HEURÍSTICAS DE SELECCIÓN DE ATRIBUTOS PARA DATOS DE GRAN DIMENSIONALIDAD

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

Memoria de Tesis Doctoral para optar al grado de
Doctor en Informática por la Universidad de Sevilla
presentada por

D. Roberto Ruiz Sánchez

Directores:

Dr. D. José C. Riquelme Santos
Dr. D. Jesús S. Aguilar Ruiz

Sevilla, junio de 2006

Tesis Doctoral parcialmente subvencionada
por la Comisión Interministerial de Ciencia y Tecnología
con los proyectos TIN2004–00159 y TIN2004–06689–C03–03

D. José Cristóbal Riquelme Santos, Profesor Titular de Universidad, y D. Jesús Salvador Aguilar Ruiz, Profesor Titular de Universidad, ambos adscritos al área de Lenguajes y Sistemas Informáticos,

CERTIFICAN QUE

D. Roberto Ruiz Sánchez, Licenciado en Informática por la Universidad de Sevilla, ha realizado bajo nuestra supervisión el trabajo de investigación titulado:

HEURÍSTICAS DE SELECCIÓN DE ATRIBUTOS PARA DATOS DE GRAN DIMENSIONALIDAD

Una vez revisado, autorizan la presentación del mismo como Tesis Doctoral en la Universidad de Sevilla y estiman oportuna su presentación al tribunal que habrá de valorarlo. Igualmente, autorizan la presentación para obtener la acreditación de Doctorado Europeo.

Fdo. D. José C. Riquelme Santos
Profesor Titular de Universidad
Área de Lenguajes y Sistemas Informáticos

Fdo. D. Jesús Salvador Aguilar Ruiz
Profesor Titular de Universidad
Área de Lenguajes y Sistemas Informáticos

D. José Cristóbal Riquelme Santos, Profesor Titular de Universidad, y D. Jesús Salvador Aguilar Ruiz, Profesor Titular de Universidad, ambos adscritos al área de Lenguajes y Sistemas Informáticos, como directores de la tesis

HEURÍSTICAS DE SELECCIÓN DE ATRIBUTOS PARA DATOS DE GRAN DIMENSIONALIDAD
AUTOR: ROBERTO RUIZ SÁNCHEZ

proponen la siguiente composición del Tribunal titular, a fin de que la Comisión de Doctorado designe al Tribunal encargado de juzgar la tesis doctoral.

PRESIDENTE:	DR. D. MIGUEL TORO
VOCALES:	DR. D. FRANCISCO HERRERA
	DR. D. DANIEL RODRIGUEZ
	DR. D. FEDERICO DIVINA
SECRETARIO:	DR. D. RICARD GAVALDÁ
SUPLENTES:	DR. D. RAÚL GIRALDEZ
	DRA.D^{a}. ALICIA TRONCOSO

Fdo. D. José C. Riquelme Santos
Profesor Titular de Universidad
Área de Lenguajes y Sistemas Informáticos

Fdo. D. Jesús Salvador Aguilar Ruiz
Profesor Titular de Universidad
Área de Lenguajes y Sistemas Informáticos

A Tránsito,
Irene y David

Agradecimientos

Desde estas líneas quiero mostrar mi más sincero agradecimiento a todas aquellas personas que de alguna manera han contribuido, tanto en el plano profesional como en el personal, al desarrollo de esta tesis doctoral.

A los directores de esta tesis, José Riquelme y Jesús Aguilar, por la confianza y el apoyo incondicional que siempre he recibido. Ellos han hecho posible y han facilitado el camino de iniciación en el que hoy es mi trabajo, tanto en la docencia como en la investigación. Gracias por los consejos, por el ánimo y por estar siempre ahí.

También quiero agradecer a todos los miembros del grupo de investigación BIGS, Raúl, Paco, Alicia, Daniel, Domingo, Norberto, Isa y Juan Antonio, a los cuales en algún momento he recurrido para solicitar su consejo o ayuda. Y a todos mis compañeros del Departamento de Lenguajes y Sistemas Informáticos. A Rafael Carrasco y Juan por ayudarme siempre que los he necesitado (que no ha sido poco), y a mi colega Daniel Rodríguez de la Universidad de Reading. Con muchos de los cuales he compartido buenos momentos además de objetivos y horas de trabajo.

Esta memoria está dedicada a mis padres, hermana y familia política y, muy especialmente a Tránsito, que ha compartido el esfuerzo necesario para la preparación de esta tesis, y sin su paciencia y comprensión no habría sido posible. A todos ellos les pido perdón por el poco tiempo que les he dedicado.

Deseo resaltar que los responsables de que hoy en día pueda trabajar en algo que realmente me gusta y satisface han sido, por un lado quien me ofreció la oportunidad, Jesús Aguilar, al que admiro y aprecio; y por otro lado a Tránsito, mi mujer, quien me apoyó y animó a la hora de tomar semejante decisión, contando con su incompensable ayuda en todo momento.

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Resumen

Esta tesis doctoral se enmarca en el campo del aprendizaje automático, y aborda uno de sus principales problemas, como es el identificar un conjunto representativo de atributos para construir un modelo de clasificación.

El objetivo de este trabajo es proponer algoritmos de selección de atributos que sean capaces de actuar sobre bases de datos de muy alta dimensión, es decir, a partir de miles de atributos. Existen en la literatura múltiples propuestas para el problema conocido como *feature selection*. Sin embargo, la búsqueda de subconjuntos óptimos de atributos para la clasificación de conjuntos de datos presenta el inconveniente de su complejidad temporal. Por ello, la utilización del clasificador final como medida para evaluar la bondad del subconjunto de atributos seleccionado (conocida como evaluación *wrapper*) está limitada por el tamaño de la base de datos.

En las propuestas *BIRS* (*Best Incremental Ranked Subset*) y *BARS* (*Best Agglomerative Ranked Subset*), desarrolladas en este documento, se realiza una búsqueda guiada en el espacio de soluciones a partir de una ordenación inicial de los atributos. De esta manera, aunque evidentemente la búsqueda no es exhaustiva, reducimos lo suficiente este dominio para permitir una búsqueda eficiente y, por las pruebas realizadas, eficaz. Para la ordenación inicial, dada la elevada dimensión del destino final de estos algoritmos, se presenta el criterio de evaluación *SOAP* (*Selection Of Attributes by Projections*), un nuevo método más rápido que utiliza el número de cambios de etiqueta (o *NLC–Number of Label Changes*) como medida para la evaluación individual de atributos, calculada analizando las proyecciones de los elementos del conjunto de datos sobre cada dimensión o atributo.

Un campo de aplicación de este tipo de algoritmos es la bioinformática, y más concretamente el análisis de la información presente en los conjuntos de datos de expresión genética procedentes de experimentos con microarrays. Como es conocido, cualquier ser vivo dispone de miles de genes y existe una evidente relación entre la información genética de un individuo y ciertas enfermedades. Sin embargo, establecer cuáles son los genes concretos que determinan la diferencia entre distintos individuos es una tarea que necesita unos algoritmos eficientes.

Se presenta un exhaustivo conjunto de pruebas sobre datos reales que muestran la capaci-

dad de las técnicas propuestas para determinar un conjunto mínimo de atributos con un poder clasificatorio incluso superior a los datos originales. Este computo es realizado con un coste en tiempo menor que el de propuestas recientes, lo que permite su aplicación a datos de enorme dimensión como son los provinientes de microarrays.

Capítulo 1

Introducción

Ahora que hemos reunido tantos datos ¿qué hacemos con ellos?
U.M. FAYYAD.

1.1. Planteamiento

Esta pregunta es común en numerosas organizaciones. Con la revolución digital capturar información es fácil y almacenarla es extremadamente barato. Pero ¿para qué se almacenan los datos? Además de la facilidad y conveniencia, se almacenan datos porque se piensa que son un activo valioso en sí mismos. Para los científicos los datos representan observaciones cuidadosamente recogidas de algún fenómeno en estudio; en los negocios, los datos guardan informaciones sobre mercados, competidores y clientes; en procesos industriales recogen valores sobre el cumplimiento de objetivos; etc.

Sin embargo, en general, los datos en bruto raramente son provechosos. Su verdadero valor radica en la posibilidad de extraer información útil para la toma de decisiones o la exploración y comprensión de los fenómenos que dieron lugar a los datos. Tradicionalmente, el análisis de estos datos ha sido efectuado mediante técnicas estadísticas. Sin embargo, el incremento en la cantidad de datos y en el número de parámetros hace necesaria la aparición de nuevas metodologías y herramientas para un tratamiento automático de los registros depositados en bases de datos.

El proceso completo de extraer conocimiento a partir de bases de datos se conoce como $\mathcal{KDD}$ (*Knowledge Discovery in Databases*). Este proceso comprende diversas etapas que van desde la obtención de los datos hasta la aplicación del conocimiento adquirido en la toma de decisiones. Entre esas etapas se encuentra la que puede considerarse como el núcleo del proceso $\mathcal{KDD}$ denominada minería de datos ($\mathcal{DM}$–*data mining*). Esta fase es crucial para la obtención

de resultados apropiados, pues durante la misma se aplica el algoritmo de aprendizaje automático encargado de extraer el conocimiento inherente a los datos. No obstante, esta fase se ve influida en gran medida por la calidad de los datos que llegan para su análisis desde la fase previa.

Entre la gran cantidad de técnicas de preprocesado existente, en este trabajo se tratarán las relacionadas con la selección de atributos o características. Algunos problemas comunes a la mayoría de los algoritmos de aprendizaje automático cuando trabajan con muchos atributos se refieren a tiempos de ejecución muy elevados, pobre generalización o incomprensión de los resultados obtenidos. Por ello, desde la década de los sesenta, las investigaciones relacionadas con la selección de atributos intentan reducir el espacio de hipótesis de las bases de datos en tareas concretas, en un intento de encontrar subconjuntos de atributos que proporcionen un mejor rendimiento de los algoritmos de aprendizaje.

Existen numerosos algoritmos de selección de características en la bibliografía, donde se combinan técnicas de búsqueda y de evaluación de subconjuntos de atributos, llegando incluso a analizar todas las combinaciones posibles de atributos. Sin embargo, dado el escenario cambiante de las bases de datos donde ha aumentado tanto el número de filas como de columnas, la mayoría de estas técnicas no son aplicables. En este contexto, el nuevo marco de trabajo que se abre a las técnicas de selección de atributos es el punto de partida de esta investigación, con la necesidad de algoritmos capaces de obtener subconjuntos predictivos en grandes bases de datos.

1.2. Objetivos

Este trabajo de investigación tiene como objetivo el estudio y propuesta de un método de selección de atributos que se pueda aplicar a bases de datos de elevada dimensión en el marco del aprendizaje supervisado, en concreto para la tarea de clasificación.

La selección de atributos se puede considerar como un problema de búsqueda en un cierto espacio de estados, donde cada estado corresponde con un subconjunto de atributos, y el espacio engloba todos los posibles subconjuntos que se pueden generar. El proceso de selección de atributos puede entenderse como el recorrido de dicho espacio hasta encontrar un estado (combinación de atributos) que optimice alguna función definida sobre un conjunto de atributos. En general, un algoritmo de selección consta de dos componentes básicos: medida de evaluación y método de búsqueda. En este trabajo se hace una nueva propuesta para el primer componente y dos propuestas para el segundo, pudiéndose utilizar cada una independientemente en combinación con otras técnicas existentes.

Respecto a la evaluación de atributos, en el capítulo 4 se presenta una nueva medida para la selección de atributos, calculada analizando las proyecciones de los elementos del conjunto de datos sobre cada dimensión o atributo. La simplicidad y rapidez en la evaluación de los atributos son sus principales ventajas. La medida propuesta permite ordenar los atributos por orden de importancia en la determinación de la clase.

En la actualidad, el evaluar todas las combinaciones de atributos posibles de una base de datos es intratable, incluso aquellas con un número reducido de atributos, debido al coste computacional asociado a la operación. Por ejemplo, en una base de datos con veinte atributos, el número de evaluaciones posibles supera el millón, si se intentara aplicar a una base de datos con miles de características podría ser interminable. Este problema de eficiencia sirvió de impulso para plantear nuevas técnicas capaces de reducir la búsqueda en el espacio formado por el conjunto de los atributos, las cuales desembocaron finalmente en el desarrollo de dos heurísticas, denominadas *BIRS* (*Best Incremental Ranked Subset*) y *BARS* (*Best Agglomerative Ranked Subset*), descritas en el capítulo 5.

Para comprobar la bondad de la selección basada en la medida propuesta, se presentan los resultados de diversas comparativas con otros métodos de selección sobre un conjunto de bases de datos aplicando diferentes algoritmos de aprendizaje. Es importante señalar que las pruebas realizadas son totalmente reproducibles, ya que se eligieron para ello bases de datos accesibles muy conocidas en el área, incluidas en el *UCI Machine Learning Repository* y el *UCI KDD Archive* (tablas A.1 y A.2), todas ellas utilizadas por numerosos autores. Además, cinco bases de datos (tabla A.3) que fueron generadas para el congreso *NIPS* (*Neural Information Processing Systems*).

Actualmente, además de continuar profundizando en el desarrollo de la técnica de selección, se estudia su aplicación en bases de datos genómicas. La tecnología Microarray está fomentando el desarrollo de nuevos algoritmos de selección de atributos que, originalmente, se habían diseñado para bases de datos provenientes de otras fuentes. Ahora, con varios miles de genes, la extracción del mejor subconjunto de genes relevantes es un proceso computacionalmente inviable. La importancia de estos resultados es notoria cuando se trata de realizar diagnósticos de enfermedades graves, tales como el cáncer. El capítulo 6 describe esta aplicación práctica y presenta los resultados obtenidos hasta el momento.

1.3. Período de investigación

1.3.1. Aportaciones originales

Artículos publicados en revistas

- Projection-based measure for efficient feature selection. Journal of Intelligent and Fuzzy Systems, IOS Press Volume 12, Numbers 3-4, ISSN 1064-1246, 2002.

- Incremental wrapper-based gene selection from microarray expression data for cancer classification. Pattern Recognition, (aceptado, pendiente publicación). Elsevier Science, ISSN 0031-3203.

Artículos publicados en congresos internacionales

- Separation surfaces through genetic programming. Engineering of Intelligent Systems (IEA-AIE). Lecture Notes in Artificial Intelligence. ISBN: 3-540-42219-6, vol. 2070, Springer-Verlag. Budapest, Hungary, 2001.

- SNN: A supervised clustering algorithm. Engineering of Intelligent Systems (IEA-AIE). Lecture Notes in Artificial Intelligence. ISBN: 3-540-42219-6, vol. 2070, Springer-Verlag. Budapest, Hungary, 2001.

- SOAP: efficient feature selection of numeric attributes. VIII Iberoamerican Conference on Artificial Intelligence (IBERAMIA). Lecture Notes in Computer Science. ISSN 0302-9743, vol. 2527, Springer-Verlag. Sevilla, España, 2002.

- NLC: A measure based on projections. 14th International Conference on Database and Expert Systems Applications (DEXA). Lecture Notes in Computer Science. ISSN 0302-9743, vol. 2736, Springer-Verlag. Praga, República Checa, 2003.

- Fast Feature ranking algorithm. 7th International Conference on Knowledge-Based Intelligent Information and Engineering System (KES). Lecture Notes in Computer Science. ISSN 0302-9743, vol. 2773, Springer-Verlag. Oxford, Reino Unido, 2003.

- Fast feature selection by means of projections. 16th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems (IEA/AIE). Lecture Notes in Computer Science. ISSN 0302-9743, vol. 2718, Springer-Verlag. Loughborough, Reino Unido, 2003.

- Wrapper for ranking feature selection. Intelligent Data Engineering and Automated Learning (IDEAL), 5th Int. Conf. on Intelligent Data Engineering and Automated Learning. Lecture Notes in Computer Science, ISSN 0302-9743, vol. 3177, Springer-Verlag. Exeter, Reino Unido, 2004.

- Databases reduction. Sixth International Conference on Enterprise Information System (ICEIS). ISBN 972-8865-00-7. Oporto, Portugal, 2004.

- Analysis of feature rankings for classification. Advances in Intelligent Data Analysis VI (IDA), 6th International Symposium on Intelligent Data Analysis. Lecture Notes in Computer Science. ISSN 0302-9743, vol. 3646, Springer-Verlag. Madrid, España, 2005.

- Heuristic search over a ranking for feature selection. Computational Intelligence and Bioinspired Systems, 8th International Work-Conference on Artificial Neural Networks (IWANN). Lecture Notes in Computer Science, ISSN: 0302-9743, vol. 3512, Springer-Verlag. Barcelona, España, 2005.

- Segmentation of software engineering datasets using the M5 algorithm. International Conference in Computational Science (ICCS). Lecture Notes in Computer Science, ISSN: 0302-9743, vol. 3994, Springer-Verlag. Reading, Reino Unido, 2006.

- Gene ranking from microarray data for cancer classification–a machine learning approach. 10th International Conference on Knowledge-Based Intelligent Information and Engineering System (KES). (Aceptado, pendiente publicación) Lecture Notes in Computer Science. ISSN 0302-9743, Springer-Verlag. Bournemouth, Reino Unido, 2006.

Artículos publicados en congresos nacionales

- Reducción de bases de datos. VIII jornadas de Ingeniería del Software y Bases de Datos (JISBD). ISBN 84-688-3836-5. Alicante, España, 2003.

- Extracción de genes relevantes en bases de datos genómicas. IX jornadas de Ingeniería del Software y Bases de Datos (JISBD). ISBN 84-688-8983-0, Málaga, España, 2004.

- Evaluación de rankings de atributos para clasificación. Tendencias de la Minería de Datos en España, workshop de la red de minería. ISBN 84-688-8442-1, DigitalTres. Sevilla, España, 2004.

- Búsqueda secuencial de subconjuntos de atributos sobre un ranking. CEDI 2005: III Taller de Minería de Datos y Aprendizaje. ISBN 84-9732-449-8, Thomson. Granada, España, 2005.

1.3.2. Proyectos de investigación

Esta investigación ha sido parcialmente subvencionada por diferentes entidades, con los siguientes proyectos diferenciados según su vigencia:

Proyectos vigentes

- **Proyecto**: MINDAT-PLUS. Minería de datos para los usuarios en diferentes áreas de aplicación.
  - *Entidad que financia*: Junta de Andalucía
  - *Entidades participantes*: 10 grupos de universidades andaluzas
  - *Investigador principal*: F. Herrera (U. de Granada)
  - *Duración*: 2006–08
  - *Página web*: http://sci2s.ugr.es/MINDAT-PLUS/
- **Proyecto**: MINERVA. Técnicas emergentes de minería de datos para la extracción de conocimiento de grandes volúmenes de información: aplicación a datos científicos e industriales.
  - *Entidad que financia*: MEC
  - *Referencia*: TIN2004–00159
  - *Entidades participantes*: Universidad de Sevilla y Universidad de Huelva
  - *Investigador principal*: Dr. José C. Riquelme Santos (U. Sevilla)
  - *Duración*: 2005–07
  - *Página web*: http://www.lsi.us.es/minerva
- **Proyecto**: IN2MED. Taxonomías de modelos para la medición y evaluación de procesos software.
  - *Entidad que financia*: MEC
  - *Referencia*: TIN2004–06689–C03–03

  - *Entidades participantes*: Universidades de Sevilla, Huelva, Cádiz, País Vasco y Oviedo
  - *Investigador responsable*: Dra. Isabel Ramos Román (U. Sevilla)
  - *Duración*: 2005–07

- **Proyecto**: Red española de minería de datos y aprendizaje automático.
  - *Entidad que financia*: MEC
  - *Referencia*: TIN2004–21343–E
  - *Entidades participantes*: 25 grupos de 22 universidades españolas. 220 investigadores, 130 doctores
  - *Investigador coordinador*: Dr. José C. Riquelme Santos (U. Sevilla)
  - *Duración*: 2005–06
  - *Página web*: http://www.lsi.us.es/ riquelme/redmidas

- **Proyecto**: Knowledge discovery in data streams.
  - *Entidad que financia*: Acción integrada Hispano–Lusa
  - *Referencia*: HP2004–0044.
  - *Entidades participantes*: Universidad de Sevilla y Universidad de Oporto (Portugal)
  - *Investigador responsable*: Jesús S. Aguilar-Ruiz (LSI, Sevilla) y Joao Gama (LIACC, Porto)
  - *Duración*: 2005–06

- **Proyecto**: Sistemas Informáticos
  - *Entidad que financia*: Plan Andaluz de Investigación, Junta de Andalucía
  - *Entidades participantes*: Universidad de Sevilla y Universidad Pablo de Olavide
  - *Investigador coordinador*: Dr. José C. Riquelme Santos (U. Sevilla)
  - *Duración*: anual

Proyectos finalizados

- **Proyecto**: Knowledge discovery network of excellence.
  - *Entidad que financia*: Unión Europea.

  - *Referencia*: IST–2001–33086.
  - *Investigador coordinador*: Codrina Lauth (Instituto Fraunhofer)
  - *Duración*: 2001–05
  - *Página web*: http://www.kdnet.org

- **Proyecto**: Definición de diseño de un sistema de métricas para la valoración y estimación de proyectos de Ingeniería del Software.

  - *Proyecto Coordinado*: Mejora de los procesos para la toma de decisiones en la gestión de proyectos de Ingeniería del Software.
  - *Entidad que financia*: CICYT
  - *Referencia*: TIC2001–1143–C03–02
  - *Entidades participantes*: Universidad de Sevilla, Universidad de Huelva, Universidad de Cádiz y SADIEL S.A.
  - *Entidades participantes en el coordinado*: Universidad del País Vasco, Universidad de Sevilla y Universidad de Oviedo
  - *Investigador principal coordinador*: Dr. J. Javier Dolado Cosín (U. País Vasco)
  - *Investigador principal del subproyecto*: Dr. José C. Riquelme Santos (U. Sevilla)
  - *Duración*: 2002–04
  - *Página web*: http://www.sc.ehu.es/jiwdocoj/argo/argo.html

- **Proyecto**: Red española de minería de datos y aprendizaje automático.

  - *Entidad que financia*: CICYT
  - *Referencia*: TIC2002–11124–E
  - *Entidades participantes*: 17 grupos de 15 universidades españolas. Más de 100 investigadores, 50 de ellos doctores
  - *Investigador coordinador*: Dr. José C. Riquelme Santos (U. Sevilla)
  - *Duración*: 2003–04
  - *Página web*: http://www.lsi.us.es/redmidas

- **Proyecto**: Planificación de los sistemas de distribución y generación de energía eléctrica mediante técnicas de optimización y minería de datos.

- *Entidad que financia*: C. de Educación y Ciencia-Junta de Andalucia
- *Referencia*: ACC–1021–TIC–2002
- *Entidades participantes*: Departamento de Lenguajes y Sistemas Informáticos y Departamento de Ingeniería Eléctrica. Universidad de Sevilla
- *Investigador coordinador*: Dr. José C. Riquelme Santos (U. Sevilla)
- *Duración*: 2003–04

1.4. Organización

El contenido de esta memoria de investigación se encuentra dividido en los siguientes capítulos:

Capítulo 2: Aprendizaje automático supervisado. En este capítulo se presenta una visión general de este tipo de aprendizaje, haciendo mayor hincapié en la tarea de la clasificación. Asimismo, son descritos los clasificadores más utilizados y la forma de evaluar su rendimiento, así como la forma de representar los datos y su preparación antes de aplicar las técnicas de aprendizaje.

Capítulo 3: Selección de atributos. Se explica el estado del arte sobre los diversos aspectos teóricos que comprenden este estudio. Además, son catalogados la mayoría de los algoritmos conocidos, y descritos los más referenciados.

Capítulo 4: *SOAP*. La descripción detallada de nuestra primera propuesta principal, a la que hemos llamado *SOAP* (*Selection of attributes by projections*), se desarrolla en este capítulo, donde además se lleva a cabo las definiciones necesarias, la metodología para su evaluación y los experimentos necesarios para su validación. Además, se presentan distintas formas de comparar los rankings generados por algoritmos de selección de atributos, mostrando la diversidad de interpretaciones posibles en función del enfoque dado al estudio que se realice.

Capítulo 5: Búsqueda de subconjuntos en rankings de atributos. Se describe uno de los objetivos principales de este documento, la obtención de un subconjunto de atributos a partir de un ranking. Los métodos, denominados *BIRS* (*Best Incremental Ranked Subset*) y *BARS* (*Best Agglomerative Ranked Subset*), se basan en la idea de relevancia y redundancia, en el sentido de que un atributo (o conjunto) ordenado se escoge si añade información

al incluirlo en el subconjunto final de atributos elegidos. Estas heurísticas conducen a mejoras considerables en la exactitud obtenida, en comparación con el conjunto completo y frente a otros algoritmos de selección de atributos, junto a una notable reducción en la dimensionalidad.

Capítulo 6: Extracción de genes relevantes en bases de datos genómicas. Se aplican las técnicas basadas en un ranking preliminar, presentadas en el capítulo anterior, para selección de genes relevantes. En los experimentos se han utilizado varias bases de datos reales, obteniendo resultados de clasificación mejores con los genes seleccionados que con todo el conjunto inicial.

Conclusiones y Trabajos Futuros. Finalmente, se resumen las conclusiones obtenidas durante esta investigación, las cuales nos impulsan a considerar ciertas alternativas para obtener mayor rendimiento en propuestas futuras.

Apéndice A: Bases de datos. Describe las bases de datos utilizadas en los experimentos realizados.

Capítulo 2

Aprendizaje automático supervisado

Los grandes avances en las tecnologías de la información han provocado un aumento sin precedentes tanto en el volumen de datos como en su flujo. Códigos de barra, tarjetas electrónicas, sensores remotos, transacciones bancarias, satélites espaciales o el descifrado del código genético humano son ejemplos de la necesidad de filtrar, analizar e interpretar volúmenes de datos que se miden en terabytes.

A continuación se introduce el término *aprendizaje automático supervisado* en relación con los de minería de datos y extracción de conocimientos. En las siguientes secciones se tratan con detalles aspectos como la representación de los datos, la clasificación, la evaluación del rendimiento de un clasificador y la descripción de la preparación de los datos para la minería.

2.1. Introducción

Desde hace más de dos décadas se vienen desarrollando y utilizando complejos algoritmos para la extracción de patrones útiles en grandes conjuntos de datos. Gran parte de esta información representa transacciones o situaciones que se han producido, siendo útil no sólo para explicar el pasado, sino para entender el presente y predecir la información futura. En muchas ocasiones, el método tradicional de convertir los datos en conocimiento consiste en un análisis e interpretación realizada de forma manual por especialistas en la materia estudiada. Esta forma de actuar es lenta, cara y altamente subjetiva. De hecho, la enorme cantidad de datos desborda la capacidad humana de comprenderlos y el análisis manual hace que las decisiones se tomen según la intuición de los especialistas.

A finales de la década de los 80, la creciente necesidad de automatizar todo este proceso inductivo abre una línea de investigación para el análisis inteligente de datos. Al conjunto de métodos matemáticos y técnicas software para análisis inteligente de datos y búsqueda de regu-

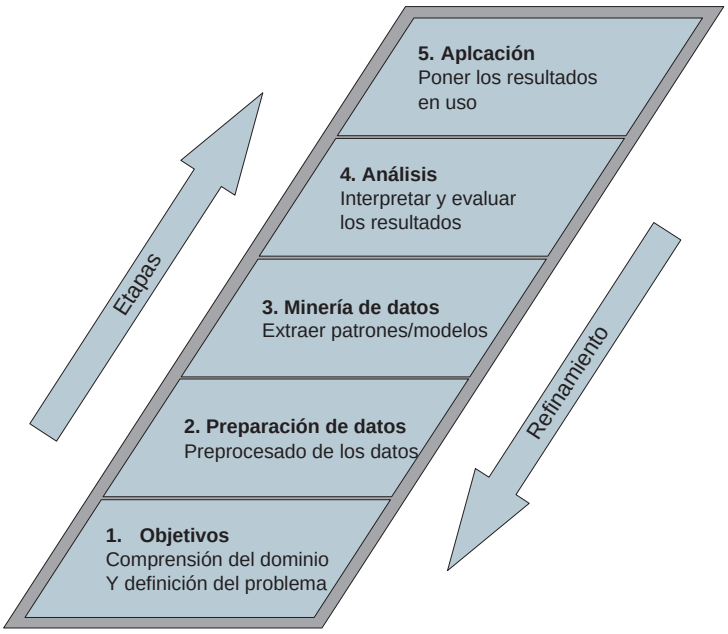


Figura 2.1: Esquema General de $\mathcal{KDD}$ (Knowledge Discovery in Databases).

laridades y tendencias en los mismos, aplicados de forma iterativa e interactiva, se denominaron técnicas de *minería de datos o data mining* ($\mathcal{DM}$). Su nombre proviene de las similitudes encontradas entre buscar valiosa información de negocio en grandes bases de datos y minar una montaña para encontrar una veta de metales valiosos. Su tarea fundamental es la de encontrar modelos inteligibles a partir de los datos, y para que el proceso sea efectivo debe ser automático, generando patrones que ayuden en la toma de decisiones beneficiosas para la organización.

Para obtener conclusiones válidas y útiles al aplicar minería de datos, es necesario complementar este proceso con una adecuada preparación de los datos previa al proceso de minería y un análisis posterior de los resultados obtenidos. Así, podemos afirmar que el proceso de minería de datos pertenece a un esquema más amplio, reflejado en la figura 2.1, denominado extracción o *descubrimiento de conocimiento en bases de datos* ($\mathcal{KDD}$, *Knowledge Discovery in Databases*). Se puede intuir que el $\mathcal{KDD}$ es un campo multidisciplinar, donde las principales áreas contribuyentes son el aprendizaje automático, las bases de datos y la estadística.

El aprendizaje automático es el área de la Inteligencia Artificial que se ocupa de desarrollar técnicas capaces de aprender, es decir, extraer de forma automática conocimiento subyacente en la información. Constituye junto con la estadística el corazón del análisis inteligente de los datos. Los principios seguidos en el aprendizaje automático y en la minería de datos son los mismos: la máquina genera un modelo a partir de ejemplos y lo usa para resolver el problema. Uno de los modelos de aprendizaje más estudiado es el *aprendizaje inductivo*, que engloba todas aquellas técnicas que aplican inferencias inductivas sobre un conjunto de datos para adquirir

conocimiento inherente a ellos.

En general, las tareas de un proceso de aprendizaje, al igual que las de uno de minería, pueden ser clasificadas en dos categorías: descriptivas y predictivas. Las primeras describen el conjunto de datos de una manera resumida y concisa, presentando propiedades generales e interesantes de los datos. Por contra las tareas predictivas construyen uno o varios modelos que realizan inferencia sobre el conjunto de entrenamiento para intentar predecir el comportamiento de nuevos datos. La variedad de términos existente en la literatura especializada para nombrar las tareas y las técnicas utilizadas para desempeñar cada una de ellas, hace que, a veces, sea difícil aclararse con los nombres. Se suele utilizar el término *aprendizaje supervisado* para los métodos predictivos y *aprendizaje no supervisado* para el resto. En esta memoria, el aprendizaje siempre será entendido como supervisado, donde los casos pertenecientes al conjunto de datos tienen a priori asignada una clase o categoría, siendo el objetivo encontrar patrones o tendencias de los casos pertenecientes a una misma clase.

Esta investigación se centra principalmente en la segunda etapa del proceso del $\mathcal{KDD}$, es decir en la preparación de datos, y más concretamente en la selección de características. Aunque este trabajo no abarca otras fases del proceso, se resalta la orientación de los atributos seleccionados hacia métodos de aprendizaje supervisado.

2.2. Representación de los datos

A continuación se establecerán algunas definiciones que describen formalmente los conceptos que se manejarán a lo largo de este documento.

Definición 2.1 *Un **dominio** es un conjunto de valores del mismo tipo. Aunque existen distintas clasificaciones de los dominios, para los propósitos de esta investigación se distinguen dos tipos: continuo (conjunto infinito de valores reales) y nominal (conjunto finito de valores discretos). Se representa Dom().*

Definición 2.2 *Se denomina **Universo de discurso** al entorno donde se define un determinado problema y viene representado como el producto cartesiano de un conjunto finito de dominios.*

Definición 2.3 *Un **atributo**, o también denominado **característica**, es la descripción de alguna medida existente en el universo de discurso que toma valores en un determinado dominio. El atributo i–ésimo se representa X_i, su valor x_i y su dominio como $Dom(X_i)$, que según la clasificación descrita previamente puede ser de dos tipos, continuo o discreto. Si es continuo existe un rango $[a, b] \subseteq \mathbb{R}$ de valores posibles, y si es discreto existe un conjunto finito de*

valores posibles. Se denomina vector atributos $\overline{x} = x_1, \ldots, x_n$ *al conjunto de valores correspondiente a cada uno de los atributos, y* $\mathcal{X}$ *al espacio formado por el conjunto de los atributos,* $\mathcal{X} = Dom(X_1) \times \ldots \times Dom(X_n)$*, siendo n el total de atributos.*

Definición 2.4 *En el marco del aprendizaje supervisado, se dispone de un atributo especial de salida denominado* ***clase****, que indica la pertenencia a un determinado grupo de casos. Se denomina* ***etiquetas*** *de clase al conjunto o dominio de valores que la clase puede tomar (nominal en el caso de la clasificación). La clase es el atributo sobre el cual se realiza la predicción, por lo que es también denominada* atributo de decisión*, para diferenciarla del resto de atributos denominados* de condición*. El atributo clase se representa Y y su dominio Dom(Y), teniendo k valores posibles* $y_1, \ldots, y_k$*.*

Definición 2.5 *Un* ***ejemplo****,* ***muestra****,* ***instancia*** *o* ***caso*** *es una tupla del universo de discurso representada por un conjunto de valores de atributos, cada uno de un dominio respectivamente, y una etiqueta de clase que lo clasifica. El ejemplo se representa e.*

Definición 2.6 *Se define un* ***conjunto de datos*** *como un subconjunto finito de ejemplos* e_j*, donde* $j = 1, \ldots, m$*. Un conjunto de datos, o base de datos, se caracteriza por el número de ejemplos m que contiene y por el número n de atributos y su tipo.*
La entrada a un algoritmo de aprendizaje supervisado es un conjunto $\mathcal{E}$ *de m instancias* $(\overline{x_j}, y_j)$*, donde* $j = 1, \ldots, m$*, cada una compuesta por n valores de entrada* $x_{j,i}$ *con* $(i = 1, \ldots, n)$ *y uno de salida* y_j*, a* $\mathcal{E}$ *se le llama conjunto de datos etiquetado.*

2.3. Clasificación

Aprender cómo clasificar objetos a una de las categorías o clases previamente establecidas, es una característica de la inteligencia de máximo interés para investigadores tanto de psicología como de informática, dado que la habilidad de realizar una clasificación y de aprender a clasificar, otorga el poder de tomar decisiones.

Definición 2.7 *Sea* $\mathcal{E}$ *un conjunto de datos, el objetivo de la* ***clasificación*** *es aprender una función* $\mathcal{L} : \mathcal{X} \rightarrow Y$*, denominada* ***clasificador****, que represente la correspondencia existente en los ejemplos entre los vectores de entrada y el valor de salida correspondiente, es decir, para cada valor de* $\overline{x}$ *tenemos un único valor de Y.*

Además, Y es nominal, es decir, puede tomar un conjunto de valores $y_1, y_2, \ldots, y_k$ denominados clases o etiquetas. La función aprendida será capaz de determinar la clase para cada

nuevo ejemplo sin etiquetar. Sin lugar a dudas, el éxito de un algoritmo de aprendizaje para clasificación depende en gran medida de la calidad de los datos que se le proporciona.

La aplicación de un algoritmo de aprendizaje tiene como objetivo extraer conocimiento de un conjunto de datos y modelar dicho conocimiento para su posterior aplicación en la toma de decisiones. Existen distintas formas de representar el modelo generado, representación proposicional, árboles de decisión, reglas de decisión, listas de decisión, reglas con excepciones, reglas jerárquicas de decisión, reglas difusas y probabilidades, redes neuronales, están entre las estructuras más utilizadas. Sin embargo, dado que este trabajo se enmarca dentro de la preparación de datos, no se entrará en detalle en este área.

En este trabajo, se utilizarán tres algoritmos de aprendizaje clasificadores para comparar los efectos de la selección de atributos, uno probabilístico (Naïve Bayes), otro basado en las técnicas de vecinos más cercanos (ib1) y un tercero basado en árboles de decisión (C4.5). Los algoritmos de aprendizaje empleados se han elegido por ser representativos de diferentes tipos de clasificadores, usandose con frecuencia en los estudios comparativos y en bastantes aplicaciones de minería [117, 103].

2.3.1. Naïve Bayes

Naïve Bayes es una técnica de clasificación descriptiva y predictiva basada en la teoría de la probabilidad del análisis de T. Bayes [14], que data de 1763. Esta teoría supone un tamaño de la muestra asintóticamente infinito e independencia estadística entre variables independientes, refiriéndose en nuestro caso a los atributos, no a la clase. Con estas condiciones, se puede calcular las distribuciones de probabilidad de cada clase para establecer la relación entre los atributos (variables independientes) y la clase (variable dependiente). Concretamente, dado el ejemplo $e = (x_1, \ldots, x_n)$, donde x_i es el valor observado para el i-ésimo atributo, la probabilidad a posteriori de que ocurra la clase y_l teniendo k valores posibles $\{y_1, \ldots, y_k\}$, viene dada por la regla de Bayes,

$$P(y_l|x_1, \ldots, x_n) = \frac{P(y_l) \prod_{i=1}^{n} P(x_i|y_l)}{P(x_1, \ldots, x_n)} \tag{2.1}$$

donde $P(y_l)$ es la proporción de la clase y_l en el conjunto de datos; e igualmente, $P(x_i|y_l)$ se estima a partir de la proporción de ejemplos con valor x_i cuya clase es y_l. Como podemos deducir, el cálculo de $P(x_i|y_l)$ obliga a que los valores x_i sean discretos, por lo que si existe algún atributo continuo, éste debe ser discretizado previamente. Aplicando la ecuación 2.1, la clasificación de un nuevo ejemplo e se lleva a cabo calculando las probabilidades condicionadas de cada clase y escogiendo aquella con mayor probabilidad. Formalmente, si $Dom(Y) = \{y_1, \ldots, y_k\}$ es el conjunto de clases existentes, el ejemplo e será clasificado con aquella clase y_l que satisface la

expresión 2.2.

$$\forall j \neq i \ / \ P(y_i|x_1, \ldots, x_n) > P(y_j|x_1, \ldots, x_n) \tag{2.2}$$

Como se puede observar, el clasificador bayesiano es un método sencillo y rápido. Además, puede demostrarse teóricamente que maximiza la exactitud de la predicción de manera óptima. Sin embargo, la suposición de independencia estadística de las variables es una limitación importante, ya que este hecho es relativamente infrecuente.

