
UNIVERSIDAD
DE GRANADA

**Departamento de Ciencias de la Computación e Inteligencia Artificial**

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

# Pre-procesamiento de datos para aprendizaje de Distribución de Etiquetas

**Tesis Doctoral**

Presenta:
**Manuel González López**

Directores:
**Dr. Salvador García López**
**Dr. José Ramón Cano de Amo**

Granada, Enero 2021

**Editor:** Universidad de Granada. Tesis Doctorales
**Autor:** Manuel González López
**ISBN:** 978-84-1306-830-5
**URI:** http://hdl.handle.net/10481/68012

*¿Heredarán los robots la Tierra?*
*Sí, pero serán nuestros hijos.*

MARVIN MINSKY

# Agradecimientos

He de comenzar agradeciendo a mis directores, Salva y José Ramón, sin ellos este trabajo de tesis doctoral no hubiera sido posible. Gracias a ambos por el tiempo dedicado y por vuestros valiosos consejos. Ha sido especialmente emotivo volver a encontrarme con Salva, con el que ya compartí mi primera etapa universitaria. Me ha complacido observar de primera mano el grandísimo investigador y profesor en el que sea convertido desde entonces.

Quiero extender mis agradecimientos a la empresa con la que llevo colaborando más de 10 años, NGA Human Resources - Alight. La idea de realizar un doctorado siempre ha estado en mi cabeza pero ellos han sido los detonantes para poder hacer realidad ese objetivo, ofreciéndome los recursos y tiempo necesarios para llevar a buen puerto esta aventura. Estoy seguro de que los conocimientos adquiridos a lo largo de mi formación darán un valor añadido una vez aplicados a la empresa.

Como no, agradecer a Ángela, por su apoyo incondicional desde el principio. Estando siempre a mi lado en los momentos buenos y en los no tan buenos, haciéndome mirar hacia adelante y siendo mi válvula de escape cuando más lo necesitaba. Has hecho este camino mucho más fácil. Prometo compensarte los fines de semana que hemos tenido que sacrificar a lo largo de estos últimos años.

Por último, doy gracias a mis padres y hermana por estar siempre ahí y por haberme transmitido, entre otras muchas cosas, el valor del esfuerzo. Sois mis pilares fundamententales en la vida.

De nuevo, gracias a todos.

# Índice general

# Capítulo 1

# Introducción

Los avances tecnológicos de los últimos años han promovido la generación y almacenamiento de una ingente cantidad de datos provenientes de los entornos comerciales, industriales, científicos y sociales. El uso inteligente de estos datos puede suponer una gran ventaja competitiva, lo que ha provocado un creciente interés por el desarrollo de metodologías que permitan la extracción de conocimiento útil a partir de ellos. Esta avalancha de datos es intratable por el ser humano y por ello requiere de métodos de análisis automatizados; hecho que ha llevado al surgimiento de una disciplina cada vez más presente en la Era de la Información: la Minería de Datos.

La minería de datos es, en términos generales, el proceso que intenta descubrir patrones e información oculta en grandes volúmenes de datos [63, 79]. Es la etapa de análisis del proceso más general de descubrimiento de conocimiento en bases de datos (*Knowledge Discovery in Databases* o KDD) [84]. La premisa básica para aprender de los datos es el uso de un conjunto de observaciones para descubrir un proceso subyacente. Es una hipótesis muy amplia, y difícil de encajar en un marco único. Como resultado, han surgido diferentes paradigmas de aprendizaje para hacer frente a diferentes situaciones y diferentes supuestos.

- Cuando las observaciones contienen ejemplos explícitos de cuál debería ser el resultado correcto para determinadas entradas, entonces nos encontramos dentro del entorno de **aprendizaje supervisado**. El aprendizaje es supervisado en el sentido de que algún "supervisor" se ha tomado la molestia de analizar cada entrada y asignarle una salida.

- En el **aprendizaje no supervisado**, las observaciones no contienen ninguna información de salida. El aprendizaje no supervisado puede verse como la tarea de encontrar "patrones interesantes" en los datos de entrada. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori, se tratan los objetos de entrada como un conjunto de variables aleatorias, construyendo un modelo de densidad para el conjunto de datos.

Las técnicas de **clasificación** son un tipo de aprendizaje supervisado que tienen como objetivo predecir las etiquetas de clase categóricas de las nuevas instancias, basándonos en observaciones pasadas. Se distinguen diferentes tipos de clasificación

según el número de clases de salida: clasificación binaria, cuando solo se pueden asignar dos clases diferentes (0 o 1). Clasificación multi-clase, donde se pueden asignar múltiples categorías a las observaciones. Clasificación multi-etiqueta (*Multi-Label Learning* o MLL), cuando las etiquetas de clase no son mutuamente excluyente entre sí. Por ejemplo podríamos querer clasificar una imagen de entrada en imagen de playa o no (clasificación binaria). Podríamos seleccionar la clase que mejor se ajusta a la imagen: nubes, agua, árboles, arena, casas (clasificación multi-clase). O bien podría ser clasificada al mismo tiempo como agua y arena, o bien como árboles, arena y nubes, etc. (clasificación multi-etiqueta).

No obstante, en muchos problemas del mundo real podemos encontrar casos para los cuales la clasificación MLL todavía no es suficiente ya que el nivel de descripción de cada etiqueta no es el mismo. Si retomamos nuestro ejemplo de clasificación de imágenes podría ser útil saber que nuestra imagen de muestra está compuesta principalmente por agua, con mucha arena y nubes, algunos árboles y pocas casas.

Tanto la clasificación multi-clase como la clasificación multi-etiqueta pretenden responder a la pregunta: "¿qué etiqueta(s) puede describir el caso? Sin embargo, ninguna de ambas técnicas puede responder directamente a la pregunta siguiente con más ambigüedad: "¿cuánto describe cada etiqueta el caso?". Donde la importancia relativa de cada etiqueta también está implicada en la descripción de la instancia. Una forma más natural de etiquetar una instancia $x$ es asignar un número real $d_x^y$ a cada una de las posibles etiquetas $y$, representando el grado en que $y$ describe $x$. Esta generalización de la clasificación multi-etiqueta, que será la base de nuestro estudio, es lo que se denomina Distribución de Etiquetas (*Label Distribution Learning*), o a partir de ahora, LDL [49] (Figura 1.1).

El concepto de LDL aparece por primera vez en la literatura en el año 2013 en [52] pero no es hasta 2016 cuando se describe formalmente este novedoso paradigma de aprendizaje que extiende y generaliza los paradigmas de aprendizaje clásicos [49]. Desde entonces, se han propuesto diversos métodos para tratar los problemas de tipo LDL pudiendo agrupar los algoritmos existentes en tres categorías principales, a saber: transformación de problemas, adaptación de algoritmos y algoritmos especializados.

En la transformación de problemas, la idea principal consiste en convertir los ejemplos de entrenamiento en ejemplos ponderados de una sola etiqueta para ser posteriormente tratados con algoritmos de clasificación convencionales. Otra alternativa es adaptar algoritmos de aprendizaje clásicos para manejar los problemas de LDL modificando directamente algunas restricciones. Por otro lado, los algoritmos especializados pueden tratar directamente con datos de tipo LDL. Estos métodos aprenden maximizando/minimizando directamente la similitud/distancia entre las distribuciones de etiquetas previstas y reales.

En los últimos años, se han presentado varios estudios focalizados en el desarrollo de nuevos algoritmos de aprendizaje o en la adaptación de métodos ya existentes, diseñados para tratar problemas de tipo LDL. Algunos ejemplos de los más relevantes son los siguientes: algoritmos basados en instancias (AA-$k$NN [49]), algoritmos


Figura 1.1: Tipos de clasificación.

de optimización basados en modelos de máxima entropía (SA-IIs [49], SA-BFGS [49], LDL-SCL [120]), árboles de decisión (LDL Forests [99]), algoritmos de *deep learning* (Deep LDL [44]), estrategias de *ensembles* (*Logistic boosting regression for* LDL [109], *Structured random forest for* LDL [23]).

Una vez revisados algunos conceptos y procesos básicos de la minería de datos, el siguiente paso es cuestionar la calidad de los datos a utilizar. Los datos de entrada deben ser proporcionados en la cantidad, estructura y formato que se adapte perfectamente a cada tarea de la minería de datos. Desafortunadamente, las bases de datos del mundo real están altamente influenciadas por factores negativos como la presencia de ruido, valores perdidos, datos inconsistentes y superfluos y tamaños enormes en ambas dimensiones, ejemplos y características. Por lo tanto, los datos de baja calidad conducirán a un rendimiento de baja calidad del posterior algoritmo de aprendizaje [48]. Por estos motivos y a pesar de que la minería de datos se considera el núcleo del proceso KDD, el pre-procesamiento de datos es una etapa fundamental que permite a los algoritmos de aprendizaje posteriores ser más eficaces a la hora de generar los modelos de aprendizaje.

El pre-procesamiento de datos abarca numerosas tareas entre las cuales podemos destacar las siguientes:

- Integración datos [89]: abarca en parte la etapa de extracción de datos del proceso KDD, ya que consiste en fusionar los datos proveniente de diferentes fuentes dando como resultado un único conjunto de datos consistente y

unificado.

- Transformación de datos: este proceso incluye varias subtareas como la normalización o discretización de valores, la homogeneización de los datos, la generalización de conceptos para atributos categóricos, etc.
- Limpieza de datos: es el proceso de identificar las partes incorrectas, incompletas, inexactas, irrelevantes o faltantes de los datos y luego modificarlas, sustituirlas o suprimirlas según sea necesario [48].
- Reducción de datos: es el conjunto de técnicas que, de una forma u otra, obtienen una representación reducida de los datos originales pero manteniendo la estructura esencial y la integridad de los datos originales [11, 74].

La correcta combinación y aplicación de estas tareas sobre el conjunto de datos original dará como resultado datos de calidad que serán la base para la etapa de minería de datos.

## 1.1. Objetivos

LDL es una generalización de la tarea de clasificación [104] y por lo tanto es vulnerable a los mismos problemas que los algoritmos de clasificación convencionales: conjuntos de datos no balanceados [36], cuando hay una desproporción en el número de ejemplos de las diferentes clases; datos con ruido [88], debido a imperfecciones en la adquisición, transmisión o almacenamiento de datos; superposición [86], cuando las características de entrada no son suficientes para diferenciar correctamente entre instancias de diferentes clases; o irregularidades [27], situaciones en las que la distribución de los puntos de datos, el muestreo del espacio para generar el conjunto de entrenamiento y las características que describen cada punto de datos se desvían de lo que podría haber sido ideal, siendo sesgadas, incompletas y/o engañosas.

El objetivo general de este trabajo de tesis es el perfeccionamiento de los algoritmos de clasificación actuales de tipo LDL utilizando para ello diferentes estrategias y técnicas de pre-procesamiento de datos. Focalizaremos nuestro estudio en tres técnicas de pre-procesamiento que han demostrado ser muy eficaces aplicadas sobre diferentes problemas de clasificación. Cada una de estas técnicas será estudiada, extendida para cumplir con las restricciones que supone un problema de tipo LDL, implementada y evaluada sobre los conjuntos de datos disponibles.

Más concretamente, en esta tesis se pretenden llevar a cabo los siguientes objetivos:

- Estudio y revisión bibliográfica del estado del arte en algoritmos de clasificación de tipo LDL y problemas que se han abordado con dichos algoritmos. Asimismo, estudio de los diferentes métodos y estrategias en el ámbito de pre-procesamiento de datos.

- Mejorar los conjuntos de datos LDL existentes: debido a la alta dimensionalidad de los vectores de características y de etiquetas con respecto al número de muestras que contienen, es muy probable que exista una falta de información en los mismos, provocando un mal rendimiento de los algoritmos de aprendizaje. Bajo esta hipótesis, nuestro objetivo es mejorar los conjuntos de datos añadiendo una etapa de pre-procesamiento, en la cual, utilizaremos técnicas de generación de instancias. Este enfoque de *oversampling* que ha demostrado ser útil usado en problemas de clasificación clásica podría extenderse para ser aplicado al paradigma LDL.

- Mejorar el rendimiento de los métodos LDL: debido al elevado número de características de entrada de los conjuntos de datos podríamos encontrarnos con problemas de rendimiento, tanto en la fase de entrenamiento como en la de predicción, de los algoritmos de aprendizaje. Para tratar de paliar estos efectos negativos podemos utilizar técnicas de reducción de datos como la selección de características o la selección de instancias. Además de la mejora de rendimiento, esta etapa de pre-procesamiento también podría ayudar a tratar posibles problemas de ruido en los datos.

- Reducir la complejidad del problema: los conjuntos de datos LDL tienen un número alto de etiquetas de salida y, por norma general, cuantas más etiquetas hay en un problema más complejo es de abordar. Para hacer frente a los problemas que sufren los algoritmos de clasificación convencionales cuando son aplicados sobre problemas complejos, podemos aplicar una transformación de datos previa que divida el problema original en subconjuntos más pequeños y por consiguiente más simples de tratar. Un método que ha resultado ser eficaz es la aplicación de estrategias de descomposición [39]. Dichas técnicas descomponen el problema de forma que pueda ser tratado por clasificadores binarios. Estas estrategias han sido ampliamente utilizadas para abordar problemas multi-clase y, en nuestro caso, podrían extenderse para lidiar con problemas de tipo LDL.

## 1.2. Metodología

Esta tesis se ha construido siguiendo el esquema del método científico tradicional. Además, requiere la combinación de metodologías prácticas y teóricas durante su desarrollo. Por lo tanto se propone el siguiente método de trabajo y experimentación:

- Observación: En este fase obtendremos los datos necesarios sobres los cuales trabajaremos en fases posteriores. Llevaremos a cabo un estudio pormenorizado del problema de clasificación LDL y la aplicación de técnicas de aprendizaje supervisado para resolución del mismo. Asimismo, realizaremos un estudio sobre la posibilidad y necesidad de la aplicación de técnicas de pre-procesamiento sobre los conjuntos de datos iniciales para la resolución del problema.

- Formulación de hipótesis: Diseño de nuevos algoritmos de pre-procesamiento de datos adaptados a las restricciones LDL. Los métodos desarrollados deben

cumplir con los objetivos mencionados anteriormente para poder llevar a cabo el perfeccionamiento de los algoritmos de clasificación actuales de tipo LDL utilizando para ello diferentes estrategias y técnicas de pre-procesamiento de datos.

- Recogida de observaciones: Obtención de resultados por los modelos propuestos, aplicados sobres los conjuntos de datos LDL disponibles y utilizando diferentes medidas de rendimiento. A fin de analizar su eficacia en cuanto a precisión así como manteniendo tiempos de ejecución razonables.

- Contraste de hipótesis: Comparación de los resultados obtenidos al utilizar técnicas de pre-procesamiento de datos respecto a otros modelos "estado del arte" de la literatura. Con el fin de analizar la calidad de las propuestas en términos de eficacia.

- Demostración o refutación de hipótesis: Aceptación o rechazo y modificación, según proceda, de las técnicas desarrolladas como consecuencia de las pruebas realizadas. Si fuese necesario, deberán repetirse los pasos anteriores para formular nuevas hipótesis que puedan ser probadas.

- Tesis o teoría científica: Extracción y aceptación de las conclusiones obtenidas durante el proceso de investigación. Redacción de las conclusiones, recopilando todo el proceso y los resultados publicados en la memoria de la tesis.

## 1.3. Resumen

Para alcanzar los objetivos que acabamos de plantear, y según lo establecido en el artículo 18.4 de las Normas Reguladoras de las Enseñanzas Oficiales de Doctorado y del Título de Doctor por la Universidad de Granada, esta tesis doctoral será presentada como un compendio de artículos publicados por el doctorando en medios científicos relevantes en su ámbito de conocimiento. Dichas publicaciones constituyen el núcleo de la tesis, y corresponden a tres artículos científicos publicados en revistas internacionales indexadas por la base de datos JCR (*Journal Citation Reports*) del *Science Citation Index*.

Por tanto, la memoria se estructura de la siguiente manera: el Capítulo 2 presenta los conceptos teóricos y antecedentes del proceso de extracción de conocimiento y minería de datos, así como los estudios más relevantes realizados sobre la distribución de etiquetas, introduce conceptos básicos sobre la etapa de pre-procesamiento de datos, las estrategias de descomposición y presenta las métricas y los conjuntos de datos utilizados en el apartado experimental. Al final de la misma sección presentaremos un resumen de las estrategias de pre-procesamiento prometedoras para LDL. Seguidamente, en el Capítulo 3 exponemos las publicaciones realizadas. Cada una de ella se corresponde con uno de los objetivos previamente planteados:

- La primera propuesta es un método de generación sintética de muestras adaptado a las restricciones que supone un modelo LDL. Bajo de la hipótesis de una falta de información en los conjuntos de datos de LDL existentes, necesitamos mejorar los datos originales para aumentar la eficacia de los algoritmos

de aprendizaje. Una de las propuestas más conocidas y utilizadas para tratar este problema es la técnica de muestreo de datos [6] en la que las instancias de entrenamiento se modifican de manera que se produzca una distribución de clases más eficiente que permita a los clasificadores mejorar su rendimiento. Este enfoque aún no ha sido estudiado para el paradigma LDL y puesto que podría resultar prometedor, hemos desarrollado un método de *oversampling* [87] que crea un superconjunto de datos partiendo del conjunto de datos original, creando nuevas instancias a partir de las existentes. La técnica ideada se basa en uno de los enfoques más utilizados en este área llamado: *Synthetic Minority Oversampling Technique* (SMOTE) [21].

La publicación asociada a esta propuesta queda recogida en el Capítulo 4.

- La segunda propuesta, enfocada en mejorar el rendimiento de los métodos LDL, se centra en el algoritmo AA-$k$NN [49], adaptación a LDL del conocido $k$-NN y que ha demostrado ser un algoritmo muy competitivo en estudios experimentales previos, logrando resultados aceptables y permitiendo un modelo explicable [7]. Sin embargo, como cualquier otro algoritmo basado en instancias, adolece de varios inconvenientes: necesita grandes requisitos de memoria para almacenar el conjunto de entrenamiento, no es eficiente en la predicción debido a los múltiples cálculos de similitudes entre las muestras de test y entrenamiento y presenta una baja tolerancia al ruido porque utiliza todos los datos como relevantes. La propuesta aplica dos técnicas de reducción de datos como son la selección de prototipos [47], y la selección de características [101]. El resultado es un método novedoso para abordar simultáneamente la selección de prototipos y la selección de características específicas a cada etiqueta de salida, específicamente diseñado para el algoritmo AA-$k$NN.

  La publicación asociada a esta propuesta queda recogida en el Capítulo 5.

- La última propuesta consiste en una transformación de datos orientada a reducir la complejidad del problema. La técnica ideada es una estrategia de descomposición adaptada para tratar problemas LDL y que se inspira en una de las estrategias más conocidas en este área: el esquema *"One-Vs-One"* (OVO) [64], donde el problema original se divide en problemas binarios que distinguen los diferentes pares de clases, seguidamente, cada división se entrena con un clasificador base. Este método suele requerir un paso adicional para fusionar las salidas de los clasificadores simples con el fin de producir el resultado final. Para el caso de LDL, el método resultante combina una estrategia de descomposición capaz de manejar la distribución de etiquetas de salida en lugar de valores discretos de clase así como un mecanismo capaz de proporcionar una salida de acuerdo con las restricciones de LDL. Por otro lado, mientras que OVO utiliza un clasificador binario como clasificador base, en nuestra propuesta tenemos que incluir un clasificador LDL que pueda tratar con los valores reales de las etiquetas de salida.

  La publicación asociada a esta propuesta queda recogida en el Capítulo 6.

Analizaremos los resultados de todas estas propuestas en el Capítulo 7. Para

terminar, el Capítulo 8 recopila las conclusiones alcanzadas e introduce las futuras líneas de investigación.

# Capítulo 2

# Conceptos Teóricos y Antecedentes

En esta sección empezaremos describiendo el proceso de descubrimiento de información (Sección 2.1) así como la etapa principal del mismo: la minería de datos (Sección 2.2). Posteriormente se presentan los fundamentos y los estudios más relevantes realizados sobre Distribución de Etiquetas, así como las métricas y conjuntos de datos utilizados en la parte experimental (Sección 2.3). Además, se introducen algunos conceptos sobre la etapa de pre-procesamiento de datos y estrategias de descomposición así como alternativas prometedoras de pre-procesamiento para LDL (Sección 2.4), proporcionando los antecedentes necesarios para presentar adecuadamente el estudio realizado en esta tesis.

## 2.1. Proceso de descubrimiento de información

El proceso de descubrimiento de conocimiento en bases de datos (*Knowledge Discovery in Databases* o KDD) [84] puede definirse como el conjunto de etapas que posibilitan la identificación de patrones válidos, novedosos, potencialmente útiles y comprensibles en los datos [38]. Es un análisis exploratorio automático de datos que se divide en varias etapas:

- Especificación del problema: designación y organización del dominio de aplicación, los conocimientos previos pertinentes obtenidos por los expertos y los objetivos finales perseguidos por el usuario final.
- Extracción de datos: selecciona los datos desde una o varias fuentes de información relevantes para el problema, habitualmente con ayuda de conocimiento experto. Los datos extraídos se integran para ser procesados en siguientes etapas.
- Pre-procesamiento de datos: esta etapa incluye operaciones de limpieza de datos (como la eliminación del ruido y de los datos incoherentes), la integración de datos (en la que se pueden combinar varias fuentes de datos en una sola), la transformación de datos (en la que los datos se transforman y consolidan en formas adecuadas para tareas específicas de gestión de datos u operaciones

de agregación) y la reducción de datos, incluida la selección y extracción tanto de características como de instancias. El objetivo final del pre-procesamiento de datos es la obtención de datos de calidad, también conocidos como *Smart Data* en el área del *Big Data* [77], para su uso en siguientes etapas.

- Minería de datos: es el proceso esencial en el que se utilizan los métodos de aprendizaje para extraer patrones de datos. Este paso incluye la elección de la tarea de minería de datos más adecuada, la elección del algoritmo de aprendizaje según la tarea que queremos llevar a cabo y finalmente, el empleo y adaptación del algoritmo seleccionado al problema, mediante el ajuste de los parámetros y los procedimientos de validación.

- Evaluación: estimación e interpretación de las patrones extraídos en base a medidas de interés.

- Explotación de resultados: la última etapa puede consistir en utilizar directamente los conocimientos, incorporar los conocimientos a otro sistema para procesos posteriores o simplemente informar sobre los conocimientos descubiertos.

## 2.2. Minería de datos

Cabe mencionar que todas las etapas del proceso de KDD están interconectadas (Figura 2.1). Cada etapa condiciona las etapas restantes y todas son imprescindibles para conseguir los objetivos del proceso. Sin embargo, no todos los pasos requieren la misma dedicación ni tiempo. El núcleo central que une y condiciona el resto de etapas es la minería datos. Tradicionalmente, se divide en tres ramas según el tipo de conocimiento a extraer:

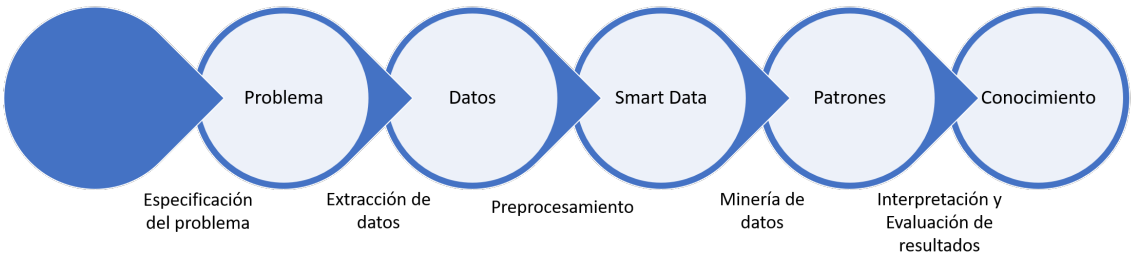



Figura 2.1: Etapas del KDD.

- Aprendizaje predictivo o supervisado: aquí el objetivo es aprender un mapeo de las entradas $x$ a las salidas $y$. En la configuración más simple, cada entrada de entrenamiento $x_i$ es un vector D-dimensional de números, representando lo que denominan características o atributos. Sin embargo, $x_i$ podría ser un objeto estructurado complejo, como una imagen, un mensaje de correo electrónico, una serie de tiempo, un gráfico, etc. De manera similar, la forma de la variable de salida o de respuesta puede ser en principio cualquier cosa, pero la mayoría

de los métodos asumen que $y_i$ es una variable categórica o nominal de algún conjunto finito o que $y_i$ es un escalar de valor real. Cuando $y_i$ es categórico, el problema se conoce como **clasificación** o reconocimiento de patrones [33], y cuando $y_i$ es de valor real, el problema se conoce como **regresión** [32].

- Aprendizaje descriptivo o no supervisado: aquí sólo se nos dan entradas y el objetivo es encontrar "patrones interesantes" en los datos [1]. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori, se tratan los objetos de entrada como un conjunto de variables aleatorias, construyendo un modelo de densidad para el conjunto de datos. La agrupación de un conjunto de datos (***clustering*** [81]) en subconjuntos de objetos similares es un ejemplo típico de este tipo de aprendizaje. También la **asociación** [65] que permite detectar relaciones relevantes entre las diferentes variables de los datos.

- Aprendizaje semi-supervisado [122]: es un híbrido entre la tarea de clasificación predictiva y la análisis descriptivo. Es un paradigma de aprendizaje relacionado con el diseño de modelos en presencia de datos etiquetados y no etiquetados. Esencialmente, los desarrollos en este campo utilizan las muestras no etiquetadas para modificar la hipótesis obtenida utilizando unicamente las muestras etiquetadas.

Entremos un poco más en detalle en el tipo de aprendizaje que será objeto de estudio en esta tesis: la clasificación. Recordemos que el objetivo es modelar un mapeo de las entradas $x$ a las salidas $y$, donde $y \in \{1, \ldots, C\}$, siendo $C$ el número de clases. En este punto podemos a su vez diferenciar diferentes tipos de clasificación basándonos en el número de clases de salida:

- Clasificación binaria: cuando el número de clases de salida $C = 2$, estamos en el caso de clasificación clásica o clasificación binaria. En cuyo caso asumimos un valor de salida $y \in \{0, 1\}$ clasificando las muestras en positivas o negativas.

- Clasificación multi-clase: la clasificación no tiene porque ser binaria sino que podríamos clasificar una muestra en un conjunto finito de etiquetas. Estamos en el caso donde el número de clases de salida $C > 2$.

- Clasificación multi-etiqueta: las etiquetas de clase no tienen porque ser mutuamente excluyentes. Podemos decir que la clasificación multi-etiqueta (*Multi-Label Learning* o MLL) es una generalización de la clasificación tradicional donde varias etiquetas de salida puedes ser asignadas a una misma instancia [1].

## 2.3. Distribución de Etiquetas

En los problemas de clasificación clásicos buscamos responder a la pregunta: "¿Qué etiqueta describe mejor a esta muestra?". El paradigma original, o *Single-Label Learning* (SLL), responde a esta pregunta dando como salida un único valor considerado como el que mejor representa a la instancia que queremos clasificar. Sin

embargo, hay un número creciente de problemas en los que una muestra puede tener varias etiquetas asociadas simultáneamente, por ejemplo, problemas de clasificación de imágenes [12], genética [8], etc. El aprendizaje multi-etiqueta [53, 66, 82, 103] es una generalización de la clasificación tradicional en la que se pueden asignar múltiples etiquetas a cada instancia. MLL permite modelar mejor la ambigüedad pudiendo asociar varias etiquetas de salida en vez de solo una pero, en muchos problemas del mundo real, podemos encontrar casos en los que el MLL todavía no es suficiente ya que el nivel de descripción de cada etiqueta no es el mismo.

### 2.3.1. Definición de Distribución de Etiquetas

El concepto de Distribución de Etiquetas (*Label Distribution Learning* o LDL) aparece por primera vez en la literatura en el año 2013 [52] pero no es hasta 2016 [49] cuando se describe formalmente este novedoso paradigma de aprendizaje que extiende y generaliza los paradigmas de aprendizaje clásicos SLL y MLL. El objetivo de LDL es responder a la pregunta: "¿Cuánto o en qué grado describe cada etiqueta a la muestra?".

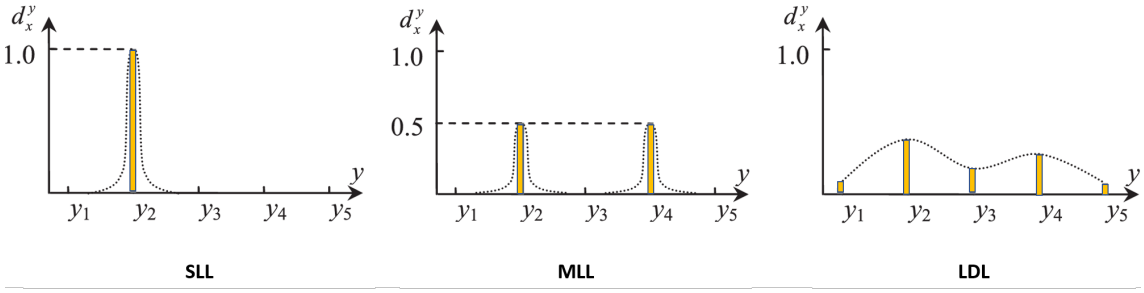



Figura 2.2: Ejemplos de clasificación usando diferentes paradigmas [49].

Podemos formalizar un problema LDL como un conjunto de $m$ muestras de entrenamiento $S = \{(x_1, D_1), ..., (x_m, D_m)\}$, donde $x_i = \{x_{i1}, x_{i2}, ..., x_{iq}\}$ es un vector $q$-dimensional. Para cada muestra $x_i$, la distribución de etiquetas se representa como $D_i = \{d_{x_i}^{y_1}, d_{x_i}^{y_2}, ..., d_{x_i}^{y_c}\}$ donde $y_i \in Y | i \in \{i, \cdots, c\}$, de tal manera que $Y = \{y_1, \cdots, y_c\}$ representa el conjunto completo de etiquetas. La constante $c$ es el número de posibles etiquetas y $d_{x_i}^{y_j}$ es el grado de descripción de la $j$-ésima etiqueta de salida para la $i$-ésima instancia. De acuerdo a esta definición, un problema LDL debe cumplir las siguientes restricciones:

- Cada grado de descripción $d_x^y$ debe tener un valor real entre $[0, 1]$: $d_x^y \in [0, 1]$.
- La suma de la distribución de etiquetas debe ser la unidad: $\sum_{y=1}^{c} d_x^y = 1$.

Desde la primera aparición del concepto LDL, se han realizado numerosos estudios aplicando la metodología LDL para resolver problemas de la vida real, por ejemplo: estimación de la edad a partir de imágenes faciales [52], recuento de multitudes en la video-vigilancia pública [118], clasificación de la belleza a partir de imágenes [92], reconocimiento de la personalidad de un individuo usando los medios de comunicación social [111], clasificación de emociones a partir de imágenes [112],

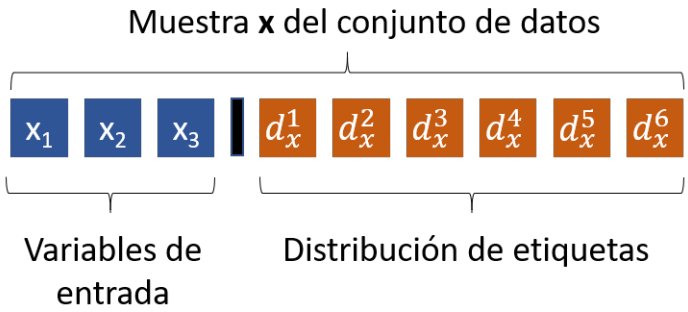



Figura 2.3: Representación de una muestra LDL.

predicción previa al lanzamiento de la opinión del público sobre las películas [50], detección de la orientación de la cara [110], etc.

A partir de estos y otros estudios, hemos extraído los conjuntos de datos que nos servirán de base para la parte experimental de esta tesis. Describimos cada uno de ellos con más detalle en la Sección 2.3.4.

### 2.3.2. Estado del arte LDL

La resolución de un problema de LDL puede ser abordada desde diferentes perspectivas. Según el enfoque elegido, el algoritmo que se desarrolle variará considerablemente, ya sea un algoritmo completamente nuevo desarrollado específicamente para tratar las limitaciones de LDL, o una adaptación de los algoritmos de clasificación existentes, reformulados para trabajar con estas limitaciones. El estudio original sobre LDL publicado en [49] proponía tres categorías para clasificar los algoritmos. La primera es la Transformación del Problema (*Problem Transformation* o PT), una forma sencilla de transformar un problema LDL a un paradigma SLL clásico. El segundo es la Adaptación de Algoritmos (*Algorithm Adaptation* o AA), en el que los algoritmos se adaptan a los métodos de aprendizaje existentes para ocuparse directamente de la distribución de las etiquetas. Por último, los algoritmos especializados (*Specialized Algorithms* o SAs), a diferencia de la estrategia indirecta de transformación del problema y adaptación del algoritmo, coinciden directamente con el problema de LDL. A continuación veremos algunos ejemplos de algoritmos de cada tipo y veremos cuales de ellos pueden considerarse como "estado del arte".

#### Transformación del problema

Una forma sencilla de transformar un problema de tipo LDL en un problema de tipo SLL es cambiando los ejemplos de entrenamiento en ejemplos ponderados de una sola etiqueta. Cada ejemplo de entrenamiento $(x_i, D_i)$ se transforma en $c$ ejemplos de una sola etiqueta $(x_i, y_i)$ con un peso $d_{x_i}^{y_j}$, siendo $i = 1, ..., n$ y $j = 1, ..., c$. El conjunto de entrenamiento es entonces modificado, incrementando su tamaño de acuerdo al peso de cada ejemplo. El nuevo conjunto de entrenamiento resultante se convierte así en un conjunto de entrenamiento estándar de una sola etiqueta que

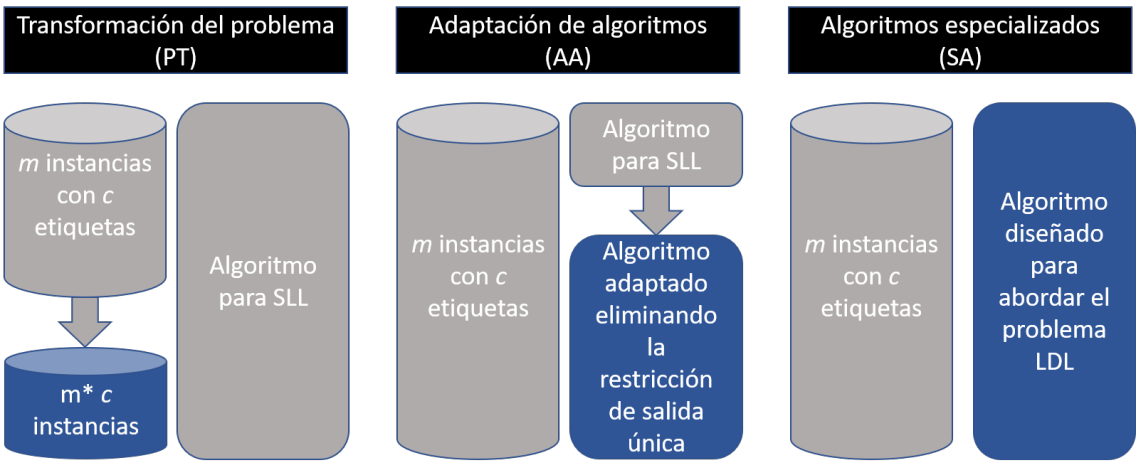



Figura 2.4: Metodologías para resolver un problema de tipo LDL.

incluye $c \times n$ ejemplos, sobre el cual podemos aplicar cualquier algoritmo SLL. Para predecir la distribución de etiquetas de una instancia $x$, el modelo debe ser capaz de predecir la confianza/probabilidad de cada etiqueta $y_j$, que puede considerarse como el grado de descripción de la etiqueta correspondiente, $d_{x_i}^{y_j} = P(y_j|x)$.

Dos de los algoritmos más representativos de esta categoría son PT-Bayes y PT-SVM [49]. El clasificador Bayesiano asume una distribución gaussiana para cada clase, y la probabilidad posterior calculada por la regla de Bayes se considera como el grado de descripción de la etiqueta correspondiente. En cuanto al basado en SVM [98], las estimaciones de probabilidad se obtienen mediante un método multiclase de emparejamiento por pares [108], en el que la probabilidad de cada SVM binario se calcula mediante una implementación mejorada de las probabilidades posteriores de Platt [72], y las estimaciones de probabilidad de la clase se obtienen resolviendo un sistema lineal cuya solución está garantizada por la teoría de las cadenas finitas de Markov.

### Adaptación de Algoritmos

Ciertos algoritmos de aprendizaje existentes pueden extenderse naturalmente para tratar las distribuciones de etiquetas. El método más representativo de esta categoría es el denominado AA-$k$NN [49], adaptación del conocido clasificador $k$-NN. El proceso es el siguiente: dada una nueva instancia $x$, sus $k$ vecinos más cercanos se buscan en el conjunto de entrenamiento. Luego, la media de las distribuciones de etiquetas de los $k$ vecinos más cercanos se calcula como la distribución de etiquetas de $x$ según la fórmula:

$$p(y_j|x) = \frac{1}{k} \sum_{i \in N_k(x)} d_{x_i}^{y_j}, (j = 1, 2, ..., c),$$

donde $N_k(x)$ es el conjunto de índices de los $k$ vecinos más cercanos a $x$ en el conjunto de entrenamiento.

Otra propuesta en esta categoría es un algoritmo basado en redes neuronales con propagación hacia atrás (*backpropagation* o BP). AA-BP [49] es una red neuronal

compuesta de tres capas de $q$ unidades de entrada (número de características de $x$) que reciben las instancias $x$, y $c$ (número de etiquetas) unidades de salida, cada una de las cuales determinan el grado de descripción de la etiqueta $y_j$. Para SLL o MLL, la salida deseada suele ser un vector con '1's en las posiciones correspondientes a las etiquetas positivas de la instancia de entrada, y '0's en caso contrario. En el caso LDL, la salida se convierte en la verdadera distribución de etiquetas de la muestra de entrada. Por lo tanto, el objetivo del algoritmo AA-BP es minimizar la raíz cuadrada de la sumatoria del error de la salida de la red neuronal en comparación con la distribución de etiquetas real. Para asegurar que cada salida cumpla con las restricciones $d_x^y \in [0,1]$ y $\sum_{y=1}^{c} d_x^y = 1$, se aplica la función softmax en cada unidad de salida.

#### Algoritmos especializados

Los algoritmos especializados, a diferencia de la estrategia indirecta de transformación del problema y adaptación del algoritmo, tratan directamente con el problema LDL. En el estudio original de LDL [49] se propusieron los métodos SA-IIS y SA-BFGS que son dos algoritmos especializados que aprenden optimizando una función de energía basada en un modelo de máxima entropía.

Estudios posteriores han logrado mejorar los resultados obtenidos por estos algoritmos originales utilizando diferentes estrategias. Los métodos LDLogitBoost y AOSO-LDLogitBoost propuestos en [109], son una combinación del método de *boosting* y la regresión logística aplicada al modelo LDL. *Deep Label Distribution Learning* (DLDL) [44] y LDL basado en conjuntos de redes neuronales [115] son dos buenos ejemplos de éxito en la aplicación de redes neuronales en LDL. Inspirado en los árboles de decisión diferenciables [70], se estudió una estrategia basada en técnicas de *Ensembles* de tipo *Forest* especificamente diseñada para LDL y propuesta en [99]. Dicho estudio sirvió de base para la propuesta *Structured Random Forest* (StructRF) [23]. BC-LDL [105] y DBC-LDL [106] utilizan técnicas de codificación binaria para hacer frente a problemas LDL de gran tamaño. La clasificación con LDL (LDL4C) [104] es otra propuesta interesante cuando el modelo de distribución de etiquetas aprendidas se trata generalmente como un modelo de clasificación.

### 2.3.3. Métricas de evaluación

La salida de un algoritmo LDL es una distribución de etiquetas, que es diferente tanto de la salida de una sola etiqueta de SLL como de la salida de un conjunto de etiquetas de MLL. Por consiguiente, las medidas de evaluación de los algoritmos LDL deben ser diferentes de las utilizadas para los algoritmos SLL y MLL. Para medir la distancia/similitud entre las distribuciones de probabilidad se pueden aplicar varias medidas de distancia/similitud entre las distribuciones de etiquetas previstas y reales. Proponemos utilizar un conjunto de seis medidas para comparar los diferentes algoritmos de LDL:

- Distancia de Chebyshev: es una métrica definida en un espacio vectorial donde la distancia entre dos puntos (representados por sus vectores) es la mayor de sus diferencias a lo largo de cualquiera de sus dimensiones coordenadas.

- Distancia de Clark: se trata de una extensión del coeficiente de divergencia [24] para su uso con múltiples caracteres.

- Distancia de Canberra: es una medida numérica de la distancia entre pares de puntos en un espacio vectorial. Es una versión ponderada de la distancia de Manhattan (distancia entre dos puntos medidos a lo largo de ejes en ángulo recto). La distancia de Canberra es una función métrica utilizada a menudo para datos dispersos alrededor de un origen.

- Divergencia de Kullback-Leibler: también conocida como divergencia de la información, ganancia de la información o entropía relativa, es una medida no simétrica de la similitud o diferencia entre dos funciones de distribución de probabilidad $p(x)$ y $q(x)$. Específicamente, la divergencia Kullback-Leibler de $q(x)$ respecto a $p(x)$ es una medida de la información perdida cuando se usa $q(x)$ como aproximación de $p(x)$.

- Similitud Coseno: es una métrica que se utiliza para medir cuán similares son dos vectores distintos de cero, independientemente de su tamaño. Mide el coseno del ángulo entre dos vectores proyectados en un espacio multidimensional. Cuanto menor sea el ángulo, mayor será la similitud del coseno.

- Similitud Intersección: tiene su mayor valor, 1, cuando todos los términos de la primera distribución de probabilidad son idénticos a los términos correspondientes de la segunda distribución de probabilidad. De lo contrario, la similitud es menor que 1. En el caso extremo, cuando ambas distribuciones son muy diferentes, entonces la similitud será cercana a 0.

Cada una de estas medidas proviene de una familia sintáctica diferente: Minkowski, $X^2$, $L_1$, entropía de Shannon, producto escalar e intersección, respectivamente [17]. Teniendo en cuenta que provienen de diferentes familias y que son significativamente diferentes tanto en la sintaxis como en la semántica, tienen una buena oportunidad de reflejar diferentes aspectos de un algoritmo LDL. Suponiendo que la verdadera distribución de la etiqueta es $D = \{d_1, d_2, ..., d_c\}$, y la predicción de la distribución es $\hat{D} = \{\hat{d}_1, \hat{d}_2, ..., \hat{d}_c\}$, entonces la formulación de las seis medidas se resume en el Cuadro 2.1.

