UNIVERSIDAD NACIONAL DE LA PLATA

UNIVERSIDAD DE GRANADA

Análisis y diseño de técnicas de preprocesamiento de instancias escalables para problemas no balanceados en Big Data. Aplicaciones en situaciones de emergencias humanitarias.

Ing. María José Basgall

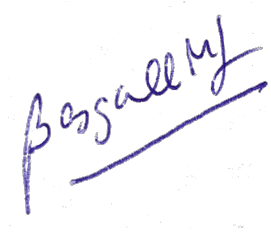
Tesis doctoral en cotutela
Programa de Doctorado en Tecnologías de la Información y la Comunicación.

Directores:
Prof. Dr. Marcelo Naiouf (UNLP, Argentina)
Prof. Dr. Alberto Fernández Hilario (UGR, España)

2022

La doctoranda **María José Basgall** y los directores de la tesis **Marcelo Naiouf** y **Alberto Fernández Hilario**.
Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por la doctoranda bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Marzo, 2022.

Director de la Tesis:	Director de la Tesis:	Doctoranda:
		
Fdo.: Marcelo Naiouf	Fdo.: Alberto Fernández Hilario	Fdo.: María José Basgall

*We are drowning in information and starving for knowledge.— (***Rutherford D. Roger)**

Índice general

Siglas

kNN *k - Nearest Neighbors*

ACC *Accuracy*

AI *Artificial Intelligence*

AUC *Area under the ROC Curve*

COVID-19 *Coronavirus Disease 2019*

DAG *Directed Acyclic Graph*

DIS *Democratic Instance Selection*

DM *Data Mining*

DS *Data Science*

DT *Decision Tree*

EDA *Exploratory Data Analysis*

EH Emergencias Humanitarias

FCNN rule *Fast Condensed Nearest Neighbor rule*

FCNN_MR *FCNN for Big Data*

FDR^2-BD *Fast Data Reduction Recommendation tool for tabular Big Data*

FN *False Negative*

FNR *False Negative Rate*

FP *False Positive*

FPR *False Positive Rate*

FS *Feature Selection*

GM *Geometric Mean*

GOD *Grid Overlapping Degree*

HDFS *Hadoop Distributed File System*

IoT *Internet of Things*

IS *Instance Selection*

MIFs *Most Important Features*

ML *Machine Learning*

MR *MapReduce*

OvRs *Overlapping Rules*

PLT *Predictive Loss Threshold*

PR *Prototype Reduction*

PRs *Pure Rules*

RDD *Resilient Distributed Datasets*

RF *Random Forest*

RMHC *Random Mutation Hill Climbing*

ROS *Random OverSampling*

RPR *Recommended Percentage Reduction*

RUS *Random UnderSampling*

SMOTE *Synthetic Minority Oversampling TEchnique*

SP *Stream Processing*

SSMA *Steady-State Memetic Algorithm*

TN *True Negative*

TNR *True Negative Rate*

TP *True Positive*

TPR *True Positive Rate*

Parte I

Introducción y base teórica

1 Introducción

En los últimos años, se ha visto en aumento la generación continua de datos provenientes de distintas fuentes [Wu+14]. Las aplicaciones de Internet (redes sociales, correo electrónico, comercio electrónico), los sensores de dispositivos de uso diario (relojes inteligentes, *smart homes*, *smart cities*), los sensores de experimentos científicos (CERN[1], Fermilab[2]), son sólo algunos ejemplos [Lu17; Bou+21]. A su vez, esta situación se ha visto exacerbada desde el comienzo de la pandemia COVID-19 con el notorio incremento del uso de Internet [Dom21] (a causa del crecimiento de las herramientas de comunicación online y del teletrabajo) y de la digitalización y toma de datos cotidiana.

Este enorme volumen de datos, realmente variado en su tipología, y que se genera y procesa a gran velocidad, es lo que se conoce como Big Data [Mar13]. La importancia de los datos está en extraer conocimiento de ellos para realizar tareas de predicción de comportamientos a futuro, realizar segmentación en grupos, generar asociaciones de interés, o cualquier otra tarea que permita caracterizar la información, todo ello en aras de facilitar la toma de decisiones. Para ello, y dada la enorme complejidad del proceso, es imprescindible la aplicación de modelos de Aprendizaje Automático (en inglés *Machine Learning* (ML)). De todo ello se puede extraer que tanto Big Data como ML son tendencia en estos escenarios en donde se busca procesar grandes volúmenes de datos en tiempos razonables y con el fin de obtener conocimiento útil a partir de los mismos y realizar aproximaciones descriptivas, predictivas, diagnósticas o prescriptivas. La búsqueda bibliográfica en el conocido portal científico *Web-of-Science*[3] revela que el tópico central de esta tesis, «Big Data», es un área de investigación muy popular, con más de 87000 publicaciones que contienen dicho término en el título, en el resumen o en las palabras claves del artículo, en los últimos 10 años (a fecha de 24 de noviembre de 2021). Además, hay que tener en cuenta que *Web-of-Science* ofrece una estimación conservadora; por lo tanto, es probable que el número real de publicaciones sea mucho mayor. Estas disciplinas están siendo aplicadas cada vez más a distintas áreas. Se conoce su aplicación en la medicina, en la detección de fraudes electrónicos y seguridad informática en general, en el descubrimiento de nueva física, en problemas de bioinformática, en uso eficiente

1 https://home.cern/
2 https://www.fnal.gov/
3 https://www.webofscience.com/wos/woscc/basic-search

de energía eléctrica, en política, en deportes, entre tantos otros ejemplos [CDH16; MD13; Tuf14; Gon17; Sah+21; Ari+22]. Tal es su relevancia que dichos campos de conocimiento no sólo forman parte de los planes de estudios en formación de postgrado de carreras afines, sino que últimamente se han comenzado a incorporar en la oferta de materias de grado en prestigiosas instituciones.

Por ser tendencia, hay que tener en cuenta que son áreas muy dinámicas y, en consecuencia, demandan una continua actualización. Además, para el análisis de Big Data basado en principios de ML, las técnicas de descubrimiento de conocimiento que hasta ahora ofrecían buenos resultados, no siempre soportan manejar grandes conjuntos de datos por temas de escalabilidad. Es por ello que necesitan ser adaptadas para trabajar en entornos distribuidos siguiendo un enfoque "divide y vencerás", o se deben crear nuevas técnicas o estrategias para lidiar con este nuevo escenario. Asimismo, Big Data supone restricciones en cuanto a capacidad de cómputo, de comunicación, de almacenamiento, entre otras cuestiones; por lo tanto, se debe tener acceso a la infraestructura adecuada, que provea los recursos computacionales necesarios [Fer+14].

A su vez, hay que tener en cuenta que los conjuntos de datos normalmente pueden tener ciertas características o complejidades no deseadas [DDC18; MTH20] que interfieren en la efectividad del proceso de extracción del conocimiento [GLH15]. La presencia de ellas puede deberse tanto a la naturaleza del problema que los datos representan, como a la forma en la que son capturados. Entre las complejidades, pueden encontrarse redundancia, ruido, valores faltantes, zonas ambiguas del problema, significativa desproporción de los datos que representan a distintos conceptos claves (desbalance o desequilibrio), alta dimensionalidad en los datos, entre otras. Frente a la presencia de algunas de estas características, los datos deben ser tratados debido a que la mayoría de los modelos de aprendizaje asumen que los datos están libres de ellas [Fer+18b]. Las técnicas que se apliquen en cada caso, conforman la etapa de preprocesamiento del proceso de extracción del conocimiento en Ciencia de Datos. Ya en escenarios de datos de tamaño estándar, es conocido que esta tarea es una de las que más tiempo demanda para ser completada; por consiguiente, en escenarios Big Data la situación se intensifica. Sin embargo, preprocesar los conjuntos Big Data con el fin de obtener datos de buena calidad, conocido como Smart Data [Iaf14; Lue+20], es un paso necesario para conseguir que los modelos alcanzados resulten de utilidad. En este contexto, Smart Data usualmente representa a un subconjunto del conjunto Big Data original, el cual no tiene por qué tener un volumen tan elevado, y a partir del cual será utilizado como entrada a los algoritmos para la extracción de conocimiento.

En consecuencia, esta línea de investigación aborda al preprocesamiento distribuido y escalable de conjuntos Big Data, con el fin de obtener Smart Data.

Particularmente se centra en los conjuntos de datos que representan un problema de clasificación, es decir, instancias de datos con sus respectivas etiquetas de clase, y a partir de las cuales se lleva a cabo un proceso de aprendizaje para obtener un modelo que represente a dichos datos. A continuación, y mediante la utilización de dicho modelo, poder clasificar nuevas instancias de datos de las cuales se desconoce su etiqueta de clase.

Dado el impacto que tienen las características intrínsecas de los datos en el rendimiento de los modelos de aprendizaje y que existen pocas soluciones escalables y capaces de manejar Big Data relacionadas a este tema, en esta tesis se presentan tres propuestas para identificar y/o tratar algunas de esas características. Todas las propuestas son completamente escalables, con el fin de procesar Big Data en tiempos razonables.

En todos los casos, las implementaciones de nuestras propuestas se han llevado a cabo utilizando el framework de computación distribuida denominado Apache Spark [Zah+16b] y su librería MLlib [Men+16] dedicada para ML, a través del lenguaje de programación Scala[4]. Las herramientas generadas son *open source* y se encuentran disponibles en sus respectivos repositorios o como parte del repositorio de paquetes de terceros para Apache Spark llamado Spark Packages[5].

1.1. Objetivos

El objetivo general de esta tesis doctoral es contribuir al área de preprocesamiento en el contexto de Big Data, dada la escasa cantidad de soluciones distribuidas en esta temática capaces de manipular estos tipos de datos. Puntualmente, proponer nuevas soluciones escalables con foco al tratamiento de las características intrínsecas de los datos que se encuentran más frecuentes en problemas Big Data. En concreto, atender al desbalance de datos, la redundancia y alta dimensionalidad, y al solapamiento de clases.

Para ello, se establecen los siguientes objetivos específicos:

- Habilitar que un algoritmo del estado del arte altamente empleado para el tratamiento de la distribución de clases en escenarios de datos tradicionales (Small Data), sea capaz de obtener resultados adecuados a partir de grandes conjuntos de datos de manera distribuida y en tiempos de ejecución razonables.

- Diseñar e implementar una metodología rápida y escalable para la reducción tanto en instancias como en atributos para los conjuntos Big Data que

4 https://www.scala-lang.org/
5 https://spark-packages.org/

presentan alta redundancia y dimensionalidad, manteniendo la capacidad predictiva del conjunto de datos original.

- Diseñar e implementar una estrategia para la caracterización escalable de los datos en el contexto de clasificación en Big Data, con foco en las zonas ambiguas del problema.
- Aplicar el conocimiento adquirido durante la fase de desarrollo para resolver problemas de interés relacionados con emergencias humanitarias.

1.2. Metodología

La presente tesis se ha llevado a cabo siguiendo el esquema del método científico tradicional e integrando metodologías tanto teóricas como prácticas. En concreto, las tareas de investigación y experimentación han sido los siguientes:

1. **Estudio de los fundamentos teóricos del tema y el estado del conocimiento tecnológico del mismo:** estudio exhaustivo del problema de aprendizaje automático con conjuntos de datos Big Data, con foco en el preprocesamiento de los datos a gran escala, como tarea fundamental en el proceso de descubrimiento de conocimiento. Análisis de las herramientas tecnológicas existentes para el desarrollo de soluciones distribuidas que soporten Big Data.
2. **Formulación de hipótesis:** diseño de técnicas de preprocesamiento para abordar el tratamiento y caracterización de un conjunto de complejidades intrínsecas que pueden estar presentes en los datos, con el objetivo de mejorar la calidad de los mismos. Los algoritmos diseñados y desarrollados se enmarcan dentro de los objetivos anteriormente planteados con el fin de abordar de manera adecuada los problemas dentro del escenario Big Data.
3. **Obtención de resultados:** mediante diferentes métricas para la evaluación del desempeño, a partir de las soluciones propuestas sobre conjuntos del mundo real en entornos Big Data.
4. **Contraste de hipótesis:** cotejo de las observaciones obtenidas por nuestras propuestas contra soluciones del estado del arte, con el propósito de analizar la calidad de nuestros aportes.
5. **Prueba o rechazo de la hipótesis:** aceptación, o rechazo y modificación, de las técnicas desarrolladas como consecuencia de los experimentos reali-

zados y los resultados obtenidos. De ser necesario, se repetirán los pasos anteriores para formular nuevas hipótesis.

6. **Evaluación y publicación de los resultados obtenidos:** la participación en eventos de la especialidad y la publicación de resultados en revistas es de fundamental importancia para la construcción del conocimiento.

7. **Tesis científica:** redacción de las conclusiones extraídas a lo largo del proceso de investigación, documentando todas las propuestas, experimentaciones y resultados conseguidos.

1.3. Organización de la tesis

La memoria de tesis está estructurada en cuatro partes principales, cada una compuesta por uno o más capítulos. A su vez, cada capítulo, lleva asociado una sección final con un resumen de contenido en aras de condensar las lecciones aprendidas y los principales puntos de interés de cada uno de ellos, resultando especialmente relevante para los capítulos de contribución original (Parte II: Aportes). Así pues, la memoria de tesis se organiza como sigue:

Parte I Además de la presente introducción, esta parte inicial abarca todo lo referente al sustento teórico y estado del arte de los temas involucrados en este trabajo. En concreto, en el Capítulo 2 se comentan las características intrínsecas (o complejidades) que pueden estar presentes en un conjunto de datos. Además, se introduce al análisis exploratorio de los datos, el cual tiene por objetivo obtener información de interés a partir de un conjunto de datos. A continuación, en el Capítulo 3 se introduce el tema del aprendizaje a partir de datos, profundizando en una tarea en particular: la clasificación de los datos supervisada. Además, se mencionan algunas medidas para evaluar dicho aprendizaje y las formas comúnmente utilizadas de validar los resultados obtenidos. Luego, en el Capítulo 4, se aborda el tópico de las técnicas de preprocesamiento de datos. Puntualmente aquellas que efectúan un tratamiento sobre los conjuntos de datos con el fin de contrarrestar los efectos de las complejidades que puedan estar presentes en los mismos y poder así mejorar el rendimiento predictivo en la tarea de clasificación. Por último, el Capítulo 5 presenta una introducción sobre Big Data, y se comentan los conceptos relacionados y herramientas comúnmente utilizadas en dicho contexto.

Parte II Esta parte se centra en la descripción de las contribuciones de esta tesis. Particularmente, en el Capítulo 6, se detalla y analiza nuestra propuesta

de una solución totalmente escalable del estado del arte en datos de tamaño tradicionales adecuada para el tratamiento de conjuntos de datos Big Data no balanceados. Por su parte, el foco del Capítulo 7 es la condensación de datos en Big Data. En dicho capítulo se presenta nuestra propuesta metodológica para la reducción escalable de conjuntos Big Data basada en características y en instancias. Además, mediante una extensa experimentación sobre un amplio conjunto de datos de grandes volúmenes de información, se evalúa nuestro aporte. Finalmente, en el Capítulo 8, se describe una metodología, basada en una técnica de multiresolución, para la caracterización escalable de problemas de clasificación Big Data centrada en las zonas solapadas (o ambiguas) de los mismos.

Parte III En esta parte el foco está puesto en la aplicación de nuestras propuestas sobre un grupo de conjuntos Big Data relacionados a distintas situaciones y aspectos de emergencias humanitarias que no han sido utilizados durante el desarrollo de nuestros aportes, con el fin de no generar un posible sesgo hacia tales escenarios. Estos datos tabulares representan problemas de clasificación binaria para situaciones tales como detección de incendios forestales, clasificación de pacientes con el virus COVID19, predicción de la dimensión del daño ocasionado por un terremoto, entre otras situaciones vinculadas a emergencias humanitarias. En concreto, se busca examinar la bondad de la obtención de los modelos predictivos sobre estos casos de estudio, utilizando para ello los desarrollos realizados durante la presente tesis.

Parte IV Finalmente, se comentan las conclusiones y los principales aportes, y se especifican los trabajos futuros. Además, se listan las contribuciones.

Existe además un apartado de apéndices que extienden a los capítulos principales de esta tesis. Así, en el Apéndice A, se profundiza en el entorno experimental concerniente a nuestras propuestas. En primer lugar, se describe al resto de los problemas de clasificación que representan a cada uno de los conjuntos Big Data utilizados en las distintas experimentaciones de esta tesis. Luego, se detallan las infraestructuras computacionales empleadas para llevar a cabo los experimentos. La sección concluye ahondando sobre la herramienta elegida para el desarrollo de nuestras propuestas: Apache Spark. En concreto se presenta, por un lado, sus principales ventajas y características mediante una comparación contra otra herramienta usualmente empleada para el procesamiento de Big Data; y, por otro lado, se muestra un compendio de todas las primitivas de Apache Spark utilizadas en los distintos desarrollos enmarcados en esta tesis.

Por último, en el Apéndice B se comentan dos propuestas relacionadas al

procesamiento escalable de flujo de datos en el contexto Big Data. En el primer trabajo se aplica una tarea de agrupamiento (clustering) sobre datos de la red social Twitter[6]. El segundo trabajo tiene aplicación en el campo de la econometría, con datos de transacciones de criptomonedas. El objetivo es calcular, en tiempo cortos de respuestas, un indicador ampliamente utilizado en el análisis del mercado para la detección de memoria a largo plazo, llamado exponente de Hurst.

Ambos trabajos se ubican en este apéndice dado que, aunque la temática sobre Big Data es compartida, su relación con los objetivos de la presente tesis es solo parcial. En concreto, ambas propuestas fueron diseñadas para resultar escalables y por tanto capaces de trabajar con Big Data. Sin embargo, su foco no está puesto en el tratamiento de las complejidades de los datos, sino que son aproximaciones enmarcadas en la tarea de flujos de datos, con aplicación puntual a una red social y a criptomonedas. Su inclusión se debe al posible interés del lector con vistas a la ampliación del diseño de soluciones Big Data en diferentes contextos.

6 https://twitter.com

2 Características de los datos

En este capítulo se describen las características o complejidades intrínsecas que pueden estar presentes en un conjunto de datos. Dichas características se pueden deber tanto a la forma en la que los datos son capturados, como a la naturaleza del problema que los datos representen, o a ambas razones. Independientemente del motivo por el cual se encuentren presentes en un conjunto de datos, es importante identificar dichas complejidades debido a que interfieren en la efectividad del proceso de extracción del conocimiento. En otras palabras, la calidad de los datos que se utilicen guarda una fuerte relación con la calidad de los resultados y/o modelos a obtener. Por lo tanto, en la Sección 2.1 se describen las características o complejidades intrínsecas más comúnmente presentes en los conjuntos de datos. Posteriormente, en la Sección 2.2, se comenta brevemente un análisis inicial a llevar a cabo cuando se trabaja con un conjunto de datos desconocido, con el fin de obtener información útil sobre el mismo. Asimismo, se presentan algunas métricas o medidas de complejidades de los datos más populares del estado del arte. Por último, en la Sección 2.3, se comenta brevemente la información presentada en este capítulo.

2.1. Complejidades intrínsecas de los datos

La mayoría de las veces, en el proceso del descubrimiento de conocimiento a partir de un conjunto de datos, no todos los ejemplos (o instancias) van a ser relevantes. Por lo general, hay un subconjunto de datos que contiene información útil y, a partir de él, hay una parte más pequeña de datos que retienen el verdadero conocimiento. Por lo tanto, es necesario deshacerse de estas "capas" de datos que no generan más que la degradación de la calidad del conjunto y dificultan la creación de un modelo que represente el conocimiento real. Así pues, cualquiera sea el tamaño del problema representado por los datos, el aprendizaje de un modelo a partir de datos de baja calidad genera resultados también de baja calidad (comúnmente se asocia esta situación con el principio GIGO *"Garbage In, Garbage Out"* [GLH15]).

Teniendo en cuenta estas cuestiones, es indispensable comprender claramente la naturaleza de los datos y los "inconvenientes" o características que ellos contienen. De este modo, se podrá seleccionar las técnicas de preprocesamiento

más apropiadas para cada caso, y así crear modelos precisos pero no sesgados [Fer+18b].

A continuación se comentan las principales complejidades o características que pueden estar presentes en los datos, a saber, valores perdidos, valores atípicos, ruido, falta de datos, redundancia, alta dimensionalidad, datos no balanceados, y solapamiento:

Valores perdidos [Par96; Mon+19]. Se da cuando faltan valores de algunos de los atributos (o características) que conforman un dato o instancia. Esta situación se muestra en la Sección 2.1, donde los valores perdidos están representados por etiquetas tipo "Not a Number" (*NaN*). Existen muchas razones para que los conjuntos de datos de aplicaciones del mundo real contengan valores perdidos. El olvido de rellenar un campo al momento de la carga o que la fuente de datos sea un dispositivo o sensor defectuoso, la falta de validación en la toma de datos, los errores de programación en la recopilación o almacenamiento de los datos, son algunos ejemplos [AY09; Azu05]. La importancia de identificar y actuar en pos de manejar los valores perdidos radica en que muchos algoritmos de aprendizaje asumen que todos los valores son numéricos y que mantienen un valor significativo.

F1	F2	F3	F..
2.0	NaN	4.0	...
9.0	10.0	NaN	...
NaN	17.0	7.0	...

Figura 2.1: Valores perdidos.

Valores atípicos [DBP17]. Se debe a la presencia de valores raros o atípicos (en inglés *outliers*) en los atributos de las instancias, extremadamente alejados del orden de magnitud del resto de los valores. Un ejemplo de esto se presenta en la Figura 2.2, donde los valores atípicos se encuentran hacia la derecha de la imagen. En un problema de aprendizaje, un valor atípico puede incrementar el error en los resultados obtenidos sin siquiera ser realmente una instancia que represente parte del problema del que se quiere aprender, es decir, son instancias significativamente distintas a las del concepto que se supone intentan representar.

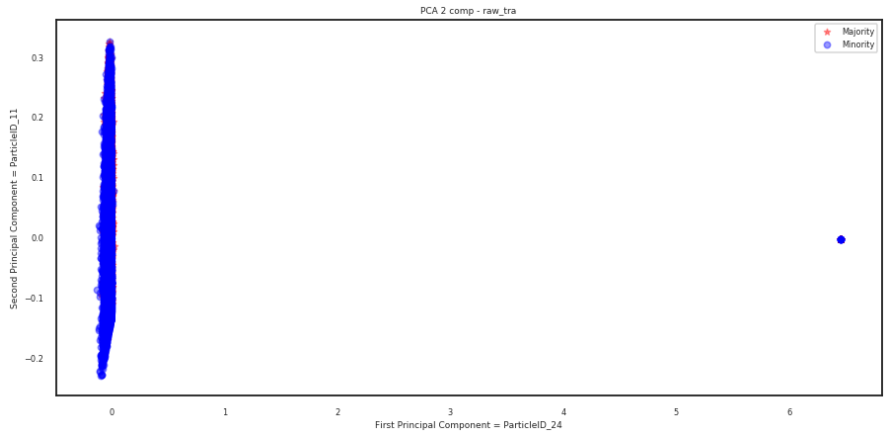


Figura 2.2: Valores atípicos.

Ruido [ZW04]. Se da una situación de ruido cuando en un conjunto de datos se tiene un atributo que en la mayoría de las instancias toma valores, por ejemplo, de tipo real, y en alguna toma un valor de tipo carácter o de otro tipo distinto al real. A modo de ejemplo, se presenta la Figura 2.3, con ruido en la segunda y tercera característica. Actualmente, las utilidades para la lectura de datos a partir de un archivo de muchos lenguajes de programación, aplican una tarea de inferencia de los tipos de datos de cada atributo, basados en los valores que toman la mayoría de las instancias para dicho atributo. En caso de detectar ruido, lo informan o lo completan con un valor pre-establecido especificado por el usuario para estos casos. Pero además, se puede dar otro tipo de ruido en los datos y es cuando, en una instancia etiquetada (es decir, un dato al que se le asocia un concepto clave), hay un error en la asignación de dicha etiqueta. Este tipo de ruido tiene un claro impacto en el proceso de aprendizaje dado que se está indicando que ciertos valores de atributos representan una cosa, cuando en realidad no lo es.

F1	F2	F3	F..
2.0	=	4.0	...
9.0	10.0	3.0	...
5.0	17.0	+	...

Figura 2.3: Ruido.

Pequeños disjuntos [HAP89]. La presencia de esta complejidad de los datos sucede cuando las instancias representantes de un mismo concepto no se encuentran en una región homogénea en el espacio de entrada. Esta característica se muestra en la Figura 2.4 y, en general, se la vincula como consecuencia de otras de las mencionadas anteriormente dado que, en muchos casos, surge de la presencia, por ejemplo, de la esparcidad y escasez presentes en los datos no balanceados, del solapamiento de instancias y de la presencia de un pequeño grupo de instancias ruidosas en los datos.

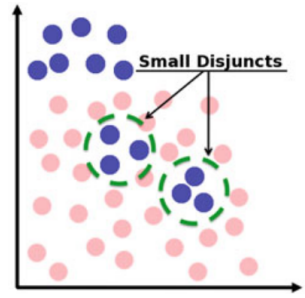


Figura 2.4: Pequeños disjuntos.

Falta de datos o rareza de los datos [Wei04; HBK19; Has+20]. Como su nombre lo sugiere, esta complejidad se relaciona a la falta de información o densidad en los datos. Es decir, no hay suficientes datos para que los algoritmos de aprendizaje lleven a cabo las generalizaciones sobre el concepto subyacente, produciendo modelos pobres. En la Figura 2.5 se muestra un ejemplo de la falta de datos de las instancias representadas con color rojo.

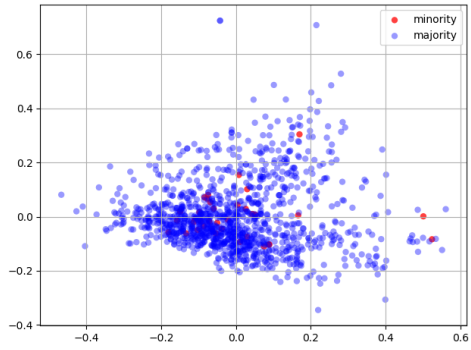


Figura 2.5: Falta o rareza de los datos.

Redundancia [Reh+16]. Se trata de las instancias que representan la misma información o concepto. En la Figura 2.6 se muestra un ejemplo de la redundancia en un conjunto de datos, asociando cada color a un tipo único de instancias. Se puede tener una redundancia estricta en términos de instancias repetidas y/o una redundancia conceptual, sin necesidad de que cada una de las instancias redundantes tengan exactamente los mismos valores de atributos generando áreas de muestras "sobre-representadas". La presencia de redundancia en un conjunto de datos genera además un impacto negativo en cuanto a los recursos computacionales requeridos (almacenamiento, memoria principal, capacidad de procesamiento, entre otros), como a los tiempos de ejecución necesarios para procesar datos que representan lo mismo (y no agregan información valiosa).

F1	F2	F3	F4	Concepto
azul	3.5	2	0.24	tipo1
verde	6.75	23	0.55	tipo2
verde	6.75	23	0.55	tipo2
negro	1.0	17	0.32	tipo3
verde	6.75	23	0.55	tipo2
azul	3.5	2	0.24	tipo1

Figura 2.6: Redundancia.

Alta dimensionalidad [ZOT14]. Se presenta cuando las instancias de un conjunto de datos tienen un número significativamente elevado de atributos. En escenarios de pequeñas cantidades de datos, hasta podría suceder que el número de características exceda al número de instancias de dicho conjunto de datos. En la mayoría de los casos donde las instancias contienen una alta dimensionalidad de características, muchas de ellas aportan muy poco o nada al proceso de extracción de conocimiento. Esta situación se puede observar en la Figura 2.7, en donde las columnas coloreadas corresponden a los atributos con poder discriminante. Así pues, de manera similar a la redundancia, la alta dimensionalidad impone además restricciones en los tiempos de procesamiento de los datos como en disponibilidad de infraestructura, cuando se podrían mitigar o evitar. Particularmente estas dos características de los datos se encuentran más frecuente en escenarios de grandes volúmenes de datos, donde en muchos conjuntos se replican datos y/o se tienen en cuenta una gran cantidad de atributos tan sólo para conseguir generar un conjunto "Big Data". En el Capítulo 5 se profundiza sobre este tema.

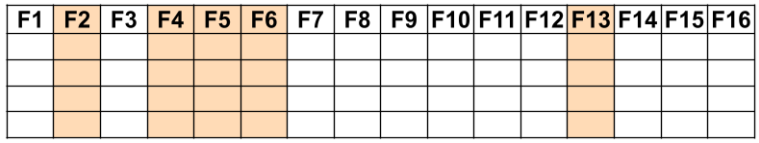


Figura 2.7: Alta dimensionalidad.

Datos no balanceados [CJK04; Fer+18b]. Ocurre cuando existe una importante desigualdad o desequilibrio al contabilizar la cantidad de instancias que representan a cada uno de los conceptos claves de un problema. En la Figura 2.8 se presenta un ejemplo de esta característica intrínseca de los datos. La extracción de conocimiento a partir de datos no balanceados es un tema significativo debido a la gran cantidad de problemas reales en los que el concepto clave de interés está representado por una pequeña porción de los datos.

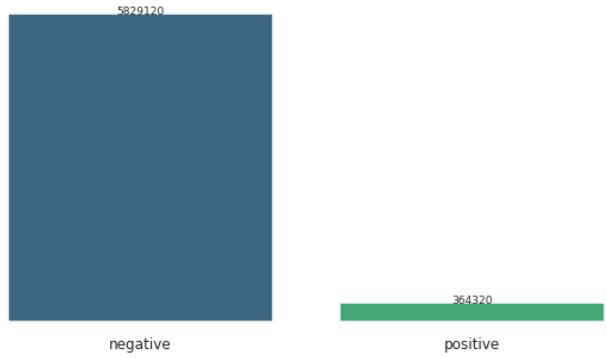


Figura 2.8: Datos no balanceados.

Solapamiento [PBM04; Fer+18b]. Se da cuando se tienen dos o más instancias etiquetadas con los mismos o valores de atributos muy próximos, pero sus etiquetas corresponden a distintos conceptos claves (instancias contradictorias). En la Figura 2.9 se muestra esta situación. Se dice que dichas instancias son contradictorias, conflictivas o ambiguas porque cubren el mismo espacio del problema pero representan a distintos conceptos claves del mismo. En otras palabras, cuando los valores de las características de las instancias no son lo suficientemente discriminantes para distinguir entre distintos conceptos.

F1	F2	F3	F4	Concepto
verde	6.75	23	0.55	tipo2
verde	6.75	23	0.55	tipo1
verde	6.75	23	0.55	tipo2

Figura 2.9: Solapamiento.

2.2. Análisis exploratorio de datos

Al trabajar con un conjunto de datos, y con el fin de descubrir sus características intrínsecas, un típico primer paso es llevar a cabo un Análisis Exploratorio de Datos (*Exploratory Data Analysis* (EDA)) [Tuk08]. Si bien no hay una serie de pasos prefijados para este tipo de análisis, en general se busca obtener información estadística de los datos, llevar a cabo un análisis visual para conocer la distribución de los mismos (ya sea por cada atributo, como también discriminando por cada concepto clave del problema), como también se pueden incluir métricas de complejidad [Lor+19] que incluyen información que puede ser de utilidad.

Respecto a la estadística descriptiva, es útil contar con información sobre la tendencia central, la dispersión de los datos, y su distribución. Por su parte, el análisis mediante gráficos también contribuye a vislumbrar algunos aspectos tales como valores atípicos o correlaciones entre variables. Existe una enorme cantidad de alternativas sobre visualización de los datos que se pueden utilizar para profundizar el conocimiento acerca de las complejidades de un conjunto de datos. De manera similar, se encuentra en la bibliografía un grupo de medidas de complejidad de los datos las cuales se enfocan en general en el solapamiento y en la proporción de los conceptos claves del problema. De este modo, lo que se busca es tener una estimación sobre las causas de la dificultad del problema a resolver para así aplicar las técnicas de preprocesamiento más apropiadas, y mejorar la calidad de los modelos aprendidos. A continuación se describen algunas de las métricas más utilizadas y con mayor relevancia para los objetivos de la presente tesis doctoral. Sus respectivas fórmulas se omiten deliberadamente para facilitar la presentación, sin embargo se pueden consultar en [Lor+19].

F1. Máxima relación discriminante de Fisher Esta métrica mide el solapamiento entre las características de los diferentes conceptos claves del problema. En concreto, calcula el solapamiento de cada característica por separado y toma la más baja o restrictiva de todas, que está asociada de manera inversa al valor más alto calculado para F1.

La F1 es una métrica de complejidad con un coste computacional reducido, y esto la hace de gran interés para los problemas con gran volumen de datos. Sin embargo, sólo estudia los solapamientos entre instancias de diferentes conceptos claves del problema, considerando una única característica. Esto disminuye la calidad de la información extraída al tener una visión univariante.

F2. Volumen de la región solapada La métrica F2 calcula el solapamiento entre las instancias de los diferentes conceptos claves del problema. En este caso, considera el dominio (valores máximos y mínimos) de todas las características. Por este motivo, se denomina "Volumen" de la región de solapamiento.

Aunque tiene un coste computacional mayor que F1, representa una simulación más realista del funcionamiento de los algoritmos de aprendizaje porque considera múltiples características. Sin embargo, no cuenta el número de instancias afectadas en el área de solapamiento, sólo considera el dominio de solapamiento.

F3. Máxima eficiencia de las características individuales La base de esta métrica de complejidad es dar cuenta de si los conceptos claves del problema son linealmente separables por una sola característica. Para ello, calcula la relación entre los ejemplos que no están en la zona de solapamiento y el número total de ejemplos.

F3 es una métrica de complejidad restrictiva, porque considera su separabilidad por una sola característica, mientras que usualmente los modelos construidos por la mayoría de los algoritmos para la extracción de conocimiento tienden a emplear todas las características y la relación entre ellas.

La métrica F3 aborda la principal desventaja de las métricas F1 y F2 al contar el número de instancias afectadas. Sin embargo, tal como se ha comentado, tiene la misma problemática que F1 ya que tiene en cuenta una única característica.

F4. Eficiencia de las características colectivas La métrica F4 es una extensión natural de la métrica F3, que añade un componente más restrictivo al considerar todas las características. El proceso de cálculo de la métrica consiste en los siguientes tres pasos iterativos: 1) Se calcula F3 para determinar qué característica es la más discriminatoria. 2) Se descartan las instancias que están fuera del área de solapamiento correspondiente a la característica seleccionada en el paso 1. 3) Se elimina la característica seleccionada en el paso 1 y se repite el procedimiento hasta que se consideren todas las características.

La métrica F4 es la más apropiada en la literatura para estudiar la complejidad de un problema de clasificación. Cuenta el número de muestras afectadas y también considera todas las características en un proceso iterativo. Como desventaja,

es la más lenta de las métricas propuestas debido a su proceso de recuento e iteración.

C2. Ratio de desequilibrio Es la métrica más utilizada en la literatura para medir el desequilibrio entre conceptos claves de un problema.

C2 es una métrica importante por la información que brinda y su rápido cálculo. Proporciona el ratio entre el número de instancias (`#mayor/#menor` o `mayor:menor`) de los diferentes conceptos claves del problema. Típicamente se suele considerar que un conjunto es no balanceado cuando el ratio está por encima de 2:1.

2.3. Comentarios del capítulo

En este capítulo se han presentado las principales complejidades que se pueden encontrar en un conjunto de datos. La presencia de éstas implica una degradación en mayor o menor medida de la calidad de los datos, que tiene como consecuencia un desempeño más pobre en el proceso de extracción de conocimiento. Tanto su impacto, como la forma de tratar a dichas características intrínsecas, tendrá relación con su tipo, grado de presencia, como también por la combinación que se de entre ellas. En esta tesis el foco está puesto en los datos no balanceados, en la redundancia, la alta dimensionalidad y el solapamiento, para un tipo de problemas de datos en particular, como se describe y justifica más adelante en los sucesivos capítulos.

Además, en este capítulo se han mencionado algunos de los pasos que suelen formar parte del Análisis Exploratorio de Datos, una típica tarea inicial al trabajar con un conjunto de datos que tiene por objetivo obtener información de interés.

3 Aprendiendo de los datos

Los conjuntos de datos representan distintos escenarios a partir de los cuales se busca obtener conocimientos de utilidad. Para esto, es necesario emplear técnicas o algoritmos que aprendan sobre dichos datos. Dentro de las distintas áreas que se ocupan de esto, Machine Learning es de las más populares. A su vez, la tarea de clasificar datos es ampliamente utilizada en el proceso de extracción de conocimiento y, puntualmente, es una de las tareas abordadas por ML. Para sentar las bases conceptuales sobre el aprendizaje supervisado en general, y la clasificación en particular, que son eje central de esta tesis doctoral, en primer lugar, se presenta una visión general de ML en la Sección 3.1. A continuación, la Sección 3.2 se centra en el tema de la clasificación de datos por ser el escenario elegido en el cual se enfoca esta tesis. Particularmente, se introducen los conceptos fundamentales sobre la predicción de clases, y se presentan algunos de los algoritmos más populares para generar modelos de clasificación. Luego, en la Sección 3.3, se describen las métricas normalmente utilizadas para evaluar el rendimiento de un algoritmo de este tipo, así como las técnicas de validación de los resultados obtenidos. Finalmente, en la Sección 3.4, se comenta brevemente el capítulo.

3.1. Aprendizaje Automático

El ML [Mit13] es una disciplina que provee algoritmos los cuales extraen automáticamente conocimientos a partir de un conjunto de datos. Típicamente, los algoritmos realizan su trabajo en dos etapas que pueden ser llevadas a cabo en diferentes momentos. En primer lugar, se ejecuta un proceso de aprendizaje o entrenamiento, y en segundo lugar, una tarea de predicción.

En relación al aprendizaje, un algoritmo normalmente sigue un enfoque supervisado, no supervisado o semi-supervisado. El aprendizaje supervisado parte de un conjunto de instancias de entrenamiento del cual se conoce la salida esperada para cada una de ellas. A estas salidas se las denominan "etiquetas o clases" y son generadas previamente mediante la intervención de una persona experta en el campo de estudio del que provengan los datos, por ejemplo. El objetivo del aprendizaje es tratar de determinar las relaciones existentes entre los valores de los atributos y los valores de salidas. Por su parte, y en contraposición al aprendizaje supervisado, el aprendizaje no supervisado no requiere conocer las etiquetas

de los datos de entrenamiento. En este caso, como su nombre lo indica, no está supervisado por salidas preestablecidas, de hecho las mismas no se encuentran definidas. El objetivo de este tipo de algoritmos es descubrir relaciones implícitas presentes en los datos. A su vez, el enfoque de aprendizaje semi-supervisado [ZG09] es una combinación de los dos anteriormente nombrados. Una porción reducida de los datos posee su valor de salida definido, mientras que no es así para el resto de las instancias. Por consiguiente, se deben aplicar técnicas de ambos enfoques involucrados con el fin de obtener información valiosa. Un clásico ejemplo de este tipo de aprendizaje se da en los problemas de auto-etiquetado de los datos [TGH15] donde se busca automatizar la tarea de etiquetar instancias a partir de un pequeño subconjunto de datos manualmente etiquetados.

En todos los casos antes mencionados, la etapa de aprendizaje genera un modelo que luego es utilizado en la etapa de predicción. Respecto a los datos de entrada, se tiene el conjunto de entrenamiento a utilizar en la etapa de aprendizaje, y el conjunto de prueba que consiste en datos desconocidos o no vistos en el entrenamiento, y que son utilizados para la segunda etapa. A partir de ambos conjuntos de datos, se podrá evaluar el desempeño del modelo generado.

Mediante los distintos enfoques, ML ofrece una amplia gama de algoritmos. Así pues, para el aprendizaje supervisado se pueden distinguir dos grandes tareas: Regresión [CM07] y Clasificación [RS01]. La primera consiste en predecir el valor de salida (valor en un dominio continuo) para nuevos ejemplos mediante una función que expresa la relación entre los datos de entrada con la salida; mientras que la segunda busca determinar cuál es la clase (de entre una cantidad finita de etiquetas de clase conocidas) a la que una instancia nunca vista debería pertenecer en base al aprendizaje que se pudo llevar a cabo con los datos disponibles de entrenamiento. En el caso de la regresión, un ejemplo podría ser explicar el precio de las propiedades en una región determinada a partir de la información de ciertas características como el tamaño, cantidad de ambientes, de baños, si posee patio, etc. Para el caso de la clasificación, un problema típico puede ser la presencia o no de una enfermedad a partir de la información de síntomas, resultados de estudios, edad, etc. Respecto al aprendizaje no supervisado, también se distinguen dos grupos de problemas: Agrupamiento [Har75] y Asociación [AIS93]. El agrupamiento de los datos, también conocido como clustering o segmentación, se enfoca en el descubrimiento de grupos naturales en los datos, es decir, permiten conocer la estructura de los mismos. Para esto, se basan en la similitud entre las instancias, buscando que los miembros (instancias) pertenecientes a cada grupo sean lo más parecidas posible, pero que exista la máxima separación posible entre los diferentes grupos. Por su parte, la asociación busca identificar relaciones presentes entre los datos, por ejemplo, conocer la correlación existente entre los valores de los atributos del problema.

En esta tesis, el objetivo está en la resolución de la tarea de clasificación; para ello, se llevará a cabo una comparativa sobre el comportamiento de un mismo clasificador de características dadas, pero desde dos puntos de vista diferentes: datos en crudo (*raw*) frente a datos preprocesados. De este modos se pretende demostrar la importancia del tratamiento de la complejidad de los datos para facilitar la construcción de un modelo robusto y preciso. Dichas técnicas de preprocesamiento se describen más adelante.

3.2. Clasificación de los datos

Como se ha comentado anteriormente, dentro del área del aprendizaje supervisado de ML, la clasificación de datos es una tarea muy extendida en donde los algoritmos aprenden a partir de datos etiquetados con el objetivo de predecir la etiqueta de clase de datos no vistos. Para ello, existen diferentes paradigmas de algoritmos de clasificación que se utilizan en múltiples campos de aplicación [BB15; BB20]. La medicina, el descubrimiento de nueva física, la bioinformática, y la seguridad informática, son sólo ejemplos de algunas de las áreas en donde se presentan problemas de clasificación de datos.

Por lo tanto, en un conjunto de datos que represente un problema de clasificación de un espacio de entrada X, existe una instancia (u observación) $x \in X$ que puede expresarse como un vector de atributos de longitud $|A|$, siendo A un conjunto de atributos o características descriptivas. A su vez, cada instancia, x_T, del conjunto de entrenamiento T ($T \subset X$) tiene asociado un valor categórico que representa su etiqueta de clase C_{x_T}, la cual pertenece al conjunto finito C de las posibles clases del problema. Un problema es binario cuando el conjunto de etiquetas C está conformado por dos elementos, o multiclase se constituye por tres o más.

Entonces, el objetivo de una tarea de clasificación es mapear los elementos del espacio de entrada con los de C ($f : X \rightarrow C$), a partir del aprendizaje sobre los datos de entrenamiento T. El mapeo de los datos puede llevarse a cabo mediante distintos enfoques, algunos de ellos se comentan más adelante. En cualquiera de los casos, se busca conseguir un modelo que se ajuste correctamente frente a datos nunca antes vistos, es decir, que tenga una buena capacidad de generalización; además, otra característica de interés será que las predicciones que se obtengan sean precisas. Para llevar a cabo esto, se debe encontrar una aproximación adecuada de la frontera o límite de decisión entre las clases. Según la complejidad del modelo, se puede dar lugar a un sobre- o un infra- ajuste. Por un lado, el sobre-ajuste (overfitting), es cuando se genera un modelo altamente complejo el cual se adapta tan precisamente a los datos de entrenamiento, que no le permite ser flexible frente a nuevos datos y genera errores al querer cla-

sificarlos. Por otro lado, el infra-ajuste (underfitting) se da a partir de modelos extremadamente sencillos conseguidos del aprendizaje pobre de los datos de entrenamiento. El resultado de este fenómeno también es una mala predicción de las nuevas instancias. Un ejemplo de esto se puede observar en la Figura 3.1 donde se muestran las fronteras de decisión de un problema binario (círculos y cruces como clases) y con dos atributos (X_1 y X_2), frente a una situación de sobreajuste, ajuste óptimo (o adecuada generalización) e infraajuste.

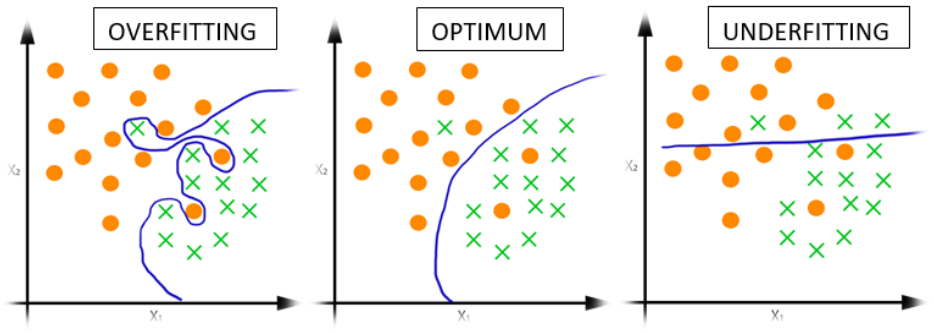


Figura 3.1: El problema de la generalización. Fronteras de decisión entre las clases de un problema binario.