2.3.2. Vecinos más cercanos

Las técnicas de vecinos más cercanos (*NN, Nearest Neighbours*) basan su criterio de aprendizaje en la hipótesis de que los miembros de una población suelen compartir propiedades y características con los individuos que los rodean, de modo que es posible obtener información descriptiva de un individuo mediante la observación de sus vecinos más cercanos. Los fundamentos de la clasificación por vecindad fueron establecidos por E. Fix y J. L. Hodges [59, 60] a principio de los años 50. Sin embargo, no fue hasta 1967 cuando T. M. Cover y P. E. Hart [33] enuncian formalmente la regla del vecino más cercano y la desarrollan como herramienta de clasificación de patrones. Desde entonces, este algoritmo se ha convertido en uno de los métodos de clasificación más usados [31, 32, 35, 53, 5]. La regla de clasificación *NN* se resume básicamente en el siguiente enunciado:

Sea $\mathcal{E} = \{e_1, \ldots, e_m\}$ un conjunto de datos con m ejemplos etiquetados, donde cada ejemplo e_j contiene n atributos $(x_{j1}, \ldots, x_{jn})$, pertenecientes al espacio métrico $\mathcal{X}$, y una clase $y_l \in \{y_1, \ldots, y_k\}$. La clasificación de un nuevo ejemplo e' cumple que

$$e' \dashv y_l \Leftrightarrow \forall j \neq i \ / \ d(e', e_i) < d(e', e_j) \tag{2.3}$$

donde $e' \dashv y_l$ indica la asignación de la etiqueta de clase y_l correspondiente a e_i al ejemplo e' y d expresa una distancia definida en el espacio n-dimensional $\mathcal{X}$.

Así, un ejemplo es etiquetado con la clase de su vecino más cercano según la métrica definida por la distancia d. La elección de esta métrica es primordial, ya que determina qué significa más cercano. La aplicación de métricas distintas sobre un mismo conjunto de entrenamiento puede producir resultados diferentes. Sin embargo, no existe una definición previa que indique si una métrica es buena o no. Esto implica que es el experto quien debe seleccionar la medida de distancia más adecuada. La regla *NN* puede generalizarse calculando los k vecinos más cercanos y asignando la clase mayoritaria entre esos vecinos. Tal generalización se denomina *k–NN*. Este algoritmo necesita la especificación a priori de k, que determina el número de vecinos que

se tendrán en cuenta para la predicción. Al igual que la métrica, la selección de un k adecuado es un aspecto determinante. El problema de la elección del k ha sido ampliamente estudiado en la bibliografía. Existen diversos métodos para la estimación de k [45]. Otros autores [54] han abordado el problema incorporando pesos a los distintos vecinos para mitigar los efectos de la elección de un k inadecuado. Otras alternativas [137] intentan determinar el comportamiento de k en el espacio de características para obtener un patrón que determine a priori cuál es el número de vecinos más adecuado para clasificar un ejemplo concreto dependiendo de los valores de sus atributos. El algoritmo *k–NN* se engloba dentro de las denominadas técnicas de aprendizaje perezoso (*lazy learning*), ya que no genera una estructura de conocimiento que modele la información inherente del conjunto de entrenamiento, sino que el propio conjunto de datos representa el modelo. Cada vez que se necesita clasificar un nuevo ejemplo, el algoritmo recorre el conjunto de entrenamiento para obtener los k vecinos y predecir su clase. Esto hace que el algoritmo sea computacionalmente costoso tanto en tiempo, ya que necesita recorrer los ejemplos en cada predicción, como en espacio, por la necesidad de mantener almacenado todo el conjunto de entrenamiento. Pese a los numerosos inconvenientes respecto a la eficiencia (coste computacional) y la eficacia (elección de la métrica y el k adecuados), *k–NN* tiene en general un buen comportamiento. Cover y Hart [33] demostraron que, cuando el número de ejemplos tiende a infinito, el error asintótico de *NN* está acotado superiormente por el doble del error de Bayes (óptimo).

2.3.3. Árboles de decisión

Los árboles de decisión, son una de las formas más sencillas de representación del conocimiento adquirido. Dentro de los sistemas basados en árboles de decisión, habitualmente denominados *TDIDT* (*Top Down Induction of Decision Trees*), se pueden destacar dos familias o grupos: la familia *ID3*, cuyos máximos representantes son el propio algoritmo *ID3* propuesto por Quinlan [132] y el sistema CLS de Hunt et al. [75]; y la familia de árboles de regresión, cuyo exponente más significativo es CART, desarrollado por Breiman et al. [23]. Los *TDIDT* se caracterizan por utilizar una estrategia de divide y vencerás descendente, es decir, partiendo de los descriptores hacia los ejemplos, dividen el conjunto de datos en subconjuntos siguiendo un determinado criterio de división. A medida que el algoritmo avanza, el árbol crece y los subconjuntos de ejemplos son menos numerosos. *ID3* puede considerarse como una versión preliminar de *C4.5*, el cual resuelve algunos inconvenientes de su antecesor sobre el uso de atributos continuos, el tratamiento de valores ausentes y el proceso de poda. De los sistemas *TDIDT*, los pertenecientes a la familia *ID3* son los más referenciados en el campo del aprendizaje, por lo que serán expuestos con más detalle a continuación.

ID3

El método de clasificación experimental ID3 (*Induction Decision Trees*), desarrollado por J. R. Quinlan [130, 131, 132], genera un árbol de decisión paralelo de forma recursiva, aplicando un criterio de división basado en el concepto de medida de la información de Shannon. Cada nodo interno de dicho árbol contiene un test sobre uno de los atributos, de cuyo valor dependerá el camino a seguir para clasificar un ejemplo, y cada hoja contiene una etiqueta de clase. Así, la clasificación de un ejemplo se lleva a cabo recorriendo el árbol desde la raíz hasta una de las hojas que determinará la clase del mismo. Inicialmente, el algoritmo toma todo el conjunto de datos $\mathcal{E}$. Si todos los ejemplos pertenecen a una misma clase, el proceso finaliza, insertando un nodo hoja con dicha clase. En caso contrario, se selecciona aquel atributo X_i que mejor divide el conjunto de datos y se inserta un nodo con dicho atributo para establecer un test. Una vez creado el nodo, para cada valor distinto x_{iv} del atributo X_i, se traza un arco y se invoca recursivamente al algoritmo para generar el subárbol que clasifica los ejemplos de $\mathcal{E}$ que cumplen que $X_i = x_{iv}$. Dicha invocación es realizada sin tener en cuenta el atributo X_i y substrayendo del conjunto de datos $\mathcal{E}$ todos aquellos ejemplos donde $X_i \neq x_{iv}$. El proceso se detiene cuando todas las instancias de un conjunto pertenecen a la misma clase. *ID3* utiliza una propiedad estadística denominada *ganancia de información* como heurística de selección de atributos para fijar un test. Esta propiedad no es más que la reducción esperada de la entropía (desorden) de los datos al conocer el valor de un atributo. Así, el atributo X_i seleccionado para determinar la división será aquel que mayor ganancia obtenga respecto al conjunto $\mathcal{E}$, según la ecuación 2.4,

$$Ganancia(\mathcal{E}, X_i) = Ent(\mathcal{E}) - \sum_{v=1}^{|X_i|} \frac{|\mathcal{E}(x_{iv})|}{|\mathcal{E}|} \times Ent(\mathcal{E}(x_{iv})) \tag{2.4}$$

donde $|X_i|$ es el número de valores distintos de del atributo X_i; $\mathcal{E}(x_{iv})$ es el subconjunto de $\mathcal{E}$ para el cual $X_i = x_{iv}$, siendo $|\mathcal{E}(x_{iv})|$ su cardinal; $|\mathcal{E}|$ es el número total de ejemplos; y $Ent(\cdot)$ es la entropía, definida a continuación.

Definición 2.8 (Entropía) *La entropía es la medida del desorden de un sistema mediante la incertidumbre existente ante un conjunto de casos, del cual se espera uno sólo. Sea $\mathcal{E}$ un conjunto de datos etiquetados con clases del conjunto $Dom(Y) = \{y_1, \ldots, y_k\}$ y $frec(y_l, \mathcal{E})$ el número de ejemplos de $\mathcal{E}$ con clase y_l. Entonces se define la entropía del conjunto $\mathcal{E}$ como*

$$Ent(\mathcal{E}) = -\sum_{l=1}^{k} \frac{frec(y_l, \mathcal{E})}{|\mathcal{E}|} \times \log_2 \left(\frac{frec(y_l, \mathcal{E})}{|\mathcal{E}|} \right) \tag{2.5}$$

donde $\frac{frec(y_l,\mathcal{E})}{|\mathcal{E}|}$ *es la probabilidad de que se dé un ejemplo con clase* y_l*, y* $\log_2\left(\frac{frec(y_l,\mathcal{E})}{|\mathcal{E}|}\right)$ *es la información que transmite un ejemplo de clase* y_l*. La entropía es máxima cuando todas las cases presentan la misma proporción.*

Pese a su simplicidad y bajo coste computacional, *ID3* presenta inconvenientes importantes, algunos de los cuales son corregidos por su sucesor *C4.5*. Los más evidentes son la incapacidad para trabajar con atributos continuos y tratar valores ausentes. Sin embargo, presenta una serie de problemas que afectan directamente a la precisión del árbol generado. En primer lugar, la heurística usada para establecer los test es propensa a seleccionar aquellos atributos con mayor número de valores distintos, ya que a mayor número de particiones, la entropía de cada subconjunto tiende a ser menor. En segundo lugar, *ID3* resulta muy vulnerable a la presencia de ruido e inconsistencia en los datos, lo cual ocasiona la generación de *hojas muertas* que clasifican ejemplos de más de una clase.

C4.5

El algoritmo *C4.5* fue propuesto por Quinlan [133] a finales de los años 80 para mejorar las carencias de su predecesor *ID3*. Desde entonces, ha sido uno de los sistemas clasificadores más referenciados en la bibliografía, principalmente debido a su extremada robustez en un gran número de dominios y su bajo coste computacional. *C4.5* introduce principalmente las siguientes mejoras:

1. Trata eficazmente los valores desconocidos calculando la ganancia de información para los valores presentes.

2. Maneja los atributos continuos, aplicando una discretización previa.

3. Corrige la tendencia de *ID3* a seleccionar los atributos con muchos valores distintos para establecer los test cambiando el criterio de división.

C4.5 produce un árbol de decisión similar al de *ID3*, con la salvedad de que puede incluir condiciones sobre atributos continuos. Así, los nodos internos pueden contener dos tipos de test según el dominio del atributo seleccionado para la partición. Si el atributo X_i es discreto, la representación es similar a la de *ID3*, presentando un test con una condición de salida (rama $X_i = x_{iv}$) por cada valor x_{iv} diferente del atributo. Por contra, si el atributo X_i es continuo, el test presenta dos únicas salidas, $X_i \leq Z$ y $X_i > Z$, que comparan el valor de X_i con el umbral Z. Para calcular Z, se aplica un método similar al usado en [23], el cual ordena el conjunto de t valores distintos del atributo X_i presentes en el conjunto de entrenamiento, obteniendo el conjunto de

valores $\{x_{i1}, x_{i2}, \ldots, x_{it}\}$. Cada par de valores consecutivos aporta un posible umbral $Z = \frac{x_{iv}+x_{i(v+1)}}{2}$, teniendo en total $t-1$ umbrales, donde t es como mucho igual al número de ejemplos. Una vez calculados los umbrales, *C4.5* selecciona aquel que maximiza el criterio de separación. Como se mencionó anteriormente, el criterio de maximización de la ganancia de información usado en *ID3* produce un sesgo hacia los atributos que presentan muchos valores distintos. *C4.5* resuelve este problema usando la razón de ganancia (*gain ratio*) como criterio de separación a la hora de establecer un test. Esta medida tiene en cuenta tanto la ganancia de información como las probabilidades de los distintos valores del atributo. Dichas probabilidades son recogidas mediante la denominada *información de separación* (*split information*), que no es más que la entropía del conjunto de datos $\mathcal{E}$ respecto a los valores del atributo X_i en consideración, siendo calculada como

$$InformacionDeSeparacion(\mathcal{E}, X_i) = -\sum_{v=1}^{|X_i|} \frac{|\mathcal{E}(x_{iv})|}{|\mathcal{E}|} \times \log_2\left(\frac{|\mathcal{E}(x_{iv})|}{|\mathcal{E}|}\right) \tag{2.6}$$

donde $|X_i|$ es el número de valores distintos del atributo X_i; $\mathcal{E}(x_{iv})$ es el subconjunto de $\mathcal{E}$ para el cual $X_i = x_{iv}$, siendo $|\mathcal{E}(x_{iv})|$ su cardinal; y $|\mathcal{E}|$ es el número total de ejemplos. La información de separación simboliza la información potencial que representa dividir el conjunto de datos, y es usada para compensar la menor ganancia de aquellos test con pocas salidas. Con ello, tal y como muestra la ecuación 2.7, la razón de ganancia es calculada como el cociente entre la ganancia de información (ecuación 2.4) y la información de separación (ecuación 2.6). Tal cociente expresa la proporción de información útil generada por la división.

$$RazonDeGanancia(\mathcal{E}, X_i) = \frac{Ganancia(\mathcal{E}, X_i)}{InformacionDeSeparacion(\mathcal{E}, X_i)} \tag{2.7}$$

C4.5 maximiza este criterio de separación, premiando así a aquellos atributos que, aun teniendo una ganancia de información menor, disponen también de menor número de valores para llevar a cabo la clasificación. Sin embargo, si el test incluye pocos valores, la información de separación puede ser cercana a cero, y por tanto el cociente sería inestable. Para evitar tal situación, el criterio selecciona un test que maximice la razón de ganancia pero obligando a que la ganancia del mismo sea al menos igual a la ganancia media de todos los test examinados [117]. *C4.5* ha resultado ser un sistema muy efectivo en la práctica, capaz de ofrecer una representación relativamente simple de los resultados con un bajo coste computacional. En concreto, para un conjunto de datos con m ejemplos y n atributos, el coste medio de construcción del árbol es de $\Theta(mnlog_2m)$, mientras que la complejidad del proceso de poda es de $\Theta(m(log_2m)^2)$.

2.4. Evaluación del rendimiento de un clasificador

2.4.1. Precisión

Evaluar el comportamiento de los algoritmos de aprendizaje es un aspecto fundamental del aprendizaje automático, no sólo es importante para comparar algoritmos entre sí, sino que en muchos casos forma parte del propio algoritmo de aprendizaje. La forma más habitual de medir la eficiencia de un clasificador es la *precisión predictiva*. Un clasificador, cada vez que se le presenta un nuevo caso, debe tomar una decisión sobre la etiqueta que se le va a asignar. Considerando un error como una clasificación incorrecta de un ejemplo, se puede calcular fácilmente la tasa de error, o su complementaria, la tasa de acierto.

Definición 2.9 *Se denomina **precisión** (Γ) de un clasificador al resultado de dividir el número de clasificaciones correctas por el número total de muestras examinadas.*

Dado un conjunto etiquetado $\mathcal{E}$ de m instancias $(\overline{x_j}, y_j)$, donde $j = 1, \ldots, m$, cada una compuesta por n valores de entrada $x_{j,i}$ con $(i = 1, \ldots, n)$ y uno de salida y_j, y dado el clasificador $\mathcal{L}$ visto en la definición 2.7, en la expresión siguiente, si $\mathcal{L}(\overline{x_j}) = y_j$ entonces se cuenta 1, y 0 en cualquier otro caso.

$$\Gamma(\mathcal{E}, \mathcal{L}) = \frac{1}{m} \sum_{j=1}^{m} \left(\mathcal{L}(\overline{x_j}) = y_j \right)$$

Teniendo en cuenta la aplicación a la tarea de clasificación que se le da a los algoritmos de selección en esta memoria de tesis, la definición de precisión dada sobre el conjunto total de datos, aplicada a un subconjunto de atributos $\mathcal{S}$ queda de la siguiente manera:

$$\Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L}) = \frac{1}{m} \sum_{j=1}^{m} \left(\mathcal{L}(\mathcal{S}(\overline{x_j})) = y_j \right)$$

Por tanto, se tiene que $\Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L})$ es la precisión aplicando el clasificador $\mathcal{L}$ a la base de datos con los atributos que pertenecen al subconjunto $\mathcal{S}$.

La precisión es una buena estimación de cómo se va a comportar el modelo para datos desconocidos similares a los de prueba. Sin embargo, si se calcula la precisión sobre el propio conjunto de datos utilizado para generar el modelo, se obtiene con frecuencia una precisión mayor a la real, es decir, serán estimaciones muy optimistas por utilizar los mismos ejemplos en la inducción del algoritmo y en su comprobación [61]. La idea básica es estimar el modelo con una porción de los datos y luego comprobar su validez con el resto de los datos. Esta separación es necesaria para garantizar la independencia de la medida de precisión resultante, de no ser así, la precisión del modelo será sobreestimada [44].

Para tener seguridad de que las predicciones sean robustas y precisas, se consideran dos etapas en el proceso de construcción de un modelo, entrenamiento y prueba, partiendo los datos en dos conjuntos, uno de entrenamiento y otro de test.

2.4.2. Técnicas de evaluación

Partiendo de la necesidad de dividir el conjunto de datos, se dispone de diversas estrategias de validación de un sistema de aprendizaje dependiendo de como se haga la partición del conjunto.

El método de evaluación más básico, la **validación simple**, utiliza un conjunto de muestras para construir el modelo del clasificador, y otro diferente para estimar el error, con el fin de eliminar el efecto de la sobreespecialización. De entre la variedad de porcentajes utilizados, uno de los más frecuentes es tomar 2/3 de las muestras para el proceso de aprendizaje y el 1/3 restante para comprobar el error del clasificador. El hecho de que sólo se utiliza una parte de las muestras disponibles para llevar a cabo el aprendizaje, es el inconveniente principal de esta técnica, al considerar que se pierde información útil en el proceso de inducción del clasificador. Esta situación se deteriora si el número de muestras para construir el modelo es muy reducido, ya sea porque el porcentaje elegido, o por no disponer de más datos.

Otra técnica de evaluación denominada **validación cruzada**, se plantea para evitar la ocultación de parte de las muestras al algoritmo de inducción y la consiguiente perdida de información. Con esta técnica el conjunto de datos se divide en k particiones mutuamente exclusivas, conteniendo todas aproximadamente el mismo número de ejemplos, indicándose en muchos casos como **validación cruzada con k particiones** (*k-fold cross validation*). En cada evaluación, se deja uno de los subconjuntos para la prueba, y se entrena el sistema con los k-1 restantes. Así, la precisión estimada es la media de las k tasas obtenidas. En este caso, los subconjuntos de prueba son independientes, no así los de entrenamiento. Por ejemplo, en una validación cruzada con 10 particiones, cada par de subconjuntos de entrenamiento comparten el ochenta por ciento de los datos.

La utilización de este estimador supone la elección de k, el número de particiones. Breiman ([23]) encuentra que para el CART los mejores resultados se obtienen con un valor de k=10, a la misma conclusión llega Kohavi en su tesis [87]. En general éste es un número de particiones más utilizado. Una posible mejora en la utilización de la validación cruzada es la **estratificación** que consiste en mantener en cada una de las particiones una distribución de las etiquetas similar a la existente en el conjunto de aprendizaje, para evitar una alta varianza en la estimación [23]. Además, es una práctica común, repetir la validación cruzada con k particiones un número determinado de veces para hacer más estable la estimación de la precisión.

Un caso particular de este método de evaluación es la **validación cruzada dejando uno fuera** (*leaving-one-out cross validation*), donde k es igual al número de ejemplos del conjunto de datos. En este caso, el clasificador entrena con todas las muestras menos una que deja fuera para realizar la prueba [97]. Además de la elevada varianza de la tasa de aciertos obtenida, el mayor inconveniente de este método es el alto coste computacional que supone el aprendizaje del clasificador k veces, por lo que no se suele utilizar cuando el número de muestras es elevado o el proceso de inducción del clasificador es computacionalmente costoso. La ventaja de esta técnica es que todos los casos son utilizados en el proceso de aprendizaje y en el de prueba, dando lugar a un estimador con sesgo muy pequeño.

Otra técnica para estimar el error de un modelo cuando se dispone de pocos datos es la conocida como ***bootstrapping***. Las técnicas de estimación basadas en este concepto fueron introducidas por Efron [55], encontrándose desarrolladas en más detalle en [56, 57]. Estas técnicas se proponen para reducir la alta variabilidad que exhibe la validación cruzada en muestras pequeñas, consiguiendo un aumento de eficiencia comparable a un aumento en el tamaño de la muestra de un 60 %. Suponiendo que se tiene un conjunto de datos con m ejemplos, la forma de aplicar esta validación consiste en realizar un muestreo aleatorio con reposición de m ejemplos, para formar el conjunto de aprendizaje. Todas aquellas muestras que no sean elegidas formarán parte del conjunto de prueba. Este proceso se repite un número prefijado de veces y después se actúa como en el caso de la validación cruzada, promediando las precisiones. Lo interesante de este proceso es que las repeticiones son independientes y esto hace que el método sea más robusto estadísticamente.

2.4.3. Comparación del rendimiento

En el apartado anterior se expuso como evaluar un algoritmo de aprendizaje obteniendo un valor de precisión. En el caso de que estemos interesado en comparar dos técnicas de aprendizaje, se hará utilizando test de hipótesis. Una de las pruebas estadísticas más populares para este propósito es el llamado t-test (Student's t-test). Si suponemos que los valores de precisión se calculan bajo las mismas condiciones, es decir utilizando las mismas muestras, se denomina test pareado. Esta prueba se utiliza para certificar si las medias de dos grupos son estadísticamente diferentes, y si es así, qué técnica es la mejor. En nuestros experimentos, al igual que en los trabajos similares, se ha supuesto que hay suficientes ejemplos y que los valores siguen una distribución normal. Para determinar si la diferencia es significativa, se debe fijar un nivel de confianza y comparar con el valor límite de la variable t-Student en la tabla correspondiente para esos grados de libertad e intervalo de confianza.

El método tradicional de realizar un contraste consiste en dividir el rango de discrepancias

que puede observarse cuando la hipótesis nula, H_0, es cierta en dos regiones: una región de aceptación de H_0 y otra de rechazo. Se consideran diferencias «demasiado grandes» las que tienen una probabilidad pequeña α (normalmente 0,1, 0,05 o 0,01) de ocurrir si H_0 es cierta. Si rechazamos H_0 cuando ocurre una discrepancia de probabilidad α, este número puede interpretarse como la probabilidad que estamos dispuestos a asumir de rechazar H_0 cuando es cierta y se denomina error tipo I. Sin embargo, existe otro posible error, aceptar H_0 cuando es falsa, denominándose error tipo II. Dependiendo del conjunto de datos y de la técnica de evaluación practicada, estos tipos de errores pueden verse aumentados o disminuidos.

Dietterich [46] compara varios métodos de evaluación mediante experimentos con datos reales y simulados. Antes de la recomendación final de su trabajo, avisa de que los test estadísticos descritos deben ser vistos como aproximados, tests heurísticos, mas que métodos estadísticos rigurosamente correctos a causa de los inconvenientes propios de cada test (entrenamiento con muestras de menor tamaño que el original, asunción de independencia, solapamiento de subconjunto de entrenamiento, etc.). Además, los experimentos se realizaron sólo con dos algoritmos de aprendizaje sobre tres bases de datos, por lo que informa que sus conclusiones se consideren como tentativas. Recomienda utilizar validación cruzada 5×2 (Alpaydin [11] propone una modificación) cuando las técnicas de aprendizaje son lo suficientemente eficientes para ejecutarse diez veces, o utilizar el test de McNemar´s en el caso de una única ejecución. No se debe utilizar el t-test sobre una serie de pruebas donde el conjunto de datos se divide aleatoriamente en dos. Recomienda precaución al interpretar los resultados de t-test pareado de la validación cruzada con diez particiones. Este test tiene una elevada probabilidad de error tipo I, sin embargo, se recomienda en los casos donde se le dé más importancia al error tipo II.

2.5. Preparación de los datos

La utilidad de la extracción de información de los datos depende en gran medida de la calidad de éstos. Como se pudo comprobar en la figura 2.1, existe una fase de preparación de los datos previa a su análisis (figura 2.2), donde se realiza una serie de tareas que describiremos a continuación. Pyle [128] indica que el propósito fundamental de esta fase es el de manipular y transformar los datos en bruto, de manera que la información contenida en el conjunto de datos pueda ser descubierta o más fácilmente accesible.

Dado que en muchas ocasiones los datos provienen de diferentes fuentes, pueden contener valores impuros (incompletos, con ruido e inconsistentes), pudiendo conducir a la extracción de patrones poco útiles. Además, se puede reducir el conjunto de datos (selección de características y de instancias), mejorando la eficiencia del proceso de minería de datos posterior. También

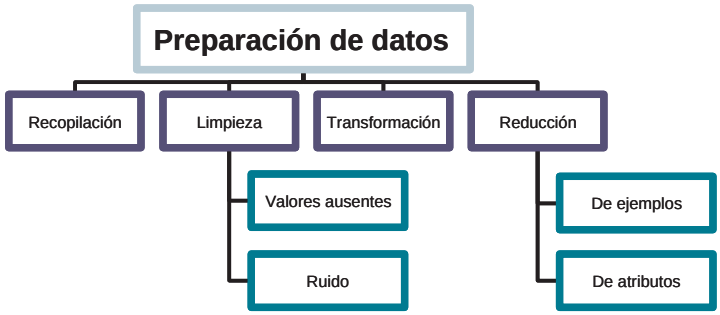


Figura 2.2: Fase de preparación de los datos.

existe la posibilidad de recuperar información incompleta, eliminar *outliers*, resolver conflictos, etc., generando un conjunto de datos de calidad, que conduciría a mejores patrones.

La preparación o preprocesamiento de datos engloba a todas aquellas técnicas de análisis de datos que permiten mejorar la calidad de un conjunto de datos, de modo que los métodos de extracción de conocimiento (minería de datos) puedan obtener mayor y mejor información (mejor porcentaje de clasificación, reglas con más completitud, etc.) [186]. La lista de tareas que se incluyen en esta fase se pueden resumir en cuatro: recopilación de datos, limpieza, transformación y reducción, no teniéndose que aplicar siempre en un mismo orden (ver figura 2.2).

2.5.1. Recopilación

Para poder comenzar a analizar y extraer algo útil en los datos es preciso, en primer lugar, disponer de ellos. Esto en algunos casos puede parecer trivial, partiendo de un simple archivo de datos, sin embargo en otros es una tarea muy compleja donde se debe resolver problemas de representación, de codificación e integración de diferentes fuentes para crear información homogénea.

2.5.2. Limpieza

En esta fase se resuelven conflictos entre datos, comprobando problemas de ruido, valores ausentes y outliers [84].

Valores ausentes

La ausencia de valores en los atributos de algunos ejemplos de las bases de datos es relativamente frecuente, debido principalmente a fallos cometidos durante el proceso de adquisición de los datos, sea manual o automático. Aunque algunos métodos solventan este problema du-

rante el proceso de aprendizaje, es común aplicar alguna técnica que trate estos ejemplos antes de ofrecerlos al algoritmo de minería de datos. La técnica de tratamiento de valores ausentes más simple, aunque también la menos recomendable, consiste en eliminar aquellos ejemplos que presenten algún atributo sin valor. El mayor inconveniente de esta técnica es que se podría eliminar información útil para el aprendizaje contenida en los atributos correctos. Para poder mantener los ejemplos en el conjunto de datos, habría que rellenar los valores ausentes con algún valor válido. Una solución sencilla es asignar una constante, por ejemplo «desconocido», si el atributo es discreto, o infinito, si es continuo. Aunque esta solución es también muy simple y no elimina información, el algoritmo de aprendizaje podría interpretar erróneamente esas constantes y entender que son valores interesantes. Por esta razón, es recomendable sustituir las ausencias por valores cuya influencia en el aprendizaje sea mínima. En este sentido, la media o la moda, dependiendo si el atributo es continuo o discreto respectivamente, pueden ser valores más apropiados que una constante. Para que el valor de sustitución no sea único para todos los ejemplos con ausencias en un mismo atributo, la media o la moda no se calcula a partir de todos los datos, sino considerando sólo aquellos ejemplos que tienen la misma clase que el que se pretende completar. Aunque este método no es muy exacto es uno de los más populares. Finalmente, una técnica más precisa, aunque también más costosa computacionalmente, consiste en sustituir las ausencias por el valor más probable aplicando algún clasificador (regresión, clasificador Bayesiano o inducción de árboles de decisión) para predecir dicho valor.

Ruido

Ruido es un error aleatorio o variación en el valor de un atributo, debido normalmente a errores en la medida del mismo. A diferencia de la ausencia de valores, el ruido es más difícil de detectar a simple vista, ya que son valores presentes en el conjunto de datos que pueden provocar que el algoritmo de minería de datos obtenga soluciones erróneas. Para mitigar los efectos del ruido en el aprendizaje se aplican las denominadas técnicas de suavizado (*smoothing*). El método de suavizado más sencillo, conocido como *binning*, consiste en ordenar los valores de un atributo y distribuir tales valores en grupos o recipientes (*bins*) de igual número de valores o de igual rango, independientemente de los valores que contenga. Tras esta partición, se realiza un tratamiento local, sustituyendo los valores de cada grupo por la media, mediana o moda de dicho grupo. Aunque la aplicación de esta técnica suaviza los efectos del ruido, no garantiza la eliminación del mismo, ya que un atributo puede tomar valores que no correspondan a las características del ejemplo al que pertenece. Además, este método no corrige sólo los posibles outliers, sino que realiza cambios en todos los valores, por lo que no es muy recomendable. Una estrategia más apropiada es aplicar algún método de clustering para detectar los outliers y poder

tratarlos posteriormente. Una vez detectados los outliers, se elimina el ejemplo o bien se aplica algún método de sustitución similar a los descritos para el tratamiento de valores ausentes que introduzca al ejemplo en uno de los clusters de su misma clase.

2.5.3. Transformación

En ocasiones, la forma en que viene dada la información originalmente no es la más adecuada para adquirir conocimiento a partir de ella. En esas situaciones se hace necesario la aplicación de algún tipo de trasformación para adecuar los datos al posterior proceso de aprendizaje, como por ejemplo normalización o cambio de escala, discretización, generalización o extracción de atributos. Esta última trasformación está estrechamente relacionada con la selección de características detallada más adelante y consiste en construir nuevos atributos a partir de combinaciones de los originales. Según la clasificación de las tareas que se incluyen en la fase de preprocesado (ver figura 2.3), se considera técnica de transformación aquella destinada a modificar los datos para mejorar el proceso de aprendizaje y no a corregir errores en los mismos.

Como ejemplo de necesidad de transformación en los datos, se puede observar la situación que se plantea a continuación. Un gran número de algoritmos de aprendizaje operan exclusivamente con espacios discretos, sin embargo, muchas bases de datos contienen atributos de dominio continuo, lo que hace imprescindible la aplicación previa de algún método que reduzca la cardinalidad del conjunto de valores que estas características pueden tomar, dividiendo su rango en un conjunto finito de intervalos. Esta trasformación de atributos continuos en discretos se denomina *discretización*. Menos frecuente es la transformación inversa denominada *numerización*.

2.5.4. Reducción

Los investigadores dedicados al aprendizaje automático supervisado, y concretamente, al estudio de algoritmos que produzcan conocimiento en alguna de las representaciones usuales (listas de decisión, árboles de decisión, reglas de asociación, etc.) suelen realizar las pruebas con bases de datos estándares y accesibles a toda la comunidad científica (la gran mayoría de ellas de reducido tamaño), con objeto de verificar los resultados y validarlos con independencia. No obstante, y una vez asentadas estas propuestas, algunos de estos algoritmos sufren modificaciones orientadas a problemas específicos, los cuales, contienen una cantidad de información muy superior (decenas de atributos y decenas de miles de ejemplos) a la de las bases de datos de prueba. La aplicación de tales técnicas de minería de datos es por tanto una tarea que consume una enorme cantidad de tiempo y memoria, aun con la potencia de los ordenadores

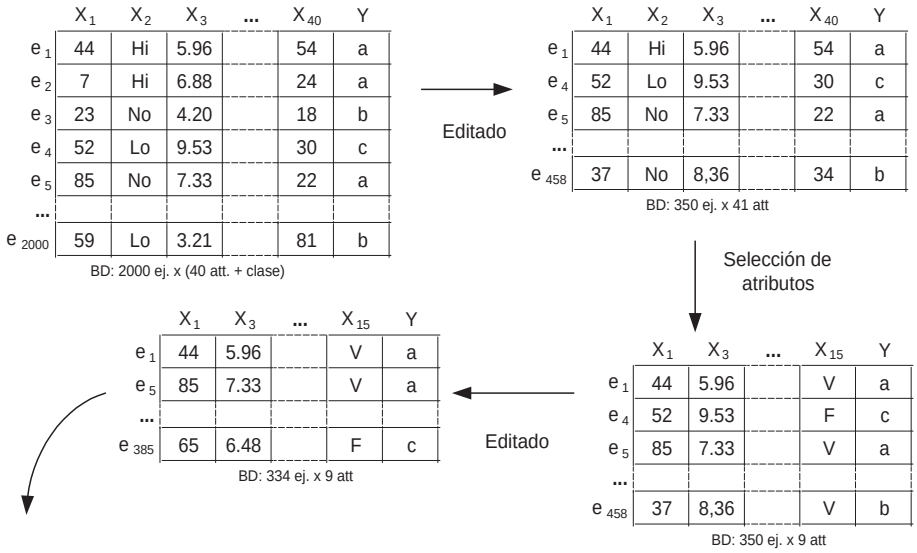


Figura 2.3: Reducción de los datos en ambos sentidos: ejemplos y atributos.

actuales, que hace intratable la adaptación del algoritmo para solucionar el particular problema. Es conveniente, pues, aplicar técnicas de reducción a la base de datos, estando orientadas fundamentalmente hacia dos objetivos: técnicas de editado (reducción del número de ejemplos) y técnicas selección de atributos (eliminar aquellos atributos que no sean relevantes para la información inherente a la base de datos). La figura 2.3 es un ejemplo donde se puede observar los dos tipos de reducción.

Editado

Las técnicas de editado tienen como objetivo reducir el número de ejemplos de un conjunto de datos $\mathcal{E}$, obteniendo un subconjunto $\mathcal{S}$ que contenga el mismo conocimiento que $\mathcal{E}$. Para ello se pueden seguir dos estrategias: formar $\mathcal{S}$ a partir de la selección o rechazo de ejemplos contenidos en $\mathcal{E}$, siendo estrictamente $\mathcal{S} \subseteq \mathcal{E}$; o bien construir $\mathcal{S}$ en base a prototipos [30] o reglas [50, 151], que representen grupos de ejemplos de $\mathcal{E}$, aunque dichos prototipos no coincidan con ejemplos de $\mathcal{E}$.

Evidentemente, la búsqueda del subconjunto $\mathcal{S}$ se lleva a cabo aplicando algún tipo de heurística, ya que una búsqueda exhaustiva es impracticable por su elevado coste computacional. Dependiendo del sentido de esta búsqueda, la técnicas de reducción de ejemplos se clasifican en: *incrementales*, donde el conjunto $\mathcal{S}$ es inicialmente vacío y se le van añadiendo ejemplos de $\mathcal{E}$ seleccionados según un determinado criterio; y *decrementales*, donde inicialmente $\mathcal{S} = \mathcal{E}$ y se van eliminando ejemplos o generalizando éstos en reglas o prototipos. Aunque los métodos decrementales suelen ser más costosos computacionalmente, los incrementales son más sensibles

al orden de los ejemplos en el conjunto $\mathcal{E}$.

Existen numerosas técnicas de editado ligadas a la técnica de los vecinos más cercanos [33]. Podemos citar los trabajos de [72], donde incluye en el conjunto de prototipos aquellos ejemplos cuya clasificación es incorrecta utilizando la técnica del vecino más cercano (1–*NN*); o [172], donde elimina aquellos ejemplos cuya clasificación es incorrecta utilizando la técnica del vecino más cercano; o [138]; o [164]; las variantes basadas en vecinos de Voronoi [86], vecinos de Gabriel (dos ejemplos son vecinos de Gabriel si la esfera con diámetro el segmento que une los dos ejemplos no contiene a otro ejemplo) o vecinos relativos [165] (dos ejemplos son vecinos relativos si para todo ejemplo de la base de datos la distancia entre los dos ejemplos es menor que la mayor de las distancias que unen a cualquier ejemplo con los dos ejemplos investigados). Todas ellas necesitan de una distancia y en algunos casos el coste computacional es elevado. Si consideramos m ejemplos y n atributos, las primeras citadas tienen un orden $\Theta(mn^2)$, la técnica de los vecinos de Voronoi tiene $\Theta(nm^2)$, y las técnicas de los vecinos de Gabriel y vecinos relativos tienen $\Theta(mn^3)$.

Un acercamiento muy distinto se realiza mediante el algoritmo denominado *EPO* [1] (*Editado mediante Proyección Ordenada*), obteniendo una reducción importante de registros con un coste computacional inferior a los algoritmos convencionales $\Theta(mnlogn)$, sin necesidad de cálculo de distancias. Trabaja indistintamente con atributos continuos [2] y discretos [136]. Otra aproximación diferente se presenta en [25], donde se realiza una reducción de datos basada en la selección evolutiva de instancias.

Selección de atributos

En la selección de características se intenta escoger el subconjunto mínimo de atributos de acuerdo con dos criterios: que la tasa de aciertos no descienda significativamente; y que la distribución de clase resultante, sea lo más semejante posible a la distribución de clase original, dados todos los atributos. En general, la aplicación de la selección de características ayuda en todas las fases del proceso de minería de datos para el descubrimiento de conocimiento.