### 2.3.4. Conjuntos de datos

Esta sección presenta una compilación de los conjuntos de datos más utilizados en los artículos LDL referenciados. Son quince conjuntos de datos del mundo real cuyas características quedan resumidas en el Cuadro 2.2.

Los diez primeros conjuntos de datos (*Yeast*) se recogieron a partir de experimentos biológicos llevados a cabo con una levadura incipiente llamada *Saccharomyces cerevisiae* [34]. Cada conjunto de datos incluye 2.465 genes de levadura, y un vector de perfil filogenético asociado con una longitud de 24 se utiliza para representar cada gen. El nivel de expresión génica (después de la normalización) en cada instante temporal proporciona una medida natural del grado de descripción de la etiqueta

Cuadro 2.1: Métricas de evaluación para clasificadores LDL. ↓ *significa que el valor más bajo es el mejor y* ↑ *significa lo contrario.*

| Medida | Fórmula |
|---|---|
| Distancia de Chebyshev↓ | $Dis(D,\hat{D}) = max_j\|d_j - \hat{d}_j\|$ |
| Distancia de Clark↓ | $Dis(D,\hat{D}) = \sqrt{\sum_{j=1}^{c} \frac{(d_j-\hat{d}_j)^2}{(d_j+\hat{d}_j)^2}}$ |
| Distancia de Canberra↓ | $Dis(D,\hat{D}) = \sum_{j=1}^{c} \frac{\|d_j-\hat{d}_j\|}{d_j+\hat{d}_j}$ |
| Divergencia de Kullback-Leibler↓ | $Dis(D,\hat{D}) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$ |
| Similitud Coseno↑ | $Sim(D,\hat{D}) = \frac{\sum_{j=1}^{c} d_j\hat{d}_j}{\sqrt{\sum_{j=1}^{c} d_j^2}\sqrt{\sum_{j=1}^{c} \hat{d}_j^2}}$ |
| Similitud Intersección↑ | $Sim(D,\hat{D}) = \sum_{j=1}^{c} min(d_j,\hat{d}_j)$ |

correspondiente. Los grados de descripción (niveles de expresión génica normalizados) de todas las etiquetas (instantes temporales) constituyen una distribución de etiquetas para un gen de levadura particular. Los datos se obtuvieron de los instantes de tiempo durante los siguientes procesos: el ciclo de división celular después de la sincronización por detención del factor alfa (*Yeast_alpha*, 18 instantes de tiempo); elutriación centrífuga (*Yeast_elu*, 14 instantes de tiempo), y con un mutante cdc15 sensible a la temperatura (*Yeast_cdc*, 15 puntos temporales); esporulación (*Yeast_spo/spo5/spoem*, 7 instantes de tiempo más cuatro muestras adicionales); choque por alta temperatura (*Yeast_heat*, 6 instantes de tiempo); agentes reductores (*Yeast_dtt*, 4 instantes de tiempo) y baja temperatura (*Yeast_cold*, 4 instantes de tiempo); y desplazamiento diaúxico (*Yeast_diau*, 7 instantes de tiempo).

Los conjuntos de datos *SJAFFE* y *SBU_3DFE* son extensiones de dos bases de datos de imágenes de expresiones faciales ampliamente utilizadas, *JAFFE* [78] y *BU_3DFE* [113]. La base de datos *JAFFE* contiene 213 imágenes de expresiones en escala de grises representadas por 10 modelos femeninas japonesas. De cada imagen se extrae un vector de rasgos de 243 dimensiones utilizando el método *Local Binary Patterns (LBP)* [4]. Cada imagen es puntuada por 60 personas en seis etiquetas de emociones básicas (felicidad, tristeza, sorpresa, miedo, ira y asco) con una escala de 5 niveles. La puntuación media de cada emoción se utiliza para representar la intensidad de la emoción. En lugar de considerar sólo la emoción con la mayor intensidad, como se hace en *JAFFE*, el conjunto de datos *SJAFFE* (Scored JAFFE) mantiene todas las puntuaciones y las normaliza en una distribución de etiquetas sobre las seis etiquetas de emoción. De manera similar, para la base de datos más grande *BU_3DFE* que contiene 2.500 imágenes de expresiones faciales, cada imagen es puntuada por 23 personas de la misma manera que en *JAFFE*, resultando en la versión de distribución de etiquetas del conjunto de datos *SBU_3DFE* (Scored BU_3DFE).

El conjunto de datos *Movie* incluye las valoraciones de los usuarios sobre una lista de películas. El conjunto de datos incluye 7.755 películas y 54.242.292 valoraciones en una escala de 1 a 5 estrellas de 478.656 usuarios diferentes. Dichas valoraciones provienen de la plataforma de streaming ®Netflix. La distribución de las etiquetas

| No. | Conjuntos de Datos | Muestras($m$) | Características($q$) | Etiquetas($c$) |
|---|---|---|---|---|
| 1 | Yeast_alpha | 2465 | 24 | 18 |
| 2 | Yeast_cdc | 2465 | 24 | 15 |
| 3 | Yeast_cold | 2465 | 24 | 4 |
| 4 | Yeast_diau | 2465 | 24 | 7 |
| 5 | Yeast_dtt | 2465 | 24 | 4 |
| 6 | Yeast_elu | 2465 | 24 | 14 |
| 7 | Yeast_heat | 2465 | 24 | 6 |
| 8 | Yeast_spo | 2465 | 24 | 6 |
| 9 | Yeast_spo5 | 2465 | 24 | 3 |
| 10 | Yeast_spoem | 2465 | 24 | 2 |
| 11 | SJAFFE | 213 | 243 | 6 |
| 12 | SBU_3DFE | 2500 | 243 | 6 |
| 13 | Movie | 7755 | 1869 | 5 |
| 14 | Natural_Scene | 2000 | 294 | 9 |
| 15 | Human_Gene | 30542 | 36 | 68 |

Cuadro 2.2: Conjuntos de datos usados en los experimentos.

de clasificación se calcula para cada película como el porcentaje de cada nivel de clasificación. Las características de la película se extraen de los metadatos como el género, director, actor, país, presupuesto, etc. Después de transformar los atributos categóricos en vectores binarios, el vector de características final extraído de cada película es de 1.869 dimensiones.

El conjunto de datos *Natural_Scene* [12] recopila clasificaciones de 2.000 imágenes de escenas de la naturaleza. Hay nueve posibles etiquetas asociadas a estas imágenes: planta, cielo, nube, nieve, edificio, desierto, montaña, agua y sol. Se pide a diez clasificadores humanos que etiqueten las imágenes. Para cada imagen, primero seleccionan de las nueve etiquetas candidatas las que creen que son relevantes para la imagen, y luego clasifican las etiquetas relevantes en orden descendente de relevancia para la imagen. Cada clasificador humano toma sus decisiones de forma independiente, y las clasificaciones de las etiquetas resultantes son, como es de esperar, muy inconsistentes. Posteriormente, las clasificaciones inconsistentes de cada imagen se transforman en una distribución de etiquetas mediante un proceso de programación no lineal [51], que encuentra la distribución de etiquetas común más acorde con todas las clasificaciones personales. Por último, para cada imagen, un vector de características de 294 dimensiones se extrae por el método propuesto en [12].

Finalmente, el conjunto de datos *Human_Gene* es un conjunto de datos del mundo real a gran escala, recopilado a partir de la investigación biológica sobre la relación entre los genes humanos y las enfermedades. Hay en total 30.542 genes humanos incluidos en este conjunto de datos, cada uno de los cuales está representado por los 36 descriptores numéricos de una secuencia de genes propuestos en [114]. Las etiquetas corresponden a 68 enfermedades diferentes. El nivel de expresión génica (después de la normalización) de cada enfermedad se considera como el grado de descripción de la etiqueta correspondiente. Los grados de descripción (nivel de expresión génica

normalizado) de las 68 etiquetas (enfermedades) constituyen una distribución de etiquetas para un gen humano particular.

## 2.4. Pre-procesamiento de datos

Hoy en día, uno de los principales desafíos de los algoritmos de aprendizaje supervisado sigue siendo cómo tratar los conjuntos de datos en bruto. La recolección de datos es generalmente un proceso no supervisado, que conduce a conjuntos de datos con información redundante, datos ruidosos o características irrelevantes. Por lo tanto, el pre-procesamiento de datos es un paso importante en el proceso de extracción de datos, que puede mitigar este tipo de problemas y generar conjuntos de datos mejorados [48]. En el Apartado 2.4.1 describimos las categorías más importantes de pre-procesamiento para, a continuación en el Apartado 2.4.2, introducir las estrategias de descomposición, métodos de clasificación que pueden ser categorizados como un tipo de pre-procesamiento basado en la transformación de datos.

### 2.4.1. Definición y categorías de pre-procesamiento

En esta sección describiremos la categorización general en la que podemos dividir el conjunto de técnicas de pre-procesamiento de datos. El objetivo es ofrecer un breve resumen de las técnicas de pre-procesamiento con las que deberíamos estar familiarizados para entender el trabajo realizado en esta tesis.

#### Integración de datos

Consiste en el proceso de fusión de datos desde variadas y diferentes fuentes. Este proceso debe realizarse con cuidado para evitar redundancias e inconsistencias en el conjunto de datos resultante. Las operaciones típicas que se realizan dentro de la integración de datos son la identificación y unificación de variables y dominios, el análisis de la correlación de atributos, la duplicación de tuplas y la detección de conflictos en los valores de los datos de diferentes fuentes.

En la literatura se pueden encontrar enfoques automáticos utilizados para integrar los datos, desde técnicas que localizan y emparejan los esquemas de los datos [31], hasta procedimientos automáticos que reconcilian diferentes esquemas [30].

#### Transformación de datos

En esta etapa de pre-procesamiento, los datos se convierten o consolidan de manera que el resultado del proceso de extracción pueda aplicarse o sea más eficiente. Las subtareas dentro de la transformación de datos son la homogeneización, la creación de características, la agregación, la normalización, la discretización y la generalización de los datos.

La mayoría de estas se segregarán como tareas independientes, debido a que la transformación de datos engloba un gran número de diferentes técnicas de pre-

procesamiento de datos. Las tareas que requieren supervisión humana son las técnicas clásicas de transformación de datos, como la generación de informes, los nuevos atributos que agregan los ya existentes y la generalización de conceptos especialmente en atributos categóricos.

En el caso particular de la clasificación en problemas de tipo multi-clase podemos aplicar una transformación de datos denominada: **estrategia de descomposición**, que veremos con más detalle en el Apartado 2.4.2.

### Limpieza de datos

Por limpieza de datos se entiende el proceso de identificar las partes incorrectas, incompletas, inexactas, irrelevantes o faltantes de los datos y luego modificarlas, sustituirlas o suprimirlas según sea necesario. La limpieza de datos se considera un elemento fundamental del pre-procesamiento de datos. Es un concepto general que abarca o se superpone a otras técnicas conocidas de preparación de datos. En esta área se incluyen el tratamiento de los datos perdidos, la detección de anomalías y datos sucios (fragmentos de los datos originales que no tienen sentido) así como el tratamiento del ruido.

La presencia de ruido en los datos es un problema común que produce varias consecuencias negativas en los problemas de clasificación. El ruido es un problema inevitable, que afecta a los procesos de recopilación y preparación de datos en las aplicaciones de minería de datos, donde los errores comúnmente ocurren. El rendimiento de los modelos de aprendizaje construidos en tales circunstancias dependerá en gran medida de la calidad de los datos de entrenamiento, pero también de la solidez frente al ruido del propio modelo. Por lo tanto, los problemas que contienen ruido son problemas complejos y las soluciones acertadas son a menudo difíciles de lograr sin usar técnicas especializadas.

Podemos diferenciar diferentes tipos de ruido en los datos [16]:

- **Ruido de clase (o etiqueta)**: ocurre cuando una muestra está mal etiquetada. El ruido de clase puede atribuirse a varias causas, como la subjetividad durante el proceso de etiquetado, los errores de introducción de datos o la inadecuada información utilizada para etiquetar cada ejemplo.

- **Ruido de atributo**: se refiere a la corrupción de los valores de uno o más atributos. Ejemplos de ruido de atributos son: valores de atributos erróneos, valores de atributos desconocidos o ausentes y valores de atributos incompletos.

Los filtros de ruido son mecanismos de pre-procesamiento para detectar y eliminar las instancias ruidosas en el conjunto de entrenamiento. La separación de la detección de ruido y el aprendizaje tiene la ventaja de que las instancias ruidosas no influyen en el diseño del modelo de clasificación [43]. Los filtros de ruido están generalmente orientados a detectar y eliminar los casos de ruido de clase, sin embargo, la eliminación de instancias con ruido de atributo resulta contraproducente [123], ya que dichas instancias todavía contienen información valiosa en otros atributos que pueden ayudar a construir el clasificador.

Para tratar los problemas de ruido en los atributos podemos usar otros enfoques. Una estrategia eficaz consiste en entrenar no a un solo clasificador sino a varios (estrategia de conjuntos o *Ensembles* [85]), aprovechando sus puntos fuertes particulares y consiguiendo clasificadores resistentes al ruido hasta cierto punto, incluso cuando el ruido no es tratado o limpiado.

### Reducción de datos

La reducción de datos comprende el conjunto de técnicas que, de una forma u otra, obtienen una representación reducida de los datos originales pero manteniendo la estructura esencial y la integridad de los datos originales. La reducción de datos es un paso opcional, ahora bien, los algoritmos utilizados en minería de datos tienen tiempos de ejecución que depende de varios parámetros y algunos de estos parámetros suelen ser proporcionales al tamaño de la base de datos de entrada. Si dicho tamaño es excesivo el funcionamiento del algoritmo puede ser prohibitivo y, por lo tanto, la tarea de reducción de datos puede llegar a ser tan crucial como la fase de preparación de los mismos. En cuanto a otros factores, como la disminución de la complejidad y la mejora de la calidad de los modelos producidos, el papel de la reducción de datos es igualmente decisivo.

Podemos agrupar las técnicas de reducción de datos en cuatro grupos: **selección de características**, consiste en reducir la dimensionalidad de los datos; **selección de instancias** y **selección de prototipos**, para eliminar muestras redundantes y/o conflictivas; **discretización**, que divide en intervalos el dominio continuo de un atributo; **extracción de características y generación de instancias**, para rellenar posibles huecos en los conjuntos de datos.

Seguidamente pasamos a describir con más detalle el conjunto de técnicas de reducción de datos que han sido utilizadas en esta tesis.

#### Selección de características

El objetivo de la selección de características es identificar las características del conjunto de datos que son importantes, y descartar las que sean redundantes o irrelevantes. Dado que la selección de características reduce la dimensionalidad de los datos, los algoritmos de minería de datos pueden ser más eficientes en tiempo de ejecución y obtener mejores resultados [11]. Por lo tanto, podemos definir la selección de características como un proceso que elige un subconjunto óptimo de características de acuerdo con un determinado criterio [75]. La selección del criterio debe hacerse de acuerdo con los propósitos de la selección de características.

Los motivos para realizar selección de características son numerosos [95]: eliminar los datos irrelevantes, aumentar la exactitud de la predicción de los modelos aprendidos, reducir el coste de los datos, mejorar la eficiencia del aprendizaje o reducir la complejidad de la descripción del modelo resultante, mejorando la comprensión del mismo.

La selección de características puede considerarse como un problema de búsqueda, en el que cada estado del espacio de búsqueda corresponde a un subconjunto concreto de características seleccionadas. La estrategia de búsqueda utilizada de-

penderá del número total de características del conjunto de datos, pudiendo utilizar algoritmos de fuerza bruta que obtendrán el subconjunto óptimo cuando el número de características es bajo, o bien, técnicas de búsqueda heurística que seguramente encontrará un subconjunto no óptimo de características pero en un tiempo razonable.

Otro factor a tener en cuenta es cómo medir la calidad o bondad de un subconjunto de características. Normalmente, las métricas de evaluación funcionan en dos direcciones: 1) buscando el rendimiento en términos de eficacia y 2) buscando el rendimiento en términos de eficiencia. La elección de la métrica de evaluación influye fuertemente en el algoritmo, y son estas métricas las que distinguen entre las tres categorías principales de algoritmos de selección de características: métodos *wrapper*, métodos de filtro y métodos embebidos [76].

- Los **métodos de envoltura** o ***wrappers*** [69] utilizan un modelo predictivo para puntuar los subconjuntos de características. Cada nuevo subconjunto se utiliza para entrenar un modelo, que se evalúa con un conjunto de reserva (*hold-out set*). Contando el número de errores cometidos en ese conjunto (la tasa de error del modelo) se obtiene la puntuación para ese subconjunto. Como los métodos de envoltura entrenan un nuevo modelo para cada subconjunto, son muy intensivos en computación, pero generalmente proporcionan el conjunto de características de mejor calidad.

- Los métodos de **filtro** funcionan independientemente del método de aprendizaje empleado posteriormente. El nombre "filtro"procede de filtrar las características indeseables antes de aprender. Utilizan heurísticas basadas en las características generales de los datos para evaluar la bondad de los subconjuntos de características. Un modelo de filtro consta de dos etapas: 1) la selección de características utilizando medidas como la información, la distancia, la dependencia o la coherencia; 2) aprendizaje y prueba, el algoritmo aprende de los datos de entrenamiento con el mejor subconjunto de características obtenido y probado sobre el conjunto de test. Los métodos de filtros son normalmente menos intensivos computacionalmente que los métodos *wrappers*, pero producen un conjunto de características que no está relacionado con el algoritmo de aprendizaje empleado posteriormente. Lo cual normalmente da como resultado un rendimiento más bajo que un conjunto resultante de un método de envoltura, hablando en términos de eficacia.

- Los métodos **embebidos** [62] son similares al enfoque *wrappers* en el sentido de que las características se seleccionan específicamente para un determinado algoritmo de aprendizaje. Pero en este enfoque, las características se seleccionan durante el proceso de aprendizaje. Los métodos embebidos que integran la selección de características como parte del proceso de entrenamiento pueden ser más eficientes en varios aspectos: pueden aprovechar todos los datos disponibles al no requerir la división en conjuntos de entrenamiento y test; pueden lograr una solución más rápida al no tener que re-entrenar un modelo para cada subconjunto de características explorado.

La selección de características ha sido ampliamente estudiada y aplicada a problemas de clasificación clásicos. Sin embargo, en el caso de MLL y el LDL, tenemos

que dar un paso más, ya que la estrategia de seleccionar un conjunto de características compartidas por todas las etiquetas puede no ser óptima, debido a que cada etiqueta puede ser descrita por un subconjunto específico de características propias. El aprendizaje de características específicas a cada etiqueta se ha estudiado ampliamente en problemas de clasificación MLL. Por ejemplo, LIFT [117] construye subconjuntos de características específicas de cada etiqueta utilizando técnicas de *clustering*. LLSF [67] propone el aprendizaje de las características específicas de cada etiqueta de clase considerando las correlaciones de las etiquetas por pares (es decir, de segundo orden). MLFC [116] es también un método de aprendizaje de múltiples etiquetas, que intenta aprender las características específicas de cada etiqueta explotando las correlaciones entre ellas. En cuanto a LDL podemos citar la propuesta LDLSF [91]. Este último es un método inspirado en el LLSF adaptado para tratar los problemas de LDL mediante la selección conjunta de características específicas de las etiquetas, la selección de características comunes y la explotación de las correlaciones de las etiquetas. Sin embargo, los estudios que podemos encontrar sobre este tema para el LDL son todavía muy escasos.

#### Selección de instancias

Un problema frecuente en los conjuntos de datos reales es su gran volumen, así como la presencia de ruido y anomalías que complican el proceso de aprendizaje. La selección de instancias consiste en escoger las muestras más representativas de un conjunto de datos determinado pero manteniendo la estructura esencial y la integridad del conjunto de datos inicial. Teniendo como objetivo el igualar o mejorar el rendimiento de los algoritmos de aprendizaje cuando utilizan todos los datos disponibles [74].

La selección de instancias se ha aplicado con éxito a varios tipos de problemas como, aprendizaje no balanceado [80], flujos de datos [90], regresión [100], descubrimiento de subgrupos en conjuntos de datos de gran tamaño [13], etc. Sin embargo, hasta la fecha, la selección de instancias no se ha investigado ampliamente en el ámbito del MLL y hasta la fecha sólo se han publicados unos pocos estudios al respecto [5, 20, 68]. En cuanto a LDL, no hemos podido encontrar ningún estudio hasta la fecha.

#### Selección de prototipos

En un principio, varias propuestas para seleccionar los datos más relevantes del conjunto de entrenamiento se propusieron pensando principalmente en el algoritmo $k$-NN [25]. Posteriormente, cuando el aprendizaje basado en instancias [3], también conocido como aprendizaje perezoso o *Lazy Learning* [2], fue acuñado, reuniendo todos aquellos métodos que no realizan una fase de entrenamiento durante el aprendizaje, el término de selección de prototipos surgió en la literatura. Hoy en día, el conjunto de métodos de selección de instancias también incluye las propuestas pensadas para trabajar con otros métodos de aprendizaje, como árboles de decisión, ANNs o SVM. Con el objetivo de diferenciarlos del resto, los métodos de selección de prototipos [47] son métodos de selección de instancias que buscan encontrar con-

juntos de entrenamiento que ofrezcan la mejor precisión de clasificación y tasas de reducción utilizando clasificadores basados en instancias.

Una categorización ampliamente utilizada de los métodos de selección de prototipos distingue tres tipos de técnicas [47]:

- **Condensación**, donde el objetivo es mantener los puntos fronterizos, preservando la precisión del sistema de entrenamiento.
- **Edición**, donde el objetivo es eliminar puntos límite que se consideran ruido o que no coinciden con sus vecinos pero sin eliminar los puntos internos de cada conjunto de datos.
- **Métodos híbridos**, que tratan de encontrar un pequeño conjunto de datos de entrenamiento manteniendo el rendimiento del clasificador.

El mejor enfoque que tiene en cuenta el equilibrio entre reducción y precisión suele ser la técnica híbrida.

Los métodos de selección de prototipos son, hoy en día, ampliamente usados dentro del campo de la reducción de datos. En la literatura se han presentado varios enfoques de algoritmos que pueden estudiarse en las propuestas [47, 83].

#### Extracción de características/Generación de instancias

Es una ampliación tanto de la selección de características como de la selección de instancias, permitiendo la modificación de los valores internos que representan cada ejemplo o atributo. En la extracción de características [73], aparte de la operación de eliminación de atributos, los subconjuntos de atributos pueden fusionarse o pueden contribuir a la creación de atributos sustitutivos artificiales. En lo que respecta a la generación de instancias o prototipos [102], el proceso es similar en el sentido de los ejemplos. Permite crear o ajustar ejemplos de muestras artificiales, partiendo de las instancias originales, dando como resultado un nuevo conjunto de datos de entrenamiento extendido.

Un ejemplo de una técnica ampliamente utilizada para esta tarea es *Synthetic Minority Oversampling Technique* (SMOTE) [21]. Esta técnica de pre-procesamiento lleva a cabo una técnica de sobre-muestreo para re-equilibrar el conjunto de entrenamiento original. El concepto básico del procedimiento SMOTE es realizar una interpolación utilizando muestras de clases minoritarias vecinas para mejorar la capacidad de generalización del clasificador. La técnica de pre-procesamiento SMOTE se ha convertido en pionera en la comunidad de investigadores y se ha aplicado ampliamente en diversos escenarios como el reconocimiento facial, la ingeniería de software, el diagnóstico médico o las redes sociales. Debido a su popularidad e influencia, SMOTE está considerado como uno de los algoritmos de pre-procesamiento/muestreo de datos más influyentes en el aprendizaje automático y la minería de datos [37, 45].

### 2.4.2. Estrategias de descomposición

Las estrategias de descomposición son un tipo de transformación de datos, diseñadas específicamente para abordar problemas de clasificación multi-clase. Han

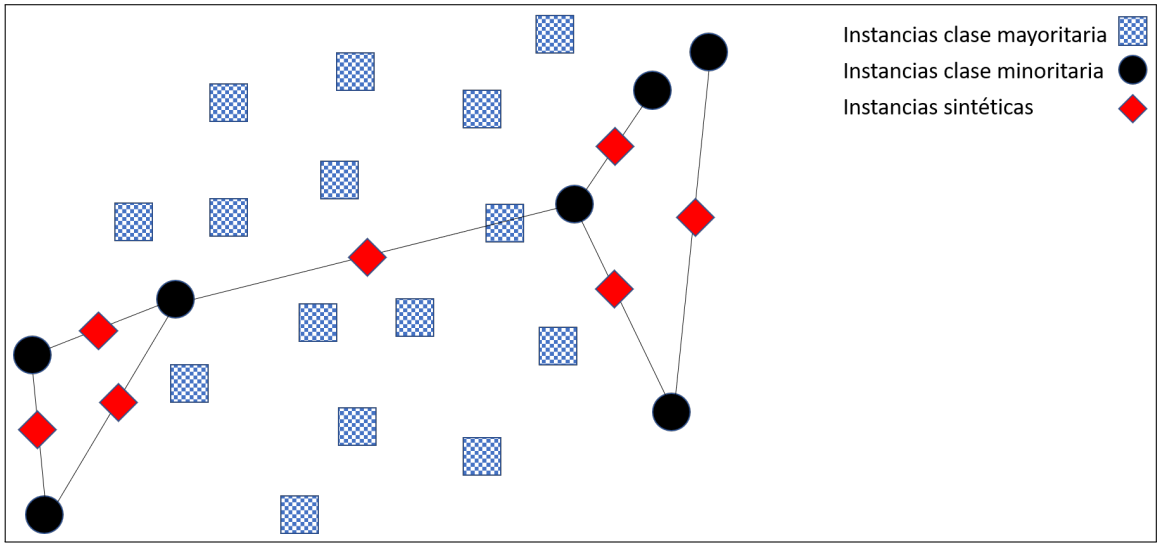



Figura 2.5: Ejemplo de utilización de SMOTE sobre conjunto de datos no balanceado.

sido ampliamente estudiadas en la literatura [39] y la misma idea subyacente está detrás de todas las propuestas de descomposición: resolver un problema de clasificación multi-clase usando clasificadores binarios. Siguiendo el paradigma de "divide y vencerás", el problema se divide en problemas de clasificación binaria más simples de abordar. Ahora bien, este método necesita de un paso adicional debido a la simplificación de los clasificadores de base: sus resultados deben ser recombinados para obtener el resultado final. La forma en que se lleva a cabo esta agregación es crucial para la calidad del resultado final. Una comparación exhaustiva de las estrategias de descomposición y los métodos de agregación se puede encontrar en [39].

Las estrategias de descomposición más comunes son *One-vs-All* (OVA) [93] y *One-vs-One* (OVO) [64], que pueden incluirse dentro del marco de trabajo ECOC (*Error Correcting Output Codes*) [29]. El primero aprende un clasificador binario para discernir entre cada par de clases, mientras que el segundo construye un clasificador binario para separar cada clase de todas las demás.

El enfoque OVA consiste en dividir el problema de clasificación multi-clase con $c$ clases en $c$ problemas de clasificación binaria. Cada clasificador binario se encarga de distinguir una de las clases de todas las demás. La fase de entrenamiento de cada clase se lleva a cabo utilizando el conjunto completo de entrenamiento, considerando como positivas las muestras de la clase única y como negativas todas las demás muestras. En la fase de validación, el ejemplo se clasifica utilizando cada uno de los clasificadores binarios. El clasificador que obtenga un resultado positivo mostrará la clase de salida. Obsérvese que la salida puede no ser única y en estos casos debe utilizarse algún tipo de mecanismo de desempate. Por ejemplo, podemos calcular la confianza de cada clasificador para predecir la salida final y seleccionar como resultado la la salida predicha por el clasificador con la mayor confianza.

El esquema de descomposición OVO [64] divide un problema de $c$ clases en $c(c-1)/2$ problemas binarios. Cada problema se aborda con un clasificador binario que distingue entre los diferentes pares de clases. La fase de aprendizaje de cada

clasificador se lleva a cabo utilizando un subconjunto de instancias que contiene una de las dos clases de salida. Las instancias con una clase diferente son ignoradas. En la fase de predicción, la muestra a clasificar es predicha por cada uno de los clasificadores entrenados previamente, obteniendo así una matriz de puntuación $R$:

$$R = \begin{pmatrix} - & r_{12} & r_{13} & ... & r_{1c} \\ r_{21} & - & r_{23} & ... & r_{2c} \\ . & . & . & . & . \\ . & . & . & . & . \\ r_{c1} & r_{c2} & r_{c3} & ... & - \end{pmatrix}$$

La salida de un clasificador dado por $r_{ji} \in [0, 1]$ es la confianza obtenida por el clasificador binario que discrimina las clases $i$ y $j$ en beneficio de la clase anterior. La confianza del clasificador para $j$ se calcula como $r_{ji} = 1 - r_{ij}$ si el clasificador no lo proporciona (la clase con mayor confianza es la clase de salida seleccionada de un clasificador). La predicción final se calcula en base a la matriz de puntuación utilizando diferentes modelos de agregación. La estrategia de voto ponderado es la más utilizada, en la que las confianzas se agregan clase por clase (por filas) y se selecciona como salida la que tiene la suma más alta.

Hay un inconveniente, conocido como el "problema del clasificador no competente" en el sistema OVO. Los clasificadores del sistema OVO no son suficientemente competentes para clasificar todas las clases del problema, ya que sólo aprenden a través de ejemplos de dos clases. Sin embargo, todos los clasificadores binarios se activarán para un muestra de prueba determinada, porque la confianza no puede conocerse a priori, lo que puede conducir a decisiones incorrectas. Para mitigar este problema existen técnicas de selección dinámica capaces de distinguir entre clasificadores competentes directamente en la fase de predicción. Los estudios realizados en [26] y [42] proporcionan una revisión de las técnicas de selección dinámica más populares para evitar el problema de los clasificadores no competentes. Por su parte, [40, 119] son algunos ejemplos de cómo aplicar con éxito la técnica OVO.

ECOC [29] proporciona un marco de trabajo matricial adecuado para modelar la descomposición de un problema de clasificación multi-clase en sub-problemas más simples. La forma de realizar la descomposición para ajustar mejor los datos utilizando un pequeño número de clasificadores ha sido un punto clave de la investigación, así como el paso de decodificación, que se ocupa de la combinación de los sub-problemas. La investigación [71] propone un marco unificado de pruebas que maneja tanto los pasos de codificación como de decodificación.

Las estrategias de OVO y OVA han demostrado ser eficaces para hacer frente a los problemas clásicos de los algoritmos de clasificación: conjuntos de datos no balanceados, datos con ruido, superposiciones o irregularidades. En [10, 36] podemos encontrar una completa revisión de cómo aplicar estrategias de descomposición a conjuntos de datos no balanceados que resuelvan la desproporción del número de instancias de las diferentes clases. Los resultados obtenidos en [46, 97] muestran que el uso de la estrategia OVO conduce a mejores rendimientos y a clasificadores más robustos cuando se trata de datos con ruido. La descomposición realizada por OVO en [96] ayuda a aumentar la separación entre clases, creando límites de decisión más regulares donde hay muestras superpuestas. Otras irregularidades, como el problema

de las "clases difíciles", también se han abordado con éxito utilizando la estrategia de descomposición de OVO en [41].

### 2.4.3. Estrategias prometedoras de pre-procesamiento para LDL

LDL es una generalización de la tarea de clasificación y por lo tanto es vulnerable a los mismos problemas que los algoritmos de clasificación convencionales: conjuntos de datos no balanceados, datos con ruido, superposiciones, irregularidades, etc. Sin embargo, a día de hoy, las propuestas de pre-procesamiento que podrían mejorar el rendimiento de los algoritmos de aprendizaje LDL existentes han sido muy escasamente estudiadas en el paradigma LDL. Las estrategias que describimos a continuación, que han sido aplicadas con éxito en problemas de clasificación convencionales, pueden extenderse para tratar también problemas de tipo LDL.

Observando los conjuntos de datos LDL disponibles, vemos que el número de muestras que contienen es bajo en comparación con la alta dimensionalidad de los vectores de características y de etiquetas. Esta escasez de datos puede provocar un mal rendimiento de los algoritmos de aprendizaje, hecho que podría remediarse añadiendo una etapa de pre-procesamiento que usara alguna técnica de generación de instancias. Este enfoque de *oversampling* que ha demostrado ser útil usado en problemas de clasificación clásica podría extenderse para ser aplicado al paradigma LDL.

Por otra parte, si queremos utilizar un algoritmo de aprendizaje basado en instancias, podríamos encontrarnos con problemas de rendimiento debido a la alta dimensionalidad de las características de entrada, que ralentizaría el proceso de cálculo de similitudes entre las muestras. Además, estos métodos suelen presentar baja tolerancia al ruido debido a que utilizan todos los datos como igualmente relevantes. Para tratar de paliar estos efectos negativos podemos utilizar técnicas de reducción de datos como la selección de características o la selección de prototipos. Existen algunos métodos para abordar la selección de instancias en el dominio MLL [5, 20, 68] pero, hasta donde sabemos, no hay estudios sobre la selección de instancias o prototipos reportados en LDL. Con respecto a la selección de características, la mayoría de los algoritmos de LDL se construyen en un espacio de características simple donde todas las características se utilizan para predecir todas las etiquetas. Sin embargo, en las aplicaciones del mundo real, una instancia se caracteriza por todas las etiquetas, pero algunas etiquetas sólo pueden determinarse por algunas características específicas propias. Aunque la selección de características específicas de las etiquetas ha sido ampliamente estudiada en MLL [67, 117], los estudios que podemos encontrar sobre este tema para LDL son todavía escasos aunque prometedores [91].

Una transformación de datos que ha resultado ser eficaz para hacer frente a los problemas de clasificación convencional es la aplicación de estrategias de descomposición. Dichas estrategias han sido ampliamente utilizadas para abordar problemas multi-clase y podrían extenderse para lidiar con la clasificación LDL. Los problemas de tipo LDL son complejos debido al alto número tanto de características como de

etiquetas de salida. Dicha complejidad hace que sean difíciles de abordar con algoritmos de clasificación convencionales, por lo que aplicar alguna de las estrategias analizadas en el Apartado 2.4.2 y dividiendo el problema original en sub-problemas más simples de tratar, podría llevar a una mejora de rendimiento tanto en eficacia como en tiempo de ejecución. Ahora bien, para conseguir aplicar con éxito un método de descomposición, habrán de tenerse en cuenta los múltiples factores diferenciadores entre MLL y LDL y extender los algoritmos en consecuencia.

En el siguiente Capítulo 3 proponemos tres estudios que desarrollan estas prometedoras estrategias de pre-procesamiento especialmente diseñadas para abordar problemas de tipo LDL.

# Capítulo 3

# Discusión de las Propuestas

## 3.1. Generación Sintética de Muestras para el Aprendizaje de la Distribución de Etiquetas

El propósito de este artículo es abordar el problema LDL desde la etapa de pre-procesamiento de datos [45, 48]. Estudiando los conjuntos de datos de LDL existentes, observamos que el número de muestras que contienen es significativamente bajo en comparación con la alta dimensionalidad de las características de entrada así como de las etiquetas de salida. Esto nos lleva a la hipótesis de que los conjuntos de datos con escasas muestras conducirán a un pobre rendimiento una vez aplicados a los algoritmos de aprendizaje.

El objetivo perseguido es mejorar el rendimiento de los clasificadores LDL subyacentes sin modificarlos, añadiendo una etapa preliminar de pre-procesamiento aplicada al conjunto de datos original. Se trata de una capa adicional que puede añadirse de forma sencilla al proceso de aprendizaje.

En los últimos años, se han propuesto muchas soluciones para tratar este problema. Una de las más conocidas y utilizadas es la técnica de muestreo de datos [6] en la que las instancias de entrenamiento se modifican de manera que se produzca una distribución de clases más eficiente que permita a los clasificadores mejorar su rendimiento.

En la literatura especializada, podemos encontrar varios estudios que muestran el efecto del remuestreo del conjunto de entrenamiento para tratar conjuntos de datos incompletos. Estos estudios han demostrado empíricamente que aplicar una etapa de pre-procesamiento suele ser una solución útil [22, 94]. Además, la principal ventaja de estas técnicas es que son independientes de los algoritmos de clasificación subyacentes.

A día de hoy, los investigadores han propuesto algunos intentos de aplicar el pre-procesamiento en LDL, como el algoritmo de selección de características descrito en [107], pero el enfoque de *oversampling* aún no ha sido estudiado ni aplicado al paradigma LDL. En este trabajo, hemos desarrollado un método de *oversampling* [87] que crea un superconjunto de datos partiendo del conjunto de datos original creando nuevas instancias a partir de las existentes. La técnica ideada se basa en uno de los enfoques más utilizados en este área llamado: *Synthetic Minority Oversampling Technique* (SMOTE) [21]. En pocas palabras, el principal objetivo es crear

nuevas muestras de las clases minoritarias para sobremuestrear el conjunto de entrenamiento interpolando varias instancias de clases minoritarias que se encuentran juntas.

La Generación Sintética de Muestras para LDL (*Synthetic Sample Generation for LDL*), o a partir de ahora SSG-LDL, es nuestra propuesta de un algoritmo de pre-procesamiento adaptado a las restricciones de LDL. Partimos de la estructura básica de SMOTE, que ampliamos para poder tratar los conjuntos de datos de LDL. En nuestro caso debemos alejarnos de SMOTE porque no queremos tratar con clases minoritarias, sino que queremos crear muestras sintéticas de los puntos más distantes espacialmente. Para ello, también introduciremos una nueva técnica de selección de puntos remotos, inspirada en la selección probabilística utilizada en los algoritmos genéticos [54]. Otro de los principales retos ha sido cómo generar muestras sintéticas a partir de las disponibles en el conjunto de entrenamiento ya que, a diferencia de SMOTE, no sólo tenemos que ocuparnos de la dimensión de las características sino que también debemos manejar la dimensión de las etiquetas para obtener una muestra sintética compatible con LDL.

Para evaluar la eficacia de la propuesta, comparamos tres clasificadores LDL de última generación aplicando nuestra etapa de pre-procesamiento SSG-LDL al conjunto de entrenamiento y midiendo seis aspectos de su desempeño. Repetimos el experimento con 15 conjuntos de datos del mundo real y validamos los resultados de las comparaciones empíricas utilizando las pruebas de Wilcoxon y tests de signos bayesianos [9, 15, 28].

La publicación asociada a esta propuesta se encuentra en el Capítulo 4.

## 3.2. Selección de Prototipos y de Características Específicas de las Etiquetas mediante Optimización Evolutiva para el Aprendizaje de la Distribución de Etiquetas

El método AA-$k$NN [49], adaptación a LDL del conocido $k$-NN, ha demostrado ser un algoritmo muy competitivo en estudios experimentales previos, logrando resultados aceptables y permitiendo un modelo explicable [7]. Sin embargo, como cualquier otro algoritmo basado en instancias, adolece de varios inconvenientes: necesita grandes requisitos de memoria para almacenar el conjunto de entrenamiento, no es eficiente en la predicción debido a los múltiples cálculos de similitudes entre las muestras de test y entrenamiento y presenta una baja tolerancia al ruido porque utiliza todos los datos como relevantes.

Con el fin de mitigar estos problemas, el propósito de este trabajo es abordar el problema desde la etapa de pre-procesamiento, aplicando técnicas de reducción de datos que nos permitirán tener una representación reducida del conjunto original pero manteniendo la estructura y la integridad del mismo [48].

Dos de las técnicas de reducción de datos más utilizadas que ya adelantamos en la Sección 2 son la selección de instancias [74], también conocida como selección

de prototipos en el caso de los algoritmos basados en instancias [47], y la selección de características o reducción de la dimensionalidad de los datos [101]. La primera técnica se centra en la búsqueda de un subconjunto óptimo de muestras para optimizar el rendimiento del clasificador, mientras que la idea principal de la segunda técnica es sustituir el conjunto original de características por un nuevo subconjunto que extraiga la información principal y proporcione una clasificación precisa.

La Selección de Prototipos y de Características Específicas de las Etiquetas mediante Optimización Evolutiva para el Aprendizaje de la Distribución de Etiquetas (*Prototype selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning*), o de aquí en adelante ProLSFEO-LDL, es nuestra propuesta de un algoritmo de pre-procesamiento adaptado a los problemas de LDL y diseñado específicamente para mitigar las desventajas del método AA-$k$NN. Presentamos un método novedoso para abordar simultáneamente la selección de prototipos y la selección de características específicas a cada etiqueta de salida. Ambas técnicas de pre-procesamiento pueden considerarse como un problema de búsqueda donde el espacio de búsqueda es enorme. Por lo tanto, hemos ideado un método de búsqueda basado en algoritmos evolutivos [121] que nos permite obtener una solución a ambos problemas en un tiempo razonable. Para ello, hemos adaptado elementos de los algoritmos evolutivos clásicos para gestionar las restricciones de LDL, hemos diseñado una representación de la solución y una forma de medir su calidad, lo que nos permite abordar ambos problemas conjuntamente.

Para evaluar la eficacia y eficiencia de la propuesta, comparamos un clasificador LDL aplicando nuestro algoritmo ProLSFEO-LDL al conjunto de entrenamiento en bruto y midiendo seis aspectos de su rendimiento. Repetimos el experimento con 13 conjuntos de datos del mundo real y validamos los resultados de las comparaciones empíricas utilizando las pruebas de Wilcoxon y tests de signos bayesianos [9, 15, 28].

La publicación asociada a esta propuesta se encuentra en el Capítulo 5.

## 3.3. Descomposición-Fusión para el Aprendizaje de la Distribución de Etiquetas

LDL es una generalización de los métodos de clasificación [104] y, por lo tanto, es vulnerable a los mismos problemas que los algoritmos de clasificación convencionales: conjuntos de datos desequilibrados [36], cuando hay una desproporción en el número de ejemplos de las diferentes clases; datos con ruido [88], debido a imperfecciones en la adquisición, transmisión o almacenamiento de los datos; superposición [86], cuando las características de entrada no son suficientes para diferenciar correctamente entre los casos de diferentes clases; o irregularidades [27], situaciones en que la distribución de los puntos de datos, el muestreo del espacio de datos para generar el conjunto de entrenamiento y las características que describen cada punto se desvían de lo que podría haber sido ideal, siendo sesgadas, incompletas y/o confusas.

En los últimos años, las estrategias de descomposición para abordar los problemas de clasificación complejos han sido ampliamente estudiadas en la literatura [97]. La

misma idea subyacente está detrás de todas las propuestas para la descomposición: resolver un problema multi-clase utilizando clasificadores binarios. Las estrategias de descomposición han demostrado ser eficaces para hacer frente a las dificultades que se han presentado anteriormente y por eso, en este estudio, proponemos un algoritmo de descomposición adaptado a las restricciones de LDL. La técnica ideada se inspira en una de las estrategias más conocidas en materia de descomposición: el esquema OVO, en el que el problema original se divide en problemas binarios que distinguen los diferentes pares de clases, cada división se entrena con un clasificador base. Este método suele requerir un paso adicional para fusionar las salidas de los clasificadores simples con el fin de producir el resultado final.