3.2.1. Modelos de clasificación

En la literatura existe una amplia variedad de algoritmos clasificadores que abarcan la tarea desde distintos enfoques. A continuación se describen aquellos modelos de clasificación que son usados en las propuestas y estudios experimentales presentados en los siguientes capítulos de esta tesis, a saber, *k - Nearest Neighbors* (*k*NN), *Decision Tree* (DT), y *Random Forest* (RF).

k - Nearest Neighbors (*k*NN) [CH67]

El algoritmo de los k vecinos más cercanos, con $k \geq 1$ y $k \in \mathbb{Z}$, es un algoritmo basado en instancias lo que significa que se basa en la mayor similitud [Che+09] o menor distancia [WS09] de una nueva instancia con cada una de las instancias del conjunto de entrenamiento, para lo cual suele emplearse la distancia euclídea. Así pues, se ordenan las instancias según la similitud/distancia y se seleccionan las k instancias más cercanas, determinando así el vecindario de cada ejemplo a clasificar. A continuación, y para definir la etiqueta de la nueva instancia, el criterio se basa en asignarle la etiqueta con mayor ocurrencia entre las instancias de su vecindario. De lo anterior se observa que este tipo de algoritmo carece

de una etapa de entrenamiento[7], por lo que todo el trabajo se realiza en la etapa de predicción/clasificación y, para ello, requiere que todos los datos de entrenamiento y prueba se encuentren en memoria principal.

En la Figura 3.2[8] se muestra un ejemplo gráfico de la clasificación de una instancia nueva (punto verde) siguiendo el modelo de *k*NN para un problema de clasificación binaria (las clases son "triángulo rojo" y "cuadrado azul"). En la imagen se pueden distinguir dos vecindarios, uno está delimitado por el círculo de línea sólida para cuando $k = 3$ y el otro por el círculo de línea discontinua para cuando $k = 5$. Así pues, basándose en la ocurrencia de clases de las instancias etiquetadas dentro de cada vecindario, para el primer caso, el nuevo ejemplo es clasificado con la clase "triángulo rojo" y para el segundo caso, se clasifica como "cuadrado azul".

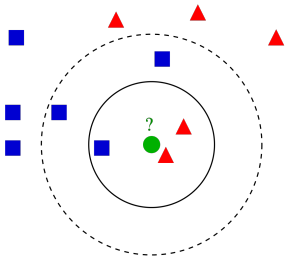


Figura 3.2: Ejemplo gráfico del algoritmo *k - Nearest Neighbors.*

Es necesario comentar que este algoritmo, por su naturaleza de trabajar con vecindarios, presenta una alta complejidad computacional respecto al tiempo de ejecución y que, para poder mejorar su rendimiento deben poder alojarse en memoria principal tanto el conjunto de datos de entrenamiento como el de prueba. Esto puede no ser un problema en conjuntos de datos de tamaño tradicional (Small Data), pero en escenarios de volúmenes de grandes datos (Big Data) es una cuestión importante a tener en cuenta para contar con el acceso a los recursos necesarios que estas restricciones imponen (véase Capítulo 5).

Decision Tree (DT) [RM14; Qui93]

Otro enfoque de clasificación popular consiste en construir un árbol de decisión basado en los datos de entrenamiento. El algoritmo del árbol de decisión es reconocido como uno de los diez algoritmos más influyentes en el descubrimiento del conocimiento [WK09], debido a su efectividad y su diseño intuitivo.

Un árbol es un grafo acíclico conectado, donde el recorrido parte desde la raíz, pasando por nodos internos y llegando a nodos terminales u hojas. Los nodos hojas representan las asignaciones de las etiqueta de clase, y las ramas o conexiones que unen cada nodo, representan las conjunciones de atributos que conducen a una cierta hoja, es decir, a una cierta etiqueta de clase. Dicho de otro modo, se particiona repetidamente los datos de entrada, con el objetivo de que

7 Por esta razón, los algoritmos de este tipo son denominados como algoritmos de aprendizaje perezoso (*"lazy learners"*).

8 Antti Ajanki AnAj, CC BY-SA 3.0, via Wikimedia Commons.

cada partición contenga elementos de una sola clase (nodo puro). Así pues, la raíz inicialmente contendrá a todo el conjunto de entrenamiento completo y, sobre la base de una medida de impureza, se selecciona la mejor característica y la división correspondiente para continuar con el particionado de los datos. Por lo tanto, para clasificar una nueva instancia, se recorre por el árbol, empezando desde la raíz y se sigue el camino según los valores de atributos que dicha instancia posea, hasta llegar a una hoja. La clase vinculada a dicha hoja es la que se asigna como etiqueta a la nueva instancia.

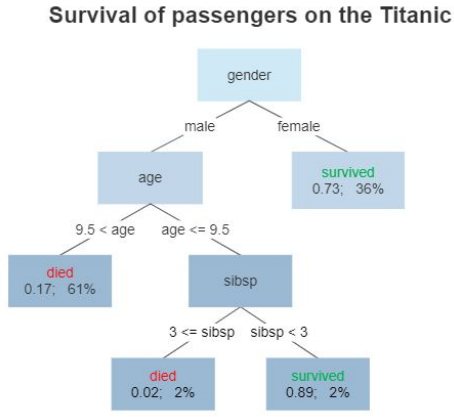


Figura 3.3: Ejemplo gráfico del algoritmo *Decision Tree* para la clasificación de un problema de dos clases.

En la Figura 3.3[9] se muestra el DT para la clasificación de un problema de dos clases. Puntualmente, el problema del que se trata es de predecir qué tipos de pasajeros sobrevivieron al naufragio del Titanic[10], basado en su género (*gender*), edad (*age*) y la cantidad de personas allegadas a bordo (pareja o hermanos/as, *sibsp*). En consecuencia, dada una nueva instancia se analiza en primer lugar su característica categórica "gender" y, dependiendo su valor, se continúa por una de las dos ramas. En el siguiente nivel del árbol se llega o bien a una hoja (indicando que sobrevivió) o bien a un nuevo particionado basado en el valor del atributo numérico "age". El razonamiento continua del mismo modo hasta recorrer el camino completo raíz - hoja según los valores de la instancia que se esté clasificando.

Para guiar al proceso de selección de las divisiones, con el fin de reducir la

9 Gilgoldm, CC BY-SA 3.0, via Wikimedia Commons.
10 https://www.kaggle.com/c/titanic/

heterogeneidad de clases de los elementos asociados a un nodo, se emplea una medida de impureza de nodo, la cual se basa en la distribución de clases existente en el mismo. Se dice que la impureza es mínima, o que el nodo es puro, cuando se está en presencia de una sola clase. En cambio, la impureza es máxima, cuando las clases están uniformemente distribuidas en el nodo. Normalmente, la impureza se mide utilizando el índice Gini o la entropía de Shannon [Sha48]. Hay que tener en cuenta que, si bien se podría pensar que lo ideal es contar con nodos terminales puros, el construir un árbol hasta dicho punto implica un mayor riesgo de sobreajustar los datos de entrenamiento y, por lo tanto, obtener un modelo demasiado complejo que no logre una buena generalización frente a datos nuevos. Una estrategia a llevar a cabo para evitar esta situación es aplicar un procedimiento de poda [Qui87], del cual existen dos enfoques. Por un lado, detener el crecimiento del árbol cuando un nodo alcanza un cierto grado de mínima impureza o cuando representa a muy pocas instancias de entrenamiento (enfoque de prepoda); por otro lado, construir el árbol hasta el punto en que todas sus hojas sean puras para luego eliminar aquellas ramas que minimicen el error de clasificación en un conjunto de prueba (enfoque de postpoda).

Random Forest (RF) [Bre01]

El algoritmo RF sigue un enfoque de clasificación basado en aprendizaje en conjunto (*ensemble learning* [Pol12; Zho12; Kun14]). Dos de las técnicas más populares para la construcción de un *ensemble* son *bagging* [Bre96] y *boosting* [Sch03]. En el caso de *bagging*, el clasificador se construye utilizando un conjunto de un mismo tipo de algoritmo de aprendizaje, donde cada uno de ellos es entrenado a partir de una submuestra aleatoria con reemplazo del conjunto de entrenamiento original. Por su parte, las técnicas de *boosting* utiliza, en cada iteración, los modelos de manera secuencial, existiendo una dependencia entre ellos mediante la reponderación de los datos de entrenamiento. Esto se lleva a cabo incrementando el peso de aquellas instancias mal clasificadas en la iteración anterior, con el fin de darle mayor importancia en la siguiente. Usualmente, tanto para la clasificación siguiendo ensembles bagging como boosting, la salida final es obtenida mediante votación.

En particular, RF es el típico algoritmo de *bagging*, que combina múltiples DTs. En la Figura 3.4[11] se muestra el proceso de entrenamiento y de clasificación utilizando un RF para un problema de clasificación binaria (las clases están representadas por el color rojo y el verde). En la parte A, se muestra la construcción de cada DT a partir de una muestra aleatoria de los datos originales. Mientras que en la parte B, se muestra el procedimiento, culminante en la votación, para

11 Machado [MMC15], material con licencia CC BY 4.0.

determinar la clase de las instancias nuevas. La lógica para recorrer cada DT con el fin de asignar una etiqueta de clase a una instancia, se ha descripto en el apartado anterior. Al final del proceso, cada DT emite un voto para la etiqueta de clase preferida, y la predicción final es elegida a partir de la etiqueta que más votos obtuvo.

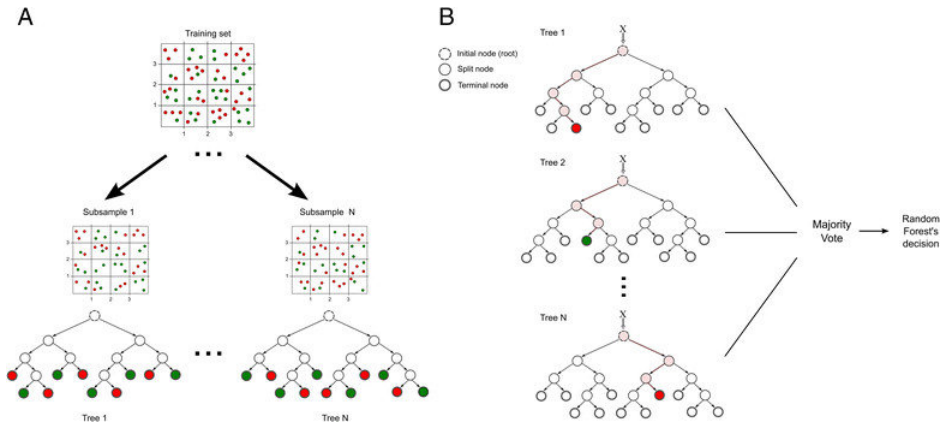


Figura 3.4: Ejemplo gráfico del algoritmo *Random Forest* para la clasificación de datos. En la parte A, se muestra la construcción de cada DT a partir de una muestra aleatoria de los datos originales. Mientras que en la parte B, se muestra el procedimiento, culminante en la votación, para determinar la clase de las instancias nuevas.

3.3. Métricas de evaluación de la calidad predictiva

Dado un modelo predictivo, es necesario contar con las medidas adecuadas para poder mensurar su desempeño en la tarea que lleva a cabo (por ejemplo, clasificar). Sin embargo, no todas las métricas que se presentarán a continuación se puede aplicar a cualquier tipo de problema de clasificación indistintamente. Por lo tanto, hay que prestar especial atención sobre qué métricas utilizar en cada caso. Estas consideraciones se comentan conforme se avanza en la presente sección.

En primer lugar, es necesario contar con información acerca de las predicciones realizadas para cada clase como de sus valores reales. Típicamente, en ML esta información se presenta mediante una matriz de confusión [Ste97], la cual organiza las instancias de cada clase según su correcta o incorrecta identificación. En la Tabla 3.1 se muestra una matriz de confusión para un problema binario[12][13],

12 Todas las métricas que se muestran a continuación son para problemas binarios dado que son los que se utilizan en esta tesis. Para problemas multiclase se sigue la misma lógica.

13 Salvo que se especifique puntualmente, el dominio de todas las métricas es real $[0, 1]$ y propor-

en donde las clases son comúnmente denominadas como "positiva" y "negativa". En un problema de clasificación con datos no balanceados, se asocia comúnmente a la clase menos representada o minoritaria como la positiva, y a la mayoritaria como la negativa. Las filas y las columnas de la matriz se corresponden al valor de clase real (*Actual*) y al valor predicho (*Predicted*), respectivamente. En las celdas se tienen el número de verdaderos positivos (*True Positive* (TP)), verdaderos negativos (*True Negative* (TN)), falsos positivos (*False Positive* (FP)) y falsos negativos (*False Negative* (FN)).

Tabla 3.1: Matriz de confusión para la evaluación del rendimiento de un problema de clasificación binaria.

Actual	Predicted	
	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

A partir de de lo anterior, hay cuatro métricas que describen a ambas clases de forma independiente:

- Tasa de verdaderos positivos (*True Positive Rate* (TPR)), es el porcentaje de instancias positivas correctamente clasificadas y se define como:

$$TPR = \frac{TP}{TP + FN}$$

- Tasa de verdaderos negativos (*True Negative Rate* (TNR)), es el porcentaje de casos negativos correctamente clasificados y se define como:

$$TNR = \frac{TN}{FP + TN}$$

- Tasa de falsos positivos (*False Positive Rate* (FPR)), es el porcentaje de instancias negativas clasificadas erróneamente y se define como:

$$FPR = \frac{FP}{FP + TN}$$

cional a los valores que toma, lo que significa que si el valor resultante es alto, el desempeño también lo es.

- Tasa de falsos negativos (*False Negative Rate* (FNR)), es el porcentaje de casos positivos mal clasificados y se define como:

$$FNR = \frac{FN}{TP + FN}$$

Una métrica típica para evaluar los modelos de clasificación cuando el problema es binario es la Exactitud (*Accuracy* (ACC)), definida por la Ecuación (3.1).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Sin embargo, es importante tener en cuenta que la medida ACC no es adecuada cuando los datos del conjunto están sesgados. A modo de ejemplo, se muestra el conjunto de datos no balanceado HIGGS_IR_16[14] del repositorio KEEL [Tri+17]. Su desproporción entre clases se observa en la Figura 3.5, y la matriz de confusión de entrenar un modelo DT y predecir sobre los datos de prueba, se muestra en la Tabla 3.2. De aplicar la Ecuación (3.1), se obtiene que $ACC = 0{,}9416$, es decir, que la precisión del modelo es significativamente alta. Sin embargo, vemos que $TPR = 0{,}0121$ y $TNR = 0{,}9998$, por lo que la bondad del modelo sólo se desempeña correctamente en predecir la clase mayoritaria y, cabe recordar, que en conjuntos de datos no balanceados el concepto de interés suele estar representado por la clase con menos instancias.

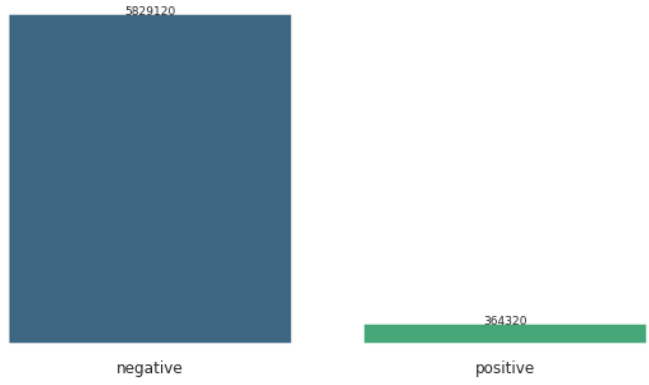


Figura 3.5: Distribución de clases del conjunto HIGGS_IR_16.

Por lo anterior, a partir de los índices que se obtienen de la matriz de confusión, se definen métricas más robustas para evaluar el rendimiento en problemas de

14 En el Apéndice A se muestra más información de cada uno de los conjuntos de datos utilizados o mencionados a lo largo de esta tesis.

Tabla 3.2: Matriz de confusión para el problema no balanceado HIGGS_IR_16.

Actual	Predicted	
	Positive	Negative
Positive	881	72022
Negative	288	1165497

clasificación no balanceada. Dos de las más utilizadas son la Media Geométrica (*Geometric Mean* (GM)) [Bar+03] y el área bajo la curva ROC (*Area under the ROC Curve* (AUC)) [JL05]. La GM intenta maximizar la precisión de cada una de las dos clases al mismo tiempo y se define por la Ecuación (3.2). La AUC evalúa qué modelo es mejor en promedio y se calcula aproximando la curva ROC continua mediante un número finito de puntos [Faw06], utilizando la regla del trapecio [MG02].

$$GM = \sqrt{TPR * TNR} \tag{3.2}$$

Entonces, para el ejemplo anterior de HIGGS_IR_16, se tiene que $GM = 0{,}1099$ y $AUC = 0{,}5059$ y, contrariamente a lo que sucedía con la ACC, estos valores bajos reflejan las dificultades que tiene el modelo de haber aprendido de datos no balanceados, lo cual manifiesta la necesidad de preprocesar el conjunto de datos para poder identificar correctamente a las instancias.

3.3.1. Métodos de validación

Anteriormente se mencionó la existencia de dos conjuntos de datos, uno con instancias conocidas para el aprendizaje del modelo (conjunto de entrenamiento) y otro con instancias nuevas para su clasificación (conjunto de prueba). En principio se podría pensar que la evaluación del modelo podría llevarse a cabo sólo utilizando los resultados de predecir sobre los propios datos de entrenamiento y comparar así los valores reales con los predichos. Sin embargo, se estaría perdiendo de vista el fenómeno de sobreaprendizaje, comentado en la Sección 3.2, que puede llevar a cabo un modelo. Por este motivo es que se necesita un conjunto independiente de instancias, es decir, que no haya sido utilizado durante el entrenamiento, y que permita evaluar la capacidad de generalización de un clasificador frente a datos no vistos. Para esto se requiere conocer las etiquetas de clase reales de los datos del conjunto de prueba, dado que se debe comparar

lo real con lo predicho. Los esquemas de validación [Kim09; Koh95] asisten a este objetivo. A continuación, se comentan dos métodos de los más relevantes y ampliamente utilizados.

Método de retención (*Holdout*)

Es el esquema de validación más básico que consiste en dividir el conjunto de datos (con datos etiquetados) en dos partes, una para el entrenamiento y otra para la prueba. La división suele llevarse a cabo de manera que la mayor parte sea para el conjunto de entrenamiento (típicamente, un 70 u 80 % de los datos). Por lo tanto, el modelo predictivo se entrena con los datos de aprendizaje y predice sobre los datos de prueba. Dado que estos últimos pertenecen a un subconjunto del conjunto original etiquetado, ahora se dispone de la información necesaria para poder comparar lo real con lo predicho. En caso de repetirse este método tomando cada vez un subconjunto diferente de los datos como datos de entrenamiento, estamos en presencia del *método de retención repetida*.

Método de validación cruzada de *k-fold* (*k-fold cross validation*)

Dada la robustez que presenta este esquema de validación, lo convierte es uno de los más utilizados para evaluar la capacidad de generalización de un modelo predictivo. La idea general es dividir los datos en *k-folds*[15] aproximadamente iguales, con $k \in \mathbb{Z}$, como se puede observar en la Figura 3.6. Típicamente se define a k como 5 o 10, sin embargo aún no existe una regla general para establecer su valor [ZY15]. A partir de las divisiones anteriores, se determinan k experimentos o iteraciones (representados por cada fila de la imagen). Así, por cada uno de los experimentos y sus *folds* se tiene un conjunto de prueba (correspondiente a un sólo *fold*, en color violeta en la figura) y un conjunto de entrenamiento agrupando los $k-1$ *folds* restantes del experimento (en color verde). Por lo tanto, por cada iteración se evalúa la calidad predictiva sobre el conjunto de prueba y, finalmente, se lleva a cabo una agregación de los resultados obtenidos para conseguir un único valor final que represente el rendimiento de la clasificación.

Respecto a la manera en que se generan los *folds*, es recomendable que, si el conjunto de datos es no balanceado, se mantenga la proporción entre clases en cada uno de ellos, así se garantiza que tanto el conjunto de entrenamiento como el de prueba presenten la misma distribución de datos.

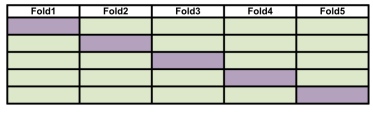


Figura 3.6: Método de validación cruzada de *k-fold*.

15 Se mantiene la palabra «*fold*» en inglés para evitar confusión con «particiones» de Apache Spark, que se verá más adelante.

3.4. Comentarios del capítulo

En este capítulo el foco se ha puesto en el aprendizaje a partir de los datos, con el fin de obtener conocimiento valioso sobre los problemas que ellos representan. Para llevar a cabo dicho objetivo, *Machine Learning* es una disciplina de gran popularidad y que provee una amplia gama de algoritmos para la extracción de conocimiento y lo hace mediante distintos enfoques (supervisado, no supervisado, semi supervisado, entre otros). Así pues, cuando se cuenta con datos etiquetados, es decir, que se conoce su salida esperada, y se busca modelar la relación entre los valores que conforman al dato y su etiqueta para poder predecir dicha relación frente a nuevos datos, se pueden aplicar algoritmos de aprendizaje supervisado para la clasificación de datos. Dada la relevancia de los problemas de clasificación para esta tesis, en este capítulo se ha abordado el tema de la clasificación de datos con mayor detalle. A su vez, y dado que existen numerosos algoritmos clasificadores y que además se basan en distintos enfoques, se han comentado sólo aquellos que tienen relación con las propuestas y estudios experimentales que se describen a lo largo de esta tesis. Por otro lado, se detallaron las métricas para poder evaluar el desempeño predictivo de un modelo clasificador, como así también los métodos comúnmente empleados para la validación del modelo y su capacidad de predecir datos nuevos.

4 Preprocesamiento de los datos

Obtener un modelo que permita representar completamente los datos no es una tarea fácil. En concreto, hay muchos casos en los que la calidad de los datos afecta en gran medida a la fase de aprendizaje y, por consiguiente, a los resultados. Por lo tanto, debemos centrarnos en las complejidades o características intrínsecas de los datos descriptas en la Sección 2.1, principalmente, en la distribución sesgada de las clases que es una de las complejidades que, en conjunción con el resto de características, ocasiona una mayor degradación en el desempeño de un modelo [Fer+18b].

Así pues, luego de determinar cuáles de las características se encuentran presentes en un conjunto de datos, se deben emplear herramientas acordes con el fin de compensar sus efectos negativos, es decir, se deben aplicar técnicas de preprocesamiento de los datos [GLH15]. Particularmente, en esta tesis nos centramos en cuatro de las nueve complejidades de los datos descriptas anteriormente. En concreto, el foco está puesto principalmente en el desequilibrio de los datos (Sección 4.1), así como en la redundancia y la alta dimensionalidad de los datos (Sección 4.2), y en las zonas ambiguas o solapamiento de clases (Sección 4.3). Dado que son de frecuente aparición en los escenarios de clasificación de datos Big Data (foco de nuestra investigación), generan impacto negativo en el entrenamiento de los modelos y, en el caso de redundancia y alta dimensionalidad, implican una demanda respecto a almacenamiento, y capacidad y tiempo de cómputo, cuando en realidad no aportan al adecuado modelado de los datos para la extracción de conocimientos de calidad. Por último, se comenta el presente capítulo en la Sección 4.4.

4.1. Desequilibrio de clases

En el área de los problemas de clasificación, el escenario de conjuntos de datos no balanceados aparece con frecuencia en muchas aplicaciones reales, tales como, la detección de fraudes con tarjetas de crédito, experimentos de búsqueda de nueva física, diagnóstico médico de enfermedades raras, problemas de bioinformática, entre muchos otros [Guo+17]. La principal propiedad de este tipo de problemas de clasificación es que los ejemplos de una clase están significativamente subrepresentados respecto a los de las otras [Fer+18b; Lóp+13]. Sin embargo, y como puede reconocerse en la mayoría de los casos de estudios

mencionados anteriormente, la clase minoritaria suele representar el concepto más importante a aprender, y es difícil identificarla porque puede estar asociada a casos excepcionales, o porque la adquisición de datos de estos ejemplos es costosa.

Dado que la mayoría de los algoritmos de clasificación no comprenden los sesgos en la distribución de los datos, esto puede generar modelos de clasificación subóptimos, es decir, una buena cobertura de los ejemplos mayoritarios, mientras que los minoritarios se clasifican erróneamente con frecuencia, pudiendo ser consideradas como ruido o valores atípicos si se encuentran en una zona del espacio muy poco representada por su clase (estos grupos de instancias también se conocen como "pequeños disyuntos" o *"small-disjuncts"* [Lóp+13]). Sin embargo, estos casos extremos pueden implicar un conocimiento importante y, por ende, es necesario tener cuidado de que tales clusters no sean descartados tan fácilmente. Por lo tanto, aquellos algoritmos que obtienen un buen comportamiento en el marco de la clasificación estándar, no necesariamente logran el mejor rendimiento para conjuntos de datos no balanceados [Fer+18b].

Para tratar el problema del desequilibrio de clases, se ha propuesto un gran número de enfoques [GSM12]. Estos enfoques se pueden clasificar en los siguientes grupos como se describen a continuación: **a)** los enfoques a nivel de algoritmos (enfoques internos); **b)** los métodos sensibles al costo; y **c)** los enfoques a nivel de datos (o enfoques externos).

Los primeros dos grupos gestionan el desequilibrio en los datos de manera interna al clasificador. El tercer grupo presenta un enfoque completamente distinto y se describe en mayor detalle dada la importancia que tiene en esta tesis.

4.1.1. Enfoques a nivel de algoritmos

Este tipo de técnicas compensa el desequilibrio de clases de manera interna, es decir, no hay manipulación de los datos con el fin de equilibrarlos sino que se modifica la forma de aprendizaje del clasificador [Len+08] para tener en cuenta el problema del desequilibrio de clases. Estos tipos de enfoques son muy dependientes del algoritmo clasificador a utilizar, por lo que se convierten en una opción de las menos elegidas como técnica de preprocesamiento ("inprocesamiento" en este caso), dado que hay que conocer en profundidad los pasos de cada algoritmo para poder llevar a cabo una adecuada adaptación para tratar con datos no balanceados en cada uno de ellos.

4.1.2. Métodos sensibles a los costes

Este tipo de soluciones denominadas comúnmente como *cost-sensitive* [Dom99] se relacionan de algún modo con el enfoque a nivel de algoritmo, en el sentido que los clasificadores tienen en cuenta los costes de clasificación errónea y, por supuesto, su objetivo es minimizarlos. Además, el coste de un falso negativo será diferente del coste de un falso positivo, es decir, hay errores que se penalizan más. Con esta estrategia de penalización se logra conseguir una mejor generalización de los datos y, por lo tanto, se minimizan la cantidad de instancias incorrectamente clasificadas. Para ello se pueden aplicar los costes a priori (ponderando las instancias según su clase, utilizando meta-costes), durante el aprendizaje (guiando el procedimiento de generación del modelo), o a posteriori (con técnicas de umbralización o "thresholding" para forzar la predicción hacia una de las clases).

4.1.3. Enfoques a nivel de datos

Así como su nombre lo indica, este tipo de soluciones se centran puntualmente en los datos. A partir del conjunto original con desequilibrio entre sus clases, se busca modificarlo de algún modo para que se obtenga un conjunto potencialmente balanceado. Este proceso se lleva a cabo sólo en el conjunto de entrenamiento y con el objetivo de re-equilibrar la distribución de datos a la proporción deseada (normalmente para alcanzar un ratio 1:1). Por lo tanto, primero se aplica alguna técnica de preprocesamiento sobre los datos no balanceados, y los datos resultantes de ello son la entrada a un algoritmo clasificador. Estos enfoques son más flexibles porque son independientes al clasificador y, por ende, ampliamente utilizados respecto al resto, dado que no se aplica modificación alguna sobre los algoritmos sino que éstos reciben un conjunto de datos acorde a lo que típicamente esperan: datos sin desproporción en sus clases.

En la literatura especializada, se puede encontrar trabajos sobre este tipo de técnicas que estudian el efecto de cambiar la distribución de clases para tratar conjuntos de datos no balanceados. Estos trabajos han demostrado empíricamente que aplicar un paso de preprocesamiento para equilibrar la distribución de clases suele ser una solución útil [GLH15]. A su vez, es de interés mencionar que dichas técnicas de preprocesamiento a menudo están diseñadas para que de manera complementaria solucionen o minimicen la acción de otras características intrínsecas, como por ejemplo, el solapamiento.

En este ámbito de investigación, los métodos existentes para equilibrar los datos son el submuestreo y el sobremuestreo, o métodos híbridos que combinan ambos enfoques.

- **Métodos de submuestreo (*Undersampling*)**, crean un subconjunto del conjunto de datos original eliminando instancias (normalmente instancias de la clase mayoritaria).
- **Métodos de sobremuestreo (*Oversampling*)**, crean un subconjunto del conjunto de datos original replicando o generando algunas instancias o creando nuevas instancias a partir de las existentes.

Dentro de estas familias de métodos, las técnicas de preprocesamiento más sencillas son los métodos no heurísticos, como la técnica de submuestreo aleatorio *Random UnderSampling* (RUS) y sobremuestreo aleatorio *Random OverSampling* (ROS) [BPM04]. Dichas técnicas funcionan eliminando instancias de la clase mayoritaria o replicando instancias de la clase minoritaria, de manera aleatoria, respectivamente. RUS y ROS cuentan con la ventaja de ser algoritmos sencillos de implementar y que presentan buena escalabilidad. Sin embargo, en el estado del arte para escenarios de datos pequeños (*Small Data*), se presentan algunas desventajas [Kra16a; Fer+18b]. En relación a RUS, existe el riesgo potencial de descartar datos útiles que afecten al proceso de aprendizaje del clasificador, ya que elimina instancias de forma aleatoria. Y respecto a ROS, el inconveniente más importante es que esta técnica puede aumentar la probabilidad de que se produzca un sobreajuste, ya que simplemente replica las instancias (genera copias exactas). En ambos casos, si la desproporción de clases es extrema, ambas desventajas se encuentran más propicias a ocurrir.

Para hacer frente a los problemas de las técnicas aleatorias antes mencionados, se han propuesto métodos más sofisticados. Entre ellos, el llamado *Synthetic Minority Oversampling TEchnique* (SMOTE) [Cha+02; Fer+18a] se ha convertido en uno de los enfoques más reconocidos en esta área para problemas Small Data. En resumen, su idea principal es crear instancias artificiales mediante la interpolación de cada instancia de clase minoritaria con sus k vecinos más cercanos que también pertenezcan a esa misma clase. Por lo tanto, con la técnica SMOTE, la clase positiva se sobremuestrea tomando cada muestra de la clase minoritaria e introduciendo ejemplos sintéticos a lo largo de los segmentos de línea que unen a todos los k vecinos más cercanos de la clase minoritaria. Dependiendo de la cantidad de sobremuestreo necesaria, se eligen aleatoriamente entre los k vecinos más cercanos. Este proceso se ilustra en la Figura 4.1, donde x_i es una instancia minoritaria, x_{i1} a x_{i4} son los 4 vecinos más cercanos de la misma clase que x_i , y r_1 a r_4 las instancias sintéticas creadas por la interpolación.

Cabe destacar que, al ser una técnica de sobremuestreo, SMOTE no produce pérdida de información. Además, amplía los límites de la clase minoritaria para generalizar mejor dichos clusters. Asimismo, esta técnica permite extensiones de su algoritmo con el fin de tratar el ruido o detectar áreas específicas con

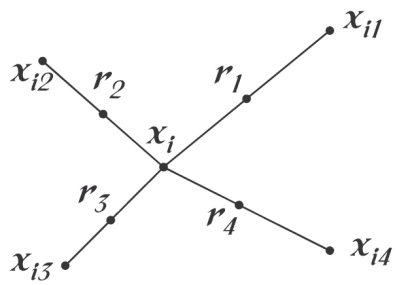


Figura 4.1: Interpolación de instancias minoritarias en SMOTE.

la necesidad de un tratamiento puntual de sobremuestreo. Sin embargo, hay que tener en cuenta que SMOTE no busca reemplazar o desestimar a las antes mencionadas soluciones aleatorias sino que todas ellas conforman al grupo de técnicas que comúnmente se aplican a conjuntos de datos no balanceados. En este sentido, no es posible establecer a priori cuál de las técnicas genera un buen desempeño a través de todos los problemas con desproporción de clases, y es tarea de todo científico de datos analizar cuál es la solución más adecuada respecto al problema a resolver.

4.2. Reducción de datos

En los conjuntos de datos es bastante común encontrar instancias redundantes como también alta dimensionalidad en los datos. Ambas complejidades pueden impactar en el aprendizaje del modelo debido a que, para el caso de la redundancia, no aporta nuevos conocimientos y, en el caso de la dimensionalidad excesiva, hay una gran parte de los atributos que no contribuyen con poder discriminante. En consecuencia, el clasificador tiene que aprender de datos y características que están presentes en el conjunto de datos, que le suponen tiempo para el correcto modelado, y que en definitiva deterioran el modelo predictivo o el aporte al proceso de extracción del conocimiento es insignificante.

Por lo tanto, para tratar la redundancia se deben aplicar técnicas de reducción de instancias que generan como resultado un conjunto de menor tamaño, con la característica que deben mantener el comportamiento o calidad predictiva del conjunto de datos original. Por ejemplo, en una tarea de clasificación, el entrenamiento de un modelo a partir de los datos reducidos tendría un rendimiento similar en comparación con el entrenamiento con el conjunto de datos completo. De manera similar, la alta dimensionalidad de los datos se puede tratar con técnicas de reducción de características que se encargan de determinar el aporte de cada atributo en la tarea predictiva, a partir de lo cual se pueden

descartar aquellos de muy baja o nula influencia. Del mismo modo que con las técnicas de reducción de instancias, las de reducción de características tienen por objetivo mantener el desempeño predictivo del conjunto de datos inicial. En la Figura 4.2 se muestra de manera general el proceso de la reducción de datos recién comentado.

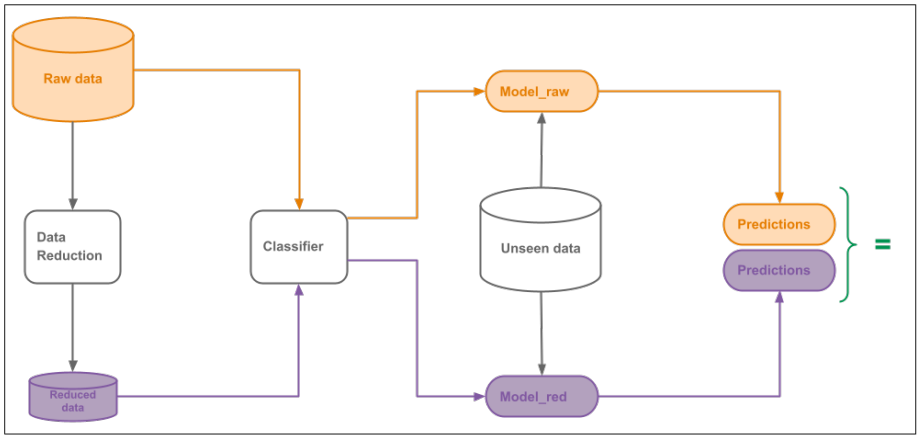


Figura 4.2: Esquema de la idea general de la reducción de datos.

Independientemente del método de reducción (o condensación) de datos que se aplique, los beneficios reales de su aplicación en el ciclo de vida de la ciencia de datos son mitigar los requisitos de almacenamiento y memoria, la complejidad del cálculo algorítmico y los tiempos de ejecución, además de que pueden obtener datos de mejor calidad al eliminar la redundancia conceptual [MTH20]. Por lo anterior, nótese que estas herramientas para la condensación de los datos cobran mayor protagonismo en escenarios Big Data.

4.2.1. Reducción de instancias (reducción horizontal)

Tradicionalmente, los métodos de reducción de datos se refieren a las tareas de preprocesamiento que suelen aplicarse en el proceso de extracción de conocimiento para reducir el tamaño de un conjunto de datos con respecto al número de ejemplos (también conocido como *reducción horizontal o por filas*) sin perder la información que representa.

En la taxonomía se pueden encontrar dos grupos de técnicas de reducción de datos ampliamente utilizadas: la selección de instancias (*Instance Selection* (IS)) [LM01] y la generación de instancias (IG) [Tri+12]. Nótese que la generación de instancias en este caso está vinculada al objetivo de la reducción (por ejemplo,

mediante la creación de prototipos), y no a la asociada con el fin de balancear los datos (como se da con SMOTE, por ejemplo).

Por un lado, el objetivo de la tarea IS es seleccionar un subconjunto de un conjunto de datos original; por otro lado, los métodos IG proporcionan un conjunto reducido de datos que puede ser construido a partir de un conjunto completo de instancias sintéticas o en combinación con una submuestra del conjunto de datos original.

En la literatura especializada, se puede encontrar que la base de la mayoría de los métodos de reducción de datos suele implicar la aplicación de un algoritmo clasificador basado en instancias. Un método de IS muy extendido es la regla del condensado rápido mediante vecino más cercano (en inglés *Fast Condensed Nearest Neighbor rule* (FCNN rule)) [Ang07], que utiliza la regla de los vecinos más cercanos para encontrar una submuestra de instancias del conjunto de datos inicial. La idea detrás de FCNN rule es que comienza con un subconjunto inicial que consiste en los centroides de cada clase y, en las iteraciones posteriores, el subconjunto se aumenta hasta que se alcanza la condición de parada. La forma de añadir instancias al subconjunto es incorporando el enemigo más cercano que está dentro de la región de Voronoi de cada instancia que pertenece al subconjunto. La condición parada se alcanza cuando no hay más enemigos. A pesar de tener unos resultados de reducción relativamente buenos, en cuanto a la complejidad, FCNN rule tiene, en el peor de los casos, una complejidad temporal cuadrática en términos del número de instancias.

Más métodos basados en instancias constituyen el estado del arte para la reducción de datos, como el algoritmo memético de estado estable (en inglés *Steady-State Memetic Algorithm* (SSMA)) [GCH08], la escalada de colina por mutación aleatoria (en inglés *Random Mutation Hill Climbing* (RMHC)) [Ska94], y la selección democrática de instancias (en inglés *Democratic Instance Selection* (DIS)) [GdG10].

4.2.2. Reducción de características (reducción vertical)

Además de la reducción horizontal, los datos también pueden ser reducidos en cuanto a sus variables (también conocido como *reducción vertical, de dimensionalidad o de columna*) no sólo porque los datos sean altamente dimensionales, sino que además incluso si no lo son, existe la necesidad de trabajar con un subconjunto de características que sean las más representativas, ya que esto potencia la robustez del modelo final aprendido. Las técnicas de reducción basadas en variables más populares son la selección de características (*Feature Selection* (FS)) y la extracción de características [LM98; LM07]. La primera, como su nombre lo indica, selecciona las características más importantes que pueden afectar al

comportamiento de un algoritmo de ML. Por su parte, la segunda construye un conjunto de nuevas características a partir de la combinación de las originales.

Entre los métodos del marco teórico de FS [GE03] para la reducción de la dimensionalidad, destacan los pertenecientes al grupo de las denominadas como soluciones embebidas. La idea se basa en obtener la contribución de cada variable después de entrenar un modelo clasificador. Por mencionar las características más relevantes de estos métodos, son los más precisos y mucho menos propensos al sobreajuste. Entre las diferentes soluciones integradas, una de las más utilizadas es RF debido a sus buenos resultados en aplicaciones de propósito general [SAV08; RD18].

De lo anterior se puede observar que se ha puesto el foco en la reducción tanto de instancias como de características desde un enfoque de selección en ambos casos.

4.3. Zonas ambiguas de un problema

Cuando se trabaja sobre un problema de clasificación de datos puede ocurrir con frecuencia que existan ciertas áreas del espacio del problema solapadas entre las distintas clases. Estas zonas son consideradas como ambiguas dado que los valores de las características no tienen un buen poder discriminante y el modelo resultante normalmente no se desempeñará bien al predecir datos nuevos que pertenezcan a dichas zonas. Esta situación puede darse más a menudo en los conjuntos Big Data dado su mayor número de instancias y su alta dimensionalidad de los datos.

Cabe mencionar que el solapamiento de clases puede comprender múltiples fuentes de complejidad y, por tanto, es más complicado de evaluar [DDC18; Pas+20]. La caracterización del problema de solapamiento de clases puede subdividirse en tres tareas secuenciales principales. En primer lugar, es importante descomponer el dominio de datos en regiones de interés. A continuación, hay que identificar las regiones problemáticas (regiones solapadas). Por último, se puede proceder a la cuantificación/medición del solapamiento de clases, y establecer su interpretación asociada.

Para manejar la presencia del solapamiento en un conjunto de datos es esencial, en primer lugar, proceder a identificar las áreas solapadas del espacio del problema. Para ello no existe un procedimiento estándar (único), y se suele considerar sencillamente el vecindario de las instancias para identificar tipologías de instancias: seguras (todos sus vecinos de la misma clase), solapadas (mismo número de vecinos de ambas clases), o incluso ruidosas o atípicas (todos los vecinos de la clase contraria). A partir de ello, se puede llevar a cabo el tratamiento

del solapamiento presente en los datos siguiendo alguno de los enfoques que se encuentra normalmente en la literatura [DKC13]:

- Descartar: Consiste en ignorar la región solapada y entrenar un modelo a partir del resto de los datos que pertenecen a la región no solapada.
- Fusionar: Considera la región solapada como una nueva clase y utiliza un modelo de clasificación de 2 niveles.
- Separar: Los datos de las regiones que se solapan y las que no se solapan se tratan por separado para construir los modelos de aprendizaje.

4.4. Comentarios del capítulo

En este capítulo se comentaron los principales enfoques para tratar algunos de los problemas que pueden estar presentes en un conjunto de datos y de los cuales repercuten negativamente en el proceso de extracción del conocimiento.

En primer lugar, se ha abarcado la desproporción de clases en los datos y se comentaron los distintos grupos de enfoques para tratar este problema. En particular, se detalló el enfoque a nivel de datos por tener estrecha relación con uno de los aportes de esta tesis.

A continuación, se comentaron las técnicas más populares para la reducción de los datos con foco en reducción de instancias (horizontal) como también en la reducción de características (vertical).

Por último, y respecto al solapamiento de clases en un conjunto de datos, se listan los distintos enfoques del estado del arte más comúnmente empleados para su tratamiento.

5 Big Data

En este capítulo, se presenta el campo de Big Data Analytics así como también los conceptos y las herramientas relacionadas a dicha temática. Para ello, en primer lugar se introduce al término Big Data y se describen sus características y conceptos asociados en la Sección 5.1. Luego, se presenta al modelo de programación distribuida llamado MapReduce en la Sección 5.2. A continuación, en la Sección 5.3, se lleva a cabo una visión general del denominado «ecosistema Hadoop» y las herramientas más importantes que lo componen. En particular se introducen dos frameworks muy utilizados para el procesamiento de aplicaciones Big Data, conocidos como Hadoop MapReduce y Apache Spark. Dada la relevancia de este último, en la Sección 5.4 se detalla a Spark comentando sus características, su arquitectura, sus extensiones, entre otros aspectos de interés. Y, finalmente, se comenta el contenido global del capítulo en la Sección 5.5.

5.1. Introducción a Big Data

En los últimos años se ha acuñado el término de Big Data para referirse a aquellos retos y ventajas que se derivan de la recopilación y procesamiento de grandes cantidades de datos [MVG21]. Este tema ha aparecido desde que se debe lidiar con colecciones de datos a escala de petabytes en adelante [Wu+14]. Los desarrollos tecnológicos a los que se enfrenta la sociedad, como el Internet de las Cosas (*Internet of Things* (IoT)), la computación en la nube (*Cloud Computing*), Internet, dispositivos móviles, entre otros, son las principales razones de este crecimiento, que conduce además a la Industria 4.0 [Lu17; Bou+21]. De hecho, esta situación se ha visto exacerbada debido a la creciente digitalización de la vida cotidiana dada por el confinamiento ocasionado a partir de la crisis sanitaria mundial causada por el agente SARS-COV-2[16], donde el uso de Internet se ha visto incrementado notablemente. En la Figura 5.1[17] se muestra el crecimiento de la población mundial de Internet (en miles de millones) para los años 2016, 2018, 2020 y 2021 [Dom21].

16 *Coronavirus Disease 2019* (COVID-19)

17 Imagen reproducida para mayor calidad a partir de la infografía de lo que ocurre en un minuto en Internet, *"Data never spleeps 9.0"*. https://web-assets.domo.com/blog/wp-content/uploads/2021/09/data-never-sleeps-9.0-586px-1.png

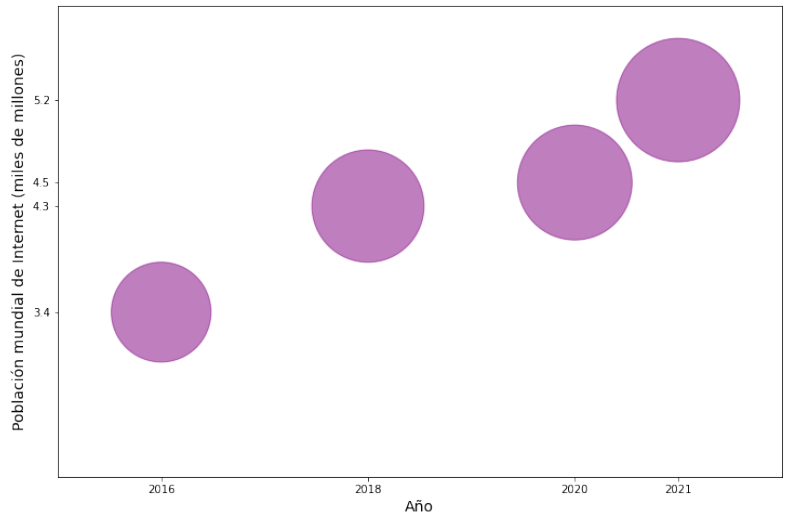


Figura 5.1: Crecimiento de la población mundial de Internet (en miles de millones) para los años 2016, 2018, 2020 y 2021.