La intención de este capítulo es la de situar el preprocesado de los datos, y más concretamente la selección de características, dentro del proceso de extracción de conocimiento. A partir de este instante, se tratarán solo conceptos relacionado con la selección de atributos, empezando por su estado del arte, y continuando con la descripción detallada de la propuesta de evaluación de atributos que se hace en este trabajo de investigación.

Capítulo 3

Selección de atributos

En los últimos años se ha producido un importante crecimiento de las bases de datos en todas las áreas del conocimiento humano. Este incremento es debido principalmente al progreso en las tecnologías para la adquisición y almacenamiento de los datos. Teóricamente, el tener más atributos daría más poder discriminatorio. Sin embargo, la experiencia con algoritmos de aprendizaje ha demostrado que no es siempre así, detectándose algunos problemas: tiempos de ejecución muy elevados, aparición de muchos atributos redundantes y/o irrelevante, y la degradación en el error de clasificación.

En este capítulo se hace revisión actual al estado del arte de selección de atributos, teniendo en cuenta los estudios previos realizados por J.J. Lorenzo [113], Molina et al. [119], Larrañaga et al. [79] y los diversos trabajos donde está presente Liu [37, 103, 111, 112]. Se organiza de la siguiente forma: en la primera sección se introduce el proceso de selección de atributos, en el que se intenta obtener los atributos que mejor definen una clase en función de algún criterio de relevancia. Por tanto, tras definir qué se entiende por selección de atributos en la sección 3.2, se enumeran algunas definiciones de relevancia de atributos existentes en la literatura en la sección 3.3. En la sección 3.4 se describe el proceso general de selección de atributos junto a sus componentes principales. Un importante número de algoritmos de selección son descritos y catalogados en la sección 3.5. Para finalizar este capítulo, en la sección 3.6 se explican diversas maneras de evaluar y comparar algoritmos de aprendizaje.

3.1. Introducción

Es un hecho que el comportamiento de los clasificadores mejora cuando se eliminan los atributos no relevantes y redundantes. La selección de los atributos relevantes se debe a la preferencia por los modelos más sencillos frente a los más complejos. Esta preferencia ha sido

utilizada con bastante frecuencia en la ciencia moderna y tiene sus orígenes en el denominado Principio de la Cuchilla de Occam (*Occam's Razor*) [64].

La selección de atributos es un campo de investigación y desarrollo productivo desde los años setenta, donde confluyen distintas áreas como el reconocimiento de patrones [16, 44, 82, 158], el aprendizaje automático [20, 83, 85, 89] y la minería de datos [37, 103, 186]. Las técnicas de selección de características se aplican en muchos entornos diferentes, como por ejemplo en la clasificación de textos [62, 182, 186], en la recuperación de imagenes [139], en la dirección de relaciones con clientes [123], en la detección de intrusos [101] y en bioinformática [80, 176, 185]. Se hace constar, que el proceso de selección de atributos, además de preceder a la clasificación, suele estar presente en las etapas previas de las principales tareas de la minería de datos, ya sean supervisadas o no, como regresión, agrupamiento y reglas de asociación [112].

3.2. Definición

Partiendo de la premisa de que en el proceso de selección de atributos se escoge un subconjunto de atributos del conjunto original, este proceso pretende elegir atributos que sean relevantes para una aplicación y lograr el máximo rendimiento con el mínimo esfuerzo.

Se debe tener en cuenta que los atributos irrelevantes y redundantes existentes en una base de datos pueden tener un efecto negativo en los algoritmos de clasificación: (1) El tener más atributos implica normalmente la necesidad de disponer de más instancias para garantizar la fiabilidad de los patrones obtenidos (variabilidad estadística entre patrones de diferente clase). Por consiguiente, el algoritmo de clasificación entrenado con todos los datos tardará más tiempo. (2) Los atributos irrelevantes y/o redundantes pueden confundir a los algoritmos de aprendizaje. Por lo que en general, el clasificador obtenido es menos exacto que otro que aprenda sobre datos relevantes. (3) Con la presencia de atributos redundantes y/o irrelevantes, el clasificador obtenido será más complejo, dificultando el entendimiento de los resultados. (4) Además, la reducción de características se podría tener en cuenta en futuras capturas de datos, reduciendo el coste de almacenamiento y tal vez el económico.

En resumen, el resultado obtenido al aplicar técnicas de selección de atributos sería:

- Menos datos → los clasificadores pueden aprender más rápidos.
- Mayor exactitud → el clasificador generaliza mejor.
- Resultados más simples → más fácil de entender.
- Menos atributos → evita obtenerlos posteriormente.

Podemos concluir que la selección es efectiva en eliminar atributos irrelevantes y redundantes, incrementando la eficiencia en las tareas de minería, mejorando el rendimiento y la comprensión de los resultados.

Definición 3.1 (Selección de atributos) *Si $\mathcal{X}$ es el conjunto de atributos, hacer selección de atributos es escoger un subconjunto $S \in \mathcal{P}(\mathcal{X})$. $\mathcal{P}(\mathcal{X})$ es el conjunto de las partes de $\mathcal{X}$, es decir, el conjunto formado por todos los subconjuntos de elementos de $\mathcal{X}$.*

Existen dos aproximaciones para realizar una reducción de dimensionalidad: selección y transformación de características; ambas son técnicas de pre-procesado que se usan frecuentemente. Aclaramos que, transformación de parámetros es el proceso a través del cual se crea un nuevo conjunto de parámetros. Existiendo dos variantes:

- Construcción de parámetros: es el proceso de descubrir información oculta sobre relaciones entre parámetros, aumentando el espacio formado por el conjunto de los atributos. Después de la construcción de parámetros, se obtienen p atributos adicionales: $X_{n+1}, X_{n+2}, \ldots, X_{n+p}$.

- Extracción de parámetros: es el proceso de extraer un conjunto de nuevos atributos a partir de los originales aplicando funciones. Tendremos: $B_1, B_2, \ldots, B_s$ $(s < n)$, siendo $B_i = F_i(X_1, X_2, \ldots, X_n)$, donde F_i es una función.

3.3. Relevancia y redundancia

Teniendo en cuenta el propósito de los algoritmos de selección de atributos de escoger los relevantes, antes de continuar con el capítulo conviene aclarar qué se entiende por relevancia. Un atributo que no es relevante no es útil para la inducción, aunque no todos los atributos que son relevantes son necesariamente útiles para esta [29]. La noción de relevancia no ha sido aún justificada rigurosamente sobre un acuerdo de común entendimiento [15].

El concepto de relevancia es necesario definirlo de forma que se pueda identificar y seleccionar qué atributos son relevantes. En [169] se recogen otras definiciones de relevancia desde un punto de vista formal en diferentes campos, aunque el interés en el presente trabajo se centra en las definiciones de relevancia en aprendizaje automático. Se comprueba en la bibliografía de este campo que no existe un consenso sobre una definición única de relevancia. Como indica Blum [20], esto es lógico debido a que las definiciones de relevancia deben tener en cuenta la pregunta «¿Relevante para qué?». En este sentido Michalski [115] clasifica los atributos que se

utilizan en el proceso de aprendizaje en tres categorías según su relevancia: completa, parcial e indirecta.

Una primera definición de relevancia [20] es la noción de ser «relevante respecto al objetivo» que se asume de clasificación.

Definición 3.2 (Relevancia con respecto a la clase) *Un atributo X_i es relevante para el concepto Y si existe un par de ejemplos e_j y e_k en el dominio de instancias $\mathcal{E}$ tal que e_j y e_k difieren sólo en el valor de X_i y en la etiqueta $y_j \neq y_k$*

En otras palabras, si existen dos instancias que sólo pueden ser clasificadas gracias a X_i. Esta definición tiene el inconveniente de que el algoritmo de aprendizaje no necesariamente puede determinar si un atributo X_i es relevante o no, usando solamente una muestra S del espacio de instancias E. Además, si la codificación de atributos es redundante (e.g. atributos que contienen la misma información), nunca se dará el caso de que dos instancias difieran solamente en un atributo. Por otra parte y debido al ruido en el conjunto de aprendizaje, se pueden detectar como atributos relevantes a aquellos que no lo son. Una definición similar a la anterior es la debida a Almuallim y Dietterich [9] pero restringida a atributos booleanos. Para tener en cuenta el conjunto de aprendizaje $\mathcal{E}$ y de esta forma evitar el inconveniente que posee la anterior definición, John et al. [83] dan las siguientes definiciones de relevancia.

Definición 3.3 (Relevancia fuerte respecto a $\mathcal{S}$) *Un atributo $X_i \in \mathcal{X}$ es fuertemente relevante a la muestra $\mathcal{S}$ si existen ejemplos $e_j, e_k \in \mathcal{S}$ que difieren solamente en el valor de X_i y $y_j \neq y_k$.*

Es decir, es la misma Definición 3.2, pero ahora $e_j, e_k \in \mathcal{S}$ y la definición es con respecto a $\mathcal{S}$.

Definición 3.4 (Relevancia débil respecto a $\mathcal{S}$) *Un atributo $X_i \in \mathcal{X}$ es débilmente relevante a la muestra $\mathcal{S}$ si existen al menos un $\mathcal{X}' \subset \mathcal{X}$ ($X_i \in \mathcal{X}'$) donde X_i sea fuertemente relevante respecto a $\mathcal{S}$.*

Es decir, puede aparecer un atributo débilmente relevante al eliminar un subconjunto de atributos donde alguno de ellos es fuertemente relevante.

Un elemento importante que aportan las anteriores definiciones es permitir seleccionar los atributos en función del grado de relevancia, siendo imprescindibles los atributos fuertemente relevantes, dado que su eliminación añade ambigüedad a las muestras del conjunto de aprendizaje. Sin embargo, los atributos débilmente relevantes pueden mantenerse o no, dependiendo de qué atributos contenga el conjunto seleccionado y del criterio que se desea optimizar.

Otra definición de relevancia debida a Wang [169] y que hace uso de conceptos de Teoría de la Información es la relevancia variable de un atributo respecto a la clase.

Definición 3.5 (Relevancia variable o entrópica) *La relevancia variable del atributo X_i con respecto a la clase Y se define como la relación existente entre la información mutua entre el atributo y la clase, y la entropía de la clase.*

$$r(X_i, Y) = I(X_i, Y)/Ent(Y)$$

La relevancia variable se aplica también al caso condicional del conocimiento de un conjunto de atributos, denominándose entonces relevancia condicional. Una diferencia que existen entre la relevancia variable y las definiciones de relevancia fuerte o débil es que se introduce un grado de relevancia, es decir, no asigna un todo o nada a los atributos. Así $r(x_i, Y) = 1$ indica la relevancia máxima y a medida que decrece hacia cero la relevancia va disminuyendo.

Desde otro punto de vista, en lugar de preocuparse de qué atributos son relevantes, se puede usar la relevancia como una «medida de complejidad» respecto al objetivo Y. Esto significa que la relevancia dependerá del inductor usado [20].

Definición 3.6 (Relevancia como medida de complejidad) *Siendo $\mathcal{S}$ una muestra de datos y un objetivo Y, se define $r(\mathcal{S}, Y)$ como el número de atributos relevantes a Y usando la Definición 3.2 sólo en $\mathcal{S}$, tal que el error sobre $\mathcal{S}$ sea el menor posible y se tenga el menor número de atributos relevantes.*

En la anterior definición se utiliza la relevancia como una medida de la complejidad de la función de evaluación, ya que el objetivo es buscar el algoritmo que obtenga mejor resultado con la menor complejidad. Es decir, encontrar el menor número de atributos necesarios para obtener un funcionamiento óptimo en el conjunto de aprendizaje con respecto al concepto que representa. Todas las definiciones anteriores son independientes del algoritmo de inducción utilizado, pero puede resultar que un atributo que es relevante según algunas de las definiciones anteriores, utilizado con un determinado algoritmo de inducción no intervenga en el proceso de inducción por lo que se considera irrelevante por dicho algoritmo. Una definición que si considera esta situación es la *utilidad incremental* [28].

Definición 3.7 (Relevancia Utilidad incremental) *Dado un conjunto de aprendizaje $\mathcal{E}$, un algoritmo de aprendizaje $\mathcal{L}$, y un conjunto de atributos $\mathcal{S}$, un atributo X_i es incrementalmente útil para $\mathcal{L}$ con respecto a $\mathcal{S}$ si la tasa de acierto de la hipótesis que $\mathcal{L}$ produce usando el conjunto de atributos $\{X_i\} \cup \mathcal{S}$ es mejor que la tasa de acierto obtenida utilizando sólo el conjunto de atributos $\mathcal{S}$.*

Esta definición se adapta al esquema de selección de atributos como un proceso de búsqueda, que es lo que se describe en la siguiente sección.

Redundancia

Un concepto ligado con el de relevancia es el de redundancia. El concepto de redundancia entre atributos se expresa normalmente en términos de correlación entre atributos. Frecuentemente se considera que dos atributos son redundantes si sus valores están completamente correlacionados. En general, se distinguen dos tipos de medidas de la correlación entre dos atributos (X, Y): lineal y no lineal. En el primer caso, se utiliza el coeficiente de correlación de Pearson, ρ, dado por

$$\rho = \frac{\sum_j (x_j - \overline{x_j})(y_j - \overline{y_j})}{\sqrt{\sum_j (x_j - \overline{x_j})^2} \sqrt{\sum_j (y_j - \overline{y_j})^2}}$$

donde $\overline{x_j}$ e $\overline{y_j}$ son la media de X e Y respectivamente. Y en el segundo, están las medidas basadas en los conceptos de entropía, o medida de la incertidumbre de una variable aleatoria. Se utiliza con frecuencia la incertidumbre simétrica, definida como

$$SU(X, Y) = 2 \left[\frac{Ganancia(X|Y)}{Ent(X) + Ent(Y)} \right]$$

donde $Ent(X)$ es la entropía de una variable X y $Ganancia(X|Y)$ es la ganancia de información de la variable X con respecto a Y (definiciones 2.5 y 2.4).

Las definiciones de correlación anteriores se dan entre pares de variables, sin embargo no está tan claro como determinar cuando un atributo está correlacionado con un conjunto de atributos. Koller y Sahami [91] aplican una técnica basada en la entropía cruzada (distancia KL, Kullback & Leibler [96]), denominada Markov blanket filtering, para eliminar los atributos redundantes. Se formaliza esta idea en la siguiente definición.

Definición 3.8 (Independencia condicional) *Se dice que dos atributos (o conjuntos de atributos) X, Y son condicionalmente independientes dado un tercer atributo Z (o conjunto) si, dado Z hace X e Y independientes. Es decir, la distribución de X, conocido Y y Z, es igual a la distribución de X conocido Z, por tanto, Y no ejerce influencia sobre X ($P(X|Y, Z) = P(X|Z)$).*

Definición 3.9 (Markov blanket) *Dado un atributo $X_i \in \mathcal{S}$ (un conjunto de atributos) y la clase C, el subconjunto $M \subseteq \mathcal{S}(X_i \notin M)$ es un* Markov blanket *de X_i si, dado M, X_i es condicionalmente independiente de $\mathcal{S} - M - \{X_i\}$ y C.*

Teóricamente, se demuestra que una vez encontrada una *Markov blanket M* de un atributo X_i en un conjunto $\mathcal{S}$, se puede eliminar con toda seguridad X_i de $\mathcal{S}$ sin incrementar la divergencia con la distribución original. Además, en un proceso de filtrado secuencial, en el que se van eliminando atributos uno a uno, un atributo etiquetado como innecesario basado en la existencia

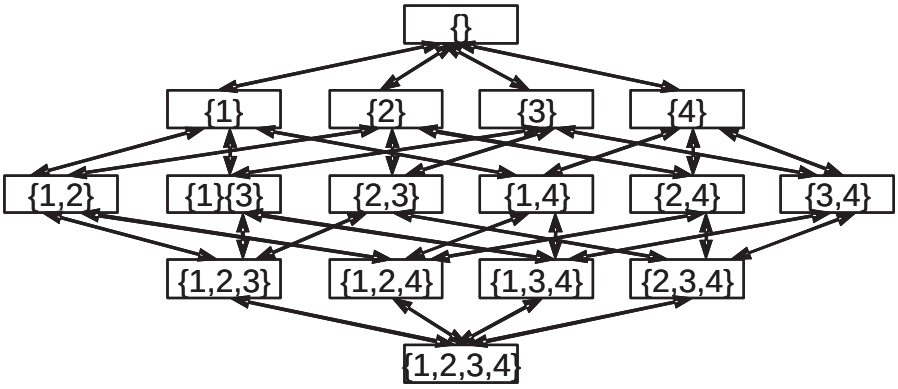


Figura 3.1: Espacion de búsqueda.

de una *Markov blanket M* no se necesitará posteriormente cuando otros atributos hayan sido eliminados.

La condición de *Markov blanket* requiere que M contenga no sólo la información que X_i tiene de la clase C, sino también toda la que tiene de los demás atributos. En [91], lo único que se dice del número de atributos del conjunto M es que debe ser pequeño y fijo. Algunos de los trabajos más referenciados que se basan en este concepto son los realizados por Xing et al. [175] y por Yu y Liu [184]. En el primero, no se precisa el número de atributos para M, pero si se sabe que es un número pequeño predefinido de los atributos más correlacionados con X_i, mientras que en el segundo, M está formado por un sólo atributo.

3.4. Proceso general

La selección de atributos se puede considerar como un problema de búsqueda [158, 98] en un espacio de estados, donde cada estado corresponde con un subconjunto de atributos, y el espacio engloba todos los posibles subconjuntos que se pueden generar. En un conjunto de datos donde el número de atributos sea cuatro ($n = 4$), el espacio total lo componen dieciséis subconjuntos (2^n), como aparece en la figura 3.1. El proceso de selección de atributos puede entenderse como el recorrido de dicho espacio hasta encontrar un estado (combinación de atributos) que optimice alguna función definida sobre un conjunto de atributos.

En general, un algoritmo de selección de atributos se basa en dos pasos básicos: generación y evaluación de subconjuntos. En la generación de nuevos subconjuntos se define un punto de partida y una estrategia para recorrer el espacio de búsqueda hasta que se cumpla un criterio de parada (figura 3.2). Aunque para finalizar el proceso de búsqueda se utilice la función de evaluación, se comentarán las distintas opciones de parada existentes en el proceso

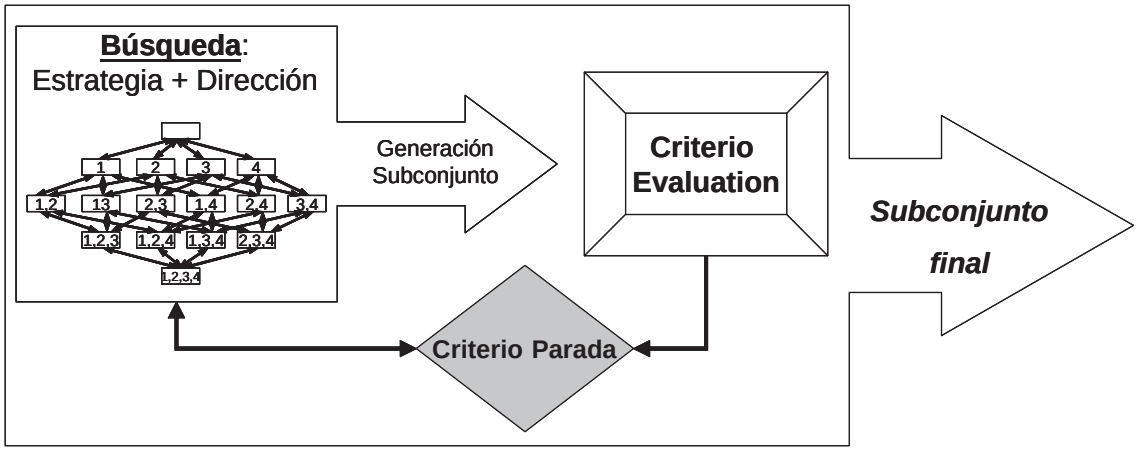


Figura 3.2: Proceso de Selección de Atributos.

de generación de subconjuntos.

3.4.1. Estudios preliminares

Existen bastantes referencias de trabajos relacionados con la selección de atributos, pero además, se han realizado estudios sobre diversos aspectos de la selección de atributos (técnicas de búsqueda, medidas de bondad de los atributos, etc.), donde se agrupan los distintos algoritmos existente en la bibliografía general.

- Langley [98] divide las funciones de evaluación en dos categorías: *Filtro* y *Envolvente* (*wrapper*). En la primera categoría se incluyen los algoritmos en los que la selección de atributos se realiza como un preproceso independiente de la fase de inducción, por lo que puede entenderse como un filtrado de los atributos irrelevantes y redundantes. Por otro lado, en los métodos de tipo *envolvente* [83], la selección de atributos y los algoritmos de aprendizaje no son elementos independientes, ya que la selección de los atributos hace uso del proceso de inducción para evaluar la calidad de cada conjunto de atributos seleccionados en cada momento.

- Posteriomente, Blum y Langley [20] establecen una clasificación de las funciones de evaluación utilizando también como criterio la dependencia que existe entre el proceso de selección y el de inducción, agrupándolas en cuatro categorías. Además de las anteriores, están los algoritmos empotrados (*embedded*), donde el inductor cuenta con su propio algoritmo de selección de atributos, como ocurre en los algoritmos que generan árboles de decisión, que utilizan sólo aquellos atributos necesarios para obtener una descripción

consistente con el conjunto de aprendizaje. En la última categoría se encuadran los esquemas de selección basados en la ponderación de los distintos atributos. Un ejemplo de este esquema de selección puede observase en las redes neuronales.

- Doak en [48, 49], establece tres categorías para las estrategias de búsqueda: exponenciales, secuenciales y aleatorias. Dentro de las primeras se encuentran aquellas estrategias que tienen complejidad $O(2^n)$ pero aseguran la obtención del subconjunto óptimo. Las estrategias secuenciales, a diferencia de las exponenciales, recorren sólo una porción del espacio de búsqueda y por tanto, no aseguran la obtención del óptimo aunque el coste computacional es polinomial. Las estrategias aleatorias se basan en visitar diferentes regiones del espacio de búsqueda sin un orden predefinido, evitando de esta forma que se pueda obtener un óptimo local de la función de evaluación para un determinado subconjunto de atributos.

- Doak [49] establece una clasificación de las medidas de evaluación basada en la naturaleza de éstas, más que de su interrelación con el proceso de inducción, estableciendo tres categorías para las funciones de evaluación similares a las anteriores: filtro, wrapper y otra categoría que incluye los métodos que calculan de forma incremental la tasa de acierto de un clasificador. Esta última viene a ser un caso especial de la segunda, cuando la estrategia de búsqueda es secuencial.

- Liu et al. [37, 103, 112] proponen una clasificación de las estrategias de búsqueda y de los criterios de evaluación muy similar a la realizada por Doak [49]. Las búsquedas las agrupan en tres categorías: completa, heurística y aleatoria. En cuanto a los criterios de evaluación, dividen el tipo filtro en varias categorías dependiendo de qué propiedades se extraen de los mismos. De esta forma, la clasificación que establecen de las funciones de evaluación es la siguiente: medidas de distancia, medidas de información, medidas de dependencia, medidas de consistencia y, por otro lado, las medidas basadas en la tasa de error de un clasificador (correspondientes a las de tipo wrapper).

Aunque la bibliografía es abundante, pocos estudios se han realizado de una manera global, basándose en un marco de trabajo unificado. No existe en la actualidad ningún algoritmo ni criterio teórico que indique cuál es el mejor método para cada problema, sin embargo existen acercamientos en esta dirección:

- Molina, Belanche y Nebot en [119], además de clasificar los algoritmos de selección existentes, proponen una forma de evaluarlos con respecto a las particularidades de relevancia, irrelevancia, redundancia y tamaño de la muestra.

- En Dash y Liu [37, 103], se resumen los procesos generales de la selección de atributos en cuatro pasos, y se clasifican los algoritmos existentes desde 1970 hasta 1997 en diferentes grupos basándose en el proceso de generación de subconjuntos y en la evaluación.

- Liu y Yu [112], siguen el marco definido en [37, 103], clasificando los algoritmos recientes, incluyendo selección de atributos no supervisada, y proponen una plataforma unificada con la finalidad de preparar el camino para desarrollar un sistema de selección automático. Se intenta: definir una plataforma común para todos los algoritmos; construir una taxonomía, para descubrir cómo se complementan unos a otros, y qué se ha olvidado; y resolver el problema de la selección de atributos automáticamente.

A continuación, se describen las fases del proceso de selección basándonos en los estudios realizados por Liu en sus diferentes publicaciones. Además, se enumeran los objetivos planteables con estos algoritmos [112].

3.4.2. Generación de subconjuntos

Todo proceso de selección de atributos tiene un punto de partida, que puede ser el conjunto completo de atributos, el conjunto vacío o cualquier estado intermedio. Tras evaluar el primer subconjunto, se examinarán otros subconjuntos generados según una dirección de búsqueda (hacia adelante, hacia atrás, aleatoria o cualquier variación o mezcla de las anteriores). El proceso terminará cuando recorra todo el espacio o cuando se cumpla una condición de parada, según la estrategia de búsqueda seguida.

Dirección de búsqueda

Se entiende por dirección de búsqueda, la relación entre los atributos de un subconjunto con el siguiente, al realizar el recorrido a través del espacio de búsqueda. Se puede seguir uno de los siguientes esquemas:

- ***Secuencial Hacia Adelante***: Este esquema empieza con el conjunto vacío y se añaden secuencialmente atributos del conjunto original. En cada iteración, se elige el mejor atributo entre los no seleccionados, basándose en alguna función de evaluación. El algoritmo puede parar cuando llegue a un número determinado de atributos seleccionados, cuando la evaluación sobrepase un valor prefijado, o simplemente, al dejar de aumentar la evaluación. Si se deja ejecutar el algoritmo sin un criterio de parada, se obtiene un ranking de atributos, es decir, una lista de atributos ordenada según cuándo se incorpore cada atributo al subconjunto final.

El principal inconveniente de hacer un recorrido siguiendo esta dirección, es la posibilidad de no detectar interacciones básicas entre atributos que sean muy interesantes para la clasificación. Es decir, puede ocurrir que atributos que por separado son irrelevantes, el estar juntos en un determinado subconjunto les haga ser muy importantes. Se puede modificar para añadir los dos mejores atributos, o los tres,..., pero relentizaría el tiempo de computación.

- ***Secuencial Hacia Atrás***: Se comienza con el conjunto completo de atributos y se eliminan secuencialmente atributos. En cada iteración, basándose en alguna función de evaluación, se escoge el atributo menos relevante de entre los originales, eliminándolo del conjunto. Al igual que en el caso anterior, el algoritmo puede parar cuando llegue a un número determinado de atributos en el subconjunto, cuando la evaluación no llegue a un valor prefijado, o simplemente, al dejar de aumentar la evaluación. Igualmente, se puede obtener un ranking de atributos si se espera a llegar hasta el final, aunque no se aconseja por ir ordenado según la irrelevancia de los atributos.

  En este caso, se remedia en parte el inconveniente de la generación *hacia adelante*, aunque permanezcan interacciones ocultas. Se puede generalizar para que devuelva subconjuntos de k elementos, aunque el coste se eleva.

- ***Aleatoria***: No se conoce el punto de inicio, ni cual es el siguiente subconjunto generado. Se pasa de un subconjunto a otro sin ningún orden establecido, añadiendo o eliminando uno o varios atributos. De esta forma se pretende evitar elegir subconjuntos de atributos que sean óptimos locales, como puede ocurrir en los casos anteriores. El criterio de parada suele ser la ejecución de un número fijo de iteraciones.

Otros esquemas se pueden obtener variando o mezclando algunos de los anteriores, como el *Bidireccional*, que realiza la búsqueda hacia adelante y hacia atrás al mismo tiempo, o el esquema *Compuesto*, que aplica una serie de pasos consecutivos hacia adelante y otra serie de pasos consecutivos hacia atrás.

Estrategia de búsqueda

Para una base de datos con n atributos, existen 2^n subconjuntos candidatos. Una búsqueda exhaustiva en este espacio es totalmente ineficiente, incluso para bases de datos pequeñas, siendo necesario el uso de diferentes estrategias para atajar este problema. A continuación, se explican brevemente las características más importantes de los tipos de estrategias de búsqueda según Liu et al. [112]: completa, secuencial y aleatoria.

- ***Completa***:

  Esta búsqueda garantiza la localización de un resultado óptimo conforme a un criterio dado. Si para seleccionar los subconjuntos óptimos se ha de examinar todos los conjuntos posibles del espacio, coincidirá con la *exhaustiva*. Sin embargo, según la medida de evaluación utilizada, puede no ser necesario examinar todos los subconjuntos posibles. Podemos decir que una búsqueda exhaustiva siempre es completa, pero a la inversa no se cumple en todos los casos [153]. Algunas implementaciones de búsqueda completa y no exhaustiva son *Branch & Bound* [122] y *Beam* [48].

  Dos implementaciones clásicas de la búsqueda exhaustiva son la búsqueda en *profundidad* y en *anchura*. Ambos métodos se pueden encontrar con generación de subconjuntos *hacia adelante* y *hacia atrás*. Representando el espacio de búsqueda mediante un árbol, el recorrido en profundidad discurre por una rama del árbol completa antes de empezar con otra, es decir, empieza con un atributo X_1, y sigue con las combinaciones de dos atributos donde intervenga el anterior, y a continuación las combinaciones de tres donde aparezca X_1, y así sucesivamente hasta el conjunto total de atributos. De este modo, se realiza el mismo proceso con el resto de atributos sin repetir ningún conjunto visitado anteriormente. En cambio, el recorrido en anchura va examina el árbol por niveles, es decir, primero cada atributo individualmente, luego todas las combinaciones de dos, a continuación de tres, y así hasta llegar al conjunto completo.

- ***Secuencial***: Son estrategias que realizan una búsqueda parcial a través del espacio formado por los atributos, con el riesgo de no encontrar subconjuntos óptimos. Lógicamente son algoritmos más rápidos que los pertenecientes al grupo anterior. En este grupo se encuentran distintas variantes de la técnica *greedy*, que añaden o eliminan atributos uno a uno [103]. Otra alternativa es añadir (o eliminar) p atributos en un paso y eliminar (o añadir) q atributos en el siguiente ($p > q$) [48]. Los algoritmos con búsqueda secuencial son simples de implementar y rápidos en generar resultados, generalmente, el orden del espacio de búsqueda es $O(n^2)$ o menor.

- ***Aleatoria***: Las dos estrategias anteriores son *deterministas*, es decir, todas las veces que las ejecutamos devuelven el mismo resultado. En este apartado se introducen una serie de algoritmos que tienen la propiedad contraria, *no deterministas*, donde se espera que en diferentes ejecuciones no proporcione exactamente el mismo resultado. Este tipo de estrategias, al contrario que las anteriores, busca a través del espacio formado por los atributos de manera aleatoria, no existiendo un estado siguiente o anterior que se pueda determinar según alguna regla. En este tipo de búsqueda se pretende usar el carácter aleatorio para

no caer en mínimos locales e incluso moverse temporalmente a otros estados con peores soluciones. Además, pueden detectar interdependencias entre atributos que la búsqueda heurística no captaría. Estos algoritmos se pueden detener en cualquier momento (*any–time* [187]), por lo que es difícil determinar su coste. Normalmente, el criterio de parada es un número de iteraciones, y no se puede tener garantías de que el subconjunto elegido sea el óptimo. Además de la implementación genérica de un algoritmo aleatorio, se encuentran en este grupo las búsquedas basadas en algoritmos evolutivos, búsqueda tabú y *Simulated Annealing* para selección de atributos.

En [156] se aplican redes neuronales en la selección de atributos. Para ello, se entrena la red y posteriormente se poda las conexiones entre capas tanto como sea posible sin sacrificar la predicción. Se eliminan las unidades de entrada sin conexión con las unidades ocultas, y se selección el resto de unidades.

3.4.3. Medidas de evaluación de atributos

El conjunto óptimo es siempre relativo a un criterio de evaluación, es decir, un subconjunto óptimo elegido según una medida de evaluación, no tiene porque ser el mismo al usar otra distinta. Sin embargo, en la práctica se comprueba con cierta frecuencia, que si un atributo es relevante aparece en subconjuntos escogidos según distintas funciones de evaluación. En este documento se sigue la clasificación realizada por Liu en sus distintos trabajos [37, 103, 112], y que es detallada a continuación.

Medidas de distancia

Conocidas también como medidas de separabilidad, divergencia o discriminación. Estas medidas estiman la capacidad de un subconjunto de atributos en separar las clases. Suponiendo que instancias de la misma clase forman una región compacta en el espacio, el conjunto de atributos a seleccionar es aquél cuya separación entre estas regiones sea máxima. Utilizando este tipo de medida se intenta seleccionar aquellos atributos que hacen que los ejemplos de la misma clase estén más juntos y los de diferente clase más separado. Ejemplos de medidas de distancia son: Euclidea, Manhattan, Mahalanobis, Bhattaacharya, Kullback-Liebler, Kolmogorov, Chernoff, etc.

Medidas de información

Se basan en la ganancia de información de un atributo. Esta propiedad estadística no es mas que la reducción esperada de la entropía (desorden) de los datos al conocer el valor de un

atributo (ver conceptos de ganancia de información y entropía en 2.4 y 2.5 respectivamente). Entre las medidas de información más frecuentes se encuentran: la entropía de Shannon, de Renyi, de grado α, cuadrática, estrictamente cóncava y de Daroczy, MDLC, información mutua.

Medidas de dependencia

Las medidas de dependencia o correlación (asociación) evalúan la habilidad para predecir el valor de una variable en función de otra. El coeficiente de correlación es una medida de dependencia clásica que se utiliza para calcular la correlación entre un atributo y la clase, prefiriéndose aquellos atributos con mayor correlación. Otro enfoque consiste en determinar la dependencia de un atributo de otros, donde el valor resultante indica el grado de redundancia del atributo [16]. La evaluación de atributos mediante medidas de dependencia, está muy relacionada con la evaluación según las medidas de información y distancia.

Medidas de consistencia

Estas medidas son más recientes que las anteriores, y se caracterizan por su fuerte dependencia del conjunto de entrenamiento [9]. Estas medidas intentan extraer el subconjunto mínimo que satisfaga una tasa de inconsistencia aceptable, establecida normalmente por el usuario.

Consideramos dos instancias inconsistentes, si toman los mismos valores para todos sus atributos menos la etiqueta de la clase. El contador de inconsistencia es el número de instancias implicadas menos el número de instancias de la clase mayoritaria. Por ejemplo, existen n instancias relacionadas, entre ellas, n_1 pertenecen a la etiqueta A, n_2 a la etiqueta B y n_3 a C, donde $n_1 + n_2 + n_3 = n$. Si n_3 es el mayor de los tres, el contador de inconsistencia es $n - n_3$. Y la tasa de inconsistencia de un conjunto de datos es la suma de todos los contadores de inconsistencia dividido por el total de instancias del conjunto.

Esta medida es *monótona*, de forma que dados dos conjuntos de atributos S_1 y S_2, si $S_1 \subset S_2$, entonces, se puede asegurar que la tasa de inconsistencias de S_1 es siempre mayor o igual que la de S_2. Esta medida, usando tablas *hash*, tiene un coste de evaluación de orden lineal al número de instancias del conjunto [103]. Debemos tener en cuenta que los atributos deben ser discretos o estar discretizados.

Existe un problema al usar este criterio en bases de datos con un atributo que identifique individualmente cada instancia (DNI, número seguridad social,...) al no existir inconsistencia en los datos. Obviamente, este atributo sería irrelevante para los algoritmos de inducción. El problema se puede solucionar dejándolo fuera del proceso de selección si está identificado, o ejecutando una vez el algoritmo para identificarlo y posteriormente para elegir el subconjunto.

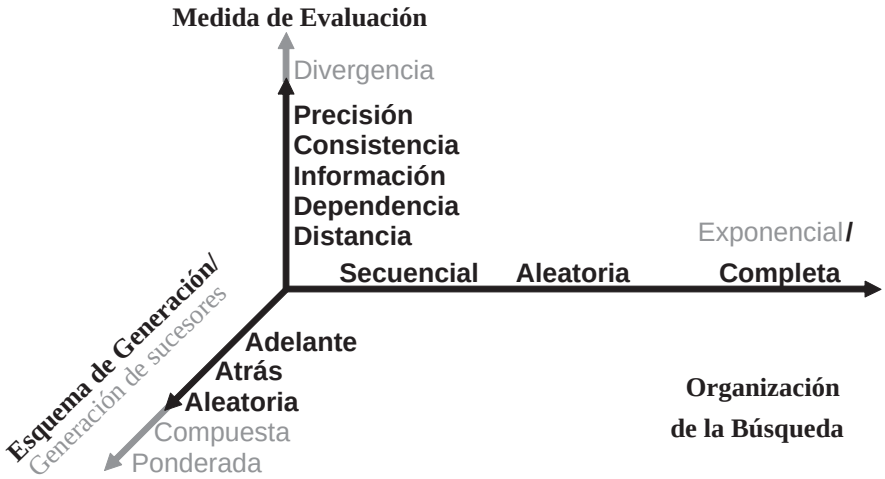


Figura 3.3: Principales dimensiones de la selección de atributos

Medidas de exactitud

En aprendizaje supervisado, el principal objetivo de un clasificador es maximizar la exactitud en la predicción de nuevos ejemplos, esto hace que la exactitud sea aceptada y muy utilizada como medida de evaluación. Como se seleccionan los atributos mediante la utilización del clasificador que posteriormente se emplea en la predicción de las etiquetas de la clase para los ejemplos desconocidos, el nivel de exactitud es alto, pero se añade un elevado coste computacional al algoritmo de selección.

Según las clasificaciones realizadas en los estudios de algoritmos comentados en el apartado 3.4.1, a los método que utilizan este tipo de medida se les denomina *envoltorios* (*wrappers*), y filtros a los demás.

3.4.4. Objetivos a optimizar

Los algoritmos de selección de atributos se pueden dividir en tres grupos conforme al objetivo final que se desea optimizar: 1) los algoritmos que buscan un subconjunto con un tamaño especificado que optimice un criterio de evaluación dado, 2) aquellos métodos que localizan el subconjunto de menor tamaño que satisfaga una cierta restricción sobre el criterio de evaluación, y 3) los que intentan encontrar un compromiso entre el tamaño del subconjunto y el valor de su criterio de evaluación.