Diseñamos una estrategia de descomposición que puede manejar la distribución de etiquetas de salida, capaz de tratar con valores reales en lugar de salidas multi-clase. Mientras que OVO utiliza un clasificador binario como clasificador base (el algoritmo de aprendizaje utilizado para resolver problemas binarios), en nuestra propuesta nos basamos en un clasificador LDL específico. Además, también proponemos un método de fusión, capaz de proporcionar una salida de acuerdo con las restricciones de LDL, y que también nos permitirá descartar los clasificadores no competentes cuando su salida probablemente no sea de interés.

La Descomposición-Fusión para el Aprendizaje de la Distribución de Etiquetas (*Decomposition-Fusion for LDL*), desde ahora DF-LDL, es nuestra propuesta de descomposición para los problemas de tipo LDL. Para evaluar la eficacia de la propuesta, llevamos a cabo dos tipos de experimentos: por un lado, comparamos los resultados obtenidos por los clasificadores base con nuestro algoritmo DF-LDL utilizando el mismo clasificador como clasificador base y, por otro lado, comparamos nuestra propuesta con los algoritmos LDL "estado-del-arte", midiendo en todos los casos seis aspectos de su rendimiento. Repetimos el experimento en 17 conjuntos de datos del mundo real y validamos los resultados de las comparaciones empíricas utilizando las pruebas de Wilcoxon, la prueba de Friedman y tests de signos bayesianos [9, 28].

La publicación asociada a esta propuesta se encuentra en el Capítulo 6.

# Capítulo 4

# Generación Sintética de Muestras para el Aprendizaje de la Distribución de Etiquetas

González, M., Luengo, J., Cano, J. R., & García, S. (2021) Synthetic Sample Generation for Label Distribution Learning. Information Sciences, 544, 197-213.

| | **JCR Clarivate**<br>IF: 5.910 | **CiteScore Scopus**<br>11.3 |
|---|---|---|
| **Categoría** | Computer Science, Information Systems | Computer Science Applications |
| **Ranking** | 9/156 | 26/636 |
| **Cuartil** | Q1 | Q1 |

Factores de Impacto 2019

# Synthetic Sample Generation for Label Distribution Learning

**Manuel González**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
manuel.gonzalez.es@gmail.com

**Julián Luengo**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
julianlm@decsai.ugr.es

**José-Ramón Cano**
Department of Computer Science
University of Jaén, 23700 Linares, Jaén, Spain
jrcano@ujaen.es

**Salvador García**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
salvagl@decsai.ugr.es

**Abstract**

Label Distribution Learning (LDL) is a general learning framework that assigns an instance to a distribution over a set of labels rather than a single label or multiple labels. Current LDL methods have proven their effectiveness in many machine learning applications. As of the first formulation of the LDL problem, numerous studies have been carried out that apply the LDL methodology to various real-life problem solving. Others have focused more specifically on the proposal of new algorithms. The purpose of this article is to start addressing the LDL problem as of the data pre-processing stage. The baseline hypothesis is that, due to the high dimensionality of existing LDL data sets, it is very likely that this data will be incomplete and/or that poor data quality will lead to poor performance once applied to the learning algorithms. In this paper, we propose an oversampling method, which creates a superset of the original dataset by creating new instances from existing ones. Then, we apply already existing algorithms to the pre-processed training set in order to validate the efficacy of our method. The effectiveness of the proposed SSG-LDL is verified on several LDL datasets, showing significant improvements to the state-of-the-art LDL methods.

**Keywords**: label distribution learning, data pre-processing, oversampling, machine learning

# 1. Introduction

A supervised learning process essentially consists of assigning a label or set of labels to each of the samples. In existing learning paradigms, there are mainly two cases of label assignment: (1) a single label is assigned to an instance, and (2) multiple labels are assigned to each instance. Single-label learning (SLL) assumes that all instances in the training set are labeled according to the first case. Multi-Label Learning (MLL) [29, 40] allows instances in the training set to be labeled using the second method. Thus, MLL can deal with ambiguity when an instance belongs to more than one class (label) [40].

However, in real-life problems, we can find cases for which MLL is not sufficient since the level of description of each label is not the same. The Label Distribution Learning (LDL) concept showed up for first time in 2013 [26] and was formally described in 2016 [23] in order to deal with label ambiguity in mapping when one instance is not necessarily mapped to one label. The aim of this paradigm is to answer the question "how much does each label describe the instance?" instead of "which label can describe the instance?" [23].

From the first formulation of the LDL problem, numerous studies have been carried out applying the LDL methodology to various real-life problem solving situations, e.g., sense of beauty recognition [34], facial age estimation [26], personality recognition in social media [44], image emotion classification [45], pre-release prediction of crowd opinion on movies [24], crowd counting in public video surveillance [49], ... Other studies have focused more particularly on the creation of new learners or the improvement of existing learners such as: AA-$k$NN, SA-IIS, SA-BFGS [23], LDL forests [36], Logistic boosting regression for LDL [43], Deep Label Distribution [20], LDL-SCL [50], Structured random forest for LDL [14].

The purpose of this article is to address the LDL problem from the data pre-processing stage [21, 22]. Studying the existing LDL datasets, we observe that the number of samples they contain is significantly low as opposed to the high dimensionality of input characteristics as well as output labels. This leads us to the hypothesis that the datasets with sparse data will lead to poor performance once applied to the learning algorithms.

The pursued objective is to improve the performance of the underlying LDL learners without modifying them, by adding a preliminary pre-processing stage applied to the original dataset. It is an extra layer that can be added in a simple way to the machine learning process.

Over the last few years, many solutions have been proposed to deal with this problem. One of the most well-known and commonly used is the data sampling technique [4] in which the training instances are modified in such a way as to produce a more efficient class distribution that allow learners to perform in a similar manner to standard learning.

In the specialized literature, we can find several studies that show the effect of resampling the training set to treat incomplete datasets. Those studies have empirically proved that, applying a pre-processing step is usually a useful solution [12, 35]. Furthermore, the main advantage of these techniques is that they are independent of the underlying learner.

Nowadays, researchers have proposed a few attempts to apply pre-processing in LDL, as the feature selection algorithm described in [42], but the oversampling approach has not yet been studied or applied to the LDL paradigm and that is why, in this paper, we have developed an oversampling method [33], which creates a superset of the original dataset by creating new instances from existing ones. The technique devised is based on one of the most renowned approaches in the oversampling area called: "Synthetic Minority Oversampling Technique" (SMOTE) [11]. Briefly, its main objective is to create new minority class examples to oversample the training set by interpolating several minority class instances that lie together.

The problem statement can thus be described as building a new optimized dataset, an extension of the original, which improves the performance of the underlying learners.

Synthetic Sample Generation for LDL, or from now on SSG-LDL, is our proposal for a pre-processing algorithm adapted to LDL restrictions. We will start from the basic structure of SMOTE, which we will extend to be able to deal with LDL datasets. In our case we have to take a step away from SMOTE because we do not want to deal with minority classes, rather we want to create synthetic samples of the most spatially distant points. To this end, we will also introduce a new technique to select remote points, inspired by the probabilistic selection used in genetic algorithms [27]. Another main challenge is how to generate synthetic samples from the ones available in the training set since, unlike SMOTE, we do not only have to deal with the dimension of the characteristics but we must also handle the dimension of the labels to get a synthetic sample that is compatible with LDL. It is noteworthy to mention that SSG-LDL is not a simple data augmentation such as those commonly used in deep learning to preprocess and to enlarge the set of training images, such as those used in [20].

In order to evaluate the proposal proficiency, we will compare three state-of-the-art LDL learners applying our SSG-LDL pre-processing step to the training set and measuring six aspects of their performance. We will repeat the experiment over 15 real-world datasets and validate the results of the empirical comparisons using Wilcoxon and Bayesian Sign tests.

The rest of the paper is organized as follows. First, a brief review and discussion of the related work on LDL and synthetic feature generation are given in Section 2. The proposed Synthetic Sample Generation LDL pre-processing method is described in Section 3. Then the details of the experiments are reported in Section 4. Finally, the results and conclusions are drawn in Section 5 and Section 6.

## 2. Preliminaries

In this section, the formulation of LDL (Section 2.1) and the LDL algorithms considered for our case study (Section 2.2) are presented. Furthermore, some basic concepts on synthetic sample generation are introduced (Section 2.3), providing the necessary background required to properly present the study carried out in this paper.

## 2.1. Formulation of Label Distribution Learning

Suppose that we are given a set of $m$ samples $S = \{(x_1, D_1), ..., (x_m, D_m)\}$ to train, where $x_i = \{x_{i1}, x_{i2}, ..., x_{iq}\}$ is a $q$-dimensional vector. For each instance $x_i$, the label distribution is denoted by $D_i = \{d_{x_i}^{y_1}, d_{x_i}^{y_2}, ..., d_{x_i}^{y_c}\}$ where $y \in Y, Y = \{y_1, y_2, ..., y_c\}$ denotes the complete set of labels. The constant $c$ is the number of possible labels and $d_{x_i}^{y_j}$ is the description degree of the particular $j$th label $y_j$ to the particular $i$th instance $x_i$. According to the definition, each description degree should satisfy the constraints $d_x^y \in [0, 1]$ and $\sum_y d_x^y = 1$.

## 2.2. Label Distribution Learning Algorithms

The resolution of an LDL problem can be approached from different points of view. Depending on the approach chosen, the algorithm to be developed will vary substantially, being either a completely new algorithm developed specifically to deal with LDL constraints, or an adaptation of existing classification algorithms, reformulated to work with such constraints. In all cases, the algorithms chosen for our study are "state of the art" algorithms.

The first one is the AA-$k$NN adaptation described in [23], given a new instance $x$, its $k$ nearest neighbors are first found in the training set. Then, the mean of the label distributions of all the $k$ nearest neighbors is calculated as the label distribution of $x$. We have chosen this algorithm due to its simplicity and the ease with which it can be enhanced, it will provide a good starting point for a comparative analysis with other more complex methods.

The second is the specialized algorithm SA-BFGS that learns by optimizing an energy function based on the maximum entropy model using a quasi-Newton method. Of the six LDL algorithms proposed by Geng [23], the SA-BFGS stood out due to its great advantage in precision and efficiency.

Another state-of-the-art specialized algorithm is called Structured Random Forest (StructRF) [14]. It is a general LDL model that treats the distribution as an integral whole. In StructRF, all label distributions are mapped to a discrete space at each split node in a random forest. StructRF has proven to be able to train fast and it reaches higher accuracies and lower standard deviations among different measurements.

## 2.3. Motivation for using oversampling technology

Nowadays, one of the main challenges for supervised learning algorithms is still how to deal with datasets that are either incomplete or have significantly skewed class distributions. Both problems can be addressed using an initial data preprocessing that allows us to improve the quality obtained by the learning algorithm used.

The synthetic generation of samples allows us to approach this problem by generating new artificial samples from the originals, thus creating an expanded set of data that we can plug in as input to our learning algorithm.

An example of a widely used technique that achieves this task is the Synthetic Minority Oversampling Technique now widely known as SMOTE [11]. This prepro-

cessing technique carries out an oversampling technique to rebalance the original training set. The basic concept of the SMOTE procedure is to accomplish an interpolation using neighboring minority class samples to improve the generalization capacity of the classifier. The SMOTE preprocessing technique has become a pioneer in the research community and has been widely applied in different scenarios such as face recognition, social media, software engineering, medical diagnosis or social networks. Due to its popularity and influence, SMOTE is considered to be one of the most influential data preprocessing/sampling algorithms in machine learning and data mining [19, 22].

As a result, the synthetic generation of samples has been used successfully in multiple areas of study such as:

- Dynamic environments in which data arrives continuously, or data stream. An example of a pre-processing technique to deal with these data streams is Learn++.NSE-SMOTE [16].

- When a data stream is received over time and we have time information at our disposal, we refer to time series classification. The methods SPO and INOS [7] propose an integration of SMOTE in time series classification.

- In a perfect situation, we want to train classifiers using diverse labeled data that thoroughly represent all classes. However, in many real-life applications, there is a huge amount of unlabeled data and thus obtaining a representative subset is a complex process. We refer to semi-supervised classification that utilizes unlabeled data to improve the predictive performance. Several methods based on SMOTE have been developed for this learning paradigm, GS4 [32], SEG-SSC [38] and OCHS-SSC [17]. These methods generate synthetic examples to diminish the drawbacks produced by the absence of labeled examples.

- Several ideas, also based on SMOTE, have been proposed to tackle multi-instance learning, e.g. Instance-SMOTE and Bag-SMOTE [41]. The Instance-SMOTE algorithm creates synthetic minority instances in each bag, without creating new bags. Besides, Bag-SMOTE creates new synthetic minority bags with new instances.

- In multilabel classification [29] each data instance is associated with a vector of outputs, instead of only one value. MLSMOTE [10] is the most popular extension of SMOTE designed for multilabel classification. Its objective is to produce synthetic instances related to minority labels.

- Regression tasks consider the output variable as continuous and hence, the values are represented by real numbers. SMOTER is the SMOTE-based contribution of oversampling regression [37].

## 2.4. Principles of the proposed method

Up until now, all existing LDL studies and algorithms have dealt with raw datasets [14, 20, 23, 36, 43, 50] and have not applied any kind of pre-processing that might later facilitate the learning of the algorithm. The pursued objective of this study is to improve the performance of the underlying LDL learners without modifying them. To achieve this goal we have created an oversampling method adapted to the LDL restrictions in order to extend the initial datasets by applying synthetic sample generation and use them as input for the state-of-the-art LDL algorithms. Later we will analyze the results and see if they can improve the results obtained by learners. The mechanism of the proposed method can be depicted as in Figure 4.1.

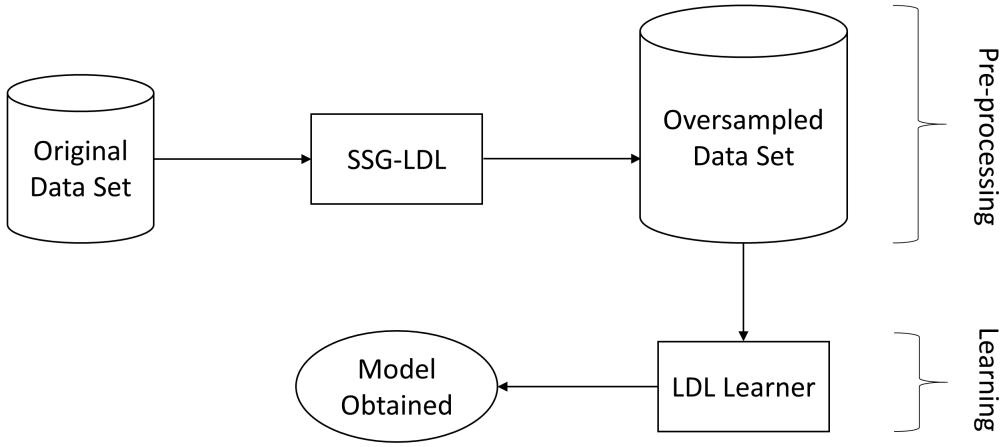



Figure 4.1: Principle of the proposed method

# 3. Synthetic Sample Generation for Label Distribution Learning

SSG-LDL is the proposed pre-processing algorithm that has been adapted to LDL restrictions.

The main idea of this method is to obtain an optimized dataset that will be made up of the original samples plus the artificially generated ones. We define a synthetic sample as a sample generated based on a set of real samples, on which we apply interpolation techniques between remote points that are within a defined neighborhood. The new samples obtained using this procedure are thus added to the raw dataset. Throughout the following subsections we will explain the entire process in detail, covering from how to select the candidate samples to how to obtain a new synthetic sample specially designed to meet the constraints of an LDL problem.

The formal procedure works as follows. First, the total percentage of oversampling $N$ (an integer value) is set up, then, an iterative process is carried out,

composed of several steps. In the initial step, a remote sample is selected at random from the training set (this process is described in Section 3.2). Next, its $k$ nearest neighbors are obtained. Finally, one of these $k$ instances is randomly chosen to compute the new sample by interpolation. This synthetic sample generation process is described in detail in Section 3.3 where we will explain how to interpolate the features vector and also how the label dimension is created using the $k$ nearest neighbor label vectors. This process is repeated iteratively until $N \times m/100$ synthetic samples are obtained. The whole process is summarized in Algorithm 1.

**Algorithm 1:** SSG-LDL

**Function** `SSG-LDL(`*S[], N, k*`):`

```
1   input  : S[] ← original training set ;
          N ← % of oversampling ;
          k ← considered neighbors ;
2   output: S'[] ← the oversampled training set
3   S' = S ;
4   T = m * N/100 ← nº synthetic samples to create;
5   for i=1 to T do
6       j = SelectSample(S) ;
7       ss = CreateSyntheticSample(S, j, k) ;
8       S' ← ss ;
9       i = i + 1;
10  end
```

**End Function**

## 3.1. Distances

As previously mentioned, the first step of the algorithm consists in selecting a random sample located far away from the others. To measure the distance between two samples we have chosen the Euclidean distance, adapted to LDL, that we will apply not only to the dimension of the features ($x$) but also to the dimension of the labels ($D$). The two independent parameters $f_x$ and $f_y$ allow us to have two degrees of freedom when calculating the distance between two samples. Keeeping this in mind is of interest if a wider space must be covered when creating synthetic LDL samples. Therefore, the final distance between two samples is the sum of the feature's distance multiplied by the factor $f_x$ and the label's distance multiplied by the factor $f_y$. What we calculate at this step, and what will later help us to select remote samples, is the average distance between a sample and all the others as summarized in Equation 4.1.

$$\overline{DIST}[x_i] = f_x \frac{\sum_{l=1}^{m} \text{euclidean}(x_i, x_l)}{m} + f_y \frac{\sum_{l=1}^{m} \text{euclidean}(D_i, D_l)}{m}. \tag{4.1}$$

## 3.2. Divergence Cumulative Selection

A part of the SSG-LDL algorithm is based on choosing isolated samples from which synthetic samples are generated in order to fill in the data set in less populated areas. To do so we will rely on the distances calculated in the previous section, adding some randomness to the process. The implemented method is inspired by the selection mechanism called "roulette wheel selection", widely used in genetic algorithms [27].

In divergence cumulative selection, the probability of being selected will increase as the distance increases. This has been achieved by dividing the mean distance of a sample to other samples by the total distance, thereby normalizing them to 1 (cumulative probability vector). Then a random selection is made, similarly to how a roulette wheel is spun. Therefore, the probability $P$ that sample $x_i$ will be selected is defined according to Equation 4.2.

$$P_{x_i} = \frac{\overline{DIST}[x_i]}{\sum_{j=1}^{m} \overline{DIST}[x_j]}. \tag{4.2}$$

The whole process is summarized in Algorithm 2. The algorithm generates a random number between 0 and the sum of the frequencies. Then, it goes through the array of frequencies, producing a running total. At some point the running total exceeds the generated threshold. The index at that point corresponds to the selected instance.

**Algorithm 2:** Divergence Cumulative Selection Algorithm

```
Function SelectSample(S[]):
1     input  : S[] ← original training set
2     output: i ← the selected sample index
      #Calculate the total distance
3     total_dist = 0 ;
4     for i=1 to m do
5         total_dist = total_dist + DIST[i] ;
      end
      #Return a random position according to the cumulative probabilities
       vector
6     r = random(0, 1) * total_dist ;
7     for i=1 to m do
8         r = r − DIST[i] ;
9         if r < 0 then
10            return i ;
11
          end
12
      end
End Function
```

## 3.3. Synthetic Features Generation

Remember that in the formulation of an LDL problem we have, on the one hand, a vector of features that quantifies the sample and, on the other hand, a vector of labels that qualifies the sample. It is an important constraint to take into account when generating a synthetic sample as we will have to create both dimensions. The algorithm we have developed here is therefore divided as follows:

To generate the synthetic vector of characteristics we randomly choose one of the closest neighbors (the definition of close is made by taking into account the distance vector calculated in Section 3.1). From these two samples a new one is interpolated that will be located in an intermediate point between the two. The calculation of this intermediate point also introduces a randomness factor in order to carry out the most uniform possible sampling of the data space. Figure 4.2 could be used to help understand this process.

Regarding the label synthetic creation, the value assigned to each position of the label is the mean of the $k$ nearest neighbors.

The formal procedure works as follows. First, the $k$ nearest neighbors of the sample selected by the selection algorithm are calculated and their indexes are stored. The second step is to create the different elements of the feature array, and to do so we randomly choose one of the neighbors obtained in the previous step and calculate a random intermediate value between the sample and the selected neighbor. This process is repeated iteratively for each of the $q$ features of the sample. Finally, the vector of labels is generated by taking into account all the nearest $k$ neighbors and obtaining the value for the new sample as the arithmetic mean of each label vector. The whole process is summarized in Algorithm 3.

A simple example of SSG-LDL is illustrated in Figure 4.2. Here, an $S_i$ remote sample is selected as the basis for the creation of a new synthetic data point. Based on a distance metric and using a value of $k$=5, five nearest neighbors (samples $S_{i1}$ to $S_{i5}$) are chosen from the training set. Finally, the synthetic sample generation is carried out in order to obtain a new synthetic sample $SS$. Specifically in this example the value obtained for the first characteristic of the synthetic sample is equal to $0.4 + (0.15 - 0.4) \times random(0, 1)$. For a random value of 0.6 we get a result of 0.25. The rest of the positions of the characteristic vector are calculated in the same way. As for the label vector, the synthetic value obtained corresponds to the average of the values of the 5 neighbors. In our case, the value of the first label is equal to $(0.1 + 0.7 + 0.2 + 0.0 + 0.8)/5 = 0.36$. The other positions of the label vector are calculated in the same way.

**Algorithm 3:** Create a Synthetic Sample

**Function** `CreateSyntheticSample(`$S[]$, $i$, $k$`):`

**input :** $S[]$ ← original training set
$i$ ← selected sample index
$k$ ← nearest neighbors to be considered

**output:** $ss$ ← the synthetic sample created

#Step 1 - Compute $k$ nearest neighbors for $i$, and save the indexes in the $nnarray$

#Step 2 - Generate the synthetic features

$nn = \text{random}(1, k)$;

**for** $attr = 1$ *to* $q$ **do**

  Compute: dif $= S[nnarray[nn]].x_{attr} - S[i].x_{attr}$;

  Compute: gap $= \text{random}(0, 1)$;

  $ss.x_{attr} = S[i].x_{attr} + gap \times dif$;

**end**

#Step 3 - Generate the synthetic labels

**for** $lbl = 1$ *to* $c$ **do**

  **for** $nn = 1$ *to* $k$ **do**

    $ss.d_{lbl}$ += $S[nnarray[nn]].d_{lbl}$;

  **end**

  $ss.d_{lbl}$ /= $k$;

**end**

**return** $ss$;

**End Function**

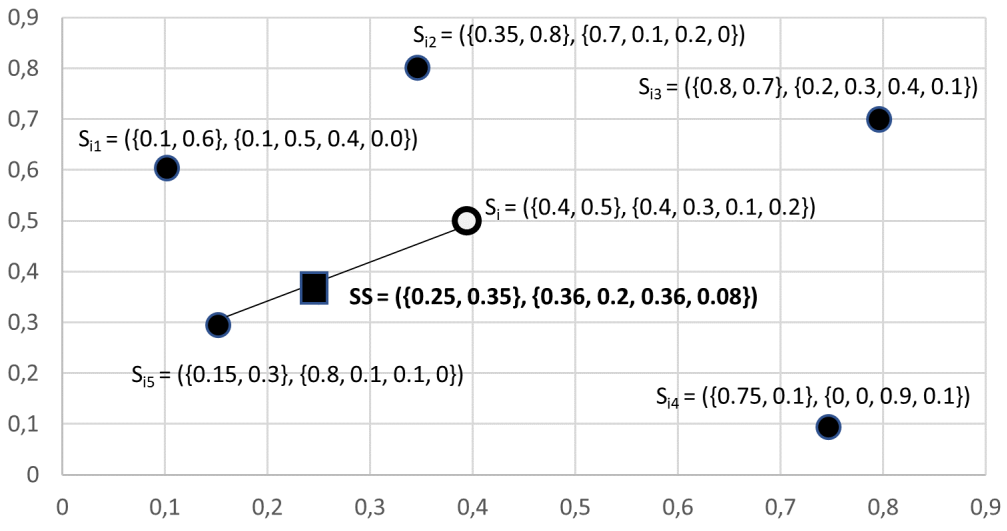



Figure 4.2: An illustration of how to create the synthetic data points in the SSG-LDL algorithm

# 4. Experimental Framework

This section is devoted to introducing the experimental framework used in the different empirical studies of the paper. In our experiments, we have included 15 datasets of a good variety of real-world problems. These are referred to in the following Section 4.1.

In order to evaluate the learners' proficiency, we have employed six measures (described in Section 4.2) for the different aspects of their performance.

For each dataset and learner, these measures were computed over a set merged from the test predictions of a 10-fold cross validation set (10-fcv). In the first pre-processing step we apply our synthetic sample generation algorithm described in Section 3 to each training set and then we use them to train the learners.

Finally, the Wilcoxon test and Bayesian Sign test [5, 15] are used to validate the results of the empirical comparisons. In the Bayesian Sign test, a distribution of the differences of the results achieved using methods $L$ (original learner without applying SSG-LDL) and $R$ (learner using the SSG-LDL pre-processing method) is computed into a graphical space divided into 3 regions: left, rope and right. The location of most of the distribution in these sectors indicates the final decision: the superiority of algorithm $L$, statistical equivalence and the superiority of algorithm $R$, respectively. The KEEL package [39] has been used to compute the Wilcoxon test and the R package rNPBST [8] was used to extract the graphical representations of the Bayesian Sign tests analyzed in the following empirical studies. The Rope limit parameter used to represent the Bayesian Sign test is 0.0001.

## 4.1. Datasets

There are 15 real-world datasets employed in the experiments in total. This is a compilation of the datasets most used in the referenced LDL articles [14, 23, 24, 36, 43, 50]. The statistics are shown in Table 4.1.

The first to the tenth datasets were collected from ten biological experiments on the budding yeast Saccharomyces cerevisiae [18]. Each dataset includes 2,465 yeast genes, and an associated phylogenetic profile vector with a length of 24 is utilized to represent each gene. In a biological experiment, the gene expression level is usually disparate at each discrete time point, so the labels correspond to the time point. Data from different time courses of gene expression in yeast S. cerevisiae were combined and clustered. The data were obtained from the time courses during the following processes: the cell division cycle after synchronization by alpha factor arrest (Yeast_alpha, 18 time points); centrifugal elutriation (Yeast_elu, 14 time points), and with a temperature-sensitive cdc15 mutant (Yeast_cdc, 15 time points); sporulation (Yeast_spo/spo5/spoem, 7 time points plus four additional samples); shock by high temperature (Yeast_heat, 6 time points); reducing agents (Yeast_dtt, 4 time points) and low temperature (Yeast_cold, 4 time points); and the diauxic shift (Yeast_diau, 7 time points). All data were gathered using DNA microarrays with elements that represented almost all the ORFs (Open Reading Frame) of the fully sequenced S. cerevisiae genome; all measurements were made against a reference sample of time 0 except for the cell-cycle experiments, in which an unsynchronized sample was used.

The contribution of each sample to the gene similarity score from a given process was weighted by the inverse of the square root of the number of samples analyzed from that process.

Datasets JAFFE [31] and BU_3DFE [47] are two widely used facial expression image datasets. In JAFFE, ten expressors posed 3 or 4 examples of each of the six basic facial expressions (fear, disgust, happiness, anger, sadness, surprise) and a neutral face for a total of 213 images of facial expressions. For the simplicity of the experimental design, only Japanese women participated. Each woman took pictures of herself while looking through a semi-reflective plastic sheet into the camera. Tungsten lights were placed to create a uniform illumination of the face. The images were then printed in monochrome and digitized using a flatbed scanner. Finally, images were labelled by 92 people with a 5 or 6 component vector with ratings averaged over all subjects. The similarities between these semantic vectors were computed using Euclidean distance. There were 100 subjects who participated in BU_3DFE face scans. The resulting database consists of 60% women and 40% men with a variety of ethnic/racial backgrounds. Each subject performed the same basic facial expressions as in JAFFE. With the exception of the neutral expression, each of the six prototypic expressions includes four levels of intensity. Therefore, there are 25 instant 3D expression models for each subject, resulting in a total of 2,500 3D facial expression models in the database. Each of these images were subsequently scored by 23 people using the same scale as used in JAFFE. Each dataset is represented by a 243-dimensional feature vector extracted using the Local Binary Patterns method (LBP) [3]. The score for each emotion is regarded as the description degree, and the description degrees (normalized gene expression level) of all the six emotions constitute a label distribution for a particular facial expression image.

Dataset Movie includes 7,755 movies. There is a total of 54,243,292 ratings from 478,656 different users on a scale from 1 to 5 integral stars from ®Netflix. The percentage of each rating level is regarded as the label distribution. There are numeric and categorical attributes in the dataset such as genre, director, country, year, budget and so on. After transforming the categorical attributes into binary vectors, the final feature vector of each movie is 1,869-dimensional.

The Natural Scene dataset [6] is collected from 2,000 natural scene images. Each image is divided into 49 blocks using a 7×7 grid, then the first and second moments (mean and variance) of each band are computed, corresponding to a low-resolution image and to computationally inexpensive texture features, respectively. The end result is a 49×2×3=294-dimension feature vector per image. The images were later labeled by three human observers with multiple labels selected from 9 possible values, i.e., sun, sky, water, cloud, mountain, snow, desert, building, and plant. Rankers were required to rank the relevant labels in descending order of relevance and, as expected, the rankings for the same image from different rankers are highly inconsistent. So, a nonlinear programming process [25] is applied to achieve the label distribution.

The Human Gene dataset is much larger than the other datasets used in this experiment. This dataset is collected from biological research on the relationship between human genes and diseases. Each of the 30,542 human genes is represented by the 36 numerical descriptors for a gene sequence proposed in [48]. This 36-D

vector was deduced using a modified method based on I-TN curve to display the specific features of protein-coding genes in Aeropyrum pernix K1 genome. There are 68 different disease labels in total, and the normalized gene expression level for each disease is considered to be the description degree of the corresponding disease label. The gene expression level of different diseases provides a measure of the description degree of the label for every human gene.

| No. | Datasets | Examples($n$) | Features($q$) | Labels($l$) |
|---|---|---|---|---|
| 1 | Yeast_alpha | 2465 | 24 | 18 |
| 2 | Yeast_cdc | 2465 | 24 | 15 |
| 3 | Yeast_cold | 2465 | 24 | 4 |
| 4 | Yeast_diau | 2465 | 24 | 7 |
| 5 | Yeast_dtt | 2465 | 24 | 4 |
| 6 | Yeast_elu | 2465 | 24 | 14 |
| 7 | Yeast_heat | 2465 | 24 | 6 |
| 8 | Yeast_spo | 2465 | 24 | 6 |
| 9 | Yeast_spo5 | 2465 | 24 | 3 |
| 10 | Yeast_spoem | 2465 | 24 | 2 |
| 11 | SJAFFE | 213 | 243 | 6 |
| 12 | SBU_3DFE | 2500 | 243 | 6 |
| 13 | Movie | 7755 | 1869 | 5 |
| 14 | Natural_Scene | 2000 | 294 | 9 |
| 15 | Human_Gene | 30542 | 36 | 68 |

Table 4.1: Datasets used in experiments

## 4.2. Evaluation Measure Selection

We propose using a set of six measures when comparing different LDL algorithms: Chebyshev Distance, Clark Distance, Canberra Metric, Kullback-Leibler Divergence, Cosine Coefficient and Intersection Similarity as shown in Table 4.2. Each of these measures come from a different syntactic family summarized in [9] and are relatively widely used in the related areas. Thus, they can represent the different aspects of the algorithms well. Another reason that has led us to choose these measures over others is that they are the most widely used in the referenced LDL studies [14, 23, 24, 36, 43, 45, 50], thus allowing us to make a more accurate comparison.

## 4.3. Parameters

Figure 4.3 shows the influence of the different values of the parameter $N$ on the results and the percentage of oversampling applied to the initial dataset. We can observe that when the oversampling percentage is over 300%, the value obtained by the learning algorithm gets stuck. This is the value selected for our synthetic sample generation algorithm in all experiments. Increasing this parameter to over 300% would negatively affect performance without achieving any significant improvements.

| Name | Formula |
|---|---|
| Chebyshev(Cheby)↓ | $Dis(D,\hat{D}) = max_j\|d_j - \hat{d}_j\|$ |
| Clark↓ | $Dis(D,\hat{D}) = \sqrt{\sum_{j=1}^{c} \frac{(d_j-\hat{d}_j)^2}{(d_j+\hat{d}_j)^2}}$ |
| Canberra(Can)↓ | $Dis(D,\hat{D}) = \sum_{j=1}^{c} \frac{\|d_j-\hat{d}_j\|^2}{d_j+\hat{d}_j}$ |
| Kullback-Leibler(KL)↓ | $Dis(D,\hat{D}) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$ |
| Cosine(Cos)↑ | $Sim(D,\hat{D}) = \frac{\sum_{j=1}^{c} d_j\hat{d}_j}{\sqrt{\sum_{j=1}^{c} d_j^2}\sqrt{\sum_{j=1}^{c} \hat{d}_j^2}}$ |
| Intersection(Inter)↑ | $Sim(D,\hat{D}) = \sum_{j=1}^{c} min(d_j,\hat{d}_j)$ |

Table 4.2: Evaluation measure for LDL learners. ↓ *means that the lowest value is the best and* ↑ *means the opposite.*

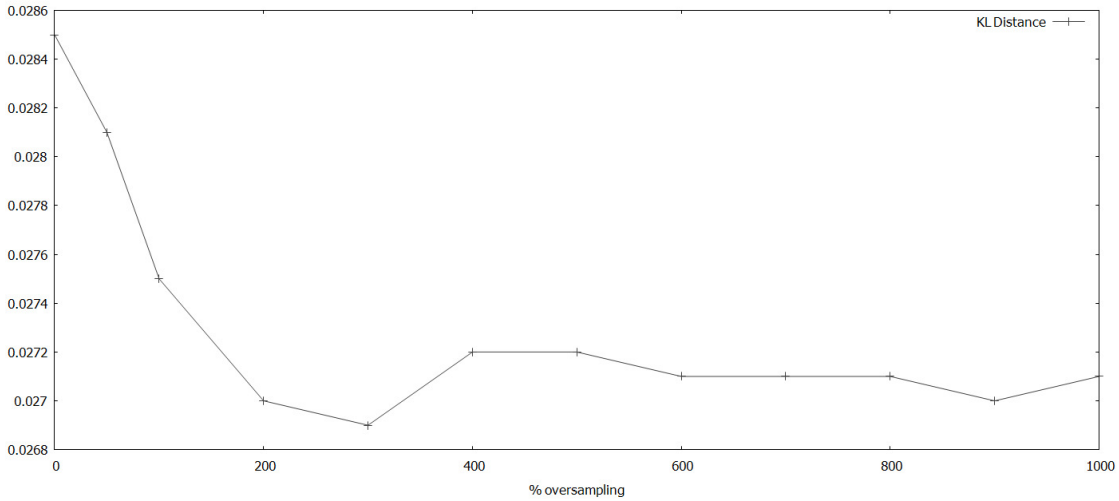



Figure 4.3: Influence of experiment parameters on the results using the yeast_spoem dataset

The figure above represents how the Kullback-Leibler divergence on the *yeast_spoem* dataset varies; these variations are similar for all other measures and datasets. Other parameter values in synthetic sample generation are: $k = 5$, where $k$ is the number of neighbors considered. With regard to the calculation of distances between points, we have taken 50% of the distance between features and 50% of the distance between labels into account.

There were 4 neighbors selected for the $k$-NN algorithm. For the BFGS algorithm, the convergence criterion $\epsilon$ has been setup to $10^{-5}$. Regarding the parameters used for the StructRF algorithm, we have respected the same values as those used in the original proposal papers: numbers of trees = 50, sampling ratio = 0.8, max. depth = 20, min. size of leaf = 5.

An overview of all these parameters can be found in Table 4.3.

The choice of parameters has been made according to the standards and recommendations given by the authors in the original proposal papers. Due to the fact that the experimental evaluation comprises a high number of datasets, having to

| Algorithm | Parameter | Description | Value |
|---|---|---|---|
| $k$-NN | $k$ | Number of selected neighbors | 4 |
| BFGS | $\epsilon$ | Convergence criterion: must be less than $\epsilon$ before successful termination | $10^{-5}$ |
| StructRF | trees | Number of trees | 50 |
| | sampling | Sampling ratio of data | 0.8 |
| | max. depth | Maximum depth of the tree | 20 |
| | min. leaf | Minimum size of the leaf | 5 |
| SSG-LDL | N | % of oversampling | 300 |
| | $k$ | Number of selected neighbors | 5 |
| | $f_x$ | Feature distance factor | 0.5 |
| | $f_y$ | Label distance factor | 0.5 |

Table 4.3: Summary of the parameters

adjust each parameter individually for each dataset would be unreasonable. In fact, our idea is just the opposite; we aim to compare the datasets in the most general scenario possible. We have not performed any tuning to adapt these parameters, because our objective is not to maximize the accuracy or any other performance measure, but to fairly compare the algorithms and their robustness in a common environment on different datasets.

# 5. Results and Analysis

This section presents the results of the empirical studies and their analyses. For each learner we will compare the version with and without pre-processing, showing their different strengths. As those algorithms have been tested using 10-fcv, the performance is represented using "mean±standard deviation". Each of the tables shows the experimental results obtained for each learner, comparing them with the results obtained from same learner but using SSG-LDL pre-processing. In each case the best result is highlighted in bold. The last row is the mean of each column. The best average is also highlighted in bold.

