el contrario, hay grandes diferencias entre ellas. Cuanto mayor sea el valor de *CVIR* tanto mayor será esta diferencia.

$$CVIR = \frac{IRLbl\sigma}{MeanIR}, IRLbl\sigma = \sqrt{\sum_{y=Y_1}^{Y_{|Y|}} \frac{(IRLbl(y)\ -\ MeanIR)^2}{|Y| - 1}} \tag{3.3}$$

3.4.4. Análisis del nivel desbalanceo en MLD

Utilizando las medidas que acaban de describirse, la Tabla 3.2 nos permite analizar el nivel de desbalanceo de algunos de los MLD más usados en la literatura. La columna **MaxIR** muestra el *IRLbl* más alto para una etiqueta en cada MLD.

MLD	#Inst.	#Atr.	#Etiq.	Card	Dens	MaxIR	MeanIR	CVIR
bibtex	7395	1836	159	2.402	0.015	20.431	12.498	0.405
cal500	502	68	174	26.044	0.150	88.800	20.578	1.087
corel5k	5000	499	374	3.522	0.009	1120.000	189.568	1.527
corel16k	13766	500	161	2.867	0.018	126.800	34.155	0.809
emotions	593	72	6	1.868	0.311	1.784	1.478	0.180
enron	1702	753	53	3.378	0.064	913.000	73.953	1.960
genbase	662	1186	27	1.252	0.046	171.000	37.315	1.449
llog	1460	1004	75	1.180	0.016	171.000	39.267	1.311
mediamill	43907	120	101	4.376	0.043	1092.548	256.405	1.175
scene	2407	294	6	1.074	0.179	1.464	1.254	0.122
slashdot	3782	1079	22	1.181	0.054	194.667	19.462	2.288
tmc2007	28596	49060	22	2.158	0.098	41.980	17.134	0.814
yeast	2417	198	14	4.237	0.303	53.412	7.197	1.884

Tabla 3.2: Análisis del desbalanceo en algunos MLD.

Los altos valores de *MeanIR* y *CVIR* que presentan `corel5k` y `mediamill` sugieren que estos MLD son los que más desbalanceo presentan y, por tanto, los que posiblemente más se beneficiarían de un método de remuestreo. Las medidas de varios otros MLD también reflejan distintos niveles de desequilibrio. En contraposición, los valores asociados a `emotions` y `scene` denotan su naturaleza de MLD equilibrados, a pesar del hecho de que tienen dos de los valores más altos para la métrica *Dens*.

Como puede apreciarse, las diferencias que se reflejaban en la gráfica de la Figura 3.1 y se destacaban en la Figura 3.2, entre `bibtex` y `enron`, quedan constatadas en sus valores para *MeanIR* y *CVIR*. Las diferencias extremas entre la frecuencia de aparición de las etiquetas más y menos representadas de `enron` las denota la métrica *MaxIR=913*, mientras que para `bibtex` está apenas por encima de 20. El nivel medio de desbalanceo para `enron`, con *Mea-*

nIR=83.9528 y *CVIR=1.9596*, es también mucho más alto que el de `bibtex`, con *MeanIR=12.4983* y *CVIR=0.4051*. A pesar de que en este caso los valores de las métricas *Card* y *Dens* son ligeramente más altas en `enron` que en `bibtex`, en general no existe una correlación entre estas dos métricas y los niveles de desbalanceo de los MLD.

A diferencia de *Card* y *Dens*, que únicamente evidencian el número promedio de etiquetas por instancia, el uso conjunto de las métricas *MeanIR* y *CVIR* indicaría si un MLD está desbalanceado o no, mientras que la métrica *IRLbl* sería de utilidad para la evaluación de etiquetas individuales. Como regla general, cualquier MLD que tenga un valor para *MeanIR* superior a 1.5 (50 % más de muestras con la etiqueta mayoritaria frente a la minoritaria considerada en cada caso, de media) y un valor para CVIR superior a 0.2 (un 20 % de varianza en los valores de *IRLbl*) debería ser considerado como desbalanceado.

3.5. Técnicas de remuestreo aleatorio

Para diseñar cualquier algoritmo de remuestreo para conjuntos de datos multietiqueta hay que partir considerando cómo se abordará la naturaleza específica de estos, ya que la variable de salida no es una etiqueta de clase sino un conjunto de ellas.

En el presente estudio se siguen dos enfoques a la hora de decidir qué casos serán considerados como minoritarios o mayoritarios. En los apartados siguientes se presentan dichos enfoques y se describe para cada uno de ellos los algoritmos propuestos.

3.5.1. Remuestreo basado en la transformación LP

El primer enfoque está basado en la transformación LP (Boutell et al. [2004]), mediante el que un conjunto de datos multietiqueta se convierte en uno multiclase, procesando cada combinación distinta de etiquetas como identificador de

clase. Esta transformación ha sido utilizada satisfactoriamente en clasificadores como RAkEL (Tsoumakas and Vlahavas [2007]) y HOMER (Tsoumakas et al. [2008]), habiéndose empleado también para otros fines como el particionamiento estratificado de conjuntos de datos multietiqueta (Sechidis et al. [2011]).

En comparación con la transformación BR, que hubiese implicado un tratamiento del desbalanceo por parejas de etiquetas, LP permite realizar un tratamiento conjunto teniendo en cuenta las relaciones entre las mismas. Por esta razón, y los antecedentes citados, pensamos que es una técnica que merece la pena ser probada en el contexto del problema que se plantea.

Se proponen dos algoritmos basados en este enfoque: LP-RUS y LP-ROS. Ambos interpretan cada combinación de etiquetas como identificador de clase, efectuando el primero un bajomuestreo mediante la eliminación de combinaciones mayoritarias (muy frecuentes) y el segundo un sobremuestreo clonando muestras con las combinaciones minoritarias.

Los algoritmos de remuestreo LP-RUS y LP-ROS, que se describen detalladamente a continuación, no dependen de las métricas de desbalanceo explicadas anteriormente. Antes de ser aplicados, por tanto, habría que evaluar qué conjuntos de datos pueden beneficiarse de esta técnica.

LP-RUS

LP-RUS es un algoritmo de *undersampling* para conjuntos de datos multietiqueta que opera eliminando aleatoriamente muestras asociadas a combinaciones de etiquetas mayoritarias. El proceso se detiene cuando el MLD se ha reducido en el porcentaje indicado como parámetro de entrada. El algoritmo realiza este trabajo según el pseudo-código mostrado en el Algoritmo 2.

Este procedimiento trata de alcanzar una representación en el MLD de cada combinación de etiquetas lo más cercana posible a una distribución uniforme. No obstante, dado que se impone un límite al tamaño mínimo del MLD mediante el parámetro P, el conjunto podría quedar con un cierto grado de desequilibrio. En

Algorithm 2 Pseudo-código del algoritmo LP-RUS.

Inputs: <Dataset> D, <Percentage> P
Outputs: MLD preprocesado

$samplesToDelete \leftarrow |D|/100 * P$ ▷ Reducción de un P %
▷ Agrupar muestras según su labelset
for $i = 1 \rightarrow |labelsets|$ **do**
    $labelSetBag_i \leftarrow$samplesWithLabelset(i)
end for
▷ Calcular el número medio de muestras por labelset
$meanSize \leftarrow 1/|labelsets| * \sum_{i=1}^{|labelsets|} |labelSetBag_i|$
▷ Obtener paquetes con etiquetas mayoritarias
for each $labelSetBag_i$ **in** $labelSetBag$ **do**
    if $|labelSetBag_i| > meanSize$ **then**
        $majBag_i \leftarrow labelSetBag_i$
    end if
end for
$meanRed \leftarrow samplesToDelete/|majBag|$
$majBag \leftarrow$ SortFromSmallestToLargest$(majBag)$
▷ Calcular # de instancias a eliminar y eliminarlas
for each $majBag_i$ **in** $majBag$ **do**
    $rBag_i \leftarrow \min(|majBag_i| - meanSize, meanRed)$
    $remainder \leftarrow meanRed - rBag_i$
    distributeAmongBags$_{j>i}(remainder)$
    for $n = 1 \rightarrow rBag_i$ **do**
        $x \leftarrow$ random$(1, |majBag_i|)$
        deleteSample$(majBag_i, x)$
    end for
end for

cualquier caso el nivel de desbalanceo debería ser siempre inferior al que existía en el conjunto de datos original.

Si bien los niveles de desbalanceo son evaluados con respecto a cada combinación de etiquetas, este método elimina muestras pertenecientes a varias combinaciones de etiquetas, no solamente la combinación mayoritaria.

LP-ROS

LP-ROS es un método de *oversampling* para conjuntos de datos multietiqueta que clona aleatoriamente muestras asociadas a combinaciones de etiquetas minoritarias, hasta que el tamaño del MLD es un $P\,\%$ mayor que el original. El procedimiento seguido se describe en el Algoritmo 3.

Algorithm 3 Pseudo-código del algoritmo LP-ROS.

Inputs: <Dataset> D, <Percentage> P
Outputs: MLD preprocesado

$samplesToGenerate \leftarrow |D|/100 * P$ ▷ Incremento de un P %
▷ Agrupar muestras según su labelset
for $i = 1 \rightarrow |labelsets|$ **do**
    $labelSetBag_i \leftarrow$samplesWithLabelset(i)
end for
▷ Calcular el número medio de muestras por labelset
$meanSize \leftarrow 1/|labelsets| * \sum_{i=1}^{|labelsets|} |labelSetBag_i|$
▷ Obtener paquetes con etiquetas minoritarias
for each $labelSetBag_i$ **in** $labelSetBag$ **do**
    if $|labelSetBag_i| < meanSize$ **then**
        $minBag_i \leftarrow labelSetBag_i$
    end if
end for
$meanInc \leftarrow samplesToGenerate/|minBag|$
$minBag \leftarrow$ SortFromLargestToSmallest$(minBag)$
▷ Calcular # de instancias a agregar y añadirlas
for each $minBag_i$ **in** $minBag$ **do**
    $rBag_i \leftarrow \min(meanSize - |minBag_i|, meanInc)$
    $remainder \leftarrow meanInc - rBag_i$
    distributeAmongBags$_{j>i}(remainder)$
    for $n = 1 \rightarrow rBag_i$ **do**
        $x \leftarrow$ random$(1, |minBag_i|)$
        cloneSample$(minBag_i, x)$
    end for
end for

En este caso se obtiene una colección de grupos minoritarios $minBag_i$ con ($|labelsetBag_i| < meanSize$), se calcula $meanInc = samplesGenerate/minBag$ y se procesan los grupos minoritarios del más grande al más pequeño para determinar un incremento $minBag_i$. Si un $minBag_i$ alcanza $meanSize$ muestras antes de haber añadido $incrementBag_i$ instancias, el exceso se distribuye entre los demás $minBag$.

Por tanto, las combinaciones de etiquetas con una menor representación se beneficiarán de un mayor número de clones, persiguiendo ajustar la frecuencia de los *labelset* a una distribución uniforme, como en el caso de LP-RUS.

3.5.2. Remuestreo con evaluación individual de etiquetas

Si bien LP es un método fácilmente aplicable y que ha mostrado su efectividad en distintos escenarios, también es cierto que presenta varias restricciones. En cuanto a la evaluación del nivel de desbalanceo, LP está limitado por la dispersión de las etiquetas que exista en el MLD. Hay MLD con tantas combinaciones de etiquetas diferentes como muestras. Esto conlleva que todas ellas serían consideradas como mayoritarias y minoritarias al mismo tiempo, por lo que LP-ROS y LP-RUS difícilmente podrían corregir el desequilibrio existente. Por ejemplo, `cal500` tiene 502 instancias asignadas a 502 combinaciones distintas de etiquetas. Por tanto todos los *labelset* de este MLD son únicos. A pesar de que los valores para *MeanIR* y *CVIR* correspondientes a este MLD son relativamente altos, lo cual indica que está desbalanceado, el enfoque LP no será capaz de reequilibrarlo ya que todas las combinaciones están representadas por igual.

Una vía alternativa para llevar a cabo esta tarea, basada en las métricas descritas en la Sección 3.4, sería evaluando el nivel de desbalanceo individual para cada etiqueta. Aquellas etiquetas cuyo *IRLbl* esté por encima de *MeanIR* serían consideradas como etiquetas minoritarias. Este criterio sería usado para extraer muestras de datos a clonar o para evitar que sean eliminadas. Todas las demás etiquetas, aquellas cuyo *IRLbl* es menor que el *MeanIR*, serían tratadas como etiquetas mayoritarias.

A diferencia de las propuestas anteriores basadas en la transformación LP, los algoritmos que se proponen aquí, ML-ROS y ML-RUS, sí aprovecharían las citadas métricas. Ambos tienen como finalidad remuestrear conjuntos de datos multietiqueta, pero las instancias a eliminar o clonar serán seleccionadas mediante una evaluación individual del desbalanceo para cada etiqueta, en lugar de combinaciones completas de etiquetas. La piedra angular de estos algoritmos son las métricas *MeanIR* e *IRLbl*.

ML-ROS

El algoritmo ML-ROS, cuyo pseudo-código se muestra en el Algoritmo 4, utiliza la medida *IRLbl* para obtener paquetes de instancias en los que aparecen etiquetas minoritarias (cuyo *IRLbl* es superior a *MeanIR*). Las instancias a clonar se toman aleatoriamente de esos paquetes. El método recalcula el *IRLbl* en cada ciclo del bucle, excluyendo del proceso aquellas etiquetas minoritarias que alcancen el nivel *MeanIR*.

Ha de tenerse en cuenta que las muestras de datos clonadas también pueden contener etiquetas no minoritarias y, como consecuencia, el proceso podría incrementar la frecuencia de las mismas en el MLD.

ML-RUS

ML-RUS es un algoritmo de *undersampling* basado en el enfoque de ML-ROS (véase Algoritmo 5). En este caso se utilizan los paquetes de instancias con etiquetas minoritarias para evitar la eliminación de las mismas. Las instancias contenidas en dichos paquetes se excluirán del proceso de borrado. Este opera aleatoriamente sobre las muestras restantes del MLD. Estas únicamente pueden ser muestras con etiquetas mayoritarias, cuyo *IRLbl* está por debajo de *MeanIR*.

Al igual que ML-ROS, ML-RUS recalcula los *IRLbl* de las etiquetas afectadas por la operación, retirando aquellas que alcancen el nivel indicado por *MeanIR*.

Algorithm 4 Pseudo-código del algoritmo ML-ROS.

Inputs: <Dataset> D, <Percentage> P
Outputs: MLD preprocesado

$samplesToClone \leftarrow |D|/100 * P$ ▷ P % de incremento
$L \leftarrow$ labelsInDataset(D) ▷ Obtener el conjunto completo de etiquetas
$MeanIR \leftarrow$ calculateMeanIR(D, L)
for each $label$ **in** L **do** ▷ Paquetes de muestras con etiquetas minoritarias
    $IRLbl_{label} \leftarrow$ calculateIRperLabel$(D, label)$
    if $IRLbl_{label} > MeanIR$ **then**
        $minBag_{i++} \leftarrow Bag_{label}$
    end if
end for
while $samplesToClone > 0$ **do** ▷ Bucle de clonación de instancias
    ▷ Clonar una muestra aleatoria de cada paquete minoritario
    for each $minBag_i$ **in** $minBag$ **do**
        $x \leftarrow$ random$(1, |minBag_i|)$
        cloneSample$(minBag_i, x)$
        if $IRLbl_{minBag_i} <= MeanIR$ **then**
            $minBag \rightarrow minBag_i$ ▷ Excluir del proceso de clonado
        end if
        - -$samplesToClone$
    end for
end while

3.5.3. Experimentación

A fin de comprobar el rendimiento de los cuatro algoritmos de preprocesamiento descritos en la sección anterior, cada uno de ellos ha sido probado sobre un conjunto de MLD y los cambios en los resultados de clasificación se han verificado con varios clasificadores distintos. A continuación se detalla la configuración de esta experimentación y se exponen los resultados obtenidos a partir de ella.

Configuración de experimentación

Los métodos de remuestreo propuestos han sido probados con los conjuntos de datos cuyas características se mostraron en la Tabla 3.2 (página 128). En ella

Algorithm 5 Pseudo-código del algoritmo ML-RUS.

Inputs: <Dataset> D, <Percentage> P
Outputs: MLD preprocesado
$samplesToDelete \leftarrow |D|/100 * P$ ▷ P % de reducción
$L \leftarrow$ labelsInDataset(D) ▷ Obtener el conjunto completo de etiquetas
$MeanIR \leftarrow$ calculateMeanIR(D, L)
for each *label* **in** L **do** ▷ Paquete de muestra con etiquetas minoritarias
    $IRLbl_{label} \leftarrow$ calculateIRperLabel($D, label$)
    if $IRLbl_{label} > MeanIR$ **then**
        $minBag \leftarrow Bag_{label}$
    end if
end for
$D' \leftarrow D - minBag$ ▷ Paquete solo con etiquetas mayoritarias
while $samplesToDelete > 0$ **do** ▷ Bucle de eliminación de instancias
    ▷ Seleccionar muestra aleatoria de D'
    $x \leftarrow$ random($1, |D'|$)
    deleteSample(D', x)
    - -$samplesToDelete$
end while
$D \leftarrow D' + minBag$

se indican algunas características básicas de los MLD: número de atributos, de muestras y de etiquetas, número medio de etiquetas por muestra y las medidas de caracterización del desbalanceo. Los MLD `emotions` y `scene` no se han utilizado, ya que su bajo *MeanIR* denota que son conjuntos de datos equilibrados.

Como puede apreciarse en la citada tabla, hay una variedad de valores en los datos para *Card* y *Dens* de estos MLD, así como grandes diferencias entre los números de etiquetas, atributos y muestras y también las métricas de desbalanceo. El objetivo es analizar los métodos de remuestreo propuestos con MLD de diversas características. Se ha utilizado un esquema de validación cruzada con 2x5 particiones, habiendo procesado las particiones de entrenamiento con cada uno de los métodos propuestos usando tres valores distintos para el parámetro que fija el porcentaje: 10, 20 y 25.

En cuanto a algoritmos de clasificación multietiqueta, se han utilizado CLR (Fürnkranz et al. [2008]), RAkEL (Tsoumakas and Vlahavas [2007]), IBLR (Cheng and Hüllermeier [2009]) y HOMER (Tsoumakas et al. [2008]). En aquellos casos en que se precisa un clasificador binario o multiclase subyacente se ha usado C4.5. El número de *clusters* para HOMER se ha fijado al mínimo entre 4 y el número de etiquetas con que cuente el MLD. Para el resto de parámetros se han utilizado parámetros por defecto. Cada algoritmo ha sido ejecutado con los MLD sin preprocesar y tras haberlos preprocesados con LP-RUS, LP-ROS, ML-RUS y ML-ROS.

En la Sección 1.6 del Capítulo 1 (página 56) se describieron las distintas métricas que es posible utilizar a la hora de evaluar el rendimiento en clasificación multietiqueta. Entre ellas se encontraban las métricas de tipo *label-based*. Estas suelen ser utilizadas cuando se aborda el problema del desbalanceo ya que, como se indica en Tang et al. [2009], el enfoque *macro-averaging* se ve muy influenciado por las categorías raras (etiquetas minoritarias), mientras que el *micro-averaging*, por la forma en que agrega los resultados y calcula la medida al final, se ve más afectado por las etiquetas mayoritarias. Por esta razón, para evaluar los métodos propuestos se ha optado por utilizar las métricas Macro-Fmeasure y Micro-Fmeasure, ya que estas aúnan bajo cada enfoque de promedio tanto *Precision* como *Recall*. Además también se ha obtenido la métrica *Accuracy*, para contar con una medida adicional complementaria. Esta métrica es una medida que evalúa tanto el rendimiento predictivo positivo como negativo de los clasificadores.

La experimentación se ha estructurado en tres etapas. Las dos primeras tienen la finalidad de escoger la mejor configuración para *undersampling* y la mejor para *oversampling*. La fase final compara estas dos mejores configuraciones con los resultados de clasificación obtenidos sin preprocesamiento. En cada etapa los resultados obtenidos se analizan estadísticamente en dos pasos, según se recomienda en Garcıa and Herrera [2008] y García et al. [2010]. En el primer paso se usa el test de Friedman para generar un ranking con los métodos y establecer si existe alguna diferencia estadísticamente significativa. El segundo paso efectúa

una comparación múltiple mediante el procedimiento *post-hoc* de Holm y Saffer, con capacidad para detectar diferencias por pares entre los algoritmos.

Selección del mejor método de *undersampling*

En las Tablas 3.6, 3.8 y 3.7 (véase la Sección 3.8, al final de este capítulo) se muestran los resultados de clasificación asociados a los métodos de *undersampling*: LP-RUS y ML-RUS. Los mejores valores para cada combinación clasificador-MLD se han destacado en negrita.

Los resultados obtenidos del análisis estadístico son los resumidos en la Tabla 3.3. Hay una columna para cada métrica de evaluación, mostrando el ranking medio para los algoritmos. El significado de los símbolos dispuestos a la derecha de los valores es el siguiente:

Tabla 3.3: Ranking promedio para *undersampling*

Algoritmo	**Accuracy**		**Micro-FM**		**Macro-FM**	
LP-RUS 10	3.114	$\downarrow$	2.795	$\leftrightarrow$	2.523	$\star$
LP-RUS 20	4.227	$\downarrow\downarrow$	3.523	$\downarrow\downarrow$	3.443	$\leftrightarrow$
LP-RUS 25	5.034	$\downarrow\downarrow$	4.432	$\downarrow\downarrow$	3.841	$\downarrow\downarrow$
ML-RUS 10	2.136	$\star$	2.193	$\star$	3.034	$\leftrightarrow$
ML-RUS 20	3.023	$\downarrow$	2.716	$\leftrightarrow$	3.988	$\downarrow\downarrow$
ML-RUS 25	3.466	$\downarrow\downarrow$	5.341	$\downarrow\downarrow$	4.170	$\downarrow\downarrow$

- $\star$: Un asterisco denota que el test de Friedman ha rechazado la hipótesis nula. Por tanto existen diferencias estadísticamente significativas entre algunos de los algoritmos. El que aparece a la izquierda del asterisco tiene el mejor ranking.
- $\leftrightarrow$: No existe diferencia estadísticamente significativa entre este algoritmo y el mejor.
- $\downarrow$: Este método es estadísticamente peor que el mejor con un *p-value*$<$ 0,1.

- ⇊: Este método es estadísticamente peor que el mejor con un *p-value*$<$ 0,05.

La Tabla 3.3 refleja que ML-RUS 10 es la configuración ganadora para las métricas *Accuracy* y *Micro-FMeasure*, mientras que para *Macro-FMeasure* aparece en segunda posición pero sin que exista diferencia significativa respecto al mejor (LP-RUS 10) desde un punto de vista estadístico. En consecuencia puede concluirse que, en promedio, ML-RUS 10 es el mejor método de *undersampling*.

Selección del mejor método de *oversampling*

En las Tablas 3.9, 3.11 y 3.10 (Sección 3.8) se presentan los resultados obtenidos para los métodos de *oversampling*. En la Tabla 3.4, correspondiente al análisis estadístico, puede apreciarse que existe una diferencia estadísticamente significativa entre los métodos ML-ROS y LP-ROS. Para *Accuracy* y *Micro-FMeasure* el ganador es ML-ROS 10, mientras que usando *Macro-FMeasure* lo es ML-ROS 25. En este último caso, no hay diferencias estadísticas respecto a ML-ROS 10. En contraposición, para *Micro-FMeasure* la configuración ML-ROS 10 aparece como estadísticamente destacada sobre ML-ROS 25. Por tanto puede concluirse que ML-ROS 10 es, en promedio, el mejor método de *oversampling*.

Tabla 3.4: Ranking promedio para *oversampling*

Algoritmo	**Accuracy**		**Micro-FM**		**Macro-FM**	
LP-ROS 10	4.830	⇊	5.193	⇊	5.170	⇊
LP-ROS 20	4.489	⇊	4.943	⇊	4.807	⇊
LP-ROS 25	4.307	⇊	4.773	⇊	4.591	⇊
ML-ROS 10	2.284	⋆	1.409	⋆	2.250	↔
ML-ROS 20	2.636	↔	2.080	↔	2.160	↔
ML-ROS 25	2.455	↔	2.602	⇊	2.023	⋆

Tras finalizar los dos primeros pasos de la experimentación puede deducirse que ML-ROS/ML-RUS son en términos generales mejores que los respectivos LP-ROS/LP-RUS, si bien no siempre existe una diferencia estadísticamente significativa. Por tanto el remuestreo basado en la evaluación individual del nivel

Tabla 3.5: Rankings promedio de la fase final

Algoritmo	**Accuracy**		**Micro-FM**		**Macro-FM**	
Base	1.852	$\leftrightarrow$	2.636	$\downarrow\downarrow$	1.898	$\leftrightarrow$
ML-RUS 10	2.364	$\downarrow\downarrow$	1.568	$\star$	2.386	$\downarrow\downarrow$
ML-ROS 10	1.784	$\star$	1.795	$\leftrightarrow$	1.716	$\star$

de desbalanceo para cada etiqueta parece ser superior al enfoque basado en la transformación LP.

Clasificador base vs mejor preprocesamiento

La tercera fase de la experimentación compara los resultados obtenidos por ML-RUS 10 y ML-ROS 10 con aquellos generados por los mismos clasificadores sin ningún preprocesamiento, designados como **Base** en todas las tablas. Las Tablas 3.12, 3.14 y 3.13 (Sección 3.8) contienen los resultados de esta fase final. La salida de los tests estadísticos (Tabla 3.5) indica que ML-ROS 10 tiene un rendimiento significativamente superior al de ML-RUS 10 con las métricas *Accuracy* y *Macro-FMeasure*. Para *Micro-FMeasure* tanto ML-RUS como ML-ROS son estadísticamente mejores que el resultado base, pero no hay diferencias significativas entre ellos.

Globalmente ML-ROS es apreciablemente mejor que ML-RUS. Incluso a pesar de que este último obtiene la mejor posición en el ranking con la métrica *Micro-FMeasure*, la diferencia respecto a ML-ROS no es significativa. Por otra parte, el rendimiento de ML-ROS según las métricas *Accuracy* y *Macro-FMeasure* es significativamente mejor que la de ML-RUS desde un punto de vista estadístico. Dado que *Micro-FMeasure* se ve más afectada por las clases mayoritarias y *Macro-FMeasure* por las minoritarias, estos resultados reflejan que ML-RUS está eliminando adecuadamente etiquetas comunes, mientras que ML-ROS incrementa la presencia de etiquetas raras. Teóricamente la eliminación de instancias con etiquetas mayoritarias debería producir un efecto similar a la adición de nuevas instancias con etiquetas minoritarias. Las dos acciones tienden a equilibrar la re-

presentación de las etiquetas en el MLD. Sin embargo, debe tenerse en cuenta que las instancias de un MLD representan a un conjunto de etiquetas, no solo a una como en clasificación tradicional. La eliminación de una muestra tiene un efecto colateral sobre un número potencialmente grande de etiquetas. Por ello, la pérdida de información provocada por ML-RUS es un punto en su contra cuando se compara con ML-ROS.

De esta experimentación exploratoria sobre cómo las técnicas de remuestreo clásicas podrían ser adaptadas para trabajar con conjuntos de datos multietiqueta pueden inferirse las siguientes consecuencias:

- Si bien únicamente se han aplicado los métodos de remuestreo más básicos, basados en la eliminación y clonado de muestras aleatorias, se ha obtenido una mejora apreciable y general en los resultados, en ocasiones incluso con diferencias estadísticamente significativas.
- El mejor método de bajomuestreo tiene un rendimiento significativamente peor que el mejor método de sobremuestreo en dos de las tres métricas de evaluación. Este resultado es consistente con estudios como el de García et al. [2012] y la propia naturaleza de los MLD, ya que la eliminación de una instancia no influye solamente sobre la etiqueta evaluada, sino también sobre el resto de etiquetas que aparecen en la muestra.
- Centrándonos en las técnicas de sobremuestreo, el enfoque ML es claramente superior al LP (véase la Tabla 3.4). El uso de combinaciones de etiquetas completas para evaluar el desbalanceo incurre en el riesgo de incrementar solo el número de instancias con etiquetas mayoritarias, ya que estas pueden aparecer en muchas combinaciones distintas. Las etiquetas minoritarias podrían aparecer habitualmente juntas, generando combinaciones que resultan ser más frecuentes que estas asociadas a las etiquetas mayoritarias. La evaluación individual del desbalanceo seguida por el enfoque ML, con la extracción de paquetes de instancias con etiquetas minoritarias, garantiza que todas las muestras clonadas incluyen alguna etiqueta minoritaria, si bien esta podría ir acompañada de etiquetas mayoritarias.

3.6. Conclusiones

Como resultado del trabajo expuesto en este capítulo se han propuesto varias métricas diseñadas para medir el nivel de desbalanceo en conjuntos de datos multietiqueta, junto con cuatro algoritmos de remuestreo cuyo funcionamiento se ha probado experimentalmente y analizado estadísticamente. LP-RUS es un algoritmo de *undersampling* aleatorio, mientras que *LP-ROS* es un algoritmo de *oversampling*, en ambos casos tomando como valor de clase la combinación de etiquetas presente en cada muestra de datos. ML-RUS y ML-ROS también son métodos de *undersampling* y *oversampling* aleatorio, pero operan evaluando el desbalanceo individual de cada etiqueta, en lugar de operar con combinaciones completas de ellas.

Las métricas propuestas pueden ser utilizadas para evaluar el nivel de desbalanceo con el objetivo de decidir si un cierto MLD podría o no beneficiarse de los métodos de remuestreo presentados. Usando estas métricas se determinó que `emotions` y `scene` no deberían ser preprocesados, ya que no presentan un desequilibrio significativo. Estas métricas son también la base del funcionamiento de los algoritmos ML-ROS y ML-RUS, que las utilizan para decidir qué instancias serán clonadas/eliminadas. Como se ha demostrado, el aprovechamiento de esta información en el diseño de los citados algoritmos ha conllevado una mejora importante en relación a otros métodos que, como LP-ROS y LP-RUS, están basados en una transformación básica, pero no usan esas métricas.

Finalmente, la implementación de estos algoritmos básicos de remuestreo junto con la experimentación llevada a cabo demuestran que las técnicas de remuestreo pueden ser una alternativa a otras propuestas publicadas para trabajar con conjuntos de datos multietiqueta desbalanceados, abriendo un nuevo camino para afrontar este problema. Los resultados obtenidos colocan a los algoritmos propuestos siempre en las primeras posiciones del ranking frente a los resultados base, en algunos casos con diferencias estadísticamente significativas. Al igual que se hizo en su día en el campo de la clasificación tradicional, el siguiente paso sería el estudio y diseño de nuevos algoritmos capaces de generar muestras multi-

etiqueta sintéticas o utilizar métodos heurísticos para la eliminación, en lugar de aleatorios. Estos temas serán los abordados en el siguiente capítulo de la presente tesis.