3.5. Algoritmos de selección

Todo algoritmo de selección de atributos se puede representar en un espacio de características, principalmente, de acuerdo a su dirección de búsqueda, a la estrategia utilizada y a la medida de evaluación que se aplique (figura 3.3). En **negrita** destacamos los valores según lo descrito en los apartados 3.4.2 y 3.4.3, mientras que la figura completa representa la caracterización de los algoritmos según la propuesta hecha en [119], donde se contemplan dos direcciones más de búsqueda (compuesta y ponderada), se separa la divergencia de las medidas de distancia, y las estrategias de búsqueda se basan en los estudios realizado por Doak en [48] (exponencial, secuencial y aleatoria)

3.5.1. Descripción de los algoritmos

En la tabla 3.1 aparecen setenta y cinco algoritmos de selección de atributos extraídos de la bibliografía relacionada. Estos algoritmos son representativos de los distintos tipos de algoritmos de selección posible, y no son en absoluto todos los que existen. La selección de atributos es un área de investigación bastante activa y en continuo cambio, por lo que existen más algoritmos, versiones y variaciones de las que se reflejan en este estudio. Se clasifican por estrategia de búsqueda y por medida de evaluación. La búsqueda secuencial es la más utilizada junto a cualquier medida de evaluación, mientras que con las estrategias completa y aleatoria, se suelen emplear medidas de consistencia y exactitud como criterio de evaluación.

La clasificación de los algoritmos de selección en diferentes grupos, sirve de ayuda en distintos sentidos. En primer lugar, muestra relaciones entre los algoritmos, de manera que los métodos situados en el mismo bloque o categoría, son más parecidos entre ellos que el resto. En segundo lugar, si se pretende ver que algoritmo es el más recomendado para un cierta tarea, permite centrarse en estudiar aquellos localizados en ciertas categorías, reduciendo así el número de algoritmos a estudiar. Y en tercer lugar, se pueden observar bloques vacíos en la tabla, que podrían ser focos de nuevas investigaciones. A continuación se describe el comportamiento de los algoritmos en general. Teniendo en cuenta que la selección de atributos la definimos como un proceso de búsqueda, y que el nombre de muchos de los algoritmos que aparecen en la tabla lo toman de la búsqueda realizada, vamos a describir los algoritmos por orden de estrategia, siendo agrupados por estrategia y criterio de evaluación. En negrita se resalta el nombre del algoritmo que se va a describir, y si al algoritmo no se le reconoce nombre alguno, en este estudio se le denominará con el nombre del autor, si es individual, o con las dos primeras letras de cada autor, si es colectivo, seguido del año de publicación.

Tabla 3.1: Catalogación de los algoritmos de selección de atributos.

Medida de Evaluación	**Estrategia de Búsqueda**		
	Completa	**Secuencial**	**Aleatoria**
Distancia	B&B [122] BFF [178]	Relief [85] ReliefF [92] ReliefS [105] Segen [154] EUBAFES [152] C-SEP [58] SFS [127]	
Información	MLDM [157]	SG [44] KoSa96 [91] FCBF [184] DTM [26] IG–CBL [27] Callan91 [24] CR [169] MIFS [13] SeLi96 [155] YaMo99 [180] MRMR [47]	
Dependencia	Bobrowski [21]	CFS [70] POE+ACC [121] PRESET [118] DVMM [161]	
Consistencia	Focus [7, 8] ABB [104] Schlimmer93 [153] MIFES–1 [125]	VCC [171] SetCover [36]	LVF [108] LVI [109] LVS [110] QBB [38]
Exactitud	BS [48] AMB&B [63] IcSk84 [76, 77] DaRu94 [40]	WSG [44, 87] BSE-SLASH [28] BDS [48] PQSS [48] RACE [120] SeLi97 [156] RC [51] QuGe84 [129] Oblivion [99] IS [78, 168] Holte [74, 88] RFE [69]	LVW [107] GA [159, 181, 167, 67] RGSS [48] RMHC-PF [160] SA [48] FSS–EBNA [79]
Inf.+Exac.		BBHFS/BDSFS [34] XiJoKa91 [175]	
	Otros:	chi2 [106],Qifs [114],DIET [90]	

Completa–Distancia

En este grupo se encuentran algunos de los algoritmos de selección más antiguos, como los basados en *Branch and Bound* (B&B) y sus variantes.

El método **B&B** (*Branch and Bound*) propuesto por Narendra y Fukunaga [122], selecciona un subconjunto lo más pequeño posible, cuya evaluación esté por debajo del umbral establecido. La dirección de búsqueda es *hacia atrás*, y el método es una variación de la búsqueda en *profundidad*, donde el usuario da valor a un parámetro que se utiliza para limitar las ramas en las que buscar (poda). Para su correcto funcionamiento necesita que la medida de evaluación utilizada sea *monótona*. Las más utilizadas son: la distancia de Mahalanobis, la función discriminante, el criterio de Fisher, la distancia de Bhattacharya, y la divergencia.

En [178] se propone un algoritmo similar, **BFF** (*Best First Feature*), donde se modifica la estrategia de búsqueda para solucionar el problema de encontrar un camino óptimo en un árbol ponderado mediante la técnica de búsqueda *best first*. Este algoritmo garantiza globalmente el mejor subconjunto sin una búsqueda exhaustiva, para algún criterio que satisfaga el principio de *monotonicidad*.

Completa–Información

Bajo esta categoría está **MDLM** (*Minimum Description Length Method*) [157]. En este algoritmo, si los atributos en un subconjunto V se pueden expresar como una función independiente de la clase F de otro subconjunto de atributos U (U y V juntos completan el conjunto de atributos), entonces una vez se conocen los valores de los atributos en U, el subconjunto de atributos V no se necesita. Utiliza como criterio de evaluación *MDLC* (*minimum description length criterion*). El algoritmo busca exhaustivamente todos los posibles subconjuntos (2^n) y como salida muestra el subconjunto que satisface *MDLC*.

Completa–Dependencia

En [21], **Bobrowski** demuestra que el criterio de homogeneidad se puede usar como medida del grado de dependencia lineal entre algunas medidas, y muestra que se puede aplicar al problema de la selección de atributos debido a su principio de *monotonicidad*. Por tanto, puede convertirse en un algoritmo de selección mediante la implementación como criterio de evaluación en B&B con el procedimiento *backtracking* o *best first*.

Completa–Consistencia

En esta categoría se encuentra el algoritmo *Focus* con sus dos variantes y dos nuevas versiones del algoritmo B&B mencionado anteriormente.

Focus y **Focus–2** [7, 8, 9]. *Focus* empieza con el conjunto vacío y lleva a cabo una búsqueda exhaustiva en *anchura* (*breadth-first*) hasta encontrar un subconjunto mínimo consistente que prediga las clases puras. Este método rinde mejor cuando el número de atributos relevantes respecto al total es pequeño, en otro caso, su coste exponencial domina. Además, no maneja bien el ruido debido a su énfasis en la consistencia, no obstante, se puede permitir un cierto grado de inconsistencia. Para reducir el coste del algoritmo, en *Focus-2* se utiliza una cola donde se almacena una parte del espacio de búsqueda.

Schlimmer93 [153] y **MIFES–1** [125] son dos variantes de *Focus*. El algoritmo de *Schlimmer* utiliza un esquema de enumeración sistemático como procedimiento de generación y la consistencia como medida de evaluación. Emplea una función heurística para hacer más rápida la búsqueda del subconjunto óptimo. Esta heurística se fija en el porcentaje de valores poco frecuentes considerando un subconjunto de atributos. Si un subconjunto es poco fiable, todos los superconjuntos donde esté incluido, también. *MIFES–1* representa el conjunto de instancias como una matriz, y se dice que un atributo cubre a un elemento de la matriz si adopta valores opuestos para una instancia positiva y otra negativa.

ABB (*Automatic Branch and Bound*) [104] es una modificación de B&B donde el límite es determinado automáticamente. Empieza con todos los atributos, eliminando un atributo cada vez según una búsqueda *breadth-first* hasta que no se puedan eliminar más sin dejar de cumplir el criterio de consistencia. *ABB* expande y poda el espacio de búsqueda. Una rama se poda cuando no puede crecer al no satisfacer el criterio de consistencia. En [104] se concluye que si se sabe que el tamaño del subconjunto de atributos relevantes es pequeño, *Focus* es una buena opción, pero si no se conoce ese valor y el número de atributos del conjunto de datos es pequeño, *ABB* es una buena opción. En [39] se señala que toma mucho tiempo aún para un número moderado de atributos irrelevantes. En general, *ABB* expande el espacio de búsqueda rápidamente al principio pero su coste exponencial lo acaba haciendo prohibitivo.

Completa–Exactitud

En este grupo se encuentran algoritmos de tipo wrapper: *Beam Search*, *Approximate Monotonic Branch and Bound*, dos modificaciones de B&B basadas en diferentes clasificadores, y una variante del algoritmo *FOCUS*.

BS (*Beam Search*) [48] (ver apartado 3.4.2) es un tipo de búsqueda *best–first* que utiliza una

cola para limitar el alcance de la búsqueda. En la cola se almacenan en orden decreciente los subconjuntos que mejor tasa de aciertos han obtenido con el clasificador *1NN*. El proceso de generación se realiza tomando el subconjunto al frente de la cola (o de manera aleatoria [3, 4], teniendo mayor probabilidad los subconjuntos mejor situados en la cola) y genera todos los posibles subconjuntos añadiendo un atributo. Cada subconjunto se coloca en la posición que le corresponda en la cola. En ambos métodos, si no hay límite en el tamaño de la cola, la búsqueda es exhaustiva; si el límite de la cola es uno, equivale a una secuencial.

AMB&B (*Approximate Monotonic Branch and Bound*) [63] se creó para combatir los inconvenientes de B&B, permitiendo un criterio de evaluación que no sea *monótono*. En este algoritmo, el límite se relaja para permitir que aparezcan subconjuntos que de otra forma se podarían, pero que debido a la *no monotonía* del clasificador pueden dar lugar a combinaciones de atributos con mejor valor que el umbral. El subconjunto seleccionado tiene que estar dentro del límite.

Ichino y Sklanksy (**IcSk84**) [76, 77] desarrollaron dos algoritmos basados en dos clasificadores: *lineal* y *caja (box)*. El algoritmo de búsqueda es B&B utilizando el clasificador como criterio de evaluación. Los clasificadores cumplen la propiedad requerida para que B&B realice una búsqueda completa.

Davies y Russell (**DaRu94**) [40] proponen un algoritmo basado en *FOCUS* que mantiene una lista de todos los conjuntos de atributos de una determinada dimensionalidad, y en la cual se van eliminando aquellos conjuntos que dan lugar a un árbol de decisión que no clasifica correctamente los ejemplos utilizados. La búsqueda es exhaustiva, en el sentido de que cuando la lista está vacía se vuelve a inicializar con todos los conjuntos de dimensionalidad superior. Por tanto, en el peor caso, se llega a la lista que contiene un único conjunto con todos los atributos.

Secuencial–Distancia

En esta categoría están contempladas diferentes versiones del algoritmo *Relief*.

Relief [85] es un algoritmo inspirado en el aprendizaje basado en casos que intenta obtener los atributos más relevantes estadísticamente. El algoritmo se basa en asignar un peso a cada atributo y seleccionar los atributos cuyo peso supera un umbral prefijado. Para este propósito, dada una instancia, *Relief* busca en el conjunto de datos dos vecinos, el más cercano de la misma clase (*near–hit*) y el de clase distinta (*near–miss*). El peso asociado a un atributo se modifica a partir de la distancia euclídea entre el valor del atributo de la instancia y el valor del mismo atributo de los vecinos encontrados. Las instancias se escogen aleatoriamente del conjunto de datos un número determinado de veces. Cuando el número de instancias de la base de datos es

pequeño, se realiza el proceso para cada una de las instancias, no existiendo aleatoriedad, por eso se incluye en esta grupo de búsqueda secuencial. En la bibliografía consultada, este método no se encuentra catalogado siempre igual, como en [119], donde la organización de la búsqueda aparece como aleatoria. *Relief* favorece a los atributos correlacionados sobre los relevantes [37], por lo que no está garantizado un conjunto óptimo [85]. En [28] se concluye que el problema con *Relief* se debe al uso de la distancia como criterio de consistencia. [92] propone una versión mejorada (**ReliefF**) en la que se seleccionan las k instancias más parecidas (de la misma y de diferente clase, respectivamente) y se toma su media.

ReliefS [105] es una versión soportada en árboles *kd–tree*, y para el cálculo de la distancia, se escoge un sólo ejemplo de los posibles dentro de cada hoja del árbol. **EUBAFES** [152] es un algoritmo similar pero con una función diferente para asignar pesos a los atributos. El algoritmo de **Segen** [154] utiliza un criterio de evaluación que minimiza la suma de una medida de discrepancia estadística y de una medida de complejidad del atributo. Encuentra el primer atributo que mejor distingue las clases, e iterativamente busca atributos adicionales que en combinación con el/los anteriores mejoren la información de la clase. El proceso se termina cuando se alcanza el criterio de representación mínimo.

Fayyad e Irani [58] proponen un tipo de medidas denominada **C-SEP** para la inducción de árboles de decisión. Estas medidas intentan maximizar la distancia entre las instancias pertenecientes a clases diferentes y la coherencia entre los correspondientes a la misma. Un ejemplo de esas medidas que se recoge en su trabajo, es el coseno del ángulo que forman los vectores de clase. Estos vectores recogen la frecuencia de ocurrencia de ejemplos de cada clase en el conjunto de aprendizaje. El método se basa en realizar una partición por un atributo en dos conjuntos, y obtener los vectores de clases de cada partición. Si la partición es perfecta (en cada partición sólo existen ejemplos de una clase) los vectores de clases son ortogonales y su coseno es mínimo; sin embargo si el atributo es irrelevante las dos particiones tendrán aproximadamente igual número de ejemplos de cada clase, los vectores de clase serán prácticamente paralelos y por tanto, la medida será máxima.

SFS (*Sequential Floating Search*) [127] que recoge las dos versiones, *hacia adelante* y *hacia atrás* (**SFFS** y **SBFS**) es un algoritmo secuencial *bidireccional*. A cada paso de selección de atributos, se realiza un paso hacia adelante añadiendo un atributo, y un paso hacia atrás, donde se suprime aquel atributo cuya ausencia hace que mejore el resto del subconjunto escogido hasta el momento (puede que no se elimine ningún atributo). Esto permite descubrir nuevas interaciones entre los atributos. Este método permite utilizar cualquier criterio de evaluación, aunque el más usual es la distancia de Bhattacharyya. De acuerdo con los estudios comparativos de [82], probablemente *SFFS* es el método subóptimo más efectivo. En [81] se concluye que los

métodos *SFFS* se desempeñan casi tan bien como el algoritmo B&B y demanda pocos recursos de cómputo. En [93, 95] se señala que este método da mejores soluciones que otros algoritmos de búsqueda secuenciales en un tiempo razonable para problemas de pequeña y mediana escala. La desventaja principal es que debe especificarse el tamaño deseado del subconjunto solución.

Secuencial–Información

El algoritmo **SFG** (*Sequential Forward Generation*) [48] empieza con el conjunto *S* vacío, y se añaden atributos uno a uno desde el conjunto original. En cada iteración, se escoge el mejor atributo del conjunto original según el criterio de información, junto con los atributos previamente seleccionados (*S*). Este algoritmo puede proporcionar un ranking, donde los atributos se encuentran situados según el orden de inclusión en el conjunto *S*. En **SBG** (*Sequential Backward Generation*) se parte del conjunto original como solución y se van eliminando los atributos menos relevantes. Estos algoritmos se corresponden con los descritos en el apartado sobre las direcciones de búsqueda (ver 3.4.2 *hacia adelante y hacia atrás*). Doak muestra que cuando el número de atributos es pequeño, en la mayoría de las ocasiones, *SBG* rinde mejor que *SFG*, tal vez porque *SBG* evalúa la contribución de un atributo con respecto a todos. En [4] se contradice lo anterior, indicando la preferencia por *SFG* cuando el número de atributos relevantes es pequeño, y *SBG* en caso contrario. En la comparativa realizada en [82] se resalta que *SFG* es más rápido en general, pero se comportan de manera similar.

Dentro de este apartado se encuentra la aplicación de métodos de inducción de árboles como algoritmos de selección de atributos. **DTM** (*Decision Tree Method)* [26], una modificación posterior [27] (**IG–CBL**) y **Callan91** et. al. en [24] ofrecen como resultado el subconjunto de atributos utilizados para formar el árbol.

Koller y Sahami (**KoSa96**) [91] aplican una técnica basada en la entropía cruzada, denominada *Markov blanket filtering*, para eliminar los atributos redundantes (ver definición en sección 3.3).

El algoritmo **FCBF** (*Fast Correlation-Based Filter*) creado por Yu y Liu [184], se basa en el concepto de *Markov blanket* (*M*) antes comentado. En este caso *M* está formado por un sólo atributo, de manera que empezando por el primer atributo de una lista ordenada por su correlación no lineal con respecto a la clase, se eliminan progresivamente los atributos redundantes con *M*. Una vez eliminados los del primer atributo, *M* será el siguiente de la lista, siguiendo el mismo procedimiento, y así hasta el final de la lista. Se descartarán aquellos atributos cuya correlación no lineal con el atributo que forme *M* sea superior a su correlación no lineal con la clase. Este método es muy rápido, pero sus resultados dependen en gran medida de un parámetro que se utiliza para analizar sólo aquellos atributos más correlacionados con la clase. Si

posee un valor muy pequeño, normalmente se obtienen grandes subconjuntos que no contienen atributos redundantes y son relevantes con respecto a la clase. Si se desea obtener subconjuntos más pequeños, se disminuirá bastante su poder predictivo.

El algoritmo **MIFS** [13] aplica *información mutua* para seleccionar un subconjunto de atributos que será la entrada de un clasificador mediante *redes neuronales*. El algoritmo se basa en calcular la información mutua de cada atributo con la clase y entre cada par de atributos. En el algoritmo se selecciona inicialmente el atributo con mayor información mutua con la clase y luego se van añadiendo atributos al conjunto de los ya seleccionados. El algoritmo termina cuando se ha seleccionado un número predeterminado de atributos. Un esquema bastante similar al anterior es el debido a Setiono y Liu [155] (**SeLi96**). Yang [180] (**YaMo99**) es otra propuesta similar pero de selección de características para visualización.

Wang [169, 170] propone el algoritmo **CR** para la selección de atributos que utiliza una medida de relevancia entre el conjunto de atributos y la clase basada en conceptos de Teoría de la Información, según la definición 3.5.

Ding y Peng [47] utilizaron la información mutua para seleccionar subconjuntos de genes con máxima relevancia y mínima redundancia, mediante la resolución de un problema de optimización con dos objetivos. El algoritmo, denominado **MRMR** (*Minimum Redundancy Maximum Relevance*), selecciona en primer lugar el atributo con mayor información mutua con respecto a la clase, y el resto de atributos son seleccionados de manera incremental. Se añadirá aquel atributo que obtenga mejores resultados en los dos objetivos, es decir, que maximice la información mutua con respecto a la clase, y minimice la información mutua con respecto a los genes que ya están incluidos en el subconjunto candidato.

Secuencial–Dependencia

CFS (*Correlation–based Feature Selection*) [70] intenta obtener el conjunto de atributos más correlacionado con la clase y con menos correlación entre sí. Se le puede asociar con distintas técnicas de búsqueda, siendo *Best First* la más utilizada.

POE-ACC [121] (*Probability of Error & Average Correlation Coefficient)*. En este algoritmo, el primer atributo elegido es el atributo con la menor probabilidad de error (POE). El próximo atributo elegido es el atributo que produce la mínima suma ponderada de POE y de ACC (coeficiente de correlación medio). ACC es la media de los coeficientes de correlación de los atributos candidatos con los atributos seleccionados hasta ese instante. Este algoritmo puede generar un ranking basándose en la suma ponderada, y como criterio de parada se requiere un número de atributos. **PRESET** [118] utiliza el concepto de *rough set*.

Secuencial–Consistencia

El método **Set Cover** [36] se basa en que la probabilidad de encontrar el conjunto más pequeño de atributos consistente es equivalente a cubrir cada uno de los ejemplos que tienen diferentes etiqueta, con algún atributo con diferentes valores para ese caso.

Wang [171] se basa en el *Vertical Compactness Criterion* (**VCC**) que realiza la compactación vertical del conjunto de aprendizaje basándose en las definiciones de inconsistencias (ver apartado 3.4.3). El subconjunto que se selecciona es el de menor dimensionalidad que no supere un umbral fijado por el usuario. El procedimiento de búsqueda que emplean es un híbrido entre la búsqueda en profundidad y en anchura.

Secuencial–Exactitud

WSFG (*Wrapper Sequential Forward Generation*) y **WSBG** (*Wrapper Sequential Backward Generation*) [44, 87] son los algoritmos *SFG* y *SBG* con la precisión de un inductor externo como criterio de evaluación.

Caruana y Freitag [28] realizan una comparativa de diferentes métodos heurísticos de selección de atributos en dos problemas obtenidos del sistema *CAP* (*Calendar Apprentice*). En dicho trabajo, la calidad de los atributos seleccionados se mide como la estimación del error de los árboles de decisión generados con *ID3/C4.5* mediante la técnica *holdout*. Los métodos comparados son búsquedas secuenciales hacia adelante y atrás, incluyendo vuelta atrás. Una modificación de la búsqueda secuencial hacia atrás es la denominada **BSE-SLASH** (*Backward Stepwise Elimination*) donde en cada paso se eliminan todos los atributos no utilizados en el árbol generado por el algoritmo ID3. De los métodos comparados, los que incluyen búsqueda bidireccional dan los mejores resultados aunque sin mucha diferencia con los otros.

Doak en [44], presenta **BDS** (*Bidirectional Search*) que realiza una búsqueda desde ambos extremos, mientras que el algoritmo **PQSS** (*(p,q) Sequential Search*) proporciona cierto grado de backtracking permitiendo añadir y eliminar atributos para cada subconjunto. Si *PQSS* empieza desde el conjunto vacío, añade más atributos que descarta en cada paso, y si empieza con el conjunto completo de atributos, entonces descarta más atributos y añade menos en cada paso.

Una aproximación basada en el clasificador del vecino más cercano es el algoritmo **RACE** [120] que utiliza una heurística de búsqueda basada en la competición de diferentes conjuntos de atributos. Puede comenzar por el conjunto vacío o por el completo, y en cada iteración obtiene el mejor subconjunto. Se estima el error por validación cruzada dejando uno fuera y continúa hasta que no exista mejora.

El algoritmo desarrollado por Queiros y Geselma (**QuGe84**) [129] es similar a *WSFG*, pero

sugiere que en cada iteración y para cada atributo se consideren diferentes interacciones con el conjunto de atributos previamente seleccionado. Por ejemplo: asumir siempre independencia de los atributos (no considerar los atributos previamente seleccionados), y nunca asumir independencia (considerar los atributos anteriormente elegidos). Se utiliza la tasa de error del clasificador Bayesiano.

El error de clasificación en una red neuronal es utilizado por Setiono y Liu [156] (**SeLi97**) para realizar la selección de atributos. La red neuronal es de tres capas completamente interconectada y una vez entrenada con todos los atributos, se procede a calcular el resultado de la red eliminando todas las conexiones de un determinado atributo con el nivel oculto. El atributo con el que la red decrementó menos su rendimiento es seleccionado y el proceso se repite sucesivamente mientras el rendimiento de la red no disminuya más de un cierto umbral respecto al rendimiento con todos los atributos.

El algoritmo **IS** (*Importance Score*) [78, 168] consiste en la ordenación de los atributos en base a la medida de dependencia. A partir de esta ordenación se obtienen aquellos atributos que inducen el conjunto de reglas con menor error. El cálculo de la medida *IS* para cada atributo se realiza después de generar un conjunto de reglas de decisión con el algoritmo *AQ* utilizando todos los atributos. A partir de este conjunto inicial de reglas, el valor *IS* de un atributo se obtiene en función del número de reglas que contienen a dicho atributo, de forma que para un atributo que no aparece en ninguna regla el valor *IS* sería nulo mientras que sería uno si está en todas las reglas. Una vez se computa el valor *IS* para cada atributo, se seleccionan inicialmente aquellos que superen un umbral (redondeo superior de la raíz cuadrada del número de atributos) para el valor *IS*. Con estos atributos se genera un conjunto de reglas cuyo error fijaría el umbral para la adición de otros nuevos atributos, ya que se irán añadiendo sucesivamente y generando conjuntos de reglas mientras la tasa de error de las reglas generadas no incremente la tasa de error inicial.

Holte [74] también propone un método, denominado **1-R**, basado en la tasa de error de las reglas generadas a partir del conjunto de atributos. A diferencia del algoritmo *IS* las reglas tienen un solo atributo y selecciona aquel cuya regla asociada produce un menor error. Los resultados obtenidos con el algoritmo *1-R* no son mucho peores en tasa de error a los que se obtienen con más atributos y reglas generadas con un algoritmo más sofisticado como el *C4.5*.

Kohavi y Frasca [88] introduce el clasificador **Holte-II** de selección de atributos basado en la propuesta de Holte. La clasificación se realiza mediante la búsqueda del ejemplo a clasificar en el conjunto de aprendizaje, asignándole la misma clase si se encuentra, si no se encuentra se le asigna la clase que es mayoritaria en dicho conjunto. En lugar de utilizar un solo atributo en el clasificador *Holte-II* se realiza una selección de atributos y los resultados son comparados

por los obtenidos con el uso del *C4.5*, llegando a similares conclusiones que Holte.

Otro método basado en la búsqueda secuencial hacia atrás es **OBLIVION** [99] en el que se utiliza como clasificador el árbol *oblivious*. Éste es un árbol de decisión donde todos los nodos en un determinado nivel hacen referencia al mismo atributo, por lo que se obtienen todas las posibles combinaciones de valores de los atributos. Una vez generado el árbol con todos los atributos, se procede a una poda de aquéllos atributos que no influyen en la clasificación. Los autores de *OBLIVION* encontraron que un árbol *oblivious* es equivalente a un clasificador del vecino más cercano en el que se eliminan ciertos atributos en el cálculo de la distancia. En *OBLIVION* se prefiere la búsqueda hacia atrás, para evitar la eliminación de un atributo que se encuentra interrelacionado con otros, razón por la cual en el trabajo de Caruana y Freitag el método *BSE-SLASH* da resultados similares a los de búsqueda secuencial hacia atrás o los que incluyen vuelta atrás.

El vecino más cercano y la búsqueda secuencial hacia atrás se emplea también en el método propuesto por Domingos **RC** [51], aunque a diferencia de *OBLIVION*, la decisión para la eliminación se realiza utilizando información local a cada ejemplo del conjunto de entrenamiento. Así para cada ejemplo se eliminan aquellos atributos que son diferentes a los del ejemplo más cercano de la misma clase. Si el error del clasificador utilizando el ejemplo sin estos atributos disminuye, se eliminan definitivamente y el proceso continúa. Si el error no disminuye no se eliminan los atributos y se repite el proceso con otro ejemplo no utilizado previamente, hasta que se han utilizado todos los ejemplos.

RFE (*Recursive Feature Elimination*) es un método de selección de genes propuesto por Guyon et al. [69]. Este algoritmo intenta escoger aquel subconjunto, con una dimensión preestablecida, que lleve a un margen de separación de clases mayor. Para ello se utiliza un clasificador **SVM** (*Support Vector Machine o máquina de soporte vectorial*). Este problema combinatorio se resuelve mediante una búsqueda secuencial hacia atrás, de manera que en cada iteración se elimina el atributo que menos decrementa el margen de separación. Para hacer el proceso más rápido, se pueden eliminar bloques (porcentajes) de atributos. El proceso termina cuando el conjunto de datos se reduce al número de atributos prefijado.

Secuencial–Información+Exactitud

Los algoritmos que se comentan a continuación combinan las ventajas de los algoritmos filtros y wrappers. Como los filtros, son rápidos y generales, mientras que al mismo tiempo emplean el conocimiento de un algoritmo de aprendizaje para guiar la búsqueda y proporcionar un criterio de parada.

BBHFS (*Boosting Based Hybrid Feature Selection*), **BDSFS** (*Boosted Decision Stump Fea-*

ture Selection) y **BDSFS-2** [34] utilizan el concepto de boosting de la teoría de aprendizaje computacional, y evalúan la bondad del subconjunto de cardinalidad K mediante el criterio de ganancia de información. En vez de definir el número de atributos a seleccionar, este algoritmo calcula la exactitud del algoritmo de aprendizaje como criterio de parada. El algoritmo de aprendizaje se aplica a cada nuevo subconjunto de cardinalidad k, si la precisión no aumenta comparándola con la precisión obtenida con *k-1* atributos, el algoritmo termina sin añadir el último atributo.

En Xing et al. [175] se propuso un método orientado a bases de datos genómicas compuesto de tres fases. En una primera fase se discretizan los atributos (genes) en dos posibles valores («on» y «off»). En la segunda, se calcula la ganancia de información de cada atributo, se genera un lista en ordenada, y se eliminan los que peor resultado ofrecen. En la última fase se aplica una técnica más intensiva propuesta por Koller y Sahami [91], denominada *Markov blanket filtering*(ver 3.9), para seleccionar un subconjunto de atributos. Los subconjuntos con diferentes cardinalidades se comparan mediante validación cruzada, la cual es más costosa que la precisión del conjunto de entrenamiento.

Aleatorio–Consistencia

Liu y Setiono [108] proponen el algoritmo **LVF** (*Las Vegas Filter*) basado a su vez en el algoritmo de búsqueda aleatoria *Las Vegas* [22]. Este algoritmo realiza saltos aleatorios en el espacio de búsqueda para permitir la localización del subconjunto de atributos más rápidamente. El uso de la aleatoriedad en este algoritmo es tal que la solución siempre se encuentra aunque se realicen elecciones erróneas a costa de más tiempo de cómputo. El algoritmo *LVF* consiste en generar aleatoriamente conjuntos de atributos e ir seleccionando en cada momento aquel que tenga el menor número de atributos y cuyo promedio de inconsistencia sea menor que el umbral fijado por el usuario. Dado que es un proceso aleatorio, un parámetro importante a fijar es el número de subconjuntos que se comprueban. Si este número es muy bajo es improbable obtener el mejor subconjunto, mientras que si es muy alto se realizarán muchas comprobaciones sobre subconjuntos después de encontrar el mejor. En el trabajo mencionado, este valor se ha establecido de forma experimental en $77 \times n^5$ (n número inicial de atributos). El otro elemento que define la calidad del subconjunto seleccionado es el promedio de inconsistencias sobre el número total de muestras. Los autores indican que utilizando mecanismos de indexado se puede calcular el promedio de inconsistencia con un coste computacional de $O(n)$.

Cuando las bases de datos son muy grandes, comprobar si contiene inconsistencias lleva mucho tiempo. **LVI** (*Las Vegas Incremental*) [109] o **LVS** (*Las Vegas Scalable*) [110] son versiones de *LVF* que permiten seleccionar los atributos sin utilizar inicialmente todos las muestras,

ya que en problemas con un gran número de muestras el coste del cálculo del promedio de inconsistencia puede ser alto. Para ello divide el conjunto inicial de muestras en dos, realiza el proceso de selección como en *LVF* en una de las muestras y luego comprueba en la segunda. Si existen inconsistencias, las muestras que las producen se mueven a la primera y se repite el proceso hasta que con los atributos seleccionados no se produzcan más inconsistencias.

Generalmente, *LVF* al principio proporciona resultados rápidamente, y a continuación los mejora de forma muy lenta. El algoritmo **QBB** (*Quick Branch and Bound*) [103] soluciona este problema combinando los algoritmos *LVF* y *ABB*. Comienza utilizando *LVF* para generar un conjunto de atributos que se utiliza como conjunto inicial en el algoritmo *ABB* y que refina la búsqueda realizada por *LVF*.

Aleatorio–Exactitud

El algoritmo **LVW** (*Las Vegas Wrapper*) [107] es una variante de *LVF*, tomando la precisión de un clasificador como medida de evaluación. No es recomendable usar este algoritmo en aplicaciones donde el tiempo sea un factor crítico.

Siedlecki y Slansky [159] proponen la utilización de un algoritmo genético (**GA**) para el recorrido del espacio de búsqueda. La función de ajuste usada en el algoritmo genético propuesto, se basa en obtener el subconjunto de atributos que tenga menor cardinalidad y con una tasa de error inferior a un umbral fijado por el usuario. Este objetivo se consigue haciendo intervenir en la función de ajuste la tasa de error del clasificador *5-NN*, el umbral y la cardinalidad del subconjunto, de forma que los subconjuntos con una tasa de error superior al umbral son penalizados independientemente del número de atributos. En Vafaie y DeJong [167], Yang y Honavar [181] y Guerra–Salcedo et al. [67] también se utiliza un algoritmo genético para recorrer el espacio de búsqueda.

Aparte de los algoritmos genéticos utilizados en los trabajos anteriores, existe un tipo de algoritmos denominados *EDA* (*Estimation Distribution Algoritms*) [100] con un funcionamiento similar a los algoritmos genéticos pero sin la utilización de los operadores de cruce o mutación. En este, las sucesivas generaciones se obtienen a partir de individuos de la población actual, con cada individuo de la nueva generación representado en función de su distribución de probabilidad en la generación actual. Para estimar la distribución de probabilidad en cada población se utiliza una *Red Bayesiana* que tiene en cuenta la interdependencia entre los atributos utilizados dando lugar a lo que los autores denominan *EBNA* (*Estimation of Bayesian Network Algorithm*). La combinación del algoritmo *EBNA* (utilizando como población las combinaciones de características de la misma forma que se utiliza en los algoritmos genéticos) con un clasificador da lugar al algoritmo **FSS-EBNA** [79]. En el artículo mencionado se utilizan varios clasificado-

res como un árbol de decisión o el clasificador bayesiano, considerando que se ha encontrado el subconjunto de atributos relevantes cuando no hay un incremento en la tasa de acierto del clasificador utilizado en una determinada generación.

Otro método basado en el recorrido aleatorio del espacio de búsqueda, aunque no haciendo uso de los algoritmos genéticos es el algoritmo **RMHC-PF** [160]. Este algoritmo se basa en la búsqueda *Random Mutation Hill Climbing (RMHC)*, que consiste en partir de un conjunto aleatorio de atributos y de forma aleatoria añadir uno nuevo o eliminar alguno de los contenidos en el conjunto e ir conservando el subconjunto que menor error produce con el clasificador del vecino más cercano.

En **RGSS** (*Random Generation plus Sequential Selection*) [48] se añade generación aleatoria de subconjuntos en los algoritmos secuenciales (*WSFG* y *WSBG*).

En este grupo, todos los algoritmos dependen del valor asignado a sus diferentes parámetros. El número máximo de iteraciones es común a la mayoría de los algoritmos, y además, para *LVW* el umbral de inconsistencia; y para *GA*, tamaño de la población inicial, tasa de cruce y de mutación.

Métodos no encuadrados en la clasificación Anterior

Existen algunos métodos y algoritmos en la literatura sobre aprendizaje automático que no se basan en las medidas vistas anteriormente o son diseñados para problemas de aprendizaje muy específicos. En este apartado se presentarán algunos de estos métodos.

Liu y Setiono [106] (**Chi2**) proponen un método de selección de atributos basado en la discretización de los atributos numéricos. Antes de la selección se procede a discretizar los atributos numéricos mediante una variación del método ChiMerge (Kerber, 1992). Al final del proceso de discretización se eliminan los atributos que han dado lugar a un solo intervalo.

QIFS (*Query–learning-based Iterative Feature–subset Selection*) [114] es un método destinado a bases de datos de gran dimensionalidad, que se basa en la idea denominada *Query by Committee* que selecciona instancias que maximizan la ganancia de información.

DIET [90] es un algoritmo que pondera los atributos para el cálculo de la distancia utilizada en el clasificador del vecino más cercano. Los pesos, a diferencia de otros trabajos comentados anteriormente, no utilizan ningún tipo de medida, definiendo el usuario el número k de pesos que pueden tener los atributos y generándose un conjunto de pesos discretos.

3.5.2. Clasificación de los algoritmos

Las clasificaciones más comunes que podemos encontrar en los trabajos relacionados con la selección de atributos, o en aquellos en los que se haga alguna referencia a la selección, son en función de:

La disponibilidad de la clase: supervisado o no, dependiendo de si se utiliza las etiquetas de la clase en el proceso de selección.

La estrategia de búsqueda: completa/exponencial, secuencial y aleatoria, descritas en el apartado 3.4.2.

La medida de evaluación: filtro y envoltorio (wrapper). El modelo filtro evalúa los atributos de acuerdo con heurísticas basadas en características generales de los datos e independientes del método de clasificación a aplicar, mientras que el modelo *wrapper* utiliza el comportamiento de un algoritmo de clasificación como criterio de evaluación de los atributos (ver apartado 3.4.3).

En el apartado 2.4.1 se definió Γ como la precisión de un clasificador $\mathcal{L}$ sobre un subconjunto de atributos. Además de un clasificador (medida wrapper) para evaluar un subconjunto de atributos podemos usar una medida tipo filtro que también devuelve un valor real sobre la bondad de dicho subconjunto. Este valor que denominaremos valor de medición se puede definir de la siguiente manera:

Definición 3.10 *Sea $\mathcal{E}$ un conjunto de datos etiquetados; $\mathcal{S}$ un subconjunto de atributos de los datos $\mathcal{E}$; se denomina* valor de medición $\Upsilon(\mathcal{E}/\mathcal{S}, Ł)$ *al resultado de aplicar el evaluador tipo filtro de subconjuntos* Ł *teniendo en cuenta sólo el subconjunto de datos $\mathcal{S}$.*

Por tanto, si Ł es un evaluador de subconjuntos, podemos usar dos expresiones semejantes: por un lado $\Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L})$, válida para evaluar a un subconjunto mediante un clasificador (medida *wrapper*), por otro lado la expresión $\Upsilon(\mathcal{E}/\mathcal{S}, Ł)$, que evalúa un subconjunto $\mathcal{S}$ mediante un filtro Ł.