The results of the different measures are shown in Tables 4.4 to 4.9.

| Dataset | $k$-NN | SSG-LDL+$k$-NN | BFGS | SSG-LDL+BFGS | StructRF | SSG-LDL+StructRF |
|---|---|---|---|---|---|---|
| Yeast_alpha | 0.0148±0.0007 | **0.0144±0.0007** | **0.0135±0.0008** | **0.0135±0.0008** | **0.0134±0.0008** | **0.0134±0.0008** |
| Yeast_cdc | 0.0177±0.001 | **0.0172±0.001** | 0.0163±0.0009 | **0.0162±0.0009** | 0.0162±0.0009 | **0.0161±0.0009** |
| Yeast_cold | 0.0554±0.0021 | **0.0542±0.002** | 0.0512±0.0018 | **0.051±0.0019** | 0.0501±0.0016 | **0.05±0.0016** |
| Yeast_diau | 0.0393±0.0009 | **0.0381±0.0012** | **0.037±0.0014** | **0.037±0.0015** | **0.0359±0.0015** | 0.036±0.0016 |
| Yeast_dtt | 0.0393±0.0016 | **0.038 ±0.0016** | 0.0361±0.0013 | **0.0359±0.0013** | 0.0353±0.0013 | **0.035±0.0012** |
| Yeast_elu | 0.0177±0.0005 | **0.0172±0.0005** | **0.0163±0.0006** | **0.0163±0.0006** | **0.0161±0.0005** | **0.0161±0.0006** |
| Yeast_heat | 0.0451±0.0012 | **0.0438±0.001** | 0.0423±0.0008 | **0.0422±0.0008** | 0.0407±0.0009 | **0.0406±0.0009** |
| Yeast_spo | 0.0643±0.0024 | **0.0621±0.0024** | 0.0584±0.0037 | **0.0582±0.0036** | 0.0578±0.003 | **0.0574±0.0031** |
| Yeast_spo5 | 0.0962±0.0043 | **0.0949±0.0043** | 0.0914±0.005 | **0.0913±0.0051** | **0.0874±0.0046** | 0.0882±0.0044 |
| Yeast_spoem | 0.0924±0.0036 | **0.0905±0.0043** | **0.0868±0.0049** | 0.0869±0.0052 | **0.0836±0.0045** | 0.0841±0.0037 |
| SJAFFE | 0.1155±0.018 | **0.1153±0.0183** | 0.1603±0.016 | **0.1142±0.0147** | **0.1094±0.0108** | 0.1141±0.0136 |
| SBU_3DFE | 0.135±0.0048 | **0.1327±0.0052** | **0.11±0.0039** | 0.1231±0.0048 | **0.1183±0.0058** | 0.1225±0.0053 |
| Movie | **0.1284±0.0066** | 0.1317±0.0081 | 0.1398±0.0161 | **0.1276±0.0095** | **0.1104±0.0048** | 0.1195±0.0071 |
| Natural_Scene | **0.3168±0.0081** | 0.318±0.0084 | 0.3549±0.0159 | **0.3217±0.0142** | **0.2738±0.0116** | 0.2863±0.0117 |
| Human_Gene | 0.0652±0.0048 | **0.0618±0.0042** | **0.0533±0.0036** | **0.0533±0.0036** | 0.0553±0.0048 | **0.0538±0.004** |
| Average | 0.0829±0.0040 | **0.0820±0.0042** | 0.0845±0.0051 | **0.0792±0.0046** | **0.0736±0.0038** | 0.0755±0.0040 |

Table 4.4: Experimental Results (mean ± std) measured by Chebyshev Distance ↓

| Dataset | $k$-NN | SSG-LDL+$k$-NN | BFGS | SSG-LDL+BFGS | StructRF | SSG-LDL+StructRF |
|---|---|---|---|---|---|---|
| Yeast_alpha | 0.2321±0.0112 | **0.2256±0.0119** | 0.2107±0.0126 | **0.2103±0.0126** | 0.2095±0.0125 | **0.2094±0.0125** |
| Yeast_cdc | 0.2375±0.0134 | **0.2314±0.0139** | 0.2165±0.0129 | **0.2161±0.013** | 0.2158±0.0134 | **0.2152±0.0132** |
| Yeast_cold | 0.1509±0.007 | **0.1475±0.0066** | 0.1398±0.0055 | **0.1394±0.0058** | 0.1367±0.0053 | **0.1366±0.0054** |
| Yeast_diau | 0.2122±0.0042 | **0.2064±0.006** | 0.2008±0.0079 | **0.2007±0.0082** | **0.1947±0.0076** | 0.195±0.0078 |
| Yeast_dtt | 0.1068±0.0045 | **0.1035±0.0045** | 0.0984±0.0039 | **0.0979±0.0038** | 0.0961±0.0034 | **0.0956±0.0035** |
| Yeast_elu | 0.2182±0.0048 | **0.2118±0.0053** | 0.1992±0.0055 | **0.1988±0.0056** | 0.1974±0.0054 | **0.1969±0.0057** |
| Yeast_heat | 0.1955±0.0044 | **0.1901±0.0034** | 0.1828±0.0027 | **0.1824±0.0028** | 0.177±0.0028 | **0.1766±0.0028** |
| Yeast_spo | 0.2715±0.0112 | **0.2633±0.0112** | 0.25±0.0164 | **0.2494±0.0159** | 0.2467±0.0139 | **0.2458±0.014** |
| Yeast_spo5 | 0.1933±0.0099 | **0.1906±0.0099** | 0.1842±0.0112 | **0.1839±0.0113** | **0.1767±0.0104** | 0.178±0.0096 |
| Yeast_spoem | 0.1374±0.0053 | **0.1345±0.0063** | **0.1293±0.0072** | **0.1293±0.0077** | **0.1247±0.0066** | 0.1254±0.0054 |
| SJAFFE | 0.4137±0.0489 | **0.4119±0.0502** | 0.6466±0.0462 | **0.4254±0.0427** | **0.39±0.035** | 0.405±0.0421 |
| SBU_3DFE | 0.4255±0.0134 | **0.4166±0.0139** | **0.3784±0.0122** | 0.3852±0.0144 | **0.368±0.0173** | 0.3786±0.0167 |
| Movie | **0.5686±0.0393** | 0.5849±0.0466 | 0.6035±0.056 | **0.5675±0.0405** | **0.5039±0.0233** | 0.5467±0.0371 |
| Natural_Scene | **1.8253±0.0343** | 1.9844±0.0382 | **2.3817±0.0239** | 2.4503±0.0203 | **2.3946±0.0227** | 2.4306±0.0239 |
| Human_Gene | 2.3872±0.1012 | **2.3171±0.0845** | 2.1111±0.082 | **2.1109±0.0794** | 2.1776±0.1232 | **2.1291±0.0905** |
| Average | **0.505±0.0209** | 0.508±0.0208 | 0.5289±0.0204 | **0.5165±0.0189** | **0.5073±0.0202** | 0.511±0.0193 |

Table 4.5: Experimental Results (mean ± std) measured by Clark Distance ↓

| Dataset | $k$-NN | SSG-LDL+$k$-NN | BFGS | SSG-LDL+BFGS | StructRF | SSG-LDL+StructRF |
|---|---|---|---|---|---|---|
| Yeast_alpha | 0.7577±0.0372 | **0.7343±0.0404** | 0.6847±0.0432 | **0.6831±0.043** | 0.6812±0.0434 | **0.6801±0.0428** |
| Yeast_cdc | 0.7179±0.0395 | **0.6981±0.04** | 0.6493±0.0394 | **0.6479±0.0401** | 0.6472±0.041 | **0.6448±0.0401** |
| Yeast_cold | 0.2611±0.0129 | **0.2552±0.0118** | 0.2407±0.0097 | **0.2402±0.0099** | 0.2359±0.0092 | **0.2356±0.0097** |
| Yeast_diau | 0.456±0.0109 | **0.4443±0.014** | **0.431±0.0186** | 0.4313±0.0195 | **0.4177±0.0178** | 0.4184±0.0187 |
| Yeast_dtt | 0.1836±0.0075 | **0.1779±0.007** | 0.1693±0.0061 | **0.1685±0.0058** | 0.1649±0.005 | **0.1643±0.0052** |
| Yeast_elu | 0.6444±0.0155 | **0.6242±0.0164** | 0.5838±0.0159 | **0.5824±0.0162** | 0.578±0.0156 | **0.5769±0.0159** |
| Yeast_heat | 0.3924±0.0096 | **0.3814±0.0076** | 0.3647±0.0058 | **0.3637±0.0059** | 0.3541±0.0063 | **0.3528±0.006** |
| Yeast_spo | 0.5594±0.0232 | **0.543±0.0234** | 0.5137±0.0335 | **0.513±0.0321** | 0.5066±0.0274 | **0.5046±0.0275** |
| Yeast_spo5 | 0.2972±0.0145 | **0.2934±0.0143** | 0.2829±0.0163 | **0.2826±0.0165** | **0.2713±0.0151** | 0.2735±0.0143 |
| Yeast_spoem | 0.1913±0.0074 | **0.1872±0.0088** | **0.1798±0.01** | 0.1799±0.0107 | **0.1733±0.0092** | 0.1744±0.0076 |
| SJAFFE | 0.8527±0.0962 | **0.8471±0.0998** | 1.3595±0.11 | **0.8783±0.0992** | **0.8089±0.0769** | 0.84±0.0938 |
| SBU_3DFE | 0.8746±0.029 | **0.8611±0.0302** | **0.7831±0.0246** | 0.8265±0.0338 | **0.7817±0.0354** | 0.8087±0.0361 |
| Movie | **1.0946±0.0766** | 1.1255±0.0927 | 1.1679±0.1163 | **1.0905±0.0833** | **0.9595±0.0441** | 1.0412±0.0737 |
| Natural_Scene | **4.2609±0.0866** | 4.8189±0.1051 | **6.5546±0.0914** | 6.7306±0.0831 | **6.4559±0.095** | 6.6115±0.1026 |
| Human_Gene | 16.2737±0.7555 | **15.7894±0.6271** | 14.4531±0.6124 | **14.4487±0.5935** | 14.8633±0.8857 | **14.5496±0.6531** |
| Average | 1.8545±0.0815 | **1.8521±0.0759** | 1.8945±0.0769 | **1.8711±0.0728** | 1.86±0.0885 | **1.8584±0.0765** |

Table 4.6: Experimental Results (mean ± std) measured by Canberra Metric ↓

| Dataset | $k$-NN | SSG-LDL+$k$-NN | BFGS | SSG-LDL+BFGS | StructRF | SSG-LDL+StructRF |
|---|---|---|---|---|---|---|
| Yeast_alpha | 0.0066±0.0006 | **0.0063±0.0006** | **0.0055±0.0006** | **0.0055±0.0006** | **0.0055±0.0006** | **0.0055±0.0006** |
| Yeast_cdc | 0.0083±0.0009 | **0.0079±0.0009** | **0.0070±0.0008** | **0.0070±0.0008** | 0.0070±0.0009 | **0.0069±0.0008** |
| Yeast_cold | 0.0142±0.0014 | **0.0136±0.0014** | **0.0122±0.0011** | **0.0122±0.0012** | **0.0118±0.0011** | **0.0118±0.0011** |
| Yeast_diau | 0.0150±0.0008 | **0.0142±0.0009** | **0.0131±0.0011** | **0.0131±0.0011** | **0.0125±0.001** | **0.0125±0.0011** |
| Yeast_dtt | 0.0073±0.0007 | **0.0069±0.0007** | 0.0063±0.0006 | **0.0062±0.0006** | 0.0061±0.0006 | **0.0060±0.0006** |
| Yeast_elu | 0.0074±0.0003 | **0.007±0.0004** | **0.0062±0.0004** | **0.0062±0.0004** | **0.0061±0.0004** | **0.0061±0.0004** |
| Yeast_heat | 0.0146±0.0007 | **0.0138±0.0005** | **0.0126±0.0004** | **0.0126±0.0004** | 0.0120±0.0004 | **0.0119±0.0004** |
| Yeast_spo | 0.0302±0.0024 | **0.0283±0.0024** | 0.0246±0.0029 | **0.0245±0.0028** | 0.0243±0.0024 | **0.0241±0.0025** |
| Yeast_spo5 | 0.0333±0.0033 | **0.0322±0.0036** | **0.0293±0.0028** | **0.0293±0.0028** | **0.027±0.0025** | 0.0276±0.0022 |
| Yeast_spoem | 0.0285±0.0023 | **0.0272±0.0026** | **0.0245±0.0023** | 0.0246±0.0024 | **0.0228±0.002** | 0.0233±0.0018 |
| SJAFFE | 0.0730±0.0162 | **0.0724±0.0165** | 0.1639±0.0245 | **0.0754±0.0166** | **0.0603±0.0089** | 0.0655±0.0116 |
| SBU_3DFE | 0.0907±0.0048 | **0.0872±0.0048** | **0.0634±0.0038** | 0.0689±0.0038 | **0.0659±0.0054** | 0.0703±0.0046 |
| Movie | **0.1255±0.0137** | 0.1313±0.0167 | 0.1543±0.0405 | **0.1211±0.0177** | **0.0901±0.0061** | 0.1046±0.0114 |
| Natural_Scene | 1.1924±0.0765 | **1.1011±0.063** | 1.112±0.0999 | **0.8061±0.0292** | **0.6436±0.0211** | 0.696±0.0259 |
| Human_Gene | 0.3006±0.0248 | **0.2815±0.0206** | 0.2365±0.0184 | **0.236±0.018** | 0.248±0.0265 | **0.239±0.0197** |
| Average | 0.1298±0.01 | **0.1221±0.009** | 0.1248±0.0133 | **0.0966±0.0066** | **0.0829±0.0053** | 0.0874±0.0056 |

Table 4.7: Experimental Results (mean ± std) measured by Kullback-Leibler Divergence ↓

| Dataset | $k$-NN | SSG-LDL+$k$-NN | BFGS | SSG-LDL+BFGS | StructRF | SSG-LDL+StructRF |
|---|---|---|---|---|---|---|
| Yeast_alpha | 0.9935±0.0006 | **0.9938±0.0006** | **0.9946±0.0006** | **0.9946±0.0006** | **0.9946±0.0006** | **0.9946±0.0006** |
| Yeast_cdc | 0.992±0.0008 | **0.9924±0.0008** | **0.9933±0.0007** | **0.9933±0.0007** | **0.9933±0.0008** | **0.9933±0.0007** |
| Yeast_cold | 0.9866±0.0012 | **0.9872±0.0011** | 0.9885±0.0009 | **0.9886±0.0010** | **0.9889±0.0009** | **0.9889±0.0009** |
| Yeast_diau | 0.9862±0.0008 | **0.987±0.0008** | **0.9879±0.0010** | **0.9879±0.0010** | **0.9885±0.0010** | **0.9885±0.0011** |
| Yeast_dtt | 0.9931±0.0005 | **0.9934±0.0005** | **0.9941±0.0004** | **0.9941±0.0004** | **0.9943±0.0004** | **0.9943±0.0004** |
| Yeast_elu | 0.9928±0.0003 | **0.9932±0.0003** | 0.994±0.0004 | **0.9941±0.0004** | **0.9941±0.0003** | **0.9941±0.0004** |
| Yeast_heat | 0.9861±0.0007 | **0.9869±0.0006** | **0.9880±0.0004** | **0.9880±0.0004** | 0.9886±0.0004 | **0.9887±0.0005** |
| Yeast_spo | 0.9716±0.002 | **0.9734±0.002** | 0.9769±0.0026 | **0.977±0.0025** | 0.9773±0.0021 | **0.9775±0.0021** |
| Yeast_spo5 | 0.9705±0.0025 | **0.9714±0.0028** | **0.9741±0.0023** | **0.9741±0.0023** | **0.9762±0.002** | 0.9756±0.0018 |
| Yeast_spoem | 0.9754±0.0019 | **0.9764±0.0023** | **0.979±0.0019** | 0.9789±0.002 | **0.9804±0.0018** | 0.9799±0.0016 |
| SJAFFE | 0.9308±0.0163 | **0.9313±0.0166** | 0.8739±0.0179 | **0.9293±0.0154** | **0.9429±0.0083** | 0.9381±0.0107 |
| SBU_3DFE | 0.9118±0.0047 | **0.915±0.0047** | **0.9389±0.0034** | 0.9324±0.0037 | **0.9348±0.0055** | 0.9309±0.0044 |
| Movie | **0.9174±0.0074** | 0.9135±0.0093 | 0.9072±0.0188 | **0.9209±0.0103** | **0.9407±0.0039** | 0.9319±0.0066 |
| Natural_Scene | **0.7043±0.0137** | 0.704±0.0108 | 0.6724±0.0149 | **0.719±0.0104** | **0.7895±0.0091** | 0.767±0.0129 |
| Human_Gene | 0.7663±0.0184 | **0.7847±0.0136** | 0.8343±0.0102 | **0.8347±0.0099** | 0.8202±0.0205 | **0.8306±0.0126** |
| Average | 0.9386±0.0048 | **0.9402±0.0045** | 0.9398±0.0051 | **0.9471±0.0041** | **0.9536±0.0038** | 0.9516±0.0038 |

Table 4.8: Experimental Results (mean ± std) measured by Cosine Coefficient ↑

| Dataset | $k$-NN | SSG-LDL+$k$-NN | BFGS | SSG-LDL+BFGS | StructRF | SSG-LDL+StructRF |
|---|---|---|---|---|---|---|
| Yeast_alpha | 0.9581±0.0020 | **0.9594±0.0022** | 0.9622±0.0024 | **0.9623±0.0024** | **0.9624±0.0024** | **0.9624±0.0023** |
| Yeast_cdc | 0.9527±0.0025 | **0.954±0.0026** | 0.9573±0.0026 | **0.9574±0.0026** | 0.9574±0.0027 | **0.9576±0.0026** |
| Yeast_cold | 0.9356±0.0031 | **0.937±0.0028** | 0.9407±0.0023 | **0.9408±0.0023** | **0.9419±0.0022** | **0.9419±0.0023** |
| Yeast_diau | 0.9367±0.0017 | **0.9383±0.002** | **0.9403±0.0027** | 0.9402±0.0028 | **0.9421±0.0026** | 0.942±0.0027 |
| Yeast_dtt | 0.9547±0.0017 | **0.9561±0.0016** | 0.9582±0.0013 | **0.9584±0.0013** | 0.9593±0.0011 | **0.9595±0.0011** |
| Yeast_elu | 0.9545±0.0011 | **0.956±0.0012** | 0.9588±0.0011 | **0.9589±0.0012** | 0.9592±0.0011 | **0.9593±0.0011** |
| Yeast_heat | 0.9356±0.0017 | **0.9374±0.0014** | 0.9401±0.001 | **0.9403±0.001** | 0.9419±0.0011 | **0.9421±0.001** |
| Yeast_spo | 0.9076±0.0036 | **0.9104±0.0037** | 0.9154±0.0053 | **0.9156±0.0051** | 0.9167±0.0042 | **0.917±0.0043** |
| Yeast_spo5 | 0.9038±0.0043 | **0.9051±0.0043** | 0.9086±0.005 | **0.9087±0.0051** | **0.9126±0.0046** | 0.9118±0.0044 |
| Yeast_spoem | 0.9076±0.0036 | **0.9095±0.0043** | **0.9132±0.0049** | 0.9131±0.0052 | **0.9164±0.0045** | 0.9159±0.0037 |
| SJAFFE | 0.8529±0.0176 | **0.8539±0.0182** | 0.7788±0.0162 | **0.8496±0.0185** | **0.8622±0.0132** | 0.8568±0.0164 |
| SBU_3DFE | 0.8394±0.0053 | **0.8425±0.0053** | **0.8616±0.0041** | 0.8521±0.0056 | **0.8592±0.0065** | 0.8546±0.0063 |
| Movie | **0.8153±0.0109** | 0.8095±0.014 | 0.804±0.0199 | **0.8171±0.0137** | **0.8432±0.006** | 0.8285±0.0107 |
| Natural_Scene | **0.5634±0.0098** | 0.5588±0.0079 | **0.5248±0.0145** | 0.5226±0.0082 | **0.5919±0.0097** | 0.5633±0.0098 |
| Human_Gene | 0.7417±0.0133 | **0.7525±0.0104** | 0.7842±0.0092 | **0.7845±0.0089** | 0.7739±0.0159 | **0.7814±0.0105** |
| Average | 0.8773±0.0055 | **0.8787±0.0055** | 0.8765±0.0062 | **0.8814±0.0056** | **0.8894±0.0052** | 0.8863±0.0053 |

Table 4.9: Experimental Results (mean ± std) measured by Intersection Similarity ↑

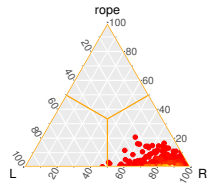
## 5.1. Evaluation of SSG-LDL+*k*-NN vs. *k*-NN

A comparison between $k$-NN and $k$-NN using a previously pre-processed dataset using SSG-LDL highlights the best ranking of the learner when using a preprocessed dataset.

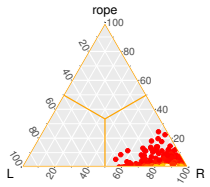
As previously mentioned, we have used the Wilcoxon Signed Ranks test and the Bayesian Sign test to corroborate the significance of the differences between our approach and the selected methods. Table 4.10 includes the outcome of the Wilcoxon test comparing both learners. All the hypotheses of equivalence are rejected with small p-values. Figure 4.4 graphically represents the difference between using data pre-processing or not and its statistical significance in terms of accuracy. The following heat-maps clearly indicate the significant superiority of SSG-LDL, as the computed distributions are always located in the right region.

| Measure | $R^+$ | $R^-$ | $p$-value |
|---|---|---|---|
| Cheby | 101 | 19 | 0.018066 |
| Clark | 92 | 28 | 0.073 |
| Can | 92 | 28 | 0.073 |
| KL | 107 | 13 | 0.005372 |
| Cos | 103 | 17 | 0.012452 |
| Inter | 93 | 27 | 0.06372 |

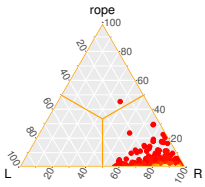
Table 4.10: Wilcoxon Signed Ranks test: SyntheticLDL+$k$-NN vs. $k$-NN



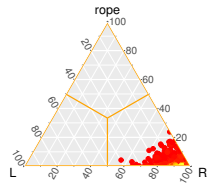

(a) Chebyshev Distance



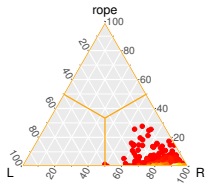

(b) Clark Distance



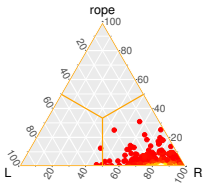

(c) Canberra Metric




(d) Kullback-Leibler Divergence




(e) Cosine Coefficient




(f) Intersection Similarity

Figure 4.4: Bayesian Sign test comparing k-NN (L) vs. SSG-LDL+k-NN (R)

## 5.2. Evaluation of SSG-LDL+BFGS vs. BFGS

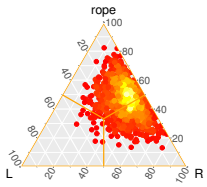
A comparison between BFGS and BFGS using a previously pre-processed dataset with synthetic feature generation also highlights the best ranking (for all distance

measures) of the learner when using a preprocessed dataset.

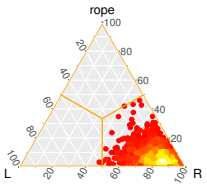
Table 4.11 includes the outcome of the Wilcoxon test comparing both learners. All the hypotheses of equivalence are rejected with small p-values. Figure 4.5 graphically represents the difference between using data pre-processing or not and its statistical significance in terms of accuracy. The heat-maps clearly indicate significant superiority, for almost all measures, when applying SSG-LDL, since the computed distributions are always located in the right region. In view of the results of this statistical analysis, there is a high probability that the results obtained will improve after applying this pre-processing method.

| Measure | $R^+$ | $R^-$ | $p$-value |
|---|---|---|---|
| Cheby | 94 | 26 | 0.05536 |
| Clark | 81 | 24 | 0.0785 |
| Can | 90 | 30 | 0.0946 |
| KL | 76 | 29 | 0.15308 |
| Cos | 89 | 31 | 0.107 |
| Inter | 87 | 33 | 0.13538 |

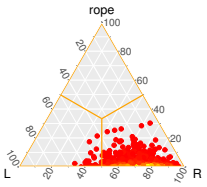
Table 4.11: Wilcoxon Signed Ranks test: SyntheticLDL+BFGS vs. BFGS



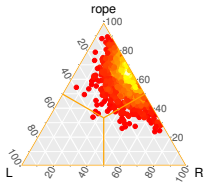

(a) Chebyshev Distance



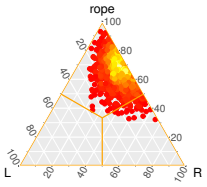

(b) Clark Distance



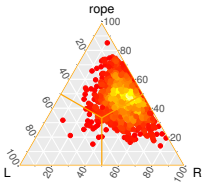

(c) Canberra Metric




(d) Kullback-Leibler Divergence




(e) Cosine Coefficient




(f) Intersection Similarity

Figure 4.5: Bayesian Sign test comparing BFGS (L) vs. SSG-LDL+BFGS (R)

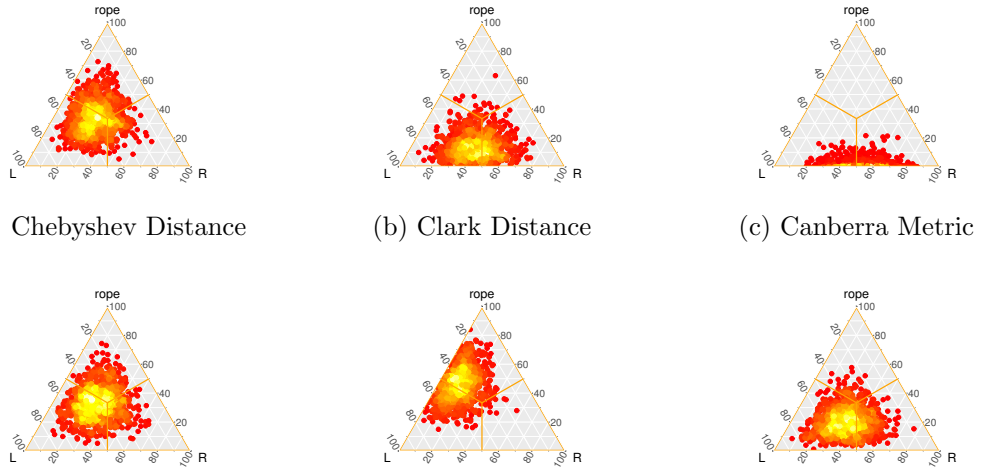
## 5.3. Evaluation of SSG-LDL+StructRF vs. StructRF

Let's see how the SSG-LDL algorithm behaves when it is combined with a learning method as robust as StructRF. As for the general classification, we see that applying pre-processing improves the results in 40% of the cases and ties in 15% of the cases. If we go into further detail we observe that the behavior varies according to the dataset: for the *Human_Gene* for example we notice that applying synthetic

sample generation does significantly improve the original results although it does not substantially alter the results for the *Yeast* datasets and causes slightly worse outputs for the sets related to facial expression images, movies or natural scenes. Table 4.12 includes the outcome of the Wilcoxon test comparing both learners. Figure 4.6 also corroborates these results statistically speaking, showing that applying pre-processing on the StructRF algorithm can be advantageous in a high percentage of cases.

| Measure | $R^+$ | $R^-$ | $p$-value |
|---|---|---|---|
| Cheby | 42 | 78 | $\geq 0.2$ |
| Clark | 49 | 71 | $\geq 0.2$ |
| Can | 53 | 67 | $\geq 0.2$ |
| KL | 44 | 76 | $\geq 0.2$ |
| Cos | 40 | 81 | $\geq 0.2$ |
| Inter | 44 | 77 | $\geq 0.2$ |

Table 4.12: Wilcoxon Signed Ranks test: SyntheticLDL+StructRF vs. StructRF




(a) Chebyshev Distance (b) Clark Distance (c) Canberra Metric

(d) Kullback-Leibler Divergence (e) Cosine Coefficient (f) Intersection Similarity

Figure 4.6: Bayesian Sign test comparing StructRF (L) vs. SSG-LDL+StructRF (R)

In summary, by applying a data pre-processing method such as the one presented here, SSG-LDL, significantly improves the results obtained by the learning algorithm.

In the case of $k$-NN, previously applying an SSG-LDL treatment makes the learner behave more efficiently in practically all datasets. This also happens when the same type of pre-processing is carried out on a BFGS learning method, obtaining notable improvement in the results. Even when using a method as robust as StructRF, it is also advantageous to apply the SSG-LDL as it can improve results in a high percentage of experiments.

# 6. Conclusion

LDL datasets suffer high dispersion that can negatively affect learners' performance.

In this paper, we proposed a novel method of synthetic data generation that adapts to LDL constraints. The SSG-LDL proposal can be applied, in a previous pre-processing stage, to any training set and then can be plugged in as an input to any learning algorithm.

We have based the design of this algorithm on techniques that have already demonstrated their potential in the area of data pre-processing. We have then extended and adapted them to be compatible with a distribution of labels as the output.

In order to verify the effectiveness of the solution designed, it has been applied to three state of the art learners in the LDL scope and we have verified that the results obtained are very encouraging.

The SSG-LDL method is the first proposal of oversampling adapted to LDL restrictions. In future research we want to make improvements on the current proposal to make it even stronger. Some ideas that we can anticipate are as follows:

- To select the instances to be oversampled, SSG-LDL uses the Euclidean distance. Some recent methods of oversampling such as MDO [2] build synthetic examples that have the same Mahalanobis distance from each examined class mean as the other minority examples. Thus, an alternative procedure that we would consider integrating in future studies is to improve learning in the region of minority instances by preserving the covariance and the probability during the generation of synthetic examples.

  The Hellinger distance metric, based on probability distributions, is more tolerant to skewed class distributions. This metric has been applied in the context of data reduction [46] and it also be of interest to introduce it in an oversampling method such as SSG-LDL.

- ProLSFEO-LDL [28] is a novel method that simultaneously addresses the prototype selection and the label-specific feature selection pre-processing techniques, specifically developed to deal with LDL restrictions. Devising a method that intelligently combines SSG-LDL as an oversampling method and ProLSFEO-LDL as a data reduction strategy could yield a very effective pre-processing method.

- Combining SMOTE and a boosting procedure such as SMOTEBoost [13] or MDOBoost [1] usually improves the prediction performance. Extending either of these two methods to use SSG-LDL and making them compatible with LDL problems is an interesting approach to follow.

- The filtering of artificial samples is a frequent operation that supports the success of SMOTE on real data. Adding a rebalancing stage to SSG-LDL similar to AMSCO [30] could mitigate the potential drawback of generating overlapping and noisy examples.

# Acknowledgements

This work is supported by the Spanish National Research Project TIN2017-89517-P.


# Compliance with ethical standards

## Conflict of interest

The authors declare that they have no conflict of interest.

# Bibliography


[1] L. Abdi and S. Hashemi, "To combat multi-class imbalanced problems by means of over-sampling and boosting techniques", *Soft Computing*, vol. 19, no. 12, pp. 3369–3385, 2015.

[2] L. Abdi and S. Hashemi, "To Combat Multi-Class Imbalanced Problems by Means of Over-Sampling Techniques", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 238–251, 2016.

[3] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 12, pp. 2037–2041, 2006.

[4] R. Barandela, J.S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems", *Pattern Recognition*, vol. 36, no. 3, pp. 849–851, 2003.

[5] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[6] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown, "Learning multi-label scene classification", *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[7] H. Cao, X.L. Li, D.Y. Woon, and S.K. Ng, "Integrated oversampling for imbalanced time series classification", *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2809–2822, 2013.

[8] J. Carrasco, S. García, M. del Mar Rueda, and F. Herrera, "rnpbst: An R package covering non-parametric and bayesian statistical tests", in *International Conference on Hybrid Artificial Intelligence Systems*, Springer, 2017, pp. 281–292.

[9] S.H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions", *City*, vol. 1, no. 2, p. 1, 2007.

[10] F. Charte, A.J. Rivera, M.J. del Jesus, and F. Herrera, "MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation", *Knowledge-Based Systems*, vol. 89, pp. 385–397, 2015.

[11] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique", *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[12] N.V. Chawla, D.A. Cieslak, L.O. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost", *Data Mining and Knowledge Discovery*, vol. 17, no. 2, pp. 225–252, 2008.

[13] N.V. Chawla, A. Lazarevic, L.O. Hall, and K.W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting", in *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2003, pp. 107–119.

[14] M. Chen, X. Wang, B. Feng, and W. Liu, "Structured random forest for label distribution learning", *Neurocomputing*, vol. 320, pp. 171–182, 2018.

[15] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[16] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data", *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2283–2301, 2013.

[17] A. Dong, F.L. Chung, and S. Wang, "Semi-supervised classification method through oversampling and common hidden space", *Information Sciences*, vol. 349, pp. 216–228, 2016.

[18] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns", *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[19] A. Fernández, S. García, F. Herrera, and N.V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary", *Journal of Artificial Intelligence Research*, vol. 61, pp. 863–905, 2018.

[20] B. Gao, C. Xing, C. Xie, J. Wu, and X. Geng, "Deep Label Distribution Learning with Label Ambiguity", *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2825–2838, 2017.

[21] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, English. 2015, vol. 72.

[22] S. García, J. Luengo, and F. Herrera, "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining", *Knowledge-Based Systems*, vol. 98, pp. 1–29, 2016.

[23] X. Geng, "Label Distribution Learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1734–1748, 2016.

[24] X. Geng and P. Hou, "Pre-release prediction of crowd opinion on movies by label distribution learning", in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 3511–3517.

[25] X. Geng and L. Luo, "Multilabel ranking with inconsistent rankers", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3742–3747.

[26] X. Geng, C. Yin, and Z. Zhou, "Facial age estimation by learning from label distributions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2401–2412, 2013.

[27] D.E. Goldberg, "Genetic algorithms in search", *Optimization, and Machine-Learning*, 1989.

[28] M. González, J.R. Cano, and S. García, "ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning", *Applied Sciences*, vol. 10, no. 9, Art. 3089, 2020.

[29] F. Herrera, F. Charte, A.J. Rivera, and M.J. Del Jesus, *Multilabel classification: Problem analysis, metrics and techniques*. Springer, 2016.

[30] J. Li, S. Fong, R.K. Wong, and V.W. Chu, "Adaptive multi-objective swarm fusion for imbalanced data classification", *Information Fusion*, vol. 39, pp. 1–24, 2018.

[31] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets", in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, IEEE, 1998, pp. 200–205.

[32] P. Moutafis and I.A. Kakadiaris, "GS4: Generating Synthetic Samples for Semi-Supervised Nearest Neighbor Classification", in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2014, pp. 393–403.

[33] R.C. Prati, G.E.A.P.A. Batista, and D.F. Silva, "Class imbalance revisited: a new experimental setup to assess the performance of treatment methods", *Knowledge and Information Systems*, vol. 45, no. 1, pp. 247–270, 2015.

[34] Y. Ren and X. Geng, "Sense Beauty by Label Distribution Learning.", in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 2648–2654.

[35] A. Roy, R.M.O. Cruz, R. Sabourin, and G.D.C. Cavalcanti, "A study on combining dynamic selection and data preprocessing for imbalance learning", *Neurocomputing*, vol. 286, pp. 179–192, 2018.

[36] W. Shen, K. Zhao, Y. Guo, and A.L. Yuille, "Label distribution learning forests", in *Advances in Neural Information Processing Systems*, 2017, pp. 834–843.

[37] L. Torgo, P. Branco, R.P. Ribeiro, and B. Pfahringer, "Resampling strategies for regression", *Expert Systems*, vol. 32, no. 3, pp. 465–476, 2015.

[38] I. Triguero, S. García, and F. Herrera, "SEG-SSC: A framework based on synthetic examples generation for self-labeled semi-supervised classification", *IEEE Transactions on cybernetics*, vol. 45, no. 4, pp. 622–634, 2015.

[39] I. Triguero, S. González, J.M. Moyano, S.r García, J. Alcalá-Fernández, J. Luengo, A. Fernández, M. José del Jesús, L. Sánchez, and F. Herrera, "KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining", *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 1238–1249, 2017.

[40] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview", *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[41] J. Wang, B. Yun, P. Huang, and Y.A. Liu, "Applying threshold SMOTE algorithm with attribute bagging to imbalanced datasets", in *International Conference on Rough Sets and Knowledge Technology*, Springer, 2013, pp. 221–228.

[42] Y. Wang and J. Dai, "Label Distribution Feature Selection Based on Mutual Information in Fuzzy Rough Set Theory", in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–2.

[43] C. Xing, X. Geng, and H. Xue, "Logistic boosting regression for label distribution learning", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4489–4497.

[44] D. Xue, Z. Hong, S. Guo, L. Gao, L. Wu, J. Zheng, and N. Zhao, "Personality recognition on social media with label distribution learning", *IEEE Access*, vol. 5, pp. 13 478–13 488, 2017.

[45] J. Yang, D. She, and M. Sun, "Joint Image Emotion Classification and Distribution Learning via Deep Convolutional Neural Network.", in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 3266–3272.

[46] L. Yin, Y. Ge, K. Xiao, X. Wang, and X. Quan, "Feature selection for high-dimensional imbalanced data", *Neurocomputing*, vol. 105, pp. 3–11, 2013.

[47] L. Yin, X. Wei, Y. Sun, J. Wang, and M.J. Rosato, "A 3D facial expression database for facial behavior research", in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, IEEE, 2006, pp. 211–216.

[48] J.F. Yu, D.K. Jiang, K. Xiao, Y. Jin, J.H. Wang, and X. Sun, "Discriminate the falsely predicted protein-coding genes in Aeropyrum Pernix K1 genome based on graphical representation", *Match-Communications in Mathematical and Computer Chemistry*, vol. 67, no. 3, p. 845, 2012.

[49] Z. Zhang, M. Wang, and X. Geng, "Crowd counting in public video surveillance by label distribution learning", *Neurocomputing*, vol. 166, pp. 151–163, 2015.

[50] X. Zheng, X. Jia, and W. Li, "Label Distribution Learning by Exploiting Sample Correlations Locally", in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 4556–4563.

# Capítulo 5

# Selección de Prototipos y de Características Específicas de las Etiquetas mediante Optimización Evolutiva para el Aprendizaje de la Distribución de Etiquetas

González, M., Cano, J. R., & García, S. (2020). ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning. Applied Sciences, 10(9), Art. 3089.

| | **JCR Clarivate**<br>IF: 2.474 | **CiteScore Scopus**<br>2.4 |
|---|---|---|
| **Categoría** | Engineering, Multidisciplinary | Computer Science Applications |
| **Ranking** | 32/91 | 299/636 |
| **Cuartil** | Q2 | Q2 |

Factores de Impacto 2019

# ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning

**Manuel González**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
manuel.gonzalez.es@gmail.com

**José-Ramón Cano**
Department of Computer Science
University of Jaén, 23700 Linares, Jaén, Spain
jrcano@ujaen.es

**Salvador García**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
salvagl@decsai.ugr.es

**Abstract**

Label Distribution Learning (LDL) is a general learning framework that assigns an instance to a distribution over a set of labels rather than to a single label or multiple labels. Current LDL methods have proven their effectiveness in many real-life machine learning applications. In LDL problems, instance-based algorithms and particularly the adapted version of the $k$-nearest neighbors method for LDL (AA-$k$NN) has proven to be very competitive, achieving acceptable results and allowing an explainable model. However, it suffers from several handicaps: it needs large storage requirements, it is not efficient predicting and presents a low tolerance to noise. The purpose of this paper is to mitigate these effects by adding a data reduction stage. The technique devised, called Prototype selection and Label-Specific Feature Evolutionary Optimization for LDL (ProLSFEO-LDL), is a novel method to simultaneously address the prototype selection and the label-specific feature selection pre-processing techniques. Both techniques pose a complex optimization problem with a huge search space. Therefore, we have proposed a search method based on evolutionary algorithms that allows us to obtain a solution to both problems in a reasonable time. The effectiveness of the proposed ProLSFEO-LDL method is verified on several real-world LDL datasets, showing significant improvements in comparison with using raw datasets.

**Keywords**: label distribution learning, evolutionary optimization, protoype selection, label-specific feature, machine learning

# 1. Introduction

A supervised learning process is the machine learning task of training a function that maps an input to an output based on data points with known outputs. Classification is the process of predicting to which of a set of categories a new observation belongs. Thus, the purpose of classification is to achieve a model that will be able to classify the right class to an unknown pattern. However, there are an increasing number of problems where a pattern can have several labels simultaneously associated. Examples can be found in genetics [6], image classification [8], etc. The generalization of the classic classification is Multi-Label Learning (MLL) [32, 33, 45, 53], where multiple labels can be assigned to each instance.

Nevertheless, in many real-world problems we may face cases where MLL is still not enough, as the degree of description of each label is different. To appoint only one dataset used in this paper, the biological experiments on yeast genes [18] during a period of time yield different levels of gene expression in a time serie. The exact level of expression at any point in time is of less importance. What becomes really decisive is the overall expression distribution over the whole period of time. If the learning goal is to predict that distribution for a particular gene, it cannot easily fit into the MLL framework because the role of each output in the distribution is critical, and there is no division of relevant and irrelevant labels at all.

The proposal of Label Distribution Learning (LDL) appeared for first time in 2013 [31] and was formally introduced in 2016 [28] in order to handle ambiguity on the label side of the mapping when one instance is not mandatorily matched to one single label. The main objective of this paradigm is to respond to the question "how much does each label describe the instance?" instead of "which label can describe the instance?".

Numerous studies have been conducted by applying the LDL framework to diverse real-life problem solving scenarios, e.g., facial age estimation [31], pre-release prediction of crowd opinion on movies [29], crowd counting in public video surveillance [69], sense beauty recognition [48], image emotion classification [62], personality recognition on social media [61], head pose estimation [60], etc. Further research has focused more on the development of new learners or on the adaptation of existing ones such as: instance-based algorithms (e.g., AA-$k$NN [28]), optimization algorithms (e.g., SA-IIS, SA-BFGS [28] or LDL-SCL [70]), decision trees (e.g., LDL forests [49]), deep learning algorithms (e.g., Deep Label Distribution [21]), or ensembles strategies (e.g., Logistic boosting regression for LDL [59], Structured random forest for LDL [15]).

AA-$k$NN [28] has proven to be a very competitive algorithm in previous experimental studies, achieving acceptable results and allowing an explainable model [5]. However, like any other instance-based algorithm, it suffers from several handicaps: it needs large memory requirements to store the training set, it is not efficient predicting due to the multiple computations of similarities between the test and training samples and it presents a low tolerance to noise because it uses all the data as relevant.

Nowadays, one of the main challenges of supervised learning algorithms is still how to deal with raw datasets. Data collection is usually an unsupervised process,

leading to datasets with redundant information, noisy data or irrelevant features. Hence, data pre-processing is an important step in the data mining process, that can mitigate this type of problem and generate improved datasets. The purpose of this paper is to address the LDL problem from the data pre-processing stage, by applying data reduction techniques which will allow us to have a reduced representation of the original data while maintaining the essential structure and integrity [24]. These pre-processing techniques often result in better performance of subsequent learners.

Two of the most widely used data reductions techniques are the instance selection [41], also known as prototype selection in case of instance-based algorithms [27], and the feature selection or reduction of the data dimensionality [51]. The first technique focuses on finding an optimal subset of samples to optimize the performance of the learner while the main idea of the second technique is to replace the original set of features by a new subset that extract the main information and provide an accurate classification.

A few methods are available to approach the instance selection in the MLL domain [4, 14, 35] but, to the best of our knowledge, there are no studies concerning instance or prototype selection reported in LDL. With regard to the feature selection, most of LDL algorithms are built on a simple feature space where all features are used to predict all the labels. However, in real-world applications, an instance is characterized by all labels, but some labels can only be determined by some specific features of their own. Although label-specific feature selection has been widely studied in MLL [34, 67], the studies we can find on this subject for LDL are still scarce [47].

Prototype selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning, or from now ProLSFEO-LDL, is our proposal for a pre-processing algorithm adapted to LDL problems and specifically designed to mitigate the handicaps of the AA-kNN method. We provide a novel method to simultaneously address the prototype selection and the label-specific feature selection. Both pre-processing techniques can be considered as a search problem where the search space is huge. Therefore, we have devised a search method based on evolutionary algorithms [71] that allows us to obtain a solution to both problems in a reasonable time. To this end, we have adapted elements of classical evolutionary algorithms to manage LDL restrictions, we have designed a representation of the solution and a way to measure its quality, which allows us to address both problems together.

In order to evaluate the proposal proficiency, we will compare an LDL-learner applying our ProLSFEO-LDL algorithm to the raw training set and measuring six aspects of their performance. We will repeat the experiment over 13 real-world datasets and validate the results of the empirical comparisons using Wilcoxon and Bayesian Sign tests [7, 12, 17].

The rest of the paper is organized as follows. First, a brief review and discussion of the foundations of LDL, a description of data reduction techniques and an introduction to the evolutionary optimization process are given in Section 2. The proposed ProLSFEO-LDL method is described in Section 3. Then the details of the experiments are reported in Section 4. Finally, the results and conclusions are drawn in Section 5 and Section 6, respectively.

# 2. Preliminaries

In this section, the foundations and the most relevant studies carried out on LDL (Section 2.1), are presented. Furthermore, some basic concepts on instance selection and label-specific features selection for classification are introduced (Section 2.2), as well as the notions needed to optimize solutions using evolutionary algorithms (Section 2.3), providing the necessary background required to properly present the study carried out in this paper.

## 2.1. Foundations of Label Distribution Learning

We can formalize a LDL problem as a set of $m$ training samples $S = \{(x_1, D_1), ..., (x_m, D_m)\}$, where $x_i = \{x_{i1}, x_{i2}, ..., x_{iq}\}$ is a $q$-dimensional vector. For each instance $x_i$, the label distribution is denoted by $D_i = \{d_{x_i}^{y_1}, d_{x_i}^{y_2}, ..., d_{x_i}^{y_c}\}$ where $y_j \in Y | j \in \{j, \cdots, c\}$, such that $Y = \{y_1, \cdots, y_c\}$ denotes the complete set of labels. The constant $c$ is the number of labels and $d_{x_i}^{y_j}$ is the description degree of the particular $j$th label $y_j$ for a particular $i$th instance $x_i$. According to the definition, for each $x_i$ the description degree should meet the constraints $d_{x_i}^{y_j} \in [0, 1]$ and $\sum_{j=1}^{c} d_{x_i}^{y_j} = 1$.

The solution to an LDL problem can be addressed from several perspectives. According to the selected approach, the algorithm to be developed may differ significantly, either a brand-new algorithm designed especially to deal with LDL constraints, or an adaptation of already available classification algorithms, reformulated to work with those constraints. The LDL study presented in [28] suggested six algorithms grouped in three categories. The first one is Problem Transformation (PT), a simple way to convert an LDL problem into a Single-Label Learning or SLL [20] problem. PT transforms the training samples into weighted single-label examples. Thus, any SLL algorithm may be applied. Two representative algorithms are PT-Bayes and PT-SVM. The second one is Algorithm Adaptation (AA), in which the algorithms are tailored to existing learning algorithms to handle directly with the label distribution. Two suitable algorithms were presented: AA-$k$NN, an adaptation of the well-known $k$-nearest neighbors method [1], and AA-BP, a three-layer backpropagation neural network. Finally, Specialized Algorithms (SAs), in contrast to the indirect strategy of PT and AA, directly match the LDL problem. SA-IIS and SA-BFGS are two specialized algorithms that learn by optimizing an energy function that is based on the maximum entropy model.