3.7. Publicaciones asociadas a este trabajo

Las métricas *IRLbl*, *MeanIR* y *CVIR*, junto con los algoritmos LP-RUS y LP-ROS, fueron presentados en HAIS'13 (Charte et al. [2013]), *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "A First Approach to Deal with Imbalance in Multi-label Datasets". In* Proc. 8th Int. Conf. Hybrid Artificial Intelligent Systems, Salamanca, Spain, HAIS'13*, volume 8073 of* LNCS*, pages 150–160, 2013. ISBN 978-3-642-40845-8. doi:10.1007/978-3-642-40846-5_16.*

Los algoritmos ML-RUS y ML-ROS, basados en las citadas medidas, junto con la experimentación descrita en este capítulo han sido publicados en la revista Neurocomputing (Charte et al. [2015]), *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "Addressing Imbalance in Multilabel Classification: Measures and Random Resampling Algorithms". Neurocomputing, 2015 (Available Online). doi: 10.1016/j.neucom.2014.08.091.*

3.8. Tablas de resultados

Tabla 3.6: Resultados de los algoritmos de *undersampling* - Accuracy

Clasificador	MLD	LP-RUS10	LP-RUS20	LP-RUS25	ML-RUS10	ML-RUS20	ML-RUS25
CLR	bibtex	0.2043	0.1958	0.1880	**0.2292**	0.2286	0.2261
HOMER	bibtex	0.2409	0.2216	0.2193	**0.2618**	0.2603	0.2580
IBLR	bibtex	0.1424	0.1276	0.1221	**0.1684**	0.1654	0.1652
RAkEL	bibtex	0.2776	0.2646	0.2657	**0.2966**	0.2937	0.2933
CLR	cal500	0.1787	0.1787	0.1787	0.1708	0.1771	**0.1796**
HOMER	cal500	0.2410	**0.2500**	0.2455	0.2346	0.2324	0.2348
IBLR	cal500	**0.1926**	**0.1926**	**0.1926**	0.1898	0.1843	0.1897
RAkEL	cal500	0.2135	0.2135	0.2135	0.2101	0.2140	**0.2154**
CLR	corel16k	0.0434	0.0409	0.0401	0.0453	0.0454	**0.0466**
HOMER	corel16k	0.1080	0.1087	0.1023	0.1118	0.1112	**0.1132**
IBLR	corel16k	0.0224	0.0208	0.0200	**0.0256**	0.0254	0.0250
RAkEL	corel16k	0.0618	0.0592	0.0592	0.0633	0.0628	**0.0638**
CLR	corel5k	0.0328	0.0319	0.0292	0.0355	**0.0368**	0.0363
HOMER	corel5k	0.0956	0.0870	0.0907	**0.1016**	0.0977	0.1010
IBLR	corel5k	0.0266	0.0253	0.0235	0.0296	0.0323	**0.0324**
RAkEL	corel5k	0.0534	0.0518	0.0480	0.0589	0.0592	**0.0594**
CLR	enron	0.3912	0.3456	0.3422	**0.4184**	0.4156	0.4156
HOMER	enron	0.3822	0.3353	0.3349	**0.4085**	0.4026	0.3976
IBLR	enron	0.2834	0.2756	0.2752	0.3005	0.2946	**0.3147**
RAkEL	enron	0.3812	0.3324	0.3292	**0.4034**	0.3966	0.4004
CLR	genbase	**0.9822**	0.9816	0.9812	0.9716	0.9528	0.9368
HOMER	genbase	0.9802	**0.9822**	0.9796	0.9764	0.9582	0.9411
IBLR	genbase	0.9785	**0.9795**	0.9770	0.9671	0.9476	0.9214
RAkEL	genbase	**0.9842**	**0.9842**	0.9839	0.9782	0.9616	0.9456
CLR	llog	0.0338	0.0278	0.0239	0.0458	**0.0504**	0.0492
HOMER	llog	0.0866	0.0927	0.0889	**0.1038**	0.0992	0.0972
IBLR	llog	0.0328	0.0288	0.0232	**0.0352**	0.0327	0.0350
RAkEL	llog	0.1243	0.1261	0.1268	**0.1325**	0.1296	0.1258
CLR	mediamill	**0.4456**	0.4370	0.4342	0.4438	0.4368	0.4334
HOMER	mediamill	0.4026	0.3870	0.3842	**0.4089**	0.4037	0.4012
IBLR	mediamill	**0.4620**	0.4539	0.4486	0.4590	0.4504	0.4455
RAkEL	mediamill	0.4115	0.4013	0.3964	**0.4144**	0.4088	0.4072
CLR	slashdot	0.2816	0.2152	0.2024	**0.3194**	0.3117	0.2991
HOMER	slashdot	**0.3403**	0.3082	0.3216	0.3314	0.3226	0.3084
IBLR	slashdot	0.0880	0.0854	0.0811	0.1486	0.1364	**0.1548**
RAkEL	slashdot	0.3015	0.2352	0.2188	**0.3392**	0.3279	0.3159
CLR	tmc2007	**0.6061**	0.5938	0.5887	0.6020	0.5860	0.5774
HOMER	tmc2007	**0.5930**	0.5760	0.5690	0.5897	0.5801	0.5701
IBLR	tmc2007	**0.5266**	0.5168	0.5113	0.5184	0.5119	0.5072
RAkEL	tmc2007	**0.5950**	0.5806	0.5731	0.5913	0.5758	0.5676
CLR	yeast	0.4621	0.4556	0.4566	**0.4706**	0.4649	0.4549
HOMER	yeast	0.4294	0.4130	0.4086	**0.4312**	0.4294	0.4144
IBLR	yeast	0.5148	0.5048	0.5017	**0.5150**	0.5102	0.5074
RAkEL	yeast	0.4242	0.4144	0.4160	**0.4344**	0.4314	0.4234

Tabla 3.7: Resultados de los algoritmos de *undersampling* - Micro-FMeasure

Clasificador	**MLD**	**LP-RUS10**	**LP-RUS20**	**LP-RUS25**	**ML-RUS10**	**ML-RUS20**	**ML-RUS25**
CLR	bibtex	0.7607	0.7429	0.7322	**0.7793**	0.7740	0.3339
HOMER	bibtex	0.3530	0.3346	0.3338	**0.3648**	0.3592	0.3408
IBLR	bibtex	**0.3610**	0.3178	0.2986	0.3494	0.3158	0.2423
RAkEL	bibtex	0.4878	0.4645	0.4573	**0.5136**	0.5045	0.3978
CLR	cal500	0.6227	0.6227	0.6227	**0.6258**	0.6151	0.2993
HOMER	cal500	0.3749	0.3822	**0.3858**	0.3732	0.3635	0.3787
IBLR	cal500	0.2827	0.2827	0.2827	0.2777	0.2720	**0.3166**
RAkEL	cal500	**0.4343**	**0.4343**	**0.4343**	0.4188	0.4163	0.3518
CLR	corel16k	0.4188	0.4054	0.3917	**0.4300**	0.4239	0.0878
HOMER	corel16k	0.2212	0.2152	0.2130	**0.2327**	0.2256	0.1868
IBLR	corel16k	0.2865	0.2356	0.2092	**0.3049**	0.2600	0.0498
RAkEL	corel16k	0.3464	0.3369	0.3339	0.3511	**0.3556**	0.1144
CLR	corel5k	0.4494	0.4440	0.4248	0.4512	**0.4540**	0.0714
HOMER	corel5k	0.2050	0.1964	0.1965	**0.2086**	0.2041	0.1708
IBLR	corel5k	0.0378	0.0356	0.0337	0.0434	0.0434	**0.0558**
RAkEL	corel5k	0.3716	**0.3756**	0.3652	0.3707	0.3717	0.1108
CLR	enron	**0.6842**	0.6677	0.6544	0.6780	0.6814	0.5467
HOMER	enron	0.5318	0.5111	0.5004	**0.5482**	0.5446	0.5086
IBLR	enron	**0.5660**	0.5418	0.5299	0.5588	0.5279	0.4312
RAkEL	enron	**0.6262**	0.6034	0.5858	0.6195	0.6124	0.5218
CLR	genbase	**0.9887**	**0.9887**	**0.9887**	0.9844	0.9852	0.9374
HOMER	genbase	0.9898	**0.9910**	0.9886	0.9852	0.9786	0.9420
IBLR	genbase	0.9768	**0.9799**	0.9767	0.9478	0.8998	0.8920
RAkEL	genbase	0.9893	0.9893	0.9893	0.9875	**0.9902**	0.9488
CLR	llog	0.5574	0.5198	0.4866	**0.5863**	0.5851	0.0802
HOMER	llog	0.1378	0.1468	0.1400	**0.1594**	0.1542	0.1452
IBLR	llog	0.0535	0.0440	0.0420	0.0580	0.0565	**0.0588**
RAkEL	llog	0.2698	0.2618	0.2634	**0.2880**	0.2871	0.1914
CLR	mediamill	0.7664	0.7560	0.7508	**0.7750**	0.7720	0.5713
HOMER	mediamill	0.5602	0.5266	0.5234	0.5882	**0.5944**	0.5372
IBLR	mediamill	0.7525	0.7398	0.7326	0.7701	**0.7708**	0.5728
RAkEL	mediamill	0.6279	0.6105	0.6013	0.6510	**0.6558**	0.5449
CLR	slashdot	0.6546	0.6520	**0.6639**	0.6315	0.6354	0.4084
HOMER	slashdot	0.5781	0.5683	0.5598	**0.5997**	0.5874	0.4001
IBLR	slashdot	0.6504	**0.6634**	0.6554	0.6505	0.6218	0.2242
RAkEL	slashdot	0.6985	**0.7146**	0.7098	0.6675	0.6672	0.4248
CLR	tmc2007	0.7400	0.7278	0.7228	**0.7500**	0.7454	0.6884
HOMER	tmc2007	0.6860	0.6691	0.6614	**0.6908**	0.6893	0.6664
IBLR	tmc2007	0.7132	0.7073	0.7036	**0.7221**	0.7220	0.6170
RAkEL	tmc2007	0.7243	0.7092	0.7016	**0.7337**	0.7263	0.6699
CLR	yeast	0.6462	0.6380	0.6286	0.6516	**0.6573**	0.6055
HOMER	yeast	0.5632	0.5504	0.5407	0.5680	**0.5795**	0.5612
IBLR	yeast	0.7133	0.7124	0.7083	**0.7137**	0.7129	0.6378
RAkEL	yeast	0.5839	0.5712	0.5644	0.5913	**0.5970**	0.5696

Tabla 3.8: Resultados de los algoritmos de *undersampling* - Macro-FMeasure

Clasificador	MLD	LP-RUS10	LP-RUS20	LP-RUS25	ML-RUS10	ML-RUS20	ML-RUS25
CLR	bibtex	0.3328	0.3238	0.3240	0.3400	0.3409	**0.3448**
HOMER	bibtex	**0.2960**	0.2870	0.2871	0.2920	0.2890	0.2907
IBLR	bibtex	**0.2060**	0.1998	0.1904	0.2050	0.1977	0.1954
RAkEL	bibtex	0.3358	0.3328	0.3356	**0.3384**	0.3371	0.3371
CLR	cal500	**0.3323**	**0.3323**	**0.3323**	0.3128	0.3137	0.3067
HOMER	cal500	0.3194	0.3274	**0.3302**	0.3194	0.3172	0.3254
IBLR	cal500	0.2770	0.2770	0.2770	0.2744	0.2680	**0.2789**
RAkEL	cal500	0.2934	0.2934	0.2934	**0.3028**	0.3013	0.3021
CLR	corel16k	0.1001	0.0990	0.0967	0.1031	0.0968	**0.1054**
HOMER	corel16k	0.1272	0.1222	0.1168	0.1322	0.1374	**0.1380**
IBLR	corel16k	**0.1056**	0.0988	0.0946	0.1049	0.1054	0.0956
RAkEL	corel16k	0.1216	0.1180	0.1176	**0.1244**	0.1197	0.1218
CLR	corel5k	**0.1410**	0.1298	0.1328	0.1304	0.1386	0.1272
HOMER	corel5k	0.1682	0.1660	0.1628	0.1852	0.1840	**0.1856**
IBLR	corel5k	0.0939	0.0909	0.0840	0.1092	0.1069	**0.1104**
RAkEL	corel5k	0.1631	0.1652	0.1552	0.1792	**0.1831**	0.1775
CLR	enron	0.4208	0.4014	0.4184	0.4132	0.4055	**0.4306**
HOMER	enron	0.3702	0.3558	0.3641	**0.3798**	0.3746	0.3604
IBLR	enron	**0.3450**	0.3333	0.3300	0.3399	0.3296	0.3180
RAkEL	enron	0.4062	0.3968	0.4086	0.4039	0.3996	**0.4126**
CLR	genbase	**0.9846**	0.9845	0.9842	0.9675	0.9530	0.9303
HOMER	genbase	0.9796	**0.9836**	0.9775	0.9718	0.9517	0.9331
IBLR	genbase	0.9678	**0.9686**	0.9670	0.9424	0.9122	0.8929
RAkEL	genbase	**0.9890**	**0.9890**	0.9887	0.9834	0.9716	0.9501
CLR	llog	0.2224	0.2037	0.2032	0.2550	**0.2670**	0.2534
HOMER	llog	0.2210	0.2336	**0.2398**	0.2267	0.2166	0.2020
IBLR	llog	0.1738	0.1627	0.1589	**0.1998**	0.1786	0.1798
RAkEL	llog	0.2784	0.2791	**0.2927**	0.2670	0.2652	0.2538
CLR	mediamill	0.2294	**0.2358**	0.2340	0.2176	0.2218	0.2188
HOMER	mediamill	0.2374	**0.2444**	0.2348	0.2290	0.2115	0.2088
IBLR	mediamill	**0.2804**	0.2796	0.2776	0.2634	0.2452	0.2362
RAkEL	mediamill	0.2778	**0.2791**	0.2766	0.2692	0.2509	0.2434
CLR	slashdot	0.3800	0.3530	0.3472	**0.3898**	0.3834	0.3661
HOMER	slashdot	**0.4059**	0.3809	0.3825	0.3766	0.3733	0.3581
IBLR	slashdot	0.2276	0.2199	0.2192	0.2242	0.2218	**0.2530**
RAkEL	slashdot	0.3841	0.3675	0.3605	**0.3982**	0.3842	0.3750
CLR	tmc2007	**0.6091**	0.6089	0.6078	0.5954	0.5781	0.5717
HOMER	tmc2007	**0.5944**	0.5911	0.5861	0.5855	0.5722	0.5662
IBLR	tmc2007	0.4716	**0.4760**	0.4758	0.4406	0.4298	0.4260
RAkEL	tmc2007	**0.6022**	0.5966	0.5942	0.5878	0.5707	0.5652
CLR	yeast	0.4481	0.4538	**0.4562**	0.4483	0.4298	0.4285
HOMER	yeast	**0.4431**	0.4345	0.4274	0.4351	0.4238	0.4164
IBLR	yeast	**0.4990**	0.4828	0.4761	0.4597	0.4671	0.4570
RAkEL	yeast	0.4450	0.4407	0.4436	**0.4474**	0.4367	0.4357

Tabla 3.9: Resultados de los algoritmos de *oversampling* - Accuracy

Clasificador	**MLD**	**LP-RUS10**	**LP-RUS20**	**LP-RUS25**	**ML-RUS10**	**ML-RUS20**	**ML-RUS25**
CLR	bibtex	0.1670	0.1688	0.1724	0.2364	0.2367	**0.2388**
HOMER	bibtex	0.1515	0.1542	0.1504	**0.2677**	0.2614	0.2548
IBLR	bibtex	0.1007	0.1014	0.1038	0.1768	0.1767	**0.1783**
RAkEL	bibtex	0.2072	0.2064	0.2066	**0.2925**	0.2882	0.2888
CLR	cal500	**0.2150**	0.2147	0.2110	0.2038	0.2138	0.2140
HOMER	cal500	0.2040	0.2062	0.2030	**0.2210**	0.2116	0.2128
IBLR	cal500	0.1940	0.1896	0.1896	0.1900	**0.1941**	0.1940
RAkEL	cal500	0.2060	0.2036	0.2047	**0.2121**	0.2102	0.2109
CLR	corel16k	0.0555	0.0576	**0.0584**	0.0480	0.0508	0.0500
HOMER	corel16k	0.0764	0.0759	0.0760	**0.1107**	0.1039	0.1026
IBLR	corel16k	0.0454	0.0472	**0.0475**	0.0292	0.0324	0.0344
RAkEL	corel16k	0.0590	0.0594	0.0603	**0.0700**	0.0698	0.0690
CLR	corel5k	0.0417	0.0426	**0.0444**	0.0390	0.0416	0.0429
HOMER	corel5k	0.0696	0.0700	0.0727	**0.0996**	0.0958	0.0946
IBLR	corel5k	0.0347	0.0361	**0.0368**	0.0327	0.0340	0.0351
RAkEL	corel5k	0.0621	0.0608	0.0608	0.0612	0.0634	**0.0650**
CLR	enron	0.3167	0.3160	0.3170	**0.4068**	0.4019	0.4030
HOMER	enron	0.2665	0.2653	0.2598	**0.4024**	0.3840	0.3926
IBLR	enron	0.2530	0.2457	0.2466	0.3155	**0.3182**	0.3147
RAkEL	enron	0.2802	0.2797	0.2783	**0.3890**	0.3838	0.3808
CLR	genbase	0.9755	0.9770	0.9764	0.9842	0.9844	**0.9849**
HOMER	genbase	0.9776	0.9792	0.9800	0.9834	**0.9849**	0.9820
IBLR	genbase	0.9798	0.9809	0.9809	**0.9842**	0.9836	0.9841
RAkEL	genbase	0.9820	0.9842	0.9844	0.9864	0.9866	**0.9871**
CLR	llog	0.0272	0.0258	0.0246	**0.0470**	0.0437	0.0443
HOMER	llog	0.0642	0.0672	0.0672	**0.1105**	0.1031	0.0961
IBLR	llog	0.0497	**0.0533**	0.0524	0.0357	0.0367	0.0353
RAkEL	llog	0.0922	0.0950	0.0940	**0.1324**	0.1286	0.1315
CLR	mediamill	0.3876	0.3884	0.3881	**0.4559**	0.4558	0.4556
HOMER	mediamill	0.2749	0.2770	0.2776	**0.4002**	0.3946	0.3925
IBLR	mediamill	0.3205	0.3210	0.3214	**0.4644**	0.4633	0.4624
RAkEL	mediamill	0.2808	0.2816	0.2816	**0.4114**	0.4062	0.4046
CLR	slashdot	0.1993	0.2212	0.2298	0.3260	0.3258	**0.3302**
HOMER	slashdot	0.2610	0.2713	0.2585	**0.3550**	0.3432	0.3488
IBLR	slashdot	0.1646	0.1700	**0.1723**	0.1343	0.1384	0.1392
RAkEL	slashdot	0.2116	0.2370	0.2458	0.3496	0.3518	**0.3578**
CLR	tmc2007	0.4845	0.4858	0.4850	0.6148	**0.6164**	0.6152
HOMER	tmc2007	0.4020	0.4056	0.4073	0.6012	0.6012	**0.6016**
IBLR	tmc2007	0.3481	0.3483	0.3492	**0.5281**	0.5232	0.5232
RAkEL	tmc2007	0.4170	0.4180	0.4182	0.6022	**0.6023**	0.6008
CLR	yeast	0.3928	0.3894	0.3866	**0.4614**	0.4567	0.4572
HOMER	yeast	0.3316	0.3293	0.3290	**0.4053**	0.3979	0.3982
IBLR	yeast	0.3910	0.3936	0.3946	**0.5142**	0.5048	0.4998
RAkEL	yeast	0.3422	0.3380	0.3394	**0.4101**	0.4022	0.4063

Tabla 3.10: Resultados de los algoritmos de *oversampling* - Micro-FMeasure

Clasificador	MLD	LP-RUS10	LP-RUS20	LP-RUS25	ML-RUS10	ML-RUS20	ML-RUS25
CLR	bibtex	0.6529	0.6514	0.6535	**0.7690**	0.7555	0.7542
HOMER	bibtex	0.2084	0.2104	0.2098	**0.3656**	0.3532	0.3520
IBLR	bibtex	0.1661	0.1698	0.1728	0.4070	0.4053	**0.4074**
RAkEL	bibtex	0.2999	0.2961	0.2946	**0.4756**	0.4567	0.4444
CLR	cal500	0.5526	0.5569	0.5526	**0.5911**	0.5729	0.5669
HOMER	cal500	0.3385	0.3404	0.3358	**0.3512**	0.3451	0.3460
IBLR	cal500	0.2790	0.2737	0.2724	0.2802	0.2822	**0.2828**
RAkEL	cal500	0.3387	0.3356	0.3376	**0.3709**	0.3510	0.3520
CLR	corel16k	0.3140	0.3209	0.3209	**0.4232**	0.4168	0.4070
HOMER	corel16k	0.1339	0.1348	0.1337	**0.2128**	0.2032	0.2002
IBLR	corel16k	0.1156	0.1204	0.1208	**0.2718**	0.2456	0.2366
RAkEL	corel16k	0.1352	0.1336	0.1335	**0.2998**	0.2602	0.2446
CLR	corel5k	0.3489	0.3484	0.3527	**0.4402**	0.4326	0.4290
HOMER	corel5k	0.1335	0.1310	0.1334	**0.2040**	0.1966	0.1915
IBLR	corel5k	0.0500	0.0523	0.0530	0.0512	0.0570	**0.0604**
RAkEL	corel5k	0.1980	0.1861	0.1839	**0.3113**	0.2932	0.2888
CLR	enron	0.5929	0.5952	0.5929	**0.6772**	0.6752	0.6762
HOMER	enron	0.3840	0.3852	0.3858	**0.5237**	0.5108	0.5074
IBLR	enron	0.4513	0.4419	0.4438	0.5934	**0.5975**	0.5914
RAkEL	enron	0.4336	0.4297	0.4254	**0.5924**	0.5845	0.5706
CLR	genbase	0.9850	0.9840	0.9846	0.9868	**0.9874**	0.9868
HOMER	genbase	0.9794	0.9845	0.9869	0.9904	**0.9916**	0.9821
IBLR	genbase	0.9814	0.9836	0.9843	0.9863	**0.9868**	0.9862
RAkEL	genbase	0.9880	0.9881	0.9887	0.9898	**0.9904**	0.9898
CLR	llog	0.3919	0.4208	0.4098	0.5974	0.5949	**0.6476**
HOMER	llog	0.0955	0.0898	0.0941	**0.1645**	0.1512	0.1520
IBLR	llog	0.0560	0.0604	0.0606	0.0688	**0.0757**	0.0752
RAkEL	llog	0.1220	0.1219	0.1203	**0.2525**	0.2429	0.2493
CLR	mediamill	0.6112	0.6141	0.6138	**0.7650**	0.7608	0.7600
HOMER	mediamill	0.3611	0.3629	0.3646	**0.5516**	0.5379	0.5354
IBLR	mediamill	0.4084	0.4068	0.4064	**0.7386**	0.7176	0.7109
RAkEL	mediamill	0.3663	0.3677	0.3681	**0.6024**	0.5863	0.5817
CLR	slashdot	0.5188	0.5272	0.5272	**0.6537**	0.6448	0.6528
HOMER	slashdot	0.3381	0.3346	0.3456	0.5554	**0.5934**	0.5775
IBLR	slashdot	0.2881	0.2911	0.2934	**0.6385**	0.5937	0.5712
RAkEL	slashdot	0.4196	0.4308	0.4281	**0.6848**	0.6786	0.6786
CLR	tmc2007	0.5973	0.6016	0.6032	**0.7530**	0.7515	0.7512
HOMER	tmc2007	0.4740	0.4752	0.4774	**0.6941**	0.6912	0.6891
IBLR	tmc2007	0.4519	0.4522	0.4529	**0.7135**	0.7049	0.7038
RAkEL	tmc2007	0.4917	0.4958	0.4970	**0.7283**	0.7214	0.7192
CLR	yeast	0.5389	0.5388	0.5356	**0.6359**	0.6338	0.6332
HOMER	yeast	0.4604	0.4604	0.4546	**0.5475**	0.5440	0.5445
IBLR	yeast	0.4976	0.5000	0.5015	**0.7039**	0.6843	0.6767
RAkEL	yeast	0.4748	0.4715	0.4733	**0.5639**	0.5578	0.5606

Tabla 3.11: Resultados de los algoritmos de *oversampling* - Macro-FMeasure

Clasificador	**MLD**	**LP-RUS10**	**LP-RUS20**	**LP-RUS25**	**ML-RUS10**	**ML-RUS20**	**ML-RUS25**
CLR	bibtex	0.2927	0.2923	0.2974	**0.3386**	0.3358	0.3379
HOMER	bibtex	0.2180	0.2212	0.2210	**0.2970**	0.2945	0.2863
IBLR	bibtex	0.1533	0.1558	0.1550	**0.2200**	0.2176	0.2184
RAkEL	bibtex	0.2754	0.2739	0.2755	**0.3288**	0.3229	0.3236
CLR	cal500	0.3236	0.3238	0.3199	0.3202	**0.3318**	0.3221
HOMER	cal500	0.2888	0.2900	0.2841	**0.3019**	0.2967	0.2958
IBLR	cal500	**0.2756**	0.2705	0.2742	0.2700	0.2750	0.2755
RAkEL	cal500	0.2915	0.2906	0.2924	0.2966	0.2971	**0.2972**
CLR	corel16k	0.1059	0.1046	0.1075	0.1033	0.1076	**0.1080**
HOMER	corel16k	0.0861	0.0893	0.0910	**0.1363**	0.1292	0.1273
IBLR	corel16k	0.0778	0.0805	0.0804	**0.1094**	0.1055	0.1085
RAkEL	corel16k	0.0835	0.0832	0.0822	**0.1278**	0.1176	0.1192
CLR	corel5k	0.1366	0.1374	0.1395	0.1355	0.1384	**0.1431**
HOMER	corel5k	0.1418	0.1398	0.1451	**0.1896**	0.1884	0.1877
IBLR	corel5k	0.0987	0.1012	0.1008	0.1157	0.1192	**0.1235**
RAkEL	corel5k	0.1555	0.1552	0.1521	0.1784	**0.1851**	0.1850
CLR	enron	0.3819	0.3760	0.3792	**0.4220**	0.3968	0.4082
HOMER	enron	0.2957	0.3117	0.2988	**0.3740**	0.3597	0.3575
IBLR	enron	0.3088	0.3046	0.3092	**0.3580**	0.3447	0.3485
RAkEL	enron	0.3188	0.3252	0.3177	**0.3930**	0.3856	0.3880
CLR	genbase	0.9732	0.9754	0.9755	0.9800	0.9802	**0.9804**
HOMER	genbase	0.9746	0.9746	0.9754	0.9814	**0.9877**	0.9778
IBLR	genbase	0.9730	0.9737	0.9734	**0.9799**	0.9756	0.9787
RAkEL	genbase	0.9834	0.9876	0.9878	0.9890	0.9893	**0.9894**
CLR	llog	0.2241	0.1763	0.1823	**0.2508**	0.2308	0.2418
HOMER	llog	0.2020	0.2040	0.1942	0.2495	**0.2510**	0.2494
IBLR	llog	0.1662	0.1694	0.1730	0.2096	**0.2240**	0.2124
RAkEL	llog	0.2316	0.2318	0.2372	0.2921	0.2744	**0.3018**
CLR	mediamill	0.2020	0.2037	0.2073	0.2322	0.2318	**0.2336**
HOMER	mediamill	0.1565	0.1572	0.1582	**0.2422**	0.2350	0.2348
IBLR	mediamill	0.2086	0.2091	0.2106	0.2800	0.2813	**0.2834**
RAkEL	mediamill	0.1705	0.1721	0.1685	**0.2618**	0.2579	0.2556
CLR	slashdot	0.3410	0.3463	0.3602	0.4061	0.4069	**0.4203**
HOMER	slashdot	0.3298	0.3272	0.3288	0.3907	**0.3978**	0.3941
IBLR	slashdot	0.2256	0.2274	0.2284	0.2319	**0.2328**	0.2112
RAkEL	slashdot	0.3280	0.3407	0.3516	0.4002	0.4024	**0.4137**
CLR	tmc2007	0.5731	0.5782	0.5777	0.6332	**0.6440**	0.6430
HOMER	tmc2007	0.4673	0.4692	0.4727	0.6068	0.6082	**0.6114**
IBLR	tmc2007	0.3880	0.3891	0.3886	0.4740	0.4765	**0.4844**
RAkEL	tmc2007	0.4870	0.4895	0.4916	0.6138	**0.6180**	0.6162
CLR	yeast	0.4206	0.4260	0.4159	0.4537	0.4464	**0.4590**
HOMER	yeast	0.3985	0.3925	0.3963	**0.4314**	0.4134	0.4182
IBLR	yeast	0.4433	0.4460	0.4467	0.4566	0.4622	**0.4704**
RAkEL	yeast	0.4233	0.4166	0.4150	**0.4528**	0.4512	0.4507

Tabla 3.12: Base vs Mejor preprocesamiento - Accuracy

Clasificador	MLD	Base	ML-RUS 10	ML-ROS 10
CLR	bibtex	0.2316	0.2292	**0.2364**
HOMER	bibtex	**0.2714**	0.2618	0.2677
IBLR	bibtex	0.1746	0.1684	**0.1768**
RAkEL	bibtex	**0.3002**	0.2966	0.2925
RAkEL	cal500	**0.2135**	0.1708	0.2038
HOMER	cal500	**0.2496**	0.2346	0.2210
CLR	cal500	0.1787	0.1898	**0.1900**
IBLR	cal500	0.1922	0.2101	**0.2121**
CLR	corel16k	0.0456	0.0453	**0.0480**
HOMER	corel16k	**0.1138**	0.1118	0.1107
IBLR	corel16k	0.0253	0.0256	**0.0292**
RAkEL	corel16k	0.0645	0.0633	**0.0700**
RAkEL	corel5k	**0.0586**	0.0355	0.0390
HOMER	corel5k	**0.1029**	0.1016	0.0996
CLR	corel5k	**0.0360**	0.0296	0.0327
IBLR	corel5k	0.0315	0.0589	**0.0612**
RAkEL	enron	0.4010	**0.4184**	0.4068
HOMER	enron	**0.4110**	0.4085	0.4024
CLR	enron	**0.4171**	0.3005	0.3155
IBLR	enron	0.3226	**0.4034**	0.3890
RAkEL	genbase	**0.9842**	0.9716	**0.9842**
HOMER	genbase	0.9792	0.9764	**0.9834**
CLR	genbase	0.9837	0.9671	**0.9842**
IBLR	genbase	0.9790	0.9782	**0.9864**
CLR	llog	0.0456	0.0458	**0.0470**
HOMER	llog	0.1053	0.1038	**0.1105**
IBLR	llog	0.0322	0.0352	**0.0357**
RAkEL	llog	**0.1419**	0.1325	0.1324
CLR	mediamill	0.4490	0.4438	**0.4559**
HOMER	mediamill	0.4088	**0.4089**	0.4002
IBLR	mediamill	**0.4660**	0.4590	0.4644
RAkEL	mediamill	**0.4194**	0.4144	0.4114
CLR	slashdot	0.3236	0.3194	**0.3260**
HOMER	slashdot	0.3534	0.3314	**0.3550**
IBLR	slashdot	0.1269	**0.1486**	0.1343
RAkEL	slashdot	0.3452	0.3392	**0.3496**
CLR	tmc2007	0.6132	0.6020	**0.6148**
HOMER	tmc2007	**0.6029**	0.5897	0.6012
IBLR	tmc2007	**0.5322**	0.5184	0.5281
RAkEL	tmc2007	**0.6044**	0.5913	0.6022
RAkEL	yeast	0.4338	**0.4706**	0.4614
HOMER	yeast	0.4292	**0.4312**	0.4053
CLR	yeast	0.4698	**0.5150**	0.5142
IBLR	yeast	**0.5210**	0.4344	0.4101