Por comodidad y para unificar la notación, a partir de ahora la evaluación de un subconjunto de atributos $\mathcal{S}$ se notará mediante $\Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L})$. De esta forma, Γ asumirá un doble papel: cuando $\mathcal{L}$ sea un clasificador será la definida en 2.9 y cuando $\mathcal{L}$ sea un filtro será la Upsilon de la definición 3.10.

La salida del algoritmo: subconjunto de atributos o ranking. Los métodos de ranking de atributos (*FR*), también denominado *feature weighting* [20, 68], asignan pesos a los atributos individualmente y los ordenan basándose en su relevancia con respecto al concepto destino o atributo clase, mientras que los algoritmos de selección de subconjunto de atributos (*FSS*) evalúan la bondad de cada uno de los subconjuntos candidatos. (Ocasionalmente, algunas estrategias en combinación con la evaluación de subconjuntos pueden proporcionar una lista ordenada de atributos).

El Algoritmo 3.1 es un ejemplo de elaboración de un ranking. La idea es evaluar cada atributo individualmente según un criterio ($evaluar(x_i, U)$), e insertarlo en orden en una lista según el valor obtenido. El coste computacional es $O(n \times m + n^2)$ donde n es el número de atributos y m el número de ejemplos: O(m) para evaluar un atributo; O(n) para situar el atributo en la posición que le corresponde según su evaluación; y el bucle *para* se repite n veces. Existen muchas variantes de este algoritmo, teniendo en común la salida de una lista de atributos ordenada.

Algoritmo 3.1 Ranking de Atributos.

Entrada: $\mathcal{E}$ - conjunto de entrenamiento, U - medida de evaluación
Salida: L lista de atributos con el más relevante en primer lugar

```
L ⇐ {}
para i = 1 hasta n hacer
   v_i ⇐ evaluar(X_i, U)
   posicionar x_i en L con respecto a v_i
fin para
```

El Algoritmo 3.2 es un ejemplo de como obtener un subconjunto de atributos. En general, devuelve un subconjunto mínimo de atributos, entre los cuales no se hace diferencia en cuanto a la relevancia. La función $generarSubconjunto(x)$ devuelve un subconjunto de atributos (siguiendo el proceso de generación comentado anteriormente en 3.4.2) para que sea evaluado. Este proceso se repite hasta que se cumpla uno de los criterios de parada (un número de atributos determinado, un número de iteraciones o que no se mejore al subconjunto anterior). El coste computacional dependerá de las tres dimensiones (figura 3.3: dirección de búsqueda, estrategia utilizada y medida de evaluación) intervinientes en el proceso de selección.

Teniendo en cuenta la distinción entre algoritmos de selección que se acaba de hacer, se formaliza el concepto de ranking de atributos.

Algoritmo 3.2 Subconjunto de Atributos.

Entrada: $\mathcal{E}$ - conjunto de entrenamiento, U - medida de evaluación
Salida: S - subconjunto de atributos

```
S ⇐ {}
repetir
  S_k ⇐ generarSubconjunto(x)
  si existeMejor(S, S_k, U) entonces
    S ⇐ S_k
  fin si
hasta criterio de Parada
```

Definición 3.11 *[Ranking de Atributos] Sea $\mathcal{E}$ un conjunto con m ejemplos y $\mathcal{X}$ el conjunto de sus n atributos $\{X_1, X_2, \ldots, X_n\}$. Sea r una función $r : \mathcal{X}_{\mathcal{E}} \rightarrow \mathbb{R}$ que asigne un valor de evaluación a cada atributo $X_i \in \mathcal{X}$ de $\mathcal{E}$. Un ranking de atributos es una función* F *que asigna un valor de evaluación (relevancia) a cada atributo y devuelve una lista de los atributos ($X_i^* \in \mathcal{X}$) ordenados por su relevancia, con $i \in \{1, \ldots, n\}$:*

$$F(\{X_1, X_2, \ldots, X_n\}) =< X_1^*, X_2^*, \ldots, X_n^* >$$

donde $r(X_1^*) \geq r(X_2^*) \geq \cdots \geq r(X_n^*)$.

Por convención, se asume que un valor alto es indicativo de un atributo relevante y que los atributos están en orden decreciente según $r(a^*)$. Se considera definido el criterio de ordenación para atributos individuales, independientes del contexto de los demás, y nos limitaremos a criterios de aprendizaje supervisado.

3.6. Evaluación y comparación

En este trabajo de investigación, la selección de atributos se enmarca dentro del aprendizaje supervisado, más concretamente, la clasificación. Por ello, para evaluar el rendimiento de un algoritmo de selección se comprueba el resultado del algoritmo de aprendizaje antes y después de la reducción. Los test de hipótesis muestran si el cambio producido en el comportamiento de un clasificador, con los datos originales y los reducidos, es estadísticamente significativo. El comportamiento del clasificador vendrá dado por: la precisión predictiva, la complejidad del modelo generado, el número de atributos seleccionados, y por el tiempo necesario para la ejecución del método. Estas mismas medidas se utilizan para comparar diferentes algoritmos de selección.

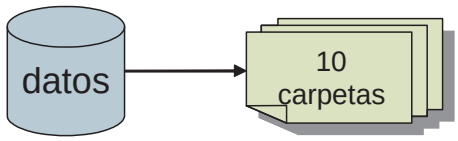


Figura 3.4: División de los datos en carpetas.

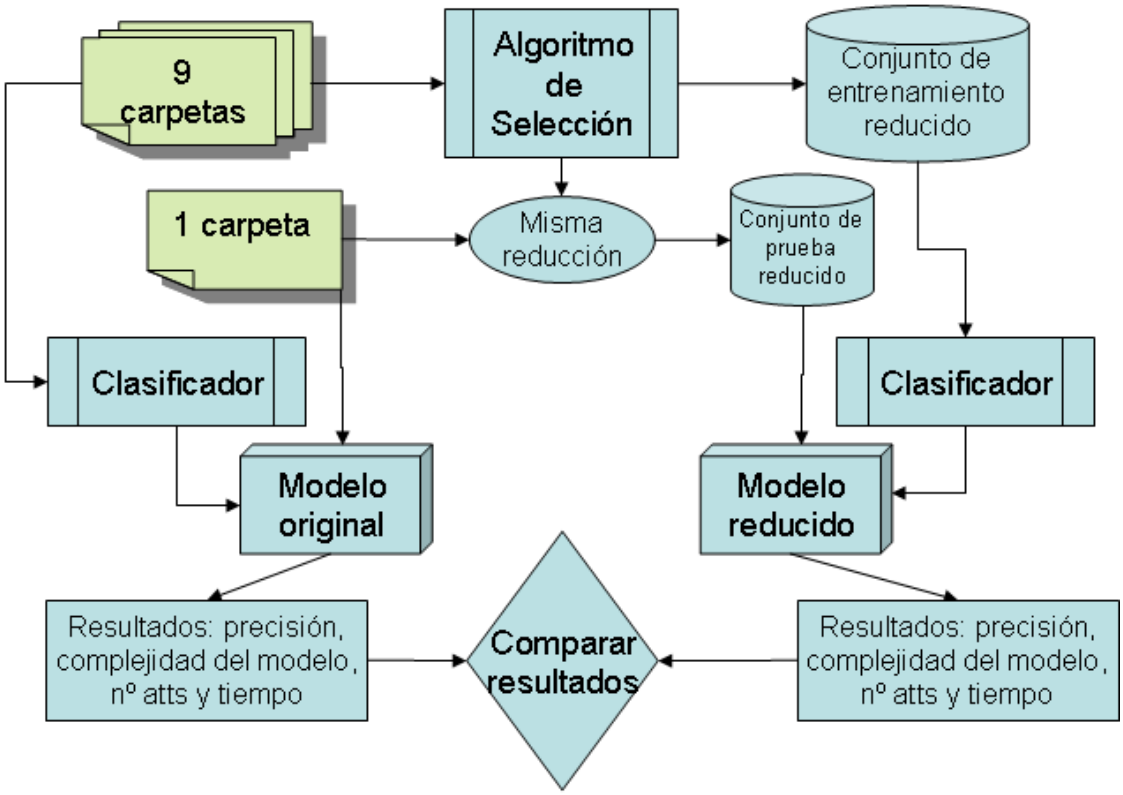


Figura 3.5: Proceso para validar los resultados al aplicar algoritmos de selección de atributos.

Es frecuente estimar las medidas anteriores como la media de la validación cruzada con k particiones, repitiendo la validación un número determinado de veces (ver apartado 2.4.2). Para que no se produzca un sobreajuste del algoritmo de selección a los datos utilizados, se realizan k reducciones, una por cada conjunto de entrenamiento. Las Figuras 3.4 y 3.5 son ilustrativas del proceso a realizar suponiendo una validación cruzada con diez particiones (10-cv). Se divide el conjunto de datos en diez bloques, carpetas o subconjuntos de ejemplos de tamaño similar (figura 3.4). Como se ha escogido validación cruzada diez, se realizarán diez evaluaciones. En cada evaluación se utilizan nueve carpetas para formar un conjunto de entrenamiento y una carpeta de prueba (ver figura 3.5), de manera que se emplee como prueba una carpeta distinta en cada evaluación. Del proceso de selección se obtiene un conjunto de entrenamiento reducido sólo con los atributos escogidos por el algoritmo. Esta misma reducción se le practica a la carpeta

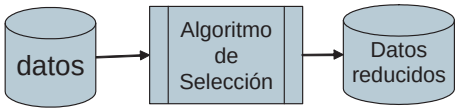


Figura 3.6: Reducción de un conjunto de datos.

de prueba obteniendo un conjunto de prueba reducido. A continuación, se obtiene un modelo de clasificación con los ejemplos pertenecientes al conjunto de entrenamiento reducido, que se utiliza para clasificar los ejemplos del conjunto de prueba reducido. Este proceso de validación se aplica a los distintos algoritmos de selección y se comparan los resultados obtenidos con cada modelo reducido. Además, con los ejemplos del conjunto de entrenamiento original formado por las nueve carpetas, se obtiene otro modelo de clasificación que se emplea para clasificar los ejemplos del conjunto de prueba original, formado por una carpeta. De esta manera, se puede comparar los resultados de clasificación obtenidos antes y después de aplicar el algoritmo de selección.

Si el coste temporal del algoritmo de selección es muy elevado (por ejemplo búsquedas completas o aproximaciones *wrappers*) o la base de datos utilizada es de grandes proporciones (atributos y/o ejemplos), el proceso anterior se hace prohibitivo, al tener que ejecutarse el método de reducción en diez ocasiones. En estos casos, se aplica la técnica de selección al conjunto de datos original, dando lugar a un conjunto reducido de datos (figura 3.6). A continuación, se obtienen los resultados del clasificador mediante validación cruzada, por ejemplo con diez particiones sobre el conjunto reducido. En resumen, el clasificador se evalúa el mismo número de veces (diez), pero el selector sólo se ejecuta una vez.

En los próximos capítulos, para evitar el sobreajuste de los algoritmos de selección, se presentan los resultados obtenidos mediante validación cruzada en la selección, es decir, realizando diez reducciones. Además, si las bases de datos utilizadas son pequeñas, el proceso se repite diez veces (10×10 reducciones).

3.7. Aplicaciones de los algoritmos de selección

Esta sección trata de situar los algoritmos de selección frente a un hipotético usuario final. Normalmente al usuario le interesa saber el tipo de situaciones o problemas donde se pueden aplicar estas técnicas, además es indispensable conocer, cuáles y de qué forma aplicarlas.

La selección de atributos es un área de investigación bastante activa desde hace varias década, que intenta desarrollar nuevos algoritmos o mejorar los ya existentes. Por consiguiente, se dispone de una gran cantidad de algoritmos, y como se ha podido comprobar en la tabla 3.1

Comparison table of the discussed method.

Method	Generation	Evaluation	Contin.	Discrete	Nominal	Large Dataset	Multiple Classes	Handle Noise	Optimal Subset
B & B	complete	distance	y	y	n	-	y	-	y++
MDLM	complete	information	y	y	n	-	y	-	n
Focus	complete	consistency	n	y	y	n	y	n	y
Relief	heuristic	distance	y	y	y	y	n	y	n
DTM	heuristic	information	y	y	y	y	y	-	n
POE+ACC	heuristic	dependency	y	y	y	-	y	-	n
LVF	random	consistency	n	y	y	y	y	y*	y**

-	method does not discuss about the particular characteristic.		
y++	if certain assumptions are valid.		
y*	user is required to provide the noise level.	y**	provided there are enough resources.

*note : "classifier error rate" not included (ie. Depend on specify classifier).

Figura 3.7: Capacidades de los métodos de selección.

son muy diferentes unos a otros. Algunos son válidos en cualquier dominio de datos mientras que otros son más específicos, unos resultan más complicados de entender y de configurar y otros menos, etc. Lo cierto es que el usuario final que tenga la necesidad de reducir el número de atributos se puede encontrar ante un verdadero problema a la hora de elegir que algoritmo aplicar. Por tanto, uno de los asuntos más interesantes de tratar es la selección inteligente de algoritmos de selección. Es decir, saber automáticamente o en función de algunos parámetros, que algoritmo se ajusta mejor a las características de los datos y a las necesidades del usuario.

3.7.1. Plataformas y guías

Existen diversas aproximaciones para ayudar en la elección del algoritmo de selección. En [37] se propone un tabla de comparación de métodos, figura 3.7, donde se puede observar una fila para cada algoritmo, y se utilizará aquel que mejor se adapte a unos parámetros de entrada, como el tamaño de las bases de datos, la presencia de ruido o no, etc. En [126], figura 3.8, se indica la forma de llegar al mejor algoritmo por medio de un diagrama de flujo, donde se va guiando al usuario, en función de sus objetivos y del grado de conocimiento que tenga del problema, hasta el algoritmo que mejor se ajuste de entre una de las cuatro aproximaciones que contempla (por aproximación, divergencia, B&B y combinación de *SFS*), siendo M el número de clases, D el número total de atributos y d el número de atributos del conjunto final. El diagrama de flujo propuesto en el trabajo [39], figura 3.9, realiza un proceso de selección mediante un algoritmo (*SetCover*), y en base al resultado se sigue el camino más adecuado hasta desembocar en un método en concreto. En este caso el estudio se ha realizado utilizando

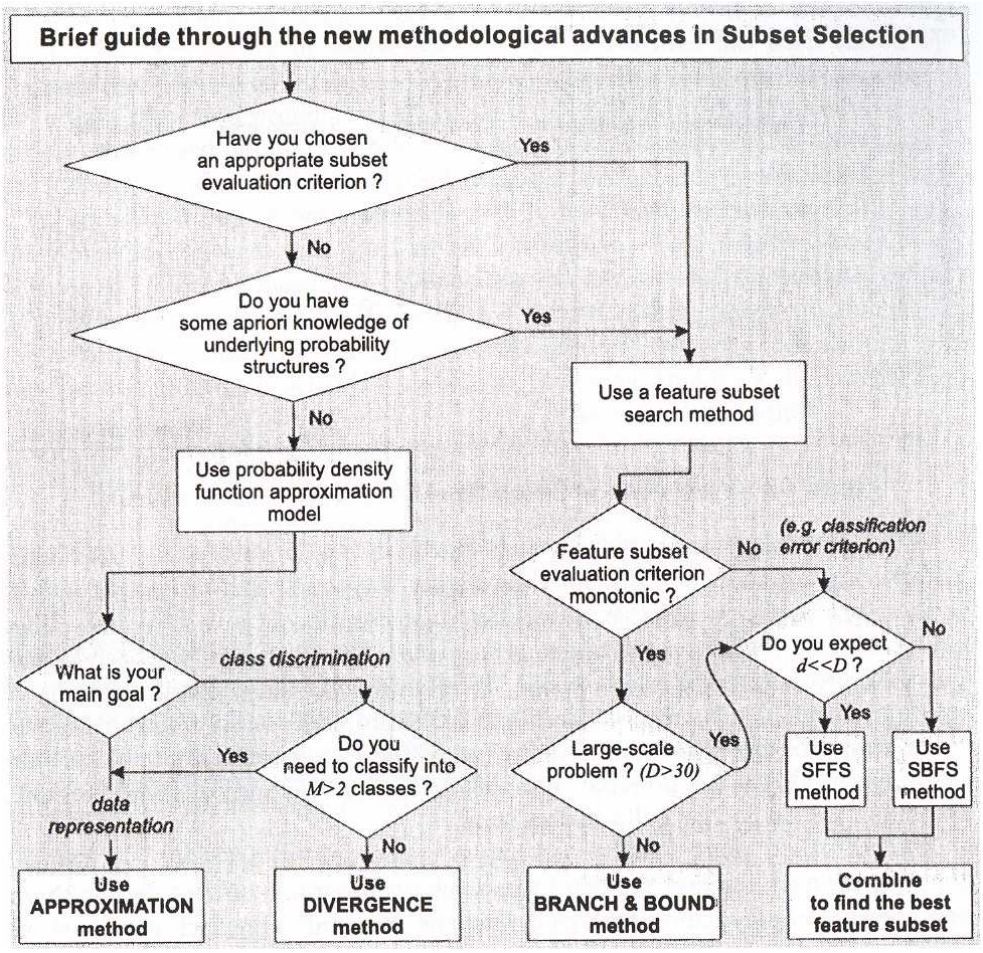


Figura 3.8: Diagrama de flujo de un prototipo de guía de algoritmos de selección.

algoritmos de selección que emplean la consistencia como criterio de evaluación (*Focus*, *ABB*, *QBB* y *SetCover*). En la figura, N representa el número total de atributos, M el número de atributos relevantes esperado, M' es el número de atributos relevantes estimado con el algoritmo *SetCover*, y M'' el estimado con *QBB*.

En [94] Kudo y Sklansky realizan un estudio comparativo de los algoritmos de selección (ver figura 3.10), dando pautas a seguir para la elección del más adecuado en función del: (1) Tipo de objetivo, A) el mejor subconjunto para un tamaño dado, B) el más pequeño que satisfaga una condición y C) combinación de los dos anteriores; (2) Tamaño de la BD, pequeño, mediano, grande y muy grande; y (3) Criterio de evaluación, monótono, aproximado y no monótono. Utiliza como criterio de evaluación la precisión obtenida con 1-NN mediante leave-one-out, y destaca la búsqueda secuencial flotante para bases de datos pequeñas y medianas, y los algoritmos genéticos para bases de datos de grandes dimensiones. BAB^{++}, *RBAB* y *RBABM*, son variaciones del algoritmo *branch and bound*.

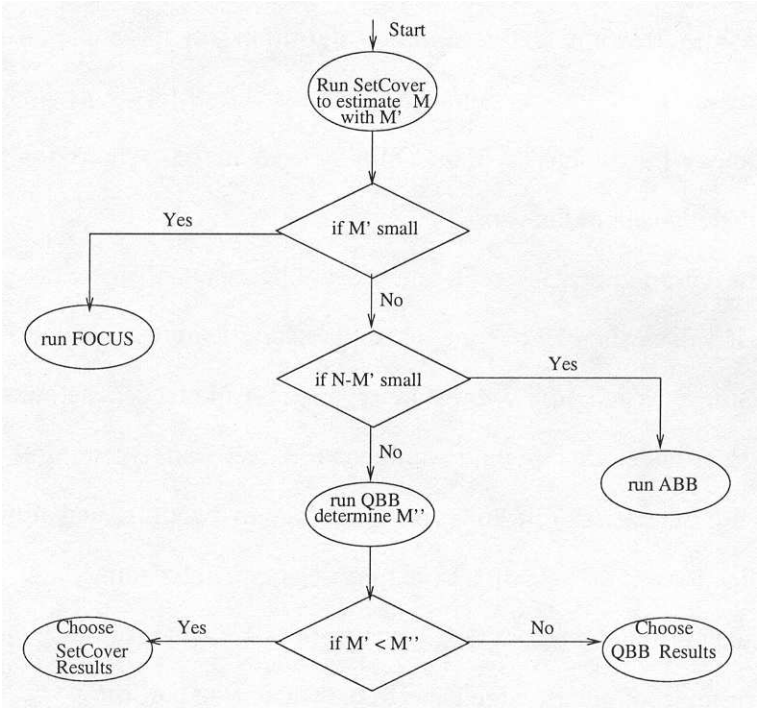


Figura 3.9: Diagrama de flujo de algoritmos que utilizan consistencia como criterio de evaluación.

En [112] se propone una plataforma unificada (mostrada en la figura 3.11) que expande la categorización realizada en la figura 3.3 introduciendo más dimensiones desde el punto de vista del usuario. Dándole un valor a cada una de las dimensiones, se podrá identificar el algoritmo que mejor se adapte. El conocimiento que se tenga de la situación y las características de los datos son dos factores determinantes para una buena elección del método de selección de atributos. El factor conocimiento cubre: el propósito de la selección, si el tiempo es importante, el tipo de salida, y la razón entre el número esperado de atributos seleccionados y el número total. Dentro de las características que interesa conocer de los datos se encuentran: saber si se dispone de información de la clase, para saber qué tipo de estudio se puede llevar a cabo; tipo de atributos, cada puede requerir un procesamiento distinto; calidad de la base de datos, si contiene ruido o valores olvidados; y la razón entre el número de atributos y el número de instancias.

3.7.2. Campo de aplicación

En aplicaciones reales, es fácil encontrar problemas tales como demasiados atributos, dependencia entre ellos, combinación de varios atributos, etc. En numerosas ocasiones, los humanos difícilmente entenderán hipótesis cuando las bases de datos contengan un gran número de variables (posiblemente miles en casos de estudios demográficos, y de decenas de miles en

Objective Type	Scale of Initial Set of Features								
	Small ($n < 20$)			Medium ($20 \leq n < 50$)			Large ($50 \leq n \leq 100$)		Very Large ($n > 100$)
	Mono.	Approx. Mono.	Non Mono.	Mono.	Approx. Mono.	Non Mono.	Mono. or Approx. Mono.	Non Mono.	
A	BAB^{++}	SFFS SBFS	*	BAB^{++} $BAB^{++}(s)$	SFFS SBFS GA	*	SFFS SBFS GA	*	GA
B	RBAB RBABM		*	RBAB w. termination RBABM w. termination GA		*	GA	*	GA
C	*	*	GA SFFS SBFS	*	*	GA SFFS SBFS	*	GA	GA

*: rarely occurs

Figura 3.10: Comparación de algoritmos de selección.

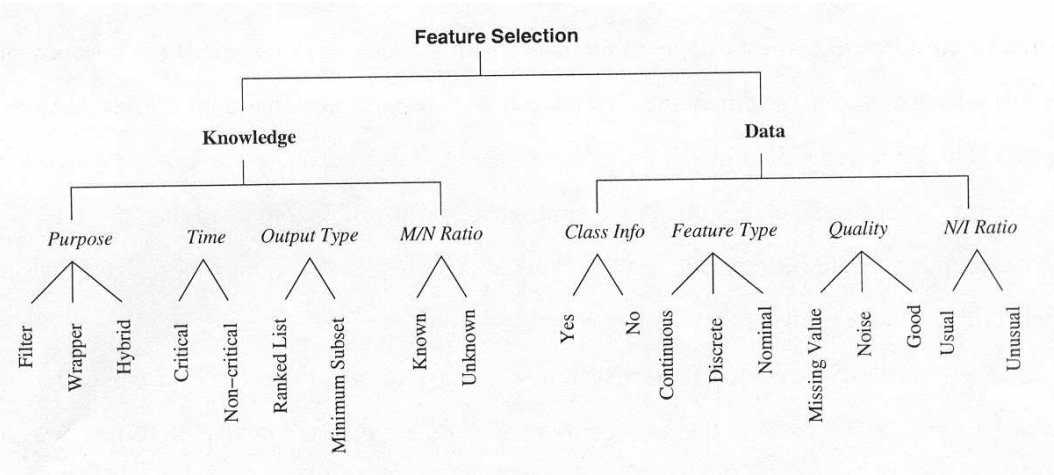


Figura 3.11: Plataforma unificada.

casos de análisis de microarrays, de análisis de documento de textos, o webs). Sin embargo, las personas si pueden encontrar fácil de entender aspectos de problemas en subespacios con menos dimensiones. La selección de atributos puede reducir la dimensionalidad para permitir aplicar algoritmos de minería de datos que en las circunstancias originales no sería posible. Se muestran a continuación algunas aplicaciones ilustrativas.

Categorización de textos es el problema de asignar automáticamente categorías predefinidas a textos libres. Esta tarea tiene una gran aplicación dado el masivo volumen de texto online disponible en Internet, correos electrónicos, bibliotecas digitales, etc. Estas bases de datos se caracterizan por la elevada dimensión del espacio formado por el conjunto de los atributos, donde cada uno consiste en un término (palabra o frase) que ocurre en un documento, y el número de términos puede ser de cientos de miles. Por tanto, se desea reducir el espacio formado por el conjunto de los atributos original sin sacrificar la exactitud de categorización.

Recuperación de imágenes. Recientemente, se ha incrementado rápidamente la cantidad de colecciones de imágenes tanto en el entorno civil como militar. Sin embargo, para que esta información sea de utilidad necesita estar organizada. En vez de anotar manualmente palabras claves, las imágenes se indexan por sus propios contenidos visuales (atributos), tales como color, textura, forma, etc.

Análisis genómico. En los últimos años se dispone de más datos estructurales y funcionales del genoma humano, presentando excelentes aportunidades y retos a la minería de datos. En particular, la tecnología de microarrays proporciona la oportunidad de lograr el nivel de expresión de miles o decenas de miles de genes. Sin embargo, el bajo número de ejemplos suele ser una característica común en estas bases de datos.

Tratamiento de clientes, detección de intrusos, marketing, seguros, deportes, comunicaciones, etc, son sólo una muestra de la gran variedad de aplicaciones que nos podemos encontrar.

3.8. Conclusiones y tendencias

Este estudio del estado del arte de la selección de atributos proporciona una guía general en los diversos aspectos que comprende esta tarea. Además de definir el concepto de selección y de analizar su proceso, se ha clasificado y descrito una gran cantidad de algoritmos existentes. La forma de evaluar y comparar los algoritmos indicada en este capítulo, se tendrá en cuenta a lo largo de este documento. Y terminando este capítulo, para ubicar este estudio en la sociedad, se ha comentado algunas de las aplicaciones más frecuentes de estos algoritmos.

Recientemente los datos se han incrementado más y más en ambos sentidos (número de instancias y de atributos) en todas las áreas del conocimiento humano. Esta gran cantidad de

datos causa serios problemas a muchos algoritmos de minería de datos con respecto a la escalabilidad y al rendimiento. Por ejemplo, bases de datos con cientos o miles de atributos pueden contener un alto grado de información irrelevante y/o redundante, con lo que se degradaría el rendimiento de un algoritmo de minería. Esto hace que cambien los nuevos algoritmos de selección. Últimamente están apareciendo algoritmos híbridos, que combinan ventajas de distintos modelos de algoritmos (filtros, wrappers, ranking, etc).

Se puede concluir que la selección de atributos permite mejorar la precisión e interpretabilidad de los métodos de aprendizaje automático, además de reducir el tamaño de la base de datos y el tiempo de los algoritmos de aprendizaje. Además, para diferentes aplicaciones puede convenir distintos algoritmos de selección de característica. Es importante no olvidar que la selección de atributos es un campo dinámico, estrechamente conectado a la minería de datos y a otras técnicas de preprocesamiento.

Capítulo 4

Criterio de evaluación de atributos basado en proyecciones

En este capítulo se presenta un nuevo criterio para medir la importancia de un atributo dentro de un marco de aprendizaje supervisado. Para ello, se implementa un algoritmo denominado *SOAP* (*Selection of attributes by projections*) basado en proyecciones. *SOAP* evalúa los atributos de una base de datos analizando las proyecciones de los elementos del conjunto de datos sobre cada dimensión o atributo individual. Esta criterio o medida hace posible situar los atributos por orden de importancia en la determinación de la clase. La simplicidad y rapidez en la evaluación de los atributos son sus principales ventajas y uno de los objetivos primordiales de este trabajo de investigación. Para comprobar la bondad de la selección basada en la medida propuesta, se presentan los resultados de diversas comparativas con otros métodos de selección sobre bases de datos bien conocidas, aplicando diferentes algoritmos de inducción.

Es fácil comparar los métodos de selección de subconjuntos de atributos, no ocurre lo mismo cuando se desea comprobar si un ranking es mejor que otro. En el presente capítulo, se exponen diversas formas de comparar rankings generados por algoritmos de evaluación de atributos, mostrando la diversidad de interpretaciones posibles en función del enfoque dado al estudio que se realice. Se parte de la premisa de la no existencia de un único subconjunto ideal para todos los casos. La finalidad de los algoritmos de evaluación de atributos es la de reducir el conjunto de datos a los primeros atributos de cada ranking, sin perder predicción frente a los conjuntos de datos originales. Aquí se propone un método que mide el comportamiento de un ranking de atributos generado y permite comparar diferentes algoritmos de listas de atributos. Para ello se basa en un nuevo concepto, el área bajo la curva de comportamiento al clasificar un ranking de atributos (*AURC–Area Under the Ranking Classification performance curve*). Las conclusiones y tendencias extraídas de este capítulo pretenden dar soporte al usuario ante

la realización de tareas de aprendizaje, donde intervengan algunos de los algoritmos de ranking aquí estudiados.

A continuación se describen las observaciones realizadas sobre varios ejemplos de bases de datos que motivaron el estudio para medir la relevancia de un atributo aquí expuesto. En la sección 4.2 se definen formalmente conceptos que se utilizan en la descripción del método de selección propuesto en la sección 4.3. Los resultados de diferentes experimentos se muestran en la sección 4.4, resaltando el apartado 4.4.3, donde se exponen trabajos relacionados con la comparación de técnicas de ranking de atributos, se definen formalmente lo que se entiende por ranking de atributos y otros conceptos que ayudarán en la comprensión de la metodología de comparación, se presentan las motivaciones que han llevado al estudio de la forma de comparar rankings, y se propone una nueva metodología. Finalmente, en la sección 4.5 se recogen las conclusiones más interesantes.

4.1. Introducción

Para ilustrar la idea principal del algoritmo, se parte de las observaciones realizadas a los conjuntos de datos *IRIS* y *WINE*, debido a que la representación gráfica de sus proyecciones bidimensionales son muy fáciles de interpretar y a que son por todos conocidos. Se han realizado tres proyecciones de *IRIS* y una de *WINE* en gráficos bidimensionales.

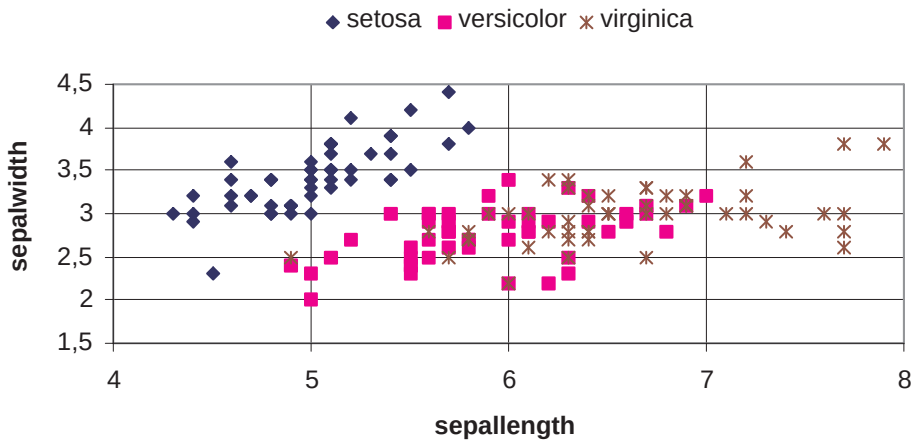


Figura 4.1: Proyección de la base de datos *IRIS* en los atributos *Sepalwidth–Sepallength*.

Como se puede observar en la figura 4.1, si se realiza una proyección de los ejemplos de *IRIS* sobre el eje de abscisas u ordenadas no se obtienen intervalos donde una clase sea mayoritaria. Sólo el intervalo [4.3, 4.8] de *Sepallength* para la clase *Setosa* o [7.1,8.0] para *Virginica*. En la

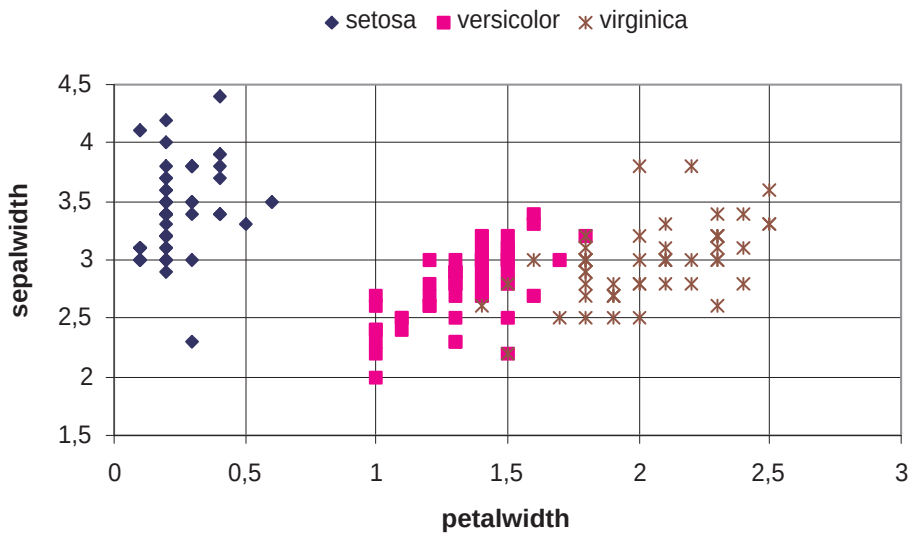


Figura 4.2: Proyección de la base de datos *IRIS* en los atributos *Sepalwidth–Petalwidth*.

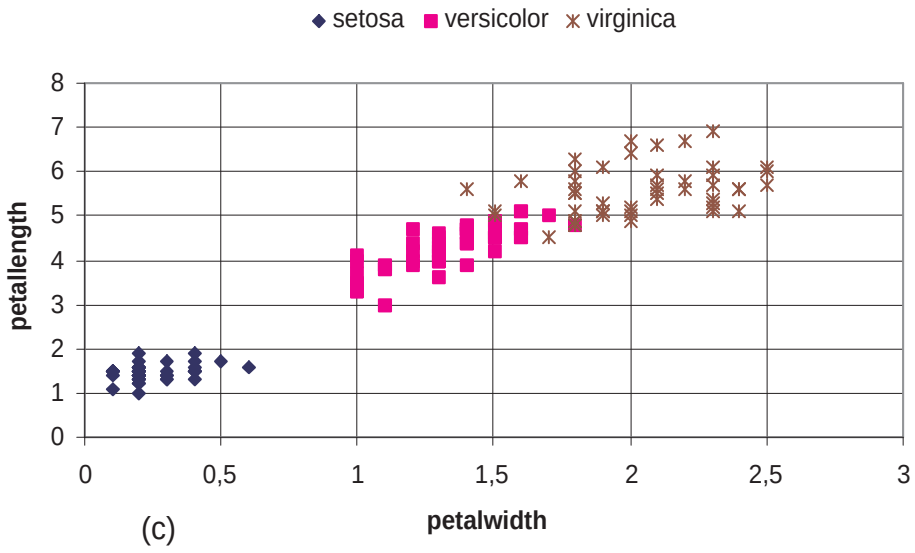


Figura 4.3: Proyección de la base de datos *IRIS* en los atributos *Petallength–Petalwidth*.

figura 4.2 con el parámetro *Sepalwidth* (en el eje de ordenadas) tampoco se aprecian intervalos nítidos y sin embargo, para el atributo *Petalwidth* se pueden advertir algunos intervalos donde la clase es única: [0,0.6] para *Setosa*, [1.0,1.3] para *Versicolor* y [1.8,2.5] para *Virginica*. Finalmente en la figura 4.3, se puede apreciar que la división de las clases es casi nítida en ambos parámetros. Esto es debido a que al proyectar las etiquetas de las clases sobre cada atributo el

número de cambios de etiqueta en la proyección es mínimo. Por ejemplo, se puede comprobar que para *Petallength* el primer cambio de etiqueta se produce para el valor 3 (de *Setosa* a *Versicolor*), el segundo en 4.5 (de *Versicolor* a *Virginica*), después hay otros cambios en 4.8, 4.9, 5.0 y el último en el punto 5.1.

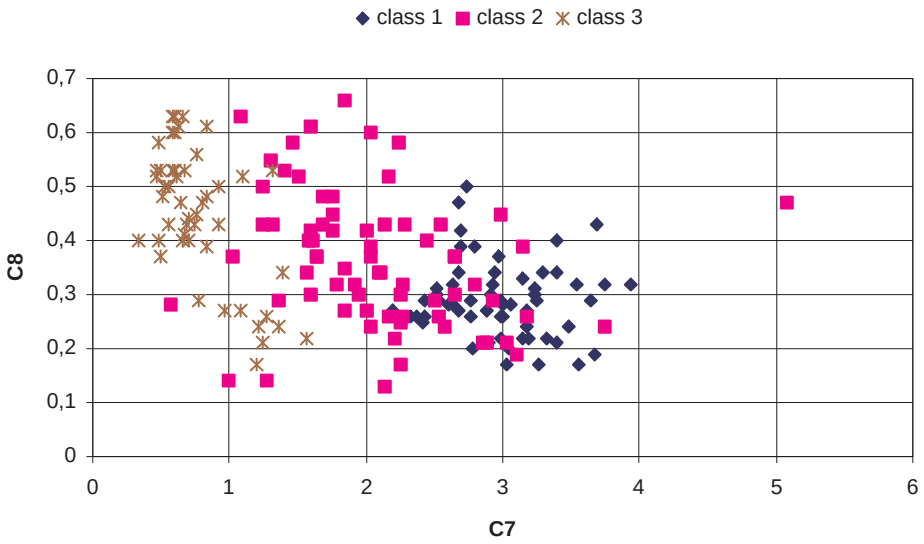


Figura 4.4: Proyección de la base de datos *WINE* en los atributos *C8*-*C7*.