Sin embargo, el primer problema para la correcta definición de "Big Data" es el propio nombre [Kra13], ya que se podría pensar que sólo está relacionado con el ***volumen*** de datos. La estructura heterogénea, la dimensionalidad diversa, y la ***variedad*** de la representación de los datos, también tiene importancia en esta cuestión. Por supuesto, también depende del tiempo de cálculo, es decir, de la eficacia y la ***velocidad*** de recepción y procesamiento de los datos, buscando dar respuestas en un tiempo transcurrido tolerable por el usuario.

Todos estas características resaltadas anteriormente se conocen como las 3V's del Big Data (se las puede observar integradas en la Figura 5.2[18]), que llevan a la definición dada por Steve Todd en la Universidad de Berkeley:

> *Big Data is when the normal application of current technology does not enable users to obtain timely, cost-effective, and quality answers to data-driven questions.—*[19]

18 Muhammad Abu Hijleh, CC BY-SA 4.0, via Wikimedia Commons

19 Que se traduce como: «Se habla de Big Data cuando la aplicación normal de la tecnología actual no permite a los usuarios obtener respuestas oportunas, rentables y de calidad a preguntas basadas en datos».

Hay que señalar que también se pueden encontrar definiciones adicionales que incluyen hasta 9V's, añadiendo términos como Veracidad, Valor, Viabilidad y Visualización, entre otros [Zik12]. Sin embargo, las 3V's anteriormente nombradas están presentes en la mayoría de las definiciones, el resto de los conceptos aún se están definiendo [Ras18; Osm19; Far+20].

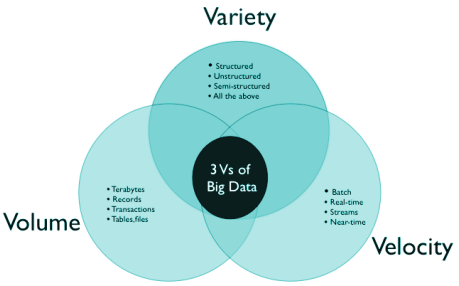


Figura 5.2: Las 3Vs del Big Data.

Por lo tanto, tratar con Big Data requiere tareas desafiantes, tales como la recolección de datos, el almacenamiento, el análisis y, con ellos, algunos debates inconclusos son la estandarización, la seguridad, la privacidad, los casos de estudio de referencia (en inglés *benchmarks*), entre otros [LJ12; Xin+14]. Así pues, la comunidad de Big Data tiene muchas cosas que definir todavía y, con el actual aumento de las ciudades inteligentes (en inglés *smart cities*), los hogares inteligentes (en inglés *smart homes*) y los nuevos avances tecnológicos, más definiciones tendrán que hacerse en un futuro próximo [Osm19].

Además, es importante mencionar que los términos Big Data y Ciencia de Datos (en inglés *Data Science* (DS)) tiene una alta sinergia entre ellos [WF13]. DS refiere al análisis llevado a cabo con fines de conocimiento y negocio [Xu+21], en la cual se ven involucradas varias disciplinas como la estadística, ML, la minería de datos (en inglés *Data Mining* (DM)) [HKP11], la inteligencia artificial (en inglés *Artificial Intelligence* (AI)) y la visualización, entre otras. Como ejemplos de las áreas de aplicación donde se da esta sinergia (todas ellas especialmente intensivas en datos) se encuentran la física de partículas [Kar+21], la bioinformática [NMB20], la medicina [Tad+21] o genómica [LP21], el análisis de redes sociales [Men+21], las repercusiones en las elecciones políticas [GCA20], las ciudades inteligentes [Ari+22] y el comercio electrónico a gran escala [WW21].

La relevancia de Big Data radica en la capacidad de extraer patrones e información oculta de los datos, y no solamente en los datos en sí. Para esto, es necesario seguir el ciclo de vida de la DS. En la Figura 5.3 se muestran sus etapas, que reúne gran parte de los contenidos descriptos previamente.

Así pues, luego de comprender el problema e identificar los objetivos a alcanzar, y después de recopilar toda la información posible, se procede a la exploración y análisis de los datos mediante el cálculo de estadísticas descriptivas o visualizaciones (Sección 2.2). A continuación, se preparan los datos lo cual involucra tareas para lidiar con ciertas complejidades no deseadas que pueden estar presentes en ellos (Capítulo 4). Por consiguiente, esta etapa de preprocesamiento de los datos

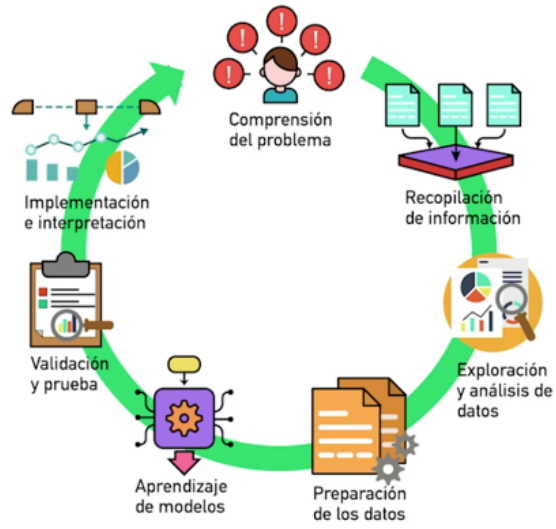


Figura 5.3: Etapas del ciclo de vida de la Ciencia de Datos.

es una de las que más tiempo demanda en el ciclo de vida de la DS, y es directo pensar que, en ambientes Big Data, esto se evidencia aún más.
Por lo general, al tratar con Big Data hay un subconjunto de datos que contiene información útil, y a partir de él hay una parte más pequeña de datos que retienen el verdadero conocimiento. Por lo tanto, es necesario deshacerse de estas "capas" de datos que no suponen más que ruido, mediante acciones que se asocian al término Smart Data, recientemente introducido en la bibliografía específica [Gil+19; Lue+20]. En concreto, el objetivo de Smart Data es convertir Big Data en crudo en datos de calidad, para la obtención de información útil y precisa. Adicionalmente a las evidentes ventajas sobre el modelado y extracción de conocimiento de calidad en el procesamiento de Big Data, la contribución de garantizar el paso de "datos en crudo" hacia el Smart Data también se encuentra en la complejidad computacional. En efecto, disminuir la cantidad de almacenamiento requerido y en reducir cómputos, permitirá obtener respuestas de un modo eficiente, y de este modo realizar pruebas sin un excesivo consumo de tiempo y/o energía, hacia lo que se conoce como *green computing*.

Una vez aplicada la fase de preprocesamiento para obtener el Smart Data, el siguiente paso de gran relevancia es el de aprendizaje de modelos mediante técnicas de ML, en el que se aplican algoritmos para extraer patrones de los datos siguiendo un enfoque supervisado, no supervisado, o semi-supervisado. Estos algoritmos pueden ser de la familia de la regresión, agrupamiento, clasificación,

entre otros (Capítulo 3). Finalmente, se valida y prueba el modelo obtenido, y se interpretan y evalúan los resultados.

Todo este procesamiento de Big Data puede llevarse a cabo en cualquier infraestructura distribuida con los requerimientos necesarios para permitir un tratamiento adecuado a la cantidad de datos que se va a manejar. Hoy en día, no es necesario desplegar un cluster completo para tratar Big Data, sino que se puede hacer pagando servicios de *Cloud Computing*. El Cloud Computing ofrece a aquellas organizaciones, que no pueden desplegar y mantener su propio cluster de ordenadores, la posibilidad de realizar análisis de Big Data pagando por los recursos necesarios para realizar estas tareas [AA21].

5.2. El modelo MapReduce

Para trabajar con conjuntos de grandes volúmenes de datos, las arquitecturas de cómputos de un sólo nodo no son capaces de abordar el almacenamiento y procesamiento requeridos de forma efectiva y eficiente, razón por la cual se vuelven indispensable los enfoques de cómputos distribuidos. Por lo tanto, hay que adaptar las técnicas tradicionales de ML para que sean totalmente escalables, o desarrollar nuevas aproximaciones dentro de este paradigma distribuido.

En consecuencia, surge el modelo de programación *MapReduce* (MR) [DG04; DG10] propuesto por Google, que ha sido ampliamente aceptado por la comunidad Big Data. Así, para poder procesar todos estos datos de la forma escalable necesaria, este framework fue diseñado basándose en un enfoque de "divide y vencerás". Básicamente, el modelo establece dos etapas heredadas de los paradigmas clásicos de programación funcional: *Map* y *Reduce*. La etapa Map recibe los datos de entrada, los convierte en pares clave/valor y le aplica operaciones con el objetivo de transformarlos. A continuación, en la fase Reduce, se ejecuta primero una tarea llamada *Shuffle* la cual se encarga de derivar todos los pares con la misma clave al mismo nodo del cluster para luego ser reducidos. Posteriormente, se efectúa una agregación sobre los valores de una misma clave con el fin de procesarlos utilizando la técnica necesaria según el problema a resolver, y así obtener el resultado final. Esta forma de procesamiento en dos etapas se ilustra en la Figura 5.4. Es importante señalar que, en la práctica, un gran número de estas operaciones Map y Reduce se podrían concatenar para alcanzar el resultado de una tarea completa.

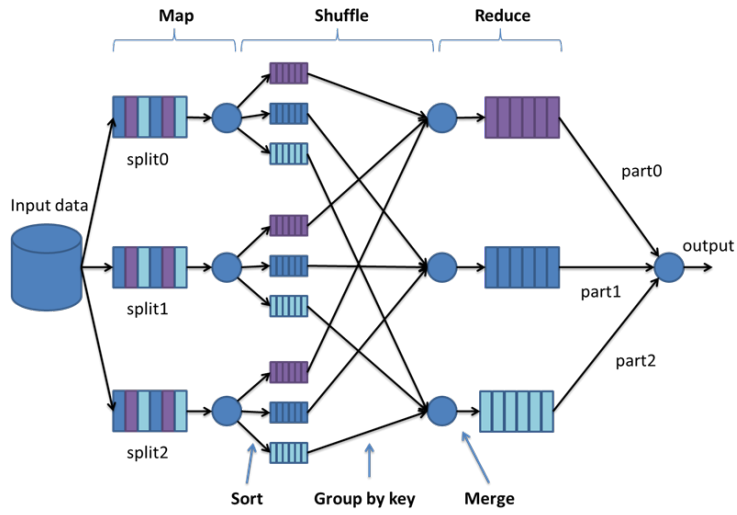


Figura 5.4: Flujo de trabajo de *MapReduce*.

5.3. El ecosistema Hadoop

Cuando se habla del ecosistema Hadoop se refiere a una gran cantidad de herramientas que cooperan entre sí para llevar a cabo la gestión de Big Data. Entre esas muchas herramientas, como las que se observan en la Figura 5.5[20], detallamos a continuación las más comúnmente utilizadas para el almacenamiento, la gestión de recursos y el procesamiento de Big Data.

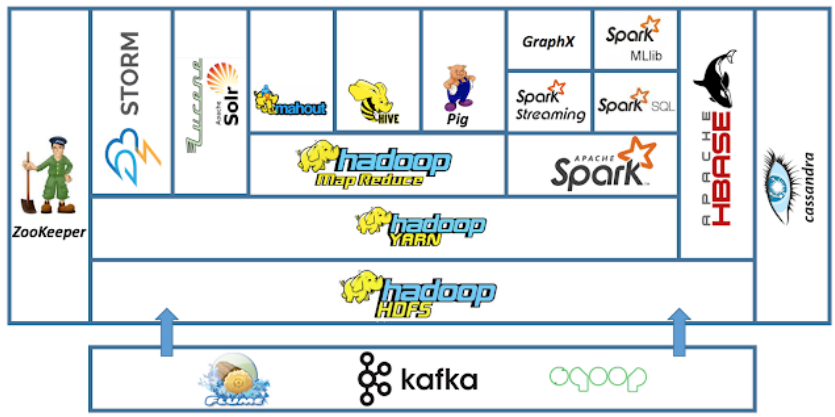


Figura 5.5: El ecosistema Hadoop.

20 https://spark.apache.org/docs/latest/img/cluster-overview.png

5.3.1. Almacenamiento: Hadoop HDFS

En cuanto al almacenamiento distribuido en escenarios Big Data, el sistema de archivos de gran popularidad es *Hadoop Distributed File System* (HDFS) [Shv+10]. Como tal, HDFS es de código abierto y proporciona eficiencia, procesamiento distribuido, almacenamiento tolerante a fallos, y robustez. Su arquitectura es de tipo *master/workers*, como se muestra en la Figura 5.6[21], lo que en la terminología de HDFS se traduce como *NameNode/DataNodes*. Como sus nombres lo sugieren, el *NameNode* se ocupa de la gestión del espacio de nombres (*namespace*) del sistema de archivos, y de los accesos a los archivos; mientras que los *DataNodes* se encargan precisamente de los datos almacenados en el nodo en el que se encuentran (normalmente hay un *DataNode* por nodo). Por consiguiente, el *NameNode* se responsabiliza de los metadatos del HDFS, y los *DataNodes* de los datos en sí siguiendo las directivas provistas por el *NameNode* (por ejemplo, para crear, replicar o borrar bloques de los archivos).

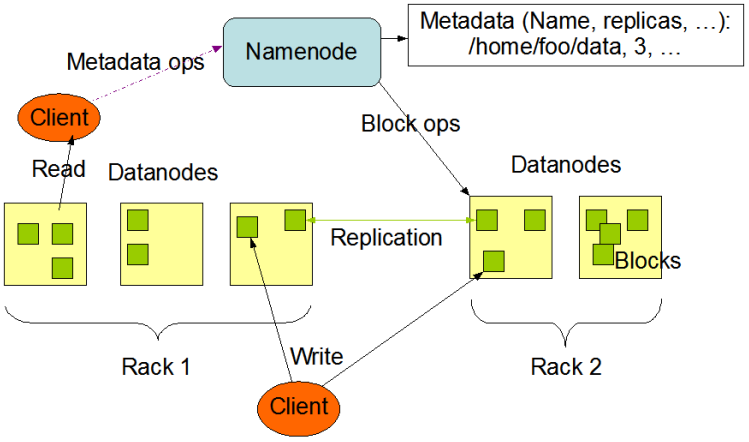


Figura 5.6: Arquitectura del HDFS.

La tolerancia a fallos que brinda HDFS se consigue mediante la replicación de los datos en distintos *DataNodes*. Además, este mecanismo permite aprovechar la localidad de los datos y así disminuir el tráfico de red y, por ende, los tiempos de ejecución.

21 https://hadoop.apache.org/docs/current1/hdfs_design.pdf

5.3.2. Gestión de recursos y monitorización de trabajos: Hadoop Yarn

En la Figura 5.7[22] se muestra la arquitectura para la gestión de recursos y la programación de trabajos de Hadoop Yarn (*Yet Another Resource Negotiator*). La idea subyacente es la separación de ambas tareas antes mencionadas mediante un manejador de recursos global (*Resource Manager, RM*) y uno por cada aplicación (*Application Master, AM*) que negocia por los recursos con el RM. A su vez, cada nodo tiene un gestor local (*Node Manager, NM*) que se responsabiliza por los recursos asignados al nodo en el que se encuentra, y responde al RM siguiendo sus directrices.

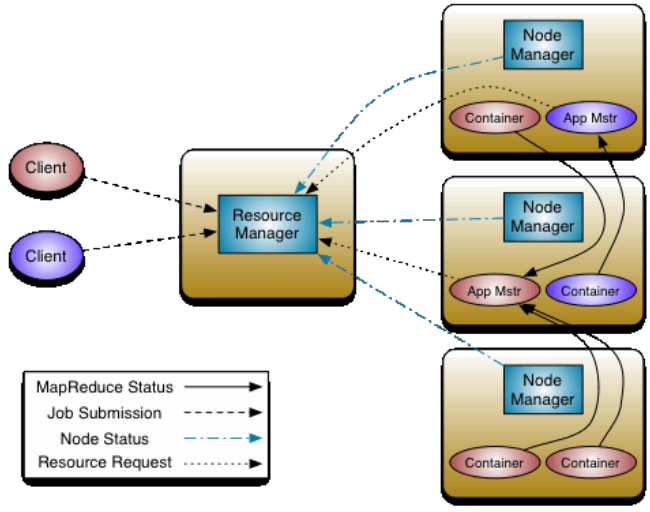


Figura 5.7: Arquitectura de Hadoop Yarn.

5.3.3. Motores de procesamiento: Hadoop MapReduce y Apache Spark

En lo referente a los motores o frameworks para el procesamiento Big Data, entre las opciones más populares se encuentran Hadoop MapReduce y Apache Spark. La idea general detrás de ellos es que se encargan de coordinar las tareas para llevar a cabo la ejecución de una aplicación distribuida y de propósito general.

Hadoop [Whi12] es la implementación *open source* de MapReduce desarrollada por Doug Cutting y Mike Cafarella, y mantenida por la comunidad *Apache*

22 https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site

Software Foundation (al igual que el resto de las herramientas que se describen como parte del ecosistema Hadoop en esta tesis). En sus inicios, este framework alcanzó una gran popularidad. Sin embargo, presenta algunas limitaciones que cobran mayor importancia de acuerdo al caso de uso en el que se esté queriendo trabajar [Lin13]. Así pues, entre sus principales debilidades se encuentran la falta de capacidad para el tratamiento de flujos de datos, la ineficiencia para el procesamiento iterativo de los datos, el intensivo uso de operaciones en disco y la dificultad de programación para llevar a cabo tareas complejas.

Por lo anteriormente expuesto, Apache Spark se posiciona como mejor opción dado que no presenta las debilidades encontradas en Hadoop y, más aún, posee significativas ventajas para el procesamiento de conjuntos Big Data. Por consiguiente, en la Sección 5.4 se lleva a cabo una descripción detallada de dicha herramienta. Asimismo, en el Apéndice A.3 se muestra una comparación entre Hadoop MapReduce y Apache Spark.

5.4. Profundizando en Apache Spark

Como se indicó en la sección anterior, Apache Spark [Zah+16b] surge como un motor unificado para el procesamiento de datos distribuidos con un modelo de programación similar al de MR, pero extiende a éste con una abstracción de intercambio de datos llamada *Resilient Distributed Datasets* (RDD). Así, cuando con otros frameworks se requiere integrar distintas herramientas separadas para poder llevar a cabo de manera completa el preprocesamiento de Big Data, en Spark esto no es necesario por sus librerías de procesamiento especializadas implementadas sobre el motor principal.

Otra de las características relevantes de esta herramienta es su capacidad para soportar aplicaciones iterativas (ej: muchos algoritmos de ML) mediante procesamiento en memoria, a la vez que conserva la escalabilidad y la tolerancia a fallos. Su uso intensivo de memoria convierte a Spark en uno de los frameworks Big Data más rápidos.

Las particularidades antes mencionadas convierten a Apache Spark en un estándar "de facto" para el desarrollo de soluciones en Big Data Analytics, siendo motivo principal de dedicar una sección en exclusiva para describir sus características más interesantes.

Los RDDs son una colección distribuida de elementos considerada como la estructura de datos básica de Spark. Estas estructuras pueden ser explícitamente persistidas en memoria principal o en disco para almacenar los resultados intermedios del procesamiento que se esté realizando. Los RDDs están divididos de manera que cada una de sus porciones puede ser operada por diferentes nodos en paralelo. En Spark, al número de estas porciones se las llama número

de particiones (lo que tradicionalmente en Hadoop se denominaba número de Maps). Asimismo, Spark provee los DataFrames y Datasets (son RDDs de filas), que son otras estructuras de datos con todos los beneficios de los RDDs pero con una optimización extra, lo cual las convierte en las más rápidas. Además, los DataFrames y Datasets proporcionan una interfaz de programación más flexible.

En la Figura 5.8 [23] [24] se muestra la arquitectura de una aplicación Spark, donde se tiene un coordinador central (*Driver*) que se comunica con los nodos trabajadores (*Executors*). La ejecución comienza en el Driver y es allí donde se crea la sesión de Spark (*SparkSession*), que actúa como punto de entrada de la aplicación y encapsula la interacción con el resto de los componentes de Spark. También se crean las estructuras de datos y se realizan las operaciones (transformaciones y acciones). Por su parte, los Executors se responsabilizan de la ejecución de las tareas (*Tasks*) y devuelven los resultados al Driver. Para tal fin, cuentan con la capacidad de leer/escribir datos en fuentes externas o sistema de almacenamiento, persistir los resultados ya sea en memoria, caché o disco, entre otras. A su vez, el *Cluster Manager* hace referencia a la herramienta externa que gestiona los recursos y monitoriza los trabajos, un ejemplo de esto puede ser Hadoop Yarn, anteriormente descripto en la Sección 5.3.2.

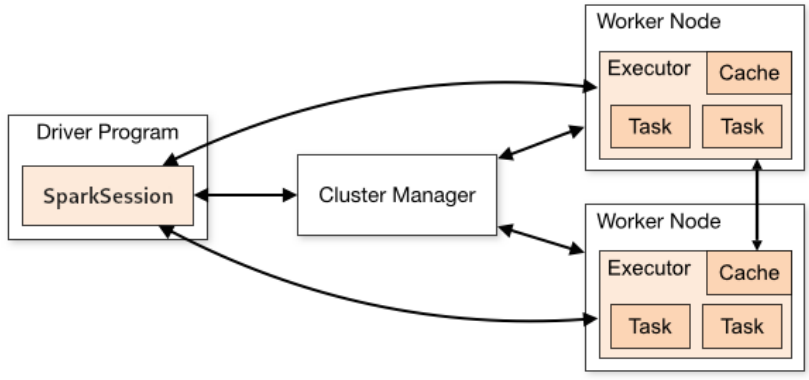


Figura 5.8: Arquitectura de Apache Spark.

Para poder implementar operaciones más complejas sobre datos distribuidos, Spark proporciona una gama más amplia de primitivas que, por ejemplo, las que brinda el framework Hadoop MapReduce. En el Apéndice A.4 se describen las primitivas Spark más comúnmente utilizadas.

23 https://spark.apache.org/docs/latest/img/cluster-overview.png

24 Se modificó la imagen original reemplazando "SparkContext" por "SparkSession" dado los cambios efectuados desde Spark 2.0: https://spark.apache.org/releases/spark-release-2-0-0.html#programming-apis

Las operaciones sobre las estructuras de datos en Spark se distinguen entre transformaciones y acciones. Las primeras crean una nueva estructura a partir de la transformación de los datos de otra estructura, mientras que las segundas disparan operaciones en las que su valor de retorno no es una estructura Spark. A su vez, es importante señalar que las transformaciones no se ejecutan inmediatamente, por eso se las llaman "operaciones perezosas" (*lazy operations*), en tanto que las acciones ponen en marcha las tareas para salir de ese estado de "pereza". Por consiguiente, mientras no haya acciones que ejecutar, Spark crea un Grafo Acíclico Dirigido (*Directed Acyclic Graph* (DAG)) compuesto por las estructuras de datos como vértices y las aristas una operación de transformación a aplicar sobre un vértice. Así, cuando se invoca una acción, Spark determina la forma más eficiente para llevar a cabo todas las tareas del DAG, y divide el grafo en etapas para luego enviadas al Cluster Manager con el fin de ser lanzadas a ejecución. Este mecanismo habilita la tolerancia a fallas dado que, recorriendo el DAG, es posible recuperar estructuras perdidas como así también permite una optimización global de los trabajos de Spark. Un ejemplo de la visualización de un DAG se presenta en la Figura 5.9[25].

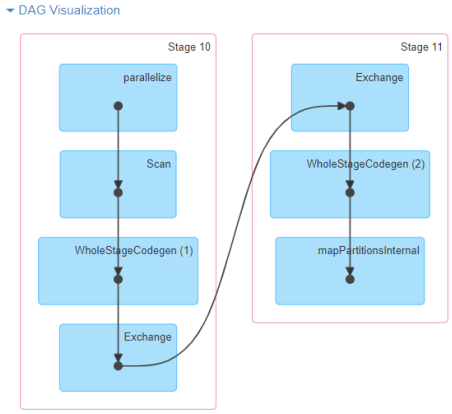


Figura 5.9: Visualización de un DAG en Spark.

Al inicio de esta sección se mencionó que existen librerías especializadas de procesamiento que acompañan al motor principal de Spark para poder llevar a cabo distintos tipos de tareas. Las mismas se describen brevemente a continuación:

- Spark SQL: es un módulo para trabajar con datos estructurados. Como su nombre lo indica, habilita realizar consultas SQL sobre los datos, así

25 https://spark.apache.org/docs/3.0.0/img/JobPageDetail2.png

como también combinarlas con las funciones primitivas o con la API de Datasets.

- Spark Streaming: esta librería provee un mecanismo tolerante a fallos para el tratamiento de flujos de datos (en inglés *streaming data*). Internamente, Spark Streaming recibe flujos de datos de entrada en tiempo real y divide los datos en lotes, que luego son procesados por el motor principal de Spark, y opcionalmente utilizando las otras librerías, para generar el flujo final de resultados también en lotes.

- Spark MLlib: este módulo se centra en las tareas de manera escalable de ML, de ahí su nombre. Su rendimiento es 100 veces más rápido con respecto a MR. Incluye no sólo algoritmos de ML sino también utilidades para facilitar el proceso de tratar con los datos. Entre los algoritmos que ofrece se pueden encontrar de la familia de los de clasificación, regresión, clustering; asimismo herramientas para la preparación de los datos y de estadísticas.

- Spark GraphX: proporciona una colección de algoritmos y constructores de grafos para simplificar las tareas de análisis de los mismos.

Cabe destacar que, debido a varias mejoras que benefician tanto al núcleo (motor principal) como a las librerías de alto nivel, las últimas versiones de Spark (la 3.x, desde junio de 2020) son aproximadamente dos veces más rápidas que las versiones 2.4.x, lo que convierte a Spark en una de las mejores opciones para procesar Big Data en tiempos aceptables.

Respecto al diseño de una solución Big Data que siga la estrategia, anteriormente descripta, de "divide y vencerás" se pueden encontrar dos enfoques en cuanto a la distribución de datos y modelos [Ram+18b; Fer+19]. Por un lado, el enfoque *local* da lugar a un modelo aproximado debido a que cada partición de datos se procesa por separado, es decir, sin ningún conocimiento de los datos pertenecientes a las demás particiones. Por lo tanto, en este enfoque cada partición de datos se encuentra aislada. Por otro lado, el diseño *global* genera un modelo exacto porque los datos y los modelos se comparten en todos los nodos del cluster. En otras palabras, los datos se dividen en una serie de particiones pero, al mismo tiempo, cada nodo puede conocer los datos de otras particiones utilizando estructuras de datos distribuidas. Por lo anterior, en la literatura se los pueden encontrar como enfoque local o aproximado, y enfoque global o exacto, indistintamente.

Hay que tener en cuenta que ambos esquemas para abordar los problemas de Big Data también tienen implicaciones directas en las características de los

datos que se introducen en los modelos. En concreto, en las aplicaciones de ML un mayor particionamiento de los datos en busca de una mayor escalabilidad para los modelos locales, puede aumentar la presencia de falta de datos para algunas clases o conceptos, dando lugar tanto a desequilibrios como a pequeñas disyuntivas [Lóp+13], que degradan la calidad de los modelos.

5.5. Comentarios del capítulo

En este capítulo se ha introducido el concepto de Big Data y comentado los aspectos más relevantes referente al mismo. Dado que este tópico ha surgido en los últimos años, queda mucha taxonomía por delante para definir y consensuar. También, se han mencionado múltiples aplicaciones y situaciones que llevan a generar escenarios Big Data, y se ha mostrado una de sus definiciones así como también comentado sus características principales: Volumen, Variedad, Velocidad. Además, debido a la complejidad para tratar los problemas Big Data, se han mencionado algunas de las disciplinas que también intervienen en dicha tarea.

A su vez, se ha enfatizado en que hay que considerar que la gran cantidad de datos en los problemas Big Data no implica que todos ellos sean útiles. De hecho, la mayoría de las veces, un subconjunto de los datos será la verdadera fuente de conocimiento valioso. Es decir, existe una relación entre la calidad de los datos y los resultados que se obtengan de trabajar con ellos, situación que manifiesta la necesidad de obtener *Smart Data*. Para esto, es necesario descubrir las complejidades (o también llamadas características intrínsecas) presentes en los datos, comprender cómo afecta aprender de datos con dichas características indeseadas y, en consecuencia, elegir las técnicas de preprocesamiento adecuadas para convertir los datos hacia Smart Data, y así luego continuar con el proceso de obtención de conocimiento.

Por otra parte, se ha descripto al modelo de programación distribuido más popular en entornos Big Data (*MapReduce*), y a las más relevantes herramientas de código abierto con foco en el almacenamiento distribuido (HDFS), en la administración de recursos (*Hadoop Yarn*) y en el procesamiento de grandes conjuntos de datos (*Hadoop MapReduce* y *Apache Spark*). Respecto a esto último, hemos profundizado en describir el framework *Apache Spark* por dos motivos. Por un lado, por su capacidad de contrarrestar las limitaciones presentes en una de las herramientas que se había tornado popular en los comienzos de Big Data (*Hadoop MapReduce*); y, por otro lado, por ofrecer múltiples ventajas y, además, proporcionar librerías especializadas para el procesamiento de Big Data, integradas en el motor principal.

Parte II

Aportes

6 Big Data no balanceado

Este capítulo se centra en un área muy importante dentro del análisis de Big Data, a saber, la clasificación con conjuntos de datos no balanceados. La principal característica de este problema es que una de las clases está sub-representada y, por lo tanto, generalmente es más complejo generar un modelo que la identifique correctamente. Los clasificadores canónicos están diseñados para optimizar la precisión global sin tener en cuenta la distribución relativa de las clases. En consecuencia, estos clasificadores pueden tender a ignorar las clases con pequeñas cantidades de instancias mientras se concentran en clasificar las de grandes cantidades de instancias con precisión. Este tema resulta de gran importancia tanto en academia como en industria debido a la gran cantidad de aplicaciones reales en las que está presente. Como ejemplo principal, se debe pensar en pacientes con una enfermedad rara [Gal+18], donde la clase minoritaria representa el concepto clave sobre el que se desea extraer conocimiento. Por esta razón, es común aplicar técnicas con el fin de equilibrar la distribución de ejemplos en las clases, de forma que el algoritmo de aprendizaje sea independiente de los datos que se den como entrada. Las técnicas de preprocesamiento para balancear datos, mencionadas en la Sección 4.1, son las más utilizadas en problemas de clasificación de Small Data y, por tanto, resulta necesario realizar una correcta adaptación para escenarios de Big Data para extrapolar su uso en este contexto. Se ha de considerar que, incluso cuando un algoritmo de preprocesamiento se centra sólo en instancias minoritarias, para problemas Big Data podría implicar un número muy grande de datos. De acuerdo con lo anterior, es indispensable rediseñar las técnicas estándar para que sean capaces de abordar problemas Big Data en tiempos de ejecución razonables, mediante computación distribuida siguiendo, por ejemplo, el enfoque "divide y vencerás" que ofrece MapReduce [Fer+17].

La organización de este capítulo es como sigue. En primer lugar, se comenta el estado actual de la clasificación no balanceada en Big Data en la Sección 6.1, puntualizando en la clasificación binaria con datos tabulares. Luego, en la Sección 6.2 se presenta nuestra propuesta escalable basada en el conocido algoritmo del estado del arte SMOTE para Big Data desde un enfoque de diseño global, y en la Sección 6.3 se analiza su comportamiento. A continuación, se lleva a cabo un estudio experimental de otras técnicas del estado del arte para balancear conjuntos Big Data en la Sección 6.4. Finalmente, se comenta el capítulo en la Sección 6.5.

6.1. Clasificación no balanceada de Big Data

El problema de la clasificación no balanceada surgió al mismo tiempo en que los investigadores se dieron cuenta de que los algoritmos de clasificación tradicionales no podían modelar bien las clases subrepresentadas [Lóp+13; Kra16b]. Cuando se trata del contexto de Big Data, este problema se acentúa dado que posee mayor cantidad de instancias como también por la, usualmente, gran dimensionalidad de los datos, haciendo el proceso de modelado aún más complejo si cabe. En este ámbito existen varias aplicaciones a problemas reales y, abordarlas, supone un reto interesante. Siendo más específicos, algunos ejemplos de aplicaciones son: la predicción de accidentes de tráfico [PKH16], el diagnóstico médico [HBK19; Has+20], los sistemas de detección de intrusos [BL21], descubrimiento de nueva física [Mur19], entre otros.

En este contexto, es importante recordar lo mencionado al final de la Sección 5.4 referido a que las tecnologías actuales para trabajar con grandes volúmenes de datos presentan dos enfoques diferentes respecto a cómo se agregan los modelos parciales. Por un lado, existen estrategias `locales` que producen un modelo aproximado aplicando una agregación directa sobre los modelos parciales. Por otro lado, hay métodos `globales`, que distribuyen tanto los datos como los modelos para construir iterativamente un resultado final.

Así pues, en [Río+14], los autores presentaron un estudio exhaustivo con el objetivo de evaluar el rendimiento de las soluciones tradicionales para el desequilibrio de clases en el contexto Big Data. Para ello, se adaptaron varias técnicas de preprocesamiento y se incrustaron en un flujo de trabajo MapReduce. En concreto, se emplearon el sobremuestreo aleatorio (ROS-BigData), el submuestreo aleatorio (RUS-BigData) y la versión de SMOTE MapReduce para Hadoop (SMOTE-H). Siguiendo la filosofía de Hadoop MapReduce, cada proceso Map se encargó de ajustar la distribución de clases para su partición de datos, ya sea mediante la replicación aleatoria de instancias de clase minoritaria (ROS-BigData), la eliminación aleatoria de instancias de clase mayoritaria (RUS-BigData) o la generación de datos sintéticos realizada mediante la técnica SMOTE (SMOTE-H). Posteriormente, un único proceso de Reducción se encargó de recoger los resultados generados por cada Map y asignarlos aleatoriamente para formar el conjunto de datos equilibrado. Todos estos métodos de preprocesamiento trabajaban localmente dentro de cada Map. De los resultados se observa que el sobremuestreo aleatorio es más robusto que las otras técnicas cuando el número de particiones de datos aumenta. Posteriormente, los autores de [Fer+17] adaptaron los algoritmos ROS y RUS para Apache Spark siguiendo un diseño global, implementado en el paquete software `Imb-sampling-ROS_and_RUS`[26].

26 Disponible en https://spark-packages.org/package/saradelrio/Imb-sampling-ROS_and_RUS

Asimismo, en [Gut+17b] se propuso un enfoque basado en el uso de la GPU para el cálculo paralelo de SMOTE. La técnica de preprocesamiento se adapta al hardware básico mediante un uso inteligente de la memoria principal, es decir, incluyendo sólo las instancias de la clase minoritaria, y el cálculo del vecindario mediante una implementación rápida en la GPU del algoritmo kNN [Gut+16]. Los autores encontraron que el aumento del porcentaje de sobremuestreo alcanza mejores resultados en la clase positiva. Un trabajo que muestra el éxito de la aplicación del sobremuestreo aleatorio con un alto nivel de replicación de instancias minoritarias se encuentra en [Tri+15a], donde se utiliza un problema bioinformático conocido como ECBDL14.

Cabe señalar que la traslación de los métodos de preprocesamiento tradicionales hacia un enfoque distribuido no es directa. En primer lugar, el framework de programación puede implicar un rediseño completo del procedimiento para adaptarlo a una estrategia de procesamiento distribuido. En segundo lugar, la división de datos necesaria para abordar los problemas de Big Data puede conducir a la falta de datos cuando se procesan los modelos locales. Estos hechos implican la necesidad de desarrollar una nueva línea de investigación sobre el tratamiento para el equilibrio de clases en escenarios Big Data, dado que todavía hay poca investigación sobre el tema [Fer+17].

Las soluciones MapReduce, que se basan en el enfoque "divide y vencerás", implican una partición de los datos que puede llevar a dos graves consecuencias [Lóp+13]. Una de ellas es la extrema falta de datos positivos en cada partición de datos, que implica una representación parcial de la información del conjunto de datos, en particular con respecto a la vecindad real de las instancias. La otra consecuencia del particionamiento, y muy relacionada con la anterior, es la presencia de datos muy locales y de baja densidad denominados *small-disjuncts* que representan los conceptos de interés. La creación de instancias minoritarias a partir de los *small-disjuncts* puede dar lugar a ruido o a una sobregeneralización, y entrarían en las zonas «seguras» de la clase mayoritaria.

6.2. SMOTE escalable para la clasificación no balanceada en Big Data

En la Sección 6.1 se menciona la existencia de una implementación de SMOTE para Big Data [Fer+17], sin embargo la misma se desarrolló utilizando Hadoop MapReduce (framework muy popular a la fecha de dicho trabajo), lo que la convierte en una solución distribuida pero no tan eficiente y escalable como si fuera diseñada para trabajar con Apache Spark. Además de lo anterior, la traslación de código de un Hadoop a Spark no es directa (véase en el Apéndice A.3 una

comparación de las características de ambos frameworks). En consecuencia, y al momento de la escritura de este capítulo, sólo se encuentran las soluciones aleatorias desarrolladas con Spark, disponibles en el paquete de software llamado `Imb-sampling-ROS_and_RUS`, mencionado también en la Sección 6.1. Por lo anterior, en esta sección se presenta SMOTE-BD, una metodología totalmente escalable y rápida de la técnica de sobremuestreo SMOTE para el preprocesamiento de conjuntos Big Data no balanceados. Su diseño sigue el enfoque global, y se utiliza el framework Apache Spark para su desarrollo. El objetivo de esta contribución es poner a disposición de la comunidad dicha técnica de preprocesamiento, tan ampliamente usada en Small Data, capaz de procesar grandes volúmenes de datos de manera paralela y en tiempos cortos de respuesta. Para ello, su desarrollo se basó en un estudio sobre las particularidades necesarias para el diseño de SMOTE totalmente escalable, y que el mismo cumpla con los criterios de un enfoque global para que su comportamiento se ajuste lo más fielmente posible a la técnica original secuencial. Así pues, junto a las técnicas para el sobre/sub muestreo aleatorio, contar con las estrategias de base para el tratamiento de un conjunto de datos con desproporción de clases.

Descripción y flujo de trabajo En la Figura 6.1 se presenta el esquema general de nuestra propuesta. En primer lugar, el algoritmo realiza un filtrado de los datos sobre el conjunto de entrenamiento `training set` (almacenado en el HDFS) para obtener los subconjuntos de instancias positivas y negativas (`Positive Data (PD)` y `Negative Data (ND)`). Los datos se particionan según el valor del parámetro de entrada relacionado de nuestro algoritmo, y se almacenan en caché para ser reutilizados en los siguientes pasos. A continuación, los datos minoritarios se normalizan (`PD normalized`) teniendo en cuenta las estadísticas del conjunto de entrenamiento completo.

Seguidamente, se obtienen los vecinos más cercanos para cada instancia positiva utilizando una implementación exacta del algoritmo *k*NN en Spark, llamada `kNN-IS` [Mai+17]. Esta solución Big Data, en primer lugar, divide el conjunto de datos de entrenamiento en un número de particiones definido por el usuario, luego calcula los vecinos para cada instancia perteneciente a un bloque de datos y, finalmente, en una fase de reducción, realiza una lista de los k vecinos más cercanos (`Neighbors`).

Posteriormente, se inicia la generación de instancias de clases minoritarias artificiales. Todos los vecinos más cercanos obtenidos en el paso anterior se transmiten a la memoria principal de todos los nodos del cluster de forma eficiente. Esto se lleva a cabo mediante la operación `broadcast` que provee el framework Spark, la cual permite mantener una variable de sólo lectura en caché en cada nodo, en lugar de enviar una copia a cada tarea. Así, para cada instancia positi-

va de una partición de datos y, utilizando la variable `broadcast`, el algoritmo genera el correspondiente número de ejemplos sintéticos (`Artificial Data`) interpolando entre cada instancia minoritaria y sus k vecinos más cercanos (procedimiento que se introdujo en la Sección 4.1).

Finalmente, nuestra propuesta realiza un proceso de desnormalización sobre el conjunto de datos artificial (`Artificial Data Denormalized`). A continuación, se unen las instancias originales positivas y negativas con las artificiales para conformar el conjunto de datos equilibrado (`Balanced Training set`), y se almacenan en el HDFS.

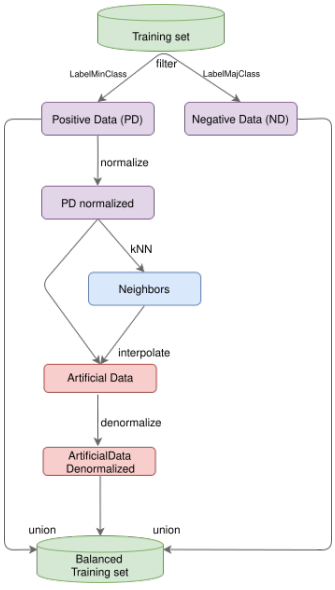


Figura 6.1: Diagrama de flujo de trabajo de SMOTE-BD.

Cabe destacar que nuestra propuesta incorpora un diseño minucioso basado en el uso de estructuras de datos y funciones escalables. En concreto, SMOTE-BD está implementado bajo el lenguaje de programación Scala para el framework Spark (versión 2.2.0). El código fuente (código libre, abierto y gratuito) de esta técnica de preprocesamiento se puede encontrar en el repositorio https://github.com/majobasgall/smote-bd o como parte del repositorio de paquetes de terceros para Apache Spark llamado `Spark Packages` en https://spark-packages.org/package/majobasgall/smote-bd.

Los Algoritmos 1 y 2 muestran la secuencia de acciones descripta anteriormente en pseudocódigo. El primero cubre el programa principal, mientras que el segundo la función para crear cada instancia artificial. Esta función invoca a otra función llamada `interpolation` que se encarga de hacer la interpolación entre dos puntos. No hay pseudocódigo de esto debido a su simplicidad. Asimismo, a modo de referencia, la Tabla 6.1 describe brevemente las primitivas de Spark utilizadas en nuestro código[27].

27 Todas las primitivas y funcionalidades de Spark utilizadas en esta tesis se listan en el Apéndice A.4

Algoritmo 1: Algoritmo SMOTE-BD.

```
1  Function SMOTE-BD(Tr, Ts, ratio, k, nP, nR, nIt, minClassLabel)
2      origData ← textFile(Tr)
3      minData ← origData.filter(labels == minClassLabel)
4      minData ← minData.map(normalize).repartition(nP)
5      neighbors ←
         kNNIS.setup(Tr, Tr, k, nR, nI).calculatekNeighbours()
6      crFactor ← (nMaj − nMin)/nMin
7      neighbors ← broadcast(neighbors)
8      balancedData = null
9      synData = null
10     for i < nIt do
11         synTmp ← minData.mapPartitionsWithIndex(
             createSynthData(index, partData, neighbors, crFactor, k))
12         if synData == null then
13             synData ← synTmp
14         else
15             synData ← synData.union(synTmp)
16         end
17     end
18     synData ← synData.map(denormalize)
19     balancedData ← synData.union(origData)
20     return balancedData
```

Algoritmo 2: Función de SMOTE-BD para crear instancias sintéticas entre los ejemplos de la clase minoritaria y sus vecinos.

```
1  Function createSynthData(index, partData, neighbors, crFactor, k)
2      artificialData = null
3      for firstInstance ← partitionData; nc = 0 to crFactor do
4          selNeighbor ← newRandom().nextInt(k)
5          secondInstance ← neighbors(selNeighbor)
6          newIntance ← interpolation(firstInstance, secondInstance)
7          artificialData.add(newInstance)
8      end
9      return artificialData
```

Tabla 6.1: Primitivas de Spark utilizadas en el desarrollo de SMOTE-BD.

Primitiva Spark	Descripción
textFile	Lee un archivo de texto desde un sistema de archivos, y lo retorna como un RDD de Strings.
filter	Selecciona todos los elementos de la estructura de datos que satisfacen un predicado.
map	Realiza una transformación a cada elemento de una estructura de datos.
MinMaxScaler	Utilidad para escalar los datos en un rango de valores.
repartition	Modifica el número de pariciones de una estructura de datos.
broadcast	Variable de sólo lectura en caché en cada nodo, en lugar de enviar una copia de la misma.
mapPartitionsWithIndex	Aplica una función a cada partición de la estructura de datos, manteniendo el índice de la partición original.
union	Combina dos estructuras de datos Spark por filas.