Subsequent works have successfully improved the results achieved by these original algorithms through different approaches. The methods LDLogitBoost and AOSO-LDLogitBoost proposed in [59], are a combination of the boosting method and the logistic regression applied to LDL framework. Deep Label Distribution Learning (DLDL) [21] and LDL based on Ensemble Neural Networks (ENN-LDL) [65] are two examples of success in the application of neural networks in LDL. Modelled on differentiable decision trees [38], an end-to-end strategy LDL forests proposed in [49] was used as the basis for Structured Random Forest (StructRF) [15]. BC-LDL [56] and DBC-LDL [57] use the binary coding strategies to address with the large-scale LDL problem. Classification with LDL (LDL4C) [55] is also

an interesting approach when the learned label distribution model is considered as a classification model. Feature selection on LDL [58] shows encouraging results by applying feature selection on LDL problems.

## 2.2. Prototype Selection and Label-Specific Feature Learning

Nowadays, one of the main challenges for supervised learning algorithms is still how to deal with raw datasets. Data pre-processing is an often unattended but important step in the data mining process [24]. Data gathering is often a poorly monitored process, resulting in low-quality datasets. If there is a lot of irrelevant and redundant information or noisy and unreliable data, then knowledge discovery is more difficult to carry out.

Data reduction techniques [24] allow us to obtain a reduced representation of the original data but maintaining the essential structure and integrity. Such pre-processing techniques usually lead to improved performance of subsequent learners.

A frequent problem with real datasets is their big volume, as well as the presence of noise and anomalies that complicate the learning process. Instance selection is to choose a subset of data to achieve the original purpose of a data mining application as if the whole data is used [41]. The optimal outcome of instance selection is a stand-alone model, with a minimal data sample that can perform tasks with little or no performance degradation.

Instance selection has been successfully applied to various problem types like imbalanced learning [44], data streams [46], regression [50], subgroup discover in large size datasets [10]. The research conducted in [25] is also of interest to investigate the impact of instance selection on the underlying structure of a dataset by analyzing the distribution of sample types. However, as of today, instance selection has not been extensively researched in the domain of MLL and to date only few methods have been made available [4, 14, 35]. As for LDL, we have not been able to find any studies to date.

Prototype Selection [24] methods are Instance Selection methods that expect to find training sets that offer the best ranking accuracy and reduction rates by using instances-based classifiers which consider a certain similarity or distance measure. A widely used categorization of prototype selection methods consists of three types of techniques [27]: Condensation, where the aim is to retain border points, preserving the accuracy of the training system; Edition, where the objective is to eliminate boundary points that are considered noise or do not match their neighbors but without removing the internal points of each dataset; or Hybrid methods that try to find a small set of training data while maintaining the performance of the classifier. The best approach considering the trade-off between reduction and accuracy is usually the hybrid technique.

To name just a few successful cases when applying prototype selection, [9] proposes a prototype selection algorithm called MONIPS that has proved to be competitive with classical prototype selection solutions adapted to monotonic classification. The experimental study presented in [16] shows a significant improvement when prototype selection is applied to dynamic classifier and ensemble selection.

Another widely used pre-processing technique is the reduction of the data dimensionality by means of feature selection [51]. The main idea is to replace the original set of features by a new subset that extract the main information and provide an accurate classification. However, in MLL and LDL, the strategy of selecting a set of characteristics shared by all labels may not be optimal since each label may be described by a specific subset of characteristics of their own. Label-specific feature learning has been widely studied in MLL. For instance, LIFT [67] firstly builds specific features of each label by performing cluster analysis on its positive and negative instances, and then conducts training and testing by querying the cluster results. LLSF [34] proposes learning label-specific features for each class label by considering pairwise (i.e., second-order) label correlations. MLFC [66] is also a multi-label learning method, which attempts to learn the specific characteristics of each label by exploiting the correlations between them. However, the studies we can find on this subject for LDL are still scarce. LDLSF [47] is a method inspired in LLSF adapted to deal with LDL problems by jointly selecting label-specific features, selecting common features and exploiting label correlations.

## 2.3. Evolutionary Optimization

Evolutionary algorithms (EAs) [71] are stochastic search mechanisms based on natural selection notions. EAs have been applied to a broad range of problems, including search problems [37], optimization problems [3], and in many areas as in economics, engineering, biology, etc. The primary idea is to maintain a population of chromosomes, that represent valid solutions to the problem and which evolve over time through a process of competition and targeted variation. CHC [19], is a well-known evolutionary model that introduces different techniques to achieve a trade-off between exploration and exploitation; such as incest prevention, reinitialization of the search process when the population converges or the search stops making progress and the competition among parents and offsprings into the replacement process.

Prototype Selection can be considered as a search problem where EAs can be applied. The search space consists of all the subsets of the training set. This can be represented by using a binary chromosome with two possible states for each gene. If the gene value is 1 then the associated instance is included in the subset, if the value is 0, this does not occur. The chromosome will have as many genes as the number of instances in the training set. EAs have been used to solve the prototype selection problem with promising results [22, 23, 26, 39, 54].

Label-specific feature selection is a very complex task. If the original set contains $q$ features and $c$ labels, the objective is to find a $q \times c$ binary matrix where each $ij$ position indicates if the individual feature $x_i$ will be taken into consideration to predicte the label $d_j$. Many search strategies can be used to find a solution to this problem but finding the optimal subset can be a huge time-consuming task. Therefore, it is justifiable to use an evolutionary algorithm that gives us an approximate solution in an acceptable time. Several studies have successfully applied feature reduction to multi-label problems using evolutionary algorithms [36, 40, 63, 68], but as of today no such technique has been used to solve the label-specific feature learning

problem.

# 3. ProLSFEO-LDL: Protoype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning

ProLSFEO-LDL is an evolutionary algorithm proposal adapted to LDL specificities that combines prototype selection and label-specific feature learning.

It uses the framework of CHC where the search space will be represented by a chromosome (or individual) in which we will code the prototype selection and label-specific feature as detailed in the next subsection 3.1. Through the evolutionary algorithm described in Algorithm 4, we will optimize this initial solution until we reach a solution that meets our expectations. We start from a parent population $P$ of size $N$, randomly initialized, to generate a new population $P'$ obtained by crossing the individuals of the parent population. The recombination method to obtain the offsprings is detailed in the subsection 3.2. Then, a survival competition is held where the best $N$ individuals from the parent population $P$ and the offspring population $P'$ are selected to compose the next generation. To determine if one individual is better than another we need to evaluate the quality of each chromosome, for this we use the fitness method described in the subsection 3.3. When the population converges or the search no longer progresses, the population is reinitiated to introduce a new diversity into the search, for this purpose we use a threshold $t$ to control when the reinitialization will take place, this step is explained in the subsection 3.4. The process of evolution iterates over $G$ generations and finally returns the best solution $B$ found in the search.

In the following sections, we explain in depth each part of the proposed method.

**Algorithm 4:** ProLSFEO-LDL: Prototype selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning

**Function** `ProLSFEO-LDL`(*S, N, G, t*)**:**

**input :** $S \leftarrow$ Training Dataset ;
$N \leftarrow$ Population size ;
$G \leftarrow$ Number of generations ;
$t \leftarrow$ Treshold ;
**output:** $B \leftarrow$ Best chromosome ;
$P$ = Initialize population of size $N$ ;
$L = t$ ;
$g = 0$ ;
**while** $g < G$ **do**
  Evaluate fitness of chromosomes in $P$ ;
  $P'$ = HUX crossover $(P)$ ;
  Evaluate fitness of chromosomes in $P'$ ;
  $New\_P$ = best $N$ chromosomes from $P \cup P'$ ;
  **if** $New\_P = P$ **then**
    $L = L - 1$ ;
  **end**
  **if** $L = 1$ **then**
    $B$ = best chromosome in $P$ ;
    $P$ = Initialize population of size $N - 1$ ;
    $P = P \cup B$ ;
    $L = t$ ;
  **else**
    $P = New\_P$ ;
  **end**
  $g = g + 1$ ;
**end**
$B$ = best chromosome from $P$;
**End Function**

## 3.1. Representation of Individuals

Each individual should represent the two parts of the algorithm. On the one hand, we have the first $m$ genes of the chromosome that represents the selection of instances, if the gene value is 1 then the associated instance is included in the subset, if the value is 0, this does not occur. And, the second part of the chromosome, consists of $q \times c$ genes representing the matrix of label-specific features. A graphical representation of genes, chromosome and population is exposed in Figure 5.1, where genes named as $ss_i$ represent if the instance $i$ will be included in the final set or not; and genes $fs_{jk}$ represent if feature $k$ will be taken into account to predicte the specific label $j$. The initialization of the chromosomes that make up the population is randomly performed from the "discrete uniform" distribution in the closed interval $[0, 1]$.

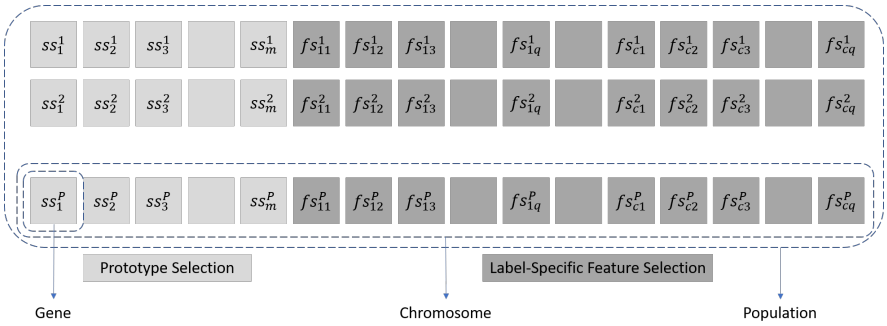



Figure 5.1: Gene, chromosome and population in evolutionary optimization.

## 3.2. Crossover

The crossover operator, HUX, performs a uniform cross that randomly exchanges exactly half of the bits that differ between the two parenting chains. No mutation is applied during the recombination phase of the CHC algorithm.

## 3.3. Fitness Function

In order to measure the quality of a particular chromosome we need to evaluate it using a LDL learner as wrapper. As the prototype selection is focused on the optimization of the AA-$k$NN method, we have chosen this same learner, but with some adaptations to support the selection of features. By default AA-$k$NN uses all features to calculate the distance between each instance. In our adaptation, each output label $j$ should be individually predicted using only the selected features from $fs = \{fs_{j1}, ..., fs_{jq}\}$ coded in the chromosome . The prediction obtained is then normalized so that the result is compatible with the LDL constraints.

The step sequence for evaluating a chromosome is as follows:

- Create a subset of instances: the selected prototypes are coded in the first $m$ genes of the chromosome. We create a subset $T$ by keeping only the elements of the original training set $S$ which have an associated gene value = 1.

- Prediction phase: we use the subset $T$ as the training set for AA-$k$NN. The prediction is then computed over a set merged from the test predictions of a 10-fold cross validation set (10-fcv) created from the original training set $S$.

- Evaluation phase: in order to measure the fitness of the chromosome, we calculate the distance between the predicted label distribution $\hat{D}$ and the real label distribution $D$ using the Kullback–Leibler divergence formula $KL(D, \hat{D}) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$, where $d_j$ and $\hat{d}_j$ are the description degree of the particular $j$th label. In our case the fitness of the chromosome directly matchs with the KL divergence.

The objective is to minimize the fitness function.

### 3.4. Reinitialization

When the population converges or the search no longer progresses, the population is reinitiated to introduce a new diversity into the search. The population is considered not sufficiently diverse when the threshold $L$ reaches 1 (line 13 in Algorithm 4). This treshold is initialized with the parameterized value $t$ (line 4 in Algorithm 4) and decreased by 1 each time that the population is not evolving (line 12 in Algorithm 4). In such a case, only the chromosome that represents the best solution found during the search is kept in the new population, and the remaining individuals are randomly generated, filling in the rest of the population (lines 14–17 in Algorithm 4). In CHC, the typical value used for threshold $t$ is equal to the total length of the chromosome divided by 4. In our case, due to the length of the chromosome, we will choose a lower value that allows a faster convergence.

## 4. Experimental Framework

This section is devoted to introducing the experimental framework used in the empirical study conducted in this paper. In our experiments, we have included 13 datasets of a wide variety of real-world problems, described in the following Section 4.1. In order to evaluate the learners proficiency, we will employ six measures described in Section 4.2 to evaluate the different aspects of experiments performed in Section 4.3.

### 4.1. Datasets

There are a total of 13 real-world datasets employed in the experiments. The summary of their characteristics is shown in Table 5.1.

Table 5.1: Datasets used in experiments.

| **No.** | **Datasets** | **Examples** ($m$) | **Features** ($q$) | **Labels** ($c$) |
|---|---|---|---|---|
| 1 | Yeast_alpha | 2465 | 24 | 18 |
| 2 | Yeast_cdc | 2465 | 24 | 15 |
| 3 | Yeast_cold | 2465 | 24 | 4 |
| 4 | Yeast_diau | 2465 | 24 | 7 |
| 5 | Yeast_dtt | 2465 | 24 | 4 |
| 6 | Yeast_elu | 2465 | 24 | 14 |
| 7 | Yeast_heat | 2465 | 24 | 6 |
| 8 | Yeast_spo | 2465 | 24 | 6 |
| 9 | Yeast_spo5 | 2465 | 24 | 3 |
| 10 | Yeast_spoem | 2465 | 24 | 2 |
| 11 | SJAFFE | 213 | 243 | 6 |
| 12 | SBU_3DFE | 2500 | 243 | 6 |
| 13 | Natural_Scene | 2000 | 294 | 9 |

The first to the tenth datasets were gathered from biological experiments on the

budding yeast Saccharomyces cerevisiae [18]. Each dataset consists of 2465 yeast genes, and an associated phylogenetic profile vector with a length of 24 is used to characterize each gene. In a biological experiment, the level of gene expression is usually uneven at each discrete time point, so the labels correspond to the time point.

Datasets JAFFE [43] and BU_3DFE [64] are two widely used facial expression image datasets. A total of 213 gray-scale expression images in the JAFFE dataset while BU_3DFE contains 2500 facial expression images. The images in JAFFE have been rated by 60 people based on the six primary emotion labels with a 5-level scale, i.e., fear, disgust, happiness, anger, sadness, surprise, and the images in BU_3DFE have been ranked by 23 people using the same scale as used in JAFFE. Each dataset is composed by a 243-dimensional feature vector extracted by the Local Binary Patterns method (LBP) [2]. The score for each emotion is considered the degree of description, and the description degrees (normalized gene expression level) of the six emotions make a label distribution for a particular facial expression image.

The Natural Scene dataset is compiled from 2000 natural scene images that have been inconsistently classified by ten human rankers. A 294-dimensional feature vector extracted in [8] represents each image and is linked with a multi-label selected from nine possible labels, i.e., sky, mountain, water, sun, cloud, snow, building, desert and plant. Then rankers are then required to sort the labels in descending order of relevance. Predictably, the ratings for the same image from different rankers are highly inconsistent. So, a nonlinear programming process [30] is applied to achieve the label distribution.

## 4.2. Evaluation Measure Selection

Several measures of distance/similarity between probability distributions can be applied to measure the distance/similarity between label distributions. We propose to use a set of six measures when comparing different LDL algorithms:

- Chebyshev Distance: is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension.

- Clark Distance: the Clark distance also called coefficient of divergence is the squared root of half of the divergence distance.

- Canberra Metric: is a numerical measure of the distance between pairs of points in a vector space. It is a weighted version of Manhattan distance. The Canberra distance is a metric function often used for data scattered around an origin.

- Kullback–Leibler Divergence: which is closely related to relative entropy, information divergence, and information for discrimination, is a non-symmetric measure of the difference between two probability distributions $p(x)$ and $q(x)$. Specifically, the Kullback–Leibler divergence of $q(x)$ from $p(x)$ is a measure of the information lost when $q(x)$ is used to approximate $p(x)$.

- Cosine coefficient: is a metric used to measure how similar two non-zero vectors are irrespective of their size. It measures the cosine of the angle between two vectors projected in a multidimensional space. The smaller the angle, higher the cosine similarity.
- Intersection similarity: has its largest value, 1, when all the terms of the first probability distribution are identical to the corresponding terms of the second probability distribution. Otherwise, the similarity is less than 1. In the extreme case, when both distributions are very different, then the similarity will be close to 0.

Each of these measures come from a different syntactic family: Minkowski family, the $X^2$ family, the $L_1$ family, the Shannon's entropy family, the inner product family, and the intersection family, respectively [13]. Taking into account that they come from different families and are significantly different in both syntax and semantics, they have a good chance to reflect different aspects of an LDL algorithm. Supposing that the real label distribution is $D = \{d_1, d_2, ..., d_c\}$, and the predicted label distribution is $\hat{D} = \{\hat{d}_1, \hat{d}_2, ..., \hat{d}_c\}$, then the formulation of the six measures is summarized in Table 5.2.

Table 5.2: Evaluation measure for LDL learners. ↓ means that the lowest value is the best and ↑ means the opposite.

| **Name** | **Formula** |
|---|---|
| Chebyshev(Cheby)↓ | $Dis(D, \hat{D}) = max_j \lvert d_j - \hat{d}_j \rvert$ |
| Clark↓ | $Dis(D, \hat{D}) = \sqrt{\sum_{j=1}^{c} \frac{(d_j - \hat{d}_j)^2}{(d_j + \hat{d}_j)^2}}$ |
| Canberra(Can)↓ | $Dis(D, \hat{D}) = \sum_{j=1}^{c} \frac{\lvert d_j - \hat{d}_j \rvert}{d_j + \hat{d}_j}$ |
| Kullback–Leibler(KL)↓ | $Dis(D, \hat{D}) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$ |
| Cosine(Cos)↑ | $Sim(D, \hat{D}) = \frac{\sum_{j=1}^{c} d_j \hat{d}_j}{\sqrt{\sum_{j=1}^{c} d_j^2} \sqrt{\sum_{j=1}^{c} \hat{d}_j^2}}$ |
| Intersection(Inter)↑ | $Sim(D, \hat{D}) = \sum_{j=1}^{c} min(d_j, \hat{d}_j)$ |

## 4.3. Experimental Setting

The proposed ProLSFEO-LDL algorithm is firstly applied, in a pre-preprocessing step, to the raw datasets obtaining a subset of training instances and a matrix of label-specific features. The subset of selected instances will make up the new preprocessed training set that the learner will train with. As we mentioned before, the selected learner is AA-$k$NN. Regarding the label-specific features, we have adapted the AA-$k$NN algorithm to support the selection of features, for this, each label is predicted separately using only the characteristics marked as selected. The prediction obtained is normalized to make it compatible with the LDL constraints.

The results will be compared with those obtained by the same learner without applying data reduction as a pre-processing step. All measures were computed over a merged set from the test predictions using a wrapped 10-fcv. Note that this

procedure is over the entire process and differs from the cross-validation used to estimate the fitness function within the optimization process.

The parameters used in experiments are summarized in Table 5.3. The AA-$k$NN algorithm used in the fitness function of ProLSFEO-LDL method and in the subsequent learner requires a value for $k$, set to four neighbors. We have selected a small value for $k$ in order to provide the most flexible fit with a low bias.

Table 5.3: Summary of the parameters.

| **Algorithm** | **Parameter** | **Description** | **Value** |
|---|---|---|---|
| ProLSFEO-LDL | N | Population size | 100 |
| | G | Number of generations | 500 |
| | t | Threshold | 10 |
| | $k$ | Number of selected neighbors in AA-$k$NN used for fitness function | 4 |
| AA-$k$NN | $k$ | Number of selected neighbors | 4 |

# 5. Results and Analysis

This section presents the results of the empirical studies and their analyses. We will compare the results obtained by AA-$k$NN with the results obtained by applying the pre-processing step ProLSFEO-LDL. In Table 5.4 the best outcome for each dataset and measure is highlighted in bold. The last row is the average aggregation result of each column. The best average is also highlighted in bold. As those algorithms have been tested using 10-fcv, the performance is represented using "mean±standard deviation".

The Wilcoxon test and Bayesian Sign test [7, 17] are used to validate the results of the empirical comparisons. In the Bayesian Sign test, a distribution of the differences of the results achieved using methods $L$ (AA-$k$NN) and $R$ (ProLSFEO-LDL) is computed into a graphical space divided into three regions: left, rope and right. The location of most of the distribution in these sectors indicates the final decision: the superiority of algorithm $L$, statistical equivalence and the superiority of algorithm $R$, respectively. KEEL package [52] has been used to compute the Wilcoxon test and the R package rNPBST [11] was used to extract the graphical representations of the Bayesian Sign tests analyzed in the following empirical studies. The Rope limit parameter used to represent the Bayesian Sign test is 0.0001.

The outcome of both statistical tests applied to our method is represented in Table 5.5 (Wilcoxon test) and Figure 5.2 (Bayesian Sign test).

Table 5.4: Experimental Results (mean ± std).

| | Cheby ↓ | | Clark ↓ | | Can ↓ | |
|---|---|---|---|---|---|---|
| **Dataset** | **AA-$k$NN** | **ProLSFEO-LDL** | **AA-$k$NN** | **ProLSFEO-LDL** | **AA-$k$NN** | **ProLSFEO-LDL** |
| Yeast_alpha | 0.0148 ± 0.0007 | **0.0145 ± 0.0007** | 0.2321 ± 0.0112 | **0.2280 ± 0.0111** | 0.7577 ± 0.0372 | **0.7451 ± 0.0376** |
| Yeast_cdc | 0.0177 ± 0.0010 | **0.0174 ± 0.0009** | 0.2375 ± 0.0134 | **0.2316 ± 0.0129** | 0.7179 ± 0.0395 | **0.6972 ± 0.0395** |
| Yeast_cold | 0.0554 ± 0.0021 | **0.0545 ± 0.0015** | 0.1509 ± 0.0070 | **0.1485 ± 0.0050** | 0.2611 ± 0.0129 | **0.2563 ± 0.0090** |
| Yeast_diau | **0.0393 ± 0.0009** | 0.0397 ± 0.0013 | **0.2122 ± 0.0042** | 0.2125 ± 0.0055 | 0.4560 ± 0.0109 | **0.4528 ± 0.0126** |
| Yeast_dtt | 0.0393 ± 0.0016 | **0.0381 ± 0.0013** | 0.1068 ± 0.0045 | **0.1035 ± 0.0036** | 0.1836 ± 0.0075 | **0.1778 ± 0.0050** |
| Yeast_elu | 0.0177 ± 0.0005 | **0.0174 ± 0.0004** | 0.2182 ± 0.0048 | **0.2137 ± 0.0048** | 0.6444 ± 0.0155 | **0.6285 ± 0.0137** |
| Yeast_heat | 0.0451 ± 0.0012 | **0.0445 ± 0.0007** | 0.1955 ± 0.0044 | **0.1928 ± 0.0035** | 0.3924 ± 0.0096 | **0.3867 ± 0.0078** |
| Yeast_spo | 0.0643 ± 0.0024 | **0.0637 ± 0.0030** | 0.2715 ± 0.0112 | **0.2690 ± 0.0124** | 0.5594 ± 0.0232 | **0.5499 ± 0.0248** |
| Yeast_spo5 | 0.0962 ± 0.0043 | **0.0951 ± 0.0052** | 0.1933 ± 0.0099 | **0.1914 ± 0.0110** | 0.2972 ± 0.0145 | **0.2938 ± 0.0169** |
| Yeast_spoem | 0.0924 ± 0.0036 | **0.0887 ± 0.0036** | 0.1374 ± 0.0053 | **0.1320 ± 0.0055** | 0.1913 ± 0.0074 | **0.1837 ± 0.0076** |
| SJAFFE | 0.1155 ± 0.0180 | **0.1088 ± 0.0090** | 0.4137 ± 0.0489 | **0.4070 ± 0.0285** | 0.8527 ± 0.0962 | **0.8359 ± 0.0624** |
| SBU_3DFE | 0.1350 ± 0.0048 | **0.1272 ± 0.0056** | 0.4255 ± 0.0134 | **0.4068 ± 0.0146** | 0.8746 ± 0.0290 | **0.8332 ± 0.0313** |
| Natural_Scene | 0.3168 ± 0.0081 | **0.3046 ± 0.0118** | **1.8253 ± 0.0343** | 1.9161 ± 0.0347 | **4.2609 ± 0.0866** | 4.5848 ± 0.1206 |
| Average | 0.0807 ± 0.0038 | **0.0780 ± 0.0035** | **0.3554 ± 0.0133** | 0.3579 ± 0.0118 | **0.8038 ± 0.0300** | 0.8174 ± 0.0299 |
| | **KL ↓** | | **Cos ↑** | | **Inter ↑** | |
| **Dataset** | **AA-$k$NN** | **ProLSFEO-LDL** | **AA-$k$NN** | **ProLSFEO-LDL** | **AA-$k$NN** | **ProLSFEO-LDL** |
| Yeast_alpha | 0.0066 ± 0.0006 | **0.0064 ± 0.0006** | 0.9935 ± 0.0006 | **0.9937 ± 0.0006** | 0.9581 ± 0.0020 | **0.9588 ± 0.0021** |
| Yeast_cdc | 0.0083 ± 0.0009 | **0.0080 ± 0.0009** | 0.9920 ± 0.0008 | **0.9924 ± 0.0008** | 0.9527 ± 0.0025 | **0.9541 ± 0.0026** |
| Yeast_cold | 0.0142 ± 0.0014 | **0.0137 ± 0.0012** | 0.9866 ± 0.0012 | **0.9871 ± 0.0010** | 0.9356 ± 0.0031 | **0.9368 ± 0.0022** |
| Yeast_diau | 0.0150 ± 0.0008 | **0.0146 ± 0.0009** | 0.9862 ± 0.0008 | **0.9866 ± 0.0009** | 0.9367 ± 0.0017 | **0.9372 ± 0.0018** |
| Yeast_dtt | 0.0073 ± 0.0007 | **0.0069 ± 0.0006** | 0.9931 ± 0.0005 | **0.9934 ± 0.0004** | 0.9547 ± 0.0017 | **0.9561 ± 0.0011** |
| Yeast_elu | 0.0074 ± 0.0003 | **0.0071 ± 0.0003** | 0.9928 ± 0.0003 | **0.9932 ± 0.0003** | 0.9545 ± 0.0011 | **0.9556 ± 0.0003** |
| Yeast_heat | 0.0146 ± 0.0007 | **0.0142 ± 0.0005** | 0.9861 ± 0.0007 | **0.9865 ± 0.0005** | 0.9356 ± 0.0017 | **0.9365 ± 0.0013** |
| Yeast_spo | 0.0302 ± 0.0024 | **0.0287 ± 0.0026** | 0.9716 ± 0.0020 | **0.9730 ± 0.0023** | 0.9076 ± 0.0036 | **0.9093 ± 0.0040** |
| Yeast_spo5 | 0.0333 ± 0.0033 | **0.0315 ± 0.0030** | 0.9705 ± 0.0025 | **0.9720 ± 0.0024** | 0.9038 ± 0.0043 | **0.9049 ± 0.0052** |
| Yeast_spoem | 0.0285 ± 0.0023 | **0.0256 ± 0.0021** | 0.9754 ± 0.0019 | **0.9777 ± 0.0019** | 0.9076 ± 0.0036 | **0.9113 ± 0.0036** |
| SJAFFE | 0.0730 ± 0.0162 | **0.0672 ± 0.0091** | 0.9308 ± 0.0163 | **0.9366 ± 0.0085** | 0.8529 ± 0.0176 | **0.8572 ± 0.0106** |
| SBU_3DFE | 0.0907 ± 0.0048 | **0.0816 ± 0.0061** | 0.9118 ± 0.0047 | **0.9197 ± 0.0058** | 0.8394 ± 0.0053 | **0.8474 ± 0.0060** |
| Natural_Scene | 1.1924 ± 0.0765 | **0.9908 ± 0.0654** | 0.7043 ± 0.0137 | **0.7295 ± 0.0126** | 0.5634 ± 0.0098 | **0.5705 ± 0.0125** |
| Average | 0.1170 ± 0.0085 | **0.0997 ± 0.0072** | 0.9534 ± 0.0035 | **0.9570 ± 0.0029** | 0.8925 ± 0.0045 | **0.8951 ± 0.0041** |

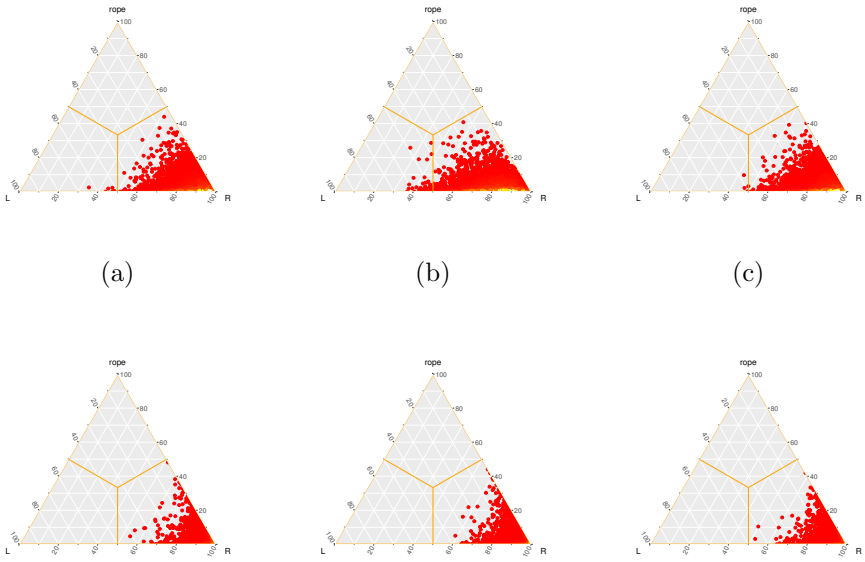
Comparing ProLSFEO-LDL with the standard LDL learner AA-$k$NN, we reach the following conclusions:

- The results of the different measures shown in Table 5.4 highlights the best ranking of ProLSFEO-LDL in the large majority of the datasets and measures.
- The Wilcoxon Signed Ranks test corroborates the significance of the differences between our approach and AA-$k$NN. As we can see in Table 5.5, all the hypotheses of equivalence are rejected with small p-values.
- With regard to the Bayesian Sign test, Figure 5.2 graphically represent the statistical significance in terms of precision between ProLSFEO-LDL and AA-$k$NN. The following heat-maps clearly indicate the significant superiority of ProLSFEO-LDL, as the computed distributions are always located in the right region.

Another interesting information to analyse is the reduction ratio obtained by the proposed method. In Figure 5.3 we show the percentage of selected prototypes and features with respect to the initial training set. In the case of features we represent the average percentage since each of the output labels can be represented by a different number of features.

The average percentage is around 53% for both, prototypes and features, varying more significantly for the SJAFFE and SBU_3DFE datasets where the reduction ratio decreases significantly. This percentaje of data reduction will have a huge

impact in the performance of the learner. In instance-based methods like AA-$k$NN, the fact of handling about half of the prototypes and features, will lead to a reduction of computation time in the prediction phase.




(a) (b) (c)

(d) (e) (f)

Figure 5.2: Bayesian Sign test comparing AA-$k$NN(L) vs. ProLSFEO-LDL(R). (**a**) Chebyshev Distance. (**b**) Clark Distance. (**c**) Canberra Metric. (**d**) Kullback–Leibler Divergence. (**e**) Cosine Coefficient. (**f**) Intersection Similarity.

Table 5.5: Wilcoxon Signed Ranks test comparing ProLSFEO-LDL vs. AA-$k$NN.

| **Measure** | $R^+$ | $R^-$ | $p$-**Value** |
|---|---|---|---|
| Cheby | 87 | 4 | 0.0017 |
| Clark | 77 | 14 | 0.0266 |
| Can | 78 | 13 | 0.0215 |
| KL | 91 | 0 | 0.0002 |
| Cos | 91 | 0 | 0.0002 |
| Inter | 91 | 0 | 0.0002 |

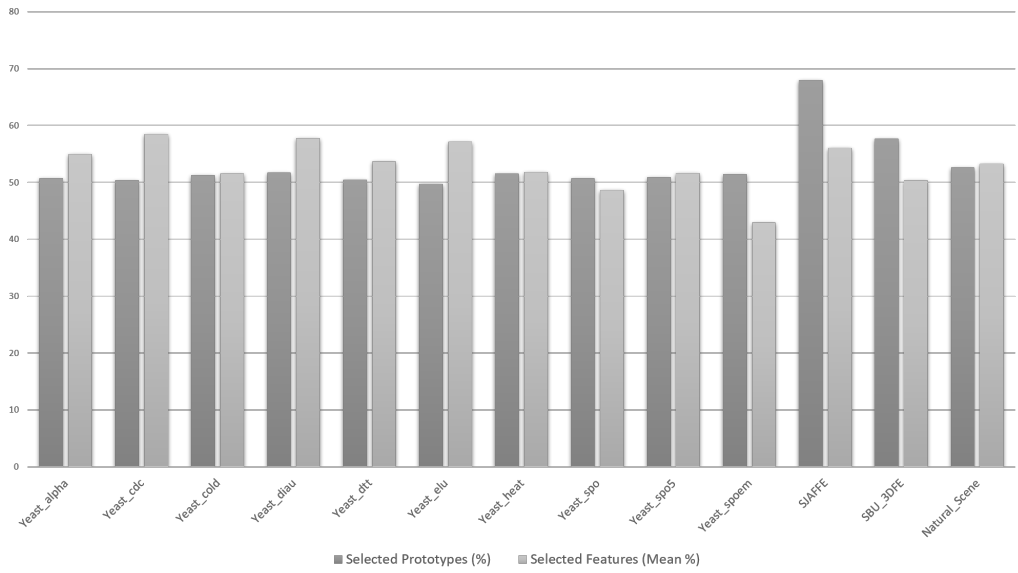



Figure 5.3: Number of selected prototypes and features (%).

# 6. Conclusions

In this paper, we proposed a novel data reduction algorithm that adapts to LDL constraints. It simultaneously address the prototype selection and the label-specific feature selection with two objectives: finding an optimal subset of samples to improve the performance of the AA-$k$NN learner and selecting a subset of characteristics specific for each one of the output label. Both tasks have been addressed as search problems using an evolutionary algorithm, based on CHC, to optimize the solution.

In order to verify the effectiveness of the solution designed, ProLSFEO-LDL has been applied on several real-world LDL datasets, showing significant improvements compared to the use of the raw training set. The results of the different measures highlights the best ranking of ProLSFEO-LDL, outcomes subsequently corroborated by statistical tests. In addition, the percentaje of data reduction reached leads to a significant improvement of prediction time.

In future studies we may introduce some further comparisons between already existing approaches and also make improvements to the presented proposal such as:

- It might be interesting to compare the results obtained by this study with the following techniques: LDLSF [47] and Binary Coding bases LDL [56].

- Complement the experimental analysis by dealing with larger datasets (Big Data). To this end, we will complement the current proposal with some of the big data reduction techniques presented in [42].

- The experimental settings use the AA-$K$NN learner to measure the quality of the solution applied over the pre-processed dataset. Other more powerful LDL learners could be considered for this task but they must previously be adapted to support the label-specific feature selection. With this we will be able to

carry out more adequate comparisons with state-of-the-art LDL methods like LDLFs proposed in [49] or StructRF [15].

- Another interesting study that we will undertake is how the presence of noise at the label side can affect the performance. In real scenarios the data gathering is often an automatic procedure that can lead to incorrect sample labelling. It would be interesting to inject some artificial noise to the training labels in order to check the robustness of the implemented approach.

# Acknowledgments

This work is supported by the Spanish National Research Project TIN2017-89517-P.

# Bibliography


[1] D.W. Aha, D. Kibler, and M.K. Albert, "Instance-based learning algorithms", *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[2] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[3] A.F. Ali and M.A. Tawhid, "A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems", *Ain Shams Engineering Journal*, vol. 8, no. 2, pp. 191–206, 2017.

[4] A. Arnaiz-González, J.F. Díez-Pastor, J.J. Rodríguez, and C. García-Osorio, "Local sets for multi-label instance selection", *Applied Soft Computing*, vol. 68, pp. 651–666, 2018.

[5] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", *Information Fusion*, vol. 58, pp. 82–115, 2020.

[6] Z. Barutcuoglu, R.E. Schapire, and O.G. Troyanskaya, "Hierarchical multi-label prediction of gene function", *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[7] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[8] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown, "Learning multi-label scene classification", *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[9] J.R. Cano, N.R. Aljohani, R.A. Abbasi, J.S. Alowidbi, and S. García, "Prototype selection to improve monotonic nearest neighbor", *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 128–135, 2017.

[10] J.R. Cano, S. García, and F. Herrera, "Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes", *Pattern Recognition Letters*, vol. 29, no. 16, pp. 2156–2164, 2008.

[11] J. Carrasco, S. García, M. del Mar Rueda, and F. Herrera, "rnpbst: An R package covering non-parametric and bayesian statistical tests", in *International Conference on Hybrid Artificial Intelligence Systems*, Springer, 2017, pp. 281–292.

[12] J. Carrasco, S. García, M.M. Rueda, S. Das, and F. Herrera, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review", *Swarm and Evolutionary Computation*, Art. 100665, 2020.

[13] S.H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions", *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 2, pp. 300–307, 2007.

[14] F. Charte, A.J. Rivera, M.J. del Jesus, and F. Herrera, "REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization", *Neurocomputing*, vol. 326, pp. 110–122, 2019.

[15] M. Chen, X. Wang, B. Feng, and W. Liu, "Structured random forest for label distribution learning", *Neurocomputing*, vol. 320, pp. 171–182, 2018.

[16] R.M.O. Cruz, R. Sabourin, and G.D.C. Cavalcanti, "Analyzing different prototype selection techniques for dynamic classifier and ensemble selection", in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 3959–3966.

[17] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[18] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns", *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[19] L.J. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination", in *Foundations of Genetic Algorithms*, vol. 1, Elsevier, 1991, pp. 265–283.

[20] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[21] B.B. Gao, C. Xing, C.W. Xie, J. Wu, and X. Geng, "Deep label distribution learning with label ambiguity", *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2825–2838, 2017.

[22] S. García, J.R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach", *Pattern Recognition*, vol. 41, no. 8, pp. 2693–2709, 2008.

[23] S. García and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy", *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, 2009.

[24] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Springer, 2015.

[25] V. García, J.S. Sánchez, A. Ochoa-Ortiz, and A. López-Najera, "Instance Selection for the Nearest Neighbor Classifier: Connecting the Performance to the Underlying Data Structure", in *Iberian Conference on Pattern Recognition and Image Analysis*, Springer, 2019, pp. 249–256.

[26] S. García, J.R. Cano, E. Bernado-Mansilla, and F. Herrera, "Diagnose effective evolutionary prototype selection using an overlapping measure", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 08, pp. 1527–1548, 2009.

[27] S. García, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[28] X. Geng, "Label distribution learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1734–1748, 2016.

[29] X. Geng and P. Hou, "Pre-release prediction of crowd opinion on movies by label distribution learning", in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 3511–3517.

[30] X. Geng and L. Luo, "Multilabel ranking with inconsistent rankers", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3742–3747.

[31] X. Geng, C. Yin, and Z.H. Zhou, "Facial age estimation by learning from label distributions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2401–2412, 2013.

[32] E. Gibaja and S. Ventura, "A tutorial on multilabel learning", *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 1–38, 2015.

[33] F. Herrera, F. Charte, A.J. Rivera, and M.J. Del Jesus, *Multilabel classification: Problem analysis, metrics and techniques*. Springer, 2016.

[34] J. Huang, G. Li, Q. Huang, and X. Wu, "Learning label-specific features and class-dependent labels for multi-label classification", *IEEE transactions on knowledge and data engineering*, vol. 28, no. 12, pp. 3309–3323, 2016.

[35] S. Kanj, F. Abdallah, T. Denoeux, and K. Tout, "Editing training data for multi-label classification with the k-nearest neighbor rule", *Pattern Analysis and Applications*, vol. 19, no. 1, pp. 145–161, 2016.

[36] M. Khan, A. Ekbal, E. Mencía, and J. Fürnkranz, "Multi-objective Optimisation-Based Feature Selection for Multi-label Classification", in *International Conference on Applications of Natural Language to Information Systems*, Jun. 2017, pp. 38–41.

[37] S.S. Khan, S.M.K. Quadri, and M.A. Peer, "Genetic Algorithm for Biomarker Search Problem and Class Prediction", *International Journal of Intelligent Systems and Applications*, vol. 8, no. 9, p. 47, 2016.

[38] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1467–1475.

[39] M. Kordos, A. Arnaiz-González, and C. García-Osorio, "Evolutionary prototype selection for multi-output regression", *Neurocomputing*, vol. 358, pp. 309–320, 2019.

[40] J. Lee and D.W. Kim, "Memetic feature selection algorithm for multi-label classification", *Information Sciences*, vol. 293, pp. 80–96, 2015.

[41] H. Liu and H. Motoda, "On issues of instance selection", *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115–130, 2002.

[42] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, *Big Data Preprocessing: Enabling Smart Data*. Springer, 2020.

[43] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets", in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, IEEE, 1998, pp. 200–205.

[44] M. Millán-Giraldo, V. García, and J.S. Sánchez, "Instance Selection Methods and Resampling Techniques for Dissimilarity Representation with Imbalanced Data Sets", in *Pattern Recognition-Applications and Methods*, Springer, 2013, pp. 149–160.

[45] J.M. Moyano, E.L. Gibaja, K.J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: models, experimental study and prospects", *Information Fusion*, vol. 44, pp. 33–45, 2018.

[46] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions", *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[47] T. Ren, X. Jia, W. Li, L. Chen, and Z. Li, "Label distribution learning with label-specific features", in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3318–3324.

[48] Y. Ren and X. Geng, "Sense Beauty by Label Distribution Learning.", in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 2648–2654.

[49] W. Shen, K. Zhao, Y. Guo, and A.L. Yuille, "Label distribution learning forests", in *Advances in Neural Information Processing Systems*, 2017, pp. 834–843.

[50] Y. Song, J. Liang, J. Lu, and X. Zhao, "An efficient instance selection algorithm for k nearest neighbor regression", *Neurocomputing*, vol. 251, pp. 26–34, 2017.

[51] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review", *Data classification: Algorithms and applications*, pp. 37–64, 2014.

[52] I. Triguero, S. González, J.M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. Fernández Hilario, J. Díaz, L. Sánchez, F. Herrera, *et al.*, "KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining", *International Journal of Computational Intelligence Systems*, vol. 10, pp. 1238–1249, 1 2017.

[53] I. Triguero and C. Vens, "Labelling strategies for hierarchical multi-label classification techniques", *Pattern Recognition*, vol. 56, pp. 170–183, 2016.

[54] S. Vluymans, I. Triguero, C. Cornelis, and Y. Saeys, "EPRENNID: An evolutionary prototype reduction based ensemble for nearest neighbor classification of imbalanced data", *Neurocomputing*, vol. 216, pp. 596–610, 2016.

[55] J. Wang and X. Geng, "Classification with label distribution learning", in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3712–3718.

[56] K. Wang and X. Geng, "Binary Coding based Label Distribution Learning.", in *International Joint Conferences on Artificial Intelligence*, 2018, pp. 2783–2789.

[57] K. Wang and X. Geng, "Discrete binary coding based label distribution learning", in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3733–3739.