A la misma conclusión se puede llegar observando la figura 4.4, donde están proyectados los ejemplos de la base de datos *WINE* sobre los atributo *C7* y *C8*. Recorriendo el eje de abscisas desde el origen hasta el mayor valor del atributo, se puede distinguir aproximadamente una serie de intervalos donde prevalece una clase. En el intervalo [0,1], prácticamente todos los elementos son de la *clase 3*. Desde aquí hasta el valor 1.5, se mezclan de la *clase 2* y *3*. Nuevamente se observa el intervalo aproximado [1.5,2.3] donde los elementos pertenecen a la *clase 2*. El siguiente intervalo [2.4,3] estaría formado por elementos de la *clase 1* y *2*, y hasta el final del recorrido predominan los elementos con etiqueta *1*. No ocurre así en el eje de ordenadas, donde no se obtienen intervalos donde una clase sea mayoritaria.

Se puede intuir que será más fácil clasificar por los atributos que presenten un menor número de cambios de etiquetas (*NLC*, *Number of Label Changes*). Ordenando los atributos ascendentemente según su *NLC*, se logra un ranking con los mejores atributos para clasificar, en el caso del conjunto Iris sería: *Petalwidth* 16, *Petallength* 19, *Sepallenth* 87 y *Sepalwidth* 120. Este resultado coincide con el conocido en la literatura, de la importancia de los atributos que miden la anchura y longitud de los pétalos contra la poca influencia de los que miden los sépalos.

En particular para los atributos representados en la figura 4.2, clasificando la base de datos

con *C4.5* únicamente por el atributo *Sepalwidth* se obtiene una tasa de acierto del 59 %, y si se clasifica por *Petalwidth* un 95 %. Los atributos utilizados en la figura 4.4 son el primero (C7) y el último (C8) en el ranking elaborado según el *NLC* de cada atributo con 43 y 139 cambios respectivamente. Aplicando de la misma forma el clasificador *C4.5* , se obtiene una tasa de acierto de 80.34 % para *C7* y 47.75 % para *C8*.

Cuando los algoritmos de selección de atributos se aplican como técnica de preprocesado para la clasificación, se está interesado en aquellos atributos que clasifican mejor los datos desconocidos hasta el momento. Si los algoritmos proporcionan un subconjunto de atributos, este subconjunto se utiliza para generar el modelo de conocimiento que clasificará los nuevos datos. Sin embargo, cuando la salida del algoritmo es un ranking, no es fácil determinar cuántos atributos son necesarios para obtener un buen resultado de clasificación.

Este capítulo también presenta distintas formas de comparar rankings de atributos, y se muestra la diversidad de interpretaciones posibles según el enfoque del estudio que se realice. Se plantea como objetivo saber si existe alguna dependencia entre clasificadores y métodos de ranking, además de dar respuesta a dos cuestiones fundamentales ¿Qué es un ranking de atributos? y ¿Cómo valorar/medir un ranking?

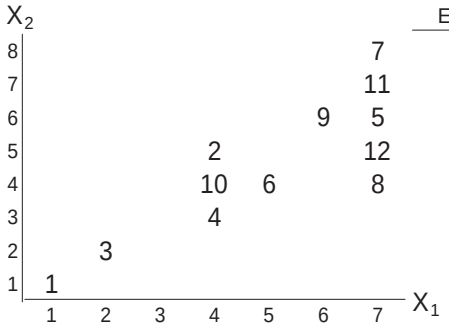
Para comprobar la bondad del algoritmo de evaluación de atributos aportado y el método de comparación propuesto en este capítulo, se utilizan distintos métodos de evaluación de atributos, y se cotejan los resultados calculando la tasa de aciertos con tres clasificadores: *C4.5*, *Naïve Bayes* y el vecino más cercano (ver sección 2.3 en la página 14).

4.2. Definiciones de proyecciones

A continuación se establecerán algunas definiciones que describen formalmente los conceptos que se manejarán a lo largo de este capítulo. Para mostrar las definiciones que se van a realizar, se utiliza una base de datos bidimensional, figura 4.5, con doce ejemplos numerados y dos etiquetas: I (números impares) y P (números pares).

Aplicando las definiciones enumeradas en la sección 2.2 (página 13) al ejemplo de la figura, se tiene que el conjunto de datos $\mathcal{E}$ está formado por 12 ejemplos e_j, donde $j = 1, \ldots, 12$, con dos atributos de entrada, X_1 y X_2, y uno de salida o clase, Y. Por tanto, cada ejemplo estará compuesto por tres valores $e_j = (\overline{x_j}, y_j) = (x_{j,1}, x_{j,2}, y_j)$, donde $j = 1, \ldots, 12$, siendo continuo el $Dom(X_1) = [1, 7]$, y el $Dom(X_2) = [1, 8]$, y discreto el $Dom(Y) = \{I, P\}$. En la figura 4.5, $\mathcal{E}$ es la tabla de la derecha, y $e_5 = (7, 6, I)$.

Definición 4.1 *Se define la función proyección* $\pi_i \; \forall i : 1..n$ *y la función etiqueta lab. Sea* $e_j =$



Ejemplos	X_1	X_2	Clase
1	1	1	I
2	4	5	P
3	2	2	I
4	4	3	P
5	7	6	I
6	5	4	P
7	7	8	I
8	7	4	P
9	6	6	I
10	4	4	P
11	7	7	I
12	7	5	P

Figura 4.5: Base de datos artificial con 12 elementos y 2 etiquetas (*P*,*I*)

$(x_{j,1}, x_{j,2}, \ldots, y)$

$$\pi_i : \mathcal{E} \rightarrow Dom(X_i) \quad \pi_i(e_j) = x_{j,i} \quad \forall j : 1..m \ \wedge \tag{4.1}$$

$$lab : \mathcal{E} \rightarrow Dom(Y) \quad lab(e_j) = y \tag{4.2}$$

En el ejemplo:

$$\begin{array}{lll} \pi_1(e_1) = 1 & \forall j : 1.,12 & \pi_1(e_j) = \{1,4,2,4,7,5,7,7,6,4,7,7\} \\ \pi_2(e_1) = 1 & \forall j : 1.,12 & \pi_2(e_j) = \{1,5,2,3,6,4,8,4,6,4,7,5\} \\ lab(e_1) = I & \forall j : 1.,12 & lab(e_j) = \{I,P,I,P,I,P,I,P,I,P,I,P\} \end{array}$$

Definición 4.2 *Para cada atributo X_i se define dos relaciones de orden en $Dom(X_i)$ que se denota $\leq$ y $<$ tales que si X_i es continuo son las relaciones de orden establecidas en $\mathbb{R}$ y si X_i es discreto son las establecidas por un orden lexicográfico en el conjunto $Dom(X_i)$.*

En el conjunto $\mathcal{E}$ se define para cada atributo X_i dos relaciones de orden $\leq_i$ y $<_i$ de forma que dados los ejemplos e y f se dice que:

$$e \leq_i f \quad si \quad \pi_i(e) \leq \pi_i(f) \tag{4.3}$$

$$e <_i f \quad si \quad \pi_i(e) < \pi_i(f) \tag{4.4}$$

Y una relación de equivalencia donde

$$e =_i f \quad si \quad \pi_i(e) = \pi_i(f) \tag{4.5}$$

Por ejemplo para $i = 1$:

$$\begin{array}{lccc} & \pi_1(e_2) = 4 & \pi_1(e_3) = 2 & \pi_1(e_4) = 4 \\ \textit{entonces}: & e_3 \leq_1 e_4 & e_3 <_1 e_4 & e_2 =_1 e_4 \\ \textit{porque}: & \pi_1(e_3) \leq_1 \pi_1(e_4) & \pi_1(e_3) <_1 \pi_1(e_4) & \pi_1(e_2) =_1 \pi_1(e_4) \end{array}$$

Definición 4.3 *Dado el conjunto $\mathcal{E}$ se denomina $\sigma_i(\mathcal{E})$ a la secuencia ordenada por el atributo X_i de los ejemplos de $\mathcal{E}$. Es decir*

$$\begin{aligned} \sigma_i(\mathcal{E}) =< e_1, \ldots, e_n > \quad / \quad & e_j \in \mathcal{E} \quad \forall j : 1..n \quad \wedge \\ & \forall j : 1..(n-1) \quad e_j \leq_i e_{j+1} \end{aligned} \tag{4.6}$$

Sobre el conjunto $\mathcal{E}$ se establecen las secuencias ordenadas:

$$\begin{aligned} \sigma_1(\mathcal{E}) &=< e_1, e_3, e_4, e_{10}, e_2, e_6, e_9, e_8, e_{12}, e_5, e_{11}, e_7 > \\ \sigma_2(\mathcal{E}) &=< e_1, e_3, e_4, e_{10}, e_6, e_8, e_2, e_{12}, e_9, e_5, e_{11}, e_7 > \end{aligned}$$

donde $\sigma_i(\mathcal{E})$ es obtenida de la ordenación de los ejemplos de $\mathcal{E}$ por el valor del atributo X_i.

Definición 4.4 *Sea $\sigma_i(\mathcal{E}) =< e_1, \ldots, e_n >$ una secuencia ordenada de $\mathcal{E}$ por el atributo X_i, entonces se dice que existe una subsecuencia ordenada múltiple (*SOM*) y se denomina S si*

$$\exists\, j, k > 0 \;/\; e_{j-1} <_i e_j =_i \cdots =_i e_{j+k} <_i e_{j+k+1} \tag{4.7}$$

siendo la SOM $S =< e_j, \ldots, e_{j+k} >$ *(si $j = 1$ la primera condición se obvia)*
Se define etiqueta mayoritaria de una SOM*, $ml(S)$, a la moda del conjunto $\{lab(e_l)\}_{l=j\ldots j+k}$.*

SOM en el ejemplo son:
$S =< e_4, e_{10}, e_2 >$ es una *SOM* por el atributo X_1, ya que los ejemplos e_4, e_{10} y e_2 tienen para X_1 el mismo valor (4) y $ml(S) = P$ al ser todos los ejemplos de la clase par.
$S =< e_{10}, e_6, e_8 >$ es una *SOM* por el atributo X_2, ya que los ejemplos e_{10}, e_6 y e_8 tienen para X_2 el mismo valor (4) y $ml(S) = P$ al ser todos los ejemplos de la clase par.

Definición 4.5 *Sea $\sigma_i(\mathcal{E}) =< e_1, \ldots, e_n >$ una secuencia ordenada de $\mathcal{E}$ por el atributo X_i se dice que existe una subsecuencia ordenada simple (*SOS*) y se denomina S si*

$$\exists\, j, k > 0 \;/\; e_{j-1} =_i e_j <_i e_{j+1} <_i \cdots <_i e_{j+k-1} <_i e_{j+k} =_i e_{j+k+1} \tag{4.8}$$

siendo la SOS $S =< e_{j+1}, e_{j+2}, \ldots, e_{j+k-1} >$ *(si j=1 la primera condición se obvia)*

El ejemplo e_{j+1} es denominado primer ejemplo, $fe(S)$, y e_{j+k-1} último ejemplo $le(S)$.
SOS en el ejemplo:

$$\begin{aligned} En\ X_1: &\quad S =< e_1, e_3 > \\ &\quad S =< e_6, e_9 > \\ En\ X_2: &\quad S =< e_1, e_3, e_4 > \\ &\quad S =< e_{11}, e_7 > \end{aligned}$$

Teorema: Dada una secuencia ordenada de ejemplos $\sigma_i(\mathcal{E})$ por algún atributo X_i siempre se podrá dividir en subsecuencias donde o bien hay una *SOS*, o una *SOM* o una sucesión de secuencias de unas y otras, teniendo en cuenta que tras una *SOS* siempre vendrá una múltiple, pero detrás de una múltiple puede aparecer una simple o múltiple. En general, $\sigma_i(\mathcal{E}) = S_1, S_2, \ldots, S_r$ donde si S_j es *SOS* entonces S_{j+1} es *SOM* y si S_j es *SOM* entonces S_{j+1} puede ser *SOS* o *SOM*.
Secuencias en el ejemplo:

$$\sigma_1(\mathcal{E}) =< \overbrace{e_1, e_3}^{Simple} >, < \overbrace{e_4, e_{10}, e_2}^{Múltiple} >, < \overbrace{e_6, e_9}^{Simple} >, < \overbrace{e_8, e_{12}, e_5, e_{11}, e_7}^{Múltiple} >$$

$$\sigma_2(\mathcal{E}) =< \underbrace{e_1, e_3, e_4}_{Simple} >, < \underbrace{e_{10}, e_6, e_8}_{Múltiple} >, < \underbrace{e_2, e_{12}}_{Múltiple} >, < \underbrace{e_9, e_5}_{Múltiple} >, < \underbrace{e_{11}, e_7}_{Simple} >$$

Definición 4.6 *Dada una secuencia ordenada S de k ejemplos* $S =< e_1, e_2, \ldots, e_k >$ *se define la función* $ch : S - \{e_k\} \rightarrow \{0, 1\}$

$$ch(e_j) = \begin{cases} 1 & si \quad lab(e_j) \neq lab(e_{j+1}) \\ 0 & si \quad lab(e_j) = lab(e_{j+1}) \end{cases} \qquad \forall j : 1 \ldots (k\text{–}1) \tag{4.9}$$

En el ejemplo, para X_2, en la *SOS* $S =< e_1, e_3, e_4 >$:

$$\begin{aligned} ch(e_1) = 0 \quad lab(e_1) = lab(e_3) \\ ch(e_3) = 1 \quad lab(e_3) \neq lab(e_4) \end{aligned}$$

Definición 4.7 *Dada una secuencia ordenada de ejemplos* $S =< e_1, \ldots, e_k >$ *se define la función ChangeCounter*

$$ChangeCounter(S) = \sum_{j=1}^{k-1} ch(e_j) \tag{4.10}$$

Para $S =< e_1, e_3, e_4 >$, $ChangeCounter(S) = 1$.

Definición 4.8 *Sea S una* SOS *de* $\sigma_i(\mathcal{E})$ *para algún* $i = 1..n$ *de un conjunto* $\mathcal{E}$. *Entonces S o es la última subsecuencia de la división establecida en el Teorema 1 o está seguida de una* SOM *que se denota* S'.

Entonces se define:

$$nch(S) = ChangeCounter(S) + ChangeAdd(S, S') \tag{4.11}$$

donde $ChangeAdd(S, S')$ *vale* 0 *ó* 1 *en función de* S'*, según la siguiente formulación:*

$$ChangeAdd(S, S') = \begin{cases} 0 \quad si \quad S' \text{ no existe} \\ \quad (i.e.\ S \text{ es la última subsecuencia}) \\ 0 \quad si \quad lab(le(S)) = ml(S') \\ 1 \quad si \quad lab(le(S)) \neq ml(S') \end{cases}$$

Cuando se da la situación de tener una *SOS* seguida de una *SOM*, para ser más exactos en el cálculo del número de cambios, se comprueba si al pasar de una subsecuencia a otra se ha producido algún cambio de etiqueta. Para ello se compara la etiqueta del último ejemplo de la *SOS* con la etiqueta mayoritaria de la *SOM*.

En el ejemplo, para X_1, en la *SOS* $S =< e_1, e_3 > \Rightarrow ChangeCounter(S) = 1$ siendo $lab(le(S)) = I$ y como le sigue una *SOM* $S' =< e_4, e_{10}, e_2 >$ que tiene como $lab(ml(S')) = P \Rightarrow ChangeAdd(S, S') = 1$, totalizando, $nch(S) = 2$.

Definición 4.9 *Sea* S *una* SOM *de* $\sigma_i(\mathcal{E})$ *para algún* $i = 1..n$ *de un conjunto* $\mathcal{E}$*. Entonces* S *o es la última subsecuencia de la división establecida en el Teorema 1 o está seguida por otra* SOM *o por una* SOS*.*

Si S es una *SOM*, el valor de la función *ChangeCounter* viene influido por el orden de los ejemplos con igual valor en el parámetro por el que se ha ordenado S. Esto hace que el valor de *ChangeCounter* no sea unívoco y por ello se define *nch* como el peor caso, es decir, el valor del mayor número de cambios de etiqueta para todas las ordenaciones posibles de ejemplos de S. Entonces, dada $S =< e_1, \ldots, e_k >$ una *SOM* (según la definición 7), se define $\Gamma(S)$ el conjunto de todas las permutaciones posibles de los ejemplos de S, y se define

$$nch(S) = max_{\tau \in \Gamma(S)}\{ChangeCounter(\tau)\} + ChangeAdd(S, S') \tag{4.12}$$

donde $ChangeAdd(S, S')$ vale 0 ó 1 en función de S', según la siguiente formulación:

$$Si\ S' \ es\ SOM\ ChangeAdd(S, S') = \begin{cases} 0 \quad si \quad S' \text{ no existe} \\ 1 \quad si \quad ml(S) \neq ml(S') \\ 0 \quad si \quad ml(S) = ml(S') \end{cases}$$

$$Si\ S'\ es\ SOS\ ChangeAdd(S,S') = \begin{cases} 0 & si \quad S'\ no\ existe \\ 1 & si \quad ml(S) \neq lab(fe(S')) \\ 0 & si \quad ml(S) = lab(fe(S')) \end{cases}$$

En el ejemplo, para el atributo X_1, se tiene la $SOM\ S =< e_8, e_{12}, e_5, e_{11}, e_7 >$, cuyas etiquetas serían <P,P,I,I,I>

$$\left.\begin{array}{cc} Posibilidades & ChangeCounter \\ < P,P,I,I,I > & 1 \\ < P,I,P,I,I > & 3 \\ < I,P,P,I,I > & 2 \\ < I,P,I,P,I > & 4 \end{array}\right\} \Rightarrow nch(S) = 4$$

Otra situación en el ejemplo, para X_2, una $SOM\ S =< e_{10}, e_6, e_8 > \Rightarrow ChangeCounter(S) = 0$ siendo $lab(ml(S)) = P$ y le sigue otra $SOM\ S' =< e_2, e_{12} >$ que tiene como $lab(ml(S')) = P \Rightarrow ChangeAdd(S,S') = 0$, totalizando, $nch(S) = 0$. Continuando con el mismo atributo, ahora se tiene $S =< e_2, e_{12} >$ una *SOM* donde $ChangeCounter(S) = 0$ y $lab(ml(S)) = P$ y $S' =< e_9, e_5 >$ con $lab(ml(S')) = I \Rightarrow ChangeAdd(S,S') = 1$ y $nch(S) = 1$.

4.3. NLC: Número de Cambios de Etiqueta

Definición 4.10 (NLC) *Dados un conjunto $\mathcal{E}$ de ejemplos, $\sigma_i(\mathcal{E})$ la secuencia ordenada por el atributo X_i y $S_1 \ldots S_r$ la división de $\sigma_i(\mathcal{E})$ en* SOS *y* SOM *del teorema 1, entonces se define la función $NLC : \mathcal{X} \rightarrow \mathbb{N}$*

$$NLC(i) = \sum_{j=1}^{r} nch(S_j) \tag{4.13}$$

Para el ejemplo, $\sigma_1(\mathcal{E}) = S_1, S_2, S_3, S_4$, donde $S_1 =< e_1, e_3 >$, $S_2 =< e_4, e_{10}, e_2 >$, $S_3 =< e_6, e_9 >$, $S_4 =< e_8, e_{12}, e_5, e_{11}, e_7 >$ con $nch(S_1) = 1$, $nch(S_2) = 0$, $nch(S_3) = 1$, $nch(S_4) = 4 \Rightarrow NLC(X_1) = 6$. Igualmente se obtiene $NLC(X_2) = 2$.

En esta memoria de tesis se propone una nueva medida para la selección de atributos que no está basada en medidas calculadas entre atributos, ni en complejos o costosos cálculos de distancia entre ejemplos. Este criterio se basa en un único valor (que se ha denominado *NLC: Number of Label Changes*) que relaciona cada atributo con la etiqueta que sirve de clasificación. Este valor se calcula proyectando los ejemplos de la base de datos sobre el eje correspondiente a ese atributo (se ordenan por él), para a continuación recorrer el eje desde el origen hasta el

mayor valor del atributo contabilizando el número de cambios de etiqueta que se producen. El algoritmo es muy simple y rápido. Tiene la capacidad de operar con variables continuas y discretas, así como también, con bases de datos con cualquier número de clases.

Algoritmo 4.1 *SOAP*.
Entrada: $\mathcal{E}$–conjunto de datos (m ejemplos, n atributos)
Salida: R–ranking de atributos
  $R \Leftarrow \{\}$
  para $i = 1$ hasta n **hacer**
    Ordenar $\mathcal{E}$ por el atributo X_i
    $NLC_i \Leftarrow ContarCambios(\mathcal{E}, X_i)$
    posicionar X_i en R con respecto a NLC_i
  fin para

Algoritmo 4.2 *ContarCambios*.
Entrada: $\mathcal{E}$–conjunto de datos (m ejemplos, n atributos), X_i–atributo a procesar
Salida: *Cambios*–Número de cambios de Etiqueta
  $Cambios \Leftarrow 0$
  para $j = 1$ hasta m **hacer**
    si $x_{j,i} \in SOM$ **entonces**
      $Cambios \Leftarrow Cambios + VerCambiosParaMismoValor()$
    sino
      si $lab(e_j) <> lab(e_{j+1})$ **entonces**
        $Cambios \Leftarrow Cambios + 1$
      fin si
    fin si
  fin para

La función *ContarCambios*, considera si trata con diferentes valores de un atributo, es decir, una secuencia ordenada simple (*SOS*), o con una subsecuencia ordenada múltiple (*SOM*), es decir, si hay más de un ejemplo con idéntico valor en un atributo (se puede dar esta situación dentro del rango de valores posibles en variables continuas y al tratar con variables discretas). En el primer caso, el algoritmo compara la presente etiqueta con la última vista, mientras que en el segundo caso, cuenta el número máximo de cambios posibles mediante la función *VerCambiosParaMismoValor*. A continuación se muestra un esquema con las situaciones posibles, y como las resolvería esta función.

1. *SOS* → cuenta un cambio por cada cambio de etiqueta.

2. *SOM* → número máximo de cambios posibles con *VerCambiosParaMismoValor*:

a) Todos los ejemplos con la misma etiqueta → ningún cambio de clase.

b) Etiquetas distintas:

1) Frecuencia de una clase superior a la mitad del número total de ejemplos de la *SOM* → (nº elementos *SOM* - nº elementos de la clase mayoritaria) * 2 (figura 4.7).
2) No existe mayoría → nº elementos *SOM* - 1 (figura 4.6).

Cuando se detecta una *SOM*, en la función *VerCambiosParaMismoValor* se acumulan los ejemplos según su etiqueta, y el algoritmo comprueba si existe una clase cuya frecuencia sea superior a la suma del resto de frecuencias (que supere la mitad del número de ejemplos), si es así, el número de cambios de etiquetas para el peor de los casos será la suma del resto de las frecuencias multiplicado por dos, ya que se podrá intercalar cada uno de los ejemplos de las clases no mayoritarias entre los de la mayoritaria. En el caso de que no exista ninguna frecuencia mayor que la suma del resto de frecuencias, el número máximo de cambios posibles será equivalente al número de ejemplos menos uno. El algoritmo permite trabajar con variables discretas, considerando cada proyección del atributo como una subsecuencia del mismo valor (*SOM*). Este método es válido para cualquier número de clases.

En la figura 4.6, después de aplicar el método de ordenación, en la secuencia *SOM* para el valor 3, entre otras, se podría tener la situación mostrada a la izquierda o a la derecha. Los cambios de etiquetas obtenidos serían dos y siete respectivamente, influyendo notablemente en el resultado final de la evaluación del atributo en cuestión. Por lo tanto, en varias ejecuciones del algoritmo sobre la misma base de datos, un atributo podría estar situado en distintas posiciones del ranking. Como se ha señalado anteriormente, una solución a este problema consiste en encontrar el peor de los casos. La heurística aplicada obtiene el máximo número de cambios de etiqueta dentro del intervalo que contiene valores iguales. De esta forma, el método *VerCambiosParaMismoValor* produciría la salida mostrada en la figura, siete cambios.

El motivo de contar el número máximo de cambios de etiquetas es doble, por un lado penalizar al atributo que presente este tipo de situaciones inconsistentes, y por otro, evitar ambigüedades que se producirían según la posición en la que aparecieran los elementos tras aplicar el algoritmo de ordenación.

Se puede observar en la figura 4.7 que la función *VerCambiosParaMismoValor* devolverá cuatro cambios, porque existe una frecuencia relativa mayor (*clase A*) que la mitad de los elementos de la subsecuencia. El resultado es (8-6)*2=4. De esta manera, siempre se encuentra el número máximo de cambios de etiquetas dentro de un intervalo conteniendo valores repetidos.

Si se aplica el algoritmo *SOAP* al ejemplo de la figura 4.5 se tiene las secuencias proyección

ordenada:

$$\{1, 3, 4, 10, 2, 6, 9, 8, 12, 5, 11, 7\}$$

$$\{1, 3, 4, 10, 6, 8, 2, 12, 9, 5, 11, 7\}$$

para X_1 y X_2 respectivamente, siendo sus proyecciones

$$\{1, 2, 4, 4, 4, 5, 6, 7, 7, 7, 7, 7\}$$

$$\{1, 2, 3, 4, 4, 4, 5, 5, 6, 6, 7, 8\}$$

Y si se dividen en subsecuencias, se tiene las siguientes particiones en las proyecciones y en los ejemplos:

$$\{[1, 2][4, 4, 4][5, 6][7, 7, 7, 7, 7]\} \rightarrow \{[1, 3][4, 10, 2][6, 9][8, 12, 5, 11, 7]\}$$

$$\{[1, 2, 3][4, 4, 4][5, 5][6, 6][7, 8]\} \rightarrow \{[1, 3, 4][10, 6, 8][2, 12][9, 5][11, 7]\}$$

En el atributo sobre el eje de abscisas (X_1) la primera subsecuencia es una *SOS* constante (impar–*I*); la segunda es una *SOM* donde todos los ejemplos son de la misma etiqueta *P*, contándose un cambio por ser distinta a la etiqueta anterior; en la tercera, se produce un cambio; y en la cuarta, se tiene una *SOM* con ejemplos de las dos etiquetas, entonces se considera el máximo *NLC* posible. Se tiene la subsecuencia [8,12,5,11,7], donde aparecen tres elementos de la clase *I* (impar) y dos de la clase *P* (par), y en una serie de ejecuciones independientes, se podría tener las siguientes situaciones: [P,P,I,I,I], [P,I,I,I,P], [I,P,I,I,P],... *NLC* igual a 1, 2 y 3 respectivamente. Por eso la función VerCambiosParaMismoValor devolverá 4, el máximo posible, que correspondería con la situación [I,P,I,P,I]. En total, a este atributo se le contabilizan seis cambios (*NLC*=6).

En el atributo representado en el eje de ordenadas (X_2) de la figura 4.5, se tiene la sucesión de cinco secuencias *SOS*+*SOM*+*SOM*+*SOM*+*SOS*, donde se contabilizan dos cambios, uno en la primera *SOS*, y otro, al pasar de la tercera a la cuarta subsecuencia.

Se dirá entonces que el atributo X_2 con 2 *NLC* será más relevante que el atributo X_1 con 6 *NLC*.

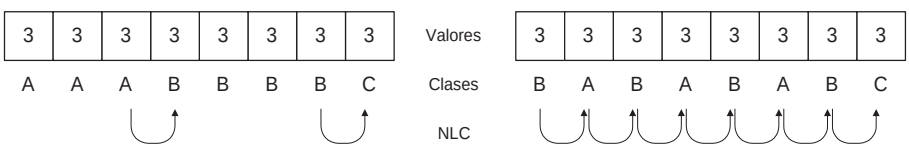


Figura 4.6: Técnica SOAP. Subsecuencia del mismo valor. Izq. dos cambios, dcha. siete.

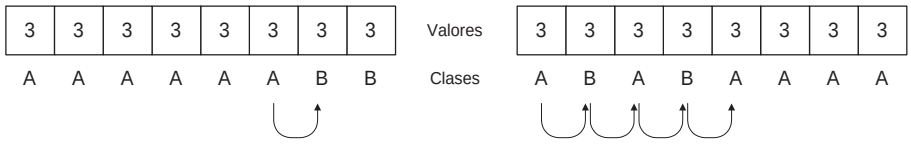


Figura 4.7: Técnica SOAP. Ejemplo de contabilización de etiquetas.

4.4. Evaluaciones experimentales

En esta sección se pretende evaluar como se comporta *SOAP* (*SP*) en términos de exactitud de clasificación y velocidad de procesado. La comparación se efectuó sobre las bases de datos cuyas propiedades se encuentran resumidas en el apéndice A. Los conjuntos de datos se encuentran divididos en grupos en función del tamaño con la intención de favorecer la diversidad de procedimientos de evaluación. Se seleccionaron tres algoritmos de aprendizaje muy diferentes, *C4.5* (C4), *ib1* (IB) y *Naïve Bayes* (NB) (ver sección 2.3 en la página 14), para evaluar la exactitud sobre los atributos ordenados por cada uno de los algoritmos de ranking.

En la comparación, se escogen cuatro criterios de evaluación individual de atributos, representativos de aproximaciones ranking muy diferentes: uno, el algoritmos *Relief* (*RL*), utiliza una medida de distancia (ver apartado 3.5.1, página 50); dos, *Ganancia de Información*, indicado por *IG*, descrito en la sección 3.4.3 página 43; tres, el método *Chi2* (*CH*), que establece el peso de cada atributo basándose en la discretización (apartado 3.5.1); y cuatro, *incertidumbre simétrica* (*SU*) que refleja la correlación no lineal (apartado 3.4.3, página 44).

Los experimentos se llevaron a cabo en un Pentium IV a 3 Gh con 512 Mb de RAM, utilizando implementaciones de los algoritmos existente en el entorno *WEKA* [173] (Waikato Environment for Knowledge Analysis), siendo también implementada en dicho entorno la propuesta hecha en el presente capítulo. Esta herramienta visual está implementada en Java, favoreciendo la portabilidad, y al ser de libre distribución es posible actualizar su código fuente para incorporar nuevas utilidades o modificar las ya existentes.

Existe escasa bibliografía específica donde se describa cómo comparar rankings de atributos. En [103] se comenta brevemente el uso de la curva de aprendizaje para mostrar el efecto de añadir atributos cuando se dispone de una lista de atributos ordenada. Se empieza con el primer atributo (el más relevante) y gradualmente se añade el siguiente más relevante de la lista de atributos, uno a uno hasta el final del ranking, obteniendo la tasa de aciertos con cada subconjunto. Uniendo cada valor obtenido se forma la curva.

En [71] se comparan rankings de atributos mediante un sólo subconjunto, el que mejor clasificación obtiene de todos los subconjuntos formados para obtener la curva de aprendizaje.

Este estudio puede servir para comparar rankings con algoritmos de selección de subconjuntos de atributos, pero no refleja si el comportamiento de una lista de atributos es mejor que el de otra. En [163], se inserta en la base de datos un atributo irrelevante (cuyos valores son aleatorios) que se utiliza de umbral al aplicar el algoritmo de ranking.

Pero lo más simple y difundido es seleccionar los atributos cuyo peso de relevancia sea mayor que un valor umbral [144, 70], normalmente establecido por el usuario. O si se quiere escoger un conjunto con k atributos, simplemente se seleccionan los k primeros de la lista ordenada. Es un método rápido de elegir atributos y comparar resultados, pero con la dificultad de establecer un parámetro que difícilmente maximiza resultados en todas las bases de datos, teniendo que estudiarlo particularmente para cada caso.

Dada la dificultad en evaluar la bondad de un ranking de atributos, en este apartado se realizan tres estudios diferentes:

- En primer lugar, se comparan resultados fijando un número de atributos.
- En segundo lugar, normalizando los valores asignados a los atributos del ranking por cada uno de los métodos, y posteriormente fijando el mismo umbral en todos los casos.
- Finalmente, se analizan las listas de atributos generadas por cada una de las medidas de evaluación utilizando el área bajo la curva de comportamiento de clasificación.

4.4.1. Los primeros k atributos

La manera más simple de comparar los rankings generados por distintos algoritmos, donde se intenta conseguir la imparcialidad en la forma de seleccionar los atributos, es fijar el mismo número de atributos para todos los métodos [147]. Así se intenta reflejar la predicción de los métodos en las mismas condiciones. Sin embargo, como los métodos que generan rankings no detectan redundancia, se puede dar el caso que entre los primeros atributos de la lista, se encuentren varios igualmente relevantes, y los resultados de clasificación no sean buenos. En definitiva, el elegir un número determinado de atributos es una manera orientativa de valorar los rankings, pero no determinante.

Bases de datos pequeñas con los k primeros atributos del ranking

A continuación se comparan los resultados obtenidos teniendo en cuenta el primer atributo de la lista, los tres y los cinco primeros atributos. Las tablas 4.1, 4.2 y 4.3 proporcionan respectivamente para NB, C4 y IB, la tasa de aciertos media por cada uno de los criterios de selección, y con el conjunto original, mediante diez ejecuciones de validación cruzada en la selección con

diez carpetas. Por columnas, se distinguen tres grupos de resultados, con 1,3 y 5 atributos, conteniendo cada uno los diferentes criterios de evaluación (*SP*–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica). Los símbolos ∘ y • respectivamente identifican si la diferencia con respecto a *SOAP* es estadísticamente mejor o peor, al nivel 0.05 de confianza. La última columna muestra el resultado del clasificador con las bases de datos originales; la penúltima fila, el promedio de las exactitudes de todas las bases de datos; y la última fila, las veces que *SP* gana o pierde significativamente.

El análisis comparativo de los datos mostrados en las tres tablas se puede realizar desde distintos puntos de vista: por clasificador; según el número de atributos utilizado (1, 3 ó 5); con respecto al conjunto original de atributos; por combinación de algunos de los anteriores; y de forma global.

En los resultados obtenidos con el clasificador NB, visibles en la tabla 4.1, no existen tendencias claras, no siendo las diferencias, en muchos casos, considerables para los tres grupos de rankings. El promedio de las exactitudes (penúltima línea de la tabla) es similar para los cinco métodos, y las diferencias significativas encontradas a favor o en contra tampoco son importantes dada la cantidad de bases de datos utilizadas. En las bases de datos donde se ha encontrado alguna diferencia, no ha sido homogénea con los tres rankings estudiados, es más, en algunos conjuntos, *SP* gana con un número de atributos y pierde con otro, por ejemplo ANN, SEG y ZOO. Tan sólo *SP* pierde en los tres casos, por una diferencia mínima, frente a *IG*.

Igualmente, en la tabla 4.2 se observa una gran similitud entre los promedios de la penúltima fila de los resultados obtenidos con el clasificador C4. No existen diferencias significativas que se mantengan en las tres pruebas realizadas para cada base de datos, pero si se observa que el *IG* y *SU* ofrecen un comportamiento algo mejor que *SP*, mientras que frente a *RL* y *CH*, *SP* gana en unos casos y pierde en otros. *SP* gana con un número de atributos y pierde con otro en VEH. Las bases de datos donde *SP* obtiene mejores resultados son AUT, GLA y SEG; y los más desfavorables con COL, HES e ION.

El comportamiento de los algoritmos que muestra la tabla 4.3 para el clasificador IB, dentro de la igualdad existente, es favorable a *SP*. En este caso, *SP* gana a todos los demás criterios con el atributo que se escoge en primer lugar y con los cinco primeros del ranking, excepto con el primero de *SU* que empata, mostrando *SP* un mejor resultado de clasificación. Con tres atributos *SP* gana a *RL* y a *CH*, y pierde frente a *IG* y *SU*. No existen diferencias significativas que se mantengan en las tres pruebas realizadas para cada base de datos, excepto para los conjuntos HEH y SEG, donde *SP* pierde y gana respectivamente.

Se puede comprobar la variabilidad de los resultados obtenidos en función del número de atributos, por ejemplo comparando *SP* y *RL* mediante el clasificador NB, *SP* gana con un atri-

buto (10/5), un poco mejor *RL* con tres (9/11), y con cinco atributos gana *RL* (5/11). En general, *SP* tiene un mejor comportamiento frente al resto con el primer atributo.

Del análisis realizado con los resultados mostrados en las tres tablas, al clasificar las bases de datos con el primer atributo y con los tres y cinco primeros atributos del ranking, se observan algunas coincidencias:

- En las pruebas realizadas con un atributo, las diferencias entre *SP* y el resto de algoritmos se mantienen en las tres tablas (aunque algunas veces no es significativa). *SP* gana con un atributo en las bases de datos AUD, AUT, LUC, y pierde con BCA; con los conjuntos HEH, GLA, GL2 e ION, se obtienen mejores tasas de aciertos con *SP* en dos clasificadores.

- Con tres atributos, *SP* destaca favorablemente en las tres ocasiones con GLA, y SEG; y destaca positivamente en dos tabla con WIN, y negativamente con dos clasificadores con ION, LUC y VEH.

- Con cinco atributos, *SP* gana con los tres clasificadores en SEG y VEH, y pierde en CRG, ION y LUC.

- Finalmente, las exactitudes de clasificación obtenidas con el conjunto total de atributos son similares a las obtenidas con *SP* para catorce conjuntos en prácticamente todas las pruebas, mientras que en siete ocasiones existen diferencias significativas con los tres rankings para los tres clasificadores. En el resto de bases de datos, lo normal es que pierda con un atributo, y esta diferencia se vaya acortando al aumentar el número de atributos. Se observan bases de datos donde la diferencia con respecto a la base de datos completa depende del clasificador aplicado (AUT, COL, CRG y SON).

Se puede concluir que, evaluando los resultados a nivel global se observa similitud en la comparación de los valores de clasificación entre los distintos algoritmos. *SP* tiene un comportamiento intermedio, es decir, ofrece mejor tasa de aciertos que *RL* y *CH*, y peor que *IG* y *SU*. Este último reporta el mejor comportamiento.

La tabla 4.4 suministra el tiempo de ejecución en milisegundos para cada uno de los algoritmos de selección de atributos. Cada valor individual corresponde al tiempo de ejecución, expresado en milisegundos, tomado por cada algoritmo de selección. Esta valor es el promedio de las tres pruebas (1, 3 y 5 atributos) que se realizan para cada clasificador (NB, C4 e IB), es decir, de nueve resultados. Además, si se le une que cada prueba se realiza mediante diez ejecuciones de validación cruzada en la selección con diez carpetas, para cada base de datos se tiene el tiempo promedio de novecientas ($9 \times 10 \times 10 = 900$) creaciones del ranking con cada

Tabla 4.1: Comparativa del criterio *SP*. Tasas de aciertos obtenidas con el clasificador NB en conjuntos de datos pequeños de *UCI* mediante 10×10CV. Por columnas, se distinguen tres grupos de resultados, con 1,3 y 5 atributos. La última columna muestra el resultado del clasificador con la bases de datos original; la penúltima fila, el promedio de las exactitudes de todas las bases de datos; y la última fila, las veces que *SP* gana o pierde significativamente.