6.3. Análisis del comportamiento de SMOTE-BD

En esta sección se realiza un estudio experimental para mostrar la calidad de la implementación de sobremuestreo propuesta. En primer lugar, se presenta una comparación entre SMOTE-BD y la metodología secuencial estándar, la cual denominamos `SMOTE Seq` (disponible en la herramienta de software KEEL). Dicho análisis comparativo se lleva a cabo para conjuntos Small Data, con el objetivo de evaluar si nuestra propuesta se desempeña adecuadamente. A continuación, se contrasta el rendimiento y la escalabilidad del algoritmo para el escenario de datos Big Data. En este caso, no es posible aplicar la herramienta secuencial debido al tamaño y dimensionalidad de los conjuntos Big Data.

Para comparar el rendimiento de SMOTE-BD, se seleccionaron ocho conjuntos de datos no balanceados del repositorio KEEL, y se dividieron en dos categorías en función del número de ejemplos que contiene cada conjunto de datos, a saber: `Big Datasets` y `Small Datasets`. En la Tabla 6.2 se muestra el resumen de los conjuntos de datos, donde se incluye el número de ejemplos (#Ex.), el número de atributos (#Atts.), el nombre de cada clase (mayoritaria y minoritaria), el número de ejemplos de cada clase, la distribución de las clases y el ratio de desequilibrio (C2). Estos conjuntos de datos se dividieron siguiendo el método de validación cruzada k-fold, con k igual a 5 (5fcv).

El comportamiento de los datos preprocesados resultantes, es decir, los datos balanceados, se evalúa utilizando el clasificador DT, implementado en la librería MLlib de Spark[28]. El DT es entrenado utilizando los datos balanceados y, a partir del modelo generado, se clasifican las instancias del conjunto de prueba. El rendimiento predictivo se evalúa a través de la GM (definida en la Ecuación (3.2)). En la Tabla 6.3 se muestran los parámetros utilizados para ambos métodos de preprocesamiento. En cuanto a la infraestructura para realizar los experimentos de este capítulo, se ha utilizado el `cluster Hadoop` de la Universidad de Granada (véase el Apéndice A.2 para mayor detalle sobre la infraestructura hardware utilizada).

Los resultados de aplicar la implementación secuencial de SMOTE y SMOTE-BD en conjuntos Small Data se representan en la Figura 6.2. Además, con el fin de exhibirlos con mayor detalle, los mismos se muestran en la Tabla 6.4. Se puede observar que, para SMOTE-BD se han realizado pruebas con diferentes números de particiones de datos, de 1 a 4, dicha elección se basa en que los conjuntos utilizados son de tamaño estándar e incrementar el número de particiones podría generar que algunas de ellas no contengan ninguna instancia.

A partir de lo anterior se observa la capacidad de SMOTE-BD de conseguir

28 De ahora en adelante, siempre que se mencione que se ha utilizado al DT nos estamos refiriendo al de la implementación completamente escalable provisto por la librería MLlib de Spark

Tabla 6.2: Resumen de los conjuntos de datos para la evaluación de SMOTE-BD.

Big Datasets	#Ex.	#Atts.	Class (maj;min)	#Class(maj; min)	%Class(maj; min)	C2
covtype7	464677	54	(negative; positive)	(448421; 16256)	(96.5; 3.5)	27.58
poker0_vs_2	450022	10	(negative; positive)	(410960; 39062)	(91.32; 8.68)	10.52
poker0_vs_5	412600	10	(negative; positive)	(410960; 1640)	(99.60; 0.4)	250.58
susy_ir4	2712175	18	(negative; positive)	(2169740; 542435)	(80; 20)	4
Small Datasets	**#Ex.**	**#Atts.**	**Class (maj;min)**	**#Class(maj; min)**	**%Class(maj; min)**	**C2**
page-blocks0	5472	10	(negative; positive)	(4913; 559)	(89.78; 10.22)	8.78
segment0	2308	19	(negative; positive)	(1979; 329)	(85.75; 14.25)	6.02
shuttle-c0-vs-c4	1829	9	(negative; positive)	(1706; 123)	(93.28; 6.72)	13.88
yeast5	1484	8	(negative; positive)	(1440; 44)	(97.04; 2.96)	32.78

Tabla 6.3: Algoritmos y parámetros de SMOTE Seq y SMOTE-BD.

Algorithms	Parameters
SMOTE Seq	k Nearest Neighbors: 5 Distance function: Euclidean Desired C2: 1
SMOTE-BD	k Nearest Neighbors: 5 Distance function: Euclidean Desired C2: 1 Number of partitions: 1 / 2 / 3 / 4 / 8 / 32 Number of reducers: 1

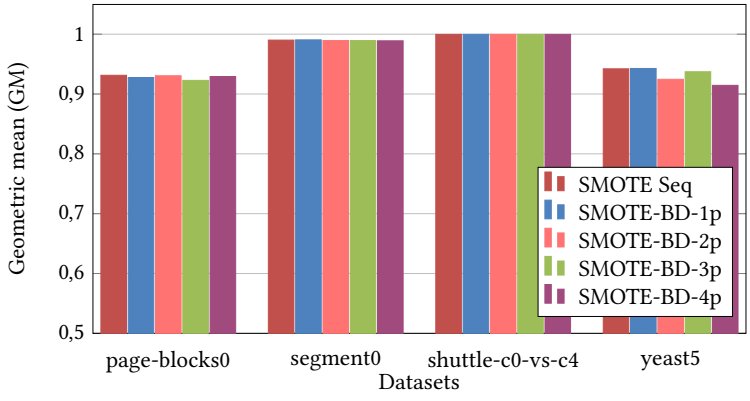


Figura 6.2: Resultados de la GM promedio de SMOTE secuencial (`SMOTE Seq`) y de SMOTE-BD sobre los conjuntos de datos pequeños.

una calidad predictiva comparable a la que se obtiene con la versión secuencial

Tabla 6.4: Resultados de la GM promedio de SMOTE secuencial (`SMOTE Seq`) y de SMOTE-BD sobre los conjuntos Small Data.

Datasets	SMOTE Seq	SMOTE-BD-1p	SMOTE-BD-2p	SMOTE-BD-3p	SMOTE-BD-4p
page-blocks0	0.9313	0.9276	0.9305	0.9228	0.9292
segment0	0.9901	0.9905	0.9893	0.9893	0.9889
shuttle-c0-vs-c4	0.9997	0.9997	0.9997	0.9997	0.9997
yeast5	0.9421	0.9427	0.9246	0.9372	0.9145

de SMOTE, demostrando ser una herramienta de preprocesamiento adecuada para balancear conjuntos de datos en entornos distribuidos.

Por otro lado, es válido aclarar que no se realiza una comparativa de los tiempos de ejecución ya que, para pequeños conjuntos de datos, la herramienta secuencial es más que suficiente para procesar dichos datos en tiempos razonables dado que caben perfectamente en memoria. Sin embargo, se refuerza la idea de que el foco de la comparación presentada anteriormente es evaluar que nuestra propuesta es capaz de conseguir resultados similares a la herramienta secuencial, independientemente del particionamiento de los datos (1 a 4 particiones) y demás características que se presentan al trabajar con un framework como es Spark. Por consiguiente, en caso de querer simular la ejecución secuencial para un conjunto Big Data, se puede utilizar SMOTE-BD con un número de particiones igual a 1 ya que se ha visto que trabaja adecuadamente obteniendo resultados compatibles a la solución secuencial para Small Data.

Respecto a los cuatro casos de estudio Big Data, en la Tabla 6.5 se muestran los resultados promedios de la GM para nuestra propuesta utilizando 1, 8 y 32 particiones. Frente a la imposibilidad de obtener resultados con SMOTE Seq para los `Big datasets` por limitaciones de escalabilidad, se utiliza en su lugar SMOTE-BD para 1 partición (y lo denominamos SMOTE-BD-1p).

Tabla 6.5: Resultados promedio de la GM obtenida de aplicar el algoritmo SMOTE-BD para 1, 8 y 32 particiones de datos, en combinación con el DT.

Datasets	SMOTE-BD-1p	SMOTE-BD-8p	SMOTE-BD-32p
covtype7	0.9227	0.9287	0.9181
poker0_vs_2	0.4582	0.4585	0.4713
poker0_vs_5	0.4162	0.4208	0.4206
susy_ir4	0.7615	0.7601	0.7585

Al aplicar nuestra propuesta para distintos números de particiones sobre el grupo `Big datasets` de esta experimentación, se refuerza la idea acerca de SMOTE-BD como solución escalable para tratar el problema de la clasificación

no balanceada en escenarios Big Data. Esto se aprecia en la estabilidad de los resultados promedio de la GM, obtenidos de ir variando el número de particiones a aplicar sobre los datos.

6.4. Un análisis de soluciones locales y globales para abordar la clasificación no balanceada de Big Data.

Hasta donde sabemos, no hay más soluciones disponibles de técnicas de preprocesamiento para problemas Big Data no balanceado que las que se comentan en la Sección 6.1. En particular, de todas ellas, las que han sido adaptadas a Apache Spark son ROS y RUS. Como contribución, en la Sección 6.2, se presentó una solución global de SMOTE desarrollada con Spark, SMOTE-BD, donde se muestra su correcto funcionamiento y que es totalmente escalable. En esta sección, en primer lugar se presenta una breve descripción de una nueva solución para Big Data de SMOTE (también implementada con Spark) pero, en este caso, siguiendo un enfoque de diseño local (y por ende, más eficiente pero inexacta), a la que denominamos SMOTE-MR. A continuación, en segundo lugar, se analiza el comportamiento de estas cuatro soluciones del estado del arte, distribuidas y escalables, para balancear un conjunto Big Data. Por consiguiente, se muestra una comparación del rendimiento de dichas soluciones en relación a parámetros de interés, como el número de particiones y el ratio de muestreo final.

Así pues, SMOTE-MR surge para poder disponer de una herramienta equivalente a la antes mencionada SMOTE-H del estado del arte, pero implementada bajo Spark para aprovechar todas sus ventajas. De manera general, SMOTE-MR sigue el mismo flujo de trabajo de SMOTE-BD que se muestra en la Figura 6.1. Además de ciertos detalles técnicos de diseño e implementación, la diferencia más significativa entre ambas aproximaciones radica al momento de obtener las instancias vecinas para la generación de las artificiales de clase minoritaria. En SMOTE-BD los vecinos de una instancia dada son los vecinos más cercanos de acuerdo al conjunto de datos global (motivo por el cual se usa la operación `broadcast`). Por su parte, en SMOTE-MR, los k vecinos más cercanos para un ejemplo de clase minoritaria se obtienen a partir de las instancias pertenecientes a la misma partición de dicho ejemplo. Por lo tanto, trabajando independientemente en cada Map, se obtienen resultados finales aproximados. Para la implementación de SMOTE-MR también se utilizó Spark 2.2.0 y el lenguaje de programación Scala. En https://github.com/majobasgall/smote-mr se puede encontrar su código fuente.

Por lo tanto, los cuatro algoritmos de preprocesamiento de conjuntos Big Data

que forman parte de este análisis son: ROS, RUS, SMOTE-BD y SMOTE-MR. A su vez, para comparar el rendimiento de los mismos, se seleccionaron tres conjuntos de datos no balanceados del repositorio UCI [Lic13], cada uno con un ratio de desproporción entre clases muy diferente. En la Tabla 6.6 se muestra un resumen de dichos conjuntos que incluye el número de instancias (*#Ex.*), número de características (*#Atts.*), número de instancias por clases (*#(maj; min)*), porcentaje de ejemplos por clase (*%(maj; min)*) y la proporción de desequilibrio (*C2*).

Tabla 6.6: Resumen de los conjuntos Big Data para la experimentación de las técnicas de remuestreo.

Datasets	#Ex.	#Atts.	#(maj; min)	%(maj; min)	C2
covtype7	464,677	54	(448,421; 16,256)	(96.5; 3.5)	27.58
HIGGS_IR_16	4,954,754	28	(4,663,298; 291,456)	(94.12; 5.88)	16
susy_ir4	2,212,186	18	(2,169,299; 542,435)	(80; 20)	4

Los parámetros utilizados para los métodos se presentan en la Tabla 6.7. Como se puede observar, se ha hecho una división para mostrar los parámetros comunes a todos los algoritmos y los específicos de los SMOTEs. El porcentaje de muestreo (*% sampling*) representa la distribución final deseada entre las clases, es decir, la proporción final. Por ejemplo, si $\%\ sampling = 100$, ambas clases tendrán la misma cantidad de instancias, es decir, una proporción de 1:1; y si $\%\ sampling = 150$, la clase minoritaria terminará con un 50 % más de instancias que la clase mayoritaria, es decir, una proporción de 1.5:1, y así sucesivamente. Como se comentó en la Sección 6.1, el porcentaje de muestreo ha influido significativamente sobre los resultados a medida que aumentan [Gut+17a]. En consecuencia, se seleccionaron tres valores para este parámetro con el fin de probar cada uno de ellos. El número de particiones (*# partitions*) establece la cantidad de porciones en la que se dividirán los datos de entrada. En cuanto a las dos versiones de SMOTE, ya que en ellas se calculan los k vecinos más cercanos de cada instancia minoritaria, se propone utilizar como valor del parámetro $k = 5$. En ambos casos también se utiliza la distancia Euclídea.

El comportamiento de los conjuntos de datos preprocesados resultantes de aplicar cada una de las técnicas se probó utilizando el clasificador DT. La métrica de evaluación seleccionada es GM, descripta en la Sección 3.3.

Los resultados promedios, a partir de la validación cruzada 5fcv, del rendimiento alcanzado (representado por la GM) por un DT tras aplicar de forma independiente cada una de las técnicas de preprocesamiento sobre cada conjun-

Tabla 6.7: Algoritmos y parámetros para las técnicas de remuestreo.

Algoritmos	Parámetros	Valores
All	% sampling	100 / 150 / 200
	# partitions	32 / 64 / 128
SMOTEs	k Nearest Neighbors	5
	Distance function	Euclidean

to de datos, para cada porcentaje de muestreo y número de particiones antes mencionados, se presentan a continuación.

En la Tabla 6.8 se muestran los resultados para todos los conjuntos de datos preprocesados con cada uno de los cuatro métodos con 32, 64 y 128 particiones de Spark, quedando fijo el porcentaje de muestreo en 100 %.

Tabla 6.8: Resultados promedio de la GM para los cuatro métodos, para 32, 64 y 128 particiones y porcentaje de muestreo de 100 %.

Dataset	Method / Part.	SMT-BD	SMT-MR	ROS	RUS
HIGGS_IR_16	32	0.6462	0.6477	0.6580	0.6560
	64	0.6468	0.6474	0.6562	0.6505
	128	0.6479	0.6476	0.6561	0.6568
covtype7	32	0.9249	0.9242	0.9363	0.9363
	64	0.9251	0.9279	0.9232	0.9271
	128	0.9306	0.9274	0.9234	0.9258
susy_ir4	32	0.7650	0.7671	0.7675	0.7633
	64	0.7671	0.7681	0.7685	0.7666
	128	0.7643	0.7671	0.7670	0.7655

Para todos los conjuntos se puede ver que los resultados de aplicar una cierta técnica de preprocesamiento son compatibles independientemente del número de particiones elegidas. Es decir, se puede ver que todas las soluciones son escalables en calidad predictiva independientemente del grado de paralelismo con el que se esté trabajando. Este comportamiento ya se había notado y comentado para SMOTE-BD en la sección anterior, y aquí nuevamente se puede apreciar no sólo para otro conjunto de datos más (HIGGS_IR_16), sino también para un mayor número de particiones. En el caso concreto de SMOTE-MR y las técnicas de preprocesamiento aleatorio, no existían antecedentes experimentales de que se

cumpliera esta condición que resulta claramente positiva para Big Data. Resulta necesario indicar que, en el caso particular de SMOTE-MR, el hecho de tratarse de una implementación basada en un diseño local, hacía considerar la hipótesis sobre la degradación de los resultados con respecto al incremento del número de particiones; la razón es que al utilizarse las instancias positivas exclusivamente de la partición actual, debía existir una tendencia a la sobregeneralización. Esta situación nos hace considerar que puede existir un alto nivel de redundancia en los datos, y que por tanto los clústeres de cada clase estén ampliamente poblados por lo que, salvo para valores extremos en el número de particiones, el comportamiento de SMOTE-BD y SMOTE-MR será similar.
A su vez, de la tabla también se observa que, entrenar un DT a partir de los datos preprocesados por cualquiera de las técnicas de esta experimentación, alcanza un rendimiento predictivo similar de aplicar una u otra. Sin embargo, hay que tener en cuenta que, en otros conjuntos de datos, esta situación puede que no ocurra ya que podría suceder que la aplicación de una técnica genere resultados superadores al resto, tal como se comenta en la Sección 4.1. Para finalizar este apartado del estudio, es necesario mencionar que en nuestra experimentación dicho comportamiento se repite también para el resto de los porcentajes de muestreo utilizados. Por cuestión de claridad y simplificación de contenido en este capítulo, se han mostrados únicamente valores para 100 % de sampling (es decir, ratio 1:1).

Respecto al análisis con foco en los distintos porcentajes para la proporción final entre clases, en la Tabla 6.9 se presentan los resultados para cada conjunto, cada método, cada porcentaje de muestreo y el número de particiones igual a 32.

Tabla 6.9: Resultados promedio de la GM para los cuatro métodos, para 32 y porcentaje de sobremuestreo de 100, 150 y 200.

	Part.	32			
	Method	SMT-BD	SMT-MR	ROS	RUS
Dataset	Perc.				
HIGGS_IR_16	100	0.6462	0.6477	0.6580	0.6560
	150	0.6172	0.6157	0.6271	0.6223
	200	0.6006	0.5973	0.5334	0.5245
covtype7	100	0.9249	0.9242	0.9363	0.9363
	150	0.9250	0.9232	0.9123	0.9123
	200	0.9236	0.9190	0.9186	0.9110
susy_ir4	100	0.7650	0.7671	0.7675	0.7633
	150	0.7528	0.7578	0.7667	0.7389
	200	0.7350	0.7357	0.7364	0.7191

En general se observa que los mejores rendimientos se alcanzan de aplicar un equilibrio absoluto sobre los datos. Para mejor visualización de estos resultados, se colorea con verde los valores correspondientes al porcentaje del 100 % sólo en aquellos casos que supera al resto de los porcentajes de muestreo. Del mismo modo ocurre con el resto de los números de particiones elegidos para esta experimentación (64 y 128), que se omiten deliberadamente por cuestión de espacio y claridad pero que se pueden consultar en la publicación original que se referencia al final de este capítulo.

En cuanto a los tiempos de ejecución, y como es esperado, RUS y ROS son los que mejor se comportan (en ese orden, considerando la etapa de entrenamiento y clasificación del DT), debido a la simplicidad de sus algoritmos. A continuación, sigue comprensiblemente SMOTE-MR, en virtud de la naturaleza local de su enfoque. Los datos particionados se procesan localmente en cada nodo, lo que se traduce en tiempos más bajos. La herramienta SMOTE-BD presenta los tiempos más altos de entre los algoritmos probados, utilizando un enfoque de datos distribuidos, pero garantizando resultados exactos independientemente de la escalabilidad deseada.

6.5. Comentarios del capítulo

En este capítulo se han comentado las herramientas existentes para equilibrar grandes conjuntos de datos de problemas de clasificación binaria. Asimismo, se ha presentado un diseño totalmente escalable del tradicional algoritmo de sobremuestreo del estado del arte, SMOTE, siguiendo un enfoque global, al cual denominamos SMOTE-BD. Dos razones principales han motivado el diseño para un SMOTE capaz de procesar Big Data de manera totalmente distribuida. Por un lado, la falta de soluciones actuales para un área de estudio tan importante como es el preprocesamiento de los datos y, en particular, el balanceo de clases. Por otro lado, considerar un procedimiento global que tenga en cuenta todo el vecindario de cada instancia de clase minoritaria. Los experimentos realizados en diferentes conjuntos Small Data y Big Data muestran la calidad del diseño y la implementación propuestos. De la experimentación se observa, que el proceso de generación de instancias con el fin de balancear las clases de un problema que realiza SMOTE-BD se desempeña adecuadamente, alcanzando los mismos resultados que la solución secuencial en conjuntos de tamaño tradicional. Además, a partir de la estabilidad alcanzada en los resultados obtenidos de aplicar distinto grado de paralelismo (tanto en datos de tamaño tradicional como en Big Data), se observa que es un algoritmo totalmente escalable.

A su vez, se ha introducido una variante de SMOTE-BD que sigue un diseño local, y a la cual llamamos SMOTE-MR. La misma surge como una alternativa más

actual a un SMOTE previo, implementado en Hadoop MapReduce (framework del cual se menciona sus principales debilidades en el Capítulo 5). Así pues, ambas versiones de SMOTE presentadas en este capítulo han sido desarrolladas utilizando el framework Apache Spark y el lenguaje de programación Scala.

También, se ha llevado a cabo un análisis de nuestras dos versiones de SMOTE y las soluciones aleatorias, ROS y RUS (también desarrolladas en Spark y disponibles en un repositorio público), en función del grado de muestreo y el número de particiones de datos. De los resultados se observa que todas las técnicas utilizadas son escalables por mantener valores de calidad predictiva compatibles, independientemente del número de particiones elegidas. Esto ya se había visto para SMOTE-BD pero en este análisis se refuerza con el mismo comportamiento para un conjunto de datos más, y además se observa misma situación para las técnicas restantes. Sin embargo se aclara que no tiene por qué ser siempre de este modo, sino que se han visto casos en la bibliografía en donde una técnica puede generar un mejor desempeño que otras. Por lo tanto, frente a un problema no balanceado, usualmente se aplican al menos estas técnicas de preprocesamiento para encontrar cuál es más beneficiosa para cada conjunto de datos en particular. Es por esto que nuestro aporte toma mayor relevancia dado que, hasta el momento de su desarrollo, sólo estaban disponibles las soluciones aleatorias. Este hecho evidenciaba la necesidad de contribuir con el desarrollo de otras soluciones tan populares del estado del arte en Small Data y que sean capaces de procesar Big Data de manera distribuida y rápida, mediante el uso de un framework tan popular en Big Data Analytics, como es Apache Spark.

Por todo lo anterior, el tratamiento para el equilibrio de clases en escenarios Big Data es una línea de investigación en desarrollo, de la que se requiere más contribuciones escalables, ya sea de las técnicas de uso común en Small Data como otras soluciones. En futuras investigaciones sobre esta línea se puede llevar a cabo el desarrollo de modelos híbridos centrados en las zonas donde el remuestreo es especialmente necesario. En particular, las zonas de interés son aquellas con presencia de pequeños disjuntos y solapamientos, de las cuales primero tienen que poder ser detectadas adecuadamente dentro de un gran volumen de datos, con probablemente una alta dimensionalidad de los mismos, y en tiempos de procesamiento razonables.

Los resultados presentados en este capítulo forman parte de los siguientes artículos:

Basgall, M. J., Hasperué, W., Naiouf, M., Fernández, A., & Herrera, F. (2019). *An Analysis of Local and Global Solutions to Address Big Data Imbalanced Classification: A Case Study with SMOTE Preprocessing.* Cloud Computing and Big Data (Vol. 1050, pp. 75–85). Springer International Publishing. https://doi.org/10.1007/978-3-030-27713-0_7

Basgall, M. J., Hasperué, W., Naiouf, M., Fernández, A., & Herrera, F. (2018). *SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data.* Journal of Computer Science and Technology, 18(03), e23. https://doi.org/10.24215/16666038.18.e23.

7 Reducción escalable en escenarios Big Data

En este capítulo, se analiza la reducción (también llamada condensación) de datos en Big Data. Las técnicas de preprocesamiento de reducción de datos juegan un rol importante en los problemas de Big Data por su capacidad de obtener una versión reducida de un conjunto de datos de gran tamaño, manteniendo la representatividad de los datos originales. Dado que trabajar con un conjunto Big Data impone restricciones por los importantes requerimientos de recursos computacionales para poder procesarlo, es necesario y ventajoso contar con técnicas escalables de condensación de datos. En la Sección 7.1 se presenta una visión general de las técnicas de reducción de datos en el contexto de Big Data, particularmente, datos tabulares que representan problemas de clasificación binaria. Se analizan las soluciones escalables disponibles y sus principales características. A continuación, en la Sección 7.2, se detalla nuestra propuesta metodológica para la reducción escalable de grandes conjuntos de datos basada en características y en instancias, la cual se denomina Fast Data Reduction Recommendation tool for tabular Big Data (FDR^2-BD). Nuestro estudio experimental se desarrolla en la Sección 7.3, donde se evalúa nuestra metodología y se la compara con técnicas del estado del arte en esta temática, demostrando que su rendimiento es superior desde distintos aspectos. Por último, en la Sección 7.4, se comenta brevemente el capítulo.

7.1. Condensación de datos en Big Data

Cuando se analizan los problemas de Big Data tabulares disponibles públicamente para la clasificación binaria, se puede observar la presencia de una notable redundancia conceptual de información en los datos [MTH20] (instancias y/o características redundantes) que conlleva un coste computacional innecesario. Además, se sabe que los clasificadores sólo necesitan un conjunto de instancias que representen correctamente un problema para generar un modelo adecuado [Aca+18]. En este sentido, mantener los datos originales (en bruto), incluyendo estas características e instancias redundantes, tiene un fuerte impacto negativo relacionado con la escalabilidad y el almacenamiento.

Además de lo anterior, no todos los usuarios tienen acceso directo a la infraestructura computacional adecuada para Big Data o la capacidad de trabajar con frameworks dedicados a Big Data, como Apache Spark [Zah+16a]. Es posible

alquilar un servicio externo de computación en la nube, pero esto puede suponer un coste excesivo a la hora de realizar todo el ciclo de ciencia de datos. Además, hay que tener en cuenta que existen algoritmos de ML de última generación que aún no han sido desarrollados para ser escalables. En cambio, se pueden encontrar varias implementaciones para conjuntos de datos de menor tamaño en paquetes estándar para Python, R, entre otros lenguajes de programación.

La aplicación de técnicas de preprocesamiento centradas en la condensación/reducción de datos, como la selección de características e instancias [LM07; LM01; Cai+18; Ros+21], ayuda a hacer frente a dicha redundancia conceptual no deseada. Disponer de una versión reducida de los datos originales manteniendo la mayor cantidad de información posible es siempre una situación deseable [HFB19; Lue+20], pero aún más en los problemas de Big Data en los que están implícitas tareas de procesamiento que requieren mucho tiempo. Para los conjuntos de datos de tamaño estándar, es decir, los conjuntos de datos Small Data, se han diseñado un sinfín de propuestas de reducción de datos [Tri+12; Gar+12]. Sin embargo, la mayoría de ellas se centran o bien en la selección de las características más representativas o bien en las instancias. Pocas propuestas se basan en una combinación o sinergia entre ambos métodos de filtrado [PAG15].

A pesar del gran número de técnicas de reducción de datos disponibles en escenarios de Small Data, no hemos podido encontrar la misma situación para el área de Big Data Analytics. La mayoría de los enfoques estándar de reducción de datos actuales para Big Data son altamente costosos en términos de tiempo de ejecución y recursos, debido principalmente al uso de soluciones basadas en clustering y distancias (complejidad algorítmica $O(dn^2)$), que no son fáciles de computar de forma distribuida sin ser aproximadas.

En este contexto, se pueden encontrar pocas soluciones distribuidas a los enfoques tradicionales de reducción basados en instancias. Algunas de ellas utilizan el framework MRPR [Tri+15b] el cual trabaja siguiendo el modelo de procesamiento MapReduce. El propósito de MRPR es distribuir el funcionamiento de algoritmos de reducción de prototipos (*Prototype Reduction* (PR)) a través de un clúster de elementos informáticos, proponiendo varias estrategias algorítmicas para integrar múltiples soluciones parciales (conjuntos reducidos de prototipos) en una sola. El modelo propuesto permite aplicar los algoritmos de PR sobre problemas de clasificación de Big Data sin una pérdida de precisión significativa. De manera sintetizada, MRPR divide los datos de entrada y, siguiendo un enfoque local de MapReduce, se aplica a cada subconjunto de datos la tarea IS correspondiente. A continuación, se dispone de diferentes estrategias para fusionar el conjunto reducido de datos obtenido por cada Map.

De las soluciones existentes descriptas en la Sección 4.2, tanto las técnicas FCNN rule como SSMA se han hecho escalables (*FCNN for Big Data* (FCNN_MR)

[Gar+19] y SSMA-SFLSDE [TGH11], respectivamente) utilizando el framework anteriormente mencionado. De similar manera, la implementación paralela de DIS (MR-DIS [Arn+17]) también sigue un enfoque local, mientras que la de RMHC (RHMC_MR [Gar+19]) aplica el algoritmo *k*NN repetidamente siguiendo un enfoque global. En [Tri+19], los autores han comparado los métodos de IS para Big Data, y han llegado a la conclusión de que no hay un método claramente superador en general. Sin embargo, en lo que respecta a su tiempo de ejecución, FCNN_MR es el único enfoque cuyo coste computacional está dentro de un rango aceptable .

Con respecto a los métodos de reducción de características disponibles para Big Data, en [Ram+18a], se pueden encontrar varios métodos del estado del arte basados en la teoría de la información implementados de manera distribuida. Además, la implementación totalmente paralela de RF se incluye como parte de los algoritmos de ML disponibles que proporciona la librería MLlib [Men+16] de Apache Spark.

7.2. FDR^2-BD: Una herramienta rápida de recomendación de reducción de datos para problemas de clasificación de Big Data tabular

Dado el estado del arte sobre técnicas para la condensación de grandes volúmenes de datos, es evidente la falta de soluciones escalables para este fin, más aún la falta de ellas en dirección a la reducción dual (horizontal y vertical) de los datos. En esta sección, se presenta y describe FDR^2-BD, una metodología para la condensación de información en problemas de clasificación de Big Data tabular. El resultado es una herramienta de apoyo a la decisión que proporciona un conjunto de parámetros recomendados, en términos de valores de reducción de características e instancias, mediante un procedimiento que involucra selección de características, validación cruzada y un proceso de hiperparametrización (como se muestra en la Figura 7.1). Así, los valores de salida mencionados están bien soportados, ya que se agregan/obtienen a partir de diferentes subconjuntos de los datos de entrenamiento.

Los objetivos de esta propuesta son los siguientes:

(a) Determinar si un conjunto de datos es reducible o no mediante un submuestreo estratificado uniforme.

(b) Estimar el máximo porcentaje de reducción posible y las características más importantes de un conjunto de datos.

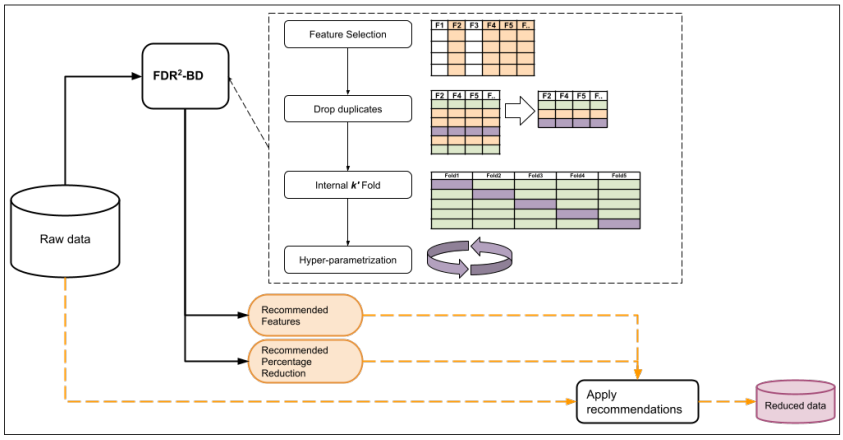


Figura 7.1: Esquema general de FDR^2-BD.

Ambas respuestas se basarán en el mantenimiento de un rango de calidad predictiva sobre los datos brutos originales, determinado por un umbral. El fundamento de este requisito es que la simple disminución de los datos no sirve de nada si los datos resultantes no son representativos de la información contenida en el conjunto de datos original. Así, una capacidad muy importante de nuestra propuesta es incluir el mencionado umbral de pérdida aceptable en la calidad predictiva que actúa como "controlador" entre la tasa de reducción y los valores de predicción obtenidos. En algunos casos, la necesidad de reducir los datos es tal que los usuarios están dispuestos a aceptar un ligeramente inferior poder predictivo que el que se consigue con los datos originales a cambio de una submuestra de los datos que les permita tratarlos con mayor rapidez y eficacia.

7.2.1. Descripción y flujo de trabajo

El diseño completo de la metodología de FDR^2-BD aborda tres requisitos que consideramos relevantes para las soluciones de reducción de datos:

1. En primer lugar, la capacidad de operar automáticamente mediante un proceso de hiperparametrización, que se basa en una lista ordenada descendente de Porcentajes de Reducción Recomendados (*Recommended Percentage Reduction* (RPR)). Para ello, trabaja sobre un conjunto de datos siguiendo un esquema de validación cruzada para asegurar que se analizan todos los datos.

2. En segundo lugar, la posibilidad de alcanzar el máximo nivel de reducción de datos mediante el uso de umbrales de pérdida aceptable de rendimiento

predictivo. Tanto la lista de los porcentajes (`listaPerc`) como el umbral de pérdida predictiva (*Predictive Loss Threshold* (PLT)) influyen en las condiciones de finalización del proceso de hiperparametrización, y pueden ser fijados por el usuario experto en los datos.

3. Por último, la capacidad de ser una metodología escalable capaz de tratar con Big Data en tiempos razonables, aprovechando la computación en paralelo. Para conseguirlo, los detalles de la implementación técnica se describen en la Sección 7.2.2.

En esta metodología, el usuario puede configurar la lista `listaPerc` para establecer una mayor o menor granularidad en los RPR a utilizar, así como también establecer cuánta calidad predictiva está dispuesto a perder para ganar en reducción de datos. Esto comprende el escenario a partir del cual se considera que el conjunto de datos no es reducible debido a una gran pérdida en el rendimiento del modelo. Entre las diferentes medidas de rendimiento predictivo, esta propuesta utiliza la GM (véase la Ecuación (3.2)) porque consolida la calidad de todas las clases representadas, independientemente de su distribución a priori.

En el flujo de trabajo de FDR^2-BD, tal y como se representa en la Figura 7.2, se puede distinguir tres etapas diferentes que se enumeran a continuación:

1. En primer lugar, se puede lograr una reducción inicial de la dimensionalidad (reducción vertical o por columnas) mediante la selección de las características más importantes.

2. En segundo lugar, si la etapa de FS genera instancias redundantes, es decir, filas exactamente duplicadas, éstas se eliminan (reducción horizontal de filas).

3. En tercer lugar, el proceso principal se lleva a cabo mediante una hiperparametrización con el objetivo de decidir qué porcentaje de instancias puede eliminarse manteniendo un umbral de calidad deseado. A partir de la decisión resultante, se puede llevar a cabo una reducción vertical final e intensiva.

En lo que sigue, se detalla cada unoa de las etapas enumeradas del procedimiento.

Primera etapa: Selección de características. Algunos de los atributos de este tipo de conjuntos de datos pueden no ser representativos ni necesarios para la clasificación del ejemplo y, por tanto, pueden omitirse. Con esta reducción

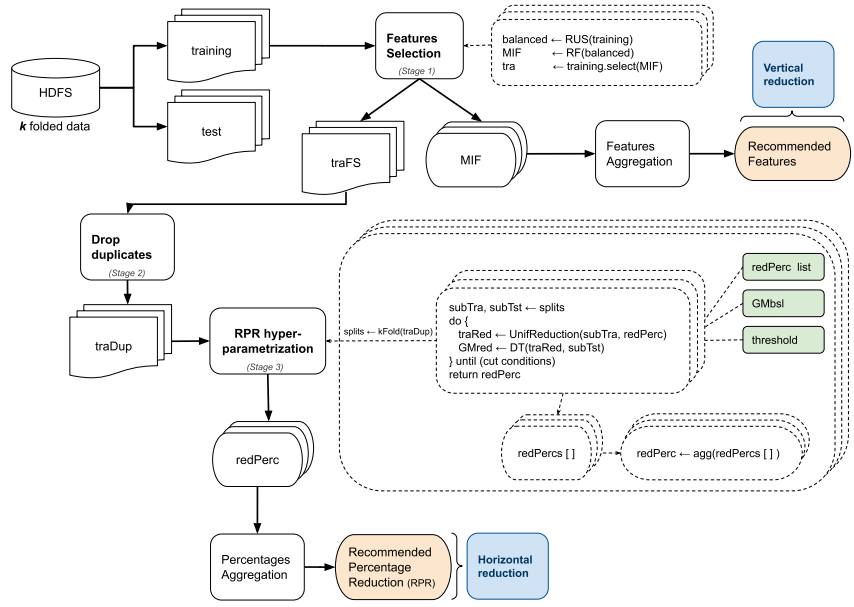


Figura 7.2: Flujo de trabajo de FDR^2-BD mostrando las tres etapas principales del procedimiento.

vertical, la redundancia conceptual puede encontrarse fácilmente como instancias duplicadas.

Los datos de entrada iniciales siguen el método de validación cruzada *k*-fold, con k igual a 5 (5fcv), por lo que para cada *fold*/pliegue o subconjunto de entrenamiento se aplica la tarea de FS. Nuestro diseño tiene en cuenta el posible caso de una distribución no balanceada de las clases. De hecho, esto puede causar un sesgo de la importancia de la características hacia los ejemplos de la clase mayoritaria. Para hacer frente a este problema, la etapa FS incluye una tarea de preprocesamiento de instancias para esos conjuntos de clases desiguales, con el objetivo de transformarlos en conjuntos equilibrados. En nuestra metodología, se utiliza el enfoque estándar llamado RUS para conseguir un equilibrio de clases perfecto.

Para la identificación de la importancia de las características se utiliza un modelo embebido, debido a su mayor robustez en comparación con el uso de valores estadísticos. Entre los diferentes modelos, se elige a RF por su gran diversidad y por ser un modelo ensemble, además por ser totalmente escalable y nativo de Spark como se ha mencionado anteriormente. Finalmente, para definir las características más importantes (denominadas como *Most Important Features* (MIFs)), se calcula la varianza explicada acumulada; a partir de ello, las

características elegidas son aquellas que pueden cubrir hasta el 90 % de dicha varianza.

Los resultados de esta etapa son los k conjuntos de entrenamiento reducidos verticalmente y sus correspondientes k MIFs. A su vez, a partir de estos últimos, se realiza una agregación con el objetivo de obtener el conjunto de características recomendado en todos los pliegues. Para ello, el criterio utilizado es encontrar las características que fueron seleccionadas como importantes en más del 50 % de los k pliegues.

Segunda etapa: Eliminación de duplicados. La reducción de la dimensionalidad del problema hace que los clusters de datos aumenten su densidad, e incluso se pueden encontrar instancias duplicadas en el conjunto de datos. Por lo tanto, es muy necesario eliminar los ejemplos redundantes (manteniendo sólo una ocurrencia de cada uno) para evitar su influencia negativa, que puede afectar o distorsionar el procesamiento durante la siguiente etapa.

Tercera etapa: Hiperparametrización RPR. El objetivo de esta etapa es obtener el máximao porcentaje de reducción posible de las instancias según un umbral cualitativo de calidad predictiva. Para ello, se realiza un proceso de validación cruzada interno (Internal ***k'*** Fold) mediante la subdivisión de los k conjuntos no redundantes (generados por la etapa anterior) en k' pliegues. Cada subpliegue se representa como k' pares de conjuntos de entrenamiento y de prueba (denominados `subTra` y `subTst`), y se analizan mediante un proceso de hiperparametrización de muestreo uniforme de datos para encontrar k valores de porcentaje de reducción.

El proceso de hiperparametrización considera cada `subTra` y aplica un procedimiento iterativo realizando una reducción de datos estratificada basada en un submuestreo uniforme (`UnifReduction`). Cada paso de este procedimiento considera un nivel de reducción según el elemento actual de la lista `redPerc`. En la Sección 7.3.1 se sugiere una definición por defecto de esta lista junto con parámetros adicionales del algoritmo propuesto.

Para determinar si el procedimiento iterativo ha finalizado, se consideran dos condiciones: por un lado, que la evaluación del rendimiento del conjunto de datos reducido resultante esté fuera del umbral deseado y, por otro lado, que no haya más valores en la lista `redPerc`, indicando que el problema no puede condensarse en ninguno de los porcentajes de `redPerc` sin que se vea afectado el rendimiento predictivo.

Para comprobar la calidad del conjunto reducido (`traRed`), se entrena un DT utilizando `traRed`, y luego se clasifica el conjunto `subTst`. El rendimiento

predictivo se evalúa a través de la GM (denominada `GMred`), y se compara con el rango determinado por la GM media del método *baseline* (`GMbsl`) y por el PLT.

Así, el proceso itera generando diferentes tamaños de conjuntos de datos reducidos hasta que se alcanza una de las condiciones de parada. Si la `GMred` actual está dentro del rango deseado de rendimiento predictivo, su porcentaje de reducción correspondiente será la salida, y como cada uno de ellos se obtuvo del procedimiento de *k'* subpliegues, es necesario agregarlos para para tener un único valor de salida.

Dado que la herramienta FDR^2-BD funciona siguiendo un esquema de hiperparametrización, es necesario proporcionar algún criterio de agregación para fusionar todos los parámetros de salida individuales obtenidos durante el proceso. En concreto, se siguen dos enfoques complementarios: (1) se selecciona el porcentaje de reducción mínimo, siendo un enfoque más conservador, y (2) se elige el porcentaje de reducción medio. Sea cual sea el criterio que se establezca, la salida de este paso es un porcentaje de reducción por cada *k'*. A continuación, se realiza una tarea más de agregación sobre las salidas anteriores para obtener el porcentaje de reducción recomendado por la herramienta (RPR) para el conjunto de datos de entrada con el umbral de pérdida predictivo (PLT) deseado.

Cabe mencionar que, como FDR^2-BD actúa como una herramienta de recomendación de parámetros, no genera un conjunto de datos reducido, sino que devuelve un par de parámetros de recomendación para modificar el conjunto de datos con el fin de reducirlo de forma dual. En otras palabras, hasta que el usuario no aplique los pasos que se muestran en el pseudocódigo del Algoritmo 3, no se realizará la reducción de los datos. Tras condensar los datos aplicando la reducción de características e instancias recomendada (líneas 3 y 4, respectivamente), se entrena un único modelo a partir de ellos con el objetivo de obtener los resultados de la clasificación. Nótese que a partir del paso indicado en la línea 3, puede no ser necesario utilizar un framework de Big Data para realizar estas acciones, siempre y cuando los datos quepan en la memoria principal.

7.2.2. Resumen de la implementación técnica

FDR^2-BD fue desarrollada utilizando el lenguaje de programación Scala bajo el framework Apache Spark para aprovechar sus bondades, como se describen en la Sección 5.4. El diseño de nuestra propuesta sigue un enfoque global que considera todo el conjunto de datos a la vez, generando resultados exactos. La versión de Spark elegida es la 3.0.1 debido a que supera a las versiones anteriores en escalabilidad, no sólo en sus funcionalidades principales sino también en sus librerías. En cuanto a las estructuras de datos, se seleccionaron los Dataframes y los Datasets por encima de la estructura de datos base (RDD), ya que proporcionan

Algoritmo 3: Pseudocódigo que muestra el uso de FDR^2-BD para la reducción de datos.

Require: data, redPercList, GMbsl, PLT, k, aggCriterion

1: (recommFeats, RPR) ←
   ejecutar FDR^2-BD usando los parámetros de entrada
2: (training, test) ← dividir `data`
3: tra ← seleccionar las características `recommFeats` a partir de `training`
4: condensedTra ← aplicar UniformReduction sobre `tra` con `RPR`
5: model ← entrenar el clasificador usando `condensedTra`
6: results ← clasificar `test` a partir de `model`

los beneficios de los RDD con una optimización extra, lo que significa que son los más rápidos. Es importante destacar que las etapas del flujo de trabajo de nuestra herramienta fueron implementadas utilizando tanto las primitivas de Spark como algunos algoritmos y utilidades para construir pipelines de ML distribuido proporcionados por la librería MLlib. Todas ellas son totalmente eficientes y robustas, y facilitan el propósito de hacer nuestra propuesta totalmente escalable. Las primitivas y funcionalidades más relevantes utilizadas en nuestro código se detallan en la Tabla 7.1[29].

Al ser una herramienta de código libre, la implementación de FDR^2-BD se encuentra disponible en el siguiente repositorio: https://github.com/majobasgall/big_data_reduction_recommender.

Tabla 7.1: Primitivas y utilidades de Spark utilizadas en FDR^2-BD.

Operación Spark	Descripción
map	Realiza una transformación a cada elemento de una estructura de datos
filter	Selecciona todos los elementos de la estructura de datos que satisfacen un predicado
sample	Toma una muestra de un conjunto de datos
union	Combina dos estructuras de datos Spark por filas
kFold	Divide los datos en k pares de conjuntos de entrenamiento y validación, siguiendo un muestreo Bernoulli
dropDuplicates	Elimina las filas duplicadas de un dataframe
RandomForestClassifier	Son los algoritmos de aprendizaje distribuidos para la clasificación de datos
DecisionTreeClassifier	

7.3. Estudio experimental

En esta sección, se valida experimentalmente nuestra propuesta FDR^2-BD. En primer lugar, se comenta nuestro entorno experimental en la Sección 7.3.1,

29 En el Apéndice A.4 se encuentra un compendio de todas las primitivas y funcionalidades utilizadas en esta tesis.

compuesto por los conjuntos de datos e infraestructura utilizada, los métodos de comparación, y las medidas de evaluación. En la Sección 7.3.2, se presenta el estudio relativo a la reducción del volumen de datos generado por el uso de FDR^2-BD. A continuación, en la Sección 7.3.3 se muestra la influencia de FS con respecto a la reducción horizontal. Luego, en la Sección 7.3.4, se lleva a cabo la comparación entre los diferentes algoritmos mencionados en la descripción del entorno experimental de trabajo. Este estudio comparativo se centra en los detalles de condensación y en el rendimiento predictivo alcanzado. Por último, el estudio de escalabilidad de nuestra propuesta se presenta en la Sección 7.3.5.