[58] Y. Wang and J. Dai, "Label Distribution Feature Selection Based on Mutual Information in Fuzzy Rough Set Theory", in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–2.

[59] C. Xing, X. Geng, and H. Xue, "Logistic boosting regression for label distribution learning", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4489–4497.

[60] L. Xu, J. Chen, and Y. Gan, "Head pose estimation using improved label distribution learning with fewer annotations", *Multimedia Tools and Applications*, pp. 1–22, 2019.

[61] D. Xue, Z. Hong, S. Guo, L. Gao, L. Wu, J. Zheng, and N. Zhao, "Personality recognition on social media with label distribution learning", *IEEE Access*, vol. 5, pp. 13 478–13 488, 2017.

[62] J. Yang, D. She, and M. Sun, "Joint Image Emotion Classification and Distribution Learning via Deep Convolutional Neural Network.", in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 3266–3272.

[63] J. Yin, T. Tao, and J. Xu, "A multi-label feature selection algorithm based on multi-objective optimization", in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2015, pp. 1–7.

[64] L. Yin, X. Wei, Y. Sun, J. Wang, and M.J. Rosato, "A 3D facial expression database for facial behavior research", in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, IEEE, 2006, pp. 211–216.

[65] Y. Zhai, J. Dai, and H. Shi, "Label Distribution Learning Based on Ensemble Neural Networks", in *International Conference on Neural Information Processing*, Springer, 2018, pp. 593–602.

[66] J. Zhang, C. Li, D. Cao, Y. Lin, S. Su, L. Dai, and S. Li, "Multi-label learning with label-specific features by resolving label correlations", *Knowledge-Based Systems*, vol. 159, pp. 148–157, 2018.

[67] M.L. Zhang and L. Wu, "Lift: Multi-label learning with label-specific features", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 107–120, 2014.

[68] Y. Zhang, D.W. Gong, and M. Rong, "Multi-objective differential evolution algorithm for multi-label feature selection in classification", in *International Conference in Swarm Intelligence*, Springer, 2015, pp. 339–345.

[69] Z. Zhang, M. Wang, and X. Geng, "Crowd counting in public video surveillance by label distribution learning", *Neurocomputing*, vol. 166, pp. 151–163, 2015.

[70] X. Zheng, X. Jia, and W. Li, "Label distribution learning by exploiting sample correlations locally", in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018, pp. 4556–4563.

[71] Z.H. Zhou, Y. Yu, and C. Qian, *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.

# Capítulo 6

# Descomposición-Fusión para el Aprendizaje de la Distribución de Etiquetas

González, M., González-Almagro, G., Triguero, I., Cano, J. R., & García, S. (2021). Decomposition-Fusion for Label Distribution Learning. Information Fusion, 66, 64-75.

| | **JCR Clarivate**<br>IF: 13.669 | **CiteScore Scopus**<br>21.9 |
|---|---|---|
| **Categoría** | Computer Science, Artificial Intelligence | Information Systems |
| **Ranking** | 2/137 | 1/300 |
| **Cuartil** | Q1 | Q1 |

Factores de Impacto 2019

# Decomposition-Fusion for Label Distribution Learning

**Manuel González**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
manuel.gonzalez.es@gmail.com

**Germán González-Almagro**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
germangalmagro@ugr.es

**Isaac Triguero**
The Computational Optimisation and Learning Lab. School of Computer Science
University of Nottingham. Jubilee Campus. Nottingham NG8 1BB, United Kingdom
isaac.triguero@nottingham.ac.uk

**José-Ramón Cano**
Department of Computer Science
University of Jaén, 23700 Linares, Jaén, Spain
jrcano@ujaen.es

**Salvador García**
Department of Computer Science and Artificial Intelligence
University of Granada, 18071 Granada, Spain
salvagl@decsai.ugr.es

**Abstract**

Label Distribution Learning (LDL) is a general learning framework that assigns an instance to a distribution over a set of labels rather than to a single label or multiple labels. Current LDL methods have proven their effectiveness in many real-life machine learning applications. However, LDL is a generalization of the classification task and as such it is exposed to the same problems as standard classification algorithms, including class-imbalanced, noise, overlapping or irregularities. The purpose of this paper is to mitigate these effects by using decomposition strategies. The technique devised, called Decomposition-Fusion for LDL (DF-LDL), is based on one of the most renowned strategy in decomposition: the One-vs-One scheme, which we adapt to be able to deal with LDL datasets. In addition, we propose a competent fusion method that allows us to discard non-competent classifiers when their output is probably not of interest. The effectiveness of the proposed DF-LDL method is verified on several real-world LDL datasets on which we have carried out two types of experiments. First, comparing our proposal with the base learners and, second, comparing our proposal with the state-of-the-art LDL algorithms. DF-LDL shows

significant improvements in both experiments.

**Keywords**: label distribution learning, decomposition strategies, one vs. one, machine learning

# 1. Introduction

A supervised learning process is the machine learning task of training a predictive model using data points with known outputs. Classification is the problem of identifying to which of a set of categories a new observation belongs. Hence, the aim of classification is to obtain a model that will be able to assign the correct class to an unknown pattern. However, there is a growing number of problems in which a pattern can have several labels simultaneously associated. Examples are found in image classification [6] or genetics [3], etc. Multi-Label Learning (MLL) [30, 33, 38, 47] is a generalization of the traditional classification where multiple labels may be assigned to each instance.

Nevertheless, in many real-world problems we can find cases in which MLL is still not sufficient since the level of description of each label is not the same. To name just one example from the datasets used in this paper, the biological experiments on the yeast genes [16] over a period of time result in different levels of gene expression in a time series. The precise degree of expression at each point in time is of minor significance. What is really crucial is the distribution of the overall expression over the entire period of time. If the learning task is to predict that distribution for a given gene, it can hardly fit into the MLL framework because the role of the individual output in the distribution is crucial, and there is no partitioning of relevant and irrelevant labels at all.

The Label Distribution Learning (LDL) concept appeared for first time in 2013 [29] and was formally described in 2016 [26] in order to deal with ambiguity on the label side of the mapping when one instance is not necessarily mapped to one single label. The aim of this paradigm is to answer the question "how much does each label describe the instance?" instead of "which label can describe the instance?".

From the first formulation of the LDL problem, numerous studies have been carried out applying the LDL methodology to various real life problem solving situations, e.g.: sense beauty recognition [41], facial age estimation [29], personality recognition on social media [55], image emotion classification [56], pre-release prediction of crowd opinion on movies [27], crowd counting in public video surveillance [62], head pose estimation [54], etc. Other studies have focused more on developing new learners or on adapting existing learners such as: instance-based algorithms (e.g., AA-$k$NN [26]), optimization algorithms (e.g., SA-IIS, SA-BFGS [26] or LDL-SCL [64]), decision trees (e.g., LDL forests [45]), deep learning algorithms (e.g., Deep Label Distribution [24]), or ensembles strategies (e.g., Logistic boosting regression for LDL [53], Structured random forest for LDL [10]).

LDL is a generalization of the classification task [49] and as a result is vulnerable to the same problems as conventional classification algorithms: imbalanced datasets [17], when there is a disproportion in the number of examples of the different classes; noisy data [40], because of imperfections in data acquisition, transmission or storage;

overlapping [39], when the input features are not sufficient to correctly differentiate among instances of different classes; or irregularities [12], situations where the distribution of data points, the sampling of the data space to generate the training set, and the characteristics that describe each data point deviate from what might have been ideal, being biased, skewed, incomplete and/or misleading.

Over the last years, decomposition strategies for addressing classification problems have been widely studied in the literature [44]. The same underlying idea is behind all proposals for decomposition: to solve a multi-class problem using binary classifiers. Decomposition strategies have proven to be efficient in dealing with the difficulties presented above and that is why, in this paper, we propose a decomposition algorithm adapted to LDL restrictions. The devised technique is inspired by one of the most renowned strategies in decomposition: the One-vs-One (OVO) [32] scheme, where the original problem is divided into binary problems that distinguish between the different pairs of classes, every single division will be trained with a base classifier. This method usually requires an additional step to fuse the outputs from single classifiers in order to produce the final result.

We design a decomposition strategy that can handle label distribution, capable of dealing with real values instead of multi-class outputs. While OVO uses a binary classifier as base learner (the learning algorithm used for solving binary problems), in our proposal we will rather rely on a specific LDL learner. In addition to these, we also propose a fusion method, capable of providing an output according to the LDL constraints, and that also will allow us to discard non-competent classifiers when their output is probably not of interest.

Decomposition-Fusion for LDL, from now DF-LDL, is our decomposition proposal for LDL type problems. In order to evaluate the proposal proficiency, we will carry out two types of experiments: on the one hand we will compare the results obtained by the base learners with our DF-LDL algorithm using the same learner as base classifier and, on the other hand, we will compare our proposal with the state-of-the-art LDL algorithms, measuring in all cases six aspects of their performance. We will repeat the experiment over 17 real-world datasets and validate the results of the empirical comparisons using Wilcoxon, Friedman rank and Bayesian Sign tests [4, 13].

In summary, the main contributions of this paper are:

- A decomposition strategy to handle LDL problems.
- A fusion method custom-designed to provide an output compliant with LDL restrictions and that also allows to exclude non-competent classifiers.
- From a technical point of view, the proposed solution could be implemented in a very fast way taking advantage of any existing LDL learner.

The rest of the paper is organized as follows. First, a brief review and discussion of the foundations of LDL and decomposition strategies for classification are given in Section 2. The proposed Decomposition-Fusion for LDL method is described in Section 3. Then the details of the experiments are reported in Section 4. Finally, the results and conclusions are drawn in Section 5 and Section 6, respectively.

# 2. Preliminaries

In this section, the foundations, as well as the most relevant studies carried out on LDL (Section 2.1), are presented. Furthermore, some basic concepts on decomposition strategies for classification are introduced (Section 2.2), providing the necessary background required to properly present the study carried out in this paper. We will conclude this section by explaining the motivations for applying decomposition strategies to LDL-type problems.

## 2.1. Foundations of Label Distribution Learning

We can formalize an LDL problem as a set of $m$ training samples $S = \{(x_1, D_1), ..., (x_m, D_m)\}$, where $x_i = \{x_{i1}, x_{i2}, ..., x_{iq}\}$ is a $q$-dimensional vector. For each instance $x_i$, the label distribution is denoted by $D_i = \{d_{x_i}^{y_1}, d_{x_i}^{y_2}, ..., d_{x_i}^{y_c}\}$ where $y_i \in Y | i \in \{i, \cdots, c\}$, such that $Y = \{y_1, \cdots, y_c\}$ denotes the complete set of labels. The constant $c$ is the number of possible labels and $d_{x_i}^{y_j}$ is the description degree of the particular $j$th label $y_j$ for a particular $i$th instance $x_i$. According to the definition, each description degree should meet the constraints $d_x^y \in [0, 1]$ and $\sum_{y=1}^{c} d_x^y = 1$.

Solving an LDL problem can be approached from different perspectives. Depending on the approach chosen, the algorithm to be developed will vary considerably, either a completely new algorithm developed specifically to deal with LDL constraints, or an adaptation of existing classification algorithms, reformulated to work with these constraints. The LDL study published in [26] proposed six algorithms classified in three categories. The first one is Problem Transformation (PT), a straightforward way to transform an LDL problem into a Single-Label Learning or SLL [18] problem is to change the training examples into weighted single-label examples. In this way, any SLL algorithm can be applied. Two representative algorithms are PT-Bayes and PT-SVM. The second one is Algorithm Adaptation (AA), where algorithms are adapted to existing learning algorithms to deal directly with the label distribution. Two suitable algorithms were presented: AA-$k$NN, an adaptation of the well-known $k$-nearest neighbors method [1], and AA-BP, a tree-layer backpropagation neural network. Finally, Specialized Algorithms (SAs), unlike the indirect strategy of problem transformation and algorithm adaptation, match directly the LDL problem. SA-IIS and SA-BFGS are two specialized algorithms that learn by optimizing an energy function based on the maximum entropy model.

Further studies have succeeded in improving the results obtained by these original algorithms using different strategies. The methods LDLogitBoost and AOSO-LDLogitBoost proposed in [53], are a combination of the boosting method and the logistic regression applied to LDL model. Deep label distribution learning (DLDL) [24] and Label Distribution Learning Based on ensemble neural networks [60] are two good examples of success applying neural networks on LDL. Inspired by differentiable decision trees [34], an end-to-end strategy LDL forests proposed in [45] which served as the basis for Structured Random Forest (StructRF) [10]. BC-LDL [50] and DBC-LDL [51] use the binary coding techniques to deal with the large-scale LDL problem. Classification with LDL (LDL4C) [49] is another interesting proposition

when learned label distribution model is generally treated as a classification model. Feature selection on LDL [31, 52] shows promising results by applying selection of characteristics on label distribution problems.

## 2.2. Decomposition Strategies for Classification

Decomposition strategies for addressing multi-class problems have been widely studied in the literature [20]. The same underlying idea is behind all proposals for decomposition: to solve a multi-class problem using binary classifiers. Following the "divide and rule" paradigm, the problem of multiple classes is divided into simpler binary classification problems. However, this method needs an additional step because of the simplification of the base classifiers: their outputs must be recombined to obtain the final result. How this aggregation is carried out is crucial to the quality of the final result. An exhaustive comparison of decomposition strategies and aggregation methods can be found in [20].

The most common decomposition strategies are One-vs-All (OVA) [42] and One-vs-One (OVO) [32], which can be included within Error Correcting Output Codes (ECOC) framework [14]. The first learns a binary classifier to discern between each pair of classes, while the second builds a binary classifier to separate each class from all the other classes.

The One-vs-All (OVA) [42] approach consists in dividing the multiple $c$ class problem into binary $c$ classification problems. Each binary classifier is faced up by a binary classifier which is responsible of distinguishing one of the classes from all other classes. The training phase of each classifier is carried out using the complete training set, considering as positive the samples of the single class and as negative all other samples. In the validation phase, the example is classified using each of the binary classifiers. The classifier that obtains a positive output will show the output class. Note that the output may not be unique and in these cases some sort of tiebreaker mechanism must be used. For example, we can calculate the confidence of each classifier to decide the final output by predicting the class from the classifier with the highest confidence.

The One-vs-One (OVO) [32] decomposition scheme divides a problem of $c$ classes into $c(c-1)/2$ binary problems. Each problem is addressed by a binary classifier that distinguishes between the different pairs of classes. The learning phase of each classifier is carried out using a subset of instances containing one of the two output classes. Instances with a different class are ignored.

In the prediction phase, the sample to be validated is predicted by each of the classifiers trained previously, thus obtaining a score matrix $R$:

$$R = \begin{pmatrix} - & r_{12} & r_{13} & ... & r_{1c} \\ r_{21} & - & r_{23} & ... & r_{2c} \\ . & . & . & . & . \\ . & . & . & . & . \\ r_{c1} & r_{c2} & r_{c3} & ... & - \end{pmatrix}$$

The output of a classifier given by $r_{ji} \in [0, 1]$ is the confidence of the binary classifier discriminating classes $i$ and $j$ in favor of the previous class. The confidence

of the classifier for $j$ is calculated as $r_{ji} = 1 - r_{ij}$ if the classifier does not provide it (the class with the larger confidence is the selected output class of a classifier).

The final prediction is calculated based on the score matrix using different aggregation models. The weighted voting strategy is the most used strategy, where confidences are aggregated class by class (by rows) and the one with the highest sum is selected as output. There is a disadvantage, known as the "non-competent classifier problem" [19] in the OVO system. The classifiers in the OVO system are not sufficiently competent to classify all the classes in the problem, as they are only learned through examples of two classes. However, all binary classifiers will be triggered for a given test model, because the competence cannot be known a priori, which can lead to incorrect decisions. To mitigate this problem there exist dynamic selection techniques that are able to distinguish between competent sorters directly in the prediction phase. The studies carried out in [11] and [23] provide a review of the most popular dynamic selection techniques to avoid the non-competent classifiers problem. On their side, [21, 63] are some examples of how to successfully apply the OVO technique.

ECOC [14] provides a suitable matrix framework for modelling the decomposition of a multi-class classification problem into simpler sub-problems. How to perform the decomposition to fit better the data using a small number of classifiers has been a key point of the research, as well as the decoding step, which deals with the combination of the sub-problems. The research [35] proposes an evidential unified framework that handles both the coding and decoding steps.

## 2.3. Motivation for using decomposition strategies on LDL

LDL is a generalization of the classification model and as such is exposed to the same problems as classic classification algorithms: imbalanced datasets, noisy data, overlapping or irregularities. OVO and OVA strategies have proven to be efficient in dealing with these kinds of difficulties. In [5, 17] we can find a complete review of how to apply decomposition strategies to imbalanced datasets that solve the disproportion of the number of examples of the different classes. The results obtained in [25, 44] show that using the OVO strategy lead to better performances and more robust classifiers when dealing with noisy data, especially with the most disruptive noise schemes. Decomposition performed by OVO in [43] helps to increase the separation between classes, creating more regular decision boundaries where there are overlapping samples. Other irregularities, such as the problem of the "difficult classes", have also been successfully addressed using the OVO decomposition strategy [22].

Applying a decomposition technique to an LDL-type data set, comparing output labels and breaking relationships between non-working pairs, could lead to better performance of the base LDL algorithms due to the proven improvement achieved by these techniques when they have been applied on datasets presenting the above problems. However, it should be recalled at this point that the LDL paradigm differs significantly from a multi-class problem, what we have is an output label distribution. Therefore, the OVO strategy would not serve us as it is. On the one hand, we need a new decomposition strategy capable of dealing with real values

instead of labels and on the other hand, a new aggregation strategy capable of providing an output according to the LDL constraints. As base learner we can use any of the LDL-specific learners.

# 3. DF-LDL: Decomposition-Fusion for Label Distribution Learning

DF-LDL is our decomposition proposal for LDL type of problems. The main idea of this method is to decompose the original $c$-label problem by dividing it into $c(c-1)$ problems, that is, comparing one to one all the labels that make up the output as we will explain in Section 3.1. The next step is to combine the outputs to obtain the final prediction, for this, we have conceived the competent fusion method which will be exposed in Section 3.2. Finally, we will illustrate how it works in Section 3.3.

## 3.1. Decomposition strategy

As opposed to a multi-class classification problem that would decompose the problem into $c(c-1)/2$ problems (one for each possible class pair), in LDL we need a decomposition mechanism able to carry out a comparison between the real values of each label. The strategy consists in making an inequality comparison ($a \geq b$) between each pair of output labels, creating subsets of samples containing only the samples whose label value is greater than the value of the compared label. The goal is to create subsets that break up relationships between pairs of labels that do not work. In this way, we will obtain the double of classifiers that in the original OVO strategy (because we are comparing real values and not grouping them into pairs of classes), that is $c(c-1)$ subsets on which we will start the training phase obtaining a matrix of learners like:

$$L = \begin{pmatrix} - & l_{12} & l_{13} & ... & l_{1c} \\ l_{21} & - & l_{23} & ... & l_{2c} \\ l_{31} & l_{32} & - & ... & l_{3c} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ l_{c1} & l_{c2} & l_{c3} & ... & - \end{pmatrix}$$

The detailed process is summarized in Algorithm 5.

Starting from the total set of training samples $S$ (containing $m$ samples in total), we classify each sample $(x_k, D_k)$ into the subsets $SS1$ or $SS2$ depending on whether the value of label $i$ (represented by $d_{x_k}^{y_i}$) is greater or lesser than the value of label $j$ (represented by $d_{x_k}^{y_j}$). Therefore, the subset $SS1$ will contain the samples that maximize the value of the output label $i$ with respect to the output label $j$, $SS2$ is consequently the complementary set to $SS1$. Each subset is trained separately using the base classifier $l$. This process is repeated $c(c-1)/2$ times obtaining as result the $L$ learner matrix that will contain a total of $c(c-1)$ learners. Note that the base learner $l$ can be any LDL-compatible learning algorithm.

**Algorithm 5:** Training stage of DF-LDL

**Function** `Fit`(*S, m, c, l*)**:**

**input :** $S \leftarrow$ Training Dataset ;
$m \leftarrow$ Number of Samples ;
$c \leftarrow$ Number of Labels ;
$l \leftarrow$ Base Learner ;
**output:** $L \leftarrow$ Matrix of learners ;
**for** *i=1 to c* **do**
  **for** *j=i+1 to c* **do**
    #Create the subsets of samples
    $SS1 = \emptyset$ ;
    $SS2 = \emptyset$ ;
    **for** *k=1 to m* **do**
      **if** $d_{x_k}^{y_i} \geq d_{x_k}^{y_j}$ **then**
        Append: $(x_k, D_k)$ to $SS1$ ;
      **else**
        Append: $(x_k, D_k)$ to $SS2$ ;
      **end**
    **end**
    #Fit the base learners using the subsets
    $l$.setTrainingSet(SS1);
    $l$.fit();
    $L_{ij} = l$;
    $l$.setTrainingSet(SS2);
    $l$.fit();
    $L_{ji} = l$;
  **end**
**end**
**End Function**

## 3.2. Competent Fusion method

In the prediction phase, the sample to be validated is usually predicted by each one of the classifiers trained previously but, as we have previously stated, our aim has been to design a dynamic competent aggregation or fusion method that allows us to build the final solution using only the learners that can minimize the error. We have been inspired by the dynamic classifier selection for One-vs-One strategy proposed in [23] that tries to avoid the non-competent classifiers when their output is probably not of interest. We consider an initial prediction of each instance to decide whether a classifier may be competent or not. Obviously, this initial prediction must be very run-time efficient in order to not penalize the total prediction time. For these reasons the technique chosen for this initial prediction is the AA-$k$NN [26] that obtains a prediction with an appropriate quality/time trade-off.

The competent fusion procedure summarized in Algorithm 6 works as follow: (1) Predict the example $x_0$ using AA-$k$NN and obtaining the initial prediction $p_{knn}$. Given the instance $x_0$, its $k$ nearest neighbors are first found in the training set. Then, the mean of the label distributions of all the $k$ nearest neighbors is calculated as the label distribution of $x_0$, i.e.,

$$p_{knn}(y_j|x_0) = \frac{1}{k} \sum_{i \in N_k(x_0)} d_{x_i}^{y_j}, (j = 1, 2, ..., c),$$

where $N_k(x_0)$ is the index set of the $k$ nearest neighbors of $x_0$ in the training set. (2) Use this initial prediction to compare one by one the output values of the labels. If the value of the label $i$, $p_{knn}^{y_i}$, is greater or equal than the value of the label $j$, $p_{knn}^{y_j}$, then we mark as selected the learner $L_{ij}$, otherwise we select the opposite learner $Lji$. (3) Predict the label distribution of the example $x_0$ using the selected learner in previous step and add this prediction to the final output $p$. (4) Repeat points 2 and 3 for each pair of labels. (5) The final label distribution $p$ is calculated as the average of predictions obtained by all selected learners:

$$p(y_j|x_0) = \frac{1}{c(c-1)/2} \sum_{ij \in L(x_0)} p_{l_{ij}}^{y_j}, (j = 1, 2, ..., c),$$

where $L(x_0)$ is the index set of selected learners for $x_0$. In total $c(c-1)/2$ learners will participate in the prediction.

## 3.3. An illustrative example

In order to show how the proposal works, let us take the toy dataset $S = \{(x_1, D_1), (x_2, D_2), (x_3, D_3)\}$ illustrated below containing $m = 3$ samples with $q = 2$ features and $c = 3$ output labels:

$$S = \begin{Bmatrix} x_1 & D_1 \\ x_2 & D_2 \\ x_3 & D_3 \end{Bmatrix} = \begin{Bmatrix} (0.4, 0.7) & (0.1, 0.85, 0.05) \\ (0.01, 0.9) & (0.75, 0.25, 0) \\ (0.5, 0.5) & (0.05, 0.05, 0.9) \end{Bmatrix}.$$

Following the process described in Section 3.1 we need to obtain a 3x3 matrix of learners. In order to build the first learner $l_{12}$, we will compare the output label

**Algorithm 6:** Predict an example in DF-LDL

**Function** `Predict(`$S$, $c$, $L$, $x_0$, $k$`):`

**input :** $S \leftarrow$ Training Dataset ;
$c \leftarrow$ Number of Labels ;
$L \leftarrow$ Matrix of learners ;
$x_0 \leftarrow$ Example to predict ;
**output:** $p \leftarrow$ Prediction ;
#Initial prediction using AA-kNN
AA-$k$NN.setTrainingSet(S);
$p_{knn}$ = AA-$k$NN.predict($x_0$);
#Select learners and predict
$p = 0$;
**for** *i=1 to c* **do**
  **for** *j=i+1 to c* **do**
    **if** $p_{knn}^{y_i} \geq p_{knn}^{y_j}$ **then**
      $p = p + L_{ij}.predict(x_0)$;
    **else**
      $p = p + L_{ji}.predict(x_0)$;
    **end**
  **end**
**end**
**return** $p/(c * (c-1)/2)$ ;

**End Function**

1 and 2 of each sample. In our case:

$$\begin{pmatrix} d_1^1 & d_1^2 \\ d_2^1 & d_2^2 \\ d_3^1 & d_3^2 \end{pmatrix} = \begin{pmatrix} 0.1 & 0.85 \\ 0.75 & 0.25 \\ 0.05 & 0.05 \end{pmatrix}.$$

As $d_2^1 \geq d_2^2$ and $d_3^1 \geq d_3^2$, the subset of samples used for learner $l_{12}$ will be $(x_2, D_2), (x_3, D_3)$. Therefore the subset of samples for learner $l_{21}$ will be $(x_1, D_1), (x_3, D_3)$.

Repeating this procedure for each pairs of labels we obtain the learner matrix below, for each learner we are showing the subset of samples selected:

$$L = \begin{pmatrix} - & \{(x_2, D_2), (x_3, D_3)\} & \{(x_1, D_1), (x_2, D_2)\} \\ \{(x_1, D_1), (x_3, D_3)\} & - & \{(x_1, D_1), (x_2, D_2)\} \\ \{(x_3, D_3)\} & \{(x_3, D_3)\} & - \end{pmatrix}$$

Once we have trained each of the classifiers using the subsets generated, we will predict the distribution of labels of a test instance, $x_0 = (0.2, 0.8)$, according to the method explained in the Section 3.2. Let us remember that the process consisted of five steps: (1) we predicted the output values through a AA-$k$NN algorithm using the whole training set. The $k$-NN prediction obtained for $x_0$ is:

$$p_{knn} = (0.3, 0.38, 0.32)$$

Then, in step (2), we use that prediction to discard incompetent learners by comparing each pair of output labels. In order to check if the first learner $l_{12}$ will be part of the final prediction, we will compare the output label 1 and 2. Value 0.3 is not greater or equal to 0.38, then learner $l_{12}$ will be discarded and, consequently, the opposite one $l_{21}$ will be selected.

In step (3), $x_0$ will be predicted by the selected learner, adding this prediction to the final output $p_{x_0}$. Repeating this process for each pair of labels, step (4), the learners that will be considered to obtain the final prediction are:

$$\begin{pmatrix} - & \cancel{l_{12}} & \cancel{l_{13}} \\ \mathbf{l_{21}} & - & \mathbf{l_{23}} \\ \mathbf{l_{31}} & \cancel{l_{32}} & - \end{pmatrix}$$

Finally, as described in step (5), the final label distribution $p_{x_0}$ is divided by 3 (the total of learners that have been involved in building the solution). Assuming that the values obtained by each of the learners are as follows:

$$p_{l_{21}} = (0.2, 0.7, 0.1)$$

$$p_{l_{23}} = (0.4, 0.6, 0)$$

$$p_{l_{31}} = (0.25, 0.5, 0.25)$$

The final label distribution $p_{x_0}$ will be:

$$p_{x_0} = (0.28, 0.6, 0.12)$$

# 4. Experimental Framework

This section is devoted to introducing the experimental framework used in the different empirical studies of the paper. In our experiments, we have included 17 datasets of a wide variety of real-world problems, described in the following Section 4.1.

In order to evaluate the learners proficiency, we will employ six measures (described in Section 4.2) for the different aspects of their performance. For each dataset and learner, these measures were computed over a merged set from the test predictions of a 10-fold cross validation set (10-fcv).

We have run two different kinds of experiments. The first one consists in comparing the results provided by the base learners with the DF-LDL method using the same base learner. The selected base learners have been SA-BFGS [26] and Structured tree (StructTree), this second one has been extracted from the Structured random forest (StructRF) [10] proposal but by training a single tree. The second experiment directly compares the DF-LDL proposal with other LDL state-of-the-art algorithms, such as: SA-BFGS, label distribution learning forests (LDLFs) [45] and StructRF.

The non-parametric statistical Wilcoxon, Friedman rank and Bayesian Sign tests [13, 4] are used to validate the results of the empirical comparisons. In the Bayesian Sign test, a distribution of the differences of the results achieved using methods $L$ (base learner in first experiment) and $R$ (DF-LDL) is computed into a graphical space divided in 3 regions: left, rope and right. The location of most of the distribution in these sectors indicates the final decision: the superiority of algorithm $L$, statistical equivalence and the superiority of algorithm $R$, respectively. KEEL package [46] has been used to compute the Wilcoxon and Friedman rank tests and the R package rNPBST [7] was used to extract the graphical representations of the Bayesian Sign tests analyzed in the following empirical studies. The Rope limit parameter used to represent the Bayesian Sign test is 0.0001.

## 4.1. Datasets

There are a total of 17 real-world datasets employed in the experiments. The summary of their characteristics is shown in Table 6.1.

The first 11 datasets are originally LDL problems. To complete the experiment, we have also added 4 multi-class classification datasets and 2 multi-target regression datasets, with a double purpose: to see how LDL learners behave on other types of classification problems and because purely LDL datasets are still scarce. Note that non-LDL data sets have been adapted to satisfy LDL restrictions. This transformation is described below along with the description of each set.

The first six datasets have been collected from biological experiments on the budding yeast Saccharomyces cerevisiae [16]. It includes 2,465 yeast genes, and an associated phylogenetic profile vector with a length of 24 is utilized to represent each gene. In a biological experiment, the gene expression level is usually disparate at each discrete time point, so the labels correspond to the time point.

Datasets JAFFE [37] and BU_3DFE [57] are two widely used facial expression

Table 6.1: Datasets used in experiments

| No. | Datasets | Examples($m$) | Features($q$) | Labels($c$) | Type |
|---|---|---|---|---|---|
| 1 | Yeast_alpha | 2465 | 24 | 18 | LDL |
| 2 | Yeast_cdc | 2465 | 24 | 15 | LDL |
| 3 | Yeast_diau | 2465 | 24 | 7 | LDL |
| 4 | Yeast_elu | 2465 | 24 | 14 | LDL |
| 5 | Yeast_heat | 2465 | 24 | 6 | LDL |
| 6 | Yeast_spo | 2465 | 24 | 6 | LDL |
| 7 | SJAFFE | 213 | 243 | 6 | LDL |
| 8 | SBU_3DFE | 2500 | 243 | 6 | LDL |
| 9 | Movie | 7755 | 1869 | 5 | LDL |
| 10 | Natural_Scene | 2000 | 294 | 9 | LDL |
| 11 | Human_Gene | 30542 | 36 | 68 | LDL |
| 12 | Optdigits | 5620 | 64 | 10 | multi-class |
| 13 | Semeion | 1593 | 256 | 10 | multi-class |
| 14 | Ecoli | 327 | 7 | 5 | multi-class |
| 15 | LED7digit | 500 | 7 | 10 | multi-class |
| 16 | Wq | 1060 | 16 | 14 | multi-target |
| 17 | Jura | 359 | 15 | 3 | multi-target |

image datasets. There are 213 gray-scale expression images in the JAFFE dataset while BU_3DFE contains 2,500 facial expression images. The images in JAFFE have been scored by 60 people using the six primary emotion labels with a 5-level scale, i.e., fear, disgust, happiness, anger, sadness, surprise, and the images in BU_3DFE have been scored by 23 people using the same scale as used in JAFFE. Each dataset is represented by a 243-dimensional feature vector extracted using the Local Binary Patterns method (LBP) [2] . The score for each emotion is regarded as the description degree, and the description degrees (normalized gene expression level) of all the six emotions constitute a label distribution for a particular facial expression image.

Dataset Movie includes 7,755 movies. There is a total of 54,243,292 ratings from 478,656 different users on a scale from 1 to 5 integral stars from ®Netflix. The percentage of each rating level is regarded as the label distribution. There are numeric and categorical attributes in the dataset such as genre, director, country, year, budget and so on. After transforming the categorical attributes into binary vectors, the final feature vector of each movie is 1,869-dimensional.

The Natural Scene dataset is collected from 2,000 natural scene images that have been ranked inconsistently by ten human rankers. A 294- dimensional feature vector extracted in [6] represents each image and is associated with a multi-label selected from 9 possible labels, i.e., sun, sky, water, cloud, mountain, snow, desert, building, and plant. Then rankers are required to rank the relevant labels in descending order of relevance. As expected, the rankings for the same image from different rankers are highly inconsistent. So, a nonlinear programming process [28] is applied to achieve the label distribution.

The Human Gene dataset is much larger than the other datasets used in this

experiment. This dataset is collected from biological research on the relationship between human genes and diseases. Each of the 30,542 human genes is represented by the 36 numerical descriptors for a gene sequence proposed in [58].

The following 4 datasets correspond to standard multi-class classification problems, all of them extracted from UCI Machine Learning Repository [15]. Therefore we need a pre-processing step to transform them into LDL type datasets. Since we only have information from the class, we need to transform the classifier scores into accurate multiclass probability estimates following the process described in [59]. A base clasifier is fit on the training set of the cross-validation generator and the test set is used for calibration. The probabilities for each of the folds are then averaged for prediction. We have made use of 4 diferent base classifiers: SVC [9], $k$-NN [1], a decision tree classifier [36] and Gaussian Naive Bayes classifier [61].

The selected multi-class datasets are: Optidigits (Optical Recognition of Handwritten Digits), Semeion (Semeion Handwritten Digit Data Set, where 1593 handwritten digits from around 80 persons were scanned and documented), Ecoli, an E.Coli bacteria protein classification and LED7digit, a simple domain containing 7 boolean attributes, one for each light-emitting diode of a 7-segment display.

Jura and Wq are two multi-target regression datasets collected from the Mulan website [48]. The Jura dataset consists of measurements of concentrations of seven heavy metals (cadmium, cobalt, chromium, copper, nickel, lead, and zinc), recorded at 359 locations in the topsoil of a region of the Swiss Jura. We are interested in the distribution prediction of the concentration of metals that are more expensive to measure (primary variables) using measurements of metals that are cheaper to sample (secondary variables). The Water Quality dataset has 14 target attributes that refer to the relative representation of plant and animal species in Slovenian rivers and 16 input attributes that refer to physical and chemical water quality parameters. LDL algorithms can deal directly with multi-target datasets, the only prerequisite is to normalize the output vector in such a way that the sum is equal to 1.

## 4.2. Evaluation Measure Selection

We use a set of six measures when comparing different LDL algorithms: Chebyshev Distance, Clark Distance, Canberra Metric and Kullback-Leibler Divergence for distance calculation; Cosine Coefficient and Intersection for Similarity as shown in Table 6.2. Each of these measures come from a different syntactic family summarized in [8] and are relatively widely used in the related areas. Thus, they can adequately represent the different aspects of the algorithms.

## 4.3. Parameters

The DF-LDL proposal itself is parameter-free, but the AA-$k$NN algorithm used in the competent fusion method requires a value for $k$, set to 4 neighbors (we have selected a small value for $k$ in order to provide the most flexible fit, with a low bias). For the SA-BFGS algorithm, the convergence criterion $\epsilon$ has been setup to $10^{-5}$. Regarding the parameters used for the StructRF and LDLFs algorithms we

Table 6.2: Evaluation measure for LDL learners. ↓ *means that the lowest value is the best and* ↑ *means the opposite.*

| Name | Formula |
|---|---|
| Chebyshev(Cheby)↓ | $Dis(D,\hat{D}) = max_j\lvert d_j - \hat{d}_j\rvert$ |
| Clark↓ | $Dis(D,\hat{D}) = \sqrt{\sum_{j=1}^{c} \frac{(d_j-\hat{d}_j)^2}{(d_j+\hat{d}_j)^2}}$ |
| Canberra(Can)↓ | $Dis(D,\hat{D}) = \sum_{j=1}^{c} \frac{\lvert d_j-\hat{d}_j\rvert}{d_j+\hat{d}_j}$ |
| Kullback-Leibler(KL)↓ | $Dis(D,\hat{D}) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$ |
| Cosine(Cos)↑ | $Sim(D,\hat{D}) = \frac{\sum_{j=1}^{c} d_j\hat{d}_j}{\sqrt{\sum_{j=1}^{c} d_j^2}\sqrt{\sum_{j=1}^{c} \hat{d}_j^2}}$ |
| Intersection(Inter)↑ | $Sim(D,\hat{D}) = \sum_{j=1}^{c} min(d_j,\hat{d}_j)$ |

have kept the same values as those used in the original proposal papers, for LDLFs the maximum number of iterations has been adapted to meet the time constraints. Since StructTree starts from the same base as StructRF, the parameters used are exactly the same except that we use a single classification tree. An overview of all these parameters can be found in Table 6.3.

Table 6.3: Summary of the parameters

| Algorithm | Parameter | Description | Value |
|---|---|---|---|
| AA-$k$NN | $k$ | Number of selected neighbors | 4 |
| SA-BFGS | $\epsilon$ | Convergence criterion: must be less than $\epsilon$ before successful termination | $10^{-5}$ |
| LDLFs | trees | Number of trees | 5 |
| | depth | Maximum depth of the tree | 7 |
| | out. feat. | Output unit number of the feature learning function | 64 |
| | iters. leaf | Iteration times to update leaf node predictions | 20 |
| | batches | Number of mini-batches to update leaf node predictions | 100 |
| | iters | Maximum iterations | 2500 |
| StructTree | max. depth | Maximum depth of the tree | 20 |
| | min. leaf | Minimum size of the leaf | 5 |
| StructRF | trees | Number of trees | 50 |
| | sampling | Sampling ratio of data | 0.8 |
| | max. depth | Maximum depth of the tree | 20 |
| | min. leaf | Minimum size of the leaf | 5 |

# 5. Results and Analysis

This section presents the results of the empirical studies and their analyses. We have differentiated between two types of experiments: first we will compare the results obtained by the base learners with the results obtained by DF-LDL using the same base learner (Section 5.1), then we will compare our DF-LDL proposal with

other state-of-the-art LDL algorithms, showing their different strengths (Section 5.2).

In each of the result tables the best outcome is highlighted in bold. The last column is the mean of each row. The best average is also highlighted in bold. As those algorithms have been tested using 10-fcv, the performance is represented using "mean±standard deviation".

## 5.1. Evaluation of DF-LDL vs. Base Learners

Comparing DF-LDL with the base classifiers SA-BFGS and StructTree we reach the following conclusions:

- The results of the different measures shown in Table 6.4 and Table 6.5 highlights the best ranking of DF-LDL in the large majority of the datasets and measures.
- The Wilcoxon Signed Ranks test corroborate the significance of the differences between our approach and the base learners. Table 6.6 includes the outcome of the Wilcoxon test comparing DF-LDL with the base learners. All the hypotheses of equivalence are rejected with small p-values.
- With regard to the Bayesian Sign test, Figure 6.1 and Figure 6.2 graphically represent the difference between using DF-LDL or directly the base learner and its statistical significance in terms of precision. The following heat-maps clearly indicate the significant superiority of DF-LDL, as the computed distributions are always located in the right region.

Special mention about the excellent results obtained with our proposal for the SJAFFE dataset, for which we obtained an outstanding improvement over the base algorithms.

Nevertheless the outcomes obtained for the Clark and Canberra measures on the StructTree method deserve special attention. We note that the values obtained when using these two measures differ greatly from those obtained for the others. To understand this peculiarity we must look previously at the label distribution of each dataset, when this distribution has many zeros in its composition and we train this set with the StructTree algorithm the probability of obtaining label values equal to zero increases, falsifying the total computation of the distances. Therefore these measures are not representative of the quality of the algorithm for the specific case of the StructTree method and the datasets that present this peculiarity.