Tabla 3.13: Base vs Mejor preprocesamiento - Micro-FM

Clasificador	**MLD**	**Base**	**ML-RUS 10**	**ML-ROS 10**
CLR	bibtex	0.3371	**0.7793**	0.7690
HOMER	bibtex	0.3568	0.3648	**0.3656**
IBLR	bibtex	0.2628	0.3494	**0.4070**
RAkEL	bibtex	0.4021	**0.5136**	0.4756
RAkEL	cal500	0.3488	**0.6258**	0.5911
HOMER	cal500	**0.3978**	0.3732	0.3512
CLR	cal500	**0.2977**	0.2777	0.2802
IBLR	cal500	0.3184	**0.4188**	0.3709
CLR	corel16k	0.0846	**0.4300**	0.4232
HOMER	corel16k	0.1866	**0.2327**	0.2128
IBLR	corel16k	0.0504	**0.3049**	0.2718
RAkEL	corel16k	0.1145	**0.3511**	0.2998
RAkEL	corel5k	0.1096	**0.4512**	0.4402
HOMER	corel5k	0.1744	**0.2086**	0.2040
CLR	corel5k	**0.0706**	0.0434	0.0512
IBLR	corel5k	0.0542	**0.3707**	0.3113
RAkEL	enron	0.5334	**0.6780**	0.6772
HOMER	enron	0.5265	**0.5482**	0.5237
CLR	enron	0.5596	0.5588	**0.5934**
IBLR	enron	0.4561	**0.6195**	0.5924
RAkEL	genbase	0.9867	0.9844	**0.9868**
HOMER	genbase	0.9820	0.9852	**0.9904**
CLR	genbase	0.9852	0.9478	**0.9863**
IBLR	genbase	0.9768	0.9875	**0.9898**
CLR	llog	0.0734	0.5863	**0.5974**
HOMER	llog	0.1491	0.1594	**0.1645**
IBLR	llog	0.0560	0.0580	**0.0688**
RAkEL	llog	0.2062	**0.2880**	0.2525
CLR	mediamill	0.5928	**0.7750**	0.7650
HOMER	mediamill	0.5493	**0.5882**	0.5516
IBLR	mediamill	0.5987	**0.7701**	0.7386
RAkEL	mediamill	0.5622	**0.6510**	0.6024
CLR	slashdot	0.4416	0.6315	**0.6537**
HOMER	slashdot	0.4429	**0.5997**	0.5554
IBLR	slashdot	0.2042	**0.6505**	0.6385
RAkEL	slashdot	0.4598	0.6675	**0.6848**
CLR	tmc2007	0.7228	0.7500	**0.7530**
HOMER	tmc2007	**0.6982**	0.6908	0.6941
IBLR	tmc2007	0.6447	**0.7221**	0.7135
RAkEL	tmc2007	0.7063	**0.7337**	0.7283
RAkEL	yeast	0.5796	**0.6516**	0.6359
HOMER	yeast	**0.5763**	0.5680	0.5475
CLR	yeast	0.6168	**0.7137**	0.7039
IBLR	yeast	**0.6502**	0.5913	0.5639

Tabla 3.14: Base vs Mejor preprocesamiento - Macro-FM

Clasificador	MLD	Base	ML-RUS 10	ML-ROS 10
CLR	bibtex	0.3342	**0.3400**	0.3386
HOMER	bibtex	**0.3042**	0.2920	0.2970
IBLR	bibtex	0.2140	0.2050	**0.2200**
RAkEL	bibtex	0.3368	**0.3384**	0.3288
RAkEL	cal500	0.2934	0.3128	**0.3202**
HOMER	cal500	**0.3316**	0.3194	0.3019
CLR	cal500	**0.3323**	0.2744	0.2700
IBLR	cal500	0.2772	**0.3028**	0.2966
CLR	corel16k	0.1003	0.1031	**0.1033**
HOMER	corel16k	**0.1363**	0.1322	**0.1363**
IBLR	corel16k	**0.1141**	0.1049	0.1094
RAkEL	corel16k	0.1277	0.1244	**0.1278**
RAkEL	corel5k	**0.1774**	0.1304	0.1355
HOMER	corel5k	**0.1916**	0.1852	0.1896
CLR	corel5k	**0.1330**	0.1092	0.1157
IBLR	corel5k	0.1059	**0.1792**	0.1784
RAkEL	enron	0.4029	0.4132	**0.4220**
HOMER	enron	0.3790	**0.3798**	0.3740
CLR	enron	**0.4198**	0.3399	0.3580
IBLR	enron	0.3458	**0.4039**	0.3930
RAkEL	genbase	**0.9890**	0.9675	0.9800
HOMER	genbase	0.9780	0.9718	**0.9814**
CLR	genbase	**0.9848**	0.9424	0.9799
IBLR	genbase	0.9655	0.9834	**0.9890**
CLR	llog	0.2330	**0.2550**	0.2508
HOMER	llog	0.2380	0.2267	**0.2495**
IBLR	llog	0.1830	0.1998	**0.2096**
RAkEL	llog	0.2824	0.2670	**0.2921**
CLR	mediamill	0.2276	0.2176	**0.2322**
HOMER	mediamill	0.2404	0.2290	**0.2422**
IBLR	mediamill	**0.2806**	0.2634	0.2800
RAkEL	mediamill	**0.2774**	0.2692	0.2618
CLR	slashdot	0.3982	0.3898	**0.4061**
HOMER	slashdot	**0.3996**	0.3766	0.3907
IBLR	slashdot	**0.2382**	0.2242	0.2319
RAkEL	slashdot	**0.4038**	0.3982	0.4002
CLR	tmc2007	0.6073	0.5954	**0.6332**
HOMER	tmc2007	0.5968	0.5855	**0.6068**
IBLR	tmc2007	0.4668	0.4406	**0.4740**
RAkEL	tmc2007	0.6015	0.5878	**0.6138**
RAkEL	yeast	0.4466	0.4483	**0.4537**
HOMER	yeast	0.4334	**0.4351**	0.4314
CLR	yeast	0.4480	**0.4597**	0.4566
IBLR	yeast	**0.4944**	0.4474	0.4528

Capítulo 4

Técnicas de remuestreo heurístico

Los algoritmos de remuestreo aleatorio, a pesar de operar de manera no informada en cuanto a las características asociadas a cada instancia y de limitarse a eliminar o clonar instancias, afectando a todos sus atributos y etiquetas, son una alternativa que, como se ha demostrado en el capítulo previo, tiene capacidad para equilibrar la distribución de las etiquetas y mejorar el rendimiento en clasificación multietiqueta. Esos resultados nos alentaron a seguir trabajando en el diseño de algoritmos de remuestreo más sofisticados, basados en técnicas heurísticas en lugar de aleatorias, así como a estudiar características específicas del desbalanceo en conjuntos de datos multietiqueta.

Este capítulo comienza introduciendo las bases en que se apoyan las propuestas descritas en las secciones siguientes. La primera de ellas es **MLSMOTE**, un método de generación de instancias sintéticas inspirado en el conocido algoritmo SMOTE, diseñado para las características específicas de los conjuntos de datos multietiqueta. La segunda es **MLeNN**, un algoritmo heurístico de eliminación de muestras asociadas a etiquetas mayoritarias, teniendo en cuenta la representación de etiquetas de los vecinos más cercanos. A continuación se describe el estudio de una característica específica de los conjuntos de datos multietiqueta, como es la concurrencia de etiquetas minoritarias y mayoritarias en la misma instancia, proponiéndose la métrica **SCUMBLE** para medir dicha casuística. Finalmente

se presenta **REMEDIAL**, una propuesta de preprocesamiento cuyo pilar central es la información de concurrencia entre etiquetas mayoritarias y minoritarias aportada por la métrica previa. Para cada propuesta se facilitan los detalles del algoritmo y se describe la experimentación llevada a cabo para validar su funcionamiento.

4.1. Introducción

Parte de los fundamentos teóricos que actúan como base de las propuestas presentadas en el presente capítulo, relativos al problema que plantea el aprendizaje a partir de conjuntos de datos no balanceados (Japkowicz and Stephen [2002] y Fernández et al. [2013]) y los distintos enfoques (López et al. [2013]) con que se ha abordado hasta ahora dicho problema, tanto en clasificación tradicional como en el campo de la clasificación multietiqueta, han sido ya descritos en el capítulo previo. Aquí se tratarán exclusivamente aquellos asociados a las características específicas de las propuestas descritas en los siguientes apartados.

4.1.1. Generación de instancias sintéticas

La generación de nuevas instancias asociadas a clases minoritarias mediante clonación, también denominada generación de instancias con reemplazo, permite a los clasificadores identificar más claramente las fronteras de decisión entre clases. No obstante la efectividad de dicho método es limitada, según se apunta en Ling and Li [1998], tendiendo a generar áreas cada vez más específicas y a término un sobreentrenamiento (*overfitting*) del clasificador. En Ha and Bunke [1997], siendo la tarea estudiada la identificación de escritura manual, se comprobó que la efectividad de los clasificadores mejoraba al introducir ciertas perturbaciones en las imágenes de los caracteres, concretamente pequeñas rotaciones e inclinaciones. Sobre la base de esta idea, en Nitesh et al. [2002] se introdujo SMOTE como un algoritmo genérico, no asociado a una aplicación específica, para la generación de instancias sintéticas asociadas a la clase minoritaria.

SMOTE recurre al uso de kNN para determinar, por cada instancia asociada a la clase minoritaria, cuáles son sus k vecinos más cercanos asociados a esa misma clase, produciendo k nuevas instancias en las que se mantiene la clase y el conjunto de características es obtenido mediante interpolación a lo largo de la línea que une la instancia de referencia con cada uno de sus vecinos. De esta manera se incrementa el número de representantes asociados a la clase minoritaria sin caer en el sobreentrenamiento, incrementando la capacidad de generalización de las fronteras de decisión y mejorando el rendimiento de los clasificadores.

Aunque originalmente SMOTE fue diseñado para problemas de desbalanceo en clasificación binaria, los propios autores indican en Nitesh et al. [2002] que podría utilizarse en problemas multiclase sencillamente especificando cuál de las clases es la que quiere tratarse como minoritaria. En el campo multietiqueta se ha propuesto utilizar el algoritmo SMOTE original de manera similar, tal y como se explicó en la Sección 3.3.1 (página 124, Giraldo-Forero et al. [2013]) con resultados dispares según el método de selección de instancias a procesar.

En la Sección 4.2 de este capítulo se propone el algoritmo MLSMOTE (*(Multilabel Synthetic Minority Over-sampling Technique*), cuya finalidad es generar instancias sintéticas tomando en consideración las características específicas de los conjuntos de datos multietiqueta. En lugar de aplicar el algoritmo original SMOTE a ciertos grupos de instancias, como proponen los autores en Giraldo-Forero et al. [2013], se ha diseñado un algoritmo a medida, no dependiente del algoritmo SMOTE, capaz de operar sobre múltiples etiquetas y de generar instancias en las que no solamente el conjunto de atributos de entrada es sintético, sino que también el conjunto de etiquetas asignado a estas nuevas muestras es generado heurísticamente.

4.1.2. La regla k-NN

La conocida como *regla k-NN* ha sido el origen de distintos procedimientos de edición de los datos (Wilson [1972]) cuyo objetivo principal ha sido reducir el nivel de ruido y hacer más nítida la frontera de separación entre clases en problemas

de clasificación. Dicha regla es sencilla y básicamente establece que la clase de una instancia dada ha de coincidir con la de sus k vecinos más cercanos.

La también conocida como regla ENN de Wilson ha sido ampliamente utilizada en clasificación tradicional. La idea básica subyacente queda reflejada en el siguiente razonamiento:

1. Se selecciona una muestra C como candidata.

2. Se buscan los k vecinos más cercanos a C. Habitualmente $k = 3$.

3. Si la clase de C difiere de la clase de la mitad o más de sus vecinos se marca C para su eliminación.

En la Sección 4.3 del presente capítulo se describirá una propuesta basada en esta regla para la eliminación de muestras asociadas a etiquetas mayoritarias.

4.1.3. Concurrencia entre etiquetas en conjuntos multietiqueta desbalanceados

Un aspecto específico de los conjuntos de datos multietiqueta, y que hasta donde sabemos aún no ha sido estudiado en la literatura, es la concurrencia entre etiquetas minoritarias y mayoritarias en conjuntos de datos multietiqueta desbalanceados. Se trata de una casuística que podría influir decisivamente en el comportamiento de los algoritmos de remuestreo, por una parte, y que podría aportar información de utilidad para abrir vías alternativas de preprocesamiento, por otro.

En la Sección 4.4 se abordará el estudio de este problema, proponiéndose una métrica para evaluarlo y mostrando experimentalmente su influencia en algunos algoritmos de remuestreo. En la Sección 4.5 se presentará un algoritmo de preprocesamiento basado en la información aportada por dicha métrica.

4.2. El algoritmo MLSMOTE

En esta sección se describe el diseño del algoritmo MLSMOTE, se detalla la experimentación llevada a cabo para validar su funcionamiento y se discuten los resultados obtenidos a partir de la misma.

Tanto el algoritmo SMOTE original como la propuesta de uso de SMOTE en el contexto multietiqueta hecha por Giraldo-Forero et al. [2013] asumen la existencia de una única clase/etiqueta minoritaria. MLSMOTE asume la existencia de múltiples etiquetas minoritarias, produciendo instancias sintéticas asociadas a todas ellas. Para ello procesará individualmente el conjunto de instancias en que aparezca cada una de estas etiquetas. Cada instancia actuará como semilla para una nueva muestra sintética. Esta necesitará tanto un nuevo conjunto de atributos como de etiquetas, indicando qué etiquetas de las que existen en la instancia de referencia y sus vecinas estarán también presentes en la muestra sintética. En consecuencia, MLSMOTE ha de resolver cuatro aspectos básicos:

- **Selección de instancias minoritarias**: El algoritmo se ha diseñado con el conocimiento previo de que la mayoría de conjuntos de datos multietiqueta presentan múltiples etiquetas minoritarias. Por tanto se precisará un criterio que determine qué etiquetas tendrán dicha condición.
- **Búsqueda de vecinos cercanos**: Una vez que se ha seleccionado una instancia asociada a una etiqueta minoritaria el algoritmo buscará los vecinos más cercanos a la misma.
- **Generación del conjunto de atributos**: Habiendo seleccionado uno de los vecinos, el conjunto de atributos de la nueva instancia se obtiene mediante técnicas de interpolación.
- **Producción del conjunto de etiquetas**: Finalmente, dada la naturaleza multietiqueta del problema que está tratándose, es necesario generar un nuevo conjunto de etiquetas para la nueva instancia.

En las siguientes subsecciones se detalla cada una de estas fases del algoritmo.

4.2.1. Selección de instancias minoritarias

La generación de instancias sintéticas comienza por la selección de la instancia minoritaria que actuará como punto de referencia. Por tanto es necesario determinar qué instancias se considerarán como minoritarias. MLSMOTE se apoya para esta tarea en las métricas *IRLbl* (Ecuación 3.1, página 127) y *MeanIR* (Ecuación 3.2).

El criterio seguido para considerar si una etiqueta l es minoritaria es que se cumpla que $IRLbl(l) > MeanIR$. En otras palabras, que la frecuencia de la etiqueta l esté por debajo de la frecuencia promedio de todas las etiquetas presentes en el conjunto de datos. De esta forma el umbral utilizado para dividir el conjunto de etiquetas en dos subconjuntos: minoritarias y mayoritarias, emerge directamente del propio conjunto de datos, en lugar de establecer un nivel o número de etiquetas *ad hoc*.

Utilizando las dos métricas citadas, en las líneas 3-6 del Algoritmo 6 se obtiene por cada etiqueta minoritaria, aquellas cuyo *IRLbl* está por encima de *MeanIR*, un paquete con todas las instancias en que aparece. Cada muestra en dicho paquete será tomada como origen para buscar sus k vecinos más cercanos, siguiendo el enfoque kNN habitual. Hay que destacar que la misma instancia podría seleccionarse varias veces, ya que en ella podrían estar presentes varias de las etiquetas minoritarias.

4.2.2. Búsqueda de los vecinos cercanos

Una vez que se ha elegido una instancia minoritaria (variable *sample* en la línea 8 del Algoritmo 6), el paso siguiente será seleccionar un conjunto con los vecinos más cercanos. El tamaño de este conjunto lo determina el parámetro k. El proceso se inicia obteniendo las distancias entre *sample* y el resto de instancias existente en *minBag*. La distancia entre dos muestras se calcula agregando las diferencias entre sus correspondientes conjuntos de características. Para atributos numéricos

se utiliza la distancia euclídea, mientras que para atributos nominales recurrimos a la métrica VDM (*Value Difference Metric*, Stanfill and Waltz [1986]).

El número de vecinos cercanos considerado, asignado al parámetro k, por defecto es 5, tal y como se recomienda en la implementación original de SMOTE de Nitesh et al. [2002]. Uno de ellos es elegido aleatoriamente (*refNeigh* en la línea 13 del Algoritmo 6), actuando como vecino de referencia.

4.2.3. Conjuntos de características y etiquetas

Al llegar a la línea 15 se han seleccionado dos muestras, *sample* y *refNeigh*. Los valores de los atributos de la instancia sintética serán obtenidos interpolando (líneas 24-31 del Algoritmo 7) a lo largo de la línea que conecta estas dos instancias, para atributos numéricos, o bien tomando en consideración todos los vecinos, para atributos nominales.

En cuanto al conjunto de etiquetas de la nueva instancia, el método usado hasta ahora en las propuestas de *oversampling* ha sido la clonación del conjunto de etiquetas de la instancia que actúa como semilla. Esta técnica ignora por completo la información relativa a correlación entre etiquetas. En lugar de construir un modelo global de correlación, como se hizo en el Capítulo 2 con el algoritmo LI-MLC, MLSMOTE intenta aprovechar la información sobre correlación que puede obtenerse de la vecindad de la muestra procesada. Se han considerado y analizado empíricamente tres posibilidades:

- **Intersección**: Se incluyen en el conjunto de etiquetas sintético solamente aquellas etiquetas que aparecen en la muestra de referencia y sus k vecinos más cercanos.

- **Unión**: Se incluyen en el conjunto de etiquetas sintético todas aquellas etiquetas que aparecen tanto en la muestra de referencia como en los k vecinos más cercanos.

- **Ranking**: Se hace un conteo del número de veces que aparece cada etiqueta en la muestra de referencia y sus k vecinos más cercanos, incluyendo en el conjunto sintético aquellas que están presentes en la mitad o más de las instancias consideradas, como es habitual en los esquemas de votación.

El último enfoque de los tres descritos resultó el más efectivo, siendo implementado en el algoritmo MLSMOTE definitivo tal y como se muestra en las líneas 35-38 del Algoritmo 7.

Para terminar, la instancia sintética es finalmente añadida al conjunto de datos original (línea 18 en Algoritmo 6), procediendo el algoritmo a procesar el resto de instancias que tienen la misma etiqueta minoritaria, primero, y luego el resto de etiquetas minoritarias. Con el objetivo de alcanzar el mayor equilibrio posible, el *IRLbl* de cada etiqueta es recalculado al inicio de cada ciclo (línea 4 en Algoritmo 6). De esta forma si una etiqueta minoritaria alcanza el *MeanIR* durante el procesamiento, automáticamente se excluirá del ciclo de generación de instancias sintéticas.

4.2.4. Pseudo-código de MLSMOTE

En el Algoritmo 6 se muestra el pseudo-código del algoritmo MLSMOTE. Las entradas son D, el conjunto de datos multietiqueta a procesar, y k, el número de vecinos a usar. La salida sería D incluyendo las muestras sintéticas generadas por MLSMOTE.

El cuerpo principal del algoritmo abarca las líneas 3 a 20, un bucle que recorre todas las etiquetas existentes en el conjunto de datos. Para cada una de ellas se obtiene su *IRLbl* y, si este está por encima del *MeanIR* del conjunto de datos, se considera como minoritaria. En este caso todas las muestras en las que aparezca dicha etiqueta serán tomadas como muestras semilla, localizando sus vecinos más cercanos y generando una instancia sintética.

El Algoritmo 7 muestra la función que se encarga de producir las nuevas instancias. Esta función precisa como entradas la muestra que actúa como semilla,

Algorithm 6 Pseudo-código del algoritmo MLSMOTE.

Inputs:
D ▷ MLD a procesar
k ▷ Número de vecinos más cercanos

```
1: L ← labelsInDataset(D)                          ▷ Conjunto completo de etiquetas
2: MeanIR ← calculateMeanIR(D, L)
3: for each label in L do
4:     IRLbl_label ← calculateIRperLabel(D, label)
5:     if IRLbl_label > MeanIR then
6:         ▷ Paquetes de muestras con etiquetas minoritarias
7:         minBag ← getAllInstancesOfLabel(label)
8:         for each sample in minBag do
9:             distances ← calcDistance(sample, minBag)
10:            sortSmallerToLargest(distances)
11:            ▷ Selección del conjunto de vecinos cercanos
12:            neighbors ← getHeadItems(distances, k)
13:            refNeigh ← getRandNeighbor(neighbors)
14:            ▷ Generación del conjunto de características y etiquetas
15:            synthSmpl ← newSample(sample,
16:                                  refNeigh, neighbors)
17:            D = D + synthSmpl
18:        end for
19:    end if
20: end for
```

sus k vecinos más cercanos y un vecino aleatorio (uno de los k anteriores) que será tomado como referencia para interpolar los valores de los atributos. Como puede apreciarse, la función está dividida en dos partes. La primera (líneas 24-33) produce los valores de los atributos para la muestra sintética, mientras que la segunda (líneas 35-38) genera el conjunto de etiquetas sintético.

4.2.5. Configuración de la experimentación

A fin de determinar las bondades del algoritmo propuesto, se ha llevado a cabo una extensa experimentación que se estructura en dos partes. Ambas utilizan el mismo grupo de conjuntos de datos. Estas fases nos han servido para responder

Algorithm 7 Función: Generación de nuevas instancias sintéticas.

```
21: function NEWSAMPLE(sample, refNeigh, neighbors)
22:     synthSmpl ← new Sample                          ▷ Nueva instancia vacía
23:     ▷ Asignación del conjunto de atributos
24:     for each feat in synthSmpl do
25:         if typeOf(feat) is numeric then
26:             diff ← refNeigh.feat - sample.feat
27:             offset ← diff * randInInterval(0,1)
28:             value ← sample.feat + offset
29:         else
30:             value ← mostFreqVal(neighbors,feat)
31:         end if
32:         syntSmpl.feat ← value
33:     end for
34:     ▷ Asignación del conjunto de etiquetas
35:     lblCounts ← counts(sample.labels)
36:     lblCounts + ← counts(neighbors.labels)
37:     labels ← lblCounts > (k+1) / 2
38:     synthSmpl.labels ← labels
39:     return synthSmpl
40: end function
```

a dos cuestiones esenciales: ¿Es MLSMOTE capaz de mejorar los resultados de clasificación originales obtenidos con un algoritmo de clasificación multietiqueta? y ¿Cuál es su rendimiento frente a otras propuestas ya publicadas? Estos dos pasos se han estructurado de la siguiente forma:

- En primer lugar se ha verificado empíricamente cómo influye MLSMOTE en los niveles de desbalanceo (Sección 4.2.7) de los conjuntos de datos y cómo afecta a los resultados en clasificación (Sección 4.2.8) obtenidos por diferentes clasificadores. Los resultados de clasificación tras aplicar el preprocesamiento se comparan con los obtenidos a partir de los mismos conjuntos de datos sin preprocesar, analizándose cómo MLSMOTE influye en cada caso.

- En segundo lugar, en la Sección 4.2.9 los resultados obtenidos con MLSMOTE se comparan con los generados por otros algoritmos de *oversampling*, como LP-ROS, ML-ROS y SmoteUG (Giraldo-Forero et al. [2013]), todos ellos descritos en el capítulo previo, así como contra los devueltos por los algoritmos BR-IRUS (Tahir et al. [2012b]) y EML (Tahir et al. [2012a]) mencionados en la Sección 3.3.1 del capítulo anterior, que están específicamente diseñados para abordar el problema del desbalanceo.

En las siguientes subsecciones se detallan los conjuntos de datos empleados, algoritmos de clasificación usados, métricas de evaluación aplicadas y los resultados obtenidos en cada fase.

4.2.6. Conjuntos de datos, algoritmos y métricas

La eficacia de un método de remuestreo como MLSMOTE, que genera instancias sintéticas teniendo en cuenta características y etiquetas de otras muestras, debería verse muy influida por las características de los conjuntos de datos usados. Por ello resulta esencial probar el algoritmo propuesto con conjuntos que tengan diferentes cualidades. En este caso se han usado los MLD más utilizados en varias de las publicaciones más relevantes del campo multietiqueta. La Tabla 4.1 muestra sus características, incluyendo las medidas de desbalanceo.

Tabla 4.1: Características de los MLD usados en la experimentación.

MLD	Card	Muestras	Número de Atributos	Etiquetas	Ratio de desbalanceo Máximo	Medio
bibtex	2.402	7395	1836	159	20.43	12.50
cal500	26.044	502	68	174	88.80	20.58
corel5k	3.522	5000	499	374	1120.00	189.57
corel16k	2.867	13766	500	161	126.80	34.16
emotions	1.869	593	72	6	1.78	1.48
enron	3.378	1702	753	53	913.00	73.95
genbase	1.252	662	1186	27	171.00	37.31
mediamill	4.376	43907	120	101	1092.55	256.40
medical	1.245	978	1449	45	266.00	89.50
scene	1.074	2407	294	6	1.46	1.25
slashdot	1.181	3782	1079	22	194.67	19.46
tmc2007	2.158	28596	49060	22	41.98	17.13

Cada uno de estos MLD ha sido preprocesado con MLSMOTE, LP-ROS, ML-ROS y SmoteUG, y todos ellos han sido dados como entrada a cinco algoritmos de clasificación multietiqueta, usando un esquema de validación 2×5. Para determinar el efecto de reequilibrado de la redistribución de etiquetas producido por MLSMOTE se han usado cinco clasificadores distintos. Estos son BR (Boutell et al. [2004]), IBLR-ML (Cheng and Hüllermeier [2009]), HOMER (Tsoumakas et al. [2008]), RAkEL (Tsoumakas and Vlahavas [2007]) y CLR (Fürnkranz et al. [2008]). Donde ha sido necesario un clasificador subyacente se ha usado C4.5, empleando en todos los casos parámetros por defecto. El objetivo es analizar el impacto que tiene MLSMOTE sobre el método de transformación más utilizado (BR), un clasificador multietiqueta basado en instancias (IBLR-ML) y tres clasificadores más considerados estado del arte. Para cada combinación algoritmo-MLD se han obtenido los promedios de resultados.

Las salidas predichas para cada combinación clasificador-MLD, siempre sobre particiones de test, han sido evaluadas mediante dos métricas de rendimiento. En el contexto de este estudio, asumiendo que ciertas etiquetas son minoritarias frente a otras más frecuentes, las métricas de tipo *label-based* son las más adecuadas. Estas fueron descritas en la Sección 1.6.2 del primer capítulo. Como se indica en Tang et al. [2009], las métricas de tipo *macro-averaged* son más sensibles al rendimiento de las etiquetas minoritarias que las de tipo *micro-averaging*.

Prácticamente todas las métricas de evaluación, incluyendo *Accuracy*, *Precision* y *Recall*, pueden ser calculadas siguiendo los dos enfoques citados. En este estudio hemos usado *F-measure*, la media armónica de *Precision* y *Recall*. Por tanto se obtienen dos medidas para cada evaluación: *MacroFM* y *MicroFM*.

La significación estadística de los resultados obtenidos en cada fase se determina aplicando los tests estadísticos apropiados. Podemos encontrarnos con dos situaciones distintas:

- Cuando el análisis implica resultados de dos fuentes, por ejemplo resultados antes y después de aplicar el preprocesamiento, se utiliza el test de

signos no paramétrico de Wilcoxon (Sheskin [2003]). Este es análogo al test paramétrico T pareado.

- Para comparaciones múltiples el análisis estadístico se efectúa en dos pasos. Primero se usa el test de Friedman para obtener un ranking de los métodos y dilucidar si hay alguna diferencia estadísticamente significativa entre ellos. En segundo lugar, se efectúa una comparación múltiple mediante el procedimiento FDR de Benjamini y Hochberg Benjamini and Yekutieli [2001]. Este test es un procedimiento de tipo *step-up* para controlar la ratio de falsos positivos en escenarios de comparaciones múltiples.

Estos tests fueron ejecutados mediante la función `wilcoxsign_test` y la función `pairwise.table` de los paquetes R `coin` y `stats`, obteniéndose *p-values* exactos.

4.2.7. Influencia de MLSMOTE en el nivel de desbalanceo

Tras aplicar MLSMOTE a los conjuntos de datos se reevaluaron los niveles de desbalanceo, a fin de analizar cómo influyó el preprocesamiento en la distribución de las etiquetas. La Tabla 4.2 muestra para cada conjunto de datos los valores de *MaxIR* y *MeanIR* antes y después de aplicar MLSMOTE, así como el porcentaje de cambio para dichas medidas en cada caso. Estos valores han sido obtenidos a partir de las particiones de entrenamiento usadas en la experimentación, dado que MLSMOTE únicamente se aplica a las particiones de entrenamiento, mientras que los datos de desbalanceo mostrados anteriormente en la Tabla 4.1 corresponden a los MLD completos.

Como puede apreciarse en la Tabla 4.2, el *MaxIR* se ha incrementado ligeramente (+0.29 % a +3.04 %) en cuatro de los MLD, mientras que el resto han experimentado una reducción notable. Esto implica que la ratio entre la etiqueta más frecuente y la menos frecuente se ha mejorado en casi todos los casos. El *MeanIR*, que denota el desbalanceo promedio para todas las etiquetas, ha descendido en todos los MLD con la única excepción de `emotions`. Este conjunto de

Tabla 4.2: Cambios en los niveles de desbalanceo tras preprocesar los MLD

	MaxIR			MeanIR		
MLD	**Antes**	**Después**	$\%\Delta$	**Antes**	**Después**	$\%\Delta$
bibtex	22.81	15.66	-31.32	12.54	12.08	-3.68
cal500	133.19	118.74	-10.85	21.27	20.90	-1.78
corel5k	896.00	923.20	3.04	168.78	142.16	-15.77
corel16k	134.73	69.55	-48.38	34.32	25.67	-25.22
emotions	1.78	1.71	-4.21	1.48	1.52	2.80
enron	657.05	674.05	2.59	72.77	62.58	-14.01
genbase	136.80	137.20	0.29	32.41	28.95	-10.68
mediamill	1122.32	564.55	-49.70	257.46	159.15	-38.18
medical	212.80	214.90	0.99	72.17	69.56	-3.62
scene	1.46	1.25	-14.44	1.25	1.20	-4.21
slashdot	202.73	99.76	-50.79	20.03	11.24	-43.88
tmc2007	42.01	23.50	-44.06	17.14	12.34	-28.00

datos en realidad no está desbalanceado, tal y como denota su *MeanIR* original. Para los MLD más desbalanceados la mejora oscila entre el -14 % y -43 %.

En términos generales, de estos resultados puede inferirse que MLSMOTE produce una mejora en los niveles de desbalanceo. No obstante, el cambio en el nivel de desbalanceo no implica necesariamente mejores resultados en clasificación. El factor decisivo para obtener mejores predicciones estribará en cómo estas nuevas instancias influirán en el modelo construido por el algoritmo de clasificación. Este es el hecho analizado en la siguiente sección.

4.2.8. Comparación de MLSMOTE con resultados base

El paso siguiente será el análisis de los resultados producidos por los algoritmos de clasificación multietiqueta antes y después de aplicar MLSMOTE a los conjuntos de datos. Por tanto tendremos dos conjuntos de resultados, uno producido por los clasificadores a partir de los MLD sin remuestreo y otro obtenido de los mismos clasificadores sobre los MLD tras ser procesados por MLSMOTE. Estos resultados son los mostrados en la Tabla 4.3 y Tabla 4.4.