7.3.1. Entorno de trabajo

Conjuntos de datos e infraestructura

Con el fin de analizar la mayoría de los datos tabulares más utilizados en los problemas de clasificación de Big Data, se seleccionaron 25 conjuntos de datos de diferentes repositorios (UCI machine learning [Lic13], OpenML [Van+13], Kaggle [tea]). Una descripción resumida de cada conjunto de datos ordenados alfabéticamente se muestra en la Tabla 7.2[30], la cual contiene el número de ejemplos (#Ex.), número de atributos (#Atts.), el número de instancias de cada clase (#(class0; class1)), el porcentaje de distribución de las clases (%(class0; class1)), el número de cada tipo de característica (Co/Di/Ca, donde Co = continuas, Di = discretas, Ca = categóricas), y el tamaño del conjunto de datos (en MB).

Los conjuntos de datos utilizados para este análisis se dividieron siguiendo el método de validación cruzada *k*-fold, con k igual a 5 (5fcv).

Respecto a la infraestructura, la experimentación se llevó a cabo utilizando el `cluster Atlas`[31], el cual se describe en la Sección "Infraestructura" del Apéndice A.2.

Métodos de comparación, clasificador y parámetros

Como clasificador base para el estudio experimental, se utiliza el DT. Dos razones significativas y relacionadas entre sí apoyan la decisión de seleccionar este clasificador. Por un lado, la implementación de este algoritmo en Spark se basa en un diseño global que hace que los resultados sean más robustos, al utilizar todos los datos de entrenamiento a la vez. Por otro lado, tiene la ventaja de ser

30 En el Apéndice A se encuentra una descripción de cada conjunto de datos utilizado en esta tesis.

31 Dicha infraestructura hardware es la que utilizó también en las experimentaciones que siguen y se presentan en el resto de la memoria de tesis.

Tabla 7.2: Resumen de los 25 conjuntos Big Data utilizados en la experimentación de FDR^2-BD.

Dataset	#Ex.	#Atts.	#(class0; class1)	%(class0; class1)	Co/Di/Ca	Size (MB)
Agrawal	1,000,000	9	(672,044; 327,955)	(67.2; 32.8)	4/2/3	71.6
airlines	539,383	7	(299,119; 240,264)	(55.46; 44.54)	4/0/3	18.3
BNG_Australian	1,000,000	14	(573,051; 426,949)	(57.31; 42.69)	14/0/0	80.8
BNG_heart	1,000,000	13	(555,946; 444,054)	(55.59; 44.41)	6/0/7	65.7
census	140,246	41	(132,085; 8162)	(94.18; 5.82)	1/12/28	68.7
click_prediction	1,963,972	11	(1,636,593; 327,379)	(83.33; 16.67)	0/11/0	136.3
covtype1	581,012	54	(369,307; 211,705)	(63.56; 36.44)	10/0/44	71.7
covtype1_vs_2	495,173	54	(283,468; 211,705)	(57.25; 42.75)	10/0/44	61.1
covtype2	581,012	54	(297,544; 283,468)	(51.21; 48.79)	10/0/44	71.7
creditCard	284,015	30	(283,540; 480)	(99.83; 0.17)	28/1/0	145.2
ECBDL14-10mill-90	12,000,000	90	(11,760,000; 240,000)	(98; 2)	60/0/30	3481.6
ethylene_ECO_E_LH	4,208,261	16	(3,895,861; 312,400)	(92.58; 7.42)	16/0/0	517
ethylene_EM_E_LH	4,178,500	16	(3,840,313; 338,191)	(91.91; 8.09)	16/0/0	518.7
fars_fatal	100,968	29	(58,852; 42,116)	(58.29; 41.71)	0/1/28	59
HEPMASS_IR_16	5,578,255	28	(5,250,122; 328,133)	(94.12; 5.88)	28/0/0	1331.2
higgs	11,000,000	28	(5,827,686; 5,172,314)	(52.98; 47.02)	28/0/0	7680
HIGGS_IR_16	6,193,440	28	(5,829,120; 364,320)	(94.12; 5.88)	28/0/0	3378.7
homeCredit	307,511	171	(282,686; 24,825)	(91.93; 8.07)	1/9/161	113.1
hyperplane	1,000,000	10	(500,007; 499,993)	(50; 50)	10/0/0	91.4
klaverjas	981,541	34	(528,339; 453,202)	(53.83; 46.17)	2/32/0	79.2
MiniBooNE	129,590	49	(93,101; 36,489)	(71.84; 28.16)	49/0/0	67.7
rlcp	5,749,132	4	(5,728,201; 20,931)	(99.64; 0.36)	1/3/0	180.1
skin	245,057	3	(194,198; 50,858)	(79.25; 20.75)	0/4/0	3
susy	5,000,000	18	(2,712,173; 2,287,827)	(54.24; 45.76)	18/0/0	1740.8
SUSY_IR_16	2,881,684	18	(2,712,173; 169,511)	(94.12; 5.88)	18/0/0	1022.5

un enfoque muy eficiente para las tareas de aprendizaje y predicción respecto a otros paradigmas de clasificación.

Para confirmar la bondad de nuestra propuesta no sólo en el contexto de la reducción de datos, sino también en los resultados de clasificación de referencia, es necesario realizar una comparación justa. Por ello, se utilizarán dos enfoques como estado del arte: por un lado, un DT como modelo base y, por otro, el FCNN_MR como representante de las técnicas de reducción de datos disponibles para Big Data.

Hay que tener en cuenta que el uso del conjunto de datos en bruto para el entrenamiento del modelo puede no ser apropiado o comparable con respecto a nuestra técnica por dos razones principales. En primer lugar, la selección de características influye significativamente en la construcción y el aprendizaje del modelo DT. En segundo lugar, puede haber un claro sesgo hacia los conceptos mayoritarios para aquellos conjuntos de datos con una distribución no uniforme de los ejemplos de cada clases. Por lo tanto, el método de referencia consiste en un proceso de preprocesamiento simple (pero necesario) de dos etapas: en primer lugar, una etapa de FS y eliminación de duplicados siguiendo el mismo procedimiento que FDR^2-BD (como se describe en la Sección 7.2.1) y, en segundo

lugar, un submuestreo de instancias utilizando la técnica RUS para equilibrar los porcentajes de clase de una manera rápida y eficaz [Fer+17]. La elección se basa en que tanto FDR^2-BD como RUS realizan un muestreo uniforme en sus flujos de trabajo, y generan un conjunto de datos equilibrado (en RUS, al establecer una proporción entre clases de 1:1).

Las configuraciones de parámetros más relevantes utilizadas para cada método se muestran en la Tabla 7.3. Con respecto a los parámetros FDR^2-BD, se explican en detalle en la Sección 7.2.1. En este estudio, la lista `redPerc` se establece arbitrariamente con una granularidad del 1 % entre el 99 % y el 90 %, y una granularidad del 10 % para el resto, con el fin de obtener una amplia visión de la gran lista de conjuntos de datos. La versatilidad de esta herramienta propuesta permite a cualquier usuario interesado proporcionar una lista con una granularidad más fina, con el objetivo de analizar exhaustivamente el porcentaje de reducción de cada conjunto de datos. En este estudio experimental, FDR^2-BD se ejecuta estableciendo el PLT con un valor más estricto (1 %) y con uno más relajado (5 %). La varianza explicada acumulada indica el valor deseado que deben cubrir las características con los valores más altos de representatividad. La proporción final de 1:1 representa la misma cantidad de instancias para cada clase en el conjunto de datos resultante. Los parámetros de FCNN_MR son los recomendados por los autores (los lectores interesados pueden encontrar detalles adicionales en [Tri+15b]). Por último, el grado de paralelismo está representado por el número de particiones de datos que utiliza la sesión de Spark, establecido como `128` en esta experimentación.

Tabla 7.3: Algoritmos y parámetros utilizados en la experimentación de FDR^2-BD.

Algoritmo	Parámetro	Valor/es
FDR^2-BD	PLT	1 %, 5 %
	redPerc_list (%)	{99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 80, 70, 60, 50, 40, 30, 20, 10}
	criterio de agregación	mínimo
	internal *k*-fCV	5
FS	varianza cubierta	hasta 90 %
DT	impureza	entropia
	prof. máxima	5
RUS	proporción final	1:1
FCNN_MR	k	1
	ReducerType	join

Métricas de evaluación

Para la validación experimental de nuestra propuesta se usaron dos grupos de métricas. El primero se centra en el desempeño respecto a la condensación de datos y la capacidad de predicción. El segundo grupo, se enfoca en la evaluación de la escalabilidad.

En cuanto a la condensación de datos, se utilizan las siguientes medidas basadas en la dimensionalidad y en el número de instancias:

- Porcentaje de reducción de la dimensionalidad (`DimRed`): número de características seleccionadas respecto a la cantidad original en los datos de entrenamiento.

- Porcentaje de reducción de tamaño de los datos (`SizeRed`): número de ejemplos resultantes de la reducción horizontal con respecto al volumen de datos original.

Respecto a las métricas de evaluación del rendimiento predictivo, se utiliza la GM que intenta maximizar la precisión de cada una de las dos clases al mismo tiempo y es descripta en la Ecuación (3.2).

La evaluación de la escalabilidad de la herramienta tiene como objetivo comprender su comportamiento en un entorno paralelo, y se basa en los siguientes dos tipos de escalado: sizeup y speedup. El primero consiste en cambiar el tamaño de los datos manteniendo fijo el grado de paralelismo (número de particiones), y el segundo evalúa el comportamiento de la herramienta cuando cambia el grado de paralelismo, manteniendo constante el tamaño de los datos. El sizeup se define mediante la ecuación Ecuación (7.1), donde T_{full} es el tiempo de ejecución total obtenido al utilizar el conjunto de datos original completo, mientras que T_s es el tiempo de ejecución total al utilizar una versión del conjunto de datos original determinada por el tamaño s (s veces el conjunto de datos original). El speedup se define mediante la ecuación Ecuación (7.2), donde T_p es el tiempo de ejecución total logrado para un número de particiones establecido p, y T_1 es el tiempo cuando no se aplica el paralelismo.

En el contexto de esta tesis, el tiempo de ejecución total se considera como el período de tiempo desde que se lanza el trabajo de Spark hasta que se termina. Por lo tanto, incluye la sobrecarga típica generada por el entorno de Spark, la carga y distribución de los datos en los nodos de computación, el procesamiento del algoritmo en sí mismo, y el aprendizaje y la clasificación de los datos.

$$sizeup(s) = \frac{T_s}{T_{full}} \tag{7.1}$$

$$speedup(p) = \frac{T_1}{T_p} \tag{7.2}$$

7.3.2. Estudio de reducción del volumen de datos

El objetivo de cualquier método de reducción es encontrar si un conjunto de datos está bien representado por un subconjunto condensado del mismo. Por supuesto, existe una clara premisa de mantener el rendimiento predictivo original de cualquier modelo aprendido a partir de este conjunto reducido, o al menos con un impacto pequeño o admisible sobre el mismo.

A continuación se presentan los resultados obtenidos con nuestra herramienta para todos los grandes conjuntos de datos estudiados, centrándonos en el nivel de reducción horizontal alcanzado para dos umbrales de pérdida de calidad predictiva (PLT). Al inicio de este capítulo se menciona que los Big Datasets disponibles públicamente, por la forma en que se generan u obtienen, pueden tener redundancia, como se ve parcialmente en [MTH20]. El objetivo de este estudio experimental sobre varios Big Datasets es determinar que no sólo existe redundancia en los datos, sino que nuestra herramienta debería ser capaz de encontrar porcentajes de reducción significativos sin pérdida de capacidad predictiva.

Al aplicar nuestra propuesta sobre un conjunto de datos tabulares, uno de los resultados es el porcentaje de reducción recomendado RPR para el umbral de pérdida de predicción PLT seleccionado. El RPR puede ser "Ninguno" si la herramienta llega a la conclusión de que no se puede realizar una reducción uniforme dentro del rango de predicción dado por el PLT, o un valor de la lista `redPerc` (descripta en la Sección 7.3.1).

En la Tabla 7.4 se muestra el RPR proporcionado por la herramienta FDR^2-BD para los PLT elegidos para la experimentación, para cada conjunto de datos. Se puede observar que la gran mayoría de los conjuntos de datos (más del 80 %) consiguen una reducción significativa entre el 93 % y el 99 %, asegurando un comportamiento predictivo dentro del rango deseado. Por ejemplo, en algunos conjuntos de datos, el orden de magnitud de la reducción es de 6. Más concretamente, en el caso de `ECBDL14-10mill-90`, el conjunto de datos se reduce de 9,6 millones de instancias a 96 mil, el conjunto de datos `higgs` pasa de 8,8 millones a 109 mil, y el conjunto de datos `susy` se reduce de 4 millones de instancias a 40 mil.

Los resultados obtenidos responden claramente a la existencia de una alta redundancia conceptual en la mayoría de los conjuntos de datos. Para los conjuntos de datos `cencus` y `airlines`, no se obtienen porcentajes de reducción sugeridos. La calidad predictiva conseguida por el proceso de hiperparametrización no está dentro del rango aceptado considerando el umbral dado. En cuanto

al conjunto de datos `skin`, se puede reducir mediante esta metodología si el PLT se establece en un 5 %, obteniendo una reducción del 97 %. Cabe mencionar que el resultado de referencia de la GM para este conjunto de datos es de alrededor de 0.965, por lo que reducirlo un 97 % aceptando una pérdida del 5 % del rendimiento predictivo sigue siendo un buen compromiso. De forma similar, el conjunto de datos `fars_fatal` logra un RPR significativamente mayor al cambiar el PLT de 1 a 5 %.

Tabla 7.4: Porcentaje de reducción recomendado (RPR) por FDR^2-BD por cada umbral de pérdida de predicción (PLT).

	PLT [%]			PLT [%]	
	1	5		1	5
Dataset	**RPR [%]**		**Dataset**	**RPR [%]**	
Agrawal	99	99	fars_fatal	30	94
airlines	–	–	HEPMASS_IR_16	98	99
BNG_Australian	99	99	higgs	98	99
BNG_heart	99	99	HIGGS_IR_16	99	99
census	–	–	homeCredit	93	98
click_prediction	99	99	hyperplane	99	99
covtype1	97	99	klaverjas	99	99
covtype1_vs_2	96	98	MiniBooNE	95	99
covtype2	94	99	rlcp	99	99
creditCard	98	99	skin	–	97
ECBDL14-10mill-90	99	99	susy	99	99
ethylene_ECO_E_LH	97	99	SUSY_IR_16	98	99
ethylene_EM_E_LH	99	99			

7.3.3. La influencia de la selección de características en la reducción del volumen de datos

Para mostrar el comportamiento o caracterización de cada conjunto de datos al aplicar FDR^2-BD, la Figura 7.3 presenta las reducciones horizontales generales y desagregadas para el PLT igual al 1 %. No cabe duda de que la realización de la etapa FS como paso inicial de todo el proceso tiene, en muchos casos, una implicación significativa en cuanto a que el nivel de reducción se amplía considerablemente, en particular para aquellos conjuntos de datos con variables nominales. En este tipo de conjuntos de datos, una reducción de características hace mucho más probable la replicación de instancias, lo que significa que la información representada por el conjunto de datos es demasiado exhaustiva y que se puede extraer el mismo conocimiento con un subconjunto minoritario del mismo. Un ejemplo extremo de esto es el conjunto de datos predominante-

mente nominal `fars_fatal` que tiene la mayor reducción dada por el paso de eliminación de duplicados (casi 99 %). Al mismo tiempo, hay casos en los que la reducción se genera en ambas etapas, aunque la mayor reducción se produce en la etapa de reducción uniforme. A pesar de la gran condensación horizontal conseguida con las etapas 1 y 2, en la mayoría de los casos de estudio sigue siendo necesario aplicar el proceso de reducción de datos mediante un muestreo distribuido uniformemente sobre el espacio del problema.

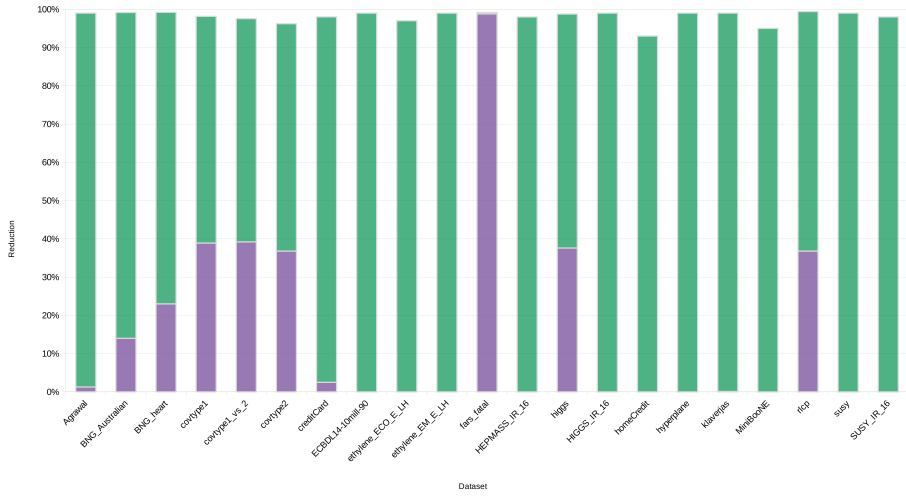


Figura 7.3: Reducción de volumen mediante la eliminación de instancias repetidas (violeta) y el muestreo uniforme (verde).

7.3.4. Detalles de la condensación de datos y evaluación del rendimiento

En la Tabla 7.5 se muestran las medidas de reducción de datos (`SizeRed` y `DimRed`) obtenidas por cada una de las técnicas comparativas y para FDR^2-BD, así como la capacidad de predicción de sus respectivos conjuntos de datos resultantes (medida a través de la GM). En estos resultados y en los siguientes, se denomina al método de referencia base como `BSL` (por *baseline*).

Se puede observar que FDR^2-BD supera con creces a los otros algoritmos en términos de condensación de datos para la mayoría de los conjuntos de datos, logrando un rendimiento predictivo similar al de referencia base. En términos generales, nuestra metodología aplica una reducción horizontal del 93 % al 99 %, como se ha destacado en la sección anterior.

Hay dos conjuntos de datos para los que nuestra herramienta obtiene resultados equivalentes a los otros dos métodos comparativos en términos de SizeRed. Se trata de los conjuntos de datos rlcp y creditCard, y las razones de este comportamiento son que rlcp es un conjunto de datos muy sencillo y de baja complejidad, por lo que no hay dificultad para que ninguno de los tres métodos alcance el mismo rendimiento. En cuanto a creditCard, es un conjunto de datos con una alta desproporción en sus clases, pero afortunadamente, los clusters de clases están bien separados y, por tanto, aplicando la técnica de referencia base también se consigue un buen modelado.

Los resultados muestran que nuestra herramienta consigue una notable diferencia en la reducción de datos: una reducción superior en más de un 45 % al resto de las técnicas experimentadas, manteniendo o incluso mejorando la calidad predictiva, por supuesto. Se destaca, además, el excelente rendimiento obtenido con la herramienta sobre los conjuntos de datos de Agrawal, higgs, hyperplane, Klaverjas y susy.

Tabla 7.5: Detalles de la condensación de datos y evaluación del rendimiento de BSL, FCNN_MR y FDR^2-BD.

	Agrawal			**BNG_Australian**			**BNG_heart**			**click_prediction**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.9475	0.9506	0.9476	0.8606	0.8332	0.8561	0.8541	0.8549	0.8534	0.6112	0.2254	0.6171
SizeRed	42	38	99	14	66	99	23	55	99	68	61	99
DimRed	67	0	67	71	0	71	62	0	62	64	0	64
	covtype1			**covtype1_vs_2**			**covtype2**			**creditCard**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.7534	0.7505	0.7585	0.7557	0.7487	0.7492	0.7411	0.7496	0.7351	0.9269	0.8662	0.9239
SizeRed	56	76	98	38	76	98	37	73	96	99	99	98
DimRed	87	0	89	85	0	91	83	0	85	67	0	73
	ECBDL14-10mill-90			**ethylene_ECO_E_LH**			**ethylene_EM_E_LH**			**HEPMASS_IR_16**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.6969	0.1514	0.6959	0.8814	0.605	0.8777	0.8692	0.7295	0.8693	0.8284	0.7347	0.8252
SizeRed	96	88	99	85	98	97	84	97	99	88	85	98
DimRed	79	0	82	50	0	50	44	0	44	86	0	86
	higgs			**HIGGS_IR_16**			**homeCredit**			**hyperplane**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.6509	0.6634	0.6488	0.6557	0.2771	0.6576	0.5953	0.1388	0.5954	0.3494	0.3432	0.5263
SizeRed	38	39	99	88	75	99	84	74	93	0	42	99
DimRed	82	0	82	79	0	79	90	0	91	60	0	60
	klaverjas			**MiniBooNE**			**rlcp**			**susy**		
	BSL	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**	**BSL**	**FCNN_MR**	**FDR^2-BD**
GM	0.8006	0.807	0.8072	0.8743	0.8458	0.8689	0.9309	0.9308	0.9306	0.7612	0.7542	0.762
SizeRed	0	41	99	49	72	95	99	99	99	0	52	99
DimRed	79	0	79	75	0	78	25	0	25	78	0	78

	SUSY_IR_16		
	BSL	**FCNN_MR**	**FDR^2-BD**
GM	0.7613	0.5604	0.7614
SizeRed	88	82	98
DimRed	78	0	78

Es importante señalar que tanto FDR^2-BD como el método BSL aplican una etapa de reducción de características basada en los valores de importancia de

las mismas. Sin embargo, los valores de DimRed podrían ser diferentes para el mismo conjunto de datos como resultado de que FDR^2-BD determina las características recomendadas a partir de la agregación de las características obtenidas de cada pliegue/*fold* de un conjunto de datos y el método BSL selecciona las características sólo en cada pliegue, por lo que podrían cambiar de un pliegue a otro. Por tanto, se demuestra que el uso de la agregación de características de todos los pliegues de los datos iguala o incluso mejora el porcentaje de reducción de características.

Con respecto al comportamiento del método de selección de instancias, nuestro método es superior a FCNN_MR en cuanto a la calidad predictiva en más del 66 % de los conjuntos de datos. Además, en aquellos casos en los que FCNN_MR consigue un GM similar a FDR^2-BD, el SizeRed oscila entre el 38 % hasta el 76 %, lo que supone un rango de valores muy inferior en comparación con los obtenidos por nuestra metodología. Además, debido a la naturaleza del método FCNN_MR (al igual que los métodos IS disponibles para Big Data), no se genera ninguna reducción vertical.

Para resumir los resultados, en la Tabla 7.6 se presentan los valores medios de las GMs y de las métricas de reducción, así como la comparación de nuestra metodología con respecto a las demás. Se muestran las diferencias entre las medidas (columna Diff) y el número de veces que nuestra herramienta supera, iguala o no supera a la herramienta comparada (columna W/T/L, que deriva de *Win/Tie/Loses* o Gana/Empata/Pierde).

Tabla 7.6: Evaluación del rendimiento y condensación de datos. Media, diferencias y recuento en donde FDR^2-BD supera, iguala o no supera (*W/T/L*) a la herramienta comparada.

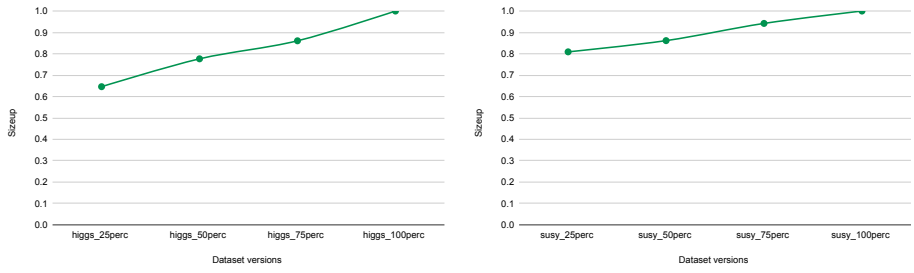
	GM			SizeRed			DimRed		
	Media	**Diff**	**W/T/L**	**Media**	**Diff**	**W/T/L**	**Media**	**Diff**	**W/T/L**
BSL	0.7669	0.0077	4/15/2	56	42	19/1/1	71	1	7/14/0
FCNN_MR	0.6438	0.1308	14/5/2	71	27	18/1/2	0	72	21/0/0
FDR^2-BD	0.7746	–	–	98	–	–	72	–	–

Se observa que nuestra propuesta es capaz de mantener la calidad predictiva incluso por encima del porcentaje estipulado por el parámetro PLT respecto a la línea base. Además, los valores de reducción obtenidos respecto a la horizontalidad y verticalidad aplicando FDR^2-BD son los mayores (98 % y 72 %, respectivamente), con diferencias significativas respecto a BSL y FCNN_MR.

7.3.5. Evaluación de la escalabilidad

En las Figuras 7.4 y 7.5, se muestran el sizeup y el speedup conseguidos para dos (`higgs` y `susy`) de los 25 conjuntos de datos experimentales, con el fin de mostrar el comportamiento general obtenido. Los conjuntos de datos seleccionados se eligen porque son ampliamente utilizados en artículos de clasificación de Big Data. Con el objetivo de evaluar el sizeup, hemos utilizado nuestra herramienta variando el tamaño de entrada del conjunto de datos original (`higgs_100perc`), generando tres versiones diferentes del mismo según su proporción del tamaño de los datos originales (`higgs_75perc, higgs_50perc, higgs_25perc`). Los resultados muestran que cuando un conjunto de datos aumenta de tamaño, el valor de scaleup crece, como es de esperar.

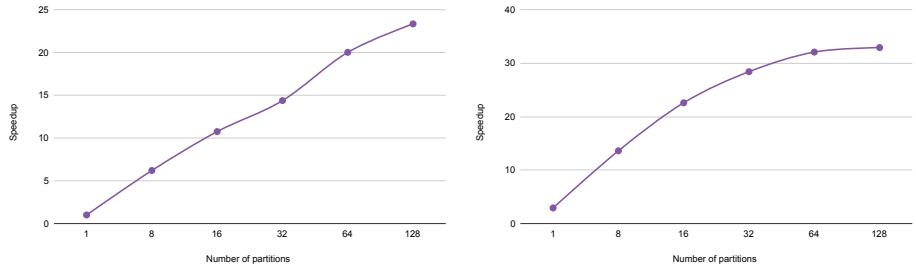
Con respecto a la evaluación del speedup, se utiliza el conjunto de datos original, y el grado de paralelismo base es `8`, duplicando el valor del grado de paralelismo anterior en cada ejecución. Es importante señalar que es difícil conseguir un aumento de velocidad lineal debido a que el aumento del número de particiones introducirá una sobrecarga de comunicación adicional. Sin embargo, se puede observar una tendencia lineal en dirección creciente a medida que se incrementa el número de particiones, lo que significa que este crecimiento realmente aumenta el grado de paralelismo de FDR^2-BD.



(a) Sizeup del conjunto Higgs.

(b) Sizeup del conjunto Susy.

Figura 7.4: Sizeup.



(a) Speedup del conjunto Higgs. **(b)** Speedup del conjunto Susy.

Figura 7.5: Speedup.

7.4. Comentarios del capítulo

En este capítulo se han comentado los beneficios de la condensación de datos mediante la reducción de conjuntos de datos tabulares en la clasificación, especialmente en el contexto de Big Data. Se ha visto que hay pocas soluciones para este tamaño de problemas de datos a pesar que ocurra lo contrario en contextos Small Data. Asimismo, muchas de las soluciones existentes no son completamente escalables o se enfocan sólo en la reducción horizontal de los datos.

Hemos presentado nuestra propuesta metodológica, llamada FDR^2-BD, que conforma una herramienta de recomendación de parámetros de condensación dual (porcentaje de reducción y características más importantes, si las hay). Se trata de un diseño sencillo y escalable, basado en un proceso de hiperparametrización totalmente transparente para el usuario, en el que se tienen en cuenta todos los datos siguiendo un procedimiento de k-fold. FDR^2-BD es capaz de analizar la reducción de datos de forma vertical y horizontal, recibiendo la visión del experto en datos mediante el establecimiento de dos simples condiciones: un umbral de calidad predictiva asociado al modelo obtenido a partir del conjunto condensado, y un rango de porcentajes de reducción a comprobar. Además, su implementación utiliza operaciones paralelas y utilidades totalmente optimizadas proporcionadas por Apache Spark.

Hemos evaluado ampliamente nuestra propuesta en los experimentos realizados en este capítulo. A lo largo de un extenso estudio experimental sobre 25 grandes conjuntos de datos con diferentes características, obtenidos de distintos repositorios de datos públicos conocidos, se ha observado la fortaleza de nuestra herramienta obteniendo valores de reducción elevados para la mayoría de los conjuntos de datos estudiados, tanto en lo que respecta a la dimensionalidad como a los porcentajes de reducción propuestos (alrededor del 70 % de reducción

de las características y 98 % de reducción de las instancias) para un umbral de pérdida predictiva del 1 %. En términos absolutos, hay conjuntos de datos que logran una reducción horizontal de cuatro, ocho e incluso nueve millones de instancias, manteniendo su capacidad de predicción para el problema original. Los máximos porcentajes de reducción posibles obtenidos superan a las técnicas estándar seleccionadas para la comparación. Dado que nuestra propuesta se basa en un proceso de hiperparametrización, este hecho confiere una mayor robustez a los valores de recomendación que se obtienen.

Otro aspecto que puede evaluarse en futuras investigaciones es probar la calidad de la propuesta en problemas multiclase, donde los límites entre conceptos pueden ser más difíciles, así como en problemas con desequilibrio. Para ello, se pueden incluir métricas de complejidad de datos para guiar el proceso en aquellas zonas del espacio donde se observe un mayor grado de redundancia.

Los resultados presentados en este capítulo forman parte del siguiente artículo:

Basgall, M. J.; Naiouf, M.; Fernández, A. *FDR^2-BD: A Fast Data Reduction Recommendation Tool for Tabular Big Data Classification Problems.* Electronics 2021, 10 (15), 1757. https://doi.org/10.3390/electronics10151757.

8 Caracterización del solapamiento en conjuntos Big Data

En este capítulo, el foco está puesto en el solapamiento de clases de los conjuntos de datos Big Data. La presencia de este tipo de complejidad de los datos genera un impacto negativo en el modelado de cualquier algoritmo clasificador. Esto se debe a que el aprendizaje se lleva a cabo a partir de zonas o áreas del espacio de datos que se consideran ambiguas por la presencia de instancias que pertenecen a distintas clases. Así pues, un clasificador presenta dificultades para modelar dichas áreas y esto repercute en la predicción de las etiquetas de clase de nuevas instancias. Por consiguiente, es útil y ventajoso poder contar con información del solapamiento presente en un conjunto de datos y que dicha caracterización sea llevada a cabo de forma escalable para la eficiente aplicación sobre conjuntos Big Data. A partir de esto, se habilita el poder tratar los datos a priori para facilitar la tarea de aprendizaje; y/o evitar el sesgo hacia alguno de los conceptos de manera directa por el clasificador.
En la Sección 8.1 se presenta un breve estado del arte acerca de la caracterización de conjuntos de datos y se introduce a la tarea de particionamiento del espacio de características, puntualizando en el enfoque de rejilla (grid). A continuación, en la Sección 8.2, se detalla nuestra propuesta metodológica para la caracterización escalable del solapamiento presente en grandes conjuntos de datos basada en un enfoque grid, la cual denominamos GridOverlap-BD. Luego, en la Sección 8.3 se desarrolla nuestro estudio experimental, donde se analiza la efectividad de la caracterización llevaba a cabo en los conjuntos de datos de la experimentación. A su vez, se propone una medida de complejidad que deriva de GridOverlap-BD, y se muestra además una breve comparación entre distintos enfoques para el tratamiento de la ambigüedad presente en los datos. Por último, en la Sección 8.4, se comenta brevemente el capítulo.

8.1. Caracterización de un conjunto de datos. Particionamiento del espacio de características

Al final de la Sección 2.2 se describen las medidas más populares que pueden extraerse de los conjuntos de datos de entrenamiento para caracterizar la complejidad de los respectivos problemas de clasificación que ellos representan. De los trabajos que se pueden encontrar en la bibliografía, en [SMS07] se centran

en el uso de algunas medidas de complejidad de los datos para describir el solapamiento de las clases, la dimensionalidad del espacio de características y la densidad de las clases, y analizan su relación con la precisión del clasificador *k*NN. Por su parte, en [KO09] se propone una estrategia que consiste en utilizar un esquema de reducción de prototipos para calcular índices que cuantifican el volumen de superposición en los datos de forma aproximada, pero rápida. A su vez, los trabajos [Skr11] y [OSL17] llevan a cabo el análisis de un grupo de métricas de complejidad para conjuntos de datos con y sin atributos irrelevantes. Por otra parte, en [GCL15] se propone una técnica de filtrado de ruido basada en las dos métricas de complejidad que más fuertemente se relacionan con el aumento en los niveles de ruido en los datos. A su vez, en [Lor+19] se presenta una extensa revisión de las métricas para caracterizar un conjunto de datos, además de revisar y discutir su uso en la literatura reciente, lo que permite encontrar oportunidades para futuros trabajos en el área. Como era de esperar, dicho trabajo incluye las métricas de complejidades detalladas en la Sección 2.2 e incorporan otras métricas un poco menos populares pero que pueden ser también de utilidad. Los autores agrupan, de manera general, las medidas para problemas de clasificación en las siguientes categorías: medidas basadas en características, medidas de vecindad, medidas de linealidad, medidas de dimensionalidad, medidas de equilibrio de clases y medidas de red. Respecto a las medidas de complejidades para datos no balanceados, en [SGS20] se presentan dos métricas basadas en el enfoque de los vecinos más cercanos ponderados, mientras que en [Bar+21] investigan la eficacia de las medidas de complejidad en conjuntos de datos reales no balanceados y cómo se ven afectados al aplicar diferentes tratamientos para el desequilibrio de los datos.
En relación a la caracterización de un problema Big Data existen muy pocos aportes, hecho que podría plantearse desde una doble perspectiva: (1) es muy complejo escalar las técnicas del estado del arte porque se basan fundamentalmente en clustering/grafos (alto costo computacional); (2) hay que considerar las características específicas de los problemas Big Data en sí, y quizá crear nuevas medidas concretas para evaluar la dificultad que entrañan. En [MTH20] se presentan dos nuevas métricas sobre redundancia y complejidad en los conjuntos de datos Big Data. Las mismas están disponibles en un paquete software totalmente escalable de acceso público y gratuito para ser ejecutado en el framework Apache Spark[32]. Los autores incorporan además las correspondientes adaptaciones paralelas de las métricas de complejidades de los datos tradicionales, que se describieron en la Sección 2.2.

Como se puede apreciar, existen múltiples aspectos que pueden ser evaluados

32 https://spark-packages.org/package/JMailloH/ComplexityMetrics

en un conjunto de datos como también muchas alternativas para tomar acciones a partir del conocimiento que las métricas ofrecen.

En este sentido, el particionamiento del espacio de características es una tarea que se lleva a cabo desde hace tiempo y la cual, puntualmente para el caso de la clasificación de datos, es ampliamente utilizada para definir la separación entre las clases de un problema. Existen diversos enfoques para llevar a cabo esta tarea, por ejemplo, creando una estructura de árbol, determinando grupos dentro de la distribución de los datos, o dividiendo el espacio de datos en hipercubos o celdas (particionamiento basado en rejilla). La capacidad de particionar el espacio de datos de un problema habilita a conocer en mayor detalle sus características intrínsecas, de lo que surgen distintas métricas de complejidad de los datos que pueden estar relacionadas a la pureza, a la separabilidad, a la incerteza, a la densidad, entre otras medidas de utilidad [Sin03a].

En particular, este capítulo se centra en el particionamiento del espacio de características basado en rejilla . Este tipo de enfoque se conoce también como *information slicing* o *multi-resolution*. La idea detrás consiste en descomponer un conjunto de datos en subconjuntos mediante la división del rango de cada atributo en tantas partes como se establezca, guiado por una resolución o granularidad. En la Figura 8.1 se muestra de manera esquemática el particionamiento del espacio de características basado en rejilla para un conjunto de datos de dos variables, donde los signos + y - representan instancias en dichas zonas del espacio que pertenecen a las distintas clases del problema. Además se muestran las rejillas resultantes de aplicar el particionamiento a distintas granularidades (L) tomando el valor 3, 5 y 7 en cada caso.

De este modo se tiene el problema dividido en celdas o hipercubos a partir de lo cual se realiza un análisis o se aplica una tarea deseada. Los autores de [Sin03a; Sin03b] presentan métricas que cuantifican ciertos conceptos de interés los cuales son analizadas a través de las distintas granularidades que se aplican al conjunto de datos. Este tipo de enfoque también se aplica a la segmentación de imágenes [SS04].

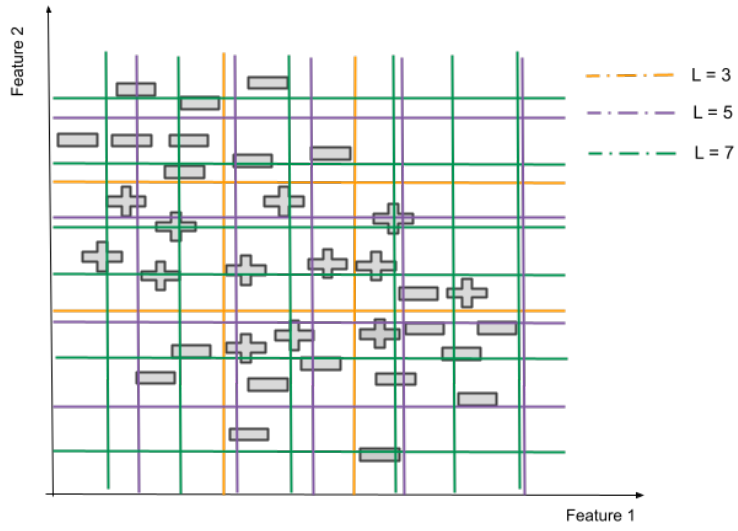


Figura 8.1: Esquema del particionamiento del espacio de características basado en rejilla para múltiples granularidades.

8.2. GridOverlap-BD, hacia la caracterización escalable del solapamiento en un conjunto Big Data

Con el fin de comprender el grado de solapamiento de un problema de clasificación de Big Data, en primer lugar se propone el diseño de una estrategia para el particionamiento del espacio de características basado en celdas, y su correspondiente implementación, denominada GridOverlap-BD, aprovechando la computación distribuida. En segundo lugar, se presenta una métrica de complejidad para cuantificar el solapamiento en un conjunto Big Data, la cual deriva de aplicar GridOverlap-BD. La métrica se muestra más adelante, junto al estudio experimental.

Como su nombre lo indica, GridOverlap-BD está compuesto por un mecanismo basado en rejilla, que sigue parcialmente el esquema Chi-PG [Elk+18]. Por lo tanto, está diseñado para determinar clusters de datos por aproximación en un entorno Big Data de forma eficiente. En este sentido, el objetivo de GridOverlap-BD es cuantificar los clusters locales en cuanto a su distribución por etiqueta de clase y el grado de solapamiento.

Es importante destacar que GridOverlap-BD podría aplicarse como etapa previa a cualquier técnica de preprocesamiento, o incluso a los clasificadores,

para conocer de antemano la caracterización respecto al solapamiento de un conjunto Big Data.

La idea detrás de GridOverlap-BD es trabajar con reglas descriptivas, que permitan ayudar a identificar cualquier conjunto de datos de forma escalable.

Los objetivos de esta propuesta son los siguientes:

(a) Determinar una medida cuantitativa sobre el solapamiento presente en los datos mediante la identificación de celdas seguras o celdas ambiguas (los tipos de celdas se describen más adelante).

(b) Brindar los subconjuntos de datos caracterizados para su posterior análisis y/o preprocesamiento, o para recabar más información para dichas áreas dependiendo si el tipo de problema lo permite.

Como aporte adicional, y haciendo uso de la herramienta propuesta, se muestra la aplicación de algunas de las estrategias descriptas en la Sección 4.3 para manejar el solapamiento de los datos.

8.2.1. Descripción y flujo de trabajo

En la Figura 8.2 se presenta el diagrama del flujo de trabajo de GridOverlap-BD.

El diseño de esta estrategia está conformado por cuatro etapas que se describen a continuación:

Selección de características (*Feature Selection*). En los conjuntos Big Data es común encontrar que sólo un subconjunto de los atributos son representativos o necesarios para la clasificación, por lo que pueden omitirse y así favorecer el trabajo con rejillas de menor cantidad de celdas que si se tuvieran en cuenta todas las características.

Discretización (*Discretization*). Para obtener la Base de Reglas Descriptivas (*Descriptive Rule Base*) se considera, en primer lugar, una partición en frecuencia del rango de cada variable, haciendo que el espacio de entrada se convierta en una rejilla compuesta por celdas, debidamente etiquetadas (de manera secuencial). A diferencia de un particionamiento uniforme del rango de cada variable, el particionamiento en frecuencia permite guiar la división según la densidad de la distribución de los datos para cada atributo. El algoritmo GridOverlap-BD tiene la capacidad de trabajar con diferentes grados de granularidad, es decir, el número con el que se particiona cada variable del problema. Cuando la granularidad es pequeña, la rejilla contiene menos celdas y, por tanto, las áreas están más

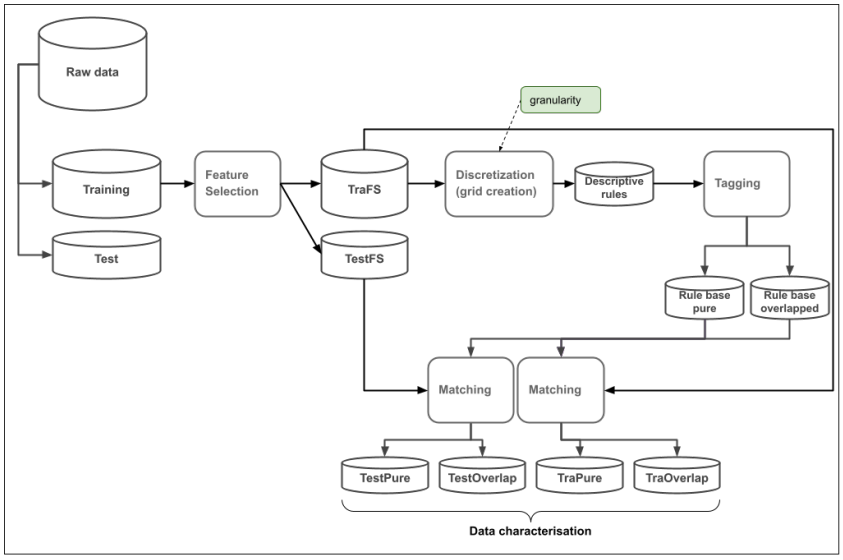


Figura 8.2: Metodología de GridOverlap-BD.

pobladas. Así, el número de granularidad permite ser más o menos específico en los subconjuntos a generar.
A continuación, se toma el conjunto completo de instancias de entrenamiento y se asigna cada una de ellas a una regla conjuntiva. El antecedente de cada regla estará compuesto por la conjunción de cada etiqueta de celda en la que se establecen los valores de los diferentes atributos, mientras que el consecuente será simplemente la etiqueta de clase de la instancia relacionada. A continuación se muestra un ejemplo:

```
IF f1=L2 AND f2=L1 AND f3=L1 AND f4=L3 THEN class=C0
```

Una vez construidas todas las reglas, se procede a agruparlas por antecedente y a contar su soporte para cada valor del consecuente, medido directamente como el recuento absoluto de cada regla encontrada durante esta etapa de mapeo de las instancias de entrenamiento a reglas.

En esta propuesta, nos referimos indistintamente a los términos «antecedente de una regla descriptiva» como a una «regla» o «área» .

Etiquetado (*Tagging*). Es importante destacar que puede haber reglas con el mismo antecedente (mismo área espacial) pero con diferente consecuente, y/o reglas con el mismo antecedente con un único consecuente. En este trabajo, el primer tipo de reglas se denomina reglas superpuestas (*Overlapping Rules* (OvRs)), mientras que las otras son las reglas puras (*Pure Rules* (PRs)).

Así pues, cualquier regla que represente una única celda puede presentar uno de estos dos casos diferentes según su composición:

- Celda pura (cp): Compuesta por PRs, es decir, presencia de una sola clase; o compuesta por OvRs con proporción de 1:10 o más entre sus clases, siendo tal la desproporción hacia una de las clases que en dicha área un clasificador generaría un sesgo hacia ella[33]. A este tipo de celdas las denominamos también como «celdas o áreas seguras».

- Celda solapada (co): Compuesta por el resto de las OvRs, es decir, reglas de doble consecuente que tengan una proporción entre las clases menor a 1:10. A este tipo de celdas las denominamos también como «celdas o áreas ambiguas».

Para ilustrar los diferentes casos de estudio enumerados anteriormente, la Figura 8.3 representa, de manera esquemática, una rejilla de dos variables para un problema binario con un grado de granularidad de tres particiones por variable.