Datos	1^{er} atributo del ranking					3 primeros atributos del ranking					5 primeros atributos del ranking					Orig.
	SP	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	NB
ANN	76,17	76,17	75,66	74,14 •	78,07 ◦	63,51	75,02 ◦	57,77 •	39,92 •	57,77 •	59,63	90,51 ◦	73,22 ◦	59,96	66,47 ◦	75,03
AUD	46,46	46,46	45,26	30,52 •	45,26	55,10	66,83 ◦	51,72	31,77 •	66,39 ◦	60,06	69,53 ◦	67,49 ◦	32,43 •	70,05 ◦	72,64
AUT	44,65	45,38	43,88	44,50	41,73	44,63	55,43 ◦	54,81 ◦	54,35 ◦	49,15	47,45	62,40 ◦	58,26 ◦	58,70 ◦	60,36 ◦	57,41
BCA	66,70	71,05	68,17	68,19	67,36	73,39	70,09	71,73	72,59	73,78	72,87	69,60	73,01	73,01	73,01	72,70
BCW	92,01	90,12	92,61	92,71	92,71	94,71	96,02	95,48	95,50	95,11	95,72	96,28	95,85	95,85	95,91	96,07
COL	81,52	81,52	81,52	81,52	81,52	81,00	82,72	81,55	81,60	82,85	82,40	84,32	83,51	83,68	82,29	78,70
COO	63,38	66,19	63,38	63,38	66,19	65,21	65,63	65,43	65,43	65,43	64,59	65,73	65,60	65,73	65,41	66,21
CRR	85,51	85,51	85,51	85,51	85,51	81,46	84,96 ◦	81,46	81,46	81,46	74,91	85,39 ◦	77,59	76,20	74,97	77,86
CRG	69,92	68,83	68,83	68,83	68,83	71,95	72,19	72,74	72,47	73,41	71,88	72,79	73,95	73,85	73,58	75,16
DIA	73,45	74,72	74,72	74,72	74,72	76,38	75,93	75,43	76,02	76,39	76,09	74,96	74,88	74,74	74,65	75,75
GLA	40,05	35,11	43,85	42,03	35,11	53,24	45,63 •	49,21	52,48	47,11	50,44	48,00	49,21	48,89	48,74	49,45
GL2	52,68	60,68 ◦	56,17	56,17	56,85	67,47	64,87	67,78	67,78	67,54	63,65	63,99	63,99	63,99	63,99	62,43
HEC	73,16	75,49	72,90	73,09	73,86	79,71	77,40	80,00	79,97	79,20	81,52	82,52	82,32	82,28	81,81	83,34
HEH	81,01	78,58	76,00 •	76,68	81,01	81,91	79,67	81,91	81,91	81,91	82,59	81,54	81,34	81,34	81,34	83,95
HES	69,85	63,74	72,00	72,04	71,67	76,59	75,81	79,74	79,85 ◦	79,59	82,70	81,78	82,67	82,67	82,56	83,59
HPT	79,17	79,31	80,01	79,95	79,43	83,47	79,17	83,35	83,61	83,61	82,70	83,26	82,26	82,57	81,55	83,81
ION	75,52	74,41	78,67	75,38	80,24	78,66	83,97	85,90 ◦	86,24 ◦	84,11	77,09	89,38 ◦	85,30 ◦	85,02 ◦	86,72 ◦	82,17
LAB	79,50	69,07	79,50	79,50	79,50	85,00	80,27	83,30	84,30	85,00	87,33	84,70	87,23	87,17	87,83	93,57
LUC	50,42	49,67	43,25	41,67	43,00	60,75	65,17	61,58	61,58	62,42	63,83	70,83	72,42	71,75	69,08	55,58
LYM	74,44	73,82	74,44	50,98 •	74,44	74,49	80,10	74,85	72,60	77,56	77,26	77,16	75,40	77,81	75,19	83,13
PRT	25,22	23,33	23,66	28,24	26,32	32,18	30,63	32,19	34,40	32,16	37,67	36,29	38,27	38,50	38,50	49,71
SEG	53,23	53,39	56,87 ◦	56,87 ◦	55,78 ◦	77,10	72,58 •	63,95 •	70,70 •	70,66 •	76,31	69,57 •	69,01 •	69,10 •	69,18 •	80,17
SON	70,61	73,03	70,32	70,32	70,28	69,24	69,67	68,74	68,70	68,45	68,99	70,85	68,60	68,45	68,35	67,71
SOY	30,85	31,89	33,75 ◦	27,33 •	33,75 ◦	53,54	59,87 ◦	64,36 ◦	51,42	64,36 ◦	70,23	69,24	65,71 •	72,72	69,24	92,94
VEH	40,98	37,21 •	40,96	40,93	40,89	40,83	42,20	40,11	40,11	40,78	41,22	41,18	40,26	40,26	40,26	44,68
VOT	95,63	95,63	95,63	95,63	95,63	94,12	95,17	94,71	94,62	94,71	92,28	93,88	92,71	92,28	92,71	90,02
VOW	34,74	30,96 •	34,47	34,74	34,74	56,81	55,84	57,86	57,69	57,86	63,13	63,13	62,38	62,19	62,39	62,90
WIN	80,33	77,31	80,33	80,33	80,33	93,78	93,01	92,57	94,31	85,65 •	94,42	96,85	93,20	93,14	94,04	96,91
ZOO	61,80	60,43	73,30 ◦	73,30 ◦	73,30 ◦	86,13	70,65 •	86,35	82,46	86,35	86,26	86,17	86,15	90,41 ◦	86,15	95,07
Pr.	64,45	63,76	65,02	63,42	65,10	70,77	71,26	70,92	69,51	71,41	71,90	74,55	73,17	71,89	72,98	75,47
G/P		2/1	1/3	4/2	0/4		3/5	2/3	3/3	3/2		1/5	2/4	2/3	1/4	

SP–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica. ◦, • mejora o empeora significativamente con respecto a *SP*.

Tabla 4.2: Comparativa del criterio *SP*. Tasas de aciertos obtenidas con el clasificador C4 en conjuntos de datos pequeños de *UCI* mediante 10×10CV. Por columnas, se distinguen tres grupos de resultados, con 1,3 y 5 atributos. La última columna muestra el resultado del clasificador con la bases de datos original; la penúltima fila, el promedio de las exactitudes de todas las bases de datos; y la última fila, las veces que *SP* gana o pierde significativamente.

Datos	1^{er} atributo del ranking					3 primeros atributos del ranking					5 primeros atributos del ranking					Orig.
	SP	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	C4
ANN	76,19	76,17	78,31 ∘	77,23 ∘	80,07 ∘	80,91	76,17 •	86,03 ∘	86,16 ∘	86,01 ∘	86,35	82,18 •	90,76 ∘	89,64 ∘	90,18 ∘	92,35
AUD	46,46	46,46	45,26	30,70 •	45,26	54,34	68,16 ∘	53,85	34,11 •	67,80 ∘	60,29	69,75 ∘	68,85 ∘	34,11 •	71,51 ∘	77,26
AUT	68,61	45,38 •	60,59	60,04 •	68,89	74,97	57,77 •	74,79	75,27	75,81	75,75	63,80 •	77,31	78,92	77,61	81,77
BCA	66,04	68,09	66,53	66,39	66,46	73,12	69,91	72,10	72,84	73,40	74,39	69,87 •	72,94	72,94	72,94	74,28
BCW	91,03	90,26	91,47	91,57	91,57	94,66	95,14	95,08	95,07	94,68	94,74	95,05	94,75	94,75	95,02	95,01
COL	81,52	81,52	81,52	81,52	81,52	80,92	85,80 ∘	81,52	81,52	84,53 ∘	81,22	85,34 ∘	84,64 ∘	84,15 ∘	85,34 ∘	85,16
COO	66,31	66,31	66,31	66,31	66,31	66,14	66,20	66,31	66,31	66,31	67,50	66,42	66,20	65,87	65,87	64,71
CRR	85,51	85,51	85,51	85,51	85,51	85,51	85,51	85,51	85,51	85,51	85,13	84,81	85,20	85,20	85,13	85,57
CRG	69,91	70,00	70,00	70,00	70,00	71,32	72,51	71,60	71,67	72,08	70,02	71,85	73,95 ∘	73,94 ∘	74,04 ∘	71,25
DIA	72,14	73,27	73,27	73,27	73,27	74,48	74,11	74,17	74,30	74,65	73,85	74,28	73,91	73,92	73,87	74,49
GLA	54,54	44,50 •	49,46	47,03 •	44,50 •	68,50	69,00	66,61	60,57	67,17	68,22	68,82	68,19	68,20	68,01	67,63
GL2	76,81	58,71 •	73,67	73,67	73,18	80,39	78,52	80,39	80,39	80,08	80,75	79,97	79,97	79,97	80,03	78,15
HEC	73,16	75,49	72,90	73,09	74,19	80,14	73,45 •	80,53	80,73	79,76	79,41	81,18	79,74	79,90	79,67	76,94
HEH	80,98	78,58	76,11 •	76,78	80,98	79,51	78,68	79,51	79,51	79,51	78,69	78,59	78,12	78,12	78,12	80,22
HES	69,70	63,67	72,30	72,30	71,85	75,52	75,56	84,96 ∘	85,22 ∘	84,63 ∘	82,70	81,07	82,11	82,11	82,00	78,15
HPT	78,06	79,31	82,10	81,92	79,32	83,18	78,16	82,02	82,73	82,67	81,90	83,60	83,20	83,07	83,00	79,22
ION	84,53	81,54	82,54	83,45	82,17	82,65	87,66 ∘	89,80 ∘	89,40 ∘	88,66 ∘	83,42	91,68 ∘	89,60 ∘	89,37 ∘	90,26 ∘	89,74
LAB	81,60	69,20 •	81,60	81,60	81,60	82,33	78,03	78,83	80,83	82,00	78,77	79,40	78,97	78,43	78,07	78,60
LUC	56,00	45,92	40,25	41,58	41,58	51,00	64,33	56,67	59,17	58,08	52,25	66,67	66,50	66,92	60,42	45,00
LYM	74,77	74,15	74,77	52,92 •	74,77	76,98	78,80	76,85	71,40	77,30	74,02	76,76	75,40	74,81	76,28	75,84
PRT	24,89	24,60	24,60	28,24 ∘	26,38	31,95	30,12	34,16	32,90	34,08	35,34	33,69	38,56	39,00	38,80	41,39
SEG	63,79	62,50	65,57	65,57	64,96	91,01	85,62 •	85,44 •	87,19 •	87,16 •	91,18	88,29 •	88,24 •	88,27 •	88,27 •	96,79
SON	71,96	70,37	72,20	72,20	72,01	70,90	71,62	71,20	71,24	71,05	70,79	72,05	70,24	70,24	70,47	73,61
SOY	30,85	31,57	33,38 ∘	27,33 •	33,38 ∘	51,60	62,09 ∘	66,91 ∘	52,32	66,91 ∘	71,38	71,40	68,77	74,52 ∘	71,40	91,78
VEH	53,32	52,16	53,32	53,31	53,17	54,98	63,08 ∘	60,18 ∘	60,18 ∘	57,07	65,31	63,36	58,65 •	58,65 •	58,69 •	72,28
VOT	95,63	95,63	95,63	95,63	95,63	95,63	95,56	95,63	95,63	95,63	95,63	95,93	95,63	95,63	95,63	96,57
VOW	33,51	30,22 •	33,28	33,51	33,51	69,23	68,02	70,58	70,37	70,55	76,99	77,03	76,51	76,85	76,57	80,20
WIN	80,11	76,41	80,11	80,11	80,11	94,89	92,97	93,59	95,00	89,36 •	93,72	94,44	93,82	93,99	93,44	92,99
ZOO	61,70	60,43	71,30 ∘	71,30 ∘	71,30 ∘	85,12	71,65 •	89,08 ∘	83,71	89,08 ∘	86,53	87,16	87,61	89,97	87,61	92,61
Pr.	67,92	64,76	67,37	66,00	67,71	74,55	74,28	76,00	74,53	76,60	76,42	77,05	77,53	76,60	77,53	78,95
G/P		5/0	1/3	5/3	1/3		5/5	1/6	2/4	2/7		4/3	2/5	3/5	2/5	

SP–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica. ∘, • mejora o empeora significativamente con respecto a *SP*.

Tabla 4.3: Comparativa del criterio *SP*. Tasas de aciertos obtenidas con el clasificador IB en conjuntos de datos pequeños de *UCI* mediante 10×10CV. Por columnas, se distinguen tres grupos de resultados, con 1,3 y 5 atributos. La última columna muestra el resultado del clasificador con la bases de datos original; la penúltima fila, el promedio de las exactitudes de todas las bases de datos; y la última fila, las veces que *SP* gana o pierde significativamente.

Datos	1er atributo del ranking					3 primeros atributos del ranking					5 primeros atributos del ranking					Orig.
	SP	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	IB
ANN	75,59	70,35	71,43	63,32	67,18	84,83	81,27 •	85,06	83,28	85,06	90,99	87,80 •	91,74	90,58	91,23	95,45
AUD	32,39	28,84	28,12	21,63 •	28,12	41,82	59,24 ◦	45,65	26,49 •	58,84 ◦	50,94	63,32	61,67	26,54 •	63,49	75,29
AUT	70,96	31,80 •	60,15	61,71	70,78	76,67	57,57 •	78,98	79,16	79,32	78,66	78,39	80,45	79,63	82,06	74,55
BCA	54,57	62,18	61,67	61,72	59,12	57,94	60,48	63,74	64,57	64,51	65,08	64,83	65,55	65,62	65,83	68,58
BCW	87,72	84,82	89,23	89,39	89,39	92,30	94,98	93,82	93,89	93,88	93,76	95,94 ◦	93,36	93,36	95,01	95,45
COL	70,76	69,19	69,19	69,19	69,19	77,30	77,31	76,46	76,41	77,98	78,48	80,60	80,15	79,39	81,16	79,11
COO	60,54	65,36	58,11	58,11	65,36	66,58	63,18	56,52 •	56,52 •	56,52 •	66,96	63,93	57,70 •	59,17 •	54,23 •	64,96
CRR	63,06	77,07	77,07	77,07	77,07	68,22	76,81	79,33	79,33	79,33	79,20	81,35	79,67	78,96	78,52	81,57
CRG	60,69	62,37	62,37	62,37	62,37	62,42	64,87	65,27	65,36	66,21	66,58	65,08	68,75	68,46	68,49	71,88
DIA	69,08	64,46	64,46	64,46	64,46	69,89	68,33	70,09	70,28	70,35	68,98	65,60	69,13	68,85	68,58	70,62
GLA	44,54	35,17	39,49	40,73	35,17	65,30	66,13	64,73	51,41 •	64,40	75,65	67,66 •	71,57	69,84	69,43	69,95
GL2	72,51	52,56 •	64,37	64,37	63,68	76,15	80,76	76,28	76,28	76,15	80,07	87,11 ◦	87,06	87,06	87,06	77,71
HEC	63,72	63,37	61,95	61,94	63,35	74,31	71,19	74,26	74,20	72,72	75,19	76,24	77,05	76,92	76,38	76,06
HEH	51,60	69,76 ◦	68,26	68,27	70,89	69,54	72,72	75,57	75,57	75,57	73,14	76,40	78,44	78,44	78,44	78,33
HES	64,89	57,48	63,41	63,11	62,59	70,15	66,19	75,37	75,41	74,93	73,48	73,81	76,37	76,37	76,81	76,15
HPT	76,50	69,14	79,37	79,30	75,40	77,99	72,69	77,54	77,92	77,91	81,45	76,73	80,42	80,63	81,02	81,40
ION	80,69	71,96 •	76,93	77,67	75,19	80,98	83,73	87,35 ◦	88,17 ◦	84,88	84,45	90,46	89,23 ◦	89,60 ◦	88,40	87,10
LAB	64,27	62,77	72,73	72,73	72,73	79,97	73,93	78,77	79,83	80,87	85,87	81,87	86,23	88,20	86,90	84,30
LUC	48,83	41,50	39,75	41,08	39,83	50,33	57,92	50,33	54,42	52,00	46,50	62,50	59,08	63,42	53,42	49,83
LYM	62,40	65,94	66,21	48,59	66,21	73,65	75,39	71,07	64,12	74,29	78,90	75,94	73,35	65,20 •	75,55	81,54
PRT	11,97	13,24	13,69	14,41	13,67	17,91	19,50	21,30	23,67	21,01	22,12	23,69	26,23	26,46	26,40	34,64
SEG	64,19	60,55	59,10 •	59,10 •	60,24 •	91,07	85,13 •	89,19 •	88,70 •	88,72 •	91,00	89,71 •	89,87	89,88	89,81	97,15
SON	65,93	66,46	67,13	67,13	67,18	67,95	67,51	71,28	71,19	71,70	71,35	74,99	69,21	69,49	68,40	86,17
SOY	20,38	22,15	22,93	20,38	22,93	43,16	51,82 ◦	56,82 ◦	43,67	56,82 ◦	64,47	61,01	61,36	63,64	61,01	90,17
VEH	44,88	42,51	44,09	44,06	44,21	54,69	57,33	58,70 ◦	58,70 ◦	55,91	63,55	58,86 •	58,29 •	58,29 •	58,29 •	69,59
VOT	92,22	90,95	90,95	90,95	90,95	92,21	92,88	93,48	93,45	93,48	93,64	94,69	93,69	93,98	93,66	92,23
VOW	28,96	23,35 •	28,30	28,65	28,65	79,69	78,39	81,74	81,68	81,74	94,34	94,34	94,74	94,96	94,74	99,05
WIN	69,47	66,69	69,88	69,88	69,88	95,16	91,09	92,80	95,23	90,56	94,65	94,69	95,78	95,96	95,22	95,57
ZOO	46,56	53,89 ◦	59,40 ◦	59,40 ◦	59,40 ◦	74,57	62,05 •	86,75 ◦	75,51	86,75 ◦	81,94	80,93	86,65	88,34	86,55	96,55
Pr.	59,31	56,75	59,65	58,65	59,83	70,09	70,01	72,35	70,50	72,84	74,88	75,46	75,96	74,73	75,73	79,34
G/P		4/2	2/1	2/1	1/1		4/2	2/4	4/2	3/4		4/3	2/1	4/1	2/0	

SP–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica. ◦, • mejora o empeora significativamente con respecto a *SP*.

algoritmo. La última fila refleja el tiempo empleado por cada uno de los algoritmo de ranking con el total de las bases de datos. Hay dos métodos que sobresalen por diferentes motivos, *SP* por su rapidez y *RL* por su lentitud (SP–129,06; RL–34098,93), mientras que los tres restantes están muy igualados (aprox. 640). El algoritmo *RL* es el que más tiempo necesita con una gran diferencia, el criterio *SP* es el más rápido en casi todas las bases de datos, necesitando en global el 20 % del tiempo empleado por los demás (*IG*, *CH* y *SU*). Esta diferencia en el tiempo necesario para elaborar el ranking es más notable en las bases de datos con mayor número de atributos numéricos y de ejemplos.

Bases de datos grandes con los k primeros atributos del ranking

La tabla 4.5 proporciona para los tres clasificadores, NB, C4 e IB, la tasa de aciertos media obtenida por cada uno de los algoritmos de selección, con los primeros cinco, diez y veinte atributos del ranking, y la obtenida con el conjunto de datos original. El promedio de ejemplos correctamente clasificados se calcula mediante diez ejecuciones de validación cruzada en la selección con diez carpetas. Se utilizan bases de datos grandes de *UCI*, cuyas características se pueden ver en la tabla A.2. Por columnas, los resultados de los algoritmos (*SP*–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica) se agrupan por número de atributos, siendo para este caso de conjuntos de datos grandes, de 5, 10 y 20 atributos, mostrando en la última columna el resultado del clasificador correspondiente con la bases de datos original. Por filas también se distinguen tres grupos, uno por cada clasificador, mostrando en la última fila de cada grupo el promedio de las exactitudes de todas las bases de datos con el clasificador correspondiente. En la tabla no se muestran los resultados con veinte atributos para la base de datos LET dado que coincide con el total de la BD original. Los símbolos $\circ$ y $\bullet$ respectivamente identifican si la diferencia con respecto a *SP* es estadísticamente mejor o peor, al nivel 0.05 de confianza.

Observando los promedios de precisión predictiva (última fila de cada grupo de clasificación en la tabla 4.5) de los diferentes métodos con cada uno de los clasificadores, para cinco, diez y veinte atributos, se destaca el comportamiento del algoritmo CH, obteniendo una leve ventaja en todos los casos, excepto aplicando el clasificador NB con cinco atributos. El resto de promedios es similar, aunque en segundo lugar estarían los obtenidos a partir de los rankings generados con *IG* y *SU*, seguidos por los de SP y en último lugar los de *RL*. En global, aunque los resultados están igualados, no son favorables a *SP*, sobre todo con las bases de datos ARR y LET, donde pierde la mayoría de las veces. Esta situación se puede deber a la localización de atributos relevantes a la cabeza de la lista generada por *SP*, pero no existiendo interacción alguna entre ellos, y probablemente conteniendo la misma información (redundantes), provocando

Tabla 4.4: Comparativa del criterio *SP*. Tiempo (milisegundos) al aplicar cada criterio a conjuntos de datos pequeños de *UCI*. La última fila refleja el tiempo empleado por cada uno de los algoritmo de ranking con el total de las bases de datos.

Datos	*SP*	*RL*	*IG*	*CH*	*SU*
ANN	5,87	3512,94	15,56	14,70	13,16
AUD	6,23	489,90	4,68	4,22	4,29
AUT	2,75	186,61	11,16	11,15	11,63
BCA	1,08	80,84	1,12	1,07	0,72
BCW	2,97	682,44	7,72	7,86	8,35
COL	2,85	386,08	4,45	4,14	4,01
COO	4,12	448,00	5,24	5,27	5,75
CRR	4,65	936,01	10,47	9,57	9,60
CRG	8,80	2416,10	12,45	12,54	11,92
DIA	3,11	763,85	12,06	11,05	10,73
GLA	1,29	91,10	9,33	9,37	9,39
GL2	0,69	45,26	3,86	3,44	3,71
HEC	1,88	167,64	3,65	3,46	3,11
HEH	1,45	158,43	2,82	2,78	3,03
HES	1,51	156,90	4,58	4,63	4,19
HPT	0,87	60,68	2,26	2,19	1,94
ION	5,89	642,76	59,15	59,39	57,25
LAB	0,28	10,71	1,01	1,08	0,97
LUC	0,69	17,03	2,59	2,47	2,50
LYM	1,08	51,87	1,46	1,45	1,22
PRT	1,87	301,19	1,93	1,73	1,37
SEG	32,83	15471,32	255,90	256,29	251,98
SON	6,07	402,39	35,20	36,96	34,51
SOY	8,59	2104,81	5,82	4,98	5,67
VEH	7,04	1987,83	32,13	31,61	30,53
VOT	2,68	305,04	1,74	1,47	1,32
VOW	9,81	2112,45	127,50	128,52	123,09
WIN	1,24	78,25	9,24	9,02	8,72
ZOO	0,90	30,54	1,02	1,06	1,08
Total	129,06	34098,93	646,12	643,48	625,74

unos resultados más pobres en conjunto.

La tabla 4.6 suministra el tiempo de ejecución para cada uno de los algoritmos de selección de atributos. La última fila refleja el tiempo en milisegundos empleado por cada uno de los algoritmo de ranking con el total de las bases de datos. Igual que con las bases de datos pequeñas, hay dos métodos que sobresalen por diferentes motivos (SP: 10218, RL: 4514809), mientras que los tres restantes están muy igualados (IG: 32028, CH: 31988 y SU: 34345). El algoritmo *RL* es el que más tiempo necesita con una diferencia drástica, sin embargo, el criterio *SP* destaca por todo lo contrario. *SP* es el más rápido, seguido de *IG*, *CH* y *SU* que necesitan aproximadamente tres veces más de tiempo, y en último lugar con una enorme diferencia, está *RL*.

4.4.2. Normalizando el ranking

El rango de valores que un criterio de evaluación le asigna a los atributos de una base de datos suele ser distinto al rango de los demás criterios. Normalmente, se fija un umbral para formar un subconjunto con aquellos atributos que estén por encima [70], pero al establecer un umbral específico para cada una de las medidas, se dificulta la comparación entre ellas. Para evitar este inconveniente, en la comparativa que se presenta a continuación, se normaliza la salida de cada algoritmo, de manera que al mejor atributo se le hace corresponder con el valor 1 y al peor con 0, y se establece un único valor umbral para todos los rankings, escogiéndose todos aquellos atributos que estén por encima del límite establecido [144]. En los experimentos de este apartado, se establece el valor 0,75 como umbral, tanto para las pruebas realizadas con las bases de datos pequeñas, como para las realizadas con las grandes.

Para cada base de datos y para cada algoritmo de aprendizaje, se realizaron diez ejecuciones con validación cruzada en la selección con diez carpetas (ver figura 3.5 en la sección 3.6, página 63), es decir, en cada una de las 100 pruebas, se pasa a los diferentes algoritmos de selección el 90 % de los datos, generando un ranking y escogiendo un subconjunto de atributos que se utiliza para reducir tanto el conjunto de entrenamiento, como el de prueba. A continuación se entrena un clasificador con el conjunto de entrenamiento reducido, y se obtienen los resultados sobre el conjunto de prueba reducido. Se almacenó la exactitud y el tiempo tomado por cada algoritmo en la elaboración del ranking, y si procede, el número de atributos seleccionados.

Bases de datos pequeñas con normalización de ranking

La tabla 4.7 proporciona para los tres clasificadores, NB, C4 e IB, la tasa de aciertos media obtenida por cada uno de los algoritmos de ranking, y la conseguida con el conjunto de datos ori-

Tabla 4.5: Comparativa del criterio *SP*. Tasas de aciertos obtenidas en bases de datos grandes de *UCI* mediante 10×10CV. Por columnas, se distinguen resultados con 5, 10 y 20 atributos; por filas, un grupo por cada clasificador.

	Datos	5 primeros atributos del ranking					10 primeros atributos del ranking					20 primeros atributos del ranking					Orig.
		SP	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	*SP*	*RL*	*IG*	*CH*	*SU*	
Clasif. NB	ADS	94,28	90,44 •	94,19	94,35	94,43 ◦	93,65	90,10 •	94,11	94,12	93,66	92,65	92,72	94,28 ◦	94,27 ◦	92,64	96,59
	ARR	55,99	59,34 ◦	66,37 ◦	58,87 ◦	66,13 ◦	55,59	60,59 ◦	64,20 ◦	58,55	63,33 ◦	54,77	65,74 ◦	63,51 ◦	57,69	64,18 ◦	61,47
	HYP	93,00	93,17	94,83 ◦	94,82 ◦	94,83 ◦	93,08	93,75	95,30 ◦	95,30 ◦	95,29 ◦	93,10	95,22 ◦	95,34 ◦	95,32 ◦	95,36 ◦	95,30
	ISO	27,49	29,97	22,85 •	24,24 •	20,58 •	35,11	39,04 ◦	26,52 •	39,87 ◦	25,79 •	41,76	46,17 ◦	36,74 •	57,40 ◦	39,22 •	83,54
	KRV	85,91	90,64 ◦	86,70	86,70	89,07 ◦	87,65	93,89 ◦	88,46	88,69	89,17 ◦	87,80	90,72 ◦	87,73	87,78	87,73	87,79
	LET	38,39	49,19 ◦	45,85 ◦	45,88 ◦	45,88 ◦	60,60	65,44 ◦	65,45 ◦	65,45 ◦	65,45 ◦	coincide con la BD original*					64,07
	MUL	85,42	52,35 •	88,23 ◦	90,11 ◦	88,23 ◦	93,00	73,76 •	90,80 •	94,71 ◦	91,03 •	94,78	77,09 •	96,18 ◦	95,35	95,92 ◦	95,28
	MUS	95,67	98,03 ◦	95,67	95,67	98,95 ◦	96,62	96,94	97,04	95,40 •	95,40 •	95,74	95,71 •	95,60 •	95,60 •	95,60 •	95,76
	MUK	88,64	83,26 •	87,78 •	87,30 •	86,26 •	88,08	82,14 •	87,98	88,49	85,89 •	82,46	82,92	83,27	84,65 ◦	83,81 ◦	83,91
	SIC	93,88	93,88	94,61	94,76	94,81	93,88	95,37	94,08	94,29	94,11	93,06	93,93	93,67	93,67	93,69	92,75
	SPL	90,73	91,18	91,18	91,18	91,18	94,34	94,33	94,40	94,47	94,40	96,19	96,32	96,18	96,25	96,18	95,42
	WAV	70,71	76,41 ◦	67,47 •	67,95 •	67,69 •	77,20	81,55 ◦	75,19 •	75,11 •	76,49 •	80,04	80,08	80,05	80,05	80,05	80,01
	Pr.	76,68	75,66	77,98	77,65	78,17	80,73	80,58	81,13	82,04	80,83	82,94	83,33	83,87	85,28	84,03	85,99
Clasif. C4	ADS	94,44	92,20 •	95,41 ◦	95,55 ◦	94,49	94,51	92,94 •	96,20 ◦	96,20 ◦	94,51	95,15	95,24	96,21 ◦	96,20 ◦	95,16	96,83
	ARR	56,37	61,30 ◦	64,96 ◦	59,92	65,01 ◦	56,02	62,43 ◦	65,20 ◦	59,70	64,74 ◦	58,21	68,39 ◦	66,42 ◦	57,62	66,50 ◦	65,78
	HYP	94,14	93,39 •	98,02 ◦	97,98 ◦	98,02 ◦	94,62	93,66 •	99,36 ◦	99,36 ◦	99,36 ◦	94,93	99,33 ◦	99,56 ◦	99,56 ◦	99,57 ◦	99,54
	ISO	32,72	32,96	25,85 •	29,32 •	22,92 •	44,57	43,52	32,27 •	45,69	32,30 •	52,43	52,00	46,75 •	64,95 ◦	48,20 •	78,28
	KRV	90,39	93,15 ◦	90,40	90,40	94,09 ◦	95,61	97,43 ◦	93,95 •	94,43 •	94,03 •	96,76	97,82	96,70	96,77	96,69	99,44
	LET	67,10	80,50 ◦	68,09 ◦	67,93	67,92	85,43	88,30 ◦	87,76 ◦	87,76 ◦	87,77 ◦	coincide con la BD original*					88,03
	MUL	85,36	56,24 •	87,91 ◦	89,40 ◦	87,90 ◦	90,04	79,94 •	89,85	93,07 ◦	89,81	91,03	85,98 •	94,11 ◦	93,59 ◦	94,01 ◦	94,63
	MUS	99,90	99,90	99,90	99,90	100,00 ◦	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
	MUK	95,57	94,63 •	95,60	95,65	95,39	95,89	95,14 •	95,90	96,00	95,88	96,30	95,20 •	96,26	96,45	96,28	96,83
	SIC	93,88	93,88	97,66 ◦	98,11 ◦	97,86 ◦	93,88	96,44 ◦	98,14 ◦	98,18 ◦	98,16 ◦	93,88	98,20 ◦	98,13 ◦	98,13 ◦	98,14 ◦	98,72
	SPL	91,10	90,92 •	90,92 •	90,92 •	90,92 •	94,12	94,14	94,22	94,22	94,22	94,27	94,30	94,26	94,28	94,26	94,17
	WAV	71,88	74,30 ◦	68,84 •	69,34 •	69,23 •	76,60	77,20	74,54 •	74,31 •	76,28	76,70	76,60	76,65	76,66	76,66	75,25
	Pr.	81,07	80,28	81,96	82,04	81,98	85,11	85,10	85,62	86,58	85,59	86,33	87,55	87,73	88,56	87,77	90,63
Clasif. IB	ADS	90,34	90,44	95,27	95,27	90,55	90,46	91,56	95,81	95,81	91,37	91,11	93,02	95,88	95,87	92,89	95,97
	ARR	43,90	47,85 ◦	56,07 ◦	51,37 ◦	55,11 ◦	46,71	50,27 ◦	57,02 ◦	52,84 ◦	56,55 ◦	51,06	56,35 ◦	57,81 ◦	51,78	57,59 ◦	53,20
	HYP	87,01	89,32	94,27 ◦	92,09 ◦	94,27 ◦	88,13	90,26	92,67 ◦	92,59 ◦	92,82 ◦	90,28	91,42	92,68	92,60	92,76	91,52
	ISO	31,76	30,78	24,19 •	27,39 •	21,47 •	45,62	44,67	34,44 •	45,93	32,50 •	56,40	53,69 •	49,90 •	67,82 ◦	50,35 •	86,39
	KRV	86,12	89,99	86,00	86,00	91,08	92,97	95,71 ◦	91,98 •	93,03	91,00 •	95,08	96,97	95,43	95,38	95,44	90,61
	LET	55,88	80,49 ◦	64,79 ◦	64,53 ◦	64,53 ◦	92,28	96,33 ◦	95,78 ◦	95,78 ◦	95,78 ◦	coincide con la BD original*					95,99
	MUL	88,98	51,59 •	87,59 •	90,06	87,60 •	95,57	75,72 •	94,19 •	96,64 ◦	93,53 •	97,30	83,73 •	97,87	97,16	97,77	97,90
	MUS	99,90	99,83 •	99,85	99,85	100,00 ◦	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00
	MUK	94,39	93,34 •	93,98 •	94,26	93,75 •	94,67	93,38 •	94,96	95,13	94,64	94,61	94,03 •	95,15 ◦	95,30 ◦	95,53 ◦	95,70
	SIC	91,95	89,38	95,71	96,78	96,49	91,71	94,15	96,66	96,48	96,60	90,55	96,15 ◦	96,28 ◦	96,24 ◦	96,38 ◦	96,10
	SPL	86,12	86,18	86,18	86,18	86,18	85,69	86,17	86,06	86,12	86,06	81,56	82,08	82,38	82,41	82,38	75,97
	WAV	66,56	70,69 ◦	62,66 •	63,20 •	62,91 •	76,36	78,58	74,07 •	73,95 •	75,78	78,55	78,71	77,77 •	77,77 •	77,77 •	73,41
	Pr.	76,91	76,66	78,88	78,92	78,66	83,35	83,07	84,47	85,36	83,89	84,23	84,20	85,56	86,58	85,35	87,73

SP–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica. ◦, • mejora o empeora significativamente con respecto a *SP*.

* LET tiene 20 atributos.

Tabla 4.6: Comparativa del criterio *SP*. Tiempo (milisegundos) al aplicar cada criterio a bases de datos grandes de *UCI*. La última fila refleja el tiempo empleado con el total de las bases de datos.

Datos	*SP*	*RL*	*IG*	*CH*	*SU*
ADS	4918,82	1335509,12	921,94	918,25	1952,19
ARR	80,11	10198,03	242,29	241,89	245,16
HYP	77,94	42959,22	60,97	62,48	73,16
ISO	1186,23	247986,94	16198,27	16160,79	16700,47
KRV	72,64	32703,80	25,17	25,54	40,06
LET	588,30	1043795,69	1467,01	1464,23	1483,98
MUL	1062,66	383640,14	7866,24	7849,82	8226,28
MUS	210,17	143761,33	49,95	46,79	57,88
MUK	1550,96	1030158,25	4571,69	4587,16	4859,23
SIC	74,39	42819,88	54,67	53,82	65,58
SPL	155,53	54259,07	40,05	39,80	77,42
WAV	241,09	147018,00	529,76	536,96	563,22
Total	10218,83	4514809,45	32027,99	31987,53	34344,63

ginal. El promedio de ejemplos correctamente clasificados se calcula mediante diez ejecuciones de validación cruzada en la selección con diez carpetas. Se utilizan bases de datos pequeñas de *UCI*, cuyas características se pueden ver en la tabla A.1. Por columnas, los resultados de los algoritmos (*SP–SOAP*, *RL–Relief*, *IG*–Ganancia de Información, *CH–Chi2* y *SU*–Incertidumbre simétrica) se agrupan por clasificador, mostrando en la última columna de cada grupo el resultado del clasificador correspondiente con la bases de datos original. La penúltima fila de la tabla muestra el promedio de las exactitudes de todas las bases de datos, y la última fila, las veces que *SP* gana o pierde significativamente. Los símbolos $\circ$ y $\bullet$ respectivamente identifican si la diferencia con respecto a *SP* es estadísticamente mejor o peor, al nivel 0.05 de confianza.

El análisis comparativo de los datos mostrados en la tabla 4.7 por los distintos algoritmos, se puede realizar desde distintos puntos de vista: por clasificador; por base de datos; con respecto al conjunto original de atributos; por combinación de algunos de los anteriores; y de forma global.

Los resultados correspondientes a los clasificadores NB, C4 e IB se pueden ver en el primer, segundo y tercer grupo vertical de la tabla, respectivamente. Aunque no existen grandes diferencias en los promedios de las exactitudes obtenidas (penúltima fila), ni en el computo de veces que gana o pierde SP frente al resto de algoritmos (ver última fila). Se obtienen mejores resultados con los rankings normalizados generados por *SP*, que los creados a partir de las medidas *RL* y CH; sin embargo *SP* pierde frente a *IG* y *SU*.

Tabla 4.7: Comparativa del criterio *SP*. Tasas de aciertos obtenidas con los tres clasificadores normalizando el ranking en bases de datos pequeñas de *UCI* mediante 10×10CV. Por columnas, se distinguen tres grupos de resultados: NB, C4 y IB. La penúltima fila muestra el promedio de las exactitudes de todas las bases de datos; y la última fila, las veces que *SP* gana o pierde significativamente.