Tabla 4.3: Resultados antes y después de aplicar MLSMOTE - MacroFM

	BR		CLR		HOMER		IBLR		RAkEL	
MLD	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**
bibtex	0.3368	**0.3456**	0.3342	**0.3429**	**0.3042**	0.2989	0.2140	**0.2479**	0.3368	**0.3456**
cal500	0.2934	**0.3124**	0.3323	**0.3474**	**0.3316**	0.3139	0.2772	**0.2783**	0.2934	**0.3124**
corel16k	0.1336	**0.1347**	0.1003	**0.1040**	0.1363	**0.1408**	**0.1141**	0.0971	0.1277	**0.1288**
corel5k	0.1774	**0.1790**	0.1330	0.1330	0.1916	**0.1923**	0.1059	**0.1130**	0.1774	**0.1790**
emotions	0.5712	**0.5841**	0.5982	**0.6107**	0.5642	**0.5733**	0.6487	**0.6511**	0.5712	**0.5841**
enron	**0.4029**	0.3936	0.4198	**0.4272**	**0.3790**	0.3740	0.3458	**0.3580**	**0.4029**	0.3936
genbase	0.9890	**0.9895**	0.9848	**0.9853**	0.9780	**0.9839**	0.9655	**0.9688**	0.9890	**0.9895**
mediamill	**0.2774**	0.2737	0.2276	**0.2308**	0.2404	**0.2424**	0.2806	**0.2862**	**0.2774**	0.2737
medical	0.8166	0.8166	0.7942	0.7942	**0.7942**	0.7786	0.6404	0.6404	0.8166	0.8166
scene	0.6314	**0.6328**	0.6400	**0.6407**	0.6113	**0.6212**	0.7427	**0.7456**	0.6314	**0.6328**
slashdot	0.4038	**0.4044**	0.3982	0.3982	**0.3996**	0.3885	**0.2382**	0.1852	0.4038	**0.4044**
tmc2007	0.6015	**0.6165**	0.6073	**0.6242**	0.5968	**0.6003**	0.4668	**0.4924**	0.6015	**0.6165**

Tabla 4.4: Resultados antes y después de aplicar MLSMOTE - MicroFM

	BR		CLR		HOMER		IBLR		RAkEL	
MLD	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**
bibtex	0.4021	**0.4097**	0.3371	**0.3484**	**0.3568**	0.3546	0.2628	**0.2701**	0.4021	**0.4097**
cal500	0.3488	**0.3662**	0.2977	**0.3619**	**0.3978**	0.3744	0.3184	**0.3388**	0.3488	**0.3662**
corel16k	0.1156	0.1156	**0.0846**	0.0832	0.1866	**0.1878**	**0.0504**	0.0484	0.1145	0.1145
corel5k	0.1096	**0.1105**	0.0706	**0.0707**	**0.1744**	0.1702	**0.0542**	0.0535	0.1096	**0.1102**
emotions	0.5845	**0.5958**	0.6072	**0.6174**	0.5766	**0.5818**	0.6730	**0.6756**	0.5845	**0.5958**
enron	**0.5334**	0.5324	0.5596	**0.5606**	**0.5265**	0.5199	0.4561	**0.4660**	**0.5334**	0.5324
genbase	0.9867	**0.9873**	0.9852	**0.9858**	0.9820	**0.9827**	0.9768	**0.9771**	0.9867	**0.9873**
mediamill	**0.5622**	0.5618	0.5928	**0.5938**	**0.5493**	0.5482	0.5987	**0.6000**	**0.5622**	0.5618
medical	0.8006	0.8006	0.7965	0.7965	**0.7994**	0.7955	0.6324	0.6324	0.8006	0.8006
scene	0.6190	**0.6231**	0.6254	**0.6275**	0.6010	**0.6132**	0.7366	**0.7398**	0.6190	**0.6231**
slashdot	0.4598	**0.5339**	0.4416	**0.4449**	0.4429	**0.4436**	**0.2042**	0.1470	0.4598	**0.5339**
tmc2007	0.7063	**0.7071**	0.7228	**0.7248**	**0.6982**	0.6956	**0.6447**	0.6378	0.7063	**0.7071**

A fin de evaluar la significación estadística de estos resultados, se ha aplicado el test estadístico de Wilcoxon fijando como hipótesis alternativa que los valores para *MacroFM*/*MicroFM* son más altos tras el preprocesamiento. La Tabla 4.5 muestra las salidas *z-score* y *p-value* para estos tests. De este análisis pueden deducirse los siguientes hechos:

Tabla 4.5: Tests de Wilcoxon analizando diferencias tras aplicar MLSMOTE

	MacroFM		**MicroFM**	
	z-score	**p-value**	**z-score**	**p-value**
BR	-1.76640	0.04199	-2.16562	0.01465
CLR	-2.85499	0.00195	-2.55246	0.00342
HOMER	0.39223	0.66138	0.94136	0.83032
IBLR	-1.37387	0.09473	-0.82432	0.22314
RAkEL	-1.76640	0.04199	-2.16562	0.01465

- Los *p-values* obtenidos denotan que los algoritmos CLR, BR y RAkEL, así como IBLR con la métrica *MacroFM*, ven mejoras estadísticamente significativas en sus resultados tras aplicar MLSMOTE. CLR es el clasificador más beneficiado (tiene los valores más bajos para *p-value* y *z-score*). Como puede apreciarse en la Tabla 4.3 (MacroFM), para CLR los resultados han mejorado en todos los casos, mientras que en la Tabla 4.4 (MicroFM) solamente hay un caso en el que no se mejora. El comportamiento de BR y RAkEL tras aplicar MLSMOTE es muy parecido, estando todos los *p-value* por debajo del umbral del 0.1.

- A pesar de que el comportamiento de IBLR al ser evaluado con *MicroFM* refleja una ligera mejora, esta difícilmente puede ser considerada estadísticamente significativa fijando el umbral para *p-value* a los habituales 0.1 o 0.05. No obstante, este enfoque blanco o negro al interpretar los *p-value* puede ampliarse para dar cabida a tonos de grises. Que no exista significación estadística no implica equivalencia. MLSMOTE tiene un buen rendimiento con ambas métricas en el caso de IBLR. De hecho, los resultados de clasificación para *MacroFM* han mejorado en 10 de los 12 MLD y en *MicroFM* en 8 de 12.

- Finalmente, según el test de Wilcoxon, HOMER no está obteniendo beneficio alguno del preprocesamiento. A pesar de que los resultados con *MacroFM* reflejan mejoras en 7 de los 12 MLD, para *MicroFM* la proporción es la inversa.

Globalmente, esta primera fase de la experimentación determina que MLSMOTE es capaz de producir una mejora general en los resultados de clasificación cuando se usa conjuntamente con CLR, RAkEL, BR y, en menor medida, IBLR. Incluso a pesar de no existir significación estadística cuando se evalúa con *MicroFM*, MLSMOTE en general tiene una influencia positiva en el funcionamiento de IBLR. Finalmente, no es recomendable usar MLSMOTE conjuntamente con HOMER, dado que el preprocesamiento tiende a deteriorar sus resultados.

Profundizando en el comportamiento de los clasificadores, IBLR y HOMER comparten una característica común: ambos aprovechan información local para hacer su trabajo. IBLR recurre a los vecinos más cercanos para hacer su predicción, mientras que HOMER depende de la misma técnica a la hora de agrupar las instancias y definir los subconjuntos de etiquetas. De hecho, HOMER y RAkEL son algoritmos muy similares, dado que ambos entrenan varios clasificadores multiclase usando subconjuntos de etiquetas. La diferencia principal estriba en el método usado para construir esos subconjuntos de etiquetas. Mientras que HOMER usa información local para realizar esta tarea, RAkEL lo hace de manera aleatoria. Del análisis de estos resultados puede concluirse que MLSMOTE tiene una influencia más positiva sobre clasificadores como BR, CLR y RAkEL, cuyo comportamiento no está sesgado por la selección de solo unas cuantas instancias cercanas.

4.2.9. MLSMOTE frente a otras propuestas

El objetivo de la segunda fase experimental es comparar el rendimiento de MLSMOTE frente a otras propuestas ya publicadas para afrontar el problema del desbalanceo en clasificación multietiqueta. En la Sección 3.3.1 del capítulo previo

se introdujeron los algoritmos EML y BR-IRUS, dos propuestas basadas en el enfoque *ensembles* para combatir el desbalanceo. En la misma sección también se mencionó el algoritmo de *oversampling* SmoteUG. Asimismo, en dicho capítulo se presentaron dos algoritmos propios de sobremuestreo: LP-ROS y ML-ROS. Estos tres últimos generan nuevas instancias, pero siguiendo diferentes estrategias. La Tabla 4.6 resume las características de estos métodos, indicando cuántas etiquetas minoritarias son tenidas en cuenta y cómo se generan tanto las características como las etiquetas de las nuevas instancias.

Tabla 4.6: Métodos de *oversampling* a comparar

Algoritmo	**# Etq. minoritarias**	**Atributos**	**Labelset**
LP-ROS	NA	Clonados	Clonado
ML-ROS	Varias	Clonados	Clonado
SmoteUG	1	Sintéticos	Clonado
MLSMOTE	Varios	Sintéticos	Sintético

Tanto EML como BR-IRUS son capaces de producir predicciones de clasificación por sí mismos, pero MLSMOTE, ML-ROS, LP-ROS y SmoteUG no. Dado que son algoritmos de remuestreo, la salida que generan ha de ser facilitada a un clasificador multietiqueta existente. Por esta razón, el primer paso será determinar qué clasificadores presentan un mejor comportamiento para cada método de preprocesamiento. Tras esto se compararán los resultados de clasificación generados por todas las configuraciones.

Selección del mejor clasificador para cada método de remuestreo

Para saber cómo se comportan los algoritmos de clasificación con cada modelo de remuestreo, los resultados producidos por todos ellos tras preprocesar los conjuntos de datos fueron ordenados en un ranking mediante el test estadístico de Friedman, obteniendo el ranking promedio para cada algoritmo (véanse la Tabla 4.7 y la Tabla 4.8). Se han incluido también los resultados de los mismos clasificadores antes de preprocesar los datos.

De la observación de estas tablas pueden derivarse las siguientes conclusiones:

Tabla 4.7: Ranking promedio para cada clasificador antes y después de aplicar remuestreo (MacroFMeasure)

Sin remuestreo		**MLSMOTE**		**ML-ROS**		**LP-ROS**		**SmoteUG**	
Rank	**Clasif.**	**Rank**	**Clasif.**	**Rank**	**Clasif.**	**Rank**	**Clasif.**	**Rank**	**Clasif.**
2.3750	BR	2.4167	BR	2.5000	CLR	1.7500	CLR	2.2917	BR
2.4583	RAkEL	2.4167	RAkEL	2.5417	BR	2.7083	BR	2.4583	RAkEL
2.8750	CLR	2.6667	CLR	2.6250	RAkEL	2.7917	RAkEL	2.5833	CLR
3.3750	HOMER	3.5000	HOMER	3.4167	HOMER	3.8333	HOMER	3.5000	HOMER
3.9167	IBLR	4.0000	IBLR	3.9167	IBLR	3.9167	IBLR	4.1667	IBLR

Tabla 4.8: Ranking promedio para cada clasificador antes y después de aplicar remuestreo (MicroFMeasure)

Sin remuestreo		**MLSMOTE**		**ML-ROS**		**LP-ROS**		**SmoteUG**	
Rank	**Clasif.**	**Rank**	**Clasif.**	**Rank**	**Clasif.**	**Rank**	**Clasif.**	**Rank**	**Clasif.**
2.3750	BR	2.3750	BR	2.3750	BR	2.4167	CLR	2.3750	BR
2.4583	RAkEL	2.4583	RAkEL	2.4583	RAkEL	2.5833	BR	2.4583	RAkEL
3.0000	CLR	2.7500	CLR	2.7083	CLR	2.5833	RAkEL	2.8333	CLR
3.2500	HOMER	3.4167	HOMER	3.3750	HOMER	3.5833	HOMER	3.4167	HOMER
3.9167	IBLR	4.0000	IBLR	4.0833	IBLR	3.8333	IBLR	3.9167	IBLR

- BR es el clasificador que consigue mejores resultados tanto antes como después de aplicar varios de los métodos de remuestreo, incluyendo SmoteUG y MLSMOTE. No es sorprendente, ya que como se indica en Luaces et al. [2012], BR es un enfoque no tan simple para afrontar la clasificación multietiqueta, siendo capaz de producir buenos resultados y ser competitivo frente a otras propuestas más elaboradas.

- El clasificador con el mejor comportamiento para los algoritmos de remuestreo LP-ROS y ML-ROS es CLR, clasificador que queda por delante de BR para las dos métricas de evaluación con la excepción de ML-ROS/MicroFM.

- HOMER e IBLR son los dos peores clasificadores en todos los casos, tanto antes como después de preprocesar los conjuntos de datos.

Comparación de resultados de clasificación

La fase final de la experimentación compara las cinco propuestas anteriores con MLSMOTE, con el objetivo de dilucidar cuál de estos enfoques de *oversampling* funciona mejor. EML y BR-IRUS son clasificadores por derecho propio, pero todos los demás, incluyendo MLSMOTE, necesitan un clasificador con el que trabajar. Se ha elegido aquel que funciona mejor con cada preprocesamiento a partir del ranking descrito en la subsección previa. Los resultados de clasificación producidos por estos seis métodos son los presentados en la Tabla 4.9 y la Tabla 4.10. Los mejores valores para cada métrica han sido destacados en negrita.

Tabla 4.9: Resultados de MLSMOTE vs otras propuestas (MacroFM)

MLD	**MLSMOTE + BR**	**ML-ROS + CLR**	**LP-ROS + CLR**	**SmoteUG + BR**	**EML**	**BR-IRUS**
bibtex	**0.3456**	0.3386	0.2927	0.3375	0.1265	0.0462
cal500	0.3124	0.3202	**0.3236**	0.2934	0.1291	0.2178
corel16k	**0.1347**	0.1033	0.1059	**0.1347**	0.0181	0.0349
corel5k	**0.1790**	0.1355	0.1366	0.1774	0.0133	0.0226
emotions	0.5841	0.6062	0.5684	0.5724	**0.6893**	0.5160
enron	0.3936	**0.4220**	0.3819	0.4029	0.1241	0.1028
genbase	**0.9895**	0.9800	0.9732	0.9890	0.7608	0.8664
mediamill	0.2737	0.2322	0.2020	**0.2861**	0.1728	0.0556
medical	**0.8166**	0.7865	0.6920	**0.8166**	0.2770	0.3958
scene	0.6328	0.6387	0.5930	0.6314	**0.7546**	0.3673
slashdot	0.4044	**0.4061**	0.3410	0.4044	0.3560	0.1142
tmc2007	0.6165	**0.6332**	0.5731	0.5933	0.5122	0.1370

A primera vista ya puede observarse que BR-IRUS y LP-ROS son claramente los métodos que peores resultados han producido. También puede apreciarse que MLSMOTE consigue el mayor número de primeras posiciones (11), seguido por EML (6) y SmoteUG y ML-ROS (5). En cuanto al análisis estadístico, una vez que el test de Friedman confirma la existencia de algunas diferencias estadísticamente significativas entre los algoritmos, se han efectuado comparaciones por pares usando el procedimiento de Benjamini y Hochberg. El método mejor posicionado en el ranking, que es MLSMOTE para las dos métricas consideradas, es

Tabla 4.10: Resultados de MLSMOTE vs otras propuestas (MicroFM)

MLD	**MLSMOTE + BR**	**ML-ROS + CLR**	**LP-ROS + CLR**	**SmoteUG + BR**	**EML**	**BR-IRUS**
bibtex	**0.4097**	0.3457	0.2972	0.4024	0.2888	0.0331
cal500	0.3662	0.3348	0.3510	0.3488	**0.4106**	0.2238
corel16k	**0.1156**	0.0894	0.1060	**0.1156**	0.0544	0.0367
corel5k	**0.1105**	0.0764	0.0804	0.1096	0.0712	0.0205
emotions	0.5958	0.6152	0.5755	0.5860	**0.7015**	0.5178
enron	0.5324	**0.5552**	0.5038	0.5334	0.5542	0.1030
genbase	**0.9873**	0.9854	0.9782	0.9867	0.9854	0.6576
mediamill	0.5618	0.6006	0.5524	0.5686	**0.6277**	0.0633
medical	**0.8006**	0.7866	0.6970	**0.8006**	0.7581	0.1392
scene	0.6231	0.6240	0.5784	0.6190	**0.7468**	0.3658
slashdot	**0.5339**	0.4456	0.3232	0.4644	0.4942	0.1062
tmc2007	0.7071	**0.7250**	0.6298	0.7046	0.7065	0.1456

tomado como método de control. Los rankings facilitados por el test de Friedman y los *p-value* correspondientes son los mostrados en la Tabla 4.11 y Tabla 4.12.

Tabla 4.11: Ranking promedio del test de Friedman y *p-values* (MacroFM)

Pos	**Algoritmo**	**Rank**	***p-value***
1	MLSMOTE + BR	2.04167	**
2	ML-ROS + CLR	2.33333	0.24452
3	SmoteUG + BR	2.54167	0.11719
4	LP-ROS + CLR	3.83333	0.00146
5	HetEnsem	4.66667	0.01968
6	BR-IRUS	5.58333	0.00092

4.2.10. Discusión de los resultados

Como puede verse en la Tabla 4.11 y Tabla 4.12, MLSMOTE funciona claramente mejor que las otras cinco propuestas, alcanzando siempre la mejor posición en el ranking. Puede concluirse que existe una mejora estadísticamente significativa de MLSMOTE con respecto a los algoritmos BR-IRUS, LP-ROS y SmoteUG. MLSMOTE también supera a EML con diferencias con significación estadísti-

Tabla 4.12: Ranking promedio del test de Friedman y *p-values* (MicroFM)

Pos	**Algoritmo**	**Rank**	***p-value***
1	MLSMOTE + BR	2.08333	**
2	HetEnsem	2.79167	0.86243
3	SmoteUG + BR	2.83333	0.05310
4	ML-ROS + CLR	2.87500	0.28280
5	LP-ROS + CLR	4.41667	0.00061
6	BR-IRUS	6.00000	0.00061

ca con la métrica *MacroFM*. Con la métrica *MicroFM* MLSMOTE queda en el ranking por encima de EML, pero sin que las diferencias se consideren estadísticamente significativas. A pesar de los *p-value* correspondientes a ML-ROS, la posición en el ranking y el número de mejores valores indican que, en promedio, MLSMOTE es también superior a dicho algoritmo.

Hay que destacar que la métrica de evaluación de rendimiento en clasificación *MicroFM* se ve muy influida por la clasificación correcta de etiquetas mayoritarias. *MacroFM*, por el contrario, es más sensible a las minoritarias. En consecuencia, el anterior análisis estadístico nos permiten concluir que MLSMOTE beneficia la clasificación de etiquetas minoritarias en mayor medida que EML, a pesar de que globalmente los resultados generados por EML son muy buenos. Este hecho puede confirmarse en la Tabla 4.9 y Tabla 4.10. EML obtiene los mejores valores con conjuntos de datos como `cal500`, `emotions` y `scene`, cuya característica común es su bajo *MeanIR*, mientras que MLSMOTE funciona mejor con aquellos MLD que tienen altos niveles de desbalanceo, como `corel5k`, `corel16k` y `medical`.

Otro factor a tener en cuenta son los recursos necesarios para ejecutar cada una de las soluciones, específicamente memoria y tiempo de ejecución. Como se indicó en el capítulo previo al introducirlos, BR-IRUS y EML tienen que entrenar múltiples clasificadores, usando todos o parte de los datos de entrenamiento. En consecuencia la memoria necesaria se incrementa, al igual que el tiempo necesario para construir los clasificadores. En contraposición, los resultados de MLSMOTE

se obtienen con un solo algoritmo de clasificación multietiqueta, con una sola fase de entrenamiento.

Globalmente, MLSMOTE es la opción más potente a la hora de afrontar el problema de la clasificación multietiqueta desbalanceada. A pesar de la buena posición de EML cuando se utiliza *MicroFM* (la segunda mejor), en la evaluación con *MacroFM* cae hasta la quinta posición. Algo parecido ocurre con ML-ROS, que alcanza la segunda mejor posición cuando se usa *MacroFM* cayendo a la cuarta con *MicroFM*. MLSMOTE obtiene en ambas métricas la mejor posición.

En general, la solución basada en MLSMOTE aporta mejores resultados de clasificación que las demás propuestas al tiempo que usa menos recursos.

4.3. El algoritmo MLeNN

Los algoritmos de *undersampling* generalmente tienen peor rendimiento que los de *oversampling* (véase García et al. [2012]), dado que provocan una pérdida de información al eliminar instancias. Esta pérdida es aún mayor en el caso de los conjuntos de datos multietiqueta, ya que cada muestra está asociada no a una sola clase sino a un conjunto de ellas. En consecuencia, resulta crítico para un algoritmo de este tipo elegir de manera adecuada las muestras a eliminar.

Adaptar la regla ENN (Sheskin [2003]) para trabajar con conjuntos de datos multietiqueta demanda buscar solución a dos aspectos clave: cómo se seleccionan las instancias candidatas y cómo se determina la diferencia de clase entre el candidato y sus vecinos. El algoritmo propuesto en este trabajo, denominado MLeNN, resuelve estos problemas primero limitando las muestras que pueden actuar como candidatas a aquellas en las que no aparece ninguna etiqueta minoritaria, y en segundo lugar definiendo una métrica de distancia para saber cuál es la diferencia entre cualquier par de *labelsets*.

A diferencia de LP-RUS y ML-RUS, los dos algoritmos de *undersampling* introducidos en el capítulo previo con un comportamiento aleatorio, MLeNN toma

las muestras a eliminar utilizando una heurística basada en los dos siguientes pilares:

1. Ninguna de las etiquetas minoritarias puede aparecer en la instancia tomada como candidata a ser eliminada.

2. El conjunto de etiquetas de la instancia candidata ha de ser diferente al de los conjuntos de sus vecinos.

Esta segunda condición está basada en la regla ENN, pero adaptada al escenario multietiqueta tal y como se explica a continuación. El Algoritmo 8 muestra el pseudo-código de MLeNN. Las métricas usadas y los detalles de implementación se facilitan en las siguientes subsecciones.

Algorithm 8 Pseudo-código del algoritmo MLeNN.

Inputs: <Dataset> D, <Threshold> HT, <NumNeighbors> NN
Outputs: Conjunto de datos preprocesado

```
1: for each sample in D do
2:     for each label in getLabelset(D) do
3:         if IRLbl(label) > MeanIR then
4:             Jump to next sample     ▷ Preservar instancia con etiquetas minoritarias
5:         end if
6:     end for
7:     numDifferences ← 0
8:     for each neighbor in nearestNeighbors(sample, NN) do
9:         if adjustedHammingDist(sample, neighbor) > HT then
10:            numDifferences ← numDifferences+1
11:        end if
12:    end for
13:    if numDifferences≥NN/2 then
14:        markForRemoving(sample)
15:    end if
16: end for
17: deleteAllMarkedSamples(D)
```

4.3.1. Selección de candidatos

A fin de poder determinar qué muestras actuarán como candidatas a ser eliminadas se necesita una forma de saber qué etiquetas son minoritarias. Aquellas instancias en las que aparezcan etiquetas minoritarias nunca serán candidatas, evitando así que alguna de las pocas muestras que representan a una etiqueta minoritaria pueda perderse.

Para completar esta tarea MLeNN se apoya en las medidas *IRLbl* y *MeanIR* propuestas en la Sección 3.4 del capítulo previo. MLeNN analiza todas las muestras del conjunto de datos, tomando como candidatas aquellas cuyo *labelset* no contiene ninguna etiqueta para la que *IRLbl(l)>MeanIR*. De esta forma se preservan todas las instancias que contengan alguna etiqueta minoritaria.

4.3.2. Evaluación de la diferencia entre *labelsets*

La regla ENN de Wilson establece que los candidatos cuya clase difiera de la mitad o más de sus k vecinos más cercanos han de eliminarse. En un contexto de clasificación no multietiqueta, al existir solo una clase con la que comparar, la diferencia entre el candidato y sus vecinos solo puede ser del 0 % (misma clase) o del 100 % (clases diferentes). Por tanto la instancia candidata se eliminaría al existir un 100 % de diferencia ente su clase y la de la mitad de sus vecinos.

Las instancias de un conjunto de datos multietiqueta tienen varias etiquetas asociadas, por lo que la diferencia entre los *labelset* de dos muestras podría ser del 100 %, pero también de cualquier valor inferior a este y superior al 0 %.

Consideremos los dos conjuntos de etiquetas mostrados en la Tabla 4.13 y la forma en que podrían evaluarse sus diferencias. Pertenecen a un conjunto de datos con 376 etiquetas diferentes, pero cada instancia tiene asociadas únicamente 3 o 4 de ellas. Esto es lo que ocurre con el conjunto `corel5k` concretamente.

Utilizando la distancia de Hamming podría concluirse que existe una diferencia de 3 unidades entre ambos *labelset*. Dado que la longitud total (número de

Tabla 4.13: Diferencia entre los *labelset* de dos instancias.

índice	1	2	3	4	5	6	...	375	376
labelset1	0	1	1	0	0	1	**0**	0	0
labelset2	1	0	1	0	0	1	**0**	1	0
#Dif.	1	1	0	0	0	0	**0**	1	0

etiquetas) es 376, esto nos daría un 0.798 % de diferencia. Sin embargo, si solo se considerasen las etiquetas realmente activas (aquellas que son relevantes para cada instancia) en cualquiera de los *labelset* el resultado sería totalmente distinto, ya que solo hay 7 etiquetas activas en total. El porcentaje de diferencia sería del 42.857 %, muy superior al anterior.

Hay muchos conjuntos de datos multietiqueta con cientos de etiquetas distintas, pero en los que solo una pequeña parte de estas están activas en cada muestra. Si se calculase la diferencia mediante la distancia de Hamming estándar los valores obtenidos serían siempre extremadamente bajos. Por tanto, considerar únicamente las etiquetas relevantes es algo que tiene sentido a la hora de evaluar las diferencias entre *labelsets*.

MLeNN calcula una distancia de Hammin *ajustada* entre los *labelset* de la muestra candidata y los de sus vecinos más cercanos, contando el número de diferencias y dividiendo entre el número de etiquetas activas. Como resultado se obtiene un valor en el rango [0,1]. Aplicando un umbral configurable (HT en el Algoritmo 8), el algoritmo determina cuáles de sus vecinos será considerado distinto.

4.3.3. Estructura de la experimentación

La experimentación llevada a cabo para analizar el comportamiento del algoritmo propuesto se ha estructurado en dos fases. El primer objetivo es determinar si MLeNN es capaz de mejorar los resultados de clasificación. Para ello se com-

para la salida de los clasificadores antes y después de preprocesar un conjunto de MLD. La segunda fase persigue comparar el rendimiento de MLeNN frente al *undersampling* aleatorio llevado a cabo por LP-RUS sobre el mismo conjunto de MLD.

Para evaluar experimentalmente el rendimiento de MLeNN se han utilizando los seis MLD cuyas características se muestran en la Tabla 4.14. Estos se han particionado siguiendo un esquema de validación 2×5. Como puede inferirse a partir de los valores para *MeanIR*, cinco de estos conjuntos de datos están realmente desbalanceados, con valores que van de 7.20 a 256.40. Por el contrario, el MLD `emotions` no podría considerarse como realmente desbalanceado. Ha sido incluido en la experimentación para comprobar el comportamiento de MLeNN sobre conjuntos de datos no desbalanceados. El algoritmo MLeNN ha sido utilizado para preprocesar todos ellos con *NN=3* (3 vecinos cercanos) y *HT=0.75* (umbral del 75 % de diferencia).

Tabla 4.14: Características de los MLD usados en la experimentación.

MLD	**Instancias**	**Atributos**	**Etiquetas**	**MaxIR**	**MeanIR**
corel5k	5000	499	374	1120.00	189.57
corel16k	13766	500	161	126.80	34.16
emotions	593	72	6	1.78	1.48
enron	1702	753	53	913.00	73.95
mediamill	43907	120	101	1092.55	256.40
yeast	2417	198	14	53.41	7.20

En cuanto a los algoritmos de clasificación empleados, se ha optado por un método de transformación básico como es BR (Godbole and Sarawagi [2004]) y dos opciones más avanzadas: CLR (Fürnkranz et al. [2008]) y RAkEL (Tsoumakas and Vlahavas [2007]). El conocido algoritmo de clasificación basado en árboles C4.5 se ha utilizado como clasificador subyacente allí donde ha sido necesario.

Tres métricas de evaluación del rendimiento en clasificación, *Accuracy*, *MacroFM* y *MicroFM*, han sido usadas para valorar las predicciones obtenidas.

4.3.4. Resultados y análisis

Los resultados obtenidos de clasificación sin preprocesamiento, tras aplicar MLeNN y tras aplicar ML-RUS, para cada algoritmo de clasificación y métrica, son los mostrados en la Figura 4.1.

En primer lugar, puede constatarse que el *undersampling* aplicado por MLeNN ha mejorado los resultados que se obtenían sin preprocesamiento en muchos casos. Hay, no obstante, algunas excepciones. La más notable es la correspondiente al MLD `emotions`, cuyos resultados tienden a empeorar tras ser preprocesado con MLeNN. Esto nos lleva a la conclusión de que un algoritmo de *undersampling*, ya sea aleatorio o heurístico, nunca debería ser aplicado a conjuntos de datos multietiqueta que no están realmente desbalanceados. Otro hecho que puede confirmarse visualmente en la Figura 4.1 es que el rendimiento de MLeNN es casi siempre superior al de LP-RUS.

Con el objetivo de respaldar estos resultados formalmente, se ha usado el test estadístico no paramétrico de Wilcoxon para comparar los resultados de MLeNN con los obtenidos sin preprocesamiento, por una parte, y con los obtenidos tras aplicar LP-RUS, por otra. Los resultados de dichos tests son los mostrados en la Tabla 4.15 y la Tabla 4.16, respectivamente. Un asterisco a la derecha de un valor denota que es el mejor para una cierta métrica. A la derecha del peor valor se indica el *p-value* exacto devuelto por el test al comparar con el mejor.