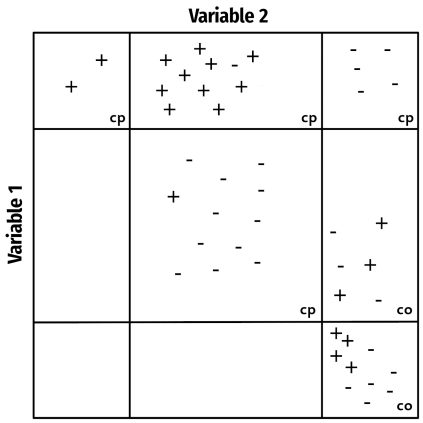


Figura 8.3: Rejilla de dos variables para un problema binario con los dos tipos de celdas, donde el símbolo "+" representa un consecuente positivo y el símbolo "-" uno negativo.

33 El valor 10 se elige arbitrariamente basado en la caracterización estándar de un problema altamente desbalanceado.

A partir de este paso de etiquetado, se tiene entonces dos bases de reglas, o una única base de reglas con dos tipologías o casuísticas, o subconjuntos que representan las áreas puras y las áreas ambiguas de un problema.

Correspondencia/mapeo (*Matching*). Por último, para cumplir con el objetivo de identificar los diferentes tipos de áreas de interés del problema, se procede a encontrar las correspondencias entre las instancias del conjunto de datos con las bases de reglas. Esta tarea se puede aplicar tanto a los datos de entrenamiento como a los de prueba, dado que estos últimos siguen la misma distribución que los primeros. A partir de ello se podrá contar con la caracterización de cada instancia que indica si pertenece a un área pura o ambigua dentro del espacio del problema.
Es de interés mencionar que la metodología GridOverlap-BD tiene relación con la granularidad elegida para la etapa de la discretización de los datos y dependerá del conjunto de datos si la correspondencia se mantiene igual a través de aplicar distintas granularidades. Para esquematizar la situación en la que la correspondencia no se mantiene igual, en la Figura 8.4 se toma el caso que se presenta en la sección anterior y se resalta en color rojo una instancia que pertenece a un área pura (para L = 3), a una ambigua (para L = 5) y a una pura (para L = 7). De este modo, se aprecia la versatilidad de la metodología propuesta, es decir, se permite al usuario llevar a cabo un proceso exhaustivo de caracterización mediante GridOverlap-BD en base a diferentes valores de granularidad. El resultado será la obtención de un número diferente de reglas (áreas) que identifiquen las instancias difíciles del problema, de modo que se ajuste al tipo de modelo que se desee aplicar para la extracción de conocimiento.

8.2.2. Comentarios de la implementación técnica

La estrategia GridOverlap-BD ha sido desarrollada utilizando el lenguaje programación Scala bajo el framework Apache Spark (v3.0.1) aplicando un diseño de enfoque global. En relación a las estructuras de datos, se seleccionaron los DataFrames y los Datasets, dado su alto desempeño.

Para determinar la importancia de los atributos para la etapa de selección de características, se utiliza el RF que su implementación es completamente escalable y nativo de Spark, dado que es provisto por la librería MLlib. La discretización de los datos se lleva a cabo mediante la utilidad llamada `QuantileDiscretizer` que está incluida también en la biblioteca MLlib. De manera general, para la implementación de GridOverlap-BD se utilizan funciones, algoritmos y utilidades completamente escalables y nativas de Spark con el fin de proporcionar una

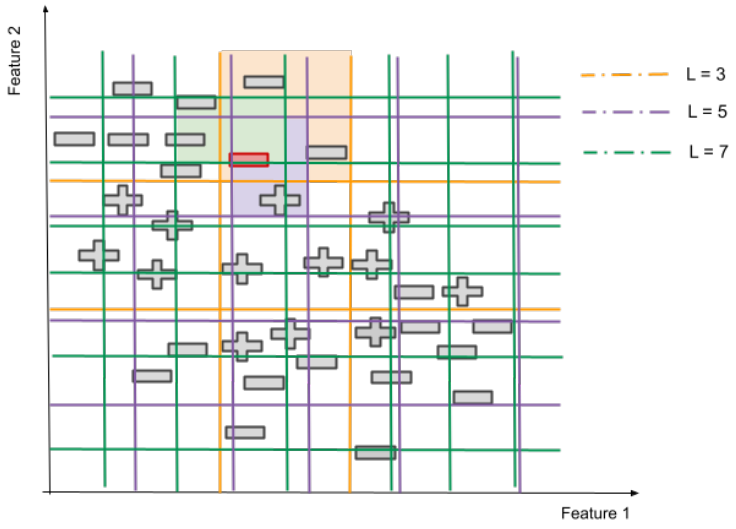


Figura 8.4: Influencia de la granularidad en el particionamiento del espacio de características basado en rejilla.

solución que aproveche al máximo los beneficios de este framework[34]. Su correspondiente código fuente se puede encontrar en https://github.com/majobasgall/big_data_overlapping_characterization.

8.3. Estudio experimental

En esta sección se lleva a cabo el estudio experimental con el fin de mostrar la calidad de nuestra propuesta. Para eso, en la Sección 8.3.1, se presentan los conjuntos de datos Big Data y los parámetros utilizados en la experimentación, además se describe la forma en la que se realizarán las comparaciones de los resultados. Luego, en la Sección 8.3.2 se evalúa el desempeño de la caracterización llevaba a cabo por GridOverlap-BD. A continuación, en la Sección 8.3.3, se presenta y aplica sobre los conjuntos experimentales una métrica propuesta para cuantificar el grado de solapamiento en los datos. Posteriormente, una breve comparación entre distintos enfoques para el tratamiento de la ambigüedad presente en los datos y un método de base, se muestra en la Sección 8.3.4.

34 En la Apéndice A.4 se encuentra un listado de todas las primitivas y funcionalidades utilizadas para la implementación de los aportes de esta tesis.

8.3.1. Entorno experimental

Conjuntos de datos e infraestructura Para aplicar GridOverlap-BD se utilizaron 9 conjuntos Big Data, obtenidos a partir de los más populares repositorios de datos[35]. De manera resumida se muestran las principales características de los datos elegidos en la Tabla 8.1, informando el número de instancias (#Ex.), número de atributos (#Atts.), el número de instancias de cada clase (#(class0; class1)), el porcentaje de distribución de las clases (%(class0; class1)), el número de cada tipo de característica (Co/Di/Ca, donde Co = continuas, Di = discretas, Ca = categóricas), y el tamaño del conjunto de datos (en MB).

Tabla 8.1: Resumen de los conjuntos Big Data utilizados en la experimentación de GridOverlap-BD.

Dataset	#Ex.	#Atts.	#(class0; class1)	%(class0; class1)	Co/Di/Ca	Size (MB)
airlines	539,383	7	(299,119; 240,264)	(55.46; 44.54)	4/0/3	18.3
BNG_Australian	1,000,000	14	(573,051; 426,949)	(57.31; 42.69)	14/0/0	80.8
BNG_heart	1,000,000	13	(555,946; 444,054)	(55.59; 44.41)	6/0/7	65.7
covtype1_vs_2	495,173	54	(283,468; 211,705)	(57.25; 42.75)	10/0/44	61.1
covtype2	581,012	54	(297,544; 283,468)	(51.21; 48.79)	10/0/44	71.7
higgs	11,000,000	28	(5,827,686; 5,172,314)	(52.98; 47.02)	28/0/0	7680
hyperplane	1,000,000	10	(500,007; 499,993)	(50; 50)	10/0/0	91.4
klaverjas	981,541	34	(528,339; 453,202)	(53.83; 46.17)	2/32/0	79.2
susy	5,000,000	18	(2,712,173; 2,287,827)	(54.24; 45.76)	18/0/0	1740.8

Parámetros más relevantes Los valores para la granularidad (representada con la letra L) a aplicar en el proceso de discretización de los datos elegidos para esta experimentación se establecen arbitrariamente, abarcando tanto valores pequeños como más grandes, y son los siguientes: $L = 7, 11, 13, 17, 19, 23, 29$.

Comparaciones Para evaluar tanto la efectividad de la caracterización que realiza nuestra propuesta como los enfoques elegidos del estado del arte para el tratamiento de las áreas solapadas, se utiliza un DT.

Como medida de evaluación de la calidad predictiva, se utiliza el porcentaje de acierto (ACC) definida en la Sección 3.3, dada la naturaleza equilibrada de las clases en los conjuntos de datos de esta experimentación.

El flujo de trabajo para la obtención de la ACC de base (*baseline*, BSL) se muestra en la Figura 8.5, donde se sigue un esquema de entrenamiento-clasificación a partir de los datos de entrenamiento originales y la predicción sobre los datos de pruebas también originales.

35 En el Apéndice A.1 se describe cada conjunto de datos junto a su repositorio de origen.

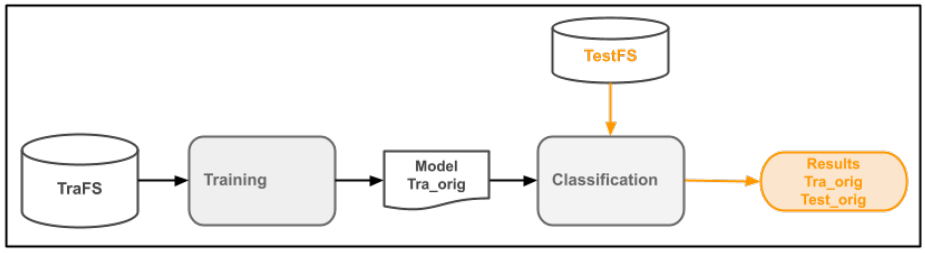


Figura 8.5: Flujo de trabajo para la obtención de los resultados de base (BSL).

8.3.2. Efectividad en la distinción de áreas puras y ambiguas

Para evaluar el desempeño de GridOverlap-BD respecto a la caracterización del solapamiento de un problema se sigue el flujo de trabajo que se muestra en la Figura 8.6. Es decir, se obtiene la exactitud de predecir, de manera independiente, sobre los subconjuntos de prueba caracterizados por GridOverlap-BD como puros y solapados (`TestPure` y `TestOverlap`, respectivamente), utilizando el modelo entrenado a partir de los datos originales (`Model Tra_orig`).

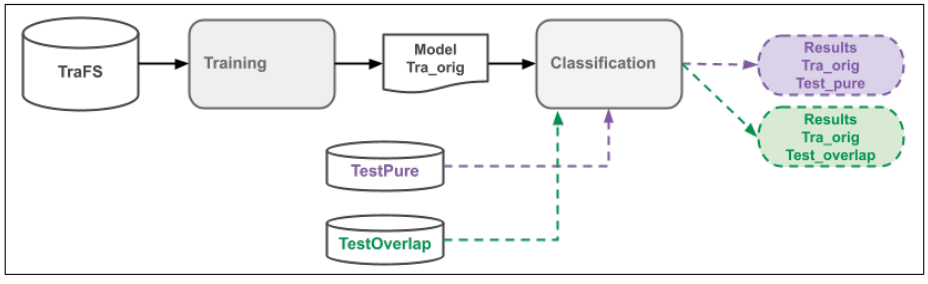


Figura 8.6: Flujo de trabajo para la obtención de los resultados de la efectividad de la caracterización.

Los resultados obtenidos se muestran en la Tabla 8.2, la cual contiene la proporción de las instancias de prueba caracterizadas como puras (`Tst_pure proportion`), la proporción de las instancias de prueba caracterizadas como solapadas (`Tst_over proportion`), la precisión de la clasificación del subconjunto puro (`ACC_pure`), y la precisión de la clasificación del subconjunto solapado (`ACC_over`). A su vez, y dado que en nuestra experimentación los resultados se obtienen a partir de aplicar GridOverlap-BD utilizando distintas granularidades, para cada una de esas columnas se muestran los valores promedios, mínimos y máximos (`AVG`, `MIN`, `MAX`, respectivamente). Esto da idea al lector sobre la anteriormente mencionada influencia de la granularidad y de los

propios datos. Sin embargo, el usuario de nuestra propuesta podrá llevar a cabo, según lo desee, la búsqueda de la granularidad que maximice la precisión o la proporción del área de interés, tal como se indicó al presentar esta herramienta GridOverlap-BD.

Tabla 8.2: Efectividad en la distinción de zonas puras y ambiguas llevaba a cabo por GridOverlap-BD.

	Tst_pure proportion			Tst_over proportion				ACC_pure			ACC_over		
dataset	**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**	**BSL**	**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
airlines	1.18 %	0.07 %	2.35 %	98.82 %	97.65 %	100.00 %	0.5640	0.7125	0.6923	0.7384	0.5628	0.5608	0.5640
BNG_Australian	51.70 %	44.68 %	58.72 %	48.30 %	41.28 %	55.32 %	0.8521	0.9349	0.9326	0.9362	0.7506	0.7357	0.7842
BNG_heart	55.42 %	54.48 %	56.36 %	44.58 %	43.64 %	45.52 %	0.8640	0.9648	0.9634	0.9666	0.7390	0.7349	0.7443
covtype1_vs_2	16.47 %	10.61 %	22.33 %	83.53 %	77.67 %	89.39 %	0.7727	0.9403	0.9366	0.9434	0.7345	0.7245	0.7529
covtype2	26.74 %	16.50 %	36.98 %	73.26 %	63.02 %	83.50 %	0.7410	0.8693	0.8086	0.9198	0.6908	0.6782	0.7057
higgs	19.14 %	7.24 %	31.05 %	80.86 %	68.95 %	92.76 %	0.6523	0.8338	0.7433	0.9196	0.6270	0.6252	0.6315
hyperplane	19.53 %	0.84 %	38.22 %	80.47 %	61.78 %	99.16 %	0.3499	0.2162	0.0900	0.3408	0.3563	0.3521	0.3588
klaverjas	49.31 %	46.13 %	52.49 %	50.69 %	47.51 %	53.87 %	0.8049	0.9356	0.9256	0.9450	0.6809	0.6506	0.7036
susy	15.12 %	13.04 %	17.20 %	84.88 %	82.80 %	86.96 %	0.7696	0.9480	0.9206	0.9633	0.7390	0.7304	0.7470

De los resultados se observa que se identifican adecuadamente las áreas seguras del conjunto de prueba. En efecto, para la mayoría de los conjuntos de datos sus instancias se clasifican bien entre un 83 a un 96 % de las veces, valor que se podría considerar por defecto como un acierto alto. Además, se advierte que en tres de ellos la cantidad de instancias en áreas puras representa en promedio a un 50 a 55 % del total de las instancias, lo que muestra un buen desempeño en la estrategia de caracterización. En relación a las áreas ambiguas del problema, la precisión en su clasificación disminuye considerablemente respecto a las de áreas puras. Este comportamiento es el esperado precisamente por tratarse de instancias pertenecientes a áreas conflictivas del espacio de los datos, con la ventaja de haber podido identificarse a priori de manera agnóstica al clasificador a utilizar.

8.3.3. Grado de solapamiento

Con el fin de cuantificar el solapamiento existente en un conjunto Big Data, proponemos una métrica que se basa en la información provista por la estrategia GridOverlap-BD. A la misma la denominamos «grado de solapamiento grid» (*Grid Overlapping Degree* (GOD)) y se obtiene a partir del valor medio del porcentaje de instancias del conjunto de entrenamiento que pertenecen a áreas ambiguas del problema (para todas las granularidades elegidas). El dominio de GOD es $[0, 1]$ y es directamente proporcional a la complejidad del problema, es decir que un valor alto en el grado de solapamiento significa una alta complejidad.

En la Tabla 8.3 se muestra, ordenado de manera descendente, el valor de GOD para cada conjunto de datos.

Tabla 8.3: Grado de solapamiento grid (GOD).

dataset	GOD
airlines	0.99
hyperplane	0.89
susy	0.81
higgs	0.79
covtype1_vs_2	0.76
covtype2	0.65
klaverjas	0.47
BNG_heart	0.45
BNG_Australian	0.38

Se puede observar que el conjunto `airlines` presenta la mayor complejidad, con un grado de ambigüedad casi total, lo que explica su bajo rendimiento para la identificación de las áreas puras vista anteriormente. De manera similar, ocurre con `hyperplane` del cual se muestra también un alto valor para GOD y, acompañado de que su ACC de base es notablemente baja, es un caso concreto de datos de cuestionable calidad y del que se debería profundizar su estudio. Por otro lado, en `BNG_Australian` y `BNG_heart`, su bajo valor de la métrica GOD refleja su baja complejidad y, por ende, su buen desempeño predictivo de base.

En la Figura 8.7 se muestra cómo la medida de complejidad propuesta se relaciona con la ACC_bsl del problema, dando que para bajos valores de complejidad se obtienen altos valores de ACC_bsl siendo éste el comportamiento esperado.

8.3.4. Comparando el desempeño del método de base contra los enfoques para tratar el solapamiento

Dado que contamos con la herramienta GridOverlap-BD para caracterizar las instancias en solapadas y no solapadas, y a partir de ello con los respectivos subconjuntos de datos, esta sección se centra en los enfoques `descartar` y `separar` del estado del arte (introducidos en Sección 4.3)[36]. El objetivo es comparar la calidad predictiva alcanzada por el uso de un DT como clasificador de base (BSL) con la que se obtiene de aplicar los enfoques antes mencionados sobre los conjuntos Big Data de esta experimentación.

En la Figura 8.8 se muestra el flujo de trabajo del enfoque `descartar`. Los datos de entrenamiento originales que han sido caracterizados como solapados no se tienen en cuenta para el modelado (como lo indica el nombre del enfoque, se descartan). En consecuencia, el clasificador aprende de los datos que fueron caracterizados como puros, y las predicciones de clase se realizan sobre el conjunto completo original de pruebas.

36 Por cuestiones de extensión se plantea para un futuro incorporar el enfoque `fusionar`.

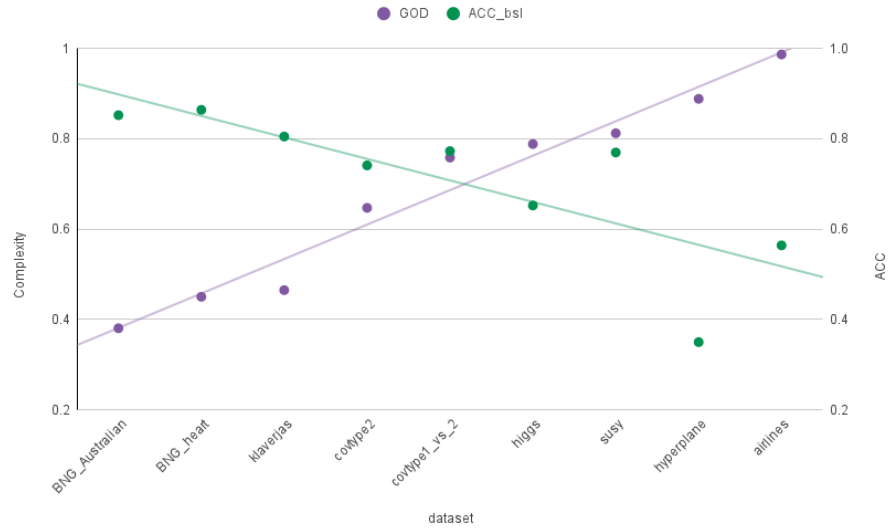


Figura 8.7: Grado de solapamiento vs precisión de base.

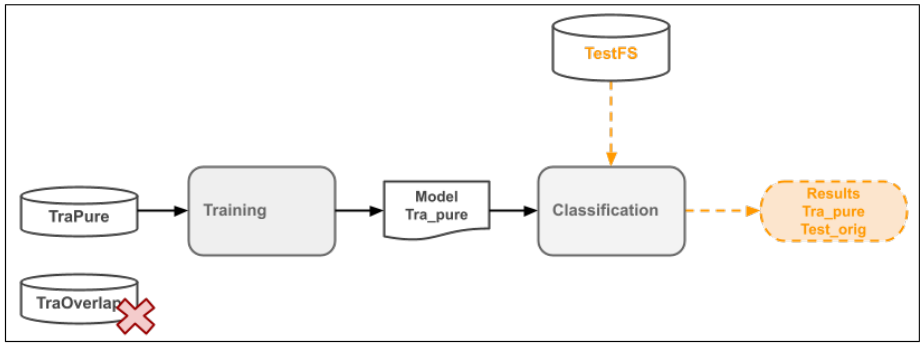


Figura 8.8: Flujo de trabajo para la obtención de los resultados siguiendo el enfoque *descartar*.

Para el enfoque `separar`, cuyo flujo de trabajo se muestra en la Figura 8.9, se obtienen dos grupos de resultados, dado que el entrenamiento se da a partir de los datos originales ya caracterizados como puros y como solapados independientemente. Y, siguiendo la misma lógica de separar, las predicciones se realizan sobre los datos de pruebas puros o los solapados, respectivamente.

Para poder ser comparados con el rendimiento predictivo de base (`ACC_bsl`), las predicciones individuales para cada tipo de área se relacionan como se muestra en la Ecuación (8.1), en donde ACC_sep_{pure} es la exactitud que se obtiene de

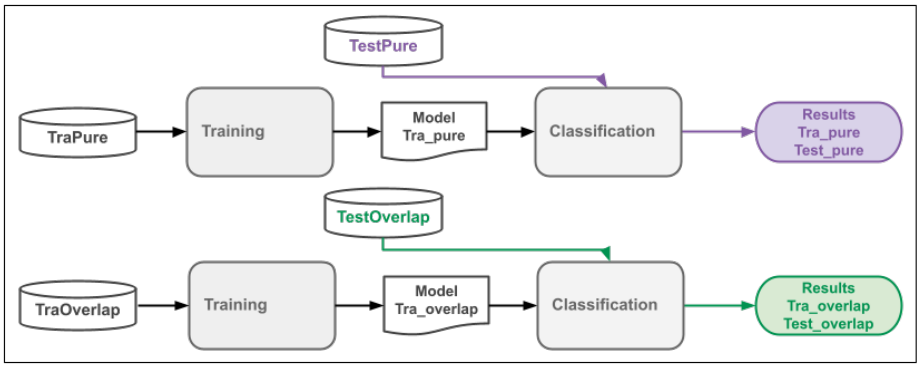


Figura 8.9: Flujo de trabajo para la obtención de los resultados siguiendo el enfoque *separar*.

entrenar un DT a partir de los datos de entrenamiento que mapean con las áreas puras del problema y clasificar sobre los de prueba que mapeen con dichas áreas. De manera similar, se obtiene la ACC_sep_{over} pero teniendo en cuenta los datos de las áreas ambiguas solamente. Por su parte, $\#tst\_pure$ y $\#tst\_over$ representan el número de instancias del conjunto de prueba que tienen correspondencia con las áreas puras y solapadas, respectivamente.

$$ACC\_sep = \frac{ACC\_sep_{pure} * \#tst\_pure + ACC\_sep_{over} * \#tst\_over}{\#tst\_pure + \#tst\_over} \tag{8.1}$$

En relación a los resultados, para el enfoque `descartar`, los mismos se omiten deliberadamente dado que, en ningún caso, superan a `ACC_bsl`. Este comportamiento se podría entender por estar experimentando en un escenario Big Data en el cual, proceder a no incluir una porción de los datos para el modelado, ocasiona un impacto negativo dado que la misma es de gran tamaño (como se ha podido ver al inicio de este estudio experimental).

Respecto al enfoque `separar`, en la Tabla 8.4 se muestran los resultados obtenidos para todos los conjuntos de datos. Puesto que GridOverlap-BD ha sido probado con diversas granularidades, se muestran los resultados a partir de aquellas que alcanzaron el mejor desempeño para cada problema, razón por la cual se tienen las columnas `ACC_sep` y `L`. Conocer la granularidad que deriva en el mejor desempeño para el enfoque `separar`, puede ser de interés para guiar a la elección de otras nuevas granularidades con la que experimentar en cada conjunto de datos. Por su parte, la columna `diff` muestra la diferencia entre `ACC_sep` y `ACC_bsl`.

Tabla 8.4: Predicciones de base y siguiendo el enfoque «separar».

	BSL	Separar		
dataset	ACC_bsl	ACC_sep	L	diff
airlines	0.5640	0.5642	13	0.02 %
BNG_Australian	0.8521	0.8521	7	0.00 %
BNG_heart	0.8640	0.8621	23	-0.19 %
covtype1_vs_2	0.7727	0.7755	17	0.28 %
covtype2	0.7410	0.7636	29	2.25 %
higgs	0.6523	0.6616	19	0.93 %
hyperplane	0.3499	0.3512	19	0.13 %
klaverjas	0.8049	0.8117	19	0.68 %
susy	0.7696	0.7702	29	0.06 %

En la mayoría de los casos se observa una leve mejora hacia la utilización de separar los datos y tratarlos como dos conjuntos independientes. Esto además ofrece otras ventajas tales como contar con dos modelos especializados para cada tipo de área, a partir de lo cual se puede llevar a cabo el ajuste de sus parámetros para alcanzar un mejor rendimiento predictivo.

8.4. Comentarios del capítulo

En este capítulo se han comentado brevemente algunas de las medidas del estado del arte para la caracterización de los datos, de lo que se ha podido ver que hay escasas soluciones para Big Data en este tópico. Además, se ha introducido al particionamiento del espacio de características, puntualizando el enfoque de rejilla (*grid*).

A su vez, se ha presentado una metodología para la caracterización escalable de problemas Big Data de clasificación, centrada en las zonas solapadas de los datos a la que hemos llamamos GridOverlap-BD (por estar basada en el enfoque *grid*). El aplicar nuestra propuesta sobre un conjunto Big Data habilita a conocer dos aspectos relevantes de los mismos: (1) la correspondencia de una instancia a un área del problema (pura o ambigua); (2) el grado de solapamiento de un conjunto de datos (como medida de complejidad escalable propuesta).
Se ha llevado a cabo un estudio experimental sobre 9 conjuntos Big Data para evaluar nuestra propuesta, de lo que se ha observado la efectiva distinción de los datos en áreas puras y ambiguas para la mayoría de los conjuntos; el grado de solapamiento (pudiendo entender con ello el bajo desempeño predictivo de algunos de los conjuntos); y una comparación sobre la solución base y los enfoques del estado del arte.

Para futuras experimentaciones se plantean las siguientes ideas y/o extensiones a esta línea de investigación: (a) profundizar el análisis para la definición

de la proporción entre clases que define a una celda pura; (b) desagregar la identificación de celdas en, por ejemplo, área ambigua balanceada, área ambigua semi-balanceada, entre otras; (c) guiar el proceso de la selección de granularidad/es para utilizar en GridOverlap-BD según el conjunto de datos con el que se trabaje; (d) emplear la caracterización para el filtrado y selección de instancias como datos de entrada a una técnica concreta de preprocesamiento; (e) construir un clasificador *ensemble* a partir de la identificación de clase de una instancia dada a través de distintas granularidades, para su posterior agregación.

Todo lo abarcado por este capítulo se vincula a un trabajo que se encuentra en redacción, para ser enviado a revisión a una revista científica.

Parte III

Casos de uso

9 Emergencias humanitarias

Tener la capacidad y las herramientas con las que adelantarse o combatir situaciones de emergencias humanitarias (EH) es de vital importancia. Cada año vemos cómo diferentes desastres naturales, crisis sanitarias, o incluso conflictos armados afectan distintas partes del mundo, causando daños incalculables. Al mismo tiempo, con el enorme avance de la tecnología y su disponibilidad a nivel global, la cantidad de datos provenientes de sensores, satélites y apps, sumado a la mayor capacidad de recolectar información en sociedades cada vez más digitalizadas, provee un escenario de Big Data tratable en el contexto de esta tesis.

En este capítulo, se aplican algunas de las técnicas desarrolladas a lo largo de esta tesis, sobre una serie de conjuntos Big Data de clasificación no balanceada, relacionados con distintos tipos y aspectos de las posibles situaciones de EH. El objetivo es mostrar su aplicación en un contexto real, así como la bondad para obtener soluciones útiles y de alta calidad. En primer lugar, se presentan los distintos conjuntos de datos de EH y el problema que cada uno de ellos representa en la Sección 9.1. A continuación, en la Sección 9.2, se lleva a cabo un breve análisis exploratorio y un proceso estándar de limpieza de los datos, fase que como se ha determinado a lo largo de esta disertación de tesis, resulta ampliamente necesaria para preparar los datos. Luego, en la Sección 9.3, se muestra el desempeño predictivo cuando se entrena un algoritmo clasificador a partir de los datos sin ningún tipo de preprocesamiento, y se lo compara luego de aplicar las técnicas de sobremuestreo para Big Data, en concreto, ROS y SMOTE-BD, siendo esta última especialmente relevante por considerarse aportación explícita en esta tesis doctoral. Nótese que en ninguno de los casos se decidió usar RUS; la razón principal es que, al tratarse de situaciones de EH, parece necesario conservar toda información que pudiera ser valiosa en estas críticas situaciones. Siguiendo la misma lógica, nuestro aporte para la reducción dual de los datos (FDR^2-BD) no se aplica en este contexto. Además, el objetivo de este estudio se enfoca principalmente a mostrar la necesidad en el uso de preprocesamiento escalable de instancias para Big Data, más allá de obtener un valor óptimo de predicción. A continuación, se extiende el análisis incluyendo los valores de la métrica GOD que surge de aplicar GridOverlap-BD con el fin de conocer el grado de solapamiento de los datos. Puesto que, para todo caso de estudio en Ciencia de Datos, se ha hecho especial hincapié en conocer a priori la dificultad del problema para anticiparse a los resultados obtenidos. En ese sentido, GOD será un estimador de la necesidad de aplicar técnicas de preprocesamiento para mejo-

rar el comportamiento de los modelos de aprendizaje. A su vez, confirmará cuán complicado resulta abordar problemas del tipo de EH en la vida real, debido tanto a la dificultad en la captación y gestión de los datos, como a la propia naturaleza del problema con una difícil separación de clases. Finalmente, en la Sección 9.4, se comenta el contenido global del capítulo.

9.1. Descripción de los conjuntos de datos de EH

Los conjuntos de datos de este capítulo no fueron utilizados previamente, ya que el objetivo principal era el desarrollo de técnicas genéricas o multipropósito, sin un sesgo con las EH.

Si bien la cantidad de datos que pueden recolectarse actualmente entra en la categoría de Big Data, la disponibilidad de dichos conjuntos de datos de manera pública es mucho más reducida o limitada. Esto se entiende principalmente debido a lo crítico de muchas de las situaciones, cuestiones de privacidad y decisiones gubernamentales de no publicar dicha información.

Sin embargo, en lo que sigue, se presentan 6 conjuntos de datos relacionados a distintos aspectos de las EH. Los mismos abarcan la detección de incendios forestales a partir de información satelital de la NASA, la detección de Covid19 a partir de la sintomatología o del resultado de análisis de sangre de pacientes, la predicción del daño generado por un terremoto, la predicción de la estabilidad de la red eléctrica, y la utilización de datos de internaciones en hospitales para predecir el tiempo de estadía de un nuevo paciente.

Incendios Forestales (`Incen`) Los incendios forestales son situaciones críticas que se dan a nivel global y con una frecuencia en aumento. Este tipo de incendios pueden tener diversas causas, pero con similares consecuencias. Destrucción de la flora y fauna de una región, desplazamiento de población y daños económicos. Si bien existen diversos recursos y dispositivos para poder monitorizar este tipo de situaciones, es indispensable automatizar el procesamiento del gran volumen de datos que ellos colectan.

En particular, el conjunto de datos utilizado para clasificar incendios forestales proviene del proyecto *Earth Data - Open access for open science*[37] de la NASA. Dichos datos fueron colectados por el espectrorradiómetro de imágenes de resolución moderada (o MODIS, por sus siglas en inglés) que brinda información para identificar incendios activos y otras anomalías térmicas (como los volcanes, por ejemplo). Estos incendios representan el centro de un pixel de 1km que es

37 https://earthdata.nasa.gov/

marcado por el algoritmo MODIS MOD14/MYD14 [Gig+03] por contener uno o más incendios dentro de ese píxel.

El conjunto de datos posee 14 atributos y la variable objetivo es la potencia radiativa del fuego (FRP). En base a la información provista por la fuente de los datos, el problema se binariza al considerar una instancia como «sin incendios» si tiene una $FRP \leq 100$, y «con incendios» en caso contrario.

Covid19 (`CoPCR` y `CoSIN`) El impacto de la pandemia del SARS-Cov2 está todavía por calcularse, pero ya ha sido demasiado alto. Las áreas en donde se puede contribuir son incontables, debido a la masividad y complejidad de la situación. De la cantidad de datos generados y recolectados, sólo una porción está disponible y, al mismo tiempo, producto de la entendible saturación del sistema sanitario, la calidad de dichos datos no siempre es la deseada. En primer lugar, se trabaja con la predicción de "estado de contagio" (es decir, «contagiado» o «no contagiado») de la enfermedad a partir de estudios de sangre y test PCR a los pacientes. Para esto, se trabaja con datos reales provenientes del esfuerzo del proyecto *COVID-19 Data Sharing/BR* de la FAPESP y la Universidad de San Pablo (Brasil), junto a hospitales de la ciudad, donde se realizaron los estudios [FAP20]. Este conjunto de datos cuenta con 23 atributos relacionados a los estudios de sangre, además de la edad y el sexo de los pacientes. La variable objetivo es el resultado del test PCR. A este conjunto de datos se lo denomina CoPCR.

Adicionalmente al caso de estudio anterior, se añade un segundo conjunto para la predicción a partir de la sintomatología de los pacientes. En este caso, los datos son simulados principalmente por dos razones. En primer lugar, debido a la escasez de datos disponibles públicamente. En segundo lugar, en virtud de la dificultad de contar con conjuntos de datos de alta calidad. Esto es esperable ya que el paciente, al momento de completar un formulario sobre sus síntomas, pueda olvidar o confundir alguno de ellos dado la situación crítica que atraviesa. La simulación[38] se basó en un estudio realizado por Mayo Clinic[39], en USA, donde se muestran las diferencias sintomatológicas entre una alergia, un resfrío, gripe (Influenza A y B) y Covid19. El conjunto tiene 21 atributos y el problema se binariza al considerar el escenario Covid19 «si» o «no». A este conjunto de datos lo llamamos CoSIN.

Gestión Hospitalaria (`Hospi`) Un aspecto importante, pero que tal vez no sea el primero que uno imagina al pensar en una EH, está relacionado con la disponibilidad de camas en el sistema hospitalario. Con la pandemia de Covid19

38 https://github.com/WalterConway/SymptomGenerator
39 https://www.mayoclinic.org/diseases-conditions/coronavirus/in-depth/covid-19-cold-flu-and-allergies-differences/art-20503981

se vio de manera evidente la saturación de hospitales y la necesidad de "aplanar la curva" para que esto no suceda. Por esta razón, una predicción del tiempo estimado que un paciente va a permanecer hospitalizado, puede ser importante para un gobierno local a la hora de gestionar y planificar sus acciones frente a una situación de EH. Conocer la cantidad de pacientes que van a estar internados más de 2 meses, por ejemplo, puede ayudar a optimizar sus tratamientos y la logística interna. El conjunto de datos utilizado pertenece a un *hackathon* realizado por Analytics Vidhya[40], con 16 atributos, siendo el tiempo de permanencia, la variable objetivo. El problema se binariza al considerar estadías «menores» o «mayores» a los 60 días, siendo la de estadías más largas la clase minoritaria.

Terremotos (`Terrem`) Sin duda uno de los desastres naturales más destructivos son los terremotos. La predicción de este tipo de eventos requiere no sólo una enorme cantidad de datos, sino de un modelo tectónico preciso y una expertise que excede a la presente tesis. Sin embargo, múltiples aspectos pueden probarse con las técnicas antes descriptas. En particular, se utiliza un conjunto de datos recolectado por la Secretaría Nacional de Planeamiento de Nepal, luego del terremoto de Gorkha[41] de 2015. El gobierno realizó una encuesta masiva con el objetivo de recabar información respecto al daño en edificios y viviendas de la zona afectada, que puede utilizarse para predecir el nivel de daño en un dado edificio luego del terremoto. El conjunto de datos cuenta con 39 atributos, y se lo binariza considerando edificios «con bajo» o «con alto» daño.

Red eléctrica (`RedEl`) La estabilidad de la red eléctrica es un problema que aplica al día a día, pero que se exacerba durante una EH. La dependencia de nuestra sociedad con la electricidad es cada vez mayor, sobre todo considerando los esfuerzos por eliminar o al menos reducir el uso de hidrocarburos. La capacidad de predecir cuándo la red eléctrica se volverá inestable es esencial. Este conjunto de datos [ABJ18; Sch+16], con 13 atributos, permite la clasificación de una red en «estable» o «inestable».

De manera resumida, se muestran las principales características de los datos de EH en la Tabla 9.1, informando el número de ejemplos (#Ex.), número de atributos (#Atts.), el número de ejemplos de cada clase (#(class0; class1)), el porcentaje de distribución de las clases (%(class0; class1)), el número de cada tipo de característica (Co/Di/Ca, donde Co = continuas, Di = discretas, Ca = categóricas), y el tamaño del conjunto de datos (en MB).

40 https://www.analyticsvidhya.com/
41 https://www.drivendata.org/competitions/57/nepal-earthquake/page/134/

Tabla 9.1: Resumen de los conjuntos de datos de EH.

Dataset	#Ex.	#Atts.	#(class0; class1)	%(class0; class1)	Co/Di/Ca	Size (MB)
CoPCR	14681	23	(12664 ; 2017)	(86.3; 13.7)	15/ 7 / 1	2.1
CoSIN	44453	21	(42405 ; 2048)	(95.4; 4.6)	0 / 0 / 21	2
Hospi	318438	16	(291154 ; 27284)	(91.4; 8.6)	3 / 4 / 9	25.7
Incen	1248606	14	(1075353; 173253)	(86.1; 13.9)	7 / 5 / 2	95.1
RedEl	60000	13	(38280 ; 21720)	(63.8; 36.2)	12/ 0 / 1	13.6
Terrem	260601	39	(235477 ; 25124)	(90.4; 9.64)	0 / 31 / 8	24.6

9.2. Análisis exploratorio de los datos

El tratamiento necesario para el estudio de un conjunto de datos de clasificación se describe a continuación de manera general. La idea de esta serie de pasos por los cuales todos los conjuntos de datos de EH antes descriptos son tratados, es la de proveer un análisis exploratorio de los mismos, brindando información concreta de algunas de sus características intrínsecas como se comentó en el Sección 2.2.

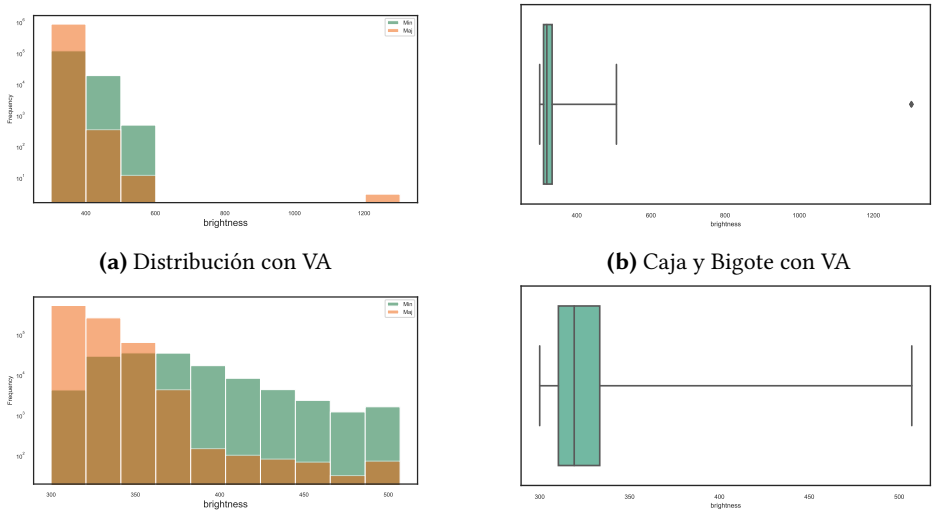
Así pues, en primer lugar, se divide en 80 % y 20 % (de manera aleatoria estratificada) los datos en un conjunto de entrenamiento y otro de evaluación, respectivamente, dejando este último intacto y "oculto" hasta la evaluación final de los diferentes modelos. A la *fold* de entrenamiento, se la analiza buscando valores faltantes y quitando las instancias repetidas (dejando sólo una de sus ocurrencias). Los resultados de este paso de limpieza pueden verse en la Tabla 9.2 para todos los conjuntos Big Data de EH, donde 3 de los 6 datasets muestran tener instancias repetidas; sin embargo, en el peor de los casos, no supera al 6 % del total de sus instancias. Por otra parte, ninguno de los conjuntos presentan valores faltantes en algunos de sus atributos.

Tabla 9.2: Valores faltantes e instancias repetidas de los conjuntos de EH.

	# Instancias Original	Valores faltantes	Instancias Duplicadas	# Instancias final
CoPCR	14681	0	0	14681
CoSIN	44453	0	963	43490
Hospi	318438	0	306	318132
Incen	1248606	0	0	1248606
RedEl	60000	0	0	60000
Terrem	260601	0	15385	245216

Luego, se continúa el análisis a través de herramientas de visualización de los datos. A modo de ejemplo, se las presenta para el conjunto Incen. Los diagramas de caja permiten no sólo conocer el rango y mediana de un atributo, sino también visualizar la presencia de valores atípicos del mismo, que se corroboran al analizar

las distribuciones de dicho atributo. Los gráficos de distribuciones usualmente permiten también identificar, de manera rápida (dependiendo del conjunto de datos), el poder dicriminante de alguna variable. Así pues, en la Figura 9.1, se muestran las distribuciones por clases y los diagramas de caja y bigote para la variable `Brightness`, antes (Figuras 9.1 (a) y 9.1 (b)) y después (Figuras 9.1 (c) y 9.1 (d)) de remover los valores atípicos que presenta el dataset. En este caso la identificación de los valores atípicos se hizo mediante el filtrado de aquellas instancias que tengan en su atributo `Brightness` un valor superior a 500 (valor que se obtiene del diagrama de cajas de arriba).



(a) Distribución con VA

(b) Caja y Bigote con VA

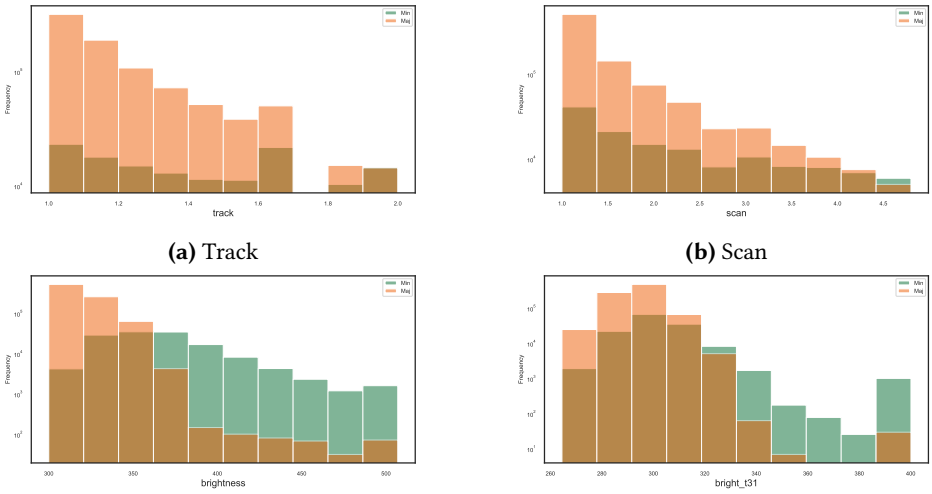
(c) Distribución sin VA

(d) Caja y Bigote sin VA

Figura 9.1: Distribuciones de clases y diagramas de caja y bigote para la variable `Brightness` del conjunto `Incen` de EH, con y sin valores atípicos (VA).

A su vez, en la Figura 9.2 se muestran las cuatro variables más importantes para dicho conjunto (esta información se obtuvo a posteriori). En este caso, se puede ver incluso antes de entrenar cualquier modelo, que las variables `Brightness` y `Bright t31` son altamente discriminantes, al notar que dichas variables muestran una distribución de densidad diferente para cada clase.

Además, se muestra la matriz de correlación entre las variables, en la Figura 9.3 (a), y la correlación de las variables independientes con respecto al atributo de clase, en la Figura 9.3 (b). Con esta nueva información, se observa que las variables `track` y `scan` tienen una correlación prácticamente total. Por otro lado, se reafirma lo comentado antes sobre la alta correlación que tiene la variable

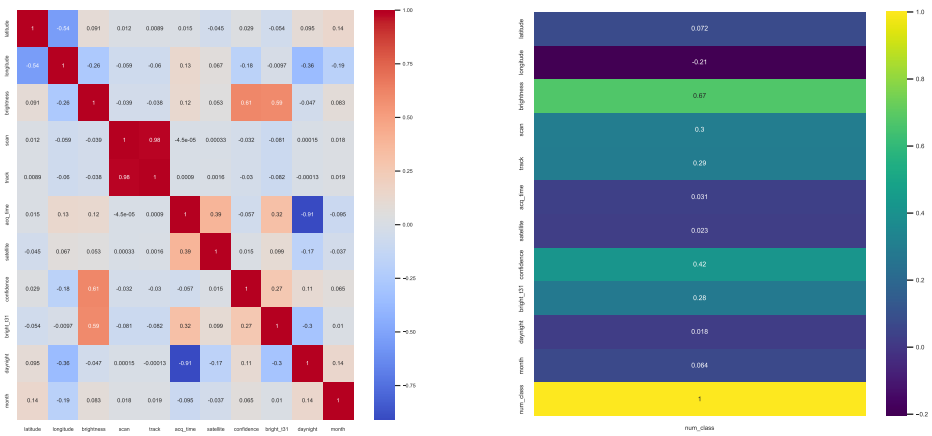


(a) Track **(b)** Scan

(c) Brightness **(d)** Bright t31

Figura 9.2: Distribuciones de clases para las variables más importantes del conjunto `Incen` de EH.