Table 6.4: Experimental Results DF-LDL (using StructTree as base learner) vs. StructTree (mean ± std)

| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Yeast_alpha | Yeast_cdc | Yeast_diau | Yeast_elu | Yeast_heat | Yeast_spo |
| Cheby↓ | StructTree | 0.0168±0.0005 | 0.0204±0.0009 | 0.0459±0.0015 | 0.0201±0.0007 | 0.0518±0.0008 | 0.0734±0.0032 |
| | DF-LDL | **0.0133±0.0008** | **0.0162±0.0009** | **0.0367±0.0013** | **0.0161±0.0005** | **0.0418±0.0008** | **0.0609±0.0024** |
| Clark↓ | StructTree | 0.2615±0.0082 | 0.2729±0.0123 | 0.2474±0.0076 | 0.2476±0.0082 | 0.2239±0.0035 | 0.3078±0.0135 |
| | DF-LDL | **0.2084±0.0125** | **0.2158±0.0134** | **0.1990±0.0062** | **0.1977±0.0053** | **0.1813±0.0028** | **0.2575±0.0124** |
| Can↓ | StructTree | 0.8506±0.0298 | 0.8216±0.0380 | 0.5321±0.0181 | 0.7302±0.0221 | 0.4481±0.0058 | 0.6369±0.0280 |
| | DF-LDL | **0.6772±0.043** | **0.6472±0.0406** | **0.4279±0.0155** | **0.5804±0.0153** | **0.3629±0.0057** | **0.5295±0.0251** |
| KL↓ | StructTree | 0.0086±0.0006 | 0.0113±0.0009 | 0.0209±0.0015 | 0.0087 ±0.0004 | 0.0200±0.0004 | 0.0310±0.0030 |
| | DF-LDL | **0.0054±0.0006** | **0.0070±0.0009** | **0.0132±0.0010** | **0.0061±0.0004** | **0.0126±0.0005** | **0.0270±0.0023** |
| Cos↑ | StructTree | 0.9916±0.0006 | 0.9894±0.0008 | 0.9811±0.0013 | 0.9907±0.0006 | 0.9817±0.0005 | 0.9627±0.0029 |
| | DF-LDL | **0.9947±0.0006** | **0.9934±0.0008** | **0.9878±0.0009** | **0.9941±0.0003** | **0.9880±0.0005** | **0.9746±0.0019** |
| Inter↑ | StructTree | 0.9530±0.0016 | 0.9460±0.0025 | 0.9262±0.0026 | 0.9485±0.0015 | 0.9265±0.0009 | 0.8948±0.0043 |
| | DF-LDL | **0.9626±0.0024** | **0.9574±0.0026** | **0.9406±0.0023** | **0.9591±0.0011** | **0.9405±0.0010** | **0.9126±0.0038** |

| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | SJAFFE | SBU_3DFE | Movie | Natural_Scene | Human_Gene | Optdigits |
| Cheby↓ | StructTree | 0.1392±0.0263 | 0.1520±0.0062 | 0.1294±0.0053 | 0.3581±0.0216 | 0.0828±0.0143 | 0.1214±0.0073 |
| | DF-LDL | **0.1044±0.0160** | **0.1198±0.0064** | **0.1123±0.0050** | **0.2672±0.0117** | **0.0499±0.0039** | **0.0874±0.0040** |
| Clark↓ | StructTree | 0.4759±0.0691 | 0.4515±0.0160 | 0.5649±0.0222 | **1.6344±0.0350** | 2.5795±0.1732 | **0.7966±0.0170** |
| | DF-LDL | **0.3792±0.0527** | **0.3783±0.0188** | **0.5081±0.0267** | 2.3719±0.0221 | **2.0755±0.0433** | 1.0088±0.0199 |
| Can↓ | StructTree | 0.9775±0.1426 | 0.9084±0.0314 | 1.0954±0.0456 | **3.7282±0.1147** | 17.9475±1.3568 | **1.7797±0.0344** |
| | DF-LDL | **0.7839±0.1088** | **0.7881±0.0378** | **0.9737±0.0515** | 6.3743±0.0934 | **14.0834±0.7432** | 2.5748±0.0573 |
| KL↓ | StructTree | 0.1084±0.0330 | 0.1157±0.0080 | 0.1322±0.0098 | 2.1534±0.1533 | 0.3896±0.0646 | 0.3063±0.0255 |
| | DF-LDL | **0.0589±0.0145** | **0.0705±0.0064** | **0.0961±0.0072** | **0.6300±0.0265** | **0.2289±0.0239** | **0.0827±0.0053** |
| Cos↑ | StructTree | 0.8973±0.0311 | 0.8857±0.0075 | 0.9154±0.0063 | 0.6487±0.0222 | 0.7342±0.0374 | 0.9166±0.0086 |
| | DF-LDL | **0.9438±0.0139** | **0.9298±0.0065** | **0.9367±0.0043** | **0.7889±0.0098** | **0.8497±0.0167** | **0.9837±0.0021** |
| Inter↑ | StructTree | 0.8254±0.0286 | 0.8268±0.0065 | 0.8157±0.0075 | 0.5409±0.0203 | 0.7143±0.0267 | 0.8698±0.0073 |
| | DF-LDL | **0.8660±0.0192** | **0.8568±0.0072** | **0.8397±0.0069** | **0.6061±0.0102** | **0.8009 ±0.0123** | **0.9035±0.0042** |

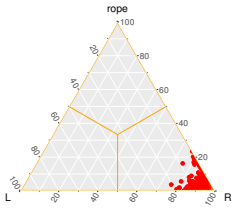
| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Semeion | Ecoli | LED7digit | Wq | Jura | **Average** |
| Cheby↓ | StructTree | 0.2355±0.0193 | 0.0953±0.0378 | 0.0568±0.0141 | 0.3465±0.0230 | 0.0886±0.0162 | 0.1196±0.0117 |
| | DF-LDL | **0.1799±0.0079** | **0.0749±0.0272** | **0.0503±0.0087** | **0.2840±0.0171** | **0.0791±0.0144** | **0.0938±0.0076** |
| Clark↓ | StructTree | **1.0923±0.0463** | 0.4596±0.0879 | **0.3028±0.0577** | **2.6063±0.0596** | 0.2998±0.0353 | **0.7544±0.0396** |
| | DF-LDL | 1.2406±0.0286 | **0.4245±0.069** | 0.3040±0.0477 | 3.1031±0.0425 | **0.2768±0.0255** | 0.7842±0.0264 |
| Can↓ | StructTree | **2.5888±0.1278** | 0.7740±0.1648 | **0.6971±0.1468** | **7.6999±0.2775** | 0.4280±0.0529 | **2.5085±0.1551** |
| | DF-LDL | 3.3518±0.0929 | **0.7230±0.1348** | 0.7093±0.1240 | 10.6209±0.2191 | **0.3829±0.0399** | 2.6230±0.1087 |
| KL↓ | StructTree | 0.5342±0.0542 | 0.1022±0.0692 | 0.0555±0.0272 | 1.0220±0.0500 | 0.0336±0.0101 | 0.2973±0.0301 |
| | DF-LDL | **0.1756±0.0125** | **0.0502±0.0336** | **0.0305±0.0075** | **0.9255±0.0413** | **0.0266±0.0063** | **0.1439±0.0112** |
| Cos↑ | StructTree | 0.8091±0.0177 | 0.9572±0.0331 | 0.9760±0.0109 | 0.5178±0.0230 | 0.9809±0.0070 | 0.8904±0.0124 |
| | DF-LDL | **0.9468±0.0077** | **0.9785±0.0181** | **0.9844±0.0052** | **0.6672±0.0091** | **0.9848±0.0053** | **0.9369±0.0061** |
| Inter↑ | StructTree | 0.7344±0.0206 | 0.9013±0.0403 | 0.9343±0.0157 | 0.4081±0.0180 | 0.9114±0.0162 | 0.8281±0.0130 |
| | DF-LDL | **0.7948±0.0095** | **0.9220±0.0292** | **0.9423±0.0102** | **0.4828±0.0134** | **0.9209±0.0144** | **0.8593±0.0088** |

Table 6.5: Experimental Results DF-LDL (using SA-BFGS as base learner) vs. SA-BFGS (mean ± std)

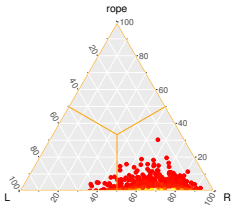
| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Yeast_alpha | Yeast_cdc | Yeast_diau | Yeast_elu | Yeast_heat | Yeast_spo |
| Cheby↓ | SA-BFGS | 0.0135±0.0008 | 0.0163±0.0009 | 0.0370±0.0014 | **0.0163±0.0006** | 0.0423±0.0008 | **0.0584±0.0037** |
| | DF-LDL | **0.0134±0.0008** | **0.0162±0.0009** | **0.0369±0.0012** | **0.0163±0.0005** | **0.0421±0.0007** | 0.0603±0.0028 |
| Clark↓ | SA-BFGS | 0.2107±0.0126 | **0.2165±0.0129** | 0.2008±0.0079 | 0.1992±0.0055 | 0.1828±0.0027 | **0.2500±0.0164** |
| | DF-LDL | **0.2103±0.0127** | 0.2166±0.0129 | **0.2001±0.0067** | **0.1989±0.0053** | **0.1819±0.0025** | 0.2552±0.0132 |
| Can↓ | SA-BFGS | 0.6847±0.0432 | **0.6493±0.0394** | 0.4310±0.0186 | 0.5838±0.0159 | 0.3647±0.0058 | **0.5137±0.0335** |
| | DF-LDL | **0.6832±0.0435** | 0.6501±0.0393 | **0.4298±0.0166** | **0.5831±0.0157** | **0.3631±0.0057** | 0.5246±0.0269 |
| KL↓ | SA-BFGS | **0.0055±0.0006** | **0.0070±0.0008** | **0.0131±0.0011** | **0.0062±0.0004** | **0.0126±0.0004** | **0.0246±0.0029** |
| | DF-LDL | **0.0055±0.0006** | **0.0070±0.0008** | **0.0131±0.0011** | **0.0062±0.0004** | **0.0126±0.0004** | 0.0260±0.0025 |
| Cos↑ | SA-BFGS | **0.9946±0.0006** | **0.9933±0.0007** | **0.9879±0.0010** | **0.9940±0.0004** | 0.9880±0.0004 | **0.9769±0.0026** |
| | DF-LDL | **0.9946±0.0006** | **0.9933±0.0007** | **0.9879±0.0010** | **0.9940±0.0004** | **0.9881±0.0004** | 0.9755±0.0021 |
| Inter↑ | SA-BFGS | 0.9622±0.0024 | **0.9573±0.0026** | 0.9403±0.0027 | 0.9588±0.0011 | 0.9401±0.0010 | **0.9154±0.0053** |
| | DF-LDL | **0.9623±0.0024** | 0.9572±0.0025 | **0.9404±0.0024** | **0.9589±0.0011** | **0.9404±0.0010** | 0.9135±0.0041 |

| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | SJAFFE | SBU_3DFE | Movie | Natural_Scene | Human_Gene | Optdigits |
| Cheby↓ | SA-BFGS | 0.1603±0.0160 | **0.1100±0.0039** | 0.1398±0.0161 | 0.3549±0.0159 | 0.0533±0.0036 | 0.0744±0.0032 |
| | DF-LDL | **0.1007±0.0126** | 0.1110±0.0004 | **0.1309±0.0101** | **0.3359±0.0159** | **0.0523±0.0041** | **0.0675±0.0028** |
| Clark↓ | SA-BFGS | 0.6466±0.0462 | 0.3784±0.0122 | 0.6035±0.0560 | 2.3817±0.0239 | 2.1111±0.0820 | 1.6751±0.0167 |
| | DF-LDL | **0.3787±0.0416** | **0.3667±0.0106** | **0.5635±0.0390** | **2.3824±0.0239** | **2.0906±0.0711** | **1.4124±0.0176** |
| Can↓ | SA-BFGS | 1.3595±0.1100 | 0.7831±0.0246 | 1.1679±0.1163 | 6.5546±0.0914 | 14.4531±0.6124 | 4.4333±0.0553 |
| | DF-LDL | **0.7776±0.0798** | **0.7593±0.0186** | **1.0846±0.0785** | **6.5532±0.0914** | **14.3439±0.7324** | **3.7289±0.0544** |
| KL↓ | SA-BFGS | 0.1639±0.0245 | 0.0634±0.0038 | 0.1543±0.0405 | 1.1120±0.0999 | 0.2365±0.0184 | 0.0896±0.0048 |
| | DF-LDL | **0.0583±0.0117** | **0.0622±0.0035** | **0.1266±0.0203** | **1.0047±0.0999** | **0.2301±0.0135** | **0.0702±0.0034** |
| Cos↑ | SA-BFGS | 0.8739±0.0179 | **0.9389±0.0034** | 0.9072±0.0188 | **0.6724±0.0149** | 0.8343±0.0102 | 0.9831±0.0019 |
| | DF-LDL | **0.9454±0.0111** | 0.9385±0.0035 | **0.9173±0.0120** | 0.6711±0.0149 | **0.8399±0.0099** | **0.9873±0.0016** |
| Inter↑ | SA-BFGS | 0.7788±0.0162 | 0.8616±0.0041 | 0.8040±0.0199 | **0.5248±0.0145** | 0.7842±0.0092 | 0.9142±0.0035 |
| | DF-LDL | **0.8682±0.0131** | **0.8637±0.0036** | **0.8163±0.0140** | **0.5248±0.0145** | **0.7898±0.0081** | **0.9223±0.0031** |

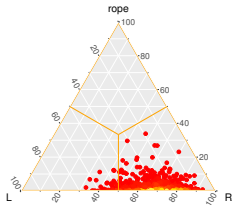
| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Semeion | Ecoli | LED7digit | Wq | Jura | **Average** |
| Cheby↓ | SA-BFGS | **0.1368±0.0084** | 0.0875±0.0188 | 0.0981±0.0086 | 0.3026±0.0223 | 0.0732±0.0046 | 0.1044±0.0076 |
| | DF-LDL | 0.1462±0.0062 | **0.0753±0.0156** | **0.0754±0.0084** | **0.3006±0.0216** | **0.0728±0.0057** | **0.0973±0.0067** |
| Clark↓ | SA-BFGS | 1.4282±0.0208 | 0.8979±0.0439 | 1.2670±0.0419 | **3.1255±0.0427** | 0.2592±0.0248 | 0.9432±0.0276 |
| | DF-LDL | **1.2428±0.0185** | **0.7059±0.0372** | **0.8589±0.0280** | 3.1257±0.0422 | **0.2540±0.0243** | **0.8614±0.0240** |
| Can↓ | SA-BFGS | 3.7601±0.0579 | 1.6400±0.0938 | 3.2747±0.0991 | 10.7824±0.2188 | 0.3674±0.0308 | 3.0473±0.0980 |
| | DF-LDL | **3.2911±0.0616** | **1.2878±0.0804** | **2.2037±0.0652** | **10.7794±0.2164** | **0.3616±0.0303** | **2.8474±0.0975** |
| KL↓ | SA-BFGS | 0.1652±0.0132 | 0.0730±0.0254 | 0.0932±0.0145 | 1.0540±0.0714 | 0.0231±0.0035 | 0.1940±0.0192 |
| | DF-LDL | **0.1460±0.0093** | **0.0466±0.0124** | **0.0538±0.0108** | **1.0266±0.0604** | **0.0226±0.0036** | **0.1717±0.0150** |
| Cos↑ | SA-BFGS | 0.9512±0.0078 | 0.9847±0.0078 | 0.9732±0.0084 | 0.6248±0.0134 | **0.9875±0.0023** | 0.9215±0.0066 |
| | DF-LDL | **0.9574±0.0053** | **0.9889±0.0055** | **0.9806±0.0064** | **0.6300±0.0128** | **0.9875±0.0026** | **0.9281±0.0053** |
| Inter↑ | SA-BFGS | **0.8330±0.0095** | 0.9073±0.0199 | 0.8737±0.0117 | 0.4431±0.0170 | **0.9628±0.0046** | 0.8448±0.0085 |
| | DF-LDL | 0.8270±0.0067 | **0.9211±0.0162** | **0.9041±0.0102** | **0.4490±0.0166** | 0.9272±0.0057 | **0.8521±0.0074** |



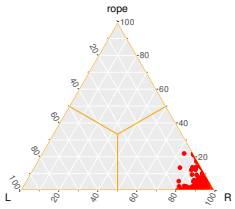

(a) Chebyshev Distance



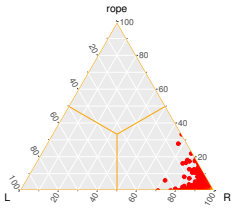

(b) Clark Distance



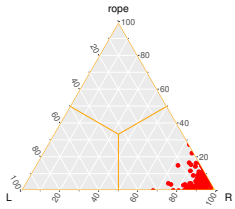

(c) Canberra Metric




(d) Kullback-Leibler Divergence

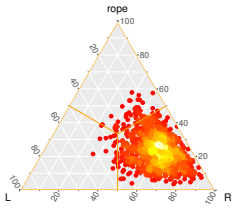



(e) Cosine Coefficient

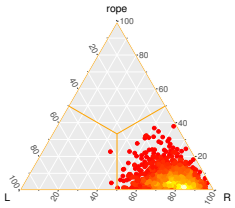



(f) Intersection Similarity

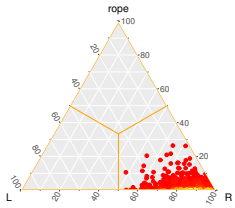
Figure 6.1: Bayesian Sign test comparing StructTree(L) vs. DF-LDL(R)



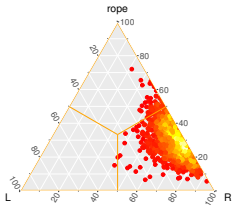

(a) Chebyshev Distance



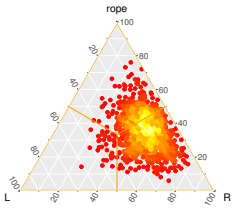

(b) Clark Distance



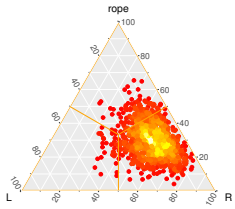

(c) Canberra Metric




(d) Kullback-Leibler Divergence




(e) Cosine Coefficient




(f) Intersection Similarity

Figure 6.2: Bayesian Sign test comparing BFGS(L) vs. DF-LDL(R)

Table 6.6: Wilcoxon Signed Ranks test comparing DF-LDL vs. Base Learners

| | DF-LDL vs. StructTree | | | DF-LDL vs. SA-BFGS | | |
|---|---|---|---|---|---|---|
| Measure | $R^+$ | $R^-$ | $p$-value | $R^+$ | $R^-$ | $p$-value |
| Cheby | 153 | 0 | $1.5258 \times 10^{-5}$ | 109.5 | 26.5 | 0.0313 |
| Clark | 93 | 60 | $\geq 0.2$ | 135.5 | 17.5 | 0.0035 |
| Can | 94 | 59 | $\geq 0.2$ | 142 | 11 | 0.0001 |
| KL | 153 | 0 | $1.5258 \times 10^{-5}$ | 124 | 12 | 0.0021 |
| Cos | 153 | 0 | $1.5258 \times 10^{-5}$ | 110 | 26 | 0.0290 |
| Inter | 153 | 0 | $1.5258 \times 10^{-5}$ | 102.5 | 33.5 | 0.0786 |

## 5.2. Evaluation of DF-LDL vs. LDL state-of-the-art algorithms

We will now see how DF-LDL behaves compared to three other state-of-the-art algorithms as SA-BFGS, LDLFs and StructRF. Note than the selected base learner used by DF-LDL for this comparison is StructTree. We reach the following conclusions:

- The results of the different measures shown in Table 6.7 highlight the best average score of DF-LDL in all cases.
- Table 6.8 includes the outcome of the Friedman rank and Holm tests in relation to the obtained results over the computed measures. DF-LDL is ranked first compared to SA-BFGS, LDLFs and StructRF. Although DF-LDL has a better ranking than StructRF, both yield very competitive results.
- Note also that the results vary considerably depending on the dataset used. LDLFs proves to be very effective on the Yeast datasets but decreases in strength on the other sets. In this experiment, DF-LDL uses the same base algorithm than StructRF, so when the number of learners used by DF-LDL is much lower than the used by StructRF, this last one provides a better accuracy (remember that the number of learners is related to the number of output labels in DF-LDL while it remains a fix value in the case of StructRF).

Another measure that is of interest to compare is the execution time. At this point we will focus only on the two proposals that have achieved the best results. In our experiment, DF-LDL uses the same base algorithm than StructRF, Struct Tree. Hence the computational complexity of the training phase is similar for both methods: $O(m^2 \times q \times trees)$ in StructRF and $O(m^2 \times q \times c(c-1))$ in DF-LDL. If $c(c-1)$ is less than the number of trees used in StructRF, our proposal DF-LDL will be faster to train and vice versa. As for the prediction, the computational complexity for StructRF is $O(q \times trees)$. For DF-LDL it is also necessary to add the time of the previous prediction AA-KNN, resulting in a complexity of $O(q \times c(c-1)/2 + mq)$. We have two factors that can make one algorithm more efficient predicting than the other. On the one hand, if $c(c-1)/2$ is less than the number of trees used in

StructRF, our proposal DF-LDL will use fewer prediction trees and vice versa. But in the case of DF-LDL it is also necessary to add the time of the previous prediction AA-KNN.

We can see an overview of the runtimes on each dataset in Table 6.9 that corroborate this fact, obtaining prediction times that improve StructRF in the vast majority of cases.

Table 6.7: Experimental Results DF-LDL vs. LDL state-of-the-art algorithms (mean ± std)

| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Yeast_alpha | Yeast_cdc | Yeast_diau | Yeast_elu | Yeast_heat | Yeast_spo |
| Cheby↓ | SA-BFGS | 0.0135±0.0008 | 0.0163±0.0009 | 0.0370±0.0014 | 0.0163±0.0006 | 0.0423±0.0008 | 0.0584±0.0037 |
| | LDLFs | **0.0129±0.0003** | **0.0159±0.0007** | **0.0344±0.0017** | **0.0156±0.0005** | **0.0392±0.0011** | **0.0561±0.0020** |
| | StructRF | 0.0134±0.0008 | 0.0162±0.0009 | 0.0359±0.0015 | 0.0161±0.0005 | 0.0407±0.0009 | 0.0578±0.0030 |
| | DF-LDL | 0.0133±0.0008 | 0.0162±0.0009 | 0.0367±0.0013 | 0.0161±0.0005 | 0.0418±0.0008 | 0.0609±0.0024 |
| Clark↓ | SA-BFGS | 0.2107±0.0126 | 0.2165±0.0129 | 0.2008±0.0079 | 0.1992±0.0055 | 0.1828±0.0027 | 0.2500±0.0164 |
| | LDLFs | **0.2021±0.0031** | **0.2126±0.0065** | **0.1872±0.0112** | **0.1908±0.0052** | **0.1706±0.0040** | **0.2427±0.0082** |
| | StructRF | 0.2095±0.0125 | 0.2158±0.0134 | 0.1947±0.0076 | 0.1974±0.0054 | 0.1770±0.0028 | 0.2467±0.0139 |
| | DF-LDL | 0.2084±0.0125 | 0.2158±0.0134 | 0.1990±0.0062 | 0.1977±0.0053 | 0.1813±0.0028 | 0.2575±0.0124 |
| Can↓ | SA-BFGS | 0.6847±0.0432 | 0.6493±0.0394 | 0.4310±0.0186 | 0.5838±0.0159 | 0.3647±0.0058 | 0.5137±0.0335 |
| | LDLFs | **0.6555±0.0099** | **0.6385±0.0146** | **0.4011±0.0221** | **0.5595±0.0148** | **0.3408±0.0076** | **0.4969±0.0173** |
| | StructRF | 0.6812±0.0434 | 0.6472±0.0410 | 0.4177±0.0178 | 0.5780±0.0156 | 0.3541±0.0063 | 0.5066±0.0274 |
| | DF-LDL | 0.6772±0.043 | 0.6472±0.0406 | 0.4279±0.0155 | 0.5804±0.0153 | 0.3629±0.0057 | 0.5295±0.0251 |
| KL↓ | SA-BFGS | 0.0055±0.0006 | 0.0070±0.0008 | 0.0131±0.0011 | 0.0062±0.0004 | 0.0126±0.0004 | 0.0246±0.0029 |
| | LDLFs | **0.0051±0.0002** | **0.0067±0.0004** | **0.0115±0.0013** | **0.0057±0.0003** | **0.0111±0.0005** | **0.0231±0.0016** |
| | StructRF | 0.0055±0.0006 | 0.0070±0.0009 | 0.0125±0.0010 | 0.0061±0.0004 | 0.0120±0.0004 | 0.0243±0.0024 |
| | DF-LDL | 0.0054±0.0006 | 0.0070±0.0009 | 0.0132±0.0010 | 0.0061±0.0004 | 0.0126±0.0005 | 0.0270±0.0023 |
| Cos↑ | SA-BFGS | 0.9946±0.0006 | 0.9933±0.0007 | 0.9879±0.0010 | 0.9940±0.0004 | 0.9880±0.0004 | 0.9769±0.0026 |
| | LDLFs | **0.9950±0.0001** | **0.9935±0.0003** | **0.9895±0.0011** | **0.9945±0.0003** | **0.9894±0.0004** | **0.9786±0.0014** |
| | StructRF | 0.9946±0.0006 | 0.9933±0.0008 | 0.9885±0.0010 | 0.9941±0.0003 | 0.9886±0.0004 | 0.9773±0.0021 |
| | DF-LDL | 0.9947±0.0006 | 0.9934±0.0008 | 0.9878±0.0009 | 0.9941±0.0003 | 0.9880±0.0005 | 0.9746±0.0019 |
| Inter↑ | SA-BFGS | 0.9622±0.0024 | 0.9573±0.0026 | 0.9403±0.0027 | 0.9588±0.0011 | 0.9401±0.0010 | 0.9154±0.0053 |
| | LDLFs | **0.9638±0.0006** | **0.9580±0.0009** | **0.9445±0.0028** | **0.9606±0.0010** | **0.9441±0.0012** | **0.9186±0.0028** |
| | StructRF | 0.9624±0.0024 | 0.9574±0.0027 | 0.9421±0.0026 | 0.9592±0.0011 | 0.9419±0.0011 | 0.9167±0.0042 |
| | DF-LDL | 0.9626±0.0024 | 0.9574±0.0026 | 0.9406±0.0023 | 0.9591±0.0011 | 0.9405±0.0010 | 0.9126±0.0038 |

| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | SJAFFE | SBU_3DFE | Movie | Natural_Scene | Human_Gene | Optdigits |
| Cheby↓ | SA-BFGS | 0.1603±0.0160 | **0.1100±0.0039** | 0.1398±0.0161 | 0.3549±0.0159 | 0.0533±0.0036 | **0.0744±0.0032** |
| | LDLFs | 0.1158±0.0153 | 0.1302±0.0046 | 0.1170±0.0033 | 0.3604±0.0260 | 0.0531±0.0159 | 0.6270±0.0129 |
| | StructRF | 0.1094±0.0108 | 0.1183±0.0058 | **0.1104±0.0048** | 0.2738±0.0116 | 0.0553±0.0048 | 0.1034±0.0048 |
| | DF-LDL | **0.1044±0.0160** | 0.1198±0.0064 | 0.1123±0.0050 | **0.2672±0.0117** | **0.0499±0.0039** | 0.0874±0.0040 |
| Clark↓ | SA-BFGS | 0.6466±0.0462 | 0.3784±0.0122 | 0.6035±0.0560 | 2.3817±0.0239 | 2.1111±0.0820 | 1.6751±0.0167 |
| | LDLFs | 0.4175±0.0361 | 0.4011±0.0110 | 0.5269±0.0123 | 2.4841±0.0300 | 2.1240±0.0637 | 2.5616±0.0242 |
| | StructRF | 0.3900±0.0350 | **0.3680±0.0173** | **0.5039±0.0233** | 2.3946±0.0227 | 2.1776±0.1232 | 1.0620±0.0225 |
| | DF-LDL | **0.3792±0.0527** | 0.3783±0.0188 | 0.5081±0.0267 | **2.3719±0.0221** | **2.0755±0.0433** | **1.0088±0.0199** |
| Can↓ | SA-BFGS | 1.3595±0.1100 | 0.7831±0.0246 | 1.1679±0.1163 | 6.5546±0.0914 | 14.4531±0.6124 | 4.4333±0.0553 |
| | LDLFs | 0.8797±0.0804 | 0.8610±0.0253 | 0.9995±0.0233 | 6.8694±0.1319 | 14.5737±0.4372 | 7.8649±0.0948 |
| | StructRF | 0.8089±0.0769 | **0.7817±0.0354** | **0.9595±0.0441** | 6.4559±0.0950 | 14.8633±0.8857 | 2.7195±0.0638 |
| | DF-LDL | **0.7839±0.1088** | 0.7881±0.0378 | 0.9737±0.0515 | **6.3743±0.0934** | **14.0834±0.7432** | **2.5748±0.0573** |
| KL↓ | SA-BFGS | 0.1639±0.0245 | 0.0634±0.0038 | 0.1543±0.0405 | 1.1120±0.0999 | 0.2365±0.0184 | 0.0896±0.0048 |
| | LDLFs | 0.0707±0.0132 | 0.0765±0.0045 | 0.0988±0.0043 | 0.9453±0.0569 | 0.2398±0.0719 | 1.0301±0.0446 |
| | StructRF | 0.0603±0.0089 | **0.0659±0.0054** | **0.0901±0.0061** | 0.6436±0.0211 | 0.2480±0.0265 | 0.1056±0.0081 |
| | DF-LDL | **0.0589±0.0145** | 0.0705±0.0064 | 0.0961±0.0072 | **0.6300±0.0265** | **0.2289±0.0239** | **0.0827±0.0053** |
| Cos↑ | SA-BFGS | 0.8739±0.0179 | **0.9389±0.0034** | 0.9072±0.0188 | 0.6724±0.0149 | 0.8343±0.0102 | 0.9831±0.0019 |
| | LDLFs | 0.9334±0.0125 | 0.9249±0.0040 | 0.9347±0.0028 | 0.6653±0.0185 | 0.8353±0.0251 | 0.6879±0.0193 |
| | StructRF | 0.9429±0.0083 | 0.9348±0.0055 | **0.9407±0.0039** | **0.7895±0.0091** | 0.8202±0.0205 | 0.9758±0.0035 |
| | DF-LDL | **0.9438±0.0139** | 0.9298±0.0065 | 0.9367±0.0043 | 0.7889±0.0098 | **0.8497±0.0167** | **0.9837±0.0021** |
| Inter↑ | SA-BFGS | 0.7788±0.0162 | **0.8616±0.0041** | 0.8040±0.0199 | 0.5248±0.0145 | 0.7842±0.0092 | **0.9142±0.0035** |
| | LDLFs | 0.8501±0.0151 | 0.8450±0.0046 | 0.8350±0.0040 | 0.4605±0.0220 | 0.7833±0.0235 | 0.3619±0.0131 |
| | StructRF | 0.8622±0.0132 | 0.8592±0.0065 | **0.8432±0.0060** | 0.5919±0.0097 | 0.7739±0.0159 | 0.8871±0.0050 |
| | DF-LDL | **0.8660±0.0192** | 0.8568±0.0072 | 0.8397±0.0069 | **0.6061±0.0102** | **0.8009±0.0123** | 0.9035±0.0042 |

| Measure | Method | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Semeion | Ecoli | LED7digit | Wq | Jura | **Average** |
| Cheby↓ | SA-BFGS | **0.1368±0.0084** | 0.0875±0.0188 | 0.0981±0.0086 | 0.3026±0.0223 | 0.0732±0.0046 | 0.1044±0.0076 |
| | LDLFs | 0.3569±0.0459 | 0.4888±0.0329 | 0.4719±0.0461 | 0.3065±0.0116 | 0.1115±0.0168 | 0.1949±0.0140 |
| | StructRF | 0.2099±0.0042 | **0.0716±0.0240** | 0.0532±0.0114 | 0.2859±0.0185 | **0.0689±0.0090** | 0.0965±0,0069 |
| | DF-LDL | 0.1799±0.0079 | 0.0749±0.0272 | **0.0503±0.0087** | **0.2840±0.0171** | 0.0791±0.0144 | **0.0938±0.0076** |
| Clark↓ | SA-BFGS | 1.4282±0.0208 | 0.8979±0.0439 | 1.2670±0.0419 | 3.1255±0.0427 | 0.2592±0.0248 | 0.9432±0.0276 |
| | LDLFs | 1.6710±0.0889 | 1.5454±0.0502 | 1.9121±0.1527 | 3.1115±0.0295 | 0.3429±0.0348 | 1.0767±0.0336 |
| | StructRF | 1.3193±0.0197 | 0.4650±0.0748 | 0.3293±0.0591 | **3.1006±0.0443** | **0.2416±0.0216** | 0.7996±0.0305 |
| | DF-LDL | **1.2406±0.0286** | **0.4245±0.069** | **0.3040±0.0477** | 3.1031±0.0425 | 0.2768±0.0255 | **0.7842±0.0264** |
| Can↓ | SA-BFGS | 3.7601±0.0579 | 1.6400±0.0938 | 3.2747±0.0991 | 10.7824±0.2188 | 0.3674±0.0308 | 3.0473±0.0980 |
| | LDLFs | 4.7213±0.3042 | 3.2622±0.1264 | 5.5698±0.5812 | 10.7510±0.1516 | 0.5034±0.0588 | 3.5264±0.1236 |
| | StructRF | 3.5802±0.0661 | 0.7990±0.1509 | 0.7832±0.156 | 10.6299±0.2312 | **0.3406±0.0309** | 2.7004±0.1169 |
| | DF-LDL | **3.3518±0.0929** | **0.7230±0.1348** | **0.7093±0.1240** | **10.6209±0.2191** | 0.3829±0.0399 | **2.6230±0.1087** |
| KL↓ | SA-BFGS | **0.1652±0.0132** | 0.0730±0.0254 | 0.0932±0.0145 | 1.0540±0.0714 | 0.0231±0.0035 | 0.1940±0.0192 |
| | LDLFs | 0.4456±0.0874 | 0.7172±0.0748 | 0.7356±0.0945 | 1.1061±0.0372 | 0.0435±0.0099 | 0.3278±0.0296 |
| | StructRF | 0.2135±0.0095 | **0.0448±0.0267** | 0.0326±0.0095 | 0.9391±0.0406 | **0.0206±0.0031** | 0.1489±0.0101 |
| | DF-LDL | 0.1756±0.0125 | 0.0502±0.0336 | **0.0305±0.0075** | **0.9255±0.0413** | 0.0266±0.0063 | **0.1439±0.0112** |
| Cos↑ | SA-BFGS | **0.9512±0.0078** | **0.9847±0.0078** | 0.9732±0.0084 | 0.6248±0.0134 | 0.9875±0.0023 | 0.9215±0.0066 |
| | LDLFs | 0.8451±0.0406 | 0.6986±0.0341 | 0.6735±0.0359 | 0.5887±0.0104 | 0.9744±0.0076 | 0.8648±0.0126 |
| | StructRF | 0.9316±0.0065 | 0.9831±0.0141 | 0.9835±0.0053 | 0.6634±0.0101 | **0.9880±0.0033** | 0.9347±0.0056 |
| | DF-LDL | 0.9468±0.0077 | 0.9785±0.0181 | **0.9844±0.0052** | **0.6672±0.0091** | 0.9848±0.0053 | **0.9369±0.0061** |
| Inter↑ | SA-BFGS | **0.8330±0.0095** | 0.9073±0.0199 | 0.8737±0.0117 | 0.4431±0.0170 | **0.9628±0.0046** | 0.8448±0.0085 |
| | LDLFs | 0.6098±0.0480 | 0.4876±0.0315 | 0.4854±0.0451 | 0.3999±0.0118 | 0.8885±0.0168 | 0.7469±0.0144 |
| | StructRF | 0.7656±0.0062 | **0.9251±0.0026** | 0.9389±0.0132 | 0.4745±0.0149 | 0.9311±0.0090 | 0.8548±0.0082 |
| | DF-LDL | 0.7948±0.0095 | 0.9220±0.0292 | **0.9423±0.0102** | **0.4828±0.0134** | 0.9209±0.0144 | **0.8593±0.0088** |

Table 6.8: Friedman rank and Holm test applied to the results amoung the tested algorithms

| **Measure** | **Control Method:** DF-LDL (2.1176) | | | |
|---|---|---|---|---|
| | **i** | **Algorithm (Rank)** | **Z** | ***p*-Value** |
| Cheby | 3 | SA-BFGS (3) | 1.9926 | 0.0463 |
| | 2 | LDLFs (2.7059) | 1.3284 | 0.1840 |
| | 1 | StructRF (2.1765) | 0.1328 | 0.8943 |
| **Measure** | **Control Method:** DF-LDL (1.9706) | | | |
| | **i** | **Algorithm (Rank)** | **Z** | ***p*-Value** |
| Can | 3 | SA-BFGS (3.2941) | 2.9889 | 0.0028 |
| | 2 | LDLFs (2.7059) | 1.6605 | 0.0968 |
| | 1 | StructRF (2.0294) | 0.1328 | 0.8943 |
| **Measure** | **Control Method:** DF-LDL (2.2353) | | | |
| | **i** | **Algorithm (Rank)** | **Z** | ***p*-Value** |
| Cos | 3 | SA-BFGS (2.7941) | 1.2620 | 0.2069 |
| | 2 | LDLFs (2.7059) | 1.0627 | 0.2879 |
| | 1 | StructRF (2.2647) | 0.0664 | 0.9470 |

| **Measure** | **Control Method:** DF-LDL (1.9706) | | | |
|---|---|---|---|---|
| | **i** | **Algorithm (Rank)** | **Z** | ***p*-Value** |
| Clark | 3 | SA-BFGS (3.2941) | 2.9889 | 0.0028 |
| | 2 | LDLFs (2.7059) | 1.6605 | 0.0968 |
| | 1 | StructRF (2.0294) | 0.1328 | 0.8943 |
| **Measure** | **Control Method:** DF-LDL (2.1765) | | | |
| | **i** | **Algorithm (Rank)** | **Z** | ***p*-Value** |
| KL | 3 | SA-BFGS (2.8824) | 1.5941 | 0.1109 |
| | 2 | LDLFs (2.7059) | 1.1956 | 0.2319 |
| | 1 | StructRF (2.2353) | 0.1328 | 0.8943 |
| **Measure** | **Control Method:** DF-LDL (2.1471) | | | |
| | **i** | **Algorithm (Rank)** | **Z** | ***p*-Value** |
| Inter | 3 | SA-BFGS (2.8824) | 1.6605 | 0.0968 |
| | 2 | LDLFs (2.7647) | 1.3948 | 0.1631 |
| | 1 | StructRF (2.2059) | 0.1328 | 0.8943 |

Table 6.9: Execution times (in seconds), measured on a machine with ®Intel Core i7-7300HQ processor (4 cores, 6MB cache, 2.5GHz - 3.5GHz) and 16GB DDR4 2400MHz RAM

| Datasets | Fit time | | Prediction time | |
|---|---|---|---|---|
| | StructRF | DF-LDL | StructRF | DF-LDL |
| Yeast_alpha | **574.35** | 4709.50 | **0.10** | 0.52 |
| Yeast_cdc | **736.34** | 2196.00 | **0,12** | 0,29 |
| Yeast_diau | 717.97 | **421.10** | 0.11 | **0,09** |
| Yeast_elu | **712.89** | 1365.01 | **0.11** | 0,24 |
| Yeast_heat | 710.97 | **203.28** | 0.10 | **0.06** |
| Yeast_spo | 522.11 | **201.99** | 0.10 | **0.06** |
| SJAFFE | 143.77 | **86.00** | 0.05 | **0.01** |
| SBU_3DFE | 2272.50 | **1370.02** | 0.07 | **0.04** |
| Movie | 10413.56 | **4160.43** | 0.39 | **0.20** |
| Natural_Scene | **2866.39** | 4122.04 | 0.08 | **0.06** |
| Human_Gene | **4317.67** | 390,365.00 | **0.78** | 35.34 |
| Optdigits | **1432.00** | 2335.03 | 0.24 | **0.22** |
| Semeion | **630.05** | 1161.71 | 0.06 | **0.05** |
| Ecoli | 42.28 | **11.52** | 0.008 | **0.006** |
| LED7digit | **2866.38** | 5058.16 | 0.08 | **0.07** |
| Wq | **306.38** | 717.15 | **0.03** | 0.06 |
| Jura | 54.53 | **4.23** | 0.007 | **0.001** |

# 6. Conclusion

In this paper, we proposed a novel decomposition strategy that adapts to LDL constraints. We have based the design of this algorithm on techniques like OVO that have already demonstrated their potential. The DF-LDL algorithm can use any of the already existing LDL learners as base to build a stronger classifier. In addition, the developed fusion method allows us to combine the outputs in a way that discards the less competent classifiers.

In order to verify the effectiveness of the solution designed, it has been compared, firstly, with the base learners where we have demonstrated a clear superiority of DF-LDL over practically all the datasets and measures used, and secondly, with the state-of-the-art learner in the LDL scope where DF-LDL achieves improvements in many of the cases.

We also want to highlight the performance improvement obtained in prediction times with respect to other multiple learning approaches thanks to the fusion method devised that only makes use of the most competent classifiers for each case.

As future work we want to extend the current proposal with some ideas that have emerged during the course of this study. We can anticipate some of them such as the following:

- DF-LDL can be considered an LDL-oriented framework, compatible with any LDL learning algorithm. In this study, we have experimented with two different LDL base learners but in future work we would like to complete the analysis with further ones. For instance, the LDLogitBoost [53] and the Label Distribution Learning Based on ensemble neural networks[60] are two ensemble proposals from which we can extract the underlying base classifier and use it in our DF-LDL framework.

- DF-LDL needs to train a large number of learners, especially when the number of output labels is high. An interesting approach could be to design a decomposition strategy based on ECOC [14] in order to perform the decomposition to fit better the data using a small number of classifiers.

# Acknowledgements

This work is supported by the Spanish National Research Project TIN2017-89517-P.

# Compliance with ethical standards

## Conflict of interest

The authors declare that they have no conflict of interest.

# Bibliography


[1] D.W. Aha, D. Kibler, and M.K. Albert, "Instance-based learning algorithms", *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[2] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[3] Z. Barutcuoglu, R.E. Schapire, and O.G. Troyanskaya, "Hierarchical multi-label prediction of gene function", *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[4] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[5] J. Bi and C. Zhang, "An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme", *Knowledge-Based Systems*, vol. 158, pp. 81–93, 2018.

[6] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown, "Learning multi-label scene classification", *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[7] J. Carrasco, S. García, M. del Mar Rueda, and F. Herrera, "rnpbst: An R package covering non-parametric and bayesian statistical tests", in *International Conference on Hybrid Artificial Intelligence Systems*, Springer, 2017, pp. 281–292.

[8] S.H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions", *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 2, p. 1, 2007.

[9] C.C. Chang and C.J. Lin, "LIBSVM: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[10] M. Chen, X. Wang, B. Feng, and W. Liu, "Structured random forest for label distribution learning", *Neurocomputing*, vol. 320, pp. 171–182, 2018.

[11] R.M.O. Cruz, R. Sabourin, and G.D.C. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives", *Information Fusion*, vol. 41, pp. 195–216, 2018.

[12] S. Das, S. Datta, and B.B. Chaudhuri, "Handling data irregularities in classification: Foundations, trends, and future challenges", *Pattern Recognition*, vol. 81, pp. 674–693, 2018.

[13] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[14] T.G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1994.

[15] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[16] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns", *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[17] A. Fernández, S. García, M. Galar, R.C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018.

[18] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[19] J. Fürnkranz, E. Hüllermeier, and S. Vanderlooy, "Binary decomposition methods for multipartite ranking", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp. 359–374.

[20] 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 Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[21] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "NMC: nearest matrix classification–A new combination model for pruning One-vs-One ensembles by transforming the aggregation problem", *Information Fusion*, vol. 36, pp. 26–51, 2017.

[22] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Empowering difficult classes with a similarity-based aggregation in multi-class classification problems", *Information Sciences*, vol. 264, pp. 135–157, 2014.

[23] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers", *Pattern Recognition*, vol. 46, no. 12, pp. 3412–3424, 2013.

[24] B.B. Gao, C. Xing, C.W. Xie, J. Wu, and X. Geng, "Deep label distribution learning with label ambiguity", *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2825–2838, 2017.

[25] L.P.F. Garcia, J. A Sáez, J. Luengo, A.C. Lorena, A.C.P.L.F. de Carvalho, and F. Herrera, "Using the One-vs-One decomposition to improve the performance of class noise filters via an aggregation strategy in multi-class classification problems", *Knowledge-Based Systems*, vol. 90, pp. 153–164, 2015.

[26] X. Geng, "Label distribution learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1734–1748, 2016.

[27] X. Geng and P. Hou, "Pre-release prediction of crowd opinion on movies by label distribution learning", in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, vol. 2015-January, 2015, pp. 3511–3517.

[28] X. Geng and L. Luo, "Multilabel ranking with inconsistent rankers", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3742–3747.

[29] X. Geng, C. Yin, and Z.H. Zhou, "Facial age estimation by learning from label distributions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2401–2412, 2013.

[30] E. Gibaja and S. Ventura, "A tutorial on multilabel learning", *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 1–38, 2015.

[31] M. González, J.R. Cano, and S. García, "ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning", *Applied Sciences*, vol. 10, no. 9, Art. 3089, 2020.

[32] T. Hastie and R. Tibshirani, "Classification by pairwise coupling", in *Advances in Neural Information Processing Systems*, 1998, pp. 507–513.