Tabla 4.15: MLeNN versus resultados base - Rankings promedio y *p-values*

Algoritmo	**Accuracy**		**MacroFM**		**MicroFM**	
	Rank	p-value	Rank	p-value	Rank	p-value
Base	1.778	0.0016	1.5556	0.3498	1.778	0.0055
MLeNN	1.222	$\star$	1.4444	$\star$	1.222	$\star$

Del análisis de estos resultados se puede concluir que MLeNN es un algoritmo de *undersampling* para clasificación multietiqueta competitivo, dado que siempre obtiene un mejor rendimiento que LP-RUS desde un punto de vista estadístico. Además el *undersampling* que lleva a cabo MLeNN es capaz de mejorar los

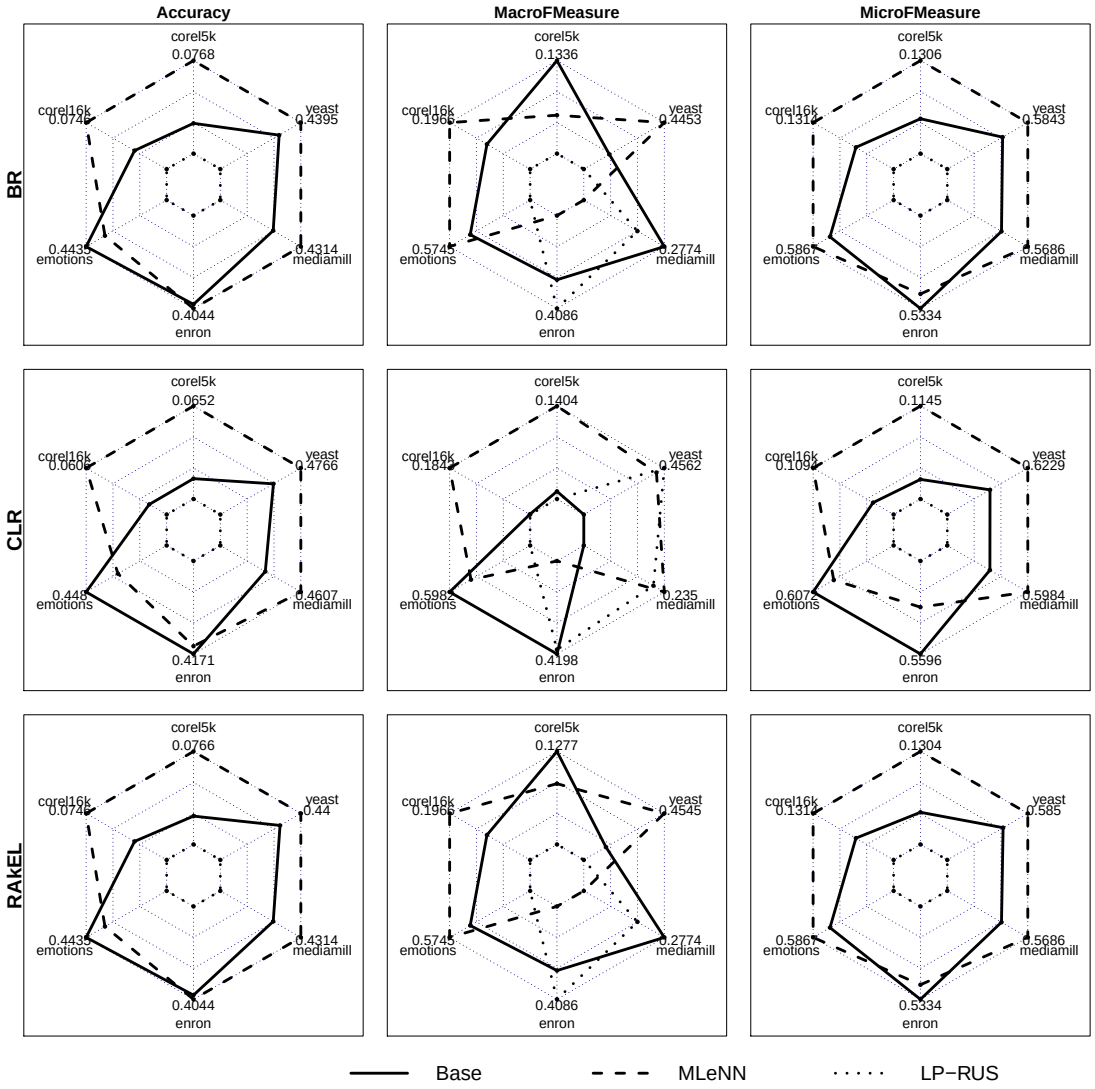


Figura 4.1: Cada gráfica muestra los resultados de clasificación correspondientes a una combinación métrica/algoritmo.

resultados de clasificación cuando se compara con los que se obtienen sin preprocesamiento. Dicha mejora tiene significación estadística en dos de las métricas de rendimiento usadas, a pesar de haberse incluido en la evaluación un MLD como `emotions` que no está desbalanceado.

Tabla 4.16: MLeNN vs LP-RUS - Rankings promedio y *p-values*

Algoritmo	**Accuracy**		**MacroFM**		**MicroFM**	
	Rank	p-value	Rank	p-value	Rank	p-value
LP-RUS	2.000	0.0001	1.6667	0.0825	2.000	0.0001
MLeNN	1.000	$\star$	1.3333	$\star$	1.000	$\star$

En definitiva, y más allá de los resultados concretos que acaban de analizarse, la experimentación nos permite aventurar que la selección cuidadosa de las muestras a eliminar, frente a la opción meramente aleatoria, supone un avance en el que es posible seguir profundizando.

4.4. Concurrencia entre etiquetas desbalanceadas

Tanto en este capítulo como en el anterior se han propuesto múltiples adaptaciones de algoritmos de remuestreo tradicionales al campo multietiqueta, demostrándose cómo en muchos casos es posible mejorar los resultados obtenidos por los clasificadores. A pesar de todo, ciertas características de los MLD pueden ser un desafío para las técnicas de remuestreo.

En esta sección se mostrará cómo la aparición conjunta en una misma instancia de etiquetas mayoritarias y minoritarias puede obstaculizar el objetivo de dichos algoritmos. La hipótesis de partida es que este síntoma, la concurrencia entre etiquetas desbalanceadas, influirá en el comportamiento de los algoritmos de remuestreo. Se propondrá una una nueva métrica, llamada *SCUMBLE* (*Score of ConcUrrence among iMBalanced LabEls*), que ha sido diseñada explícitamente para medir esta casuística. La efectividad de dicha métrica será analizada empíricamente.

La métrica *SCUMBLE* ha sido concebida con el objetivo de determinar cómo de difícil será para un algoritmo de remuestreo trabajar sobre un cierto MLD. Para ello se evaluará el nivel de concurrencia entre etiquetas desbalanceadas, facilitando como resultado una puntuación en el rango [0,1]. Una puntuación baja denotaría

un MLD en el que no hay mucha concurrencia entre etiquetas desbalanceadas, mientras que un valor alto evidenciaría el caso opuesto. La hipótesis de trabajo es que cuanto menor sea el valor devuelto por esta métrica mejor podrá trabajar un método de remuestreo sobre los datos.

4.4.1. MLD y rendimiento de algoritmos de remuestreo

La mayoría de los métodos de remuestreo hacen su trabajo eliminando instancias asociadas a la clase mayoritaria o bien creando nuevas muestras vinculadas a la menos frecuente. Dado que cada instancia puede pertenecer únicamente a una clase, esas acciones efectivamente tenderán a equilibrar la distribución de clases. Esto, sin embargo, no es necesariamente cierto cuando se trabaja con conjuntos de datos multietiqueta.

Las instancias de un MLD están habitualmente asociadas simultáneamente a dos o más etiquetas. Es completamente factible que una de esas etiquetas sea minoritaria y la otra mayoritaria. En la situación más extrema, todas las apariciones de una cierta etiqueta minoritaria podrían ocurrir conjuntamente con una mayoritaria, en las mismas instancias. En la práctica el escenario sería más complejo, ya que normalmente en un MLD hay más de una clase minoritaria/mayoritaria. En consecuencia la potencial existencia de instancias con representación simultánea de etiquetas minoritarias y mayoritarias es muy alta. Este es un hecho al que nos referiremos como concurrencia entre etiquetas desbalanceadas.

Un método de *oversampling* que clone instancias minoritarias, como el algoritmo ML-ROS propuesto en el capítulo previo, o que produzca nuevas muestras a partir de las ya existentes preservando sus *labelsets*, como es el caso del algoritmo SmoteUG introducido en Giraldo-Forero et al. [2013], podría estar también incrementando el número de instancias asociadas a las etiquetas mayoritarias. De esta forma el nivel de desbalanceo difícilmente podría reducirse si hay un alto nivel de concurrencia entre etiquetas desbalanceadas. Análogamente, un algoritmo de *undersampling* diseñado para eliminar muestras de las etiquetas mayoritarias

podría inadvertidamente provocar también la pérdida de muestras asociadas a las etiquetas minoritarias.

La ineficacia de estos métodos de resampling, al ser usados sobre ciertos conjuntos de datos, sería apreciada tras aplicar el preprocesamiento y haber evaluado los resultados de clasificación. Este es un proceso que precisa potencia de proceso y tiempo. Por dicha razón sería deseable saber anticipadamente el nivel de concurrencia entre etiquetas desbalanceadas que presenta cada MLD, ahorrando esos valiosos recursos.

4.4.2. La medida SCUMBLE

La concurrencia entre etiquetas desbalanceadas presente en un MLD puede, en algunos casos, ser explorada visualmente, tal y como se muestra en la Figura 4.2. Cada arco representa a una etiqueta, siendo la longitud del arco proporcional al número de instancias en que dicha etiqueta está presente. La gráfica superior corresponde al conjunto de datos `genbase`. En la posición de las doce en punto aparece una etiqueta llamada `P750` que es claramente minoritaria. Todas las muestras asociadas a esta etiqueta también contienen a `P271`, otra etiqueta minoritaria. La misma situación puede verse con la etiqueta `P154`. En contraposición, en el conjunto de datos `yeast` (parte inferior de la figura) es fácil ver que las muestras asociadas a clases minoritarias, como son `Class14` y `Class9`, aparecen siempre junto a una o más etiquetas mayoritarias. A primera vista podría concluirse que la concurrencia entre etiquetas desbalanceadas es superior en `yeast` que en `genbase`. Sin embargo, esta técnica de exploración visual no resulta útil con conjuntos de datos que tienen más de unas pocas docenas de etiquetas.

La métrica *SCUMBLE* pretende cuantificar la varianza de desbalanceo entre las etiquetas presentes en cada muestra de datos. Esta métrica (véase la Ecuación 4.1) está basada en el índice de Atkinson (Atkinson [1970]) y en la métrica *IRLbl* propuesta en el capítulo previo. La primera es una medida econométrica para evaluar las desigualdades sociales entre individuos de una población. La segunda

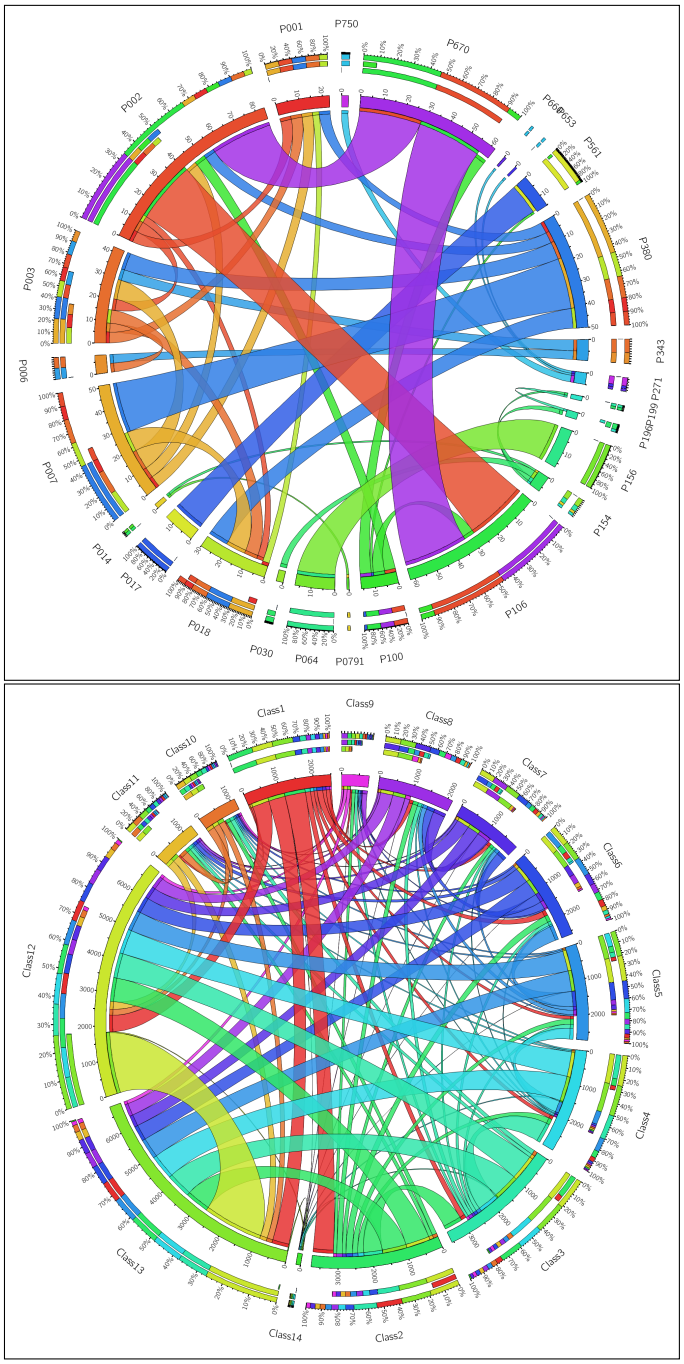


Figura 4.2: Concurrencia de etiquetas en los MLD genbase (arriba) y yeast.

es la métrica que nos permite conocer la ratio de desbalanceo de las etiquetas en un MLD.

$$SCUMBLE(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}[1 - \frac{1}{\overline{IRLbl_i}}(\prod_{l=1}^{|L|} IRLbl_{il})^{(1/|L|)}] \tag{4.1}$$

El índice de Atkinson se usa para conocer la diversidad entre los ingresos de las personas, mientras que nuestro objetivo es evaluar hasta qué punto aparecen conjuntamente etiquetas con niveles de desbalanceo muy dispares. Nuestra hipótesis es que cuanto más alto sea el nivel de concurrencia tanto más difícil será tratar estos MLD mediante algoritmos de remuestreo y, por tanto, peor será el rendimiento obtenido con ellos.

El índice de Atkinson se calcula utilizando ingresos, nosotros usaremos en su lugar el nivel de desbalanceo de cada etiqueta, usando cada instancia D_i del D como una población y las etiquetas activas en D_i como individuos. Si la etiqueta l está presente en la instancia i, entonces $IRLbl_{il} = IRLbl(l)$. En caso contrario tendremos $IRLbl_{il} = 0$. $\overline{IRLbl_i}$ corresponde al nivel medio de desbalanceo de las etiquetas presentes en la instancia i. Las puntuaciones para cada muestra son promediadas, obteniendo el valor final de *SCUMBLE*.

Determinar si la hipótesis de partida es correcta o no, y saber si esta métrica será capaz de predecir la dificultad que un MLD implica para los métodos de remuestreo, requerirá un análisis experimental.

4.4.3. Configuración de la experimentación

Para dilucidar la utilidad de la métrica *SCUMBLE* se han utilizado los seis conjuntos de datos multietiqueta indicados en la Tabla 4.17. Todos ellos están desbalanceados, por lo que teóricamente podrían beneficiarse de la aplicación de un algoritmo de remuestreo. Aparte de la medida *SCUMBLE* también se muestran los valores para *MaxIR* y *MeanIR*. Estos serán tomados como punto de referencia para el posterior análisis. Los MLD aparecen en la tabla ordenados por el valor

de *SCUMBLE*, de mayor a menor. De acuerdo a esta métrica, `corel5k` y `cal500` serían los MLD más difíciles, ya que reflejan los valores de concurrencia entre etiquetas desbalanceadas más altos y tienen diferentes niveles de desbalanceo. Por otra parte, los MLD `medical` y `genbase` podrían ser los que más se beneficiasen del remuestreo.

Tabla 4.17: Medidas de desbalanceo de los MLD antes de preprocesarlos.

MLD	**SCUMBLE**	**MaxIR**	**MeanIR**
corel5k	0.3932	896.0000	168.7806
cal500	0.3369	133.1917	21.2736
enron	0.3023	657.0500	72.7730
yeast	0.1044	53.6894	7.2180
medical	0.0465	212.8000	72.1674
genbase	0.0283	136.8000	32.4130

En cuanto a métodos de remuestreo, se han usado los algoritmos LP-ROS y LP-RUS propuestos en el capítulo previo. Ambos están basados en la técnica de transformación LP. LP-ROS lleva a cabo un *oversampling* clonando instancias con conjuntos de etiquetas minoritarios, mientras que LP-RUS efectúa un *undersampling* eliminando muestras asociadas a conjuntos de etiquetas mayoritarios. Tras aplicar estos algoritmos a los MLD se ha procedido a reevaluar las medidas de desbalanceo.

4.4.4. Resultados y análisis

Una vez que se han aplicado los algoritmos LP-ROS y LP-RUS a cada uno de los MLD, se han reevaluado los niveles de desbalanceo obteniendo los resultados que muestra la Tabla 4.18. Al comparar estos valores con los de la Tabla 4.17 es posible verificar que, en general, se ha obtenido una mejora en los niveles de desbalanceo. Si bien hay algunas excepciones, en la mayoría de los casos tanto el *MaxIR* como el *MeanIR* son más bajos tras aplicar los algoritmos de remuestreo.

Tabla 4.18: Niveles de desbalanceo tras aplicar los algoritmos de remuestreo.

	LP-ROS		LP-RUS	
MLD	**MaxIR**	**MeanIR**	**MaxIR**	**MeanIR**
corel5k	969.4000	140.7429	817.1000	155.0324
cal500	179.35838	25.4685	620.0500	68.6716
enron	710.9667	53.2547	133.1917	21.2736
yeast	15.4180	2.6116	83.8000	19.8844
medical	39.9633	10.5558	46.5698	6.3706
genbase	13.7030	4.5004	150.8000	51.1567

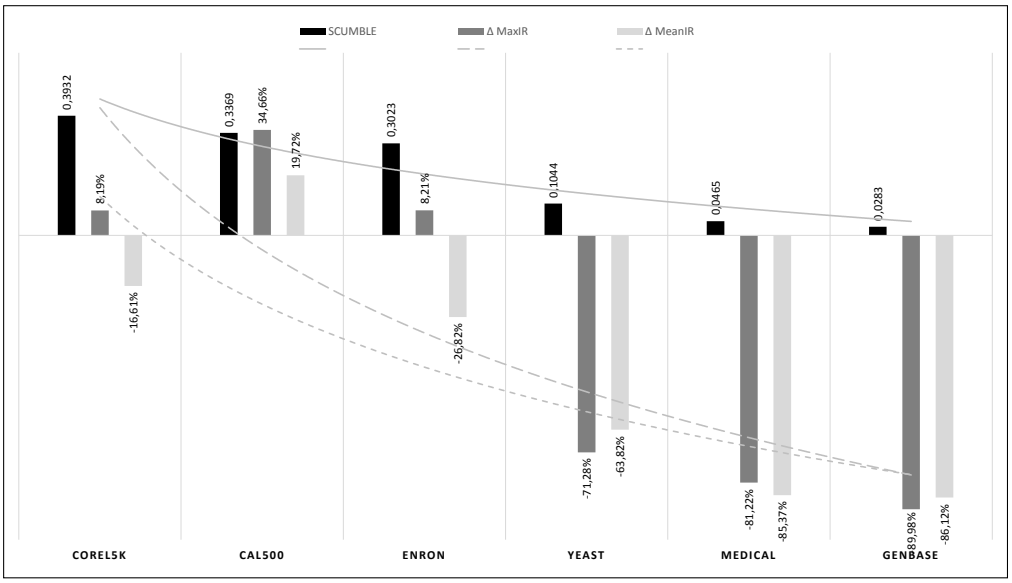


Figura 4.3: SCUMBLE vs cambios en nivel de desbalanceo tras aplicar LP-ROS.

Sería interesante saber si la reducción del desbalanceo es proporcionalmente coherente con los valores devueltos por la métrica *SCUMBLE*. Las gráficas mostradas en la Figura 4.3 y la Figura 4.4 persiguen ilustrar visualmente la conexión entre los valores de *SCUMBLE* y la variación en los niveles de desbalanceo.

Para cada conjunto de datos se ha representado el valor de *SCUMBLE* junto con el porcentaje de cambio en los valores de *MaxIR* y *MeanIR* tras aplicar

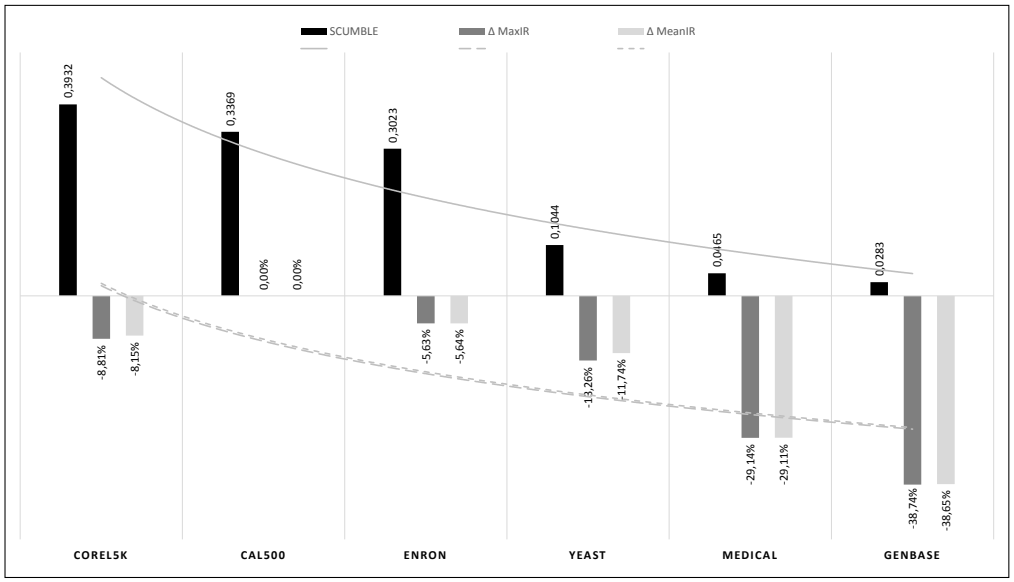


Figura 4.4: SCUMBLE vs cambios en el nivel de desbalanceo tras aplicar LP-RUS.

los métodos de remuestreo LP-ROS y LP-RUS. La tendencia de los tres valores es mostrada mediante tres curvas logarítmicas. Como puede apreciarse, existe un claro paralelismo entre la línea continua, que corresponde a *SCUMBLE*, y las líneas punteadas. Esta afinidad es notable en el caso del algoritmo LP-RUS (Figura 4.4).

Si bien las gráficas previas permiten inferir que existe una importante correlación entre la medida de *SCUMBLE* y el éxito de los métodos de remuestreo, esta relación ha de ser analizada formalmente. Para ello se ha aplicado el test de correlación de Pearson sobre los valores de *SCUMBLE* y los cambios relativos en los niveles de desbalanceo para cada algoritmo de remuestreo. Los coeficientes de correlación resultante y sus *p-value* son los mostrados en la Tabla 4.19. Como puede comprobarse, todos los coeficientes están por encima del 80 % y todos los *p-value* por debajo del 0.05. En consecuencia puede concluirse que existe una correlación estadísticamente significativa entre los valores de *SCUMBLE* y el comportamiento de los algoritmos de remuestreo probados.

Tabla 4.19: Resultados del test de correlación de Pearson.

	SCUMBLE vs ΔMaxIR		**SCUMBLE vs ΔMeanIR**	
Algoritmo	**Cor**	**p-value**	**Cor**	**p-value**
LP-ROS	0.8120	0.0497	0.9189	0.0096
LP-RUS	0.8607	0.0278	0.8517	0.0314

A partir de los resultados de este análisis parece razonable evitar el uso de algoritmos básicos de remuestreo sobre MLD cuya medida de *SCUMBLE* supere 0.1, como es el caso de `corel5k`, `cal500` y `enron`. En esta situación los beneficios obtenidos del remuestreo, si es que los hay, serán muy pequeños. El resultado podría ser incluso un empeoramiento de los niveles de desbalanceo. En promedio, el *MeanIR* para los tres MLD con *SCUMBLE* $> 0{,}3$ se ha reducido solamente un 6 %, mientras que el *MaxIR* en realidad se ha incrementado en el mismo porcentaje. En contraposición, la reducción promedio del *MeanIR* para los demás MLDs, con *SCUMBLE* $\lesssim 0{,}1$, alcanza el 52 % y la reducción del *MaxIR* el 54 %.

Con el objetivo de saber si estos cambios en los niveles de desbalanceo tendrían alguna influencia sobre los resultados de clasificación, y si existe una correlación con los valores de *SCUMBLE*, se ha utilizado el algoritmo HOMER para procesar las particiones de entrenamiento de los MLD siguiendo un esquema de 2×5 validación cruzada. Es necesario resaltar que lo que nos interesa en este momento no es el rendimiento puro en clasificación, sino cómo este cambia tras haber aplicado un algoritmo de preprocesamiento y si ese cambio tiene correlación con los valores de *SCUMBLE*. El algoritmo HOMER, por tanto, es usado como una herramienta para obtener resultados de clasificación antes y después de aplicar el remuestreo. Se podría haber usado cualquier otro algoritmo para esta tarea. También debe tenerse en cuenta que la métrica *SCUMBLE* no se utiliza en ningún caso en la experimentación para influir en el comportamiento de LP-ROS, LP-RUS o HOMER. El objetivo es explorar teóricamente las correlaciones entre los cambios en resultados de clasificación y los valores de *SCUMBLE*.

La Tabla 4.20 muestra los resultados evaluados con la métrica *F-measure*, la media armónica entre las métricas *Precision* y *Recall*. Puede apreciarse que para los tres MLD que tiene altos valores de *SCUMBLE* la aplicación del preprocesamiento ha producido un deterioro en los resultados de clasificación. Entre los demás MLD el preprocesamiento ha generado una mejora en algunos casos, mientras que en otros se experimenta un ligero empeoramiento, menor al 1 %. En consecuencia, si bien el comportamiento del algoritmo de clasificación también se verá afectado por otras características de los MLD, puede concluirse que la métrica *SCUMBLE* nos ofrece una información valiosa a la hora de determinar la conveniencia de aplicar o no un método de preprocesamiento.

Tabla 4.20: Valores de F-Measure obtenidos usando HOMER como clasificador.

MLD	**Base**	**LP-RUS**	**LP-ROS**	**ΔRUS**	**ΔROS**
corel5k	0.3857	0.2828	0.2920	-26.6788	-24.2935
cal500	0.3944	0.3127	0.3134	-20.7150	-20.5375
enron	0.5992	0.5761	0.5874	-3.8551	-1.9693
yeast	0.6071	0.6950	0.6966	14.4787	14.7422
medical	0.9238	0.9158	0.9162	-0.8660	-0.8227
genbase	0.9896	0.9818	0.9912	-0.7882	0.1617

Tras corroborar que existe una correlación entre los valores de *SCUMBLE* y el grado en que los algoritmos de remuestreo pueden reducir el nivel de desbalanceo de los MLD, y que esto parece influir también en el rendimiento en clasificación, sería preciso llevar a cabo una experimentación más extensa y un análisis en más profundidad, usando más conjuntos de datos, métodos de remuestreo y métricas de evaluación. No obstante, este trabajo nos permite afirmar que los algoritmos que se limitan a eliminar o clonar instancias no son una solución general en el campo multietiqueta. Es necesario buscar alternativas más sofisticadas que tengan en cuenta la concurrencia entre etiquetas.

4.5. El algoritmo REMEDIAL

En la anterior sección se ha demostrado cómo la concurrencia entre etiquetas desbalanceadas en un conjunto de datos multietiqueta, característica que puede ser evaluada con la métrica *SCUMBLE* propuesta en dicha sección, puede impedir que un algoritmo de remuestreo consiga su objetivo, al resultar imposible reequilibrar la distribución de etiquetas simplemente eliminando muestras o clonando las existentes.

El trabajo tratado en esta sección se apoya en la citada métrica para proponer un nuevo algoritmo, llamado REMEDIAL (*REsamling MultilabEl datasets by Decoupling highly ImbAlanced Labels*). Este evalúa la concurrencia entre etiquetas desbalanceadas de cada muestra, separando aquellas con un alto nivel de *SCUMBLE* desacoplando así las etiquetas mayoritarias de las minoritarias. Este enfoque por sí mismo ya sería interesante con ciertos conjuntos de datos. No obstante, el objetivo de REMEDIAL no es tanto competir con los algoritmos de remuestreo existentes como facilitar el trabajo de los mismos.

En las siguientes subsecciones se describe el algoritmo REMEDIAL y se detalla la experimentación llevada a cabo para comprobar su funcionamiento, analizándose los resultados obtenidos y las conclusiones extraídas a partir de los mismos.

4.5.1. Descripción del algoritmo

Como su propia denominación sugiere, REMEDIAL es un método específicamente diseñado para conjuntos de datos multietiqueta que sufren de un alto nivel de concurrencia entre etiquetas desbalanceadas. En este contexto *highly imbalanced labels* ha de entenderse como etiquetas con grandes diferencias en sus *IRLbl*. Este es un hecho que puede evaluarse con la métrica *SCUMBLE*, por lo que REMEDIAL está pensado para aquellos MLD con un alto nivel de *SCUMBLE*.

Cuando las pocas muestras en que está presente una cierta etiqueta minoritaria también contienen una o más etiquetas mayoritarias, cuya frecuencia en el

MLD es mucho más alta, la utilidad de las características de entrada para predecir las etiquetas estará sesgada hacia las mayoritarias. La hipótesis de la que parte este trabajo es que, en cierta forma, las etiquetas mayoritarias están enmascarando la presencia de las minoritarias al aparecer conjuntamente, un problema que podría solventarse hasta cierto punto separando las etiquetas de dichas instancias.

REMEDIAL es un algoritmo de remuestreo. Podría decirse que es un método de sobremuestreo, ya que genera nuevas instancias en algunos casos. Al mismo tiempo también modifica muestras existentes. Resumiendo, REMEDIAL es un algoritmo de edición más sobremuestreo, siendo un enfoque que tiene sinergias con las técnicas de remuestreo tradicionales. El pseudo-código de REMEDIAL es el mostrado en Algoritmo 9.

Algorithm 9 Pesudo-código del algoritmo REMEDIAL.

```
1:  function REMEDIAL(MLD D, Labels L)
2:      IRLbl_l ← calculateIRLbl(l in L)        ▷ Calcular los niveles de desbalanceo
3:      MeanIR ← mean(IRLbl)
4:      SCUMBLEIns_i ← calculateSCUMBLE(D_i in D)   ▷ Calcular SCUMBLE
5:      SCUMBLE ← mean(SCUMBLEIns)
6:      for each instance i in D do
7:          if SCUMBLEIns_i > SCUMBLE then
8:              D'_i ← D_i                          ▷ Clonar la instancia afectada
9:              D_i[labels_{IRLbl<=IRMean}] ← 0     ▷ Mantener las etiquetas minoritarias
10:             D'_i[labels_{IRLbl>IRMean}] ← 0     ▷ Mantener las etiquetas mayoritarias
11:             D ← D + D'_i
12:         end if
13:     end for
14: end function
```

Las métricas *IRLbl*, *MeanIR* y *SCUMBLE* se calculan en las líneas 2-5. $SCUMBLE_{Ins_i}$ es el nivel de concurrencia en la instancia D_i. El *SCUMBLE* medio para cada MLD se obtiene promediando el *SCUMBLE* individual de las muestras.

Tomando como referencia el *SCUMBLE* promedio, se procesan solo aquellas instancias en las que *SCUMBLEIns*>*SCUMBLE*. Estas instancias, que contienen etiquetas minoritarias y mayoritarias, son desdobladas en dos instancias, una

conteniendo solo las etiquetas mayoritarias y otras solamente con las minoritarias. En la línea 8, D_i, una muestra afectada por el problema en cuestión, es clonada en D'_i. La fórmula de la línea 9 edita la instancia original D_i eliminando de ella las etiquetas mayoritarias. Se consideran mayoritarias aquellas etiquetas cuyo *IRLbl* es igual o inferior a *MeanIR*. En la línea 10 se hace lo opuesto, eliminando de la instancia clonada D'_i las etiquetas minoritarias. D_i pertenece al conjunto de datos D, pero D'_i tiene que ser agregada al mismo (línea 11).

4.5.2. Configuración de experimentación

Para comprobar la influencia de REMEDIAL en el rendimiento en clasificación se han utilizado los seis conjuntos de datos mostrados en la Tabla 4.21. Estos son los mismos que se utilizaron en la experimentación de la sección previa, al proponer la medida *SCUMBLE*. Por una parte se tienen cuatro MLD con valores de *SCUMBLE* por encima de 0.1, que mostraron ser los más problemáticos al procesarse con métodos clásicos de remuestreo. Por otra tenemos dos MLD con bajos niveles de *SCUMBLE*.