`Brightness` con el atributo de clase, dando indicios de la importancia de dicha variable para el problema de clasificación.



(a) **(b)**

Figura 9.3: Matriz de correlación entre las variables, y correlación de las variables independientes con respecto al atributo de clase, para el conjunto `Incen` de EH.

9.3. Empleo de técnicas de sobremuestreo y caracterización de los datos

Una vez que se aplican todos estos pasos a cada uno de los conjuntos de datos, se considera a la *fold* de entrenamiento como limpia. Como dichos datos son no balanceados, se procede a emplear por un lado SMOTE-BD y por otro lado ROS con el fin de balancearlos para conseguir un perfecto equilibrio entre clases. A partir de ello, el conjunto de entrenamiento está listo para ser utilizado en la etapa de modelado y clasificación. En la Tabla 9.3 se muestra el desempeño predictivo (medido por la GM) cuando se entrena un RF a partir de los datos sin preprocesar (BSL) y preprocesados. Los valores se obtienen de calcular el promedio a partir de los resultados de emplear *5fcv*.

Tabla 9.3: Desempeño predictivo para los conjuntos de EH.

	CoPCR	CoSIN	Hospi	Incen	RedEl	Terrem
BSL	0.5666	0.4049	0.5469	0.9818	0.9372	0.6356
ROS	0.6519	0.5907	0.6534	0.9873	0.9443	0.7353
SMOTE-BD	**0.7204**	0.5931	**0.6974**	0.9871	0.9429	**0.7472**

De la tabla se observa que preprocesar los datos efectivamente contribuye a un modelado de mejor calidad. La mejora para los conjuntos `Incen` y `RedEl` es menor al 1 %, pero esperable dado su alto rendimiento base. Para el resto de los conjuntos, se ve que el preprocesamiento se vuelve esencial, dando mejoras de entre 10 y 15 %. Con respecto a la comparación entre SMOTE-BD y ROS, se puede ver un comportamiento superior del primero para los conjuntos `Hospi`, `Terrem` y `CoPCR`, obteniendo para este último caso una mejora de casi 7 %.

Finalmente, el grado de solapamiento (GOD) de los conjuntos EH, que deriva de aplicar GridOverlap-BD, se muestra en la Tabla 9.4.

Tabla 9.4: Grado de solapamiento (GOD) para los conjuntos de EH.

Dataset	GOD
CoPCR	0.65
Hospi	0.61
Terrem	0.58
RedEl	0.32
Incen	0.27
CoSIN	–

En primer lugar, se comenta que para el conjunto CoSIN, no se tienen resultados dado que todas sus características son categóricas, motivo por el cual el enfoque de particionar el espacio del problema utilizando la estrategia planteada en GridOverlap-BD carece de sentido.

De la tabla se puede observar que CoPCR, Hospi y Terrem presentan valores de GOD similares entre sí, que se corresponden con los cercanos valores de rendimiento de base (BSL). De manera análoga, RedEl e Incen, que presentaban los valores de base más altos, poseen valores bajos de GOD. Esta correspondencia entre la métrica y BSL coincide con lo observado en la Sección 8.3.3 cuando se presentó esta medida.

9.4. Comentarios del capítulo

En este capítulo se han descripto seis conjuntos de datos tabulares no balanceados como casos de aplicación, los cuales representan situaciones y aspectos de EH. Luego se ha llevado a cabo un breve análisis exploratorio sobre los mismos, a partir de lo cual se han mostrado algunos ejemplos del tipo de información que se puede obtener. A continuación, se ha presentado el desempeño predictivo obtenido a partir de utilizar los datos no balanceados y a partir de los datos preprocesados con las técnicas escalables ROS y SMOTE-BD, siendo esta última aporte de esta tesis. Asimismo, y con el fin de extender el análisis sobre los casos de uso, se ha empleado nuestra metodología para la caracterización del solapamiento de conjuntos Big Data y, a partir de lo cual, se han mostrado los valores de la métrica GOD. En este contexto de situaciones de EH, se ha decidido no usar RUS ni tampoco nuestro aporte para la reducción dual de los datos, FDR^2-BD, con el fin de conservar toda información que pudiera ser valiosa para este tipos de escenarios.

Parte IV

Conclusiones

10 Conclusiones y trabajo a futuro

Conclusiones y principales aportes

En la actual era de la información, el análisis asociado al escenario de Big Data permite la extracción de conocimiento de una vasta fuente de información. Una de las cuestiones de interés para extraer y explotar el valor de los datos, es adaptar y simplificar los datos en crudo que son entrada para el algoritmo de aprendizaje, lo que se conoce como "Smart Data". A pesar de la importancia de lo anterior, y su aplicación en problemas estándar, el análisis de la calidad de los datos de los conjuntos Big Data es casi un territorio inexplorado. En este sentido, un estudio exhaustivo de las características de los datos, junto con la aplicación de las técnicas de preprocesamiento adecuadas, se ha convertido en un paso obligatorio para todos los proyectos de Ciencia de Datos, tanto en la industria como en el mundo académico, y en especial aquellos asociados con análisis en Big Data.

En consecuencia, el eje principal de investigación de la presente tesis abordó el preprocesamiento distribuido y escalable de conjuntos Big Data de clasificación binaria, con el fin de obtener el ya citado Smart Data. Teniendo en cuenta el impacto que tienen las características intrínsecas de los datos en el rendimiento de los modelos de aprendizaje, así como la escasa cantidad de soluciones existentes para escenarios Big Data, en esta memoria de tesis se presentaron tres propuestas para la identificación y/o el tratamiento de las siguientes características: (a) datos no balanceados; (b) redundancia; (c) alta dimensionalidad; y (d) solapamiento.

Respecto a los datos no balanceados, se presentó SMOTE-BD, un SMOTE para Big Data basado en un estudio sobre las particularidades necesarias para que su diseño sea totalmente escalable, y que además su comportamiento se ajuste lo más fielmente posible a la técnica secuencial del estado del arte (tan popular en escenarios Small Data). Asimismo, se introdujo una variante de SMOTE-BD, denominada SMOTE-MR, que sigue un diseño tal que procesa los datos localmente en cada nodo. Dado que no existe una única técnica que siempre sea la que genere los mejores resultados, cuando se tiene que equilibrar las clases de un problema, se suelen aplicar una serie de ellas. Es por esto que nuestro aporte toma mayor relevancia puesto que, hasta el momento de su desarrollo, sólo estaban disponibles las soluciones triviales basadas en muestreo aleatorio.

En relación a la redundancia y a la alta dimensionalidad de los datos, se presen-

tó FDR^2-BD, una metodología escalable para reducir (o condensar) un conjunto Big Data de manera dual vertical y horizontal, es decir, reducción de atributos y de instancias, con la premisa de mantener la calidad predictiva respecto de los datos originales. La propuesta se basa en un esquema de validación cruzada donde se realiza un proceso de hiperparametrización que, además, soporta el manejo de conjuntos de datos no balanceados. FDR^2-BD permite conocer si un conjunto de datos dado es reducible manteniendo el poder predictivo de los datos originales dentro de un umbral que puede ser establecido por la persona experta en el dominio del problema. Por consiguiente, nuestra propuesta informa cuáles son los atributos de los datos de mayor importancia y cuál es el porcentaje de reducción uniforme de instancias que se puede llevar a cabo.
Los resultados mostraron la fortaleza de FDR^2-BD obteniendo valores de reducción muy elevados para la mayoría de los conjuntos de datos estudiados, tanto en lo que respecta a la dimensionalidad como a los porcentajes de reducción de instancias propuestos. En términos concretos, se alcanzó alrededor del 70 % de reducción de las características y 98 % de reducción de las instancias, para un umbral de pérdida predictiva máxima aceptada del 1 % del cual, en algunos casos, la calidad predictiva se mantuvo igual a la del conjunto original. Esta información condensada provee la ventaja de poder ser usada en infraestructuras más sencillas que las dedicadas para el procesamiento de Big Data, además de habilitar su uso con técnicas de explicabilidad/interpretabilidad como LIME ó SHAP, cuya complejidad computacional es al menos $O(n^2 \cdot d)$, con n y d número de instancias y variables respectivamente.

En cuanto al solapamiento, se presentó GridOverlap-BD, una metodología para la caracterización escalable de problemas Big Data de clasificación. La propuesta se apoya en el particionamiento del espacio de características basado en rejilla. GridOverlap-BD permite identificar o caracterizar las áreas del problema en dos tipologías: zonas puras y solapadas. Además, se introdujo una métrica de complejidad derivada de aplicar GridOverlap-BD, con foco en cuantificar el solapamiento presente en los datos. De la experimentación realizada, se observó que tanto la caracterización de las zonas de un problema como la cuantificación del grado de solapamiento se llevaron a cabo de manera efectiva para los conjuntos de datos del entorno experimental. Ello implica una aproximación pionera escalable y totalmente agnóstica (independiente del modelo) para la caracterización de las instancias de un problema Big Data, y la estimación de su complejidad de cara al análisis de los resultados posteriores del modelado.

Todas las propuestas fueron desarrolladas utilizando el framework Apache Spark, dado que se ha convertido en un estándar "de facto" para el procesamiento de Big Data. Además, las implementaciones se encuentran disponibles en repositorios de público acceso, en aras de facilitar la reproducibilidad de los resultados,

así como la posible extensión de las aproximaciones diseñadas en la presente tesis doctoral para cualquier investigador interesado.

Trabajo a futuro

Puesto que hay pocas soluciones para el preprocesamiento de datos en Big Data, y menos aún para problemas de clasificación con múltiples clases, nos planteamos como trabajo a futuro extender todas nuestras propuestas para problemas de este tipo. Esta extensión podría parecer trivial a simple vista, pero hay que ser consciente de la complejidad inherente al incremento del número de fronteras y/o funciones discriminantes para distinguir cada concepto de interés, la representación dispar y el sesgo específico sobre cada una de las clases, así como la influencia específica de diferentes variables de entrada.

En relación al equilibrio de clases, se podría llevar a cabo el desarrollo de modelos híbridos (combinación entre sobre y bajo-muestreo) centrados en las zonas donde el remuestreo es especialmente necesario. Asimismo, combinar la tarea de remuestreo con un paso previo de selección de las características más relevantes del problema para abarcar en conjunto a los datos no balanceados y la alta dimensionalidad normalmente presente en los conjuntos Big Data. Esta acción de reducción de dimensionalidad se ha observado indispensable para la obtención de datos de calidad a lo largo de las diferentes propuestas de la presente tesis doctoral.

En cuanto a la condensación de los datos manteniendo su poder predictivo, se propone aplicar sobre el conjunto reducido la identificación de áreas solapadas o alguna otra caracterización de los datos a definir, y así intentar mejorar aún más la calidad de los datos. Esta sinergia de aproximaciones está orientada a facilitar la identificación de áreas o instancias del problema que puedan resultar problemáticas para el modelado, y poder actuar en base tanto a su eliminación, como la captación o generación de nuevos datos en dichas zonas de interés.

Con respecto a GridOverlap-BD, se podrían desagregar los tipos actuales de celdas en más subtipos, considerando distintos escenarios de proporciones entre clases. A su vez, se podrían emplear otras métricas de complejidad de los datos como guías para seleccionar una granularidad que se adecúe al conjunto Big Data con el que se esté experimentando. Asimismo, la obtención del subconjunto de instancias solapadas puede ser utilizado como datos de entrada para una técnica concreta de preprocesamiento. Además, se propone el diseño de un clasificador *ensemble* a partir de la identificación de clase de una instancia dada a través de distintas granularidades, para su posterior agregación y obtención de etiqueta de clase final. Este tipo de solución puede ser útil no solo con el objetivo de mejorar la construcción del modelo, sino incluso para identificar ejemplos de etiqueta

dudosa que puedan ser causa de algún tipo de sesgo o falta de equidad en los datos.

Contribuciones

Las propuestas presentadas a lo largo de esta memoria de tesis forman parte de los siguientes trabajos publicados:

Basgall, M. J.; Naiouf, M.; Fernández, A. *FDR^2-BD: A Fast Data Reduction Recommendation Tool for Tabular Big Data Classification Problems.* Electronics 2021, 10 (15), 1757. https://doi.org/10.3390/electronics10151757.

Basgall, M. J., Hasperué, W., Naiouf, M., Fernández, A., & Herrera, F. (2019). *An Analysis of Local and Global Solutions to Address Big Data Imbalanced Classification: A Case Study with SMOTE Preprocessing.* Cloud Computing and Big Data (Vol. 1050, pp. 75–85). Springer International Publishing. https://doi.org/10.1007/978-3-030-27713-0_7

Basgall, M. J., Hasperué, W., Naiouf, M., Fernández, A., & Herrera, F. (2018). *SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data.* Journal of Computer Science and Technology, 18(03), e23. https://doi.org/10.24215/16666038.18.e23.

Bibliografía

[Aca+18] G. Acampora et al. «A multi-objective evolutionary approach to training set selection for support vector machine». *Knowl-based. Syst.* (2018). DOI: 10.1016/j.knosys.2018.02.022 (pág. 77).

[Adm] N. H. T. S. Administration. *Fatality Analysis Reporting System (FARS)*. nhtsa.gov/research-data/fatality-analysis-reporting-system-fars. [Accessed 12/2019] (pág. 152).

[AY09] C. C. Aggarwal y P. S. Yu. «A Survey of Uncertain Data Algorithms and Applications». *IEEE Trans. Knowl. Data Eng.* (2009) (pág. 12).

[AIS93] R. Agrawal et al. «Mining Association Rules between Sets of Items in Large Databases». *SIGMOD '93*. 1993. DOI: 10.1145/170035.170072 (pág. 22).

[AA21] A. Alshammari y A. Aldribi. «Apply machine learning techniques to detect malicious network traffic in cloud computing». *J. Big Data* (2021). DOI: 10.1186/s40537-021-00475-1 (pág. 49).

[Ang07] F. Angiulli. «Fast Nearest Neighbor Condensation for Large Data Sets Classification». *IEEE Trans. Knowl. Data Eng.* (2007). DOI: 10.1109/TKDE.2007.190645 (pág. 41).

[Ari+22] A. Arif et al. «Towards Efficient Energy Utilization Using Big Data Analytics in Smart Cities for Electricity Theft Detection». *Big Data Research* (2022). DOI: 10.1016/j.bdr.2021.100285 (págs. 4, 47).

[Arn+17] A. Arnaiz-González et al. «MR-DIS: democratic instance selection for big data by MapReduce». *Lect. Notes. Artif. Int.* (2017). DOI: 10.1007/s13748-017-0117-5 (pág. 79).

[ABJ18] V. Arzamasov et al. «Towards Concise Models of Grid Stability». *IEEE SmartGridComm*. 2018. DOI: 10.1109/SmartGridComm.2018.8587498 (pág. 122).

[Ase+14] A. S. Asensio et al. «Improving data partition schemes in Smart Grids via clustering data streams». *Expert Syst. Appl.* (2014). DOI: dx.doi.org/10.1016/j.eswa.2014.03.035 (pág. 159).

[Azu05] F. Azuaje. «Pearson RK:Mining Imperfect Data». *BioMed. Eng. OnLine* (2005). DOI: 10.1186/1475-925X-4-43 (pág. 12).

[BL21] S. Bagui y K. Li. «Resampling imbalanced data for network intrusion detection datasets». *J. Big Data* (2021). DOI: 10.1186/s40537-020-00390-x (pág. 62).

[BSW14] P. Baldi et al. «Searching for exotic particles in high-energy physics with deep learning». *Nat. Commun.* (2014) (pág. 154).

[Bar+03] R. Barandela et al. «Strategies for learning in class imbalance problems.» *Pattern Recogn.* (2003). URL: dblp.uni-trier.de/db/journals/pr/pr36.html#BarandelaSGR03 (pág. 31).

[Bar+21] V. H. Barella et al. «Assessing the data complexity of imbalanced datasets». *Inform. Sciences* (2021). DOI: 10.1016/j.ins.2020.12.006 (pág. 100).

[BPM04] G. Batista et al. «A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data». *SIGKDD Explor. Newsl.* (2004). DOI: 10.1145/1007730.1007735 (pág. 38).

[BD] R. Bhatt y A. Dhall. *Skin - UCI repository*. archive-beta.ics.uci.edu/ml/datasets/skin+segmentation. [Accessed 12/2019] (pág. 154).

[BDA] J. A. Blackard et al. *Covertype dataset - UCI repository*. archive.ics.uci.edu/ml/datasets/covertype. [Accessed 12/2019] (pág. 152).

[Bou+21] A. Bousdekis et al. «A Review of Data-Driven Decision-Making Methods for Industry 4.0 Maintenance Applications». *Electronics* (2021). DOI: 10.3390/electronics10070828 (págs. 3, 45).

[Bre96] L. Breiman. «Bagging Predictors». *Machine Learning* (1996). DOI: 10.1007/BF00058655 (pág. 27).

[Bre01] L. Breiman. «Random Forests». *Machine Learning* (2001). DOI: 10.1023/A:1010933404324 (pág. 27).

[BB15] R. Bruni y G. Bianchi. «Effective Classification Using a Small Training Set Based on Discretization and Statistical Analysis». *IEEE Trans. Knowl. Data Eng.* (2015). DOI: 10.1109/TKDE.2015.2416727 (pág. 23).

[BB20] R. Bruni y G. Bianchi. «Website categorization: A formal approach and robustness analysis in the case of e-commerce detection». *Expert Syst. Appl.* (2020). DOI: 10.1016/j.eswa.2019.113001 (pág. 23).

[Cai+18] J. Cai et al. «Feature selection in machine learning: A new perspective». *Neurocomputing* (2018). DOI: doi.org/10.1016/j.neucom.2017.11.077 (pág. 78).

[CCS04] A. Carbone et al. «Time-dependent Hurst exponent in financial time series». *Phys. A: Stat. Mech. Appl.* (2004). DOI: 10.1016/j.physa.2004.06.130 (pág. 164).

[CO] R. Cattral y F. Oppacher. *Poker Hand dataset - UCI repository*. archive.ics.uci.edu/ml/datasets/Poker+Hand. [Accessed 12/2019] (pág. 153).

[Cha+02] N. V. Chawla et al. «SMOTE: Synthetic Minority Over–sampling Technique». *J. Artif. Intell. Res.* (2002) (pág. 38).

[CJK04] N. V. Chawla et al. «Editorial: special issue on learning from imbalanced data sets». *ACM SIGKDD Explor. Newsl.* (2004). DOI: 10.1145/1007730.1007733 (pág. 16).

[Che+09] Y. Chen et al. «Similarity-based Classification: Concepts and Algorithms». *J. Mach. Learn. Res.* (2009) (pág. 24).

[CM07] V. S. Cherkassky y F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. 2007 (pág. 22).

[com] K. competition. *Flight delays*. kaggle.com/c/flight-delays-spring-2018/overview. [Accessed 12/2019] (pág. 151).

[CDH16] P. V. Coveney et al. «Big data need big theory too». *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* (2016). DOI: 10.1098/rsta.2016.0153 (pág. 4).

[CH67] T. Cover y P. Hart. «Nearest neighbor pattern classification». *IEEE Trans. Inf. Theory* (1967). DOI: 10.1109/TIT.1967.1053964 (pág. 24).

[DCB] A. Dal Pozzolo et al. *Credit Card Fraud Detection dataset - Kaggle*. kaggle.com/mlg-ulb/creditcardfraud. [Accessed 12/2019] (pág. 152).

[DKC13] B. Das et al. «Handling Class Overlap and Imbalance to Detect Prompt Situations in Smart Homes». *ICDMW 2013*. 2013. DOI: 10.1109/ICDMW.2013.18 (pág. 43).

[DDC18] S. Das et al. «Handling data irregularities in classification: Foundations, trends, and future challenges». *Pattern Recogn.* (2018) (págs. 4, 42).

[DG04] J. Dean y S. Ghemawat. «MapReduce: Simplified Data Processing on Large Clusters». *OSDI*. 2004. URL: dl.acm.org/citation.cfm?id=1251254.1251264 (pág. 49).

[DG10] J. Dean y S. Ghemawat. «MapReduce: a flexible data processing tool». *Commun. Acm* (2010). DOI: 10.1145/1629175.1629198 (pág. 49).

[DJ15] D. Desai y A. Joshi. «A Deviant Load Shedding System for Data Stream Mining». *Procedia Comput. Sci.* (2015). DOI: dx.doi.org/10.1016/j.procs.2015.03.103 (pág. 159).

[DBP17] D. Devi et al. «Redundancy-driven modified Tomek-link based undersampling». *Pattern Recogn. Lett.* (2017). DOI: 10.1016/j.patrec.2016.10.006 (pág. 12).

[Dom99] P. Domingos. «MetaCost: a general method for making classifiers cost-sensitive». *5th ACM SIGKDD*. 1999. DOI: 10.1145/312129.312220 (pág. 37).

[Dom21] Domo. *Data never spleeps 9.0*. domo.com/learn/video/data-never-sleeps-9. 2021 (págs. 3, 45).

[Elk+18] M. Elkano et al. «CHI-PG: A fast prototype generation algorithm for Big Data classification problems». *Neurocomputing* (2018). DOI: doi.org/10.1016/j.neucom.2018.01.056 (pág. 102).

[FAP20] FAPESP. *COVID-19 Data Sharing/BR*. repositoriodatasharingfapesp.uspdigital.usp.br. 2020 (pág. 121).

[Far+20] A. Z. Faroukhi et al. «Big data monetization throughout Big Data Value Chain: a comprehensive review». *J. Big Data* (2020). DOI: 10.1186/s40537-019-0281-5 (pág. 47).

[Faw06] T. Fawcett. «An introduction to ROC analysis». *Pattern Recogn.Lett.* (2006). DOI: dx.doi.org/10.1016/j.patrec.2005.10.010 (pág. 31).

[Fer+17] A. Fernandez et al. «An Insight into Imbalanced Big Data Classification: Outcomes and Challenges». *Complex Intell. Syst.* (2017) (págs. 61-63, 88).

[Fer+18a] A. Fernandez et al. «SMOTE for Learning from Imbalanced Data: Progress and Challenges.» *J. Artif. Intell. Res.* (2018). DOI: doi.org/10.1613/jair.1.11192 (pág. 38).

[Fer+19] A. Fernandez et al. «Evolutionary Fuzzy Systems for Explainable Artificial Intelligence: Why, When, What for, and Where to?» *IEEE Comput. Intell. Mag.* (2019). DOI: 10.1109/MCI.2018.2881645 (pág. 56).

[Fer+14] A. Fernández et al. «Big Data with Cloud Computing: An Insight on the Computing Environment, MapReduce and Programming Framework». *WIREs Data Min. Knowl. Disc.* (2014) (pág. 4).

[Fer+18b] A. Fernández et al. «Learning from Imbalanced Data Sets». 2018. DOI: 10.1007/978-3-319-98074-4_10 (págs. 4, 12, 16, 35, 36, 38).

[Gal+18] D. Galpert et al. «Surveying alignment-free features for Ortholog detection in related yeast proteomes by using supervised big data classifiers». *BMC Bioinformatics* (2018) (pág. 61).

[GCL15] L. P. Garcia et al. «Effect of label noise in the complexity of classification problems». *Neurocomputing* (2015). DOI: 10.1016/j.neucom.2014.10.085 (pág. 100).

[Gar+12] S. Garcia et al. «Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study». *IEEE Trans. Pattern Anal. Mach. Intell.* (2012). DOI: 10.1109/TPAMI.2011.142 (pág. 78).

[GCH08] S. García et al. «A memetic algorithm for evolutionary prototype selection: A scaling up approach». *Pattern Recogn.* (2008). DOI: 10.1016/j.patcog.2008.02.006 (pág. 41).

[GLH15] S. García et al. *Data Preprocessing in Data Mining*. 2015. DOI: 10.1007/978-3-319-10247-4 (págs. 4, 11, 35, 37).

[GSM12] V. García et al. «On the effectiveness of preprocessing methods when dealing with different levels of class imbalance». *Knowl-based. Syst.* (2012). DOI: 10.1016/j.knosys.2011.06.013 (pág. 36).

[Gar+19] D. García-Gil et al. «Big Data Preprocessing as the Bridge between Big Data and Smart Data». *BDIoT*. 2019. DOI: 10.5220/0007738503240331 (pág. 79).

[GdG10] C. García-Osorio et al. «Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts». *Artif. Intell.* (2010). DOI: doi.org/10.1016/j.artint.2010.01.001 (pág. 41).

[Gig+03] L. Giglio et al. «An Enhanced Contextual Fire Detection Algorithm for MODIS». *R. Sens. Environ.* (2003). DOI: 10.1016/S0034-4257(03)00184-6 (pág. 121).

[Gil+19] D. G. Gil et al. «Enabling Smart Data: Noise filtering in Big Data classification». *Inf. Sci.* (2019). DOI: 10.1016/j.ins.2018.12.002 (pág. 48).

[Gon17] R. J. González. «Hacking the citizenry?: Personality profiling, 'big data' and the election of Donald Trump». *Anthropology Today* (2017). DOI: 10.1111/1467-8322.12348 (pág. 4).

[GCA20] D. Grimaldi et al. «Inferring the votes in a new political landscape: the case of the 2019 Spanish Presidential elections». *J. Big Data* (2020). DOI: 10.1186/s40537-020-00334-5 (pág. 47).

[Guo+17] H. Guo et al. «Learning from class-imbalanced data: Review of methods and applications». *Expert Syst. Appl.* (2017) (pág. 35).

[Gut+17a] P. D. Gutierrez et al. «SMOTE-GPU: Big Data Preprocessing on Commodity Hardware for Imbalanced Classification». *Lect. Notes. Artif. Int.* (2017) (pág. 71).

[Gut+16] P. D. Gutiérrez et al. «GPU-SME- k NN: Scalable and memory efficient k NN and lazy learning using GPUs». *Inform. Sciences* (2016). DOI: 10.1016/j.ins.2016.08.089 (pág. 63).

[Gut+17b] P. D. Gutiérrez et al. «SMOTE-GPU: Big Data preprocessing on commodity hardware for imbalanced classification». *Lect. Notes. Artif. Int.* (2017). DOI: 10.1007/s13748-017-0128-2 (pág. 63).

[GE03] I. Guyon y A. Elisseeff. «An Introduction to Variable and Feature Selection». *J. Mach. Learn. Res.* (2003) (pág. 42).

[HKP11] J. Han et al. *Data Mining: Concepts and Techniques*. 2011. URL: sciencedirect.com/science/book/9780123814791 (pág. 47).

[HFB19] R. H. Hariri et al. «Uncertainty in big data analytics: survey, opportunities, and challenges». *J. Big Data* (2019). DOI: 10.1186/s40537-019-0206-3 (pág. 78).

[Har75] J. A. Hartigan. *Clustering Algorithms*. Wiley S. Pro. 1975 (págs. 22, 161).

[Has+20] T. Hasanin et al. «Investigating Class Rarity in Big Data». *J. Big Data* (2020). DOI: 10.1186/s40537-020-00301-0 (págs. 14, 62).

[HBK19] M. Herland et al. «The Effects of Class Rarity on the Evaluation of Supervised Healthcare Fraud Detection Models». *J. Big Data* (2019). DOI: 10.1186/s40537-019-0181-8 (págs. 14, 62).

[HAP89] R. C. Holte et al. «Concept learning and the problem of small disjuncts». *IJCAI '89* (1989) (pág. 14).

[Iaf14] F. Iafrate. «A Journey from Big Data to Smart Data». 2014. DOI: 10.1007/978-3-319-04313-5_3 (pág. 4).

[JL05] J. Huang y C. X. Ling. «Using AUC and accuracy in evaluating learning algorithms». *IEEE Trans. Knowl. Data Eng.* (2005). DOI: 10.1109/TKDE.2005.50 (pág. 31).

[Kar+21] G. Karagiorgi et al. «Machine Learning in the Search for New Fundamental Physics» (2021). DOI: 10.48550/ARXIV.2112.03769 (pág. 47).

[Kim09] J.-H. Kim. «Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap». *Comput. Stat. Data An.* (2009). DOI: 10.1016/j.csda.2009.04.009 (pág. 32).

[KA15] S.-S. Kim y H.-K. Ahn. «An improved data stream algorithm for clustering». *Comp. Geom.* (2015). DOI: dx.doi.org/10.1016/j.comgeo.2015.06.003 (pág. 159).

[KO09] S.-W. Kim y B. J. Oommen. «On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures». *Pattern Recogn.* (2009). DOI: 10.1016/j.patcog.2009.04.019 (pág. 100).

[Koh95] R. Kohavi. «A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection». IJCAI'95. 1995 (pág. 32).

[KDA19] T. Kolajo et al. «Big data stream analysis: a systematic literature review». *J. Big Data* (2019). DOI: 10.1186/s40537-019-0210-7 (pág. 159).

[Kra13] T. Kraska. «Finding the Needle in the Big Data Systems Haystack». *IEEE Internet Comput.* (2013). DOI: 10.1109/MIC.2013.10 (pág. 46).

[Kra16a] B. Krawczyk. «Learning from imbalanced data: open challenges and future directions». *Prog. Artif. Intell.* (2016). DOI: 10.1007/s13748-016-0094-0 (pág. 38).

[Kra16b] B. Krawczyk. «Learning from imbalanced data: open challenges and future directions». *Lect. Notes. Artif. Int.* (2016) (pág. 62).

[Kun14] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. 2014. DOI: 10.1002/9781118914564 (pág. 27).

[LJ12] A. Labrinidis y H. V. Jagadish. «Challenges and opportunities with big data». *VLDB* (2012). DOI: 10.14778/2367502.2367572 (pág. 47).

[Len+08] P. Lenca et al. «A Comparison of Different Off-Centered Entropies to Deal with Class Imbalance for Decision Trees». 2008. DOI: 10.1007/978-3-540-68125-0_59 (pág. 36).

[LP21] A. León y Ó. Pastor. «Enhancing Precision Medicine: A Big Data-Driven Approach for the Management of Genomic Data». *Big Data Research* (2021). DOI: 10.1016/j.bdr.2021.100253 (pág. 47).

[Li+14] Y. Li et al. «Incremental Entropy-based Clustering on Categorical Data Streams with Concept Drift». *Know.-Based Syst.* (2014). DOI: 10.1016/j.knosys.2014.02.004 (págs. 159, 160).

[Lic13] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: archive.ics.uci.edu/ml (págs. 71, 86, 151).

[Lin13] J. Lin. «Mapreduce is Good Enough?If All You Have is a Hammer, Throw Away Everything That's Not a Nail!» *Big Data* (2013). DOI: 10.1089/big.2012.1501 (pág. 53).

[LM98] H. Liu y H. Motoda. *Feature selection for knowledge discovery and data mining*. 1998 (pág. 41).

[LM01] H. Liu y H. Motoda, eds. *Instance Selection and Construction for Data Mining*. 2001. DOI: 10.1007/978-1-4757-3359-4 (págs. 40, 78).

[LM07] H. Liu y H. Motoda, eds. *Computational methods of feature selection*. 2007 (págs. 41, 78).

[Lóp+13] V. López et al. «An Insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics». *Inform. Sciences* (2013) (págs. 35, 36, 57, 62, 63).

[Lor+19] A. C. Lorena et al. «How Complex is your classification problem? A survey on measuring classification complexity» (2019). URL: arxiv.org/abs/1808.03591 (visitado 2020) (págs. 17, 100).

[Lu17] Y. Lu. «Industry 4.0: A survey on technologies, applications and open research issues». *J. Ind. Inf. Int.* (2017). DOI: 10.1016/j.jii.2017.04.005 (págs. 3, 45).

[Lue+20] J. Luengo et al. *Big Data Preprocessing: Enabling Smart Data*. 2020. DOI: 10.1007/978-3-030-39105-8 (págs. 4, 48, 78).

[LS15] E. Lughofer y M. Sayed-Mouchaweh. «Autonomous Data Stream Clustering Implementing Split-and-merge Concepts - Towards a Plug-and-play Approach». *Inf. Sci.* (2015). DOI: 10.1016/j.ins.2015.01.010 (pág. 159).

[MMC15] G. Machado et al. «What variables are important in predicting bovine viral diarrhea virus? A random forest approach». *Vet. Res.* (2015). DOI: 10.1186/s13567-015-0219-7 (pág. 27).

[MTH20] J. Maillo et al. «Redundancy and Complexity Metrics for Big Data Classification: Towards Smart Data». *IEEE Access* (2020). DOI: 10.1109/ACCESS.2020.2991800 (págs. 4, 40, 77, 90, 100).

[Mai+17] J. Maillo et al. «kNN–IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data.» *Knowl-based. Syst.* (2017) (pág. 64).

[MVG21] I. Martinez et al. «Data Science Methodologies: Current Challenges and Future Approaches». *Big Data Research* (2021). DOI: 10.1016/j.bdr.2020.100183 (pág. 45).

[Mar13] V. Marx. «The big challenges of big data». *Nature* (2013). DOI: 10.1038/498255a (pág. 3).

[MG02] S. J. Mason y N. E. Graham. «Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation». *Q. J. Roy. Meteor. Soc.* (2002) (pág. 31).

[Men+21] C. H. Mendhe et al. «A Scalable Platform to Collect, Store, Visualize, and Analyze Big Data in Real Time». *IEEE Trans. Comput. Social Syst.* (2021). DOI: 10.1109/TCSS.2020.2995497 (pág. 47).

[Men+16] X. Meng et al. «MLlib: Machine Learning in Apache Spark». *J. Mach. Learn. Res.* (2016) (págs. 5, 79).

[Mil+14] Z. Miller et al. «Twitter spammer detection using data stream clustering». *Inform. Sciences* (2014). DOI: dx.doi.org/10.1016/j.ins.2013.11.016 (pág. 159).

[Mit13] T. M. Mitchell. *Machine learning*. Series in Comp. Sci. 2013 (pág. 21).

[Mon+19] B. Montesdeoca et al. «A First Approach on Big Data Missing Values Imputation:» *BDIoT*. 2019. DOI: 10.5220/0007738403150323 (pág. 12).

[MD13] T. B. Murdoch y A. S. Detsky. «The Inevitable Application of Big Data to Health Care». *JAMA* (2013). DOI: 10.1001/jama.2013.393 (pág. 4).

[Mur19] C. W. Murphy. «Class imbalance techniques for high energy physics». *SciPost Physics* (2019). DOI: 10.21468/SciPostPhys.7.6.076 (pág. 62).

[Myt+] R. Mythily et al. «Clustering Models for Data Stream Mining». *Procedia Comput. Sci.* DOI: dx.doi.org/10.1016/j.procs.2015.02.107 (pág. 159).

[NMB20] S. Naulaerts et al. «Concise Polygenic Models for Cancer-Specific Identification of Drug-Sensitive Tumors from Their Multi-Omics Profiles». *Biomolecules* (2020). DOI: 10.3390/biom10060963 (pág. 47).

[Nor] E. C. R. of North Rhine-Westphalia ('Epidemiologisches Krebsregister'). *Record Linkage Comparison Patterns dataset - UCI repository*. archive.ics.uci.edu/ml/datasets/record+linkage+comparison+patterns. [Accessed 12/2019] (pág. 153).

[OSL17] L. C. Okimoto et al. «Complexity Measures Effectiveness in Feature Selection». *BRACIS '17*. 2017. DOI: 10.1109/BRACIS.2017.66 (pág. 100).

[Osm19] A. Osman. «A novel big data analytics framework for smart cities». *Future Gener. Comp. Sy.* (2019). DOI: 10.1016/j.future.2018.06.046 (pág. 47).

[PKH16] S.-h. Park et al. «Highway traffic accident prediction using VDS big data analysis». *J. Supercomput.* (2016). DOI: 10.1007/s11227-016-1624-z (pág. 62).

[Par96] S. Parsons. «Current Approaches to Handling Imperfect Information in Data and Knowledge Bases.» *IEEE Trans. Knowl. Data Eng.* (1996). URL: dblp.uni-trier.de/db/journals/tkde/tkde8.html#Parsons96 (pág. 12).

[Pas+20] J. D. Pascual-Triana et al. «Revisiting Data Complexity Metrics Based on Morphology for Overlap and Imbalance: Snapshot, New Overlap Number of Balls Metrics and Singular Problems Prospect». *arXiv:2007.07935* (2020). URL: arxiv.org/abs/2007.07935 (visitado 2021) (pág. 42).

[PAG15] J. Pérez-Rodríguez et al. «Simultaneous instance and feature selection and weighting using evolutionary computation: Proposal and study». *Appl. Soft Comput.* (2015). DOI: 10.1016/j.asoc.2015.07.046 (pág. 78).

[Phr+14] PhridviRaj et al. «Clustering Text Data Streams - A Tree based Approach with Ternary Function and Ternary Feature Vector». *Procedia Comput. Sci.* (2014). DOI: dx.doi.org/10.1016/j.procs.2014.05.350 (pág. 159).

[Pol12] R. Polikar. «Ensemble Learning». 2012. DOI: 10.1007/978-1-4419-9326-7_1 (pág. 27).

[PBM04] R. C. Prati et al. «Class Imbalances versus Class Overlapping: An Analysis of a Learning System Behavior». *MICAI*. 2004 (pág. 16).

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. 1993 (pág. 25).

[Qui87] J. Quinlan. «Simplifying decision trees». *Int. J. Man. Mach. Stud.* (1987). DOI: 10.1016/S0020-7373(87)80053-6 (pág. 27).

[RS01] P. H. R.O. Duda y D. Stork. «Pattern Classification». *J. Classif.* (2001). DOI: 10.1007/s00357-007-0015-9 (pág. 22).

[Ram+18a] S. Ramirez-Gallego et al. «An Information Theory-Based Feature Selection Framework for Big Data Under Apache Spark». *IEEE Trans. Syst., Man, Cybern., Syst.* (2018). DOI: 10.1109/TSMC.2017.2670926 (pág. 79).

[Ram+18b] S. Ramírez-Gallego et al. «Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce». *Information Fusion* (2018). DOI: 10.1016/j.inffus.2017.10.001 (pág. 56).

[Ras18] A. N. M. B. Rashid. «Access methods for Big Data: current status and future directions». *EAI* (2018). DOI: 10.4108/eai.28-12-2017.153520 (pág. 47).

[Ras92] E. Rasmussen. «Information Retrieval». 1992. URL: dl.acm.org/citation.cfm?id=129687.129703 (pág. 162).

[Reh+16] M. H. ur Rehman et al. «Big Data Reduction Methods: A Survey». *Data Sci. Eng.* (2016). DOI: 10.1007/s41019-016-0022-0 (pág. 15).

[Reh+14] M. Z.-u. Rehman et al. «Hyper-ellipsoidal clustering technique for evolving data stream». *Knowl-based. Syst.* (2014). DOI: dx.doi.org/10.1016/j.knosys.2013.11.022 (pág. 159).

[repa] K. repository. *Home Credit Default Risk*. kaggle.com/c/home-credit-default-risk. [Accessed 12/2019] (pág. 153).

[repb] U. repository. *Australian Credit Approval - UCI repository*. archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval). [Accessed 12/2019] (pág. 151).

[repc] U. repository. *Census - UCI repository*. archive.ics.uci.edu/ml/datasets/Census-Income+(KDD). [Accessed 12/2019] (pág. 152).

[repd] U. repository. *Heart - UCI repository*. archive.ics.uci.edu/ml/datasets/statlog+(heart). [Accessed 12/2019] (pág. 151).

[RD18] P. A. A. Resende y A. C. Drummond. «A Survey of Random Forest Based Methods for Intrusion Detection Systems». *ACM Comput. Surv.* (2018). DOI: 10.1145/3178582 (pág. 42).

[Rij] J. van Rijn. *Agrawal dataset - OpenML repository*. openml.org/d/1235. [Accessed 12/2019] (pág. 151).

[RTV] J. van Rijn et al. «Computing and Predicting Winning Hands in the Trick-Taking Game of Klaverjas». [Accessed 12/2019] (pág. 153).

[Río+14] S. del Río et al. «On the use of MapReduce for imbalanced big data using Random Forest». *Inform. Sciences* (2014). DOI: 10.1016/j.ins.2014.03.043 (pág. 62).

[Roe] B. Roe. *MiniBooNE - UCI repository*. archive.ics.uci.edu/ml/datasets/MiniBooNE+particle+identification. [Accessed 12/2019] (pág. 153).

[RM14] L. Rokach y O. Maimon. *Data Mining with Decision Trees: Theory and Applications*. 2.ª ed. 2014. DOI: 10.1142/9097 (pág. 25).

[Ros+21] M. Rostami et al. «Review of swarm intelligence-based feature selection methods». *Eng. Appl. Artif. Intel.* (2021). DOI: doi.org/10.1016/j.engappai.2021.104210 (pág. 78).

[SAV08] Y. Saeys et al. «Robust Feature Selection Using Ensemble Feature Selection Techniques». *MLKDD*. 2008. DOI: 10.1007/978-3-540-87481-2_21 (pág. 42).

[SBA20] R. Sahal et al. «Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case». *J. Manuf. Syst.* (2020). DOI: 10.1016/j.jmsy.2019.11.004 (pág. 159).

[Sah+21] S. K. Sahu et al. «An Ensemble-Based Scalable Approach for Intrusion Detection Using Big Data Framework». *Big Data* (2021). DOI: 10.1089/big.2020.0201 (pág. 4).

[SMS07] J. S. Sánchez et al. «An analysis of how training data complexity affects the nearest neighbor classifiers». *Pattern Anal. Appl.* (2007). DOI: 10.1007/s10044-007-0061-2 (pág. 99).

[Sch+16] B. Schäfer et al. «Taming Instabilities in Power Grid Networks by Decentralized Control». *Eur. Phys. J.: Spec. Top.* (2016). DOI: 10.1140/epjst/e2015-50136-y (pág. 122).

[Sch03] R. E. Schapire. «The Boosting Approach to Machine Learning: An Overview». *Lect. Notes. Stat.* 2003. DOI: 10.1007/978-0-387-21579-2_9 (pág. 27).

[Sha48] C. E. Shannon. «A Mathematical Theory of Communication». *Bell Syst. Tech. J.* (1948). DOI: 10.1002/j.1538-7305.1948.tb01338.x (pág. 27).

[Shv+10] K. Shvachko et al. «The Hadoop Distributed File System». *MSST 2010*. 2010. DOI: 10.1109/MSST.2010.5496972 (pág. 51).

[SGS20] D. Singh et al. «Weighted k -nearest neighbor based data complexity metrics for imbalanced datasets». *Stat. Anal. Data. Min.* (2020). DOI: 10.1002/sam.11463 (pág. 100).

[Sin03a] S. Singh. «PRISM ? A novel framework for pattern recognition». *Pattern Anal. Appl.* (2003). DOI: 10.1007/s10044-002-0186-2 (pág. 101).

[Sin03b] S. Singh. «Multiresolution estimates of classification complexity». *IEEE Trans. Pattern Anal. Mach. Intell.* (2003). DOI: 10.1109/TPAMI.2003.1251146 (pág. 101).

[SS04] S. Singh y M. Singh. «On the optimality of image processing pipeline». *Pattern Recogn.* (2004). DOI: 10.1016/j.patcog.2003.08.005 (pág. 101).

[Ska94] D. B. Skalak. «Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms». *ICML '94*. 1994 (pág. 41).

[Skr11] I. Skrypnyk. «Irrelevant Features, Class Separability, and Complexity of Classification Problems». *ICIAI 2011*. 2011. DOI: 10.1109/ICTAI.2011.171 (pág. 100).

[Ste97] S. V. Stehman. «Selecting and interpreting measures of thematic classification accuracy». *Remote Sens. Environ.* (1997). DOI: 10.1016/S0034-4257(97)00083-7 (pág. 28).

[SK01] W. N. Street e Y. Kim. «A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification». *7th ACM SIGKDD*. KDD '01. 2001. DOI: 10.1145/502512.502568 (pág. 154).

[SK] W. N. Street e Y. Kim. *SEA dataset - OpenML repository*. openml.org/d/161. [Accessed 12/2019] (pág. 154).

[Tad+21] K. Tadist et al. «SDPSO: Spark Distributed PSO-based approach for feature selection and cancer disease prognosis». *J. Big Data* (2021). DOI: 10.1186/s40537-021-00409-x (pág. 47).

[Tak+] N. Takahashi et al. «A Parallelized Data Stream Processing System Using Dynamic Time Warping Distance». *CISIS 2009*. DOI: 10.1109/CISIS.2009.77 (pág. 159).

[tea] K. team. *Kaggle - Datasets*. kaggle.com/datasets. [Accessed 12/2019] (págs. 86, 151).

[tra] D. M. C. 2. S.-d. track. *ECBDL14 - UCI repository*. cruncher.ico2s.org/bdcomp/. [Accessed 12/2019] (pág. 152).

[TGH11] I. Triguero et al. «Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification». *Pattern Recogn.* (2011). DOI: doi.org/10.1016/j.patcog.2010.10.020 (pág. 79).

[Tri+12] I. Triguero et al. «A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification». *IEEE Trans. Syst., Man, Cybern. C* (2012). DOI: 10.1109/TSMCC.2010.2103939 (págs. 40, 78).