[33] F. Herrera, F. Charte, A.J. Rivera, and M.J. Del Jesus, *Multilabel classification: Problem analysis, metrics and techniques*. Springer, 2016.

[34] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1467–1475.

[35] M. Lachaize, S. Le Hégarat-Mascle, E. Aldea, A. Maitrot, and R. Reynaud, "Evidential framework for error correcting output code classification", *Engineering Applications of Artificial Intelligence*, vol. 73, pp. 10–21, 2018.

[36] R. Lior *et al.*, *Data mining with decision trees: theory and applications*. World scientific, 2014, vol. 81.

[37] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets", in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, IEEE, 1998, pp. 200–205.

[38] J.M. Moyano, E.L. Gibaja, K.J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: models, experimental study and prospects", *Information Fusion*, vol. 44, pp. 33–45, 2018.

[39] R.C. Prati, G.E.A.P.A. Batista, and M.C. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior", in *Mexican International Conference on Artificial Intelligence*, Springer, 2004, pp. 312–321.

[40] R.C. Prati, J. Luengo, and F. Herrera, "Emerging topics and challenges of learning from noisy data in nonstandard classification: a survey beyond binary class noise", *Knowledge and Information Systems*, vol. 60, no. 1, pp. 63–97, 2019.

[41] Y. Ren and X. Geng, "Sense Beauty by Label Distribution Learning.", in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 2648–2654.

[42] R. Rifkin and A. Klautau, "In defense of one-vs-all classification", *Journal of machine learning research*, vol. 5, no. Jan, pp. 101–141, 2004.

[43] J. A Sáez, M. Galar, and B. Krawczyk, "Addressing the Overlapping Data Problem in Classification Using the One-vs-One Decomposition Strategy", *IEEE Access*, vol. 7, pp. 83 396–83 411, 2019.

[44] J.A. Sáez, M. Galar, J. Luengo, and F. Herrera, "Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition", *Knowledge and Information Systems*, vol. 38, no. 1, pp. 179–206, 2014.

[45] W. Shen, K. Zhao, Y. Guo, and A.L. Yuille, "Label distribution learning forests", in *Advances in Neural Information Processing Systems*, 2017, pp. 834–843.

[46] I. Triguero, S. González, J.M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. Fernández Hilario, J. Díaz, L. Sánchez, F. Herrera, *et al.*, "KEEL 3.0: an open source software for multi-stage analysis in data mining", 2017.

[47] I. Triguero and C. Vens, "Labelling strategies for hierarchical multi-label classification techniques", *Pattern Recognition*, vol. 56, pp. 170–183, 2016.

[48] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning", *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2411–2414, 2011.

[49] J. Wang and X. Geng, "Classification with label distribution learning", in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3712–3718.

[50] K. Wang and X. Geng, "Binary Coding based Label Distribution Learning.", in *International Joint Conferences on Artificial Intelligence*, 2018, pp. 2783–2789.

[51] K. Wang and X. Geng, "Discrete binary coding based label distribution learning", in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3733–3739.

[52] Y. Wang and J. Dai, "Label Distribution Feature Selection Based on Mutual Information in Fuzzy Rough Set Theory", in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–2.

[53] C. Xing, X. Geng, and H. Xue, "Logistic boosting regression for label distribution learning", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4489–4497.

[54] L. Xu, J. Chen, and Y. Gan, "Head pose estimation using improved label distribution learning with fewer annotations", *Multimedia Tools and Applications*, pp. 1–22, 2019.

[55] D. Xue, Z. Hong, S. Guo, L. Gao, L. Wu, J. Zheng, and N. Zhao, "Personality recognition on social media with label distribution learning", *IEEE Access*, vol. 5, pp. 13 478–13 488, 2017.

[56] J. Yang, D. She, and M. Sun, "Joint Image Emotion Classification and Distribution Learning via Deep Convolutional Neural Network.", in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 3266–3272.

[57] L. Yin, X. Wei, Y. Sun, J. Wang, and M.J. Rosato, "A 3D facial expression database for facial behavior research", in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, IEEE, 2006, pp. 211–216.

[58] J.F. Yu, D.K. Jiang, K. Xiao, Y. Jin, J.H. Wang, and X. Sun, "Discriminate the falsely predicted protein-coding genes in Aeropyrum Pernix K1 genome based on graphical representation", *Match-Communications in Mathematical and Computer Chemistry*, vol. 67, no. 3, p. 845, 2012.

[59] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates", in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2002, pp. 694–699.

[60] Y. Zhai, J. Dai, and H. Shi, "Label Distribution Learning Based on Ensemble Neural Networks", in *International Conference on Neural Information Processing*, Springer, 2018, pp. 593–602.

[61] H. Zhang, "The optimality of Naive Bayes", in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, vol. 2, 2004, pp. 562–567.

[62] Z. Zhang, M. Wang, and X. Geng, "Crowd counting in public video surveillance by label distribution learning", *Neurocomputing*, vol. 166, pp. 151–163, 2015.

[63] Z.L. Zhang, X.G. Luo, Y. Yu, B.W. Yuan, and J.F. Tang, "Integration of an improved dynamic ensemble selection approach to enhance one-vs-one scheme", *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 43–53, 2018.

[64] X. Zheng, X. Jia, and W. Li, "Label distribution learning by exploiting sample correlations locally", in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018, pp. 4556–4563.

# Capítulo 7

# Análisis de los Resultados

En esta sección se resumen los análisis y resultados obtenidos en cada etapa de la tesis, en relación con las publicaciones que forman el compendio y que se han expuesto en la sección anterior.

Señalar que, para todos los experimentos llevados a cabo, hemos utilizado los conjuntos de datos descritos en la Sección 2.3.4 que conforman una buena variedad de problemas del mundo real.

En todos los casos, para evaluar la eficacia de cada método, hemos empleado seis medidas (descritas en la Sección 2.3.3) que reflejan diferentes aspectos sobre el desempeño de cada algoritmo.

Finalmente, los resultados de las pruebas empíricas han sido validados a través de los estudios estadísticos descritos para cada caso en la sección anterior.

## 7.1. Generación Sintética de Muestras para el Aprendizaje de la Distribución de Etiquetas

En este trabajo, propusimos un novedoso método de generación de muestras sintéticas que se adapta a los problemas de tipo LDL. La propuesta de SSG-LDL puede aplicarse, en una etapa previa de pre-procesamiento, a cualquier conjunto de entrenamiento y luego puede conectarse como una entrada sobre cualquier algoritmo de aprendizaje.

Para validar la eficacia de nuestro método aplicamos los algoritmos LDL ya existentes sobre el conjunto de entrenamiento preprocesado con SSG-LDL. Los algoritmos escogidos para este análisis son tres de los más significativos en el ámbito LDL, a saber, AA-$k$NN [49], SA-BFGS [49] y StructRF [23], descritos en la Sección 2.3.

Los resultados obtenidos sobre cada comparativa pueden resumirse de la siguiente manera:

- Evaluación de SSG-LDL + AA-$k$NN vs. AA-$k$NN: la comparativa entre AA-$k$NN y el mismo algoritmo pero usando un conjunto de datos previamente preprocesado usando SSG-LDL, destaca el mejor rendimiento del clasificador cuando se utiliza un conjunto de datos preprocesado. La mejora obtenida es notable en la cuasi totalidad de los conjuntos de datos utilizados.

- Evaluación de SSG-LDL + SA-BFGS vs. SA-BFGS: nos encontramos con un caso similar cuando sustituimos el clasificador por el llamado SA-BFGS, ahora bien, al ser este un clasificador más robusto las diferencias no son tan notables como en el caso anterior. De cualquier forma, a la vista de los resultados de los análisis estadísticos, existe una alta probabilidad de que los resultados obtenidos mejoren tras aplicar este método de pre-procesamiento.

- Evaluación de SSG-LDL + StructRF vs. StructRF: en este último punto quisimos comprobar cómo se comporta el algoritmo SSG-LDL cuando se combina con un método de aprendizaje tan robusto como StructRF. En cuanto al ranking general, vemos que la aplicación del pre-procesamiento mejora los resultados en el 40 % de los casos y empata en el 15 % de los casos. Si entramos en más detalle observamos que el comportamiento varía según el conjunto de datos: para el conjunto de datos *Human_Gene* por ejemplo, observamos que la aplicación de la generación de muestras sintéticas mejora significativamente los resultados originales aunque no altera sustancialmente los resultados para los conjuntos de datos *Yeast* y causa resultados ligeramente peores para los conjuntos relacionados con imágenes de expresiones faciales, películas o escenas naturales. Estadísticamente hablando, lo que se demuestra es que la aplicación del pre-procesamiento en el algoritmo StructRF puede ser ventajoso en un alto porcentaje de casos.

En resumen, al aplicar un método de pre-procesamiento de datos como el que se presenta aquí, SSG-LDL, se mejoran significativamente los resultados obtenidos por el algoritmo de aprendizaje. En el caso de AA-$k$NN, la aplicación previa de un tratamiento SSG-LDL hace que el clasificador se comporte de manera más eficiente en prácticamente todos los conjuntos de datos. Esto también ocurre cuando se realiza el mismo tipo de pre-procesamiento con un clasificador SA-BFGS, obteniendo una notable mejora en los resultados. Incluso cuando se utiliza un método tan robusto como StructRF, también es ventajoso aplicar la SSG-LDL ya que puede mejorar los resultados en un alto porcentaje de los experimentos.

## 7.2. Selección de Prototipos y de Características Específicas de las Etiquetas mediante Optimización Evolutiva para el Aprendizaje de la Distribución de Etiquetas

Para este estudio, propusimos un novedoso algoritmo de reducción de datos que se adapta a las restricciones de LDL. Aborda simultáneamente la selección de prototipos y la selección de características específicas a cada etiqueta de salida. Siendo dos los objetivos perseguidos: encontrar un subconjunto óptimo de muestras para mejorar el rendimiento del clasificador AA-$k$NN [49] y seleccionar un subconjunto de características específicas para cada una de las etiquetas de salida. Ambas tareas se han abordado como un problema de búsqueda utilizando un algoritmo evolutivo, basado en el CHC [35], para optimizar la solución.

El algoritmo ProLSFEO-LDL propuesto se aplica en primer lugar, en una etapa de pre-procesamiento, sobre los conjuntos de datos en bruto, obteniendo un subconjunto de prototipos y una matriz de características específicas para cada etiqueta de salida. El subconjunto de prototipos seleccionado conforma el nuevo conjunto de entrenamiento preprocesado que se utilizará como entrada del clasificador LDL. En cuanto a la selección de características, hemos adaptado el algoritmo AA-$k$NN para que cada etiqueta se pueda predecir por separado utilizando sólo las características marcadas como seleccionadas. La predicción obtenida se normaliza para hacerla compatible con las restricciones de LDL.

Los resultados obtenidos por las diferentes medidas de evaluación destacan el mejor resultado del clasificador cuando se utiliza la etapa de pre-procesamiento ProLSFEO-LDL. Resultado que corroboran los test estadísticos.

Respecto al porcentaje de los prototipos y características seleccionados con respecto al conjunto de entrenamiento inicial, medimos una reducción media alrededor del 53 % para ambos, prototipos y características. Este ratio de reducción de datos tendrá un enorme impacto en el rendimiento del algoritmo de aprendizaje. En los métodos basados en prototipos como AA-$k$NN, el hecho de utilizar cerca de la mitad de los prototipos y características, conduce a una reducción del tiempo de cálculo en la fase de predicción.

## 7.3. Descomposición-Fusión para el Aprendizaje de la Distribución de Etiquetas

En este estudio, propusimos una novedosa estrategia de descomposición adaptada a problemas de tipo LDL. Hemos basado el diseño de este algoritmo en técnicas como *One-vs-One* [64] que han demostrado su potencial en problemas de clasificación de tipo multi-clase. El algoritmo DF-LDL puede utilizar cualquiera de los algoritmos de aprendizaje LDL ya existentes como base para construir un clasificador más fuerte. Además, el método de fusión desarrollado nos permite combinar las salidas de manera que se descarten los clasificadores menos competentes.

Para verificar la eficacia de la solución diseñada, hemos realizado dos tipos de experimentos diferentes. El primero consiste en comparar los resultados proporcionados por los clasificadores base con el método DF-LDL utilizando el mismo clasificador base. Los algoritmos base seleccionados han sido SA-BFGS [49] y StructTree, este segundo se ha extraído de la propuesta Structured Random Forest (StructRF) [23] pero entrenando un solo árbol. El segundo experimento compara directamente la propuesta DF-LDL con otros algoritmos de última generación LDL, tales como: SA-BFGS, LDLFs [99] y StructRF.

Los resultados obtenidos por las diferentes medidas de evaluación resaltan los mejores resultados obtenidos por el método DF-LDL respecto a los algoritmos base SA-BFGS y StructTree. Resultados corroborados por los test estadísticos.

En cuanto al segundo experimento que compara DF-LDL con otros algoritmos "estado del arte", DF-LDL se clasifica en primer lugar respecto al resto de propuestas. También queremos destacar la mejora del rendimiento obtenido en los tiempos de predicción con respecto a otros enfoques gracias al método de fusión ideado que

sólo utiliza los clasificadores más competentes para cada caso.

DF-LDL puede considerarse como un framework orientado a LDL, compatible con cualquier algoritmo de aprendizaje de LDL. En este estudio, hemos experimentado con dos clasificadores LDL diferentes como métodos base, pero estos pueden ser sustituidos por cualquier otro clasificador LDL de una forma fácil.

# Capítulo 8

# Conclusiones y trabajos futuros

Esta sección concluye el conjunto de la tesis y proporciona algunas líneas de investigación futuras.

## 8.1. Conclusiones

En esta tesis hemos abordado la tarea del pre-procesamiento de datos sobre problemas de clasificación de tipo LDL. Dicha tarea ha resultado muy prometedora en otros tipos de problemas de clasificación por lo que vimos la necesidad de profundizar en esta área de investigación con el fin de aportar un valor añadido a las propuestas LDL ya existentes. Para llevar a cabo este estudio nos marcamos unos objetivos desde las etapas iniciales, que han dado lugar a tres publicaciones en revistas de alto impacto. Pasemos a resumir cuales han sido dichos objetivos y como se han desarrollado.

El primer objetivo, correspondiente al estudio y revisión bibliográfica del estado del arte en LDL y estrategias de pre-procesamiento, se ha desarrollado de forma transversal a lo largo de todo el trabajo de tesis. En las primeras etapas, se investigaron a fondo las aportaciones realizadas en estos campos de estudio, permitiendo encaminar los siguientes trabajos hacía las áreas más prometedoras. Después, continuamos haciendo un seguimiento durante el desarrollo de los objetivos posteriores, con el fin de mantener actualizados los conocimiento sobre las temáticas abordadas así como para detectar estudios que pudieran ser de especial interés.

El segundo objetivo marcado fue el de mejorar los conjuntos de datos LDL existentes mediante el diseño de un método de generación sintética de muestras, específicamente diseñado para abordar problemas de tipo LDL. El método ideado, SSG-LDL [57], permite mejorar el rendimiento de los clasificadores LDL subyacentes añadiendo una etapa preliminar de pre-procesamiento aplicada al conjunto de datos original. La propuesta parte de algoritmos de *oversampling* bien conocidos que hemos extendido para tratar las particularidades intrínsecas a los conjuntos de datos LDL. El estudio comparativo llevado a cabo, enfrentando algoritmos de aprendizaje LDL con y sin pre-procesamiento SSG-LDL, recomienda el uso de este método para conseguir mejorar la eficacia de los modelos obtenidos.

El tercero de los objetivos se focaliza en mejorar el rendimiento de los métodos

LDL actuales. Diseñamos e implementamos un método híbrido de reducción de datos que combina la selección de instancias y características, optimizado utilizando algoritmos evolutivos y adaptado a las restricciones LDL. ProLSFEO-LDL [55] es una propuesta específicamente diseñada para mitigar las desventajas del método AA-$k$NN, un clasificador LDL que ha demostrado su eficacia y que permite la explicabilidad del modelo, característica muy demandada hoy en día. El análisis de resultados de nuestra propuesta resalta el aumento de eficacia del clasificador LDL al añadir la etapa de pre-procesamiento y también arroja una mejora en los tiempos de ejecución debido a la consecución de un notable ratio de reducción de datos.

Finalmente, como cuarto y último objetivo, nos propusimos reducir la complejidad intrínseca de los problemas LDL. Concretamente, aplicamos una estrategia de descomposición basada en el esquema *One-vs-One* que permite mejorar la eficacia de los algoritmos de clasificación cuando estos tratan con conjuntos de datos complejos. Al mismo tiempo permiten mitigar los problemas clásicos de los métodos de clasificación: ruido, datos no balanceados, imperfecciones en la toma de datos, irregularidades, etc. DF-LDL [56] es el resultado de este estudio, si lo comparamos con otros métodos LDL de última generación también basados en técnicas de *Ensembles* como es el algoritmo StructRF, nuestra propuesta arroja mejores resultados en la mayoría de los conjuntos de datos analizados. También destacar la flexibilidad de la propuesta ya que puede ser utilizada con cualquier clasificador LDL existente.

A la vista de estos estudios, así como de los resultados de esta tesis, podemos afirmar que la clasificación de tipo LDL adolece de problemas de datos similares a la clasificación clásica. Por lo tanto, el hecho de aplicar diferentes técnicas de preprocesamiento de datos, puede mitigar dichos problemas y mejorar las posteriores etapas del aprendizaje automático y por consiguiente obtener modelos más robustos.

## 8.2. Trabajos futuros

A partir de las conclusiones extraídas de esta tesis, se pueden proponer nuevas y prometedoras líneas de investigación, teniendo como objetivo mejorar los modelos LDL existentes. Algunas de las posibles alternativas a seguir son:

- Combinar generación sintética de muestras con reducción de datos: las dos primeras propuestas expuestas son complementarias. Por un lado, el método SSG-LDL permite obtener un conjunto de datos ampliados, obtenido por interpolación a partir del conjunto de datos original. Por otro lado, el método ProLSFEO-LDL consigue una reducción de datos eliminando las muestras que no aportan valor a la tarea de aprendizaje y seleccionando las características más relevantes para cada etiqueta de salida. Una propuesta que combinara ambos algoritmos podría conseguir unos resultados prometedores tanto en eficacia del modelo como en mejora del tiempo de entrenamiento y predicción. Ahora bien, la calidad de la propuesta resultante dependerá del equilibrio que pueda obtenerse al combinar ambas técnicas. Es una tarea compleja que puede conllevar el ajuste de numerosos parámetros hasta conseguir el resultado deseado.

- Ampliar el método ProLSFEO-LDL más allá del AA-$k$NN: todo el trabajo alrededor de esta propuesta se focaliza en el clasificador AA-$k$NN. Sin embargo el mismo enfoque podría llevarse a cabo utilizando otros algoritmos de aprendizaje LDL. Para ello el mayor reto consistiría en adaptar los algoritmos existentes para hacerlos compatibles con una selección de características específica a cada etiqueta de salida. La selección de prototipos por su lado es directamente extrapolable a cualquier modelo.

- Mejora de los tiempos de entrenamiento de DF-LDL: el método de descomposición DF-LDL necesita entrenar un gran cantidad de modelos, especialmente cuando el número de etiquetas de salida es alto. Un enfoque interesante puede ser diseñar una estrategia de descomposición basada en ECOC [29] con el fin de realizar la descomposición de manera que se ajuste mejor a los datos utilizando un menor número de clasificadores.

- *Big Data* LDL: cada vez son más numerosos los conjuntos de gran tamaño que no son abordables con las técnicas actuales. Un reto futuro sería complementar las técnicas de pre-procesamiento descritas en esta tesis con técnicas que permitan transformar grandes conjuntos de datos en datos inteligentes (*Smart Data*) [77], pudiendo obtener así modelos de calidad en tiempos razonables.

- XAI-LDL: en los últimos años, la Inteligencia Artificial eXplicable (*eXplainable* AI o XAI) se está convirtiendo en una característica crucial para el despliegue práctico de los modelos de inteligentes. Si bien los algoritmos más recientes están alcanzando niveles de rendimiento sin precedentes al aprender a resolver tareas comunes cada vez más complejas, cuando las decisiones derivadas de tales sistemas afectan en última instancia a la vida de los seres humanos (como en la medicina, la ley o la defensa), hay una necesidad emergente de entender cómo esas decisiones son proporcionadas por los métodos de la inteligencia artificial [60]. Algunos métodos LDL estudiados en esta tesis son por sí mismos comprensibles, como es el caso del algoritmo AA-$k$NN. Sin embargo, otros métodos más sofisticados como los basados en *Ensembles* (StructRF, DF-LDL) o los basados en *Deep Learning* (DLDL), requieren de etapas posteriores para conseguir la explicabilidad del modelo. En trabajos futuros se podrán aplicar algunas de las técnicas XAI *post-hoc* descritas en [7] para hacer los modelos LDL más interpretables.

- Hibridar LDL con problemas singulares: los problemas de aprendizaje supervisados singulares son aquellos que no siguen el esquema habitual de problemas supervisados [19], por ejemplo, la clasificación ordinal [61], que incluye relaciones de orden entre las etiquetas de clase, la clasificación monotónica [14] que incorpora restricciones de orden entre los atributos y la clase, el aprendizaje semi-supervisado [18], donde parte de las instancias carece de valores de variables objetivo o la clasificación imbalanceada [36] que puede considerarse un problema de información parcial debido a la falta de representación de las clases minoritarias. Podrían diseñarse nuevas técnicas de pre-procesamiento para tratar conjuntos de datos LDL que presenten alguno de estos problemas singulares, para ello, pueden servir de inspiración los estudios realizados en [58,

59] donde se exponen varios métodos de pre-procesamiento específicamente diseñados para tratar problemas singulares.

# Bibliografía


[1] C.C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[2] D.W. Aha, *Lazy learning*. Springer Science & Business Media, 2013.

[3] D.W. Aha, D. Kibler y M.K. Albert, "Instance-based learning algorithms", *Machine Learning*, vol. 6, n.º 1, págs. 37-66, 1991.

[4] T. Ahonen, A. Hadid y M. Pietikainen, "Face description with local binary patterns: Application to face recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, n.º 12, págs. 2037-2041, 2006.

[5] A. Arnaiz-González, J.F. Diez-Pastor, J.J. Rodriguez y C. García-Osorio, "Local sets for multi-label instance selection", *Applied Soft Computing*, vol. 68, págs. 651-666, 2018.

[6] R. Barandela, J.S Sánchez, V. García y E. Rangel, "Strategies for learning in class imbalance problems", *Pattern Recognition*, vol. 36, n.º 3, págs. 849-851, 2003.

[7] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila y F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", *Information Fusion*, vol. 58, págs. 82-115, 2020.

[8] Z. Barutcuoglu, R.E. Schapire y O.G. Troyanskaya, "Hierarchical multi-label prediction of gene function", *Bioinformatics*, vol. 22, n.º 7, págs. 830-836, 2006.

[9] A. Benavoli, G. Corani, J. Demšar y M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis", *The Journal of Machine Learning Research*, vol. 18, n.º 1, págs. 2653-2688, 2017.

[10] J. Bi y C. Zhang, "An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme", *Knowledge-Based Systems*, vol. 158, págs. 81-93, 2018.

[11] A.L. Blum y P. Langley, "Selection of relevant features and examples in machine learning", *Artificial intelligence*, vol. 97, n.º 1-2, págs. 245-271, 1997.

[12] M.R. Boutell, J. Luo, X. Shen y C.M. Brown, "Learning multi-label scene classification", *Pattern recognition*, vol. 37, n.º 9, págs. 1757-1771, 2004.

[13] J.R. Cano, S. García y F. Herrera, "Subgroup discover in large size data sets preprocessed using stratified instance selection for increasing the presence of minority classes", *Pattern Recognition Letters*, vol. 29, n.º 16, págs. 2156-2164, 2008.

[14] J.R. Cano, P.A. Gutiérrez, B. Krawczyk, M. Woźniak y S. García, "Monotonic classification: an overview on algorithms, performance measures and data sets", *Neurocomputing*, vol. 341, págs. 168-182, 2019.

[15] J. Carrasco, S. García, M.M. Rueda, S. Das y F. Herrera, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review", *Swarm and Evolutionary Computation*, Art. 100665, 2020.

[16] C. Catal, O. Alan y K. Balkan, "Class noise detection based on software metrics and ROC curves", *Information Sciences*, vol. 181, n.º 21, págs. 4867-4877, 2011.

[17] S.H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions", *City*, vol. 1, n.º 2, págs. 300-307, 2007.

[18] O. Chapelle, B. Schölkopf, A. Zien y col., "Semi-supervised learning, vol. 2", *Cambridge: MIT Press. Cortes, C., & Mohri, M.(2014). Domain adaptation and sample bias correction theory and algorithm for regression. Theoretical Computer Science*, vol. 519, pág. 103 126, 2006.

[19] D. Charte, F. Charte, S. García y F. Herrera, "A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations", *Progress in Artificial Intelligence*, vol. 8, n.º 1, págs. 1-14, 2019.

[20] F. Charte, A.J. Rivera, M.J. del Jesus y F. Herrera, "REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization", *Neurocomputing*, vol. 326, págs. 110-122, 2019.

[21] N.V. Chawla, K.W. Bowyer, L.O. Hall y W.P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique", *Journal of Artificial Intelligence Research*, vol. 16, págs. 321-357, 2002.

[22] N.V. Chawla, D.A. Cieslak, L.O. Hall y A. Joshi, "Automatically countering imbalance and its empirical relationship to cost", *Data Mining and Knowledge Discovery*, vol. 17, n.º 2, págs. 225-252, 2008.

[23] M. Chen, X. Wang, B. Feng y W. Liu, "Structured random forest for label distribution learning", *Neurocomputing*, vol. 320, págs. 171-182, 2018.

[24] P.J. Clark, "An extension of the coefficient of divergence for use with multiple characters", *Copeia*, vol. 1952, n.º 2, págs. 61-64, 1952.

[25] T. Cover y P. Hart, "Nearest neighbor pattern classification", *IEEE Transactions on Tnformation Theory*, vol. 13, n.º 1, págs. 21-27, 1967.

[26] R.M.O. Cruz, R. Sabourin y G.D.C. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives", *Information Fusion*, vol. 41, págs. 195-216, 2018.

[27] S. Das, S. Datta y B.B. Chaudhuri, "Handling data irregularities in classification: Foundations, trends, and future challenges", *Pattern Recognition*, vol. 81, págs. 674-693, 2018.

[28] J. Derrac, S. García, D. Molina y F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, vol. 1, n.º 1, págs. 3-18, 2011.

[29] T.G. Dietterich y G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", *Journal of Artificial Intelligence Research*, vol. 2, págs. 263-286, 1994.

[30] H.H. Do y E. Rahm, "Matching large schemas: Approaches and evaluation", *Information Systems*, vol. 32, n.º 6, págs. 857-885, 2007.

[31] A. Doan, P. Domingos y A. Halevy, "Learning to match the schemas of data sources: A multistrategy approach", *Machine Learning*, vol. 50, n.º 3, págs. 279-301, 2003.

[32] N.R. Draper y H. Smith, *Applied regression analysis*. John Wiley & Sons, 1998, vol. 326.

[33] R.O. Duda, P.E. Hart y D.G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[34] M.B. Eisen, P.T. Spellman, P.O. Brown y D. Botstein, "Cluster analysis and display of genome-wide expression patterns", en *Proceedings of the National Academy of Sciences*, vol. 95, National Acad Sciences, 1998, págs. 14 863-14 868.

[35] L.J. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination", en *Foundations of Genetic Algorithms*, vol. 1, Elsevier, 1991, págs. 265-283.

[36] A. Fernández, S. García, M. Galar, R.C. Prati, B. Krawczyk y F. Herrera, *Learning from imbalanced data sets*. Springer, 2018.

[37] A. Fernández, S. García, F. Herrera y N.V. Chawla, "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary", *Journal of Artificial Intelligence Research*, vol. 61, págs. 863-905, 2018.

[38] J.H. Friedman, "Data Mining and Statistics: What's the connection?", *Computing Science and Statistics*, vol. 29, n.º 1, págs. 3-9, 1998.

[39] M. Galar, A. Fernández, E. Barrenechea, H. Bustince y 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 Recognition*, vol. 44, n.º 8, págs. 1761-1776, 2011.

[40] M. Galar, A. Fernández, E. Barrenechea, H. Bustince y F. Herrera, "NMC: nearest matrix classification–A new combination model for pruning One-vs-One ensembles by transforming the aggregation problem", *Information Fusion*, vol. 36, págs. 26-51, 2017.

[41] M. Galar, A. Fernández, E. Barrenechea y F. Herrera, "Empowering difficult classes with a similarity-based aggregation in multi-class classification problems", *Information Sciences*, vol. 264, págs. 135-157, 2014.

[42] M. Galar, A. Fernández, E. Barrenechea, H. Bustince y F. Herrera, "Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers", *Pattern Recognition*, vol. 46, n.º 12, págs. 3412-3424, 2013.

[43] D. Gamberger, N. Lavrac y S. Dzeroski, "Noise detection and elimination in data preprocessing: experiments in medical domains", *Applied Artificial Intelligence*, vol. 14, n.º 2, págs. 205-223, 2000.

[44] B.B. Gao, C. Xing, C.W. Xie, J. Wu y X. Geng, "Deep label distribution learning with label ambiguity", *IEEE Transactions on Image Processing*, vol. 26, n.º 6, págs. 2825-2838, 2017.

[45] S. García, J. Luengo y F. Herrera, "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining", *Knowledge-Based Systems*, vol. 98, págs. 1-29, 2016.

[46] L.P.F. García, J.A. Sáez, J. Luengo, A.C. Lorena, A.C.P.L.F. de Carvalho y F. Herrera, "Using the One-vs-One decomposition to improve the performance of class noise filters via an aggregation strategy in multi-class classification problems", *Knowledge-Based Systems*, vol. 90, págs. 153-164, 2015.

[47] S. García, J. Derrac, J.R. Cano y F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, n.º 3, págs. 417-435, 2012.

[48] S. García, J. Luengo y F. Herrera, *Data preprocessing in data mining*. Springer, 2015, vol. 72.

[49] X. Geng, "Label distribution learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, n.º 7, págs. 1734-1748, 2016.

[50] X. Geng y P. Hou, "Pre-release prediction of crowd opinion on movies by label distribution learning", en *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2015-January, 2015, págs. 3511-3517.

[51] X. Geng y L. Luo, "Multilabel ranking with inconsistent rankers", en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, págs. 3742-3747.

[52] X. Geng, C. Yin y Z.H. Zhou, "Facial age estimation by learning from label distributions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, n.º 10, págs. 2401-2412, 2013.

[53] E. Gibaja y S. Ventura, "A tutorial on multilabel learning", *ACM Computing Surveys (CSUR)*, vol. 47, n.º 3, págs. 1-38, 2015.

[54] D.E. Goldberg, "Genetic algorithms in search", *Optimization, and Machine-Learning*, 1989.

[55] M. González, J.R. Cano y S. García, "ProLSFEO-LDL: Prototype Selection and Label-Specific Feature Evolutionary Optimization for Label Distribution Learning", *Applied Sciences*, vol. 10, n.º 9, Art. 3089, 2020.

[56] M. González, G. González-Almagro, I. Triguero, J.R. Cano y S. García, "Decomposition-Fusion for Label Distribution Learning", *Information Fusion*, vol. 66, págs. 64-75, 2021.

[57] M. González, J. Luengo, J.R. Cano y S. García, "Synthetic Sample Generation for Label Distribution Learning", *Information Sciences*, vol. 544, págs. 197-213, 2021.

[58] S. González, S. García, S.T. Li y F. Herrera, "Chain based sampling for monotonic imbalanced classification", *Information Sciences*, vol. 474, págs. 187-204, 2019.

[59] S. González, S. García, S.T. Li, R. John y F. Herrera, "Fuzzy k-Nearest Neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise", *Neurocomputing*, 2020.

[60] B. Goodman y S. Flaxman, "European Union regulations on algorithmic decision-making and a right to explanation"", *AI Magazine*, vol. 38, n.º 3, págs. 50-57, 2017.

[61] P.A. Gutiérrez y S. García, "Current prospects on ordinal and monotonic classification", *Progress in Artificial Intelligence*, vol. 5, n.º 3, págs. 171-179, 2016.

[62] I. Guyon y A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, vol. 3, n.º Mar, págs. 1157-1182, 2003.

[63] J. Han, J. Pei y M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[64] T. Hastie y R. Tibshirani, "Classification by pairwise coupling", *Annals of Statistics*, vol. 26, n.º 2, págs. 451-471, 1998.

[65] T. Hastie, R. Tibshirani y J. Friedman, "Unsupervised learning", en *The Elements of Statistical Learning*, Springer, 2009, págs. 485-585.

[66] F. Herrera, F. Charte, A.J. Rivera y M.J. Del Jesus, *Multilabel classification: Problem analysis, metrics and techniques*. Springer, 2016.

[67] J. Huang, G. Li, Q. Huang y X. Wu, "Learning label-specific features and class-dependent labels for multi-label classification", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, n.º 12, págs. 3309-3323, 2016.

[68] S. Kanj, F. Abdallah, T. Denoeux y K. Tout, "Editing training data for multi-label classification with the k-nearest neighbor rule", *Pattern Analysis and Applications*, vol. 19, n.º 1, págs. 145-161, 2016.

[69] R. Kohavi, G.H. John y col., "Wrappers for feature subset selection", *Artificial Intelligence*, vol. 97, n.º 1-2, págs. 273-324, 1997.

[70] P. Kontschieder, M. Fiterau, A. Criminisi y S.R. Bulò, "Deep neural decision forests", en *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2016-January, 2016, págs. 4190-4194.

[71] M. Lachaize, S. Le Hégarat-Mascle, E. Aldea, A. Maitrot y R. Reynaud, "Evidential framework for error correcting output code classification", *Engineering Applications of Artificial Intelligence*, vol. 73, págs. 10-21, 2018.

[72] H.T. Lin, C.J. Lin y R.C. Weng, "A note on Platt's probabilistic outputs for support vector machines", *Machine Learning*, vol. 68, n.º 3, págs. 267-276, 2007.

[73] H. Liu y H. Motoda, *Feature extraction, construction and selection: A data mining perspective*. Springer Science & Business Media, 1998, vol. 453.

[74] H. Liu y H. Motoda, "On issues of instance selection", *Data Mining and Knowledge Discovery*, vol. 6, n.º 2, págs. 115-130, 2002.

[75] H. Liu y H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[76] H. Liu y L. Yu, "Toward integrating feature selection algorithms for classification and clustering", *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, n.º 4, págs. 491-502, 2005.

[77] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García y F. Herrera, *Big Data Preprocessing: Enabling Smart Data*. Springer, 2020.

[78] M. Lyons, S. Akamatsu, M. Kamachi y J. Gyoba, "Coding facial expressions with gabor wavelets", en *Proceedings - 3rd IEEE International Conference on Automatic Face and Gesture Recognition, FG 1998*, 1998, págs. 200-205.

[79] O. Maimon y L. Rokach, *Data mining and knowledge discovery handbook*. Springer, 2005.

[80] M. Millán-Giraldo, V. García y J.S. Sánchez, "Instance Selection Methods and Resampling Techniques for Dissimilarity Representation with Imbalanced Data Sets", en *Pattern Recognition-Applications and Methods*, Springer, 2013, págs. 149-160.

[81] B. Mirkin, *Clustering: a data recovery approach*. CRC Press, 2012.

[82] J.M. Moyano, E.L. Gibaja, K.J. Cios y S. Ventura, "Review of ensembles of multi-label classifiers: models, experimental study and prospects", *Information Fusion*, vol. 44, págs. 33-45, 2018.

[83] J.A. Olvera-López, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad y J. Kittler, "A review of instance selection methods", *Artificial Intelligence Review*, vol. 34, n.º 2, págs. 133-143, 2010.

[84] G. Piateski y W. Frawley, *Knowledge discovery in databases*. MIT press, 1991.

[85] R. Polikar, "Ensemble based systems in decision making", *IEEE Circuits and systems magazine*, vol. 6, n.º 3, págs. 21-45, 2006.

[86] R.C. Prati, G.E.A.P.A. Batista y M.C. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior", en *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, Springer, vol. 2972, 2004, págs. 312-321.

[87] R.C. Prati, G.E.A.P.A. Batista y D.F. Silva, "Class imbalance revisited: a new experimental setup to assess the performance of treatment methods", *Knowledge and Information Systems*, vol. 45, n.º 1, págs. 247-270, 2015.

[88] R.C. Prati, J. Luengo y F. Herrera, "Emerging topics and challenges of learning from noisy data in nonstandard classification: a survey beyond binary class noise", *Knowledge and Information Systems*, vol. 60, n.º 1, págs. 63-97, 2019.

[89] D. Pyle, *Data preparation for data mining*. morgan kaufmann, 1999.

[90] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak y F. Herrera, "A survey on data preprocessing for data stream mining", *Neurocomputing*, vol. 239, n.º C, págs. 39-57, 2017.

[91] T. Ren, X. Jia, W. Li, L. Chen y Z. Li, "Label distribution learning with label-specific features.", en *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019-August, 2019, págs. 3318-3324.

[92] Y. Ren y X. Geng, "Sense Beauty by Label Distribution Learning.", en *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0, 2017, págs. 2648-2654.

[93] R. Rifkin y A. Klautau, "In defense of one-vs-all classification", *Journal of Machine Learning Research*, vol. 5, n.º Jan, págs. 101-141, 2004.

[94] A. Roy, R.M.O. Cruz, R. Sabourin y G.D.C. Cavalcanti, "A study on combining dynamic selection and data preprocessing for imbalance learning", *Neurocomputing*, vol. 286, págs. 179-192, 2018.

[95] Y. Saeys, I. Inza y P. Larrañaga, "A review of feature selection techniques in bioinformatics", *Bioinformatics*, vol. 23, n.º 19, págs. 2507-2517, 2007.

[96] J.A. Sáez, M. Galar y B. Krawczyk, "Addressing the Overlapping Data Problem in Classification Using the One-vs-One Decomposition Strategy", *IEEE Access*, vol. 7, págs. 83 396-83 411, 2019.

[97] J.A. Sáez, M. Galar, J. Luengo y F. Herrera, "Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition", *Knowledge and Information Systems*, vol. 38, n.º 1, págs. 179-206, 2014.

[98] B. Schölkopf, A.J. Smola, F. Bach y col., *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[99] W. Shen, K. Zhao, Y. Guo y A.L. Yuille, "Label distribution learning forests", en *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017, págs. 835-844.

[100] Y. Song, J. Liang, J. Lu y X. Zhao, "An efficient instance selection algorithm for k nearest neighbor regression", *Neurocomputing*, vol. 251, págs. 26-34, 2017.

[101] J. Tang, S. Alelyani y H. Liu, "Feature selection for classification: A review", *Data Classification: Algorithms and Applications*, págs. 37-64, 2014.

[102] I. Triguero, J. Derrac, S. García y F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, n.º 1, págs. 86-100, 2011.

[103] I. Triguero y C. Vens, "Labelling strategies for hierarchical multi-label classification techniques", *Pattern Recognition*, vol. 56, págs. 170-183, 2016.

[104] J. Wang y X. Geng, "Classification with label distribution learning", en *IJCAI International Joint Conference on Artificial Intelligence*, AAAI Press, vol. 2019-August, 2019, págs. 3712-3718.

[105] K. Wang y X. Geng, "Binary Coding based Label Distribution Learning.", en *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2018-July, 2018, págs. 2783-2789.

[106] K. Wang y X. Geng, "Discrete binary coding based label distribution learning", en *IJCAI International Joint Conference on Artificial Intelligence*, AAAI Press, vol. 2019-August, 2019, págs. 3733-3739.

[107] Y. Wang y J. Dai, "Label Distribution Feature Selection Based on Mutual Information in Fuzzy Rough Set Theory", en *Proceedings of the International Joint Conference on Neural Networks*, vol. 2019-July, 2019, Art. 8851998.

[108] T.F. Wu, C.J. Lin y R.C. Weng, "Probability estimates for multi-class classification by pairwise coupling", *Journal of Machine Learning Research*, vol. 5, n.º Aug, págs. 975-1005, 2004.

[109] C. Xing, X. Geng y H. Xue, "Logistic boosting regression for label distribution learning", en *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, págs. 4489-4497.

[110] L. Xu, J. Chen e Y. Gan, "Head pose estimation using improved label distribution learning with fewer annotations", *Multimedia Tools and Applications*, págs. 1-22, 2019.

[111] D. Xue, Z. Hong, S. Guo, L. Gao, L. Wu, J. Zheng y N. Zhao, "Personality recognition on social media with label distribution learning", *IEEE Access*, vol. 5, págs. 13 478-13 488, 2017.

[112] J. Yang, D. She y M. Sun, "Joint Image Emotion Classification and Distribution Learning via Deep Convolutional Neural Network.", en *IJCAI International Joint Conference on Artificial Intelligence*, 2017, págs. 3266-3272.

[113] L. Yin, X. Wei, Y. Sun, J. Wang y M.J. Rosato, "A 3D facial expression database for facial behavior research", en *FGR 2006: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, vol. 2006, 2006, págs. 211-216.

[114] J.F. Yu, D.K. Jiang, K. Xiao, Y. Jin, J.H. Wang y X. Sun, "Discriminate the falsely predicted protein-coding genes in Aeropyrum Pernix K1 genome based on graphical representation", *Match-Communications in Mathematical and Computer Chemistry*, vol. 67, n.º 3, pág. 845, 2012.

[115] Y. Zhai, J. Dai y H. Shi, "Label Distribution Learning Based on Ensemble Neural Networks", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11303 LNCS, págs. 593-602, 2018.

[116] J. Zhang, C. Li, D. Cao, Y. Lin, S. Su, L. Dai y S. Li, "Multi-label learning with label-specific features by resolving label correlations", *Knowledge-Based Systems*, vol. 159, págs. 148-157, 2018.

[117] M.L. Zhang y L. Wu, "Lift: Multi-label learning with label-specific features", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, n.º 1, págs. 107-120, 2014.

[118] Z. Zhang, M. Wang y X. Geng, "Crowd counting in public video surveillance by label distribution learning", *Neurocomputing*, vol. 166, págs. 151-163, 2015.

[119] Z.L. Zhang, X.G. Luo, Y. Yu, B.W. Yuan y J.F. Tang, "Integration of an improved dynamic ensemble selection approach to enhance one-vs-one scheme", *Engineering Applications of Artificial Intelligence*, vol. 74, págs. 43-53, 2018.

[120] X. Zheng, X. Jia y W. Li, "Label distribution learning by exploiting sample correlations locally", en *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018, págs. 4556-4563.

[121] Z.H. Zhou, Y. Yu y C. Qian, *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.

[122] X. Zhu y A.B. Goldberg, "Introduction to semi-supervised learning", *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, n.º 1, págs. 1-130, 2009.

[123] X. Zhu y X. Wu, "Class noise vs. attribute noise: A quantitative study", *Artificial Intelligence Review*, vol. 22, n.º 3, págs. 177-210, 2004.