Tabla 4.21: Datasets usados en la experimentación.

Categoría	MLD	SCUMBLE	MaxIR	MeanIR
Alto SCUMBLE > 0,1	cal500	0.3369	133.1917	21.2736
	corel5k	0.3932	896.0000	168.7806
	enron	0.3023	657.0500	72.7730
	yeast	0.1044	53.6894	7.2180
Bajo SCUMBLE < 0,1	genbase	0.0283	136.8000	32.4130
	medical	0.0465	212.8000	72.1674

Estos conjuntos de datos han sido facilitados, antes y después de ser preprocesados con REMEDIAL, a tres algoritmos de clasificación multietiqueta: BR (Boutell et al. [2004]), HOMER (Tsoumakas et al. [2008]) e IBLR (Cheng and Hüllermeier [2009]). Estos son representantes de tres enfoques distintos de clasificación multietiqueta: un multiclasificador binario, otro de clasificadores LP y

un clasificador basado en instancias. Se ha utilizado un esquema de 5×2 validación cruzada, como en los demás casos, obteniéndose tres métricas de evaluación: HammingLoss (*HL*), *MacroFM* y *MicroFM*.

4.5.3. Resultados y análisis

Los resultados obtenidos de cada clasificador con los citados MLD, antes y después de su preprocesamiento con REMEDIAL, son los mostrados en la Tabla 4.22. Los mejores resultados se han destacado en negrita.

Tabla 4.22: Resultados antes y después de aplicar REMEDIAL

		BR		**HOMER**		**IBLR**	
	MLD	**Antes**	**Después**	**Antes**	**Después**	**Antes**	**Después**
HL	cal500	0.1630	**0.1496**	0.1888	**0.1794**	0.2340	**0.2125**
	corel5k	0.0098	**0.0094**	0.0132	**0.0118**	0.0242	**0.0148**
	enron	**0.0522**	0.0524	0.0583	**0.0560**	**0.0572**	0.0573
	yeast	0.2505	**0.2240**	0.2632	**0.2433**	**0.1942**	0.2139
	genbase	**0.0012**	0.0084	**0.0016**	0.0062	**0.0022**	0.0092
	medical	**0.0107**	0.0131	**0.0108**	0.0125	0.0198	0.0198
MacroFM	cal500	**0.2934**	0.2516	0.3316	**0.3358**	**0.2772**	0.2597
	corel5k	0.1774	**0.1826**	0.1916	**0.1924**	0.1059	**0.1432**
	enron	0.4029	**0.4190**	0.3790	**0.3793**	**0.3458**	0.3114
	yeast	0.4341	**0.5204**	0.4334	**0.4626**	**0.4944**	0.4156
	genbase	0.9890	**0.9924**	**0.9780**	0.9697	**0.9655**	0.8450
	medical	**0.8166**	0.8013	**0.7942**	0.7780	**0.6404**	0.6216
MicroFM	cal500	**0.3488**	0.2506	0.3978	**0.4008**	**0.3184**	0.2934
	corel5k	**0.1096**	0.0782	**0.1744**	0.1627	**0.0542**	0.0530
	enron	**0.5334**	0.4745	**0.5265**	0.5036	**0.4561**	0.3541
	yeast	0.5787	**0.5898**	0.5763	**0.5974**	**0.6502**	0.5546
	genbase	**0.9867**	0.9012	**0.9820**	0.9284	**0.9768**	0.8902
	medical	**0.8006**	0.7350	**0.7994**	0.7582	**0.6324**	0.5830

Comenzando por los dos MLD con bajos niveles de *SCUMBLE*, los resultados producidos por REMEDIAL no son buenos casi en ningún caso. A pesar de que algunas diferencias son muy pequeñas, en general el desdoblamiento de las etiquetas ha empeorado el rendimiento en clasificación. En consecuencia puede extraerse una guía de uso a seguir a partir de estos resultados, REMEDIAL no

debería utilizarse con conjuntos datos con bajos valores de *SCUMBLE*, ya que es un algoritmo diseñado específicamente para tratar la casuística contraria.

El análisis de los resultados de los otros cuatro MLD puede dividirse en dos partes, dependiendo de cuál sea el foco de interés. Si examinamos los resultados por métrica de evaluación del rendimiento, está claro que REMEDIAL está beneficiando a las etiquetas minoritarias, con mejores valores de *MacroFM*, y también presenta un buen comportamiento general, denotado por los valores de *HL* tras el remuestreo. Cuando se utiliza *MicroFM* los resultados son mixtos, ya que para algunos MLD se mejora y para otros se empeora. Examinando los resultados por clasificador puede observarse que REMEDIAL funciona mejor con BR y con HOMER que con IBLR.

Los algoritmos basados en BR entrenan un clasificador para cada una de las etiquetas, tomando como positivas las instancias que la contienen y como negativas el resto de muestras. Cuando se procesa una etiqueta mayoritaria, todas las instancias en que aparece conjuntamente con una etiqueta minoritaria son procesadas como positivas, ignorando el hecho de que contienen otras etiquetas. El desdoblamiento de esas etiquetas tiende a equilibrar el sesgo del clasificador. Los algoritmos basados en LP, como es el caso de HOMER, es prácticamente seguro que se beneficiarán de REMEDIAL, ya que el desdoblamiento produce conjuntos de etiquetas más sencillos. Además, el número de *labelsets* distintos también se reduce tras el remuestreo. La influencia de REMEDIAL en un clasificador basado en instancias, como es IBLR, es fácil de adivinar. Los atributos de las muestras desdobladas no cambian, por lo que siguen ocupando exactamente la misma posición con respecto a la instancia que está tomándose como referencia para la búsqueda de vecinos cercanos. Por tanto el clasificador encontrará dos muestras a la misma distancia pero con conjuntos de etiquetas disjuntos, algo que puede resultar confuso dependiendo de cómo el algoritmo prediga el conjunto de etiquetas para la muestra procesada.

Globalmente, REMEDIAL podría recomendarse como remuestreo para aquellos MLD con altos niveles de *SCUMBLE* y cuando se van a usar clasificadores basados en BR o LP. En estos casos la predicción de las etiquetas minoritarias se

mejoraría y el rendimiento general del clasificador sería mejor. Estos beneficios los aporta por sí mismo, pero REMEDIAL podría ser utilizado como primer paso dirigido a facilitar el trabajo de técnicas de remuestreo tradicionales.

En resumen, el algoritmo propuesto aquí puede mejorar el rendimiento en clasificación bajo ciertas premisas y, en cualquier caso, su comportamiento podría potenciarse usándolo conjuntamente con alguno de los métodos de remuestreo propuestos en este y el capítulo previo.

4.6. Conclusiones

Como resultado del trabajo descrito en el presente capítulo se han propuesto tres nuevos algoritmos de remuestreo para conjuntos de datos multietiqueta: MLSMOTE, MLeNN y REMEDIAL, conjuntamente con una nueva métrica, *SCUMBLE*, diseñada específicamente para evaluar una característica de los MLD no estudiada hasta el momento: la concurrencia entre etiquetas desbalanceadas. Cada una de las propuestas ha sido probada experimentalmente y sus resultados analizados estadísticamente. Se ha demostrado que, en general, las técnicas de remuestreo no aleatorio, como MLSMOTE y MLeNN, producen mejores resultados que sus equivalentes aleatorios, presentados en el capítulo previo.

MLSMOTE es un algoritmo que produce instancias sintéticas, con su propio conjunto de características y etiquetas, en lugar de recurrir a la clonación. La experimentación ha demostrado su positiva influencia en los niveles de desbalanceo de los conjuntos de datos, así como la mejora en el rendimiento en clasificación respecto a resultados base, alcanzando diferencias estadísticamente significativas en casi todos los casos, sobre todo con la métrica *Macro-FMeasure* que representa mejor la correcta clasificación de etiquetas minoritarias. Asimismo, la comparación de MLSMOTE con otros cinco algoritmos de remuestreo y clasificación desbalanceada refleja su efectividad, al quedar siempre primero en los ranking y alcanzar diferencias con significación estadística en múltiples casos.

MLeNN selecciona cuidadosamente las muestras candidatas a ser eliminadas, en lugar de escogerlas aleatoriamente, usando para ello una métrica de distancia de Hamming ajustada al problema multietiqueta. La experimentación efectuada demuestra que este enfoque produce mejoras considerables, con significación estadística en dos de las tres métricas de evaluació de rendimiento usadas. En su comparación con LP-RUS, un algoritmo de *undersampling* puramente aleatorio, MLeNN alcanza mejoras estadísticamente significativas en todos los casos.

Finalmente, REMEDIAL representa un tipo de remuestreo distinto, no existente en clasificación tradicional, adaptado a la problemática que es posible evaluar con la métrica *SCUMBLE*. Los resultados de la experimentación confirman la hipótesis de partida, consiguiéndose mejoras del rendimiento en clasificación en aquellos casos en que se aplica el algoritmo sobre conjuntos de datos con un alto *SCUMBLE*.

4.7. Publicaciones asociadas a este trabajo

El algoritmo MLSMOTE está recogido en el artículo *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "MLSMOTE: Approaching Imbalanced Multilabel Learning Through Synthetic Instance Generation"*, actualmente en fase de revisión en la revista *Knowledge-Based Systems*.

El algoritmo MLeNN fue presentado en el IDEAL'14 (Charte et al. [2014c]), *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "MLeNN: A First Approach to Heuristic Multilabel Undersampling". In* Proc. 15th Int. Conf. Intelligent Data Engineering and Automated Learning, Salamanca, Spain, IDEAL'14*, volume 8669 of* LNCS*, pages 1–9, 2014. ISBN 978-3-319-10839-1. doi: 10.1007/978-3-319-10840-7_1.*

La métrica *SCUMBLE* y la experimentación asociada se presentó en HAIS'14 (Charte et al. [2014b]), *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "Concurrence among Imbalanced Labels and its Influence on Multilabel Resampling Algorithms". In* Proc. 9th Int. Conf. Hybrid Artificial Intelligent Sys-

tems, Salamanca, Spain, HAIS'14*, volume 8480 of* LNCS*, 2014. ISBN 978-3-319-07616-4. doi: 10.1007/978-3-319-07617-1_10.*

El algoritmo REMEDIAL será presentado en HAIS'15, *F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. "Resampling Multilabel Datasets by Decoupling Highly Imbalanced Labels". In* Proc. 10th Int. Conf. Hybrid Artificial Intelligent Systems, Bilbao, Spain, HAIS'15.

Capítulo 5

Análisis exploratorio de conjuntos de datos multietiqueta

Uno de los esfuerzos constantes durante el desarrollo de la presente tesis ha consistido en el estudio de las características de los conjuntos de datos multietiqueta, obteniendo como resultado varias nuevas métricas que, como se ha explicado en los tres capítulos precedentes, nos han permitido comprender mejor la problemática que plantea el trabajo con este tipo de datos.

Conocer la estructura de los conjuntos de datos, especialmente cuando son tan complejos como la mayoría de MLD usados en la literatura, es un paso vital. No obstante las herramientas disponibles para realizar ese trabajo exploratorio son muy limitadas y, en general, su uso no resulta sencillo. Este hecho dificulta el análisis exploratorio, obstaculizando en cierta medida el desarrollo de nuevas soluciones basadas en la comprensión de la estructura y naturaleza de los MLD. Esta situación es la motivación tras el trabajo descrito en este capítulo.

Usando la experiencia adquirida durante el desarrollo de las propuestas descritas en los capítulos previos se ha creado un paquete software que facilita el análisis exploratorio y la manipulación de conjuntos de datos multietiqueta desde el conocido lenguaje/entorno R. En este capítulo se describe la estructura de dicho paquete y la funcionalidad que ofrece a los usuarios de R.

5.1. Introducción

R es un lenguaje y un entorno de trabajo usado por miles de investigadores a escala mundial, siendo una de las plataformas que más posibilidades ofrece a la hora de explorar y analizar datos, así como para la implementación de algoritmos de minería de datos y aprendizaje en general. Actualmente hay disponibles decenas de paquetes para R que aportan implementaciones de árboles de decisión, máquinas de vectores soporte, redes neuronales, clasificadores basados en instancias, algoritmos de agrupamiento y minería de reglas de asociación, etc.

Los conjuntos de datos binarios y multiclase pueden gestionarse en R utilizando *dataframes*, un tipo de dato propio generable a partir del contenido de archivos y bases de datos de múltiples tipos. Generalmente se asume que la última columna de un *dataframe* será, al trabajar con conjuntos de datos para clasificación, el atributo a predecir. Los MLD también pueden ser almacenados como un *dataframe*, pero no existe ninguna infraestructura en R, ni paquete disponible en el repositorio CRAN asociado a dicho software, que facilite el trabajo con este tipo de conjuntos de datos.

Esta carencia de R, y la necesidad de contar con una herramienta cómoda y flexible que permita explorar y operar sobre cualquier MLD, motivó el desarrollo de un nuevo paquete, llamado `mldr`, para cubrir ese hueco. La finalidad del paquete `mldr` es doble. Por una parte ofrece a usuarios acostumbrados a trabajar con R la funcionalidad necesaria para obtener toda la información que se precise de un MLD, generar distintas representaciones gráficas de su contenido, transformarlo y manipular tanto atributos como etiquetas. Por otra, a través de una interfaz gráfica de usuario, se pone a disposición, también de aquellos no habituados a la línea de comandos de R, la mayor parte de las funciones exploratorias.

En las secciones siguientes se describe la funcionalidad que ofrece actualmente el paquete `mldr` y también su interfaz de usuario. Este paquete ha sido aceptado en CRAN y está actualmente disponible para cualquier usuario de R, pudiendo instalarse con los procedimientos habituales.

5.2. El paquete mldr

El paquete `mldr` tiene como objetivo ofrecer a los usuarios las funciones necesarias para llevar a cabo tareas de análisis exploratorio sobre MLD, determinando sus características fundamentales tanto estadísticamente como visualmente. Asimismo aporta las herramientas necesarias para manipular este tipo de conjuntos de datos, incluyendo la aplicación de los métodos de transformación más usuales.

Para realizar su trabajo el paquete `mldr` no depende de MULAN, MEKA u otras herramientas software. Ha sido desarrollado para hacer posible la lectura de MLD tanto en formato MULAN como MEKA, pero sin ninguna dependencia externa. De hecho, sería posible cargar conjuntos de datos multietiqueta almacenados en otros formatos, así como crearlos partiendo de cero. El paquete devuelve un objeto `mldr`, conteniendo tanto los datos del MLD como un gran número de medidas obtenidas a partir de él.

Las funciones ofrecidas por el paquete facilitan el acceso a toda esa información, producen algunas gráficas específicas y hacen posible la manipulación del contenido del MLD. Una interfaz de usuario web, desarrollada con el paquete `shiny`, pone las herramientas de análisis exploratorio del paquete `mldr` al alcance de todo tipo de usuarios, incluso aquellos con poca experiencia con R.

5.2.1. Instalación y carga del paquete mldr

El paquete `mldr` está disponible en los servidores del repositorio CRAN, por lo que puede ser instalado como cualquier otro paquete introduciendo la orden `install.packages(''mldr'')` en la línea de comandos de R. Una vez instalado, el paquete ha de ser cargado en memoria. Para ello se usa la orden `library(''mldr'')`.

Al cargar el paquete pasarán a estar disponibles tres conjuntos de datos multietiqueta incluidos en él: `birds`, `emotions` y `genbase`. El paquete `mldr` usa su propia representación interna para los MLD, a los que se asigna la clase `mldr`.

Dentro de un objeto de dicha clase, como los antes citados `emotions` o `birds`, encontraremos tanto los datos del MLD como la información obtenida a partir de esos datos.

5.2.2. Carga y creación de MLD

Aparte de los tres MLD de ejemplo incluidos en el paquete, la función `mldr()` permite cargar cualquier otro almacenado en los formatos de archivo de MULAN o MEKA. Asumiendo que tuviésemos en la carpeta actual los archivos `corel5k.arff` y `corel5k.xml`, correspondientes al MLD `corel5k` en formato MULAN, usaríamos una orden como la siguiente para cargarlo:

```
> corel5k <- mldr("corel5k")
```

Cargar un MLD en formato MEKA es igualmente sencillo. En este caso no existe un archivo XML con la información de las etiquetas, sino una cabecera especial en el archivo ARFF, un hecho que se le indica a la función `mldr()` con el parámetro `use_xml`:

```
> imdb <- mldr("imdb", use_xml = FALSE)
```

En ambos casos el resultado, siempre que se pueda cargar correctamente el conjunto de datos, será un nuevo objeto `mldr` listo para su uso.

En caso de que el MLD que nos interesa no esté en formato MULAN o MEKA, primero tendríamos que cargarlo en un `data.frame`, por ejemplo usando funciones como `read.csv()`, `read.table()` o algún otro tipo de lector más especializado. A continuación dicho `data.frame` y un vector de enteros indicando los indices de los atributos que actúan como etiquetas serían entregados a la función `mldr_from_dataframe()`. Esta es una función genérica para la creación de un objeto `mldr` a partir de cualquier `data.frame`, por lo que también podría utilizarse para generar nuevos MLD al vuelo, como se muestra en el siguiente ejemplo:

```
> df <- data.frame(matrix(rnorm(1000), ncol = 10))
> df$Label1 <- c(sample(c(0,1), 100, replace = TRUE))
> df$Label2 <- c(sample(c(0,1), 100, replace = TRUE))
> mymldr <- mldr_from_dataframe(df, labelIndices = c(11, 12),
                  name = "testMLDR")
```

Esto asignaría a la variable `mymldr` un conjunto de datos multietiqueta, de nombre `testMLDR`, con 10 atributos de entrada y 2 etiquetas.

5.2.3. Obtención de información de un MLD

Tras cargar cualquier MLD, puede obtenerse un resumen rápido de sus características principales mediante la habitual función `summary()` de R, tal y como se muestra a continuación:

```
> summary(birds)
num.attributes num.instances num.labels num.labelsets
           279           645         19
 num.single.labelsets max.frequency
            133                   73           294
cardinality    density   meanIR    scumble
   1.013953 0.05336597 5.406996 0.03302765
```

Es posible recuperar individualmente cualquiera de estas medidas mediante el miembro `measures` de la clase `mldr`, de la siguiente forma:

```
> emotions$measures$num.attributes
[1] 78

> genbase$measures$scumble
[1] 0.0287591
```

Podemos recuperar toda la información relativa a las etiquetas del MLD, incluyendo el número de veces que aparece, incluyendo *IRLbl* y *SCUMBLE*, usando el miembro `labels` de la clase `mldr`:

```
> birds$labels
                          index count        freq     IRLbl    SCUMBLE
Brown Creeper               261    14 0.021705426  7.357143 0.12484341
Pacific Wren                262    81 0.125581395  1.271605 0.05232609
Pacific-slope Flycatcher    263    46 0.071317829  2.239130 0.06361470
Red-breasted Nuthatch       264     9 0.013953488 11.444444 0.15744451
Dark-eyed Junco             265    20 0.031007752  5.150000 0.10248336
Olive-sided Flycatcher      266    14 0.021705426  7.357143 0.18493760
Hermit Thrush               267    47 0.072868217  2.191489 0.06777263
Chestnut-backed Chickadee   268    40 0.062015504  2.575000 0.06807452
Varied Thrush               269    61 0.094573643  1.688525 0.07940806
Hermit Warbler              270    53 0.082170543  1.943396 0.07999006
Swainson's Thrush           271   103 0.159689922  1.000000 0.11214301
Hammond's Flycatcher        272    28 0.043410853  3.678571 0.06129884
Western Tanager             273    33 0.051162791  3.121212 0.07273988
Black-headed Grosbeak       274     9 0.013953488 11.444444 0.20916487
Golden Crowned Kinglet      275    37 0.057364341  2.783784 0.09509474
Warbling Vireo              276    17 0.026356589  6.058824 0.14333613
MacGillivray's Warbler      277     6 0.009302326 17.166667 0.24337605
Stellar's Jay               278    10 0.015503876 10.300000 0.12151527
Common Nighthawk            279    26 0.040310078  3.961538 0.06520272
```

Lo mismo es aplicable para *labelsets* y atributos, a través de los miembros `labelsets` y `attributes` de la clase. Para acceder al contenido del MLD, los valores de sus atributos y etiquetas, se puede utilizar la función `print()` y el miembro `dataset` del objeto `mldr`.

5.2.4. Funciones para generar gráficas

El análisis exploratorio de conjuntos de datos multietiqueta puede ser tedioso, ya que la mayor parte de ellos cuenta con miles de atributos y cientos de etiquetas. El paquete `mldr` ofrece una función `plot()` específica para tratar con objetos de clase `mldr`, permitiendo la generación de varios tipos de gráficas. El primer parámetro entregado a `plot()` debe ser el objeto `mldr`, mientras que el segundo determina el tipo de gráfica a generar:

```
> plot(emotions, type = "LH")
```

Hay disponibles siete tipos distintos de gráficas: tres histogramas mostrando las relaciones entre instancias y etiquetas, dos gráficas de barras con una finalidad similar, una gráfica circular indicando los tipos de atributos y una gráfica de concurrencia entre etiquetas. Seis de ellos aparecen en la Figura 5.1, generados con las órdenes mostradas a continuación:

```
> layout(matrix(c(1,1,2,1,1,3,4,5,6), 3, 3, byrow = TRUE))

> plot(emotions, type = "LC")
> plot(birds, type = "LH")
> plot(emotions, type = "LB")
> plot(birds, type = "CH")
> plot(emotions, type = "AT")
> plot(emotions, type = "LSB")
```

El histograma de etiquetas (tipo ''LH'') relaciona etiquetas e instancias de forma que se muestra cómo de bien representadas están las etiquetas en general. El eje X se asigna al número de instancias y el eje Y a la cantidad de etiquetas. Esto implica que si un gran número de etiquetas están apareciendo en unas pocas instancias, todos los datos se concentrarán en el lado izquierdo de la gráfica. Por el contrario, si las etiquetas están en general presentes en muchas instancias los datos

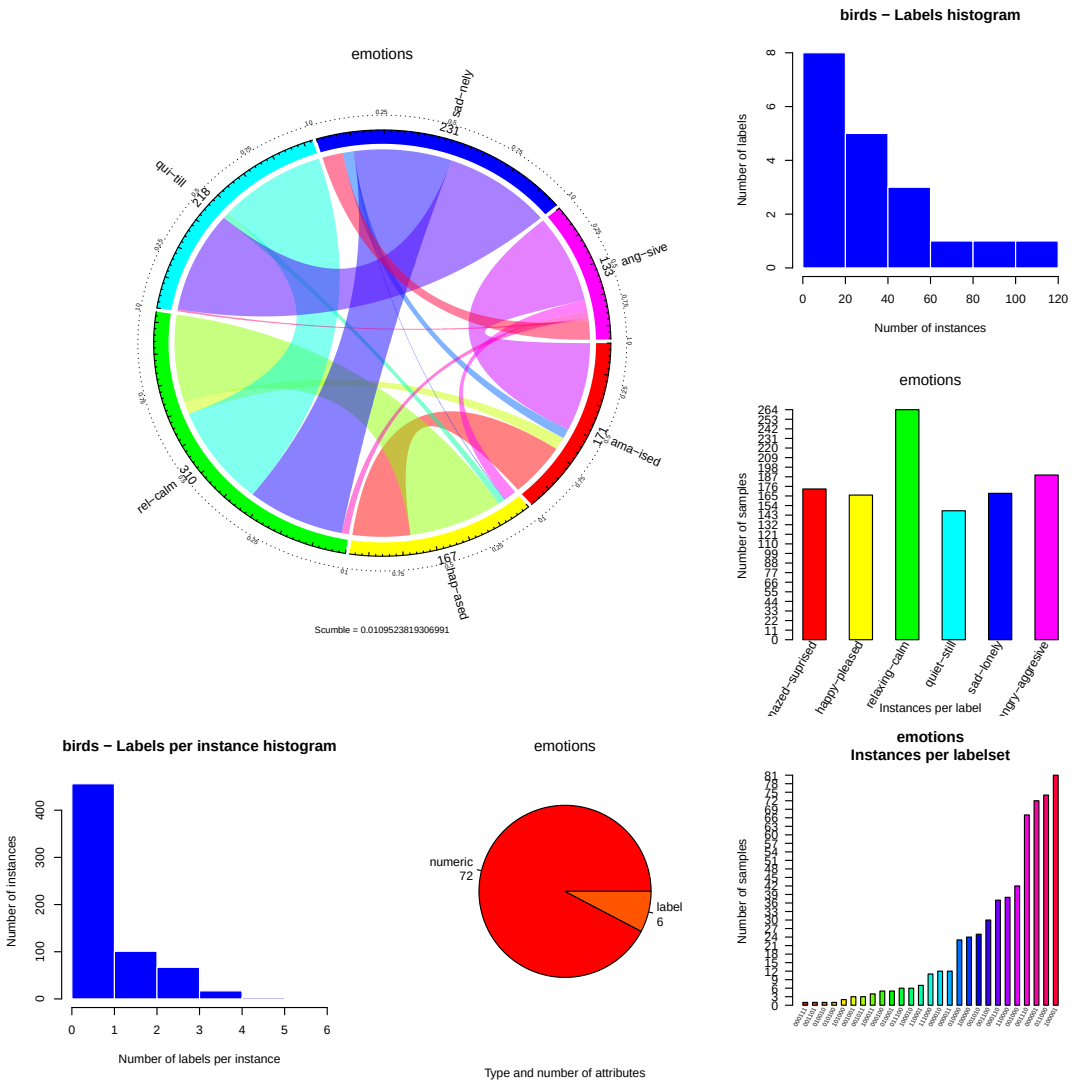


Figura 5.1: Algunas gráficas generadas por la función `plot()` del paquete mldr.

tenderán a acumularse en el lado derecho. Esta gráfica muestra el desbalanceo de las etiquetas cuando los datos se acumulan a ambos lados de la misma, lo que implica que muchas etiquetas están poco representadas y que también existen muchas muy frecuentes.

El histograma de conjuntos de etiquetas (tipo `''LSH''`) es similar al anterior en cierta manera. No obstante, en lugar de representar el número de etiquetas

presentes en las instancias lo que se muestra es el número de *labelsets*. Esto permite ver si los conjuntos de etiquetas se repiten consistentemente entre las instancias o no.

Las gráficas de barras de etiquetas y conjuntos de etiquetas muestran exactamente el número de instancias para cada etiqueta y *labelset*, respectivamente. Sus códigos son `''LB''` y `''LSB''`.

El histograma de cardinalidad (tipo `''CH''`) representa la cantidad de etiquetas presentes en las muestras, por lo que si los datos se acumulan en el lado derecho denotará que las instancias tienen una cantidad notable de etiquetas, mientras que si se concentran en la margen izquierda la situación sería la opuesta.

La gráfica de tipos de atributos (tipo `''AT''`) es una gráfica de tarta mostrando el número de etiquetas, de atributos numéricos y de atributos nominales, permitiendo apreciar la proporción entre los tipos de atributos y facilitando la comprensión de la cantidad de información de entrada y de salida.

La gráfica de concurrencia es la generada por defecto, con el tipo `''LC''`, y responde a la necesidad de explorar las interacciones entre etiquetas y, específicamente, entre etiquetas mayoritarias y minoritarias. Esta gráfica tiene forma circular, con la circunferencia particionada en mútiples arcos disjuntos que representan a las etiquetas. Cada arco tiene una longitud proporcional al número de instancias en que está presente la etiqueta. Estos arcos están a su vez divididos en bandas que los unen, mostrando las relaciones entre las etiquetas correspondientes. El ancho de cada banda indica la fuerza de la relación, al ser proporcional al número de instancias en que ambas etiquetas aparecen conjuntamente. De esta forma, una gráfica de concurrencia puede mostrar si hay etiquetas desbalanceadas que aparezcan conjuntamente.

Dado que dibujar la interacción entre un gran número de etiquetas puede producir resultados confusos, este último tipo de gráfica acepta más parámetros: `labelCount`, que acepta un entero que será usado para generar la gráfica con el número indicado de etiquetas tomadas aleatoriamente, y `labelIndices`, que

permite indicar exactamente los índices de las etiquetas a mostrar en la gráfica. Por ejemplo, para visualizar las 10 primeras etiquetas de `genbase`:

```
> plot(genbase, labelIndices = genbase$labels$index[1:10])
```

5.2.5. Funciones de transformación y filtrado

La manipulación de los conjuntos de datos es una tarea crucial en clasificación multietiqueta. Dado que la transformación es uno de los enfoques principal para abordar este problema, el paquete `mldr` implementa tanto la transformación BR como la LP. Estas pueden efectuarse mediante la función `mldr_transform`, que toma como primer parámetro el objeto `mldr` y como segundo el tipo de transformación: `''BR''` o `''LP''`. Opcionalmente puede facilitarse un vector de índices de etiquetas a ser incluidas en la transformación:

```
> emotionsbr <- mldr_transform(emotions, type = "BR")

> emotionslp <- mldr_transform(emotions, type = "LP",
                emotions$labels$index[1:4])
```

La transformación BR devolverá como resultado una lista de objetos de tipo `data.frame`, usando cada uno de ellos una de las etiquetas como clase, mientras que la transformación LP devolverá un único `data.frame` representando un conjunto de datos multiclase usando el *labelset* como clase. Estas dos transformaciones pueden ser usadas directamente para aplicar algoritmos de clasificación binarios y multiclase o implementar otros nuevos:

```
> emo_lp <- mldr_transform(emotions, "LP")
> library(RWeka)
> classifier <- IBk(classLabel ~ ., data = emo_lp,
            control = Weka_control(K = 10))
> evaluate_Weka_classifier(classifier, numFolds = 5)
```

```
=== 5 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         205               34.57   %
Incorrectly Classified Instances       388               65.43   %
Kappa statistic                          0.2695
Mean absolute error                      0.057
Root mean squared error                  0.1748
Relative absolute error                 83.7024 %
Root relative squared error             94.9069 %
Coverage of cases (0.95 level)          75.3794 %
Mean rel. region size (0.95 level)      19.574  %
Total Number of Instances              593
```

El paquete también ofrece una función de filtrado. Su uso es intuitivo ya que se usa con el operador `[]` habitual de R. Esto nos permite particionar un MLD o filtrarlo de acuerdo a cualquier condición de tipo lógico, por ejemplo:

```
> emotions$measures$num.instances
[1] 593

> emotions[emotions$dataset$.SCUMBLE > 0.01]$measures$num.instances
[1] 222
```

Combinándolo con el operador de unión, `+`, es posible implementar nuevas técnicas de preprocesamiento que modifiquen la información contenida en el MLD a fin de mejorar los resultados. Por ejemplo, el código siguiente sería una implementación básica del algoritmo REMEDIAL descrito al final del capítulo previo:

```
> mldbase <- mld[.SCUMBLE <= mld$measures$scumble]
```

```
 # Samples with coocurrence of highly imbalanced labels
> mldhigh <- mld[.SCUMBLE > mld$measures$scumble]
> majIndexes <- mld$labels[mld$labels$IRLbl <
           mld$measures$meanIR,"index"]

# Deactivate majority labels
> mldhigh$dataset[, majIndexes] <- 0
# Join the instances without changes with the filtered ones
> mldbase + mldhigh
```

En este último ejemplo las dos primeras órdenes filtran el MLD, separando las instancias que tienen un *SCUMBLE* inferior al promedio de aquellas que están por encima. Entonces la tercera línea obtiene los índices de las etiquetas con *IRLbl* por debajo de la media, estas son las etiquetas mayoritarias del MLD. Finalmente, estas etiquetas son puestas a 0 en las instancias con alto *SCUMBLE* y finalmente las dos particiones se unen.