[TGH15] I. Triguero et al. «Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study». *Knowl. Inf. Syst.* (2015). DOI: 10.1007/s10115-013-0706-y (pág. 22).

[Tri+15a] I. Triguero et al. «ROSEFW-RF». *Knowl-based. Syst.* (2015). DOI: 10.1016/j.knosys.2015.05.027 (pág. 63).

[Tri+15b] I. Triguero et al. «MRPR: A MapReduce solution for prototype reduction in big data classification». *Neurocomputing* (2015). DOI: 10.1016/j.neucom.2014.04.078 (págs. 78, 88).

[Tri+17] I. Triguero et al. «KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining:» *Int. J. Comput. Int. Sys.* (2017). DOI: 10.2991/ijcis.10.1.82 (pág. 30).

[Tri+19] I. Triguero et al. «Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data». *Data Min. Knowl. Disc.* (2019). DOI: 10.1002/widm.1289 (pág. 79).

[Tuf14] Z. Tufekci. «Big Questions for Social Media Big Data: Representativeness, Validity and Other Methodological Pitfalls». *arXiv:1403.7400* (2014). URL: arxiv.org/abs/1403.7400 (visitado 2021) (pág. 4).

[Tuk08] J. Tukey. «Exploratory Data Analysis». *The Concise Encyclopedia of Statistics*. 2008. DOI: 10.1007/978-0-387-32833-1_136 (pág. 17).

[Van] J. Vanschoren. *Click prediction dataset - OpenML repository*. openml.org/d/1218. [Accessed 12/2019] (pág. 152).

[Van+13] J. Vanschoren et al. «OpenML: Networked Science in Machine Learning». *SIGKDD Explor.* (2013). DOI: 10.1145/2641190.2641198 (págs. 86, 151).

[WF13] M. A. Waller y S. E. Fawcett. «Data Science, Predictive Analytics, and Big Data». *J. Bus. Logist.* (2013). DOI: 10.1111/jbl.12010 (pág. 47).

[WS09] K. Q. Weinberger y L. K. Saul. «Distance Metric Learning for Large Margin Nearest Neighbor Classification». *J. Mach. Learn. Res.* (2009). URL: jmlr.org/papers/v10/weinberger09a.html (pág. 24).

[Wei04] G. M. Weiss. «Mining with Rarity: A Unifying Framework». *ACM SIGKDD Explor. Newsl.* (2004). DOI: 10.1145/1007730.1007734 (pág. 14).

[Whi12] T. White. *Hadoop: the definitive guide*. Third edition. 2012 (pág. 52).

[Whia] D. Whiteson. *HEPMASS dataset - UCI repository*. archive.ics.uci.edu/ml/datasets/HEPMASS. [Accessed 12/2019] (pág. 153).

[Whib] D. Whiteson. *HIGGS dataset - UCI repository*. archive.ics.uci.edu/ml/datasets/HIGGS. [Accessed 12/2019] (pág. 153).

[Whic] D. Whiteson. *SUSY dataset - UCI repository*. archive.ics.uci.edu/ml/datasets/SUSY. [Accessed 12/2019] (pág. 154).

[WW21] K. Y. Wong y R. K. Wong. «Big data quality prediction informed by banking regulation». *Int.J.Data Sci.Anal.* (2021). DOI: 10.1007/s41060-021-00257-1 (pág. 47).

[Wu+14] X. Wu et al. «Data mining with big data». *IEEE Trans. Knowl. Data Eng.* (2014). DOI: 10.1109/TKDE.2013.109 (págs. 3, 45).

[WK09] X. Wu y V. Kumar, eds. *The top ten algorithms in data mining*. Ch. Crc. Data Min. Know. 2009 (pág. 25).

[Xin+14] Xindong Wu et al. «Data mining with big data». *IEEE Trans. Knowl. Data Eng.* (2014). DOI: 10.1109/TKDE.2013.109 (pág. 47).

[Xu+21] Z. Xu et al. «Data science: connotation, methods, technologies, and development». *Data Science and Management* (2021). DOI: 10.1016/j.dsm.2021.02.002 (pág. 47).

[Zah+16a] M. Zaharia et al. «Apache spark: A unified engine for big data processing». *Commun. Acm* (2016). DOI: 10.1145/2934664 (pág. 77).

[Zah+16b] M. Zaharia et al. «Apache Spark: a unified engine for big data processing». *Commun. Acm* (2016). DOI: 10.1145/2934664 (págs. 5, 53).

[Zar+15] P. ZareMoodi et al. «Novel Class Detection in Data Streams Using Local Patterns and Neighborhood Graph». *Neurocomput.* (2015). DOI: 10.1016/j.neucom.2015.01.037 (pág. 160).

[ZOT14] Y. Zhai et al. «The Emerging "Big Dimensionality"». *IEEE Comput. Intell. Mag.* (2014). DOI: 10.1109/MCI.2014.2326099 (pág. 15).

[ZY15] Y. Zhang e Y. Yang. «Cross-validation for selecting a model selection procedure». *J. Econometrics* (2015). DOI: 10.1016/j.jeconom.2015.02.006 (pág. 32).

[Zho12] Z.-H. Zhou. *Ensemble Methods: Foundations and Algorithms*. Ch. Crc. Mach. Learn. Pa. 2012 (pág. 27).

[ZG09] X. Zhu y A. B. Goldberg. «Introduction to Semi-Supervised Learning». *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2009). DOI: 10.2200/S00196ED1V01Y200906AIM006 (pág. 22).

[ZW04] X. Zhu y X. Wu. «Class Noise vs. Attribute Noise: A Quantitative Study». *Artif. Intell. Rev.* (2004). DOI: 10.1007/s10462-004-0751-8 (pág. 13).

[Zik12] P. Zikopoulos. *Understanding big data: analytics for enterprise class Hadoop and streaming data*. 2012 (pág. 47).

Apéndices

A Entorno experimental

En este apéndice se detalla el entorno experimental concerniente a nuestras propuestas. En el Apéndice A.1, se describen los problemas de clasificación binaria que representan cada uno de los conjuntos Big Data. Luego, en el Apéndice A.2, se detallan las infraestructuras computacionales empleadas para llevar a cabo los experimentos. A continuación, en el Apéndice A.3, se presenta, por un lado, las principales ventajas y características de Apache Spark mediante una comparación contra otra MapReduce. Por último, un compendio de todas las primitivas de Apache Spark utilizadas en los distintos desarrollos enmarcados en esta tesis se muestra en el Apéndice A.4.

A.1. Descripción de los conjuntos de datos

A continuación se describen los conjuntos de datos que se han utilizado en las distintas experimentaciones en el marco de esta tesis doctoral, excepto por los de EH que fueron descriptos en el Capítulo 9. Los mismos han sido obtenidos de diferentes repositorios de libre acceso (UCI machine learning [Lic13], OpenML [Van+13], Kaggle [tea]).

- `Agrawal` [Rij]. Estos datos se obtuvieron utilizando el generador Agrawal del framework MOA[42], denotado como AgrGen. Los datos son sobre una función de otorgamiento de préstamos, donde cada instancia pertenece a una clase de grupoA o grupoB.
- `airlines` [com]. Representa un problema de predicción de retraso de vuelos a partir de la información de aerolinea, vuelvo, aeropuerto de origen, aeropuerto de destino, dia de la semana, horario y distancia del vuelo.
- `BNG_Australian` [repb]. Este conjunto se refiere a solicitudes de tarjetas de crédito. Todos los nombres y valores de los atributos se han cambiado por símbolos sin sentido para proteger la confidencialidad de los datos.
- `BNG_heart` [repd]. Se trata de datos para predecir la presencia de enfermedades cardíacas en un paciente a partir de su edad, sexo, presión sanguínea en reposo, colesterol, entre otra información.

42 https://moa.cms.waikato.ac.nz/

- `census` [repc]. Este conjunto de datos contiene datos censales ponderados extraídos de las encuestas de población actuales de 1994 y 1995 realizadas por la Oficina del Censo de Estados Unidos.

- `click_prediction` [Van]. Los datos están relacionados con los anuncios que se muestran junto a los resultados de las búsquedas en un motor de búsqueda, y con el hecho de que la gente haya hecho clic en estos anuncios. La tarea consiste en predecir si un usuario hará clic en un anuncio determinado. Estos datos proceden del concurso KDDCup 2012, cuyo sitio web oficial ya no está disponible.

- `Covertype` [BDA]. Los datos se refieren a la predicción del tipo de cubierta forestal a partir de variables cartográficas únicamente. Este área de estudio incluye cuatro áreas silvestres situadas en el Bosque Nacional Roosevelt del norte de Colorado, Estados Unidos. Estas áreas representan bosques con mínimas perturbaciones causadas por el hombre, por lo que los tipos de cubierta forestal existentes son más el resultado de procesos ecológicos que de prácticas de gestión forestal. Este conjunto de datos es un problema de multiples clases, por lo que para convertirlo en binario, la clase de tipo de cubierta forestal Krummholz (representada por el número 7) es considera como una de las dos clases, y todo el resto conforman la otra. En este contexto, el conjunto de datos se denomina `covtype7`. Siguiendo la misma lógica, se tienen las variantes `covtype1`, `covtype1_vs_2` y `covtype2`.

- `creditCard` [DCB]. Contiene las transacciones realizadas con tarjetas de crédito en septiembre de 2013 por titulares europeos. Estas transacciones se produjeron en dos días, donde 492 son fraudes de 284.807 transacciones, lo que lleva a un problema de clasificación altamente no balanceado.

- `ECBDL14-10mill-90` [tra]. El conjunto de datos procede del campo de la predicción de estructuras proteicas, y se generó originalmente para entrenar un predictor para la pista de predicción de contactos residuo-residuo del concurso CASP9. Además, estos datos se utilizaron en la competición de ML del Evolutionary Computation for Big Data y Big Learning, en el marco de la conferencia internacional GECCO-2014. Se trata de un conjunto de datos de clasificación binaria altamente desequilibrado, compuesto por un 98 % de instancias negativas.

- `fars_fatal` [Adm]. El conjunto de datos FARS (Fatality Analysis Reporting System) es una recopilación de estadísticas sobre accidentes de tráfico realizada por el Centro Nacional de Estadísticas y Análisis de Estados

Unidos. El conjunto de datos contiene información sobre todas las personas implicadas en accidentes de tráfico en EE.UU. durante 2001, donde la mayoría de sus atributos se representan con valores nominales. El atributo de clase describe el nivel de lesión sufrido ("con lesión" o "sin lesión").

- `HEPMASS_IR_16` [Whia]. La búsqueda de partículas exóticas requiere clasificar un gran número de colisiones para encontrar los eventos de interés. Este conjunto de datos supone un reto para detectar una nueva partícula de masa desconocida. Representa un problema de clasificación binaria que busca discriminar entre eventos de señal y de fondo.

- `HIGGS` [Whib]. Este conjunto de datos trata de un problema de clasificación del campo de la física de partículas para distinguir entre un proceso de señal que produce bosones de Higgs y un proceso de fondo que no lo hace. Los datos se han producido utilizando simulaciones de Monte Carlo.

- `homeCredit` [repa]. Busca predecir la capacidad de un solicitante para la devolución de un crédito.

- `klaverjas` [RTV]. Este conjunto de datos contiene el valor teórico del juego, llamado Klaverjas, de casi un millón de configuraciones, dado el juego perfecto de ambos equipos. El objetivo es predecir si el equipo que empieza obtendrá más puntos que el otro.

- `MiniBooNE` [Roe]. Este conjunto de datos procede del experimento MiniBooNE del Fermilab[43] y se utiliza para distinguir los neutrinos electrónicos (señal) de los neutrinos muónicos (fondo).

- `poker` [CO]. Cada instancia de este conjunto de datos es un ejemplo de una mano de póker que consiste en cinco cartas extraídas de una baraja estándar de 52. Para tener un problema binario, se ha considerado sólo los casos en los que la etiqueta de clase es 0 (nada en la mano, una mano de póquer no reconocida) o 2 (dos pares de rangos iguales dentro de cinco cartas). Por esta razón, el conjunto de datos se denomina `poker0_vs_2` en este trabajo. Siguiendo el mismo procedimiento, se tiene la variante `poker0_vs_5`.

- `rlcp` [Nor]. Los registros representan datos individuales que incluyen el nombre y el apellido, el sexo, la fecha de nacimiento y el código postal, que se recogieron mediante inserciones iterativas en el transcurso de varios años. Los patrones de comparación de este conjunto de datos se basan en

43 https://www.fnal.gov/

una muestra de 100.000 registros que datan de 2005 a 2008. Los pares de datos se clasificaron como "coincidentes" o "no coincidentes" durante una extensa revisión manual en la que participaron varios documentalistas. La clasificación resultante constituyó la base para evaluar la calidad del procedimiento de vinculación de registros propios. La tarea consiste en decidir, a partir de un patrón de comparación, si los registros subyacentes pertenecen a una persona.

- sea_50 [SK]. Se trata de un conjunto de datos artificial que contiene una deriva conceptual abrupta. Cada instancia se describe mediante tres atributos y una etiqueta de clase. Estos datos y el procedimiento para generarlos se presentaron en [SK01].

- `skin` [BD]. El conjunto de datos de la piel se recoge mediante el muestreo aleatorio de los valores B,G,R de las imágenes faciales de varios grupos etarios (jóvenes, adultos y mayores), grupos étnicos (blancos, negros y asiáticos) y géneros obtenidos de la base de datos FERET y PAL. El problema busca predecir muestras de piel y muestras de no piel.

- SUSY [Whic]. Los datos se han producido utilizando simulaciones de Monte Carlo. Las primeras 8 características son propiedades cinemáticas medidas por los detectores de partículas en el Gran Colisionador de Hadrones (LHC), CERN. Las últimas diez características son funciones de las primeras 8 características; estas son características de alto nivel derivadas por los físicos para ayudar a discriminar entre un proceso de señal que produce partículas supersimétricas (SUSY) y un proceso de fondo que no [BSW14].

En la Tabla A.1 se muestra una descripción resumida de cada conjunto de datos ordenada alfabéticamente, la cual contiene el número de instancias (#Ex.), el número de atributos (#Atts.), el número de instancias de cada clase (#(maj; min)), el porcentaje de distribución de las clases (%(maj; min)), el número de cada tipo de características (Co/Di/Ca, donde Co = continuo, Di = discreto, Ca = categórico), y el tamaño del archivo (en MB).

Tabla A.1: Resumen de los conjuntos Big Data utilizados en las distintas experimentaciones, excepto por los de EH que se describen en la Sección 9.1.

Dataset	#Ex.	#Atts.	#(class0; class1)	%(class0; class1)	Co/Di/Ca	Size (MB)
Agrawal	1,000,000	9	(672,044; 327,955)	(67.2; 32.8)	4/2/3	71.6
airlines	539,383	7	(299,119; 240,264)	(55.46; 44.54)	4/0/3	18.3
BNG_Australian	1,000,000	14	(573,051; 426,949)	(57.31; 42.69)	14/0/0	80.8
BNG_heart	1,000,000	13	(555,946; 444,054)	(55.59; 44.41)	6/0/7	65.7
census	140,246	41	(132,085; 8162)	(94.18; 5.82)	1/12/28	68.7
click_prediction	1,963,972	11	(1,636,593; 327,379)	(83.33; 16.67)	0/11/0	136.3
covtype1	581,012	54	(369,307; 211,705)	(63.56; 36.44)	10/0/44	71.7
covtype1_vs_2	495,173	54	(283,468; 211,705)	(57.25; 42.75)	10/0/44	61.1
covtype2	581,012	54	(297,544; 283,468)	(51.21; 48.79)	10/0/44	71.7
covtype7	464,677	54	(448,421; 16,256)	(96.5; 3.5)	10/0/44	71.7
creditCard	284,015	30	(283,540; 480)	(99.83; 0.17)	28/1/0	145.2
ECBDL14-10mill-90	12,000,000	90	(11,760,000; 240,000)	(98; 2)	60/0/30	3481.6
ethylene_ECO_E_LH	4,208,261	16	(3,895,861; 312,400)	(92.58; 7.42)	16/0/0	517
ethylene_EM_E_LH	4,178,500	16	(3,840,313; 338,191)	(91.91; 8.09)	16/0/0	518.7
fars_fatal	100,968	29	(58,852; 42,116)	(58.29; 41.71)	0/1/28	59
HEPMASS_IR_16	5,578,255	28	(5,250,122; 328,133)	(94.12; 5.88)	28/0/0	1331.2
higgs	11,000,000	28	(5,827,686; 5,172,314)	(52.98; 47.02)	28/0/0	7680
HIGGS_IR_16	6,193,440	28	(5,829,120; 364,320)	(94.12; 5.88)	28/0/0	3378.7
homeCredit	307,511	171	(282,686; 24,825)	(91.93; 8.07)	1/9/161	113.1
hyperplane	1,000,000	10	(500,007; 499,993)	(50; 50)	10/0/0	91.4
klaverjas	981,541	34	(528,339; 453,202)	(53.83; 46.17)	2/32/0	79.2
MiniBooNE	129,590	49	(93,101; 36,489)	(71.84; 28.16)	49/0/0	67.7
poker0_vs_2	562,529	10	(513,701; 48,828)	(91.32; 8.68)	0/10/0	16.4
poker0_vs_5	412,600	10	((410,960; 1,640)	(99.6; 0.4)	0/10/0	16.4
rlcp	5,749,132	4	(5,728,201; 20,931)	(99.64; 0.36)	1/3/0	180.1
SEA_50	1,000,000	3	(614,342; 385,658)	(61.43; 38.57)	3/0/0	18.4
skin	245,057	3	(194,198; 50,858)	(79.25; 20.75)	0/4/0	3
susy	5,000,000	18	(2,712,173; 2,287,827)	(54.24; 45.76)	18/0/0	1740.8
SUSY_ir4	2,712,175	18	(2,169,740; 542,435)	(80; 20)	18/0/0	1022.5
SUSY_IR_16	2,881,684	18	(2,712,173; 169,511)	(94.12; 5.88)	18/0/0	1022.5

A.2. Infraestructura

Cluster Atlas de la UGR El clúster utilizado para todos los experimentos está compuesto por 14 nodos gestionados por un nodo maestro. Todos los nodos tienen la misma configuración de hardware y software. En cuanto al hardware, cada nodo cuenta con 2 CPU Intel Xeon E5-645, 6 núcleos (12 hilos) por procesador, 2,40 GHz y 64 GB de RAM. La red utilizada es Infiniband 40Gb/s. El sistema operativo es CentOS 6.9, con Apache Spark 2.2.0 y 2.3.0.

Cluster Hadoop de la UGR El clúster consta de 14 nodos conectados a través de una red Gigabit Ethernet. Cada nodo dispone de un microprocesador Intel Core i7-4930K a 3,40 GHz, 6 núcleos (12 hilos), 12 MB de caché, 4 TB HDD, y 64 GB de memoria principal trabajando bajo Linux CentOS 6.9. La infraestructura funciona con Hadoop 2.6.0 (Cloudera CDH5.8.0), donde el nodo principal está

configurado como NameNode y ResourceManager, y el resto son DataNodes y NodeManagers.

A.3. Hadoop MapReduce vs Apache Spark

En esta sección se presenta una comparación entre los dos *frameworks* de código abierto basados en MapReduce más utilizados: Apache Hadoop y Apache Spark. Como se comenta anteriormente en el Capítulo 5, principalmente la diferencia entre ellos radica en que el primero escribe intensamente en disco, mientras que el segundo lo hace en memoria. A pesar de la gran popularidad que alcanzó Hadoop en sus inicios, esta principal diferencia hace que Spark consiga un mejor rendimiento que Hadoop en la mayoría de los casos. También hay muchas librerías proporcionadas por Spark para diferentes propósitos que permiten que Spark se adapte a diversas aplicaciones.

De manera resumida, en la Tabla A.2 se presenta además una comparativa[44] de otros aspectos y capacidades de ambas herramientas, a partir de la cual se puede observar las otras ventajas que posiciona a Spark como una solución con mayor bondades que Hadoop.

Tabla A.2: Tabla comparativa de las características más relevantes de Hadoop MapReduce y Apache Spark.

Característica	Hadoop	Spark
Modo de procesamiento	Batch	Batch y Streaming
Modo de cómputo	Basado en disco	En memoria
Velocidad	Lento	Rápido
Cantidad de datos a procesar	Más	Menos
Costo	Alto	Bajo
Desempeño	Aceptable	Bueno
Lenguajes	Java, Python	Java, Scala, Python
Complejidad de uso	Más	Menos

A su vez, se muestran los códigos fuente de la resolución de una misma tarea que consiste en contar la repetición de palabras a partir de un archivo de entrada de texto (tarea popularmente conocida como *wordcount*), implementada utilizando cada uno de los frameworks (véase en Listados A.1 y A.2). Es notable la diferencia entre ambas resoluciones, en donde se destaca la forma sencilla y de pocas líneas de código que se requiere en Spark. Para el caso de la resolución con Hadoop, dada su extensión se ha disminuido el tamaño de fuente del mismo.

44 Ambas herramientas se encuentran en constante actualización, por lo que se mencionan las más generales al momento de la escritura de este apartado.

Listing A.1: *Wordcount* en Hadoop MapReduce.

```
// Hello.java
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

    public static class Map extends \mr{}Base implements Mapper<LongWritable, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws
              IOException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
          word.set(tokenizer.nextToken());
          output.collect(word, one);
        }
      }
    }

    public static class Reduce extends \mr{}Base implements Reducer<Text, IntWritable, Text, IntWritable> {
      public void reduce(Text key, Iterator values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws
              IOException {
        int sum = 0;
        while (values.hasNext()) {
          sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
      }
    }

    public static void main(String[] args) throws Exception {
      JobConf conf = new JobConf(WordCount.class);
      conf.setJobName("wordcount");

      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);

      conf.setMapperClass(Map.class);
      conf.setCombinerClass(Reduce.class);
      conf.setReducerClass(Reduce.class);

      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));

      JobClient.runJob(conf);
    }
}
```

Listing A.2: *Wordcount* en Apache Spark.

```
val file = spark.textFile("hdfs://...")

val counts = file.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)

counts.saveAsTextFile("hdfs://...")
```

A.4. Primitivas de Apache Spark

Una de las ideas en común para los diseños de nuestras propuestas era conseguir que el procesamiento de los datos se realice de forma eficiente y optimizada. Por lo tanto, el énfasis se ha puesto en la utilización de primitivas y utilidades[45] de Spark y de sus librerías, a lo largo de todo el flujo de trabajo de nuestras metodologías. Así pues, las primitivas y funcionalidades más relevantes empleadas en nuestros trabajos se presentan en la Tabla A.3. Es importante destacar que todas ellas son totalmente eficientes y robustas, y facilitan el propósito de hacer que nuestras propuestas sean totalmente escalables.

Tabla A.3: Lista de las primitivas y utilidades más relevantes de Apache Spark utilizadas en las propuestas de esta tesis.

Operación Spark	Descripción
textFile	Lee un archivo de texto desde un sistema de archivos, y lo retorna como un RDD de Strings.
MinMaxScaler	Utilidad para escalar los datos en un rango de valores.
repartition	Modifica el número de pariciones de una estructura de datos.
broadcast	Variable de sólo lectura en caché en cada nodo, en lugar de enviar una copia de la misma.
mapPartitionsWithIndex	Aplica una función a cada partición de la estructura de datos, manteniendo el índice de la partición original.
map	Realiza una transformación a cada elemento de una estructura de datos
filter	Selecciona todos los elementos de la estructura de datos que satisfacen un predicado
sample	Toma una muestra de un conjunto de datos
union	Combina dos estructuras de datos Spark por filas
kFold	Divide los datos en k pares de conjuntos de entrenamiento y validación, siguiendo un muestreo Bernoulli
dropDuplicates	Elimina las filas duplicadas de un dataframe
RandomForestClassifier *DecisionTreeClassifier*	Son los algoritmos de aprendizaje distribuidos para la clasificación de datos

45 https://spark.apache.org/docs/latest/

B Flujo de grandes volúmenes de datos

El procesamiento de flujos de datos (en inglés Stream Processing (SP)) es un área muy estudiada en los últimos años. SP permite llevar a cabo tareas mediante el análisis de un flujo continuo y potencialmente infinito de datos [DJ15; KDA19; SBA20]. El objetivo de SP es permitir que las tareas analicen los flujos de datos de forma online, brindando respuestas en tiempos muy cercanos al tiempo real. La principal característica de este tipo de procesamiento es que los datos del flujo llegan a una velocidad tal que no es posible almacenarlos en su totalidad y, si se pudieran almacenar, el volumen de datos sería tan grande que presentaría la dificultad de analizarlo en tiempos de respuesta cortos. En el Apéndice B.1 se comentan brevemente algunos aspectos más sobre SP. Luego, se presentan dos propuestas relacionadas al procesamiento escalable de flujos de datos en el contexto Big Data. En ambos casos se describe la idea general, omitiendo deliberadamente ciertos aspectos para facilitar la presentación, sin embargo los mismos pueden ser consultados directamente en los respectivos artículos (citados al final del capítulo). La primera propuesta tiene por objetivo aplicar una tarea de agrupamiento (clustering) sobre un flujo de datos de la red social Twitter, y se comenta en el Apéndice B.2. Por su parte, en el Apéndice B.3, se detalla la segunda propuesta que tiene aplicación en el campo de la econometría, y utiliza datos de transacciones de criptomonedas. El objetivo es calcular, en tiempo cortos de respuestas, un indicador ampliamente utilizado en el análisis del mercado para la detección de memoria a largo plazo, en concreto, el exponente de Hurst.

B.1. Trabajando con flujos de datos en Big Data

La mayor parte de las técnicas propuestas para el procesamiento de flujos de datos utilizan el modelo de tratamiento del flujo donde cada dato es procesado una única vez [Tak+; KA15; LS15; Ase+14]. Otras implementan una ventana temporal y deslizante donde almacenan los n últimos datos recibidos, o los n más representativos [Li+14; Mil+14; Myt+; Phr+14; Reh+14]. El primero de los enfoques tiene la gran desventaja de que si la distribución de los datos del flujo cambia a lo largo del tiempo es muy difícil armar un modelo de los datos que aprenda de las características nuevas y pasadas. Esto no ocurre en los modelos que utilizan una ventana temporal, ya que almacenan parte del flujo y todo el

tiempo pueden realimentar el modelo de datos con el último dato recolectado del *stream* más los *n* datos almacenados.

En algunos problemas el objetivo es justamente encontrar o adaptarse a los cambios de la distribución de los datos del flujo [Zar+15; Li+14] para lo cual los métodos del modelo simple de "recolectar-usar-descartar" un dato son útiles. En otras situaciones resulta interesante poder armar un modelo utilizando todos los datos del flujo. Como la mayoría de las veces es imposible debido al enorme volumen de información, es deseable obtener el modelo de los datos utilizando la mayor parte de éstos. En estos casos es imprescindible contar con técnicas que permitan el manejo de ventana temporal y que éstas sean lo más grande posible.

Cualquier algoritmo tradicional, ya sea de *clustering*, clasificación, regresión, aprendizaje utilizando redes neuronales, entre otros, para llevar a cabo su tarea y obtener resultados necesita realizar varias iteraciones sobre el conjunto completo de datos. Esto resulta en un desafío, ya que es prácticamente imposible garantizar una rápida respuesta frente a la situación donde se continúa colectando datos desde un flujo. Ya que finalizado el procesamiento de un lote de datos, se cuenta con mayor volumen de los mismos, cuyo procesamiento tomará aún más tiempo que el anterior mientras que se continúa con la recolección de datos. En esta situación, el sistema finalmente se satura. Para evitar el problema anterior existen dos alternativas. La primera consiste en manejar una ventana temporal del flujo mientras que la segunda intenta controlar la cantidad de iteraciones del algoritmo que realiza la tarea de extracción de conocimiento. La respuesta está en encontrar un equilibrio entre ambas alternativas ya que con ventanas temporales de gran tamaño los tiempos de respuesta serán grandes mientras que con ventanas pequeñas los tiempos de respuesta serán más inmediatos pero los resultados obtenidos estarán basados en una pequeña porción del flujo. Por lo tanto, resulta interesante contar con una técnica que permita el tratamiento de la mayor cantidad posible de datos y que los tiempos de respuestas sean cortos.

A su vez, existen dos tipos de modelos de procesamiento en SP, un dato a la vez y *micro-batching*. En el primer enfoque, los datos se procesan a medida que llegan y luego se descartan. Por otro lado, con el modelo de *micro-batching* el cómputo se lleva a cabo sobre pequeños lotes de datos capturados cada un determinado intervalo de tiempo. En la Sección 5.4 se introdujo Spark Streaming, la librería de Spark especializada para flujos de datos. Spark Streaming implementa el modelo de procesamiento de datos por *micro-batching*. La ingesta de los datos puede llevarse a cabo mediante diferentes fuentes de datos como puede ser Kafka, Flume, Twitter, ZeroMQ, Kinesis, o sockets TCP. Spark Streaming trabaja con los denominados *discretized stream (DStream)*, abstracciones de alto nivel de las estructuras de datos. Los *DStream* representan un flujo continuo de datos e, internamente, están representados por una secuencia de RDD. A su vez, con Spark

Streaming es posible configurar el tiempo de la ventana temporal al momento de recolectar los datos que serán utilizados en cada *micro-batch*.

B.2. Clustering de un flujo de datos

En esta sección se presenta una estrategia para utilizar una tarea de aprendizaje no supervisado sobre un flujo de datos en el contexto de Big Data. En concreto, aplicar el algoritmo de clustering *k-means* [Har75] sobre datos provenientes de la red social Twitter. Dentro de las características principales de esta propuesta, la misma permite manejar la cantidad de iteraciones de *k-means* manteniendo además el mayor tamaño posible del flujo disponible dentro de la ventana temporal. Para ello, se maneja el tamaño de la ventana en función de la frecuencia del flujo, la cantidad de datos almacenados para su procesamiento y el tiempo que le lleva a la tarea de clustering lograr el resultado parcial con el lote de datos actual. De esta manera, el algoritmo que realiza la tarea de extracción de conocimiento utiliza cada dato durante un número significativo de iteraciones.

La técnica propuesta consiste en dos tareas. La primera se basa en la captura del *stream* de datos y su correspondiente almacenamiento, que se ejecuta de manera continua. El *stream* es leído de forma *online*, y los datos recolectados se van guardando en un buffer el cual es almacenado en un archivo dentro del directorio de trabajo en el HDFS. El buffer no tiene límites lógicos y está organizado en archivos de tamaño 'b' datos; siendo 'b' un parámetro del algoritmo. La segunda tarea consiste en aplicar *k-means*. El número k de clusters a encontrar es determinado al comienzo del proceso y los centros iniciales son seleccionados al azar. *k-means* utiliza como entrada todos los archivos que están dentro del directorio de trabajo en el HDFS. Cuando el tamaño total de los mismos es muy grande, los archivos más antiguos son eliminados, reduciendo así la ventana de datos.

A medida que se colectan datos del flujo, estos son almacenados en un nuevo archivo. El problema de acumular archivos en el directorio es que, cuanto más datos haya, más tiempo se tardará en tratarlos. Es por ello que, a medida que se escriben nuevos datos se van eliminando los archivos más antiguos emulando así una ventana temporal sobre el flujo de datos.

La ejecución del trabajo distribuido comienza cuando el buffer del *stream* almacena los primeros b datos del flujo. En ese momento se guardan todos los datos acumulados en un archivo dentro de un directorio de trabajo en el HDFS y se lanza su ejecución. La estrategia propuesta realiza un proceso iterativo que consta de etapas. En cada etapa, el grupo de K centroides se pasa a un proceso que busca el centroide c más cercano, para cada vector de entrada v. Como salida, escribe tuplas (c, v) y, para aprovechar al máximo la capacidad del

cómputo paralelo son enviadas a k procesos, los cuales cada uno recibe todos los vectores correspondientes a un mismo *cluster* y con ellos calcula el nuevo centroide del mismo, que resulta de calcular el vector promedio entre todos los vectores recibidos. Cada uno de estos k procesos escribe como salida el nuevo centroide calculado además de todos los vectores que pertenecen al *cluster*. Esto último permite tener información de cómo están conformados los *clusters* al finalizar una etapa (esta información no es utilizada en este trabajo sin embargo es de utilidad para estudiar el propio resultado del *clustering*).

Las etapas se siguen ejecutando para ir refinando el procedimiento de *clustering* y de esa manera lograr que los centroides converjan. Este proceso iterativo continúa hasta que ocurra una de dos condiciones: (a) se alcanza un número máximo m de etapas iterativas; (b) el proceso de *clustering* alcanza un punto de equilibrio y los centroides no cambian significativamente de una iteración a otra, lo cual se define mediante un umbral de determinación de convergencia u.

Mientras se aplica la tarea de *clustering*, existe un proceso que continúa con la captura de datos llenando un nuevo buffer, hasta almacenar b datos de este buffer en un archivo nuevo. Esta condición existe para evitar que se escriban archivos con gran cantidad de datos y que la ventana sea reemplazada en su totalidad, logrando así un desplazamiento leve de la misma. Durante todo este proceso uno de los parámetros que se modifica de manera dinámica es el tamaño de la ventana que inicialmente es proporcional al tamaño del buffer con un factor de proporcionalidad llamado f. La ventana crece cuando el algoritmo logra la convergencia y, al mismo tiempo, la cantidad de datos en el buffer es menor que b. Esto significa que el algoritmo logra finalizar el modelo de los datos y, por lo tanto, se cuenta con la capacidad para tratar un volumen de datos mayor. En caso contrario, si el número de datos en el buffer es mayor que b y el algoritmo no alcanzó un estado de convergencia, se procede a la disminución del tamaño de la ventana. Para esto, el tamaño de la ventana se disminuye en b, eliminando del HDFS la cantidad de archivos necesaria. Cuando los datos en el buffer son más grandes que b, significa que el flujo de datos incrementó su velocidad, por lo tanto se aumenta el parámetro b para que sea más grande la cantidad de datos que se ingresan en la ventana, esto implica un corrimiento más abrupto de la misma.

En la Figura B.1 se muestra de manera general el procedimiento para el clustering de un flujo de datos.

La experimentación se llevó a cabo utilizando un flujo de datos proveniente de Twitter el cual posee 830745 *tweets*. Los textos de los *tweets* fueron procesados para convertirlos en vectores binarios de 90 dimensiones, donde cada elemento representa la ausencia o presencia de un tópico en particular. En todos los ensayos se utilizó la distancia del coseno [Ras92] como medida de similitud

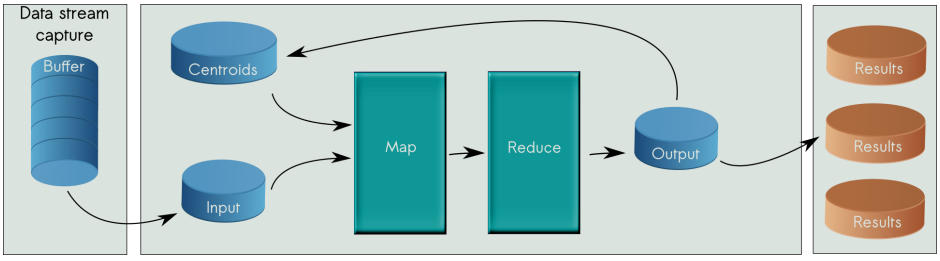


Figura B.1: Clustering de un flujo de datos.

para la ejecución del algoritmo *k-means* ya que es la más utilizada en estos tipos de problemas. Para obtener de manera empírica la combinación de los parámetros que mejor se adapta al flujo de datos analizados, se realizaron 720 pruebas modificando diferentes parámetros (tamaño del buffer (b), factor que determina el tamaño inicial de la ventana (f), factor de extensión del tamaño del buffer (e), iteraciones máximas (m), y umbral de convergencia (t).

Comentarios finales En este trabajo se presentó una técnica para el manejo de una ventana temporal de un flujo de datos, donde la característica es maximizar el tamaño de dicha ventana permitiendo que cada dato recolectado del flujo sea utilizado por el algoritmo de extracción de conocimiento la mayor cantidad de veces posibles. De los resultados obtenidos es posible apreciar que la técnica propuesta logra el manejo de ventanas de gran tamaño cumpliendo con los objetivos del trabajo. A su vez, es valido mencionar que, si bien la realización de los ensayos fue hecha con la ejecución de una tarea de *clustering* implementando una versión del algoritmo *k-means* en un entorno distribuido, es posible aplicar la técnica propuesta a cualquier tarea iterativa tan sólo con su correspondiente implementación adaptada a escenarios Big Data.

Los resultados presentados en este apartado forman parte de los siguientes artículos:

Basgall, M. J., Hasperué, W., & Naiouf, M. (2016). *Data stream treatment using sliding windows with MapReduce.* Journal of Computer Science & Technology, 16(2), 8. https://journal.info.unlp.edu.ar/JCST/article/view/498.

Basgall, M. J., Hasperué, W., Estrebou, C. & Naiouf, M. (2016). *Clustering de un flujo de datos usando MapReduce.* XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016). http://sedici.unlp.edu.ar/handle/10915/56748

B.3. Cálculo del exponente de Hurst: enfoque experimental sobre un flujo de transacciones de criptomonedas

En ciencias económicas un indicador que resulta muy útil es el de determinar la presencia de memoria o correlación a largo plazo en un mercado para saber si se pueden construir estrategias de negociación que permitan extraer un rendimiento superior. Una forma de realizar esa tarea es mediante el cálculo del exponente de Hurst [CCS04]. El poder asegurar que existe una correlación a largo plazo es un desafío al modelo financiero establecido, ya que de acuerdo a la teoría económica estándar, en un mercado competitivo los precios se debería mover aleatoriamente, reflejando una serie temporal sin memoria. Por ello, un avance natural en esta línea de investigación consiste en estudiar el comportamiento del exponente de Hurst en un entorno online.

Así pues, en esta sección se presenta el cálculo del exponente de Hurst para un flujo de datos simulado compuesto por transacciones de criptomonedas, utilizando el framework Spark Streaming. Este trabajo es a modo experimental ya que si bien es posible encontrar en internet bases de datos de transacciones de criptomonedas, al momento de este desarrollo no era posible obtener de manera online los datos de las transacciones que se necesitan para el cálculo del exponente de Hurst.

Respecto a las transacciones de criptomonedas, los datos públicamente disponibles[46] están almacenados como tripletas <TS, V, P> guardando el timestamp (TS), el volumen de comercio de criptomonedas (V) y el precio (P). Para el cálculo del exponente de Hurst solo se utiliza TS (segundos transcurridos desde las 0 horas del 1 de enero de 1970 GMT).

El flujo de datos compuesto por las tripletas se divide en ventanas temporales de la misma longitud (LV) y no solapadas, llamadas batchs o lotes. Cada batch, B_i, comienza en el tiempo ti_i y finaliza en el tiempo tf_i , siendo $LV = (tf_i - ti_i)$, además $ti_i = 1 + tf * (i - 1)$. Dentro de cada batch es posible que se obtenga del flujo una o más transacciones. De cada batch se extrae la transacción T_i donde su respectivo TS (TS_i) es el más cercano a tf_i, a partir de lo cual se calcula la distancia entre la hora de arribo de la transacción y el tiempo del fin del batch como $D_i = tf_i - TS_i$. A continuación, con los D_i se calcula el exponente de Hurst usando el método llamado Detrended Fluctuation Analysis (DFA) que es más apropiado cuando se trabaja con datos no estacionarios [CCS04]. Para esto, se calculan los $x(i)$ que representan una serie autosimilar de la serie original de

46 http://api.bitcoincharts.com/v1/csv/

precios como se muestra en Ecuación (B.1), donde $\overline{D}$ es la media aritmética de los $D_i, i = 1..M$.

$$x(i) = \sum_{t=1}^{i} [D_i - \overline{D}] \tag{B.1}$$

La serie de los $x(i)$ se divide en M/m submuestras y de cada submuestra se realiza un ajuste polinómico, que por lo general es un ajuste lineal, para obtener los $x_{fit}(i, m)$, donde m son las longitudes de intervalos dentro de la submuestra, para permitir el comportamiento fractal de la serie temporal.

La Ecuación (B.2) muestra la forma de calcular los coeficientes $F(m)$ utilizados para el cálculo del exponente de Hurst.

$$F(m) = \sqrt{(\frac{1}{M} * \sum_{i=1}^{M} [xi - x_{fit}(i, m)]^2)} \tag{B.2}$$

Finalmente, con los pares $(ln(m), ln(F(m)))$ se realiza una regresión lineal y la pendiente de dicha regresión es el exponente de Hurst.

Simulador de flujos Debido a las dificultades de tomar datos online de transacciones de criptomonedas en el mercado real, para el desarrollo de los diferentes experimentos se ha simulado un flujo de transacciones de criptomonedas. La simulación del flujo consiste en leer las transacciones desde un archivo y enviarlas una a una a un socket TCP, ya que Spark Streaming es capaz de procesar datos escuchando estos sockets.

Implementación en Spark streaming En primer lugar se configura el tamaño de ventana de cada micro-batch al tamaño deseado para el cálculo del exponente de Hurst. La longitud de la ventana no debe ser inferior a 500 observaciones, a fin de evitar sesgos en la estimación del exponente. Con esto nos garantizamos que Spark recolecte todas las transacciones del flujo que caen dentro de cada ventana asegurando que todas las ventanas sean del mismo tamaño, condición necesaria para el cálculo del exponente de Hurst. El script en bash generador del flujo, implementado y utilizado en los ensayos, genera alrededor de 1400 transacciones por segundo.

Mediante funciones de mapeo y reducción se obtiene la transacción TS_i que representa la transacción más cercana al tiempo de finalización del batch, utilizando para ello una clave única, ya que la siguiente acción a tomar es la reducción de todas las tuplas recolectadas en la ventana. Luego se genera, con la última transacción obtenida, una tupla usando el timestamp de la transacción como

clave y la distancia D_i como valor. Esta tupla se persiste ya que es necesario tener almacenado el "historial" de las últimas transacciones de cada batch del flujo procesado.

Cabe aclarar que las siguientes operaciones se realizan por cada una de las ventanas de datos analizadas, donde cada ventana agrega una nueva transacción al "historial" de transacciones. Así pues, utilizando la clase StatCounter provista por Apache Spark se calcula la media de las distancias D_i y se obtiene la menor de ellas, esta última es usada para calcular el número de lote (valor i) al cual pertenece la transacción obtenida en la última ventana.

Para el cálculo de los $x(i)$ primero se mapea la diferencia $D_i - \overline{D}$ usando el valor de i como clave. Luego se hace el producto cartesiano de lo anterior para crear una relación que permita calcular la sumatoria de las diferencias $D_i - \overline{D}$. Cada tupla resultante del producto cartesiano se mapea a un par clave valor y se aplica una reducción para sumar sus valores, y así obtener la serie $x(i)$.

Para el cálculo de la regresión, la serie $x(i)$ se mapea usando un par (m, l) como clave y $x(i)$ como valor, donde l es el número de submuestra al que pertenece cada valor $x(i)$. La idea es poder aprovechar al máximo la capacidad de Spark Streaming distribuyendo, por cada valor de m y submuestra l, el correspondiente cálculo de la regresión polinómica (lineal en este trabajo) y el cálculo de la serie $x_{fit}(i, m)$.

Como en Spark la reducción se hace por pares de valores no pudiendo acceder a todos los mismos en una misma operación, nos vimos obligados a hacer un mapeo extra para juntar todos los $x(i)$ en un único string y así poder acceder a todos los $x(i)$ pertenecientes a un mismo submuestreo para realizar la regresión lineal. Luego se usa la operación flatMap de Spark para generar los $x_{fit}(i, m)$. A su vez, se aprovecha la operación para devolver una tupla con el valor i como clave y como valor la diferencia $xi - x_{fit}(i, m)]^2$. Luego se hace un mapeo por valor de m y la reducción correspondiente para el cálculo de la sumatoria. Así se obtiene una tupla por cada valor de m utilizado.

A continuación, se generan pares de clave única y, m y $F(m)$ como valor. Dado que también se necesitan todos estos valores al mismo tiempo para el cálculo de la regresión lineal, dichos valores se convierten a string para concatenarlos y poder usarlos en la etapa siguiente. Finalmente, se recibe la única tupla producto de la operación anterior y con cada par $(m, F(m))$ recibido como valor se lleva a cabo la regresión lineal para el cálculo del exponente de Hurst.

Comentarios finales En este trabajo se presentó un algoritmo implementado en el framework Spark Streaming que obtiene de manera online, el cálculo del exponente de Hurst en un flujo simulado de transacciones de criptomonedas. Los resultados obtenidos muestran la eficiencia y la capacidad de Spark para

tratar con flujos cuya frecuencia es de 1400 transacciones por segundo pudiendo dar respuestas online usando ventanas de tamaño de 5 y 10 segundos.

Los resultados presentados en este apartado forman parte del siguiente artículo:

Basgall, M. J., Hasperué, W., Naiouf, M., & Bariviera, A. F. (2017). *Cálculo del exponente de Hurst utilizando Spark Streaming: Enfoque experimental sobre un flujo de transacciones de criptomonedas.* XXIII Congreso Argentino de Ciencias de la Computación, 10. http://sedici.unlp.edu.ar/handle/10915/63708

Agradezco enormemente…

a Marcelo y Alberto, por la oportunidad de trabajar con ellos y dirigirme en estos años;

a mis compañeros/as y amigos/as del III-LIDI y del CITIC;

a mi familia y amigos/as;

y, especialmente, a Joaco, por acompañarme siempre.