Por último, otra útil característica incluida en el paquete `mldr` es el operador de comparación `==`. Este indica si dos MLD comparten la misma estructura, lo que significaría que tienen los mismos atributos y que estos son del mismo tipo:

```
> emotions[1:10] == emotions[20:30]
[1] TRUE

> emotions == birds
[1] FALSE
```

5.2.6. Evaluación del rendimiento predictivo

Asumiendo que se hubiesen obtenido predicciones de un clasificador para las instancias de un MLD, por ejemplo mediante un conjunto de clasificadores binarios, un clasificador multietiqueta o cualquier otro algoritmo, el paso siguiente

sería evaluar el rendimiento en clasificación de dicho algoritmo. El número de métricas para esta tarea, como se describió en el primer capítulo, es extenso y algunas de ellas son relativamente complejas de calcular. El paquete `mldr` facilita la función `mldr_evaluate` para realizar esta tarea.

Tomando como entrada la partición del MLD para la que se han obtenido predicciones y las propias predicciones, la función `mldr_evaluate` devuelve una lista de 20 medidas, incluyendo métricas basadas en ejemplos y en etiquetas. Por ejemplo:

```
> # Tomar las etiquetas reales de emotions
> predictions <- as.matrix(emotions$dataset[,emotions$labels$index])
> # e introducir algo de ruido
> predictions[sample(1:593, 100),sample(1:6, 100, replace = TRUE)]
        <- sample(0:1, 100,  replace = TRUE)
> # para evaluar el rendimiento predictivo
> res <- mldr_evaluate(emotions, predictions)
> str(res)

List of 20
 $ Accuracy        : num 0.917
 $ AUC             : num 0.916
 $ AveragePrecision: num 0.673
 $ Coverage        : num 2.71
 $ FMeasure        : num 0.952
 $ HammingLoss     : num 0.0835
 $ MacroAUC        : num 0.916
 $ MacroFMeasure   : num 0.87
 $ MacroPrecision  : num 0.829
 $ MacroRecall     : num 0.915
 $ MicroAUC        : num 0.916
 $ MicroFMeasure   : num 0.872
 $ MicroPrecision  : num 0.834
```

```
 $ MicroRecall      : num 0.914
 $ OneError         : num 0.116
 $ Precision        : num 0.938
 $ RankingLoss      : num 0.518
 $ Recall           : num 0.914
 $ SubsetAccuracy   : num 0.831
 $ ROC              :List of 15
  ...
```

```
> plot(res$ROC, main = "ROC curve for emotions") # Plot ROC curve
```

Esta lista incluye además de las medidas un miembro llamado `ROC`, con la información necesaria para dibujar la curva ROC tal y como se muestra en la última línea del ejemplo previo. El resultado sería una gráfica similar a la mostrada en la Figura 5.2.

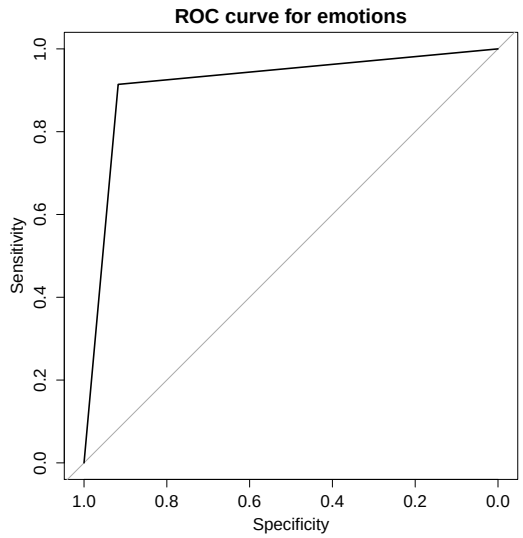


Figura 5.2: Curva ROC con los datos devueltos por `mldr_evaluate`.

5.3. La interfaz de usuario del paquete mldr

El paquete `mldr` ofrece a los usuarios una interfaz de usuario gráfica de tipo web, permitiendo la exploración interactiva y la obtención de gráficas y otros resultados. Una vez que se ha cargado el paquete `mldr`, para lanzar la interfaz gráfica de usuario no hay más que ejecutar la función `mldrGUI()`. Esta provocará que se abra el navegador web por defecto del sistema, mostrándose una interfaz estructurada en varias pestañas y un panel de contenidos.

Inicialmente se mostrará la sección `Main`, tal y como se muestra en la Figura 5.3. En ella hay opciones para seleccionar un MLD entre aquellos disponibles en memoria, así como cargar otros nuevos facilitando a la aplicación los archivos ARFF y XML. En la parte derecha se apilan varias gráficas. Estas muestran la cantidad de atributos de cada tipo (numéricos, nominales y etiquetas), la cantidad de etiquetas por instancia, la cantidad de instancias por etiqueta y el número de instancias por *labelset*. Cada gráfica puede guardarse como una imagen en el sistema de archivos. Justo debajo de las gráficas, varias tablas muestran algunas medidas básicas. La primera de ellas facilita medidas genéricas del MLD completo, tras lo cual se encuentran las medidas relativas a etiquetas, tales como *Card* y *Dens*. La última tabla muestra un resumen de medidas relativas a conjuntos de etiquetas.

La sección `Labels` contiene una tabla enumerando cada etiqueta presente en el MLD con los detalles y medidas más relevantes: su índice en la lista de atributos, el número de veces que aparece y frecuencia relativa, su *IRLbl* y *SCUMBLE*. Las etiquetas mostradas en la tabla pueden reordenarse usando las cabeceras, así como filtrarse y buscarse. Si es preciso, porque existan muchas etiquetas, la tabla se dividirá en varias páginas. Toda la información mostrada en las tablas puede ser exportada a distintos formatos de archivo. A la derecha de la tabla una gráfica muestra la cantidad de instancias en que está presente cada etiqueta. Esta es una gráfica interactiva, permitiendo al usuario ajustar el rango de etiquetas a mostrar.

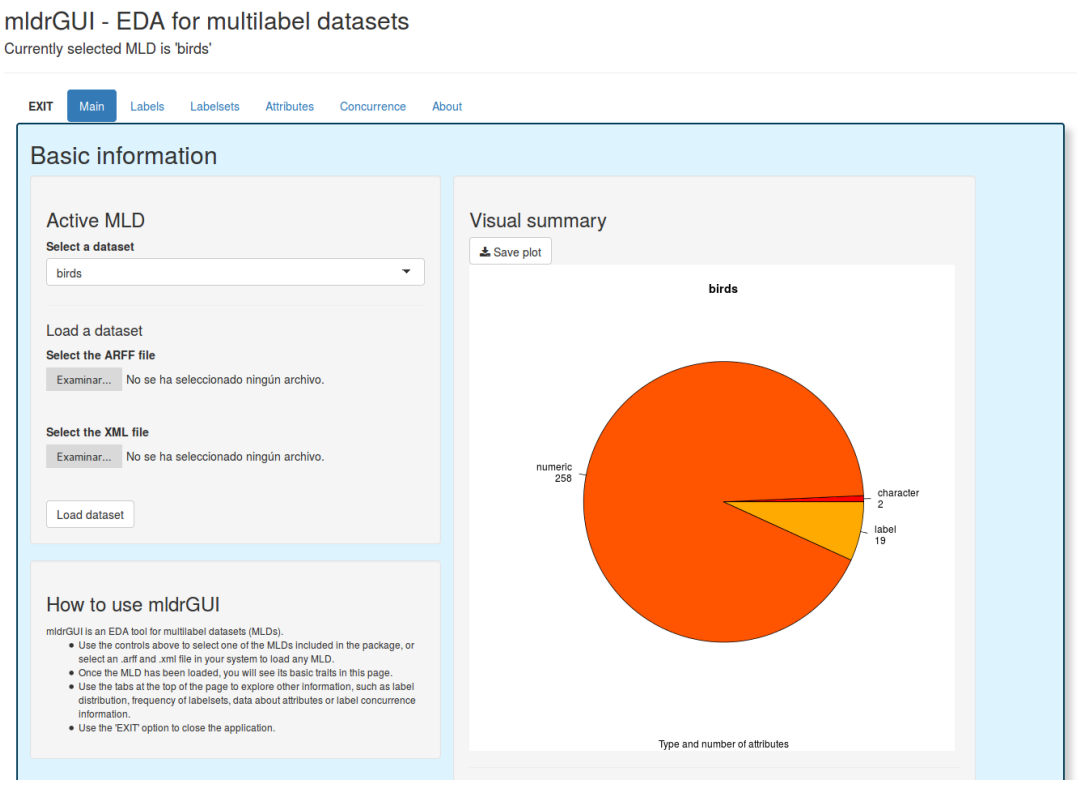


Figura 5.3: Página principal de la interfaz de usuario web.

Dado que las relaciones entre etiquetas pueden influir en el comportamiento de los algoritmos, el estudio de los *labelsets* es también un aspecto importante. Por ello la sección `Labelsets` ofrece información sobre los mismos, mostrando cada conjunto de etiquetas distinto y el número de veces que aparece. Esta lista también puede ser filtrada y reordenada, estando acompañada de una gráfica de barras representando la frecuencia de los *labelset*.

Finalmente, la sección `Concurrence` facilita una forma sencilla de visualizar la concurrencia entre etiquetas desbalanceadas (véase la Figura 5.4). La gráfica muestra la interacción entre las etiquetas seleccionadas en la tabla dispuesta a la izquierda. Por defecto se representarán las diez etiquetas con mayor *SCUMBLE*. El usuario puede marcar y desmarcar etiquetas a incluir en el gráfico haciendo clic sobre sus nombres.

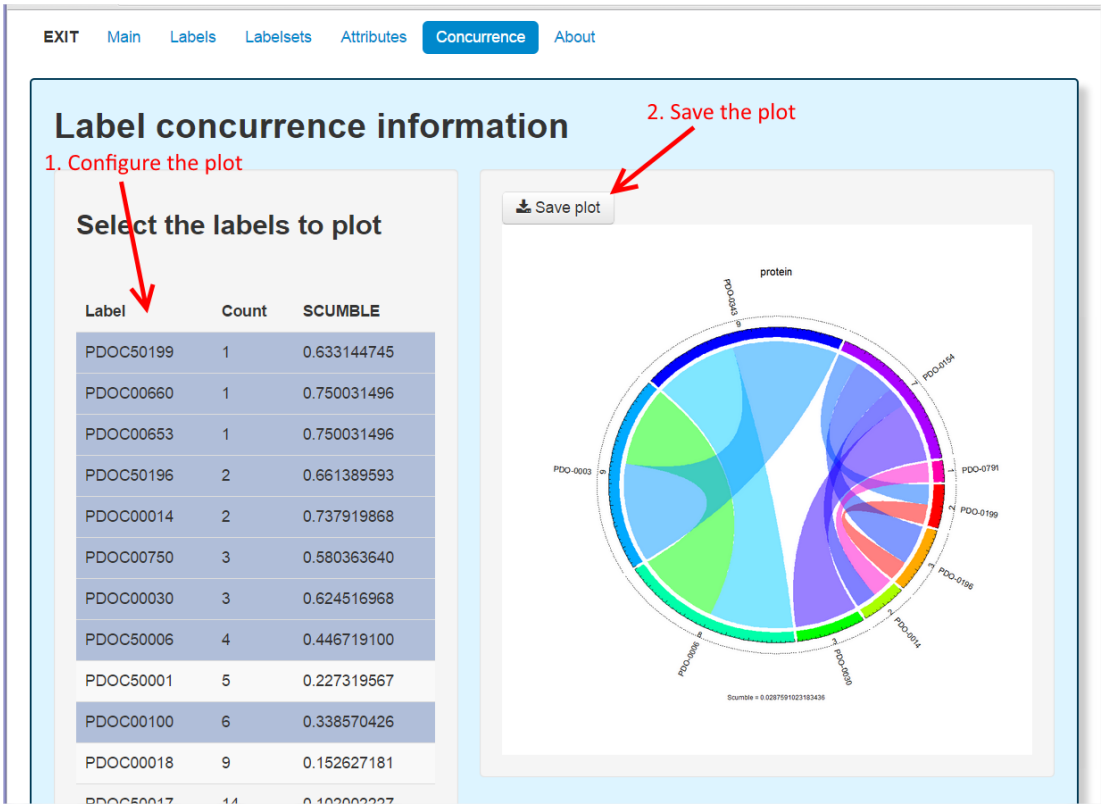


Figura 5.4: Las gráficas pueden personalizarse y almacenarse.

5.4. Conclusiones

Con el desarrollo del paquete `mldr` se ha conseguido construir una herramienta de trabajo útil no solo para parte de las propuestas presentadas en esta tesis, sino para la exploración y manipulación de conjuntos de datos multietiqueta en general desde un software de ciencia de datos tan popular como es R. Así, por ejemplo, están disponibles todas las métricas, tanto clásicas como las propuestas en esta tesis, que caracterizan un MLD, transformaciones tan populares como BR y LP, múltiples representaciones gráficas, etc.

En los dos primeros meses, desde su aceptación en el repositorio oficial CRAN, casi 600 usuarios instalaron el paquete `mldr` en sus equipos, con una media semanal que ha ido incrementándose paulatinamente desde las 50 descargas iniciales

a las más de 100 a finales de marzo. Se espera que a medida que el paquete vaya siendo más conocido dicho número siga creciendo, haciendo de `mldr` la opción natural para trabajar con datos multietiqueta en R.

A lo largo de este capítulo se ha mostrado la funcionalidad general que ofrece actualmente el paquete, poniendo especial énfasis en las opciones de exploración y su accesibilidad para usuarios no experimentados. El desarrollo del paquete continúa y se pretende que sea una plataforma de implementación de algoritmos de preprocesamiento, clasificación y tratamiento en general de conjuntos de datos multietiqueta.

5.5. Publicaciones asociadas a este trabajo

El paquete software `mldr` fue aceptado en CRAN, el repositorio oficial de paquetes de R, y está disponible en http://cran.r-project.org/web/packages/mldr/index.html bajo el título *"mldr: Exploratory Data Analysis and Manipulation of Multi-Label Data Sets"*.

El manual asociado al paquete ha sido publicado en Figshare (Charte and Charte [2015]), *F. Charte and F. D. Charte. "How to work with multilabel datasets in R using the mldr package". 03 2015. doi: 10.6084/m9.figshare.1356035.*

El artículo *F. Charte and F. D. Charte. "Working with Multilabel Datasets in R: The `mldr` Package"* se encuentra actualmente en fase de revisión en la revista *The R Journal*.

Capítulo 6

Conclusiones y futuras líneas de investigación

Este capítulo final tiene por objeto ofrecer una visión general del trabajo desarrollado a lo largo de la presente tesis, destacando los logros más importantes, enumerando las distintas publicaciones realizadas y resumiendo las líneas de investigación abiertas en las que se continuará trabajando en el futuro.

La primera fase del trabajo consistió en una revisión amplia de la bibliografía especializada en relación al tema de estudio: la clasificación multietiqueta. Esto nos permitió conocer con profundidad los aspectos fundamentales de esta tarea, así como determinar cuáles eran los problemas abiertos en los que se podría trabajar con el objetivo de ofrecer soluciones a los mismos. Con el objetivo genérico de mejorar el rendimiento en clasificación multietiqueta, se han definido los siguientes objetivos específicos:

- Reducir la dimensionalidad en el espacio de etiquetas a fin de reducir la complejidad de los clasificadores.

- Estudiar la propiedad del desbalanceo en bases de datos multietiqueta, con dos subobjetivos esenciales:

  - Proponer medidas de caracterización que permitan conocer el nivel de desbalanceo de cada conjunto de datos.
  - Definir métodos para reducir la ratio de desbalanceo en dichos conjuntos de datos, aprovechando sus características específicas allí donde sea posible.

- Facilitar el análisis exploratorio y manipulación de conjuntos de datos multietiqueta.

En la siguiente sección se ofrece una visión general de los resultados obtenidos, asociados a cada uno de esos objetivos.

6.1. Objetivos y resultados alcanzados

Los objetivos que se plantearon al inicio de la presente tesis han generado como resultado un importante número de nuevas métricas de caracterización, múltiples algoritmos y un software de propósito general para el trabajo con datos multietiqueta. Atendiendo a las tres vías de trabajo antes citadas, los resultados alcanzados son los resumidos a continuación.

6.1.1. Reducción de la dimensionalidad en el espacio de etiquetas

En el Capítulo 2 se describió una de las problemáticas que afecta a la mayor parte de los conjuntos de datos multietiqueta: la presencia de cientos o miles de etiquetas. La alta dimensionalidad en el espacio de salida es un problema inexistente en clasificación tradicional, al operarse siempre con una sola clase, pero resulta muy común en el campo multietiqueta. Esta dimensionalidad influye en la complejidad de la mayoría de clasificadores, al estar gran parte de ellos basados en técnicas de transformación como BR y LP que implican el entrenamiento de un gran número de modelos o un enorme número de combinaciones de etiquetas.

Las reglas de asociación son una herramienta que permite determinar relaciones entre elementos presentes en las transacciones de una base de datos. En el algoritmo LI-MLC, presentado en el citado capítulo, se propone el uso de las mismas a fin de extraer reglas de implicación que permiten eliminar del conjunto de datos original etiquetas con una fuerte correlación, facilitando así el trabajo de cualquier algoritmo de clasificación multietiqueta. Posteriormente las etiquetas eliminadas son restablecidas mediante la inferencia de las reglas de asociación antes obtenidas.

Como se demostró experimentalmente, la reducción de la dimensionalidad en el espacio de etiquetas al aplicar LI-MLC produce una mejora de los resultados de clasificación, incluso con significación estadística según la métrica de rendimiento empleada. Otro importante hito alcanzado es que, gracias a la reducción mencionada, el tiempo para completar la tarea de clasificación también se reduce, prácticamente en todas las ocasiones, con diferencias estadísticamente significativas.

Frente a otras propuestas existentes de reducción de dimensionalidad del espacio de etiquetas LI-MLC tiene la ventaja de no transformar el problema original en otro distinto, así como de resultar extremadamente sencillo de aplicar.

6.1.2. Análisis y propuestas para abordar el problema del desbalanceo en datos multietiqueta

Uno de los obstáculos más estudiados y para el que más propuestas existen en clasificación tradicional es el aprendizaje a partir de datos desbalanceados. Este problema adquiere mayor dificultad al trabajar con conjuntos de datos multietiqueta, ya que normalmente existen múltiples etiquetas poco frecuentes y varias que son muy frecuentes, con el problema añadido de que en ocasiones unas y otras aparecen conjuntamente en las mismas instancias. Este ha sido el tema abordado en los Capítulos 3 y 4 de la presente tesis.

A pesar de que la mayoría de trabajos publicados asumen la presencia de desbalanceo generalizado en los conjuntos de datos multietiqueta, no se habían definido métricas para evaluar esta característica y poder medirla empíricamente. Esta fue la primera tarea abordada en el Capítulo 3, proponiéndose métricas como *IRLbl*, *MeanIR* y *CVIR* que permitieron constatar la presencia de dicho problema. A continuación se exploró la vía del preprocesamiento de los datos para intentar equilibrar la distribución de las etiquetas, proponiéndose inicialmente cuatro algoritmos de remuestreo aleatorio, dos de ellos basados en la transformación LP: LP-ROS y LP-RUS, y otros dos en el análisis del nivel de desbalanceo individual por etiqueta: ML-ROS y ML-RUS. La experimentación ha demostrado que, en general, las técnicas de sobremuestreo producen mejores resultados que las de bajomuestreo, ya que estas últimas producen un pérdida de información potencialmente útil para el algoritmo de clasificación. Asimismo, los resultados reflejan que las propuestas basadas en la evaluación del nivel de desbalanceo individual de cada etiqueta, ML-RUS y ML-ROS, funcionan mejor que aquellas basadas en la transformación LP. La eliminación de instancias de ML-RUS mejora los resultados base con diferencias significativas cuando se usa la métrica *Micro-FMeasure*, más sensible a clases mayoritarias. La clonación de instancias de ML-ROS consigue lo mismo con la métrica *Macro-FMeasure*, más sensible a las clases minoritarias.

En el Capítulo 4 se presentaron desarrollos más avanzados, basados en técnicas heurísticas en lugar de aleatorias. El algoritmo MLSMOTE produce instancias sintéticas asociadas a etiquetas minoritarias. Las instancias que servirán como semilla para la generación de nuevas instancias se seleccionan usando las métricas de desbalanceo que propusimos en el capítulo previo, definiéndose asimismo varios métodos para la generación de los conjuntos de etiquetas sintéticos de las nuevas muestras. La experimentación efectuada demuestra que MLSMOTE es capaz de mejorar el rendimiento en clasificación de múltiples algoritmos, con significación estadística en muchos casos. Asimismo el sobremuestreo que lleva a cabo MLSMOTE es superior al de otras propuestas previas, tanto propias como de terceros, como LP-ROS, ML-ROS y SmoteUG, frente a los que consigue diferencias esta-

dísticamente significativas en la mayoría de los casos. Usado conjuntamente con un clasificador, MLSMOTE también alcanza mejores resultados que algoritmos específicamente diseñados para clasificación multietiqueta desbalanceada, como BR-IRUS y EML.

MLeNN es un algoritmo de selección heurística de instancias asociadas a etiquetas mayoritarias para su eliminación, un método más efectivo que la selección aleatoria. Siguiendo la regla ENN, el algoritmo compara cada muestra con sus vecinos más cercanos, usando para ello una distancia de Hamming adaptada al contexto multietiqueta. Aquellas instancias que difieren de la mitad o más de sus vecinos son eliminadas. Los resultados de experimentación que el preprocesamiento llevado a cabo por MLeNN mejora los resultados obtenidos por varios algoritmos de clasificación, con significación estadística en dos de las tres métricas empleadas. Asimismo funciona mejor que las técnicas de *undersampling* puramente aleatorias, como es el caso de LP-RUS.

Finalmente, continuando con el estudio del problema del desbalanceo en datos multietiqueta, se analiza el fenómeno de la concurrencia de etiquetas mayoritarias y minoritarias en las mismas instancias, una casuística que puede dificultar la tarea de clasificación. En este sentido se propone una nueva medida: *SCUMBLE*, mediante la cual es posible determinar la dificultad de un conjunto de datos multietiqueta para ser procesado mediante técnicas de remuestreo. Asociado a dicha medida se desarrolló el algoritmo REMEDIAL, con un enfoque de preprocesamiento diferente a todos los anteriores, específico para datos multietiqueta. Este método detecta, mediante la métrica *SCUMBLE*, qué instancias presentan etiquetas mayoritarias y minoritarias simultáneamente, procediendo a desplegarlas en dos muestras separadas. Por una parte se generan nuevas instancias y por otra se edita el conjunto de etiquetas de las existentes. La experimentación demuestra que este enfoque resulta apropiado para aquellos conjuntos de datos con un alto nivel de *SCUMBLE*, mejorando el rendimiento en clasificación en muchos casos dependiendo del clasificador utilizado.

6.1.3. Herramienta de análisis exploratorio de datos multietiqueta

Una parte del trabajo efectuado para alcanzar la consecución de los anteriores objetivos, definiendo nuevas métricas de caracterización y métodos de preprocesamiento, se ha apoyado en el desarrollo de una herramienta propia para la exploración de conjuntos de datos multietiqueta, el análisis de sus características y la manipulación de su contenido. Dicha herramienta es un paquete llamado `mldr` que permite operar con facilidad sobre conjuntos de datos multietiqueta desde R, uno de los lenguajes y entornos de trabajo más usados actualmente para tareas relacionadas con la ciencia de datos.

La estructura y funcionalidad del paquete `mldr` han sido descritos en el Capítulo 5, mostrándose múltiples ejemplos de uso que demuestran la facilidad con la que puede recuperarse un conjunto de datos multietiqueta desde un archivo, extraer sus características fundamentales, que incluyen todo tipo de métricas, tanto las clásicas definidas en la literatura como las propuestas en la presente tesis; representarlas gráficamente, transformar el conjunto con los métodos BR y LP, filtrar las instancias, etc.

El citado paquete está diseñado para que pueda ser utilizado por usuarios no familiarizados con R, gracias a la inclusión de una interfaz gráfica de usuario de tipo web. De esta forma, cualquiera interesado en explorar conjuntos de datos multietiqueta puede hacerlo directamente desde su navegador web, seleccionando opciones, configurando los datos mostrados en tablas y gráficas y almacenándolos para su posterior uso si lo precisa.

6.2. Publicaciones

Producto de los resultados que se han alcanzado, enumerados en la sección previa, esta tesis tiene asociadas las publicaciones enumeradas a continuación:

6.2.1. En revistas JCR

- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *LI-MLC: A Label Inference Methodology for Addressing High Dimensionality in the Label Space for Multilabel Classification*, IEEE Transactions on Neural Networks and Learning Systems, vol.25, no.10, pp.1842,1854, Oct. 2014. doi: 10.1109/TNNLS.2013.2296501.

- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *Addressing Imbalance in Multilabel Classification: Measures and Random Resampling Algorithms*, Neurocomputing (*Disponible online*). 2015. doi: 10.1016/j.neucom.2014.08.091.

- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *MLSMOTE: Approaching Imbalanced Multilabel Learning Through Synthetic Instance Generation*, Knowledge-Based Systems (En revisión).

- Charte, F., Charte, F.D., *Working with Multilabel Datasets in R: The mldr Package*, The R Journal (En revisión).

6.2.2. En congresos internacionales

- Rivera, A.J., Charte, F., Pérez-Godoy, M.D., del Jesus, M.J., *A First Approach to the Problem Transformation Methodology for Multi-label Classification*, Advances in Computational Intelligence, pp.41-48, 2011.
  doi: 10.1007/978-3-642-21501-8_6.

- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *Improving Multi-label Classifiers via Label Reduction with Association Rules*, Hybrid Artificial Intelligent Systems, vol.7209, pp.188-199. doi: 10.1007/978-3-642-28931-6_18

- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *A First Approach to Deal with Imbalance in Multi-label Datasets*, Hybrid Artificial Intelligent Systems, vol. 8073, pp.150-160. doi: 10.1007/978-3-642-40846-5_16

- Charte, F., Rivera, A.J., Pérez-Godoy, M.D., del Jesus, M.J., *Alternative OVA proposals for cooperative competitive RBFN design in classification tasks*, Advances in Computational Intelligence, vol.7902, pp.331-338. doi: 10.1007/978-3-642-38679-4_32
- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *Concurrence among Imbalanced Labels and its Influence on Multilabel Resampling Algorithms*, Hybrid Artificial Intelligent Systems, vol.8480, pp.110-121. doi: 10.1007/978-3-319-07617-1_10
- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *MLeNN: A First Approach to Heuristic Multilabel Undersampling*, Intelligent Data Engineering and Automated Learning, vol.8669, pp.1-9. doi: 10.1007/978-3-319-10840-7_1
- Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., *Resampling Multilabel Datasets by Decoupling Highly Imbalanced Labels*, Hybrid Artificial Intelligent Systems. Jun 2015.

6.2.3. Otros

- Charte, F., Charte, F.D., *How to work with multilabel datasets in R using the mldr package*, paquete software en CRAN, manual en Figshare, 2015. doi: 10.6084/m9.figshare.1356035

6.3. Líneas de trabajo futuras

A partir del trabajo desarrollado durante la elaboración de esta tesis se han ido abriendo nuevas vías de investigación que pueden dar lugar a líneas de trabajo futuras. A continuación se destacan las más importantes:

- **Clasificación multietiqueta basada en selección de instancias**: La experimentación con algoritmos de remuestreo, especialmente aquellos que

aplican *undersampling*, ha permitido apreciar la importancia de la pérdida de información que conlleva eliminar muestras en un conjunto de datos multietiqueta, ya que cada instancia puede ser representante de un gran número de etiquetas. La selección de la instancia adecuada, en el caso de los algoritmos MLeNN y MLSMOTE presentados en el Capítulo 4, permite mejorar de manera apreciable el comportamiento de los clasificadores. El paso siguiente sería diseñar un clasificador que aproveche toda esa experiencia para, mediante la selección de instancias, construir un modelo eficiente que tome en consideración las características específicas de este tipo de conjuntos de datos.

- **Restricción de predicciones mediante reglas de asociación negativas**: En el segundo capítulo se presentó el algoritmo LI-MLC, una hibridación de minería de reglas de asociación y clasificador multietiqueta capaz de mejorar la eficiencia sin afectar negativamente al rendimiento predictivo. Partiendo de esta base se plantearía mejorar este último factor obteniendo un conjunto de reglas de asociación extendido, en el que se incluyesen también reglas de asociación negativas, a fin de establecer restricciones en las predicciones facilitadas por el clasificador subyacente.

- **Métodos de preprocesamiento compuestos**: La parte final del cuarto capítulo demostró, mediante la métrica *SCUMBLE*, la dificultad que los algoritmos de remuestreo tradicionales encuentran al operar sobre conjuntos de datos multietiqueta. También se presentó un algoritmo alternativo de preprocesamiento, REMEDIAL, basado en la idea del desdoblamiento de aquellas instancias en las que coinciden etiquetas mayoritarias y minoritarias. El paso siguiente sería construir métodos de preprocesamiento compuestos, en los que se desdoblen las instancias cuando sea preciso antes de llevar a cabo un *oversampling* o *undersampling*, ya sea de tipo aleatorio o heurístico. Teóricamente los resultados generados por un algoritmo de ese tipo deberían mejorar a los ya obtenidos.

- **Multiclasificadores basados en información de concurrencia**: El número de algoritmos de clasificación multietiqueta basados en en *ensembles* es importante y sigue creciendo. La mayor parte de ellos son grupos de clasificadores binarios o multiclase, cuya construcción ignora por completo la información de correlación entre etiquetas, como ocurre con BR, o la integra mediante apilamiento o encadenamiento, como se describió en el Capítulo 1. Los multiclasificadores basados en la transformación LP, como es el caso de HOMER y RAkEL, incorporan esa información al diseño del *ensemble*. De manera análoga podría plantearse un modelo de multiclasificador cuya estructura se defina a partir de la información de concurrencia entre etiquetas aportada por la medida *SCUMBLE* u otras métricas definidas a propósito para tal fin. El objetivo sería contar con clasificadores especializados en las clases minoritarias y otros en las mayoritarias, cuyas predicciones se unirían para facilitar el resultado final.

- **Clasificación multietiqueta con redes neuronales de tipo *deep learning* y unidades RBF**: Las redes neuronales de tipo *deep learning*, como las RBM (*Restricted Boltzman Machines*) y las redes convolucionales, están demostrando su efectividad en áreas como el reconocimiento facial y de escenas que, al igual que ocurre con los conjuntos de datos multietiqueta, se caracterizan por una alta dimensionalidad en el espacio de entrada. Por otra parte, las redes neuronales con unidades RBF (*Radial Basis Functions*) son reconocidas como aproximadores universales con gran efectividad en clasificación. Nuestro objetivo es hibridar estas dos ideas definiendo un modelo de red neuronal con múltiples capas ocultas, en lugar de solo una como es habitual en las RBFN, aprovechando lo mejor de ambos campos.

- **Asistencia al etiquetado de entradas en foros de consulta**: Los foros de consulta en la web, como *Stack Exchange* o *Yahoo! Answers*, cada vez son más populares y diariamente son visitados por miles de personas que esperan encontrar una respuesta a sus dudas sobre cualquier tema. La mayoría de expertos que participan en estos foros, prestando su ayuda, no pueden leer el texto completo de cada entrada, por lo que suelen guiarse

por las etiquetas asignadas a las mismas. En consecuencia es de vital importancia usar las etiquetas adecuadas para obtener una respuesta. Este es un contexto claro de aplicación de un algoritmo de clasificación multietiqueta, en este caso integrado en una interfaz web en la que el usuario pudiese introducir su pregunta y obtener como sugerencia las etiquetas que debería asignarle a la entrada. Previamente sería preciso entrenar el clasificador partiendo del registro histórico de entradas con etiquetas asignadas manualmente, aplicando técnicas de minería de texto para generar un conjunto de datos multietiqueta a partir de las mismas.

Bibliografía

J. Agrawal, S. Agrawal, S. Kaur, and S. Sharma. An Investigation of Fuzzy PSO and Fuzzy SVD Based RBF Neural Network for Multi-label Classification. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, pages 677–687. Springer, 2014. 42, 55

R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994. 20, 82, 86

D. W. Aha. Editorial. In *Lazy Learning*, pages 7–10. Springer, 1997. 22, 23

R. Al-Otaibi, M. Kull, and P. Flach. Lacova: A tree-based multi-label classifier using label covariance as splitting criterion. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 74–79, Dec 2014. doi: 10.1109/ICMLA.2014.17. 40, 55

J. Alcala-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: Data set repository and integration of algorithms and experimental analysis framework. *J. of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011. 104

N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. 20

E. Alvares Cherman, J. Metz, and M. Monard. A Simple Approach to Incorporate Label Dependency in Multi-label Classification. In *Proc. 9th Mexican Int. Conf.*

on Artificial Intelligence, Pachuca, Mexico, MICAI'10, volume 6438, pages 33–43, 2010. doi: 10.1007/978-3-642-16773-7_3. 80, 111

E. Alvares-Cherman, J. Metz, and M. C. Monard. Incorporating label dependency into the binary relevance framework for multi-label classification. *Expert Syst. Appl.*, 39(2):1647 – 1655, 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.06.056. 36, 54, 111

A. B. Atkinson. On the measurement of inequality. *Journal of economic theory*, 2(3):244–263, 1970. 184

J.L. Ávila, E.L. Gibaja, and S. Ventura. Multi-label classification with gene expression programming. In *Hybrid Artificial Intelligence Systems*, pages 629–637. Springer, 2009. 49, 56

T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*, volume 1. CRC Press, 2000. 21

R Barandela, JS Sánchez, V García, and E Rangel. Strategies for learning in class imbalance problems. *Pattern Recognit.*, 36(3):849–851, 2003. 116

K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *J. Mach. Learn. Res.*, 3:1107–1135, 2003. ISSN 1532-4435. 30, 102

P. Barot and M. Panchal. Review on Various Problem Transformation Methods for Classifying Multi-Label Data. *International Journal of Data Mining And Emerging Technologies*, 4(2):45–52, 2014. 34

I. Ben-Gal. Outlier detection. In *Data Mining and Knowledge Discovery Handbook*, pages 131–146. Springer, 2005. 15

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188, 2001. 165

F. C. Bernardini, R. B. da Silva, R. M. Rodovalho, and E. B. M. Meza. Cardinality and density measures and their influence to multi-label learning methods. *Dens*, 1:1, 2014. 26

J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981. 22

M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern Recognit.*, 37(9):1757–1771, 2004. ISSN 00313203. doi: 10.1016/j.patcog.2004.03.009. 11, 31, 35, 36, 42, 55, 102, 103, 129, 164, 194

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984. 19

F. Briggs, B. Lakshminarayanan, L. Neal, X. Z. Fern, R. Raich, S. J. K. Hadley, A. S. Hadley, and M. G. Betts. Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *The Journal of the Acoustical Society of America*, 131(6):4640–4650, 2012. 30

S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. 1997 Int. Conf. on Management of data, Tucson, Arizona, USA, ACM SIGMOD'97*, pages 255–264, 1997. 85, 101

K. Brinker, J. Fürnkranz, and E. Hüllermeier. A unified model for multilabel classification and ranking. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29–September 1, 2006, Riva del Garda, Italy*, pages 489–493. IOS Press, 2006. 64

D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988. 21

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998. 23

J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan. Improving kNN multi-label classification in Prototype Selection scenarios using class proposals. *Pattern Recognition*, 2014. 45, 55

R. Cerri and A. C. de Carvalho. Hierarchical multilabel classification using top-down label combination and artificial neural networks. In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, pages 253–258. IEEE, 2010. 65

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research*, 7:31–54, 2006. 64

A. Chan and A. A. Freitas. A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In *Proc. 8th Annu. Conf. Genetic and Evolutionary Computation*, pages 27–34, New York, NY, USA, 2006. ACM Press. ISBN 1595931864. doi: 10.1145/1143997.1144002. 49, 56

F. Charte and F. D. Charte. How to work with multilabel datasets in R using the mldr package . 03 2015. doi: 10.6084/m9.figshare.1356035. 218

F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. Improving Multi-label Classifiers via Label Reduction with Association Rules. volume 7209 of *Lecture Notes in Computer Science*, chapter 18, pages 188–199. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-28930-9. doi: 10.1007/978-3-642-28931-6_18. 112

F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. A First Approach to Deal with Imbalance in Multi-label Datasets. In *Proc. 8th Int. Conf. Hybrid Artificial Intelligent Systems, Salamanca, Spain, HAIS'13*, volume 8073 of *LNCS*, pages 150–160, 2013. ISBN 978-3-642-40845-8. doi: 10.1007/978-3-642-40846-5_16. 143

F. Charte, A. Rivera, M. J. del Jesus, and F. Herrera. LI-MLC: A label inference methodology for addressing high dimensionality in the label space for multilabel

classification. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(10):1842–1854, Oct 2014a. ISSN 2162-237X. doi: 10.1109/TNNLS.2013.2296501. 112

F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. Concurrence among Imbalanced Labels and its Influence on Multilabel Resampling Algorithms. In *Proc. 9th Int. Conf. Hybrid Artificial Intelligent Systems, Salamanca, Spain, HAIS'14*, volume 8480 of *LNCS*, 2014b. ISBN 978-3-319-07616-4. 198

F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. MLeNN: A First Approach to Heuristic Multilabel Undersampling. In *Proc. 15th Int. Conf. Intelligent Data Engineering and Automated Learning, Salamanca, Spain, IDEAL'14*, volume 8669 of *LNCS*, pages 1–9, 2014c. ISBN 978-3-319-10839-1. doi: 10.1007/978-3-319-10840-7_1. 198

F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera. Addressing Imbalance in Multilabel Classification: Measures and Random Resampling Algorithms. *Neurocomputing*, 2015. doi: 10.1016/j.neucom.2014.08.091. 143

Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1): 1–6, 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007733. 116

K. Chen, B. Lu, and J.T. Kwok. Efficient Classification of Multi-label and Imbalanced Data using Min-Max Modular Classifiers. In *Int. Joint Conf. Neural Networks*, pages 1770–1775, 2006. doi: 10.1109/IJCNN.2006.246893. 123

X. Chen, Y. Zhan, J. Ke, and X. Chen. Complex video event detection via pairwise fusion of trajectory and multi-label hypergraphs. *Multimedia Tools and Applications*, pages 1–22, 2015. 32

J. Cheng and R. Greiner. Comparing bayesian network classifiers. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 101–108. Morgan Kaufmann Publishers Inc., 1999. 21

W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Mach. Learn.*, 76(2-3):211–225, 2009. doi: 10.1007/s10994-009-5127-5. 44, 55, 103, 124, 137, 164, 194

W. Cheng, E. Hüllermeier, and K. J. Dembczynski. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 279–286, 2010. 46, 56, 79

B. Chidlovskii. Multi-label wikipedia classification with textual and link features. In *Focused Retrieval and Evaluation*, pages 387–396. Springer, 2010. 32

B. Chizi and O. Maimon. Dimension reduction and feature selection. In *Data Mining and Knowledge Discovery Handbook*, pages 93–111. Springer, 2005. 16

D. A. Cieslak and N. V. Chawla. Start Globally, Optimize Locally, Predict Globally: Improving Performance on Imbalanced Data. In *Proc. 8th IEEE Int. Conf. on Data Mining, Pisa, Italy, ICDM'08*, pages 143–152, 2008. 97

A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In *Proc. 5th European Conf. Principles on Data Mining and Knowl. Discovery, Freiburg, Germany, PKDD'01*, volume 2168, pages 42–53, 2001. ISBN 978-3-540-42534-2. doi: 10.1007/3-540-44794-6_4. 39, 55

H. Cong and L. H. Tong. Grouping of triz inventive principles to facilitate automatic patent classification. *Expert Systems with Applications*, 34(1):788–795, 2008. 32

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. ISSN 0885-6125. doi: 10.1007/BF00994018. URL http://dx.doi.org/10.1007/BF00994018. 20

T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967. 20

K. Crammer and Y. Singer. A family of additive online algorithms for category ranking. *The Journal of Machine Learning Research*, 3:1025–1058, 2003. 35, 54

K. Crammer, M. Dredze, K. Ganchev, P. P. Talukdar, and S. Carroll. Automatic Code Assignment to Medical Text. In *Proc. Workshop on Biological, Translational, and Clinical Language Processing, Prague, Czech Republic, BioNLP'07*, pages 129–136, 2007. 29, 101

F. De Comité, R. Gilleron, and M. Tommasi. Learning Multi-label Alternating Decision Trees from Texts and Data. In *Proc. 3rd Int. Conf. Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany, MLDM'03,*, volume 2734, pages 35–49, 2003. ISBN 978-3-540-40504-7. doi: 10.1007/3-540-45065-3_4. 39, 55

Jan G De Gooijer and Rob J Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006. 19

K. Dembczynski, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence in multi-label classification. In *Workshop proceedings of learning from multi-label data*, pages 5–12. Citeseer, 2010. 79

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006. ISSN 1532-4435. 104

S. Dendamrongvit and M. Kubat. Undersampling Approach for Imbalanced Training Sets and Induction from Multi-label Text-Categorization Domains. In *New Frontiers in Applied Data Mining*, volume 5669 of *LNCS*, pages 40–52. Springer, 2010. ISBN 978-3-642-14639-8. doi: 10.1007/978-3-642-14640-4_4. 125

S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas. Protein Classification with Multiple Algorithms. In *Proc. 10th Panhellenic Conference on Informatics, Volos, Greece, PCI'05*, pages 448–456, 2005. doi: 10.1007/11573036_42. 11, 31, 102

M. Dorigo, G. Caro, and L. Gambardella. Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172, 1999. 21, 48

Norman Richard Draper, Harry Smith, and Elizabeth Pownell. *Applied regression analysis*, volume 3. Wiley New York, 1966. 18

S Dumais, G Furnas, T Landauer, S Deerwester, S Deerwester, et al. Latent semantic indexing. In *Proceedings of the Text Retrieval Conference*, 1995. 77

P. Duygulu, K. Barnard, J. de Freitas, and D. Forsyth. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In *Proc. 7th European Conf. on Computer Vision-Part IV, Copenhagen, Denmark, ECCV'02*, pages 97–112, 2002. doi: 10.1007/3-540-47979-1_7. 30, 102

Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003. 21

A. Elisseeff and J. Weston. A Kernel Method for Multi-Labelled Classification. In *Advances in Neural Information Processing Systems 14*, volume 14, pages 681–687. MIT Press, 2001. 31, 43, 55, 102

B. S. Everitt, S. Landau, M. Leese, and D. Stahl. Miscellaneous clustering methods. *Cluster Analysis, 5th Edition*, pages 215–255, 2011. 20

C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994. 19

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996. 12, 17

A. Fernández, V. López, M. Galar, M. J. del Jesus, and F. Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowl. Based Systems*, 42:97 – 110, 2013. ISSN 0950-7051. doi: 10.1016/j.knosys.2013.01.018. 116, 118, 154

C. Ferreira. Gene expression programming in problem solving. In *Soft Computing and Industry*, pages 635–653. Springer, 2002. 49

Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133, 1999. 39

T. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011. 18

J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73:133–153, 2008. ISSN 0885-6125. doi: 10.1007/s10994-008-5064-8. 35, 51, 54, 56, 103, 124, 137, 164, 179

M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognit.*, 44(8): 1761–1776, 2011. doi: 10.1016/j.patcog.2011.01.017. 118

M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging, boosting, and hybrid-based approaches. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, July 2012. ISSN 1094-6977. doi: 10.1109/TSMCC.2011.2161285. 119

S. Garcıa and F. Herrera. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *J. Mach. Learn. Res.*, 9 (2677-2694):66, 2008. 137

S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sciences*, 180(10):2044 – 2064, 2010. ISSN 0020-0255. doi: 10.1016/j.ins.2009.12.010. 137

S. García, J. Luengo, and F. Herrera. *Data Preprocessing in Data Mining*. Springer, 2015. 14

V. García, J.S. Sánchez, and R.A. Mollineda. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowl. Based*

Systems, 25(1):13 – 21, 2012. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2011.06.013. 118, 141, 175

X. Genga, Y. Tanga, Y. Zhua, and G. Chengb. An Improved Multi-label Classification Algorithm BRkNN. *Journal of Information & Computational Science*, 11(16):5927?–5936, 2014. 45, 55

N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, 2005. 45, 55, 57

E. Gibaja and S. Ventura. Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(6):411–444, 2014. ISSN 1942-4795. doi: 10.1002/widm.1139. URL http://dx.doi.org/10.1002/widm.1139. 54, 66

A. F. Giraldo-Forero, J. A. Jaramillo-Garzón, J. F. Ruiz-Muñoz, and C. G. Castellanos-Domínguez. Managing imbalanced data sets in multi-label problems: A case study with the SMOTE algorithm. In *Progress in Pattern Recognit., Image Analysis, Computer Vision, and Applications*, volume 8258 of *LNCS*, pages 334–342. Springer, 2013. ISBN 978-3-642-41821-1. doi: 10.1007/978-3-642-41822-8_42. 125, 155, 157, 163, 183

S. Godbole and S. Sarawagi. Discriminative Methods for Multi-Labeled Classification. In *Advances in Knowl. Discovery and Data Mining*, volume 3056, pages 22–30, 2004. doi: 10.1007/978-3-540-24775-3_5. 35, 54, 57, 79, 103, 179

B. Goethals. Frequent set mining. In *Data mining and knowledge discovery handbook*, pages 377–397. Springer, 2005. 20

E. C. Gonçalves, A. Plastino, and A. A. Freitas. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 469–476. IEEE, 2013. 48, 56

R. Grodzicki, J. Mańdziuk, and L. Wang. Improved multilabel classification with neural networks. In *Parallel Problem Solving from Nature–PPSN X*, pages 409–416. Springer, 2008. 41, 55

R. A. Groeneveld and G. Meeden. Measuring skewness and kurtosis. *The Statistician*, 33(4):391–399, 1984. ISSN 00390526. doi: 10.2307/2987742. 98

J. W. Grzymala-Busse and W. J. Grzymala-Busse. Handling missing attribute values. In *Data mining and knowledge discovery handbook*, pages 33–51. Springer, 2010. 15

Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *Proc. 22th Int. Joint Conf. on Artificial Intelligence*, volume 2 of *IJCAI'11*, pages 1300–1305, 2011. ISBN 978-1-57735-514-4. 80, 81

T. M. Ha and H. Bunke. Off-line, handwritten numeral recognition by perturbation method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5):535–539, 1997. 154

J. Han, J. Pei, Y. Yin, and R. Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Min. Knowl. Disc.*, 8 (1):53–87, 2004. ISSN 1384-5810. doi: 10.1023/B:DAMI.0000005258.31418.83. 19, 87

P. Hart. The condensed nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 14(3):515–516, May 1968. ISSN 0018-9448. doi: 10.1109/TIT.1968.1054155. 20

H. He and Y. Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE, 2013. ISBN 978-1-118-07462-6. 118

J. He, H. Gu, and W. Liu. Imbalanced multi-modal multi-label learning for subcellular localization prediction of human proteins with both single and multiple sites. *PloS one*, 7(6):7155, 2012. doi: 10.1371/journal.pone.0037155. 116, 122

J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explor. Newsl.*, 2(1):58–64, 2000. ISSN 1931-0145. doi: 10.1145/360402.360421. 81, 82

F. Höppner. Association rules. In *Data Mining and Knowledge Discovery Handbook*, pages 299–319. Springer, 2010. 19

D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *Proc. 22th Annu. Conf. Advances in Neural Information Processing Systems, NIPS'09*, volume 22, pages 772–780, 2009. 81, 91, 92

E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916, 2008. doi: 10.1016/j.artint.2008.08.002. 35, 50, 54, 56

J.S.R. Jang. Structure determination in fuzzy modeling: a fuzzy cart approach. In *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*, pages 480–485. IEEE, 1994. 22

N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, 2002. ISSN 1088-467X. 116, 118, 154

I. Jolliffe. Introduction. In *Principal Component Analysis*, pages 1–7. Springer, 1986. 19

I. Jolliffe. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2005. ISBN 9780470013199. doi: 10.1002/0470013192.bsa501. 77, 123

I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel Text Classification for Automated Tag Suggestion. In *Proc. ECML PKDD'08 Discovery Challenge, Antwerp, Belgium*, pages 75–83, 2008. 11, 29, 101

J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010. 21

T.M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Supervised neural network modeling: An empirical investigation into learning from imbalanced data with labeling errors. *IEEE Trans. Neural Netw. Learn. Syst*, 21(5):813–830, May 2010. ISSN 1045-9227. doi: 10.1109/TNN.2010.2042730. 118

B. Klimt and Y. Yang. The Enron Corpus: A New Dataset for Email Classification Research. In *Proc. ECML'04, Pisa, Italy*, pages 217–226. 2004. doi: 10.1007/978-3-540-30115-8_22. 29, 101

D. Kocev, C. Vens, J. Struyf, and S. Dzeroski. Ensembles of multi-objective decision trees. *Lecture notes in computer science*, 4701:624–631, 2007. 52, 56

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982. 21

Y. Kongsorot and P. Horata. Multi-label classification with extreme learning machine. In *Knowledge and Smart Technology (KST), 2014 6th International Conference on*, pages 81–86. IEEE, 2014. 41, 55

S. B. Kotsiantis. Supervised machine learning: A review of classification techniques, 2007. 18, 22

S. B. Kotsiantis and P. E. Pintelas. Mixture of expert agents for handling imbalanced data sets. *Annals of Mathematics, Computing & Teleinformatics*, 1: 46–55, 2003. ISSN 1109-9305. 118, 119

P. Langley et al. *Selection of relevant features in machine learning*. Defense Technical Information Center, 1994. 16

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004. 29

C. Li and G. Shi. Improvement of Learning Algorithm for the Multi-instance Multi-label RBF Neural Networks Trained with Imbalanced Samples. *J. Inf. Sci. Eng.*, 29(4):765–776, 2013. 122

X. Li, F. Zhao, and Y. Guo. Conditional restricted boltzmann machines for multi-label learning with incomplete labels. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 635–643, 2015. 47, 56

Q. Liang, Z. Wang, Y. Fan, C. Liu, X. Yan, C. Hu, and H. Yao. Multi-label classification based on particle swarm algorithm. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pages 421–424. IEEE, 2013. 49, 56

M. Lin, K. Tang, and X. Yao. Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Trans. Neural Netw. Learn. Syst*, 24(4):647–660, April 2013. ISSN 2162-237X. doi: 10.1109/TNNLS.2012.2228231. 118

C. X. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79, 1998. 154

V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Inf. Sciences*, 250:113 – 141, 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2013.07.007. 116, 118, 154

B. Lu and M. Ito. Task decomposition and module combination based on class relations: a modular neural network for pattern classification. *IEEE Trans. Neural Networks*, 10(5):1244–1256, 1999. ISSN 1045-9227. doi: 10.1109/72.788664. 123

O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intell.*, 1 (4):303–313, 2012. 36, 171

J. Luengo, S.r García, and F. Herrera. A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions

and non-parametric tests. *Expert Systems with Applications*, 36(4):7798–7808, 2009. 104

B. Ma, W. Liu, and Y. Hsu. Integrating classification and association rule mining. In *Proceedings of the 4th*, 1998. 82

G. Madjarov, D. Gjorgjevikj, and S. Džeroski. Dual layer voting method for efficient multi-label classification. In *Pattern Recognition and Image Analysis*, pages 232–239. Springer, 2011. 54, 56

G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognit.*, 45 (9):3084 – 3104, 2012. ISSN 00313203. doi: 10.1016/j.patcog.2012.03.004. 54

J. L. McClelland, D. E. Rumelhart, PDP Research Group, et al. Parallel distributed processing. *Explorations in the microstructure of cognition*, 2:216–271, 1986. 21

E. L. Mencia and J. Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer, 2008. 29

E. L. Mencía, S. Park, and J. Fürnkranz. Efficient voting prediction for pairwise multilabel classification. *Neurocomputing*, 73(7):1164–1176, 2010. 54, 56

R. S. Michalski. *A theory and methodology of inductive learning*. Springer, 1983. 22

S. Mitra and Y. Hayashi. Neuro-fuzzy rule generation: survey in soft computing framework. *Neural Networks, IEEE Transactions on*, 11(3):748–768, 2000. 22

S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998. 23

V. C. Nitesh, W. B. Kevin, O. H. Lawrence, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *J. Artificial Intelligence Res.*, 16: 321–357, 2002. doi: 10.1613/jair.953. 118, 119, 154, 155, 159

C. H. Park and M. Lee. On applying linear discriminant analysis for multi-labeled problems. *Pattern recognition letters*, 29(7):878–887, 2008. 77

S. Park and J. Fürnkranz. Multi-label classification with label constraints. In *Proc. ECML PKDD'08 Workshop on Preference Learning, Antwerp, Belgium, PL'08*, pages 157–171, 2008. 80

R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, 6 (4):321–332, 2002. 48

F. Provost and T. Fawcett. Robust classification for imprecise environments. *Mach. Learn.*, 42:203–231, 2001. ISSN 0885-6125. doi: 10.1023/A:1007601015854. 118

W. Qu, Y. Zhang, J. Zhu, and Q. Qiu. Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In *Advances in Machine Learning*, pages 308–321. Springer, 2009. 65

J. R. Quevedo, O. Luaces, and A. Bahamonde. Multilabel classifiers with a probabilistic thresholding strategy. *Pattern Recognition*, 45(2):876–883, 2012. 54

J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 19

J. R. Quinlan. C4. 5: Programs for machine learning. 1993. 19, 38

F. Thabtah R. Alazaidah and Q. Al-Radaideh. A multi-label classification approach based on correlations among labels. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 6(2), 2015. doi: 10.14569/IJACSA.2015.060208. 49, 56

J. Read and P. Reutemann. MEKA multi-label dataset repository. URL http://meka.sourceforge.net/#datasets. 102

J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 995–1000. IEEE, 2008. 37, 51, 55, 56, 90, 123, 124

J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Mach. Learn.*, 85:333–359, 2011. ISSN 0885-6125. doi: 10.1007/s10994-011-5256-5. 50, 54, 56, 79, 80, 81, 103

J. Read, L. Martino, P. M. Olmos, and D. Luengo. Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, 2015. 53, 56

L. Rokach. A survey of clustering algorithms. In *Data mining and knowledge discovery handbook*, pages 269–298. Springer, 2010. 19

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999. 39

R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000. 38, 54, 57

K. Sechidis, G. Tsoumakas, and I. Vlahavas. On the stratification of multi-label data. In *Mach. Lear. Knowl. Disc. Datab.*, pages 145–158. Springer, 2011. doi: 10.1007/978-3-642-23808-6_10. 130

D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall, 2003. 104, 165, 175

C. Silverstein, S. Brin, and R. Motwani. Beyond Market Baskets: Generalizing Association Rules to Dependence Rules. *Data Min. Knowl. Disc.*, 2(1):39–68, 1998. 85

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. 1986. 47

C. G. M. Snoek, M. Worring, J. C. van Gemert, J. M. Geusebroek, and A. W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proc. 14th Annu. ACM Int. Conf. on Multimedia, Santa Barbara, CA, USA, MULTIMEDIA'06*, pages 421–430, 2006. ISBN 1-59593-447-2. doi: 10.1145/1180639.1180727. 31, 102

T. Sobol-Shikler and P. Robinson. Classification of complex information: Inference of co-occurring affective states from their expressions in speech. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1284–1297, 2010. 32

E. Spyromitros, G. Tsoumakas, and I. Vlahavas. An empirical study of lazy multilabel classification algorithms. In *Artificial Intelligence: Theories, Models and Applications*, pages 401–406. Springer, 2008. 45, 55

B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010. 32

A. N. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace Conference*, pages 3853–3862. IEEE, 2005. doi: 10.1109/AERO.2005.1559692. 30

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Commun. ACM*, 29 (12):1213–1228, 1986. ISSN 0001-0782. doi: 10.1145/7902.7906. 159

L. E. Sucar, C. Bielza, E. F. Morales, P. Hernandez-Leal, J. H. Zaragoza, and P. Larrañaga. Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41:14–22, 2014. 47, 56

L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–676. ACM, 2008. 47, 56

Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009. 116

M. A. Tahir, J. Kittler, and A. Bouridane. Multilabel classification using heterogeneous ensemble of multi-label classifiers. *Pattern Recognit. Lett.*, 33(5): 513–523, 2012a. doi: 10.1016/j.patrec.2011.10.019. 53, 56, 116, 124, 163

M. A. Tahir, J. Kittler, and F. Yan. Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognit.*, 45(10):3738–3750, 2012b. ISSN 0031-3203. doi: 10.1016/j.patcog.2012.03.014. 116, 124, 163

F. Tai and H.T. Lin. Multi-label classification with principle label space transformation. In *Proc. 2nd Int. Workshop on Learning from Multi-Label Data, Haifa, Isreal, MLD'10*, pages 45–52, 2010. 80

J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications. Editor: Charu Aggarwal, CRC Press In Chapman & Hall/CRC Data Mining and Knowledge Discovery Series*, 2014. 16

L. Tang, S. Rajan, and V. K. Narayanan. Large scale multi-label classification via metalabeler. In *Proc. 18th Int. Conf. on World Wide Web*, WWW '09, pages 211–220, 2009. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526738. 137, 164

L. Tenenboim-Chekina, L. Rokach, and B. Shapira. Identification of label dependencies for multi-label classification. In *Working Notes of the Second International Workshop on Learning from Multi-Label Data*, pages 53–60, 2010. 37, 53, 55, 56, 80, 81

G. Tepvorachai and C. Papachristou. Multi-label imbalanced data enrichment process in neural net classifier training. In *IEEE Int. Joint Conf. on Neural*

Networks, 2008. IJCNN, pages 1301–1307, 2008. doi: 10.1109/IJCNN.2008.4633966. 123

G. Tsoumakas and I. Katakis. Multi-label classification: An overview. In *Int J Data Warehousing and Mining*, 2007. 25

G. Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Proc. 18th European Conf. on Machine Learning, Warsaw, Poland, ECML'07*, volume 4701, pages 406–417, 2007. doi: 10.1007/978-3-540-74958-5_38. 52, 56, 57, 80, 89, 103, 123, 124, 130, 137, 164, 179

G. Tsoumakas, E. S. Xioufis, J. Vilcek, and I. Vlahavas. MULAN multi-label dataset repository. URL http://mulan.sourceforge.net/datasets.html. 102

G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proc. ECML/PKDD Workshop on Mining Multidimensional Data, Antwerp, Belgium, MMD'08*, pages 30–44, 2008. 56, 80, 89, 101, 103, 123, 130, 137, 164, 194

G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, and I. Vlahavas. Correlation based pruning of stacked binary relevance models for Multi-Label learning. In *Proc. 1st Int. Workshop Learning from Multi-Label Data, Bled, Slovenia, MLD'09*, pages 101–116, 2009. 36, 54, 80

G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining Multi-label Data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, chapter 34, pages 667–685. Springer US, Boston, MA, 2010. ISBN 978-0-387-09822-7. doi: 10.1007/978-0-387-09823-4_34. 23, 26, 33, 34, 35, 57

D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic Annotation and Retrieval of Music and Sound Effects. *IEEE Audio, Speech, Language Process.*, 16(2):467–476, 2008. ISSN 1558-7916. doi: 10.1109/TASL.2007.913750. 30, 102

N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 721–728, 2002. 46, 55

A. Veloso, W. Meira Jr, M. Gonçalves, and M. Zaki. Multi-label lazy associative classification. In *Knowledge Discovery in Databases: PKDD 2007*, pages 605–612. Springer, 2007. 48, 56

J. R. Vergara and P. A. Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, 2014. 16

J. Wang, J. Feng, X. Sun, S. Chen, and B. Chen. Simplified Constraints Rank-SVM for Multi-label Classification. In *Pattern Recognition*, pages 229–236. Springer, 2014. 43, 55

J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *Proc. 16th Annu. Conf. Advances in Neural Information Processing Systems, NIPS'02*, volume 15, pages 873–880, 2002. 81, 90

A. Wieczorkowska, P. Synak, and Z. Raś. Multi-Label Classification of Emotions in Music. In *Intelligent Information Processing and Web Mining*, volume 35 of *AISC*, chapter 30, pages 307–315. 2006. ISBN 978-3-540-33520-7. doi: 10.1007/3-540-33521-8_30. 11, 31, 102

D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-2(3):408 –421, 1972. ISSN 0018-9472. doi: 10.1109/TSMC.1972.4309137. 155

Q. Wu, Y. Ye, H. Zhang, T.W.S. Chow, and S.-S. Ho. ML-TREE: A tree-structure-based approach to multilabel learning. *IEEE Trans. Neural Netw. Learn. Syst*, to be published(Available online):doi: 10.1109/TNNLS.2014.2315296, 2014. ISSN 2162-237X. doi: 10.1109/TNNLS.2014.2315296. 39, 55

Jianhua Xu. Fast multi-label core vector machine. *Pattern Recognition*, 46(3): 885–898, 2013. 43, 55

Y. Yang, G. I. Webb, and X. Wu. Discretization methods. In *Data mining and knowledge discovery handbook*, pages 113–130. Springer, 2005. 16

L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965. 21

J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga. Bayesian chain classifiers for multidimensional classification. In *IJCAI*, volume 11, pages 2192–2197, 2011. 66

G. P. Zhang. Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4): 451–462, 2000. 23

M. Zhang. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Trans. Knowl. Data Eng.*, 18(10):1338–1351, 2006. ISSN 1041-4347. doi: 10.1109/TKDE.2006.162. 40, 55

M. Zhang. Ml-rbf : RBF Neural Networks for Multi-label Learning. *Neural Process. Lett.*, 29:61–74, 2009. doi: 10.1007/s11063-009-9095-3. 41, 55

M. Zhang and Z. Wang. "mimlrbf: {RBF} neural networks for multi-instance multi-label learning ". *Neurocomputing*, 72(16?18):3951 – 3956, 2009. ISSN 0925-2312. doi: 10.1016/j.neucom.2009.07.008. 122

M. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *Proc. 16th Int. Conf. on Knowledge Discovery and Data Mining, Washington, DC, USA, ACM SIGKDD'10*, pages 999–1008, 2010. ISBN 978-1-4503-0055-1. 80, 81

M. Zhang and Z. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.*, 40(7):2038–2048, 2007. ISSN 00313203. doi: 10.1016/j.patcog.2006.12.019. 44, 55, 124

M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 26(8):1819–1837, Aug 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2013.39. 54, 79

S. Zhang, C. Zhang, and Q. Yang. Data preparation for data mining. *Applied Artificial Intelligence*, 17(5-6):375–381, 2003. 14

T. Zhou, D. Tao, and X. Wu. Compressed labeling on distilled labelsets for multi-label learning. *Mach. Learn.*, 88(1-2):69–126, 2012. 81, 92, 104

Z. Zhou and M. Zhang. Multi-instance multi-label learning with application to scene classification. In *Advances in neural information processing systems*, pages 1609–1616, 2006. 65

X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. 17