Datos	NB						C4						IB					
	SP	*RL*	*IG*	*CH*	*SU*	Orig.	*SP*	*RL*	*IG*	*CH*	*SU*	Orig.	*SP*	*RL*	*IG*	*CH*	*SU*	Orig.
ANN	54,54	76,82 ○	75,66 ○	31,14 •	78,07 ○	75,03 ○	90,09	76,17 •	78,31 •	81,31 •	80,07 •	92,35	92,06	74,57 •	71,43 •	71,17 •	67,18 •	95,45 ○
AUD	46,46	46,46	52,00 ○	30,52 •	52,00 ○	72,64 ○	46,46	46,46	53,51 ○	30,70 •	53,51 ○	77,26 ○	32,39	28,84	41,00	21,63	41,00	75,29 ○
AUT	41,94	52,61	53,90 ○	48,31	51,79 ○	57,41 ○	73,65	50,69 •	75,29	67,27	76,20	81,77 ○	74,40	45,80 •	78,34	73,25	79,72	74,55
BCA	69,39	70,60	71,42	73,03 ○	73,12	72,70	68,87	68,44	71,58	72,21	72,95 ○	74,28 ○	55,18	62,40	64,47	63,77	64,61	68,58
BCW	95,88	94,29	94,95	95,27	95,39	96,07	94,95	94,48	94,64	94,87	94,57	95,01	93,91	92,19	93,32	93,51	94,48	95,45
COL	81,52	81,52	81,52	81,52	81,52	78,70	81,52	81,52	81,52	81,52	81,52	85,16 ○	70,76	69,19	69,19	69,19	69,19	79,11
COO	65,11	66,16	63,61	62,74	66,19	66,21	66,14	66,31	66,31	66,31	66,31	64,71	65,87	65,33	62,69	62,26	65,36	64,96
CRR	85,51	85,51	85,51	85,51	85,51	77,86 •	85,51	85,51	85,51	85,51	85,51	85,57	63,06	77,07	77,07	77,07	77,07	81,57 ○
CRG	69,92	68,83	68,83	68,83	68,83	75,16 ○	69,91	70,00	70,00	70,00	70,00	71,25	60,69	62,37	62,37	62,37	62,37	71,88 ○
DIA	75,41	74,72	74,72	74,72	74,83	75,75	73,63	73,27	73,27	73,27	73,28	74,49	68,81	64,46 •	64,46 •	64,46 •	64,90	70,62
GLA	58,47	35,11 •	51,04	49,55 •	49,36 •	49,45 •	62,96	44,50 •	65,57	66,40	68,34	67,63	63,35	35,17 •	71,06	66,42	67,00	69,95
GL2	66,97	64,68	57,89 •	61,31	65,97	62,43	80,33	79,12	72,68 •	73,11 •	74,53	78,15	77,26	82,60	69,83	69,10	69,53	77,71
HEC	82,97	72,46 •	77,92	78,42	78,77 •	83,34	77,46	72,03 •	77,86	78,42	78,61	76,94	77,47	66,63 •	72,88	73,37	74,39	76,06
HEH	81,58	78,58	81,91	81,91	80,76	83,95	79,65	78,58	79,51	79,51	79,78	80,22	70,48	69,76	75,57	75,57	72,14	78,33 ○
HES	83,52	74,74 •	78,70 •	78,93 •	78,85 •	83,59	80,93	74,30 •	81,11	82,04	81,33	78,15	76,81	65,15 •	73,70	73,78	73,78	76,15
HPT	82,75	79,02	79,48	80,57	83,15	83,81	82,54	78,92	80,78	80,93	82,65	79,22	80,88	74,79	78,61	78,41	78,85	81,40
ION	83,68	89,81 ○	84,99	85,76	87,04	82,17	90,86	91,37	90,71	90,86	90,28	89,74	87,41	89,60	89,45	89,17	88,49	87,10
LAB	82,53	79,33	80,17	80,50	79,83	93,57	79,50	77,87	80,77	81,20	81,77	78,60	65,10	70,93	71,53	71,17	71,37	84,30 ○
LUC	61,00	57,58	65,08	69,08	64,75	55,58	51,92	59,83	59,50	61,00	57,50	45,00	48,00	48,67	50,92	56,67	52,92	49,83
LYM	72,81	74,23	74,44	56,55 •	72,65	83,13 ○	72,21	72,07	74,84	57,83 •	73,74	75,84	65,21	66,28	66,01	52,70	67,31	81,54 ○
PRT	27,58	28,08	32,40 ○	34,40 ○	35,88 ○	49,71 ○	27,58	28,85	34,11 ○	32,93 ○	36,03 ○	41,39 ○	15,43	16,61	21,89	23,29 ○	24,78 ○	34,64 ○
SEG	76,31	77,88	79,34 ○	68,71 •	76,14	80,17 ○	91,18	96,40 ○	96,48 ○	90,22	94,32 ○	96,79 ○	91,00	96,77 ○	96,91 ○	90,65	94,57 ○	97,15 ○
SON	68,33	70,38	69,57	69,61	69,32	67,71	70,90	70,76	71,05	70,96	71,05	73,61	69,91	68,32	69,91	69,87	70,04	86,17 ○
SOY	53,54	75,52 ○	65,21 ○	72,85 ○	67,79 ○	92,94 ○	51,60	73,10 ○	67,76 ○	74,68 ○	73,37 ○	91,78 ○	43,16	69,99 ○	57,25 ○	64,95 ○	66,65 ○	90,17 ○
VEH	40,98	40,80	40,25	40,20	40,26	44,68 ○	53,32	64,82 ○	58,59 ○	58,35 ○	58,69 ○	72,28 ○	44,88	61,47 ○	58,17 ○	58,29 ○	58,29 ○	69,59 ○
VOT	95,63	95,63	95,63	95,63	95,63	90,02 •	95,63	95,63	95,63	95,63	95,63	96,57	91,41	90,95	90,95	90,95	90,95	92,23
VOW	45,49	63,13 ○	54,10 ○	53,12	54,10 ○	62,90 ○	47,41	77,03 ○	58,54 ○	57,29	58,52 ○	80,20 ○	47,57	94,34 ○	62,27 ○	60,71	62,27 ○	99,05 ○
WIN	91,90	89,82	88,55	82,90 •	84,75 •	96,91 ○	93,49	87,90 •	86,25 •	81,90 •	88,01	92,99	93,76	84,54 •	83,49 •	75,39 •	87,65	95,57
ZOO	85,16	76,26 •	73,49 •	73,30 •	86,05	95,07 ○	83,25	77,25 •	71,49 •	71,30 •	87,81 ○	92,61 ○	75,75	67,86	60,69 •	59,40 •	86,05 ○	96,55 ○
Pr.	69,89	70,36	70,77	67,75	71,67	75,47	73,22	72,87	74,25	72,67	75,38	78,95	67,65	67,68	69,15	67,54	70,45	79,34
G/P		4/4	3/7	8/3	4/6	3/12		7/4	4/6	6/3	1/8	0/10		7/4	4/4	4/3	1/6	0/14

SP–SOAP, *RL–Relief*, *IG*–Ganancia de Inform., *CH–Chi2*, *SU*–Incert. simétrica y Orig–datos originales. ○, • mejora o empeora significativamente con respecto a *SP* al nivel 0,05.

No existen diferencias significativas que se mantengan para los tres clasificadores, excepto en SOY, y en las tres últimas bases de datos de la tabla, VOW, WIN y ZOO. Con la base de datos WIN, *SP* gana en todos los casos frente al resto de algoritmos; con ZOO también gana, excepto frente a *SU*; y *SP* obtiene malos resultados con SOY y VOW, donde pierde en todos los casos. Existe otro grupo de bases de datos, donde *SP* obtiene buenos resultados, especialmente con el clasificador NB, GLA, GL2, HEC, y HES. En el resto, gana frente a unos algoritmos con un clasificador y pierde frente a otros con el mismo u otro clasificador, por ejemplo, ANN, AUD, y AUT.

Se debe bajar el valor del umbral para escoger más atributos del ranking si se desea obtener unos resultados comparables con los obtenidos sobre el conjunto completo de los datos. Se pierde claramente con los tres clasificadores, tanto en el promedio de la tasa de aciertos, como en el computo de bases de datos donde existen diferencias significativas, siendo 3/12, 0/10 y 0/14, el cómputo para los clasificadores NB, C4 e IB respectivamente. Existe margen suficiente para bajar el umbral, elevando así el número de atributos del subconjunto final, e intentar que los resultados se aproximasen a los obtenidos con el conjunto de datos original. En la tabla 4.8, las reducciones realizadas por los distintos algoritmos en las bases de datos son importantes, quedándose con el 21,45 %, 15,48 %, 16,01 %, 16,33 % y 17,95 % de los atributos, *SP*, *RL*, *IG*, *CH* y *SU* respectivamente. El promedio de las reducciones realizadas con *SP* es menor que el resto, debido principalmente a tres bases de datos, BCW, HEC e ION, donde se queda con más del 50 %.

Al igual que en las comparativas anteriores, *SP* y *RL* sobresalen por su rapidez y lentitud respectivamente (*SP*–128,04; *RL*–34001,81), mientras que los tres restantes están muy igualados (aprox. 650). Los tiempos y las proporciones resultantes entre *SP* y los demás son similares a los obtenidos con estas mismas bases de datos en la prueba anterior con los k primeros atributos del ranking (tabla 4.4). El algoritmo *RL* es el que más tiempo necesita con una diferencia drástica, sin embargo, el criterio *SP* destaca por todo lo contrario. *SP* es más rápido que los tres algoritmos restantes (*IG*, *CH* y *SU*) en casi todas las bases de datos, necesitando en global el 19 % del tiempo empleado por los demás. Esta diferencia en el tiempo necesario para elaborar el ranking es más notable en las bases de datos con mayor número de atributos numéricos y de ejemplos.

Bases de datos grandes con normalización de rankings

La tabla 4.9 proporciona para los tres clasificadores, NB, C4 e IB, la tasa de aciertos media obtenida por cada uno de los algoritmos de selección, y la obtenida con el conjunto de datos original. El promedio de ejemplos correctamente clasificados se calcula mediante diez ejecuciones

Tabla 4.8: Comparativa del criterio *SP*. Atributos seleccionados y porcentaje retenido con respecto al original normalizando el ranking en bases de datos pequeñas de *UCI*.

Datos		*SP*		*RL*		*IG*		*CH*		*SU*	
	Ats.	Ats.	%	Ats.	%	Ats.	%	Ats.	%	Ats.	%
ANN	38	11,00	28,95	2,00	5,26	1,00	2,63	2,12	5,58	1,00	2,63
AUD	69	1,00	1,45	1,00	1,45	2,00	2,90	1,00	1,45	2,00	2,90
AUT	25	2,68	10,72	1,71	6,84	3,18	12,72	2,13	8,52	3,58	14,32
BCA	9	1,67	18,56	1,34	14,89	2,52	28,00	2,18	24,22	2,86	31,78
BCW	9	5,55	61,67	1,95	21,67	2,88	32,00	3,76	41,78	4,04	44,89
COL	22	1,00	4,55	1,00	4,55	1,00	4,55	1,00	4,55	1,00	4,55
COO	27	2,88	10,67	1,02	3,78	1,86	6,89	1,74	6,44	1,00	3,70
CRR	15	1,00	6,67	1,00	6,67	1,00	6,67	1,00	6,67	1,00	6,67
CRG	20	1,00	5,00	1,00	5,00	1,00	5,00	1,00	5,00	1,00	5,00
DIA	8	2,53	31,63	1,00	12,50	1,00	12,50	1,00	12,50	1,04	13,00
GLA	9	2,25	25,00	1,00	11,11	4,34	48,22	4,62	51,33	4,16	46,22
GL2	9	3,29	36,56	4,02	44,67	1,76	19,56	1,88	20,89	1,93	21,44
HEC	13	6,59	50,69	1,82	14,00	3,16	24,31	3,20	24,62	3,74	28,77
HEH	13	3,22	24,77	1,00	7,69	3,00	23,08	3,00	23,08	2,31	17,77
HES	13	6,34	48,77	2,61	20,08	2,83	21,77	2,88	22,15	2,95	22,69
HPT	19	4,14	21,79	4,07	21,42	1,49	7,84	1,85	9,74	2,93	15,42
ION	34	25,47	74,91	5,69	16,74	10,32	30,35	13,07	38,44	6,31	18,56
LAB	16	2,12	13,25	2,84	17,75	1,13	7,06	1,23	7,69	1,09	6,81
LUC	56	3,53	6,30	2,09	3,73	3,68	6,57	4,34	7,75	3,87	6,91
LYM	18	1,99	11,06	1,69	9,39	1,02	5,67	1,48	8,22	1,51	8,39
PRT	17	2,04	12,00	2,12	12,47	3,27	19,24	2,94	17,29	3,99	23,47
SEG	19	4,97	26,16	8,07	42,47	9,93	52,26	7,78	40,95	9,32	49,05
SON	60	3,91	6,52	3,19	5,32	2,65	4,42	2,75	4,58	2,76	4,60
SOY	35	3,00	8,57	8,00	22,86	3,30	9,43	5,19	14,83	7,00	20,00
VEH	18	1,00	5,56	5,73	31,83	4,98	27,67	4,87	27,06	5,00	27,78
VOT	16	1,90	11,88	1,00	6,25	1,00	6,25	1,00	6,25	1,00	6,25
VOW	13	1,57	12,08	5,00	38,46	2,00	15,38	1,95	15,00	2,00	15,38
WIN	13	3,00	23,08	2,17	16,69	1,97	15,15	1,46	11,23	3,18	24,46
ZOO	17	3,98	23,41	4,00	23,53	1,05	6,18	1,00	5,88	4,59	27,00
Promedios			21,45		15,48		16,01		16,33		17,95

de validación cruzada en la selección con diez carpetas. Se utilizan bases de datos grandes de *UCI*, cuyas características se pueden ver en la tabla A.2. Por columnas, los resultados de los algoritmos (*SP*–SOAP, *RL*–Relief, *IG*–Ganancia de Información, *CH*–Chi2 y *SU*–Incertidumbre simétrica) se agrupan por clasificador, mostrando en la última columna de cada grupo el resultado del clasificador correspondiente con la bases de datos original. Los símbolos ∘ y • respectivamente identifican si la diferencia con respecto a *SP* es estadísticamente mejor o peor, al nivel 0.05 de confianza. La penúltima fila de la tabla, muestra el promedio de las exactitudes de todas las bases de datos, y la última fila, las veces que *SP* gana o pierde significativamente.

Desde el punto de vista de la tasa de aciertos, sale favorecido el algoritmo *SP*, especialmente aplicando el clasificador C4. La medida *SP* gana o empata en todos los casos, excepto a SU con el clasificador NB (gana 4, pierde 5), al resto de algoritmos. *SP* obtiene las principales diferencias positivas con las bases de datos ISO, KRV y WAV, donde gana en casi todos los casos al resto de algoritmos con los tres clasificadores. Tan sólo con los conjuntos de datos LET y SPL, *SP* pierde frente a todos con los tres clasificadores. En el resto de bases de datos, unas veces gana y otras pierde. Con respecto a los conjuntos de datos originales, la mayoría de las veces *SP* pierde, 5/7, 1/11 y 2/10, con los tres clasificadores respectivamente.

La tabla 4.10 refleja las reducciones realizadas con respecto a la base de datos original. El número de atributos por encima del umbral (0,75) después de normalizar el ranking con *SP*, es algo superior al resto de criterios. El promedio de atributos retenidos con *SP* es mayor debido principalmente a dos bases de datos, ISO y WAV, donde la retención practicada es superior al 25 % del número de atributos del conjunto de datos original.

En cuanto al tiempo empleado (*SP*–10113; *RL*–4497364; *IG*–31897; *CH*–31797; y *SU*–34114) es equivalente al comentado en el apartado de bases de datos grandes con los k primeros atributos. *SP* es más rápido que el resto de algoritmos en casi todas las bases de datos, necesitando en global aproximadamente un 30 % del tiempo empleado por los demás (*IG*, *CH* y *SU*), y en último lugar con una enorme diferencia, está *RL*.

4.4.3. Evaluación del área bajo la curva de comportamiento de clasificación

Comparar dos métodos de selección de subconjuntos de atributos es trivial. Se generan los modelos de clasificación con los subconjuntos de atributos obtenidos y se evalúa la predicción de exactitud de ambos. Sin embargo, no está clara la comparación entre dos métodos de ranking de atributos, dado que la predicción de exactitud de los modelos de clasificación dependen del número (k o valor umbral) y de la calidad de los atributos seleccionados del ranking. El éxito

Tabla 4.9: Comparativa del criterio *SP*. Tasas de aciertos obtenidas con los tres clasificadores normalizando el ranking en bases de datos grandes de *UCI* mediante 10×10CV. Por columnas, se distinguen tres grupos de resultados: NB, C4 e IB. La penúltima fila muestra el promedio de las exactitudes de todas las bases de datos; y la última fila, las veces que *SP* gana o pierde significativamente.

Datos	NB						C4						IB					
	SP	*RL*	*IG*	*CH*	*SU*	Orig.	*SP*	*RL*	*IG*	*CH*	*SU*	Orig.	*SP*	*RL*	*IG*	*CH*	*SU*	Orig.
ADS	93,2	89,9 •	93,4	94,5 ◦	94,4 ◦	96,6 ◦	94,5	91,1 •	94,5	95,9 ◦	94,4 •	96,8 ◦	90,5	87,5	94,6	95,6 ◦	90,6	96,0 ◦
ARR	57,4	54,2 •	66,8 ◦	58,8	65,2 ◦	61,5 ◦	62,6	54,2 •	64,8 ◦	58,6 •	64,4	65,8 ◦	55,6	30,1 •	56,3	49,2 •	54,1	53,2 •
HYP	94,0	92,3 •	94,0	94,0	94,0	95,3 ◦	96,7	92,3 •	96,7	96,7	96,7	99,5 ◦	85,8	86,4	85,8	85,8	85,8	91,5 ◦
ISO	70,7	51,0 •	46,5 •	54,5 •	50,3 •	83,5 ◦	74,8	57,4 •	55,9 •	61,6 •	57,3 •	78,3 ◦	78,6	62,5 •	59,9 •	63,9 •	61,1 •	86,4 ◦
KRV	90,4	90,4	66,1 •	66,4 •	66,1 •	87,8 •	90,4	90,4	66,1 •	66,4 •	66,1 •	99,4 ◦	86,3	84,5	61,0 •	62,1 •	61,0 •	90,6 ◦
LET	29,6	49,3 ◦	59,9 ◦	49,1 ◦	59,9 ◦	64,1 ◦	46,9	80,5 ◦	83,0 ◦	75,5 ◦	83,0 ◦	88,0 ◦	25,9	80,6 ◦	87,4 ◦	76,7 ◦	87,4 ◦	96,0 ◦
MUL	77,0	77,2	81,6 ◦	43,5 •	88,7 ◦	95,3 ◦	76,9	85,7 ◦	85,4 ◦	50,6 •	88,2 ◦	94,6 ◦	76,2	83,7 ◦	82,9 ◦	40,4 •	86,3 ◦	97,9 ◦
MUS	98,6	98,5	98,5	98,5	98,5	95,8 •	98,9	98,5 •	98,5 •	98,5 •	98,5 •	100,0 ◦	94,8	97,7	97,7	97,7	97,7	100,0 ◦
MUK	89,6	82,6 •	88,4 •	89,5	85,9 •	83,9 •	95,8	94,9 •	95,9	96,0	95,7	96,8 ◦	94,4	93,3 •	94,5	94,6	94,5	95,7 ◦
SIC	93,9	93,9	93,9	93,9	93,9	92,8 •	96,3	93,9 •	96,3	96,3	96,3	98,7 ◦	95,3	89,4 •	95,3	95,3	95,3	96,1 ◦
SPL	75,4	89,8 ◦	89,1 ◦	89,1 ◦	89,1 ◦	95,4 ◦	76,9	89,9 ◦	89,3 ◦	89,3 ◦	89,3 ◦	94,2 ◦	70,8	84,9 ◦	83,6 ◦	83,5 ◦	83,6 ◦	76,0 ◦
WAV	80,4	52,8 •	74,8 •	77,5 •	77,1 •	80,0 •	77,4	52,1 •	74,1 •	77,0	76,7	75,3 •	79,2	45,2 •	73,5 •	77,4 •	76,8 •	73,4 •
Pr.	79,10	76,82	79,41	75,78	80,25	85,99	82,53	81,74	83,38	80,20	83,87	90,63	77,85	77,14	81,02	76,86	81,18	87,73
G/P		6/2	4/4	4/3	4/5	5/7		8/3	4/4	5/3	4/3	1/11		5/3	3/3	5/3	3/3	2/10

SP–SOAP, *RL*–Relief, *IG*–Ganancia de Inform., *CH*–Chi2, *SU*–Incert. simétrica y Orig–datos originales. ◦, • mejora o empeora significativamente con respecto a *SP*.

Tabla 4.10: Comparativa del criterio *SP*. Atributos seleccionados y porcentaje retenido con respecto al original normalizando el ranking en bases de datos pequeñas de *UCI*.

Datos		*SP*		*RL*		*IG*		*CH*		*SU*	
	Ats.	Ats.	%	Ats.	%	Ats.	%	Ats.	%	Ats.	%
ADS	1558	12,43	0,80	2,74	0,18	3,33	0,21	5,72	0,37	3	0,19
ARR	279	32,49	11,65	1,01	0,36	6,22	2,23	4,29	1,54	5,33	1,91
HYP	29	1	3,45	3	10,34	1	3,45	1	3,45	1	3,45
ISO	617	159,35	25,83	36,8	5,96	30,57	4,95	18,32	2,97	32,42	5,25
KRV	36	3	8,33	3	8,33	1	2,78	1,07	2,97	1	2,78
LET	16	2,96	18,50	5,01	31,31	7	43,75	6	37,50	7	43,75
MUL	649	3,45	0,53	21,76	3,35	3	0,46	1,08	0,17	4,01	0,62
MUS	22	1,41	6,41	1	4,55	1	4,55	1	4,55	1	4,55
MUK	166	6,49	3,91	7,1	4,28	7,78	4,69	7,53	4,54	9,42	5,67
SIC	29	1	3,45	2	6,90	1	3,45	1	3,45	1	3,45
SPL	60	3	5,00	4,39	7,32	4	6,67	4,01	6,68	4	6,67
WAV	40	12,83	32,08	1	2,50	9,86	24,65	11	27,50	10,62	26,55
Promedios			9,99		7,12		8,49		7,97		8,74

de clasificación puede variar sustancialmente dependiendo del número de atributos como se ha visto en las comparaciones anteriores y se verá mas adelante.

La utilidad de los rankings de atributos, no siempre va orientada a la selección de un número de ellos para aplicar un algoritmo de aprendizaje al conjunto de datos reducido. A veces, nos interesa saber qué características son importantes y cuales no, y si existe un orden en la relevancia de ellas. Esto dificulta la comparativa entre los rankings. En el análisis de microarrays, se utilizan estos algoritmos para descubrir qué fármacos son los más eficaces. En [65], se identifican los cincuenta genes más correlacionados con la clase, para clasificar pacientes con leucemia. Algunos algoritmos de selección de subconjuntos de atributos más complejos, suelen generar rankings en sus fases iniciales.

Antes de comparar rankings de atributos se tendría que aclarar cuándo se considera que un ranking es bueno. En general, un ranking será bueno cuando la posición que ocupa cada atributo en la lista se corresponda con su relevancia, existiendo distintos puntos de vista sobre que entender por relevancia (consultar apartado 3.3, página 33). En nuestro caso, teniendo en cuenta la aplicación que se le va a dar, selección de atributos para clasificación, un buen ranking será aquel que coloca en las primeras posiciones los mejores atributos para clasificar el conjunto de datos.

De la anterior definición, se deduce que si dos atributos tienen igual importancia, ocupan posiciones contiguas en el ranking, aún conteniendo la misma información (redundantes). Ade-

más, si los dos atributos son muy relevantes entrarían a formar parte del subconjunto final. Esta situación, que es normal en la salida de los algoritmos de ranking, es incompatible con aquella a la que se pretende llegar en cualquier proceso general de selección, donde se intenta obtener aquel subconjunto mínimo de atributos que sean relevantes para la clasificación, dejando fuera los atributos redundantes e irrelevantes, por el efecto negativo generado en el algoritmo de aprendizaje aplicado con posterioridad. Para evitar la inclusión de atributos redundantes en el subconjunto generado por un algoritmo de ranking, es necesario una fase posterior de refinamiento.

En todos los casos mencionados anteriormente, la medida utilizada para cuantificar el comportamiento de un ranking es la exactitud obtenida por un clasificador con los k primeros atributos de la lista, diferenciándose en el modo de fijar el umbral. Esto nos lleva a plantear las siguientes cuestiones: ¿Qué se está evaluando exactamente, el ranking o el método para seleccionar los atributos? ¿Es correcto?

El valor que se utiliza en la comparación depende de tres factores: ranking generado, método de fijar el umbral y algoritmo de aprendizaje. Es un hecho que la exactitud de los modelos de clasificación puede variar sustancialmente según los atributos que intervengan; por consiguiente, la forma de elegir los atributos influirá considerablemente en el resultado final. Obviamente, se está evaluando ranking y método de selección, pero parece que cobra más importancia la forma de elegir los atributos que el orden en el que se encuentran. Por lo tanto, se puede afirmar que las comparaciones serán justas, pero no completas. Nuestra propuesta es evaluar el ranking directamente, sin esa dependencia con el método de selección.

Motivación

En primer lugar, se observa la calidad de los cuatro métodos de ranking de atributos con respecto a los tres clasificadores, utilizando la base de datos *Glass2* (214 ejemplos, 9 atributos, 2 clases) como caso representativo motivador de nuestro análisis. Para ello, en el apartado anterior, se comentó que el poder predictivo de un subconjunto de variables se puede medir en términos de tasa de error o de aciertos de algún clasificador. Sin embargo, en este caso, se quiere evaluar una lista de atributos ordenada, no un subconjunto en particular.

La tabla 4.11 muestra los ranking para *chi*2 (CH), *Ganancia de información* (IG), *Relief* (RL) y *SOAP* (SP). Para cada método de ranking, la fila *rk* presenta el ranking de atributos generado con uno de los métodos, y bajo esta fila, el resultado de clasificación con *C4.5* (C4), *Naïve Bayes* (NB) y la técnica del *vecino más cercano* (NN) utilizando el número de atributos del ranking indicado en la primera fila, bajo el título «Subconjunto». Los valores en **negrita** son los mejores resultados logrados con cada una de las medidas de evaluación para cada

Tabla 4.11: Ranking de Atributos para *Glass2*. FR: método de Feature–Ranking (CH: *chi*2; IG: Information Gain; RL: Relief; SP: Soap); Cl: Clasificador (C4: C4.5; NB: Naïve Bayes; NN: 1–Nearest Neighbour); y rk: ranking de atributos.

		Subconjunto								
FR	Cl	1	2	3	4	5	6	7	8	9
CH	rk	7	1	4	6	3	2	9	8	5
	C4	**73,6**	77,9	**82,2**	78,5	75,5	74,8	73,6	76,1	75,5
	NB	57,1	57,1	66,9	69,9	63,8	63,8	63,2	62,0	62,0
	NN	66,9	**79,7**	75,5	**82,8**	**88,3**	**81,0**	**77,9**	**77,9**	**77,3**
IG	rk	7	1	4	3	6	2	9	8	5
	C4	**73,6**	77,9	**82,2**	82,2	75,5	74,8	73,6	76,1	75,5
	NB	57,1	57,1	66,9	63,8	63,8	63,8	63,2	62,0	62,0
	NN	66,9	**79,7**	75,5	**84,7**	**88,3**	**81,0**	**77,9**	**77,9**	**77,3**
RL	rk	3	6	4	7	1	5	2	8	9
	C4	57,7	67,5	80,4	76,7	75,5	75,5	74,8	77,9	75,5
	NB	**62,0**	62,6	65,0	64,4	63,8	63,8	63,8	62,6	62,0
	NN	58,9	**75,5**	**81,0**	**83,4**	**88,3**	**83,4**	**81,6**	**81,6**	**77,3**
SP	rk	1	7	4	5	2	3	6	9	8
	C4	**77,3**	77,9	**82,8**	**81,6**	**81,6**	**84,1**	74,9	73,0	75,5
	NB	52,2	57,1	66,9	65,6	62,6	63,2	63,8	62,0	62,0
	NN	72,4	**79,7**	75,5	79,8	80,4	82,2	**81,6**	**77,3**	**77,3**

Los valores en **negrita** son los mejores resultados logrados con cada medida de evaluación para cada subconjunto,

subconjunto. Es interesante observar la diferencia existente en los resultados de los distintos clasificadores con el mismo conjunto de datos (con el conjunto de datos completo y mediante validación cruzada dejando uno fuera, se obtuvo un porcentaje de clasificación de: 75.5, 62.0 y 77.3; respectivamente).

Se puede observar que el atributo más relevante para CH e IG fue 7, para RL 3 y para SP 1. Utilizando sólo el atributo 7 (CH e IG), C4 obtiene una tasa de aciertos de 73.6, siendo 57.7 en el caso de utilizar sólo el atributo 3 (RL) y 77,3 en el caso del 1 (SP). El segundo atributo seleccionado por CH e IG fue el 1, el 6 por RL y el 7 por SP, siendo el mismo subconjunto (y por tanto el mismo resultado obtenido) para CH, IG y SP. Al ser escogido el atributo 4 en los cuatro casos como tercer atributo relevante, se mantiene el mismo subconjunto para CH, IG y SP, pero a partir de este punto, SP ya no coincidirá con los otros dos, que si mantendrán el mismo ranking a partir del sexto atributo seleccionado.

Analizando la tabla 4.11 se puede inferir varias conclusiones interesantes: (a) Los cuatro métodos de ranking proporcionan diferentes listas de atributos, que obviamente conducen a

diferentes comportamientos en el algoritmo de aprendizaje. (b) El par SP con C4 es el único que obtiene una tasa de aciertos (77,3) utilizando sólo un atributo (atributo 1) que mejora a la tasa obtenida con el conjunto completo de atributos (75,5). (c) En principio, las secuencias de atributos que mejores resultados de clasificación obtienen son arbitrarias, es decir, no siguen un orden predecible que indique cual será el mejor subconjunto cuando se añada o se elimine un atributo. Observando la tabla, los mejores resultados de clasificación, independientemente del clasificador utilizado, empezando con los subconjuntos con un atributo, y añadiendo atributos ordenados del ranking uno a uno hasta tener el conjunto de datos completo, serían: (SP con C4, 77.3), ({CH ó IG ó SP} con NN, 79.7), (SP con C4, 82.8), (IG con NN, 84.7), ({CH ó IG ó RL} con NN, 88.3), (SP con C4, 84.1), ({RL ó SP} con NN, 81.6), (RL con NN, 81.6) y el último mejor valor 77,3 con NN para la base de datos completa. (d) Parece que NN ofrece mejores resultados que el resto de clasificadores. Un hecho destacable puede ser que el mejor resultado en clasificación con cinco atributos se ofrece con los rankings generados a partir de las medidas CH, IG y RL, con el clasificador NN conteniendo los atributos 1,3,4,6,7, sin embargo, el mejor resultado con seis atributos se obtenga con la medida SP y aplicando el clasificador C4 al subconjunto de atributos 1,2,3,4,5,7. La diferencia entre los rankings radica en que el atributo 6 está en el primero y no en el segundo, y los atributos 2 y 5, están en el segundo ranking y no en el primero. El atributo 6 es menos relevante cuando los atributos 5 y 2 ya están presentes en el subconjunto, es más, cuando se añade el atributo 6 al segundo ranking anterior, se empeoran los resultados. En general, la existencia de relaciones entre atributos, puede hacer que atributos que por si mismo no sean relevantes, proporcionen una mejora significativa en el comportamiento del modelo cuando se toma junto a otros atributos.

La figura 4.8 es ilustrativa de las situaciones en las que se puede encontrar al comparar diferentes rankings para un conjunto de datos. La pregunta que se plantea es ¿Qué ranking es mejor para clasificar? La respuesta estaría condicionada a lo que el usuario esté buscando. Es decir, si lo que interesa es identificar el método de ranking que obtenga el subconjunto con mejor clasificación para un algoritmo de aprendizaje dado, se escogería *Metodo1* sabiendo que para ello necesita el ochenta por ciento de los atributos. Sin embargo, se observa que en el resto de la curva, los resultados de clasificación casi siempre están por debajo de los otros dos métodos. Si se escoge un número de atributos inferior al setenta por ciento, el resultado del *Metodo1* será el peor de los tres. Si lo que se busca es el método con un comportamiento mejor a lo largo de toda la curva, se compara punto a punto la evolución de las tres curvas. El *Metodo2* pierde al principio (hasta el treinta por ciento de los atributos) frente al *Metodo3*, posteriormente siempre mejora a los otros dos, salvo puntualmente en el caso comentado anteriormente (con el ochenta por ciento de los atributos). Y por último, si se quiere seleccionar menos del treinta por ciento

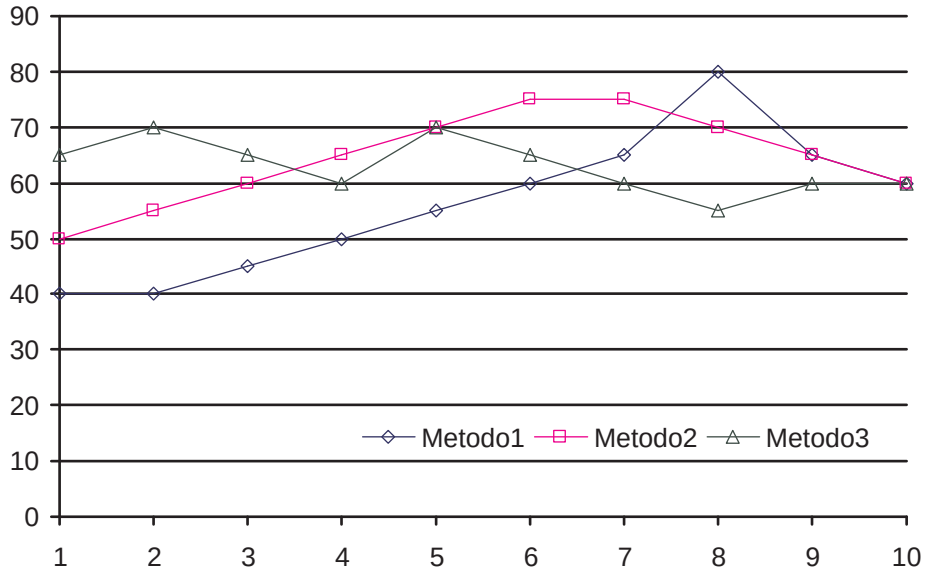


Figura 4.8: Ejemplo ficticio de tres tipos diferentes de curvas de aprendizaje. En el eje de abscisas el nº de atributos utilizados en la clasificación y en el de ordenada la tasa de aciertos.

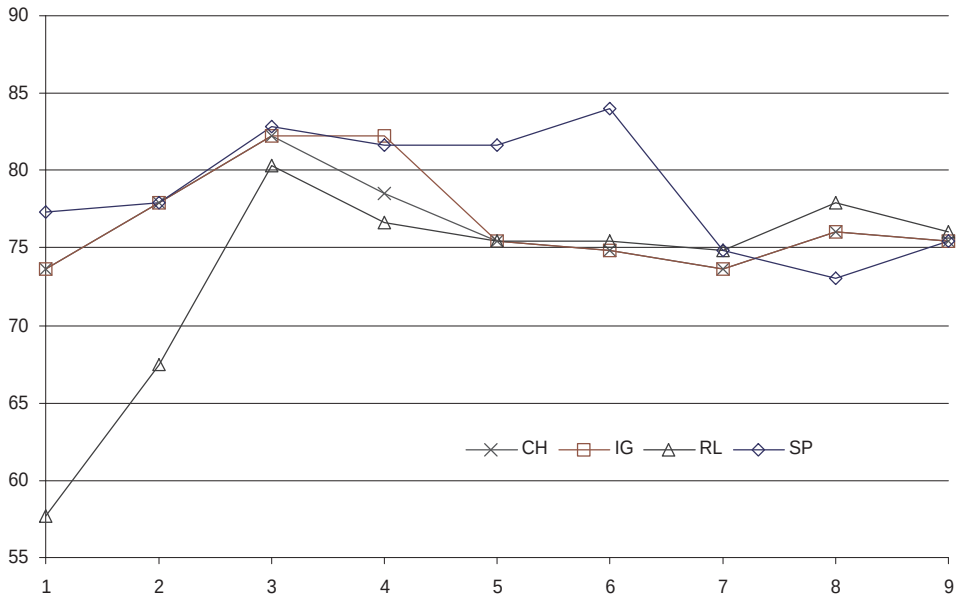


Figura 4.9: Curvas de aprendizaje obtenidas aplicando el clasificador C4 a diferentes rankings para la base de datos *Glass2* (datos de la tabla 4.11). En el eje de abscisas el nº de atributos utilizados en la clasificación y en el de ordenada la tasa de aciertos.

de los atributos, el mejor es el *Metodo3*.

La figura 4.9 muestra la exactitud en la clasificación de C4 con los cuatro métodos de ranking de atributos para la base de datos *Glass2*. Aunque la exactitud del mejor subconjunto es parecida, el comportamiento de SP es excelente para cualquier número de atributos y es el único que en casi todos los subconjuntos aparece por encima de la media. Por lo tanto se podría afirmar que es mejor ranking que los demás para esta base de datos.

El análisis basado en el mejor subconjunto, no refleja exactamente la bondad del ranking de atributos, puesto que antes o después de ese subconjunto los resultados podrían ser pésimos. La figura 4.10 muestra un comportamiento en la clasificación con C4 muy diferente para los rankings de los distintos métodos, sin embargo, si se extrae el resultado del mejor subconjunto de cada uno, se llega a la conclusión de que son rankings muy parecidos (CH:80,77; IG:80,29; RL:80,77 y SP:81,25).

En la figura 4.11 se observa las curvas generadas con los cuatro métodos de ranking para la base de datos *Segment*, utilizando el clasificado NB. Si se evalúan los rankings por el mejor subconjunto de cada método, no se podría comparar al coincidir los resultados en las cuatro listas de atributos, como se observa, se obtiene la misma exactitud de clasificación, y los subconjuntos contienen el mismo número de atributos. Sin embargo, la evolución de las curvas es diferente. Se observa que el comportamiento de los métodos CH e IG es inferior en casi todos los subconjuntos a RL y SP.

Teniendo en cuenta estas conclusiones, se quiere considerar la posibilidad de llegar a comprender mejor cuando un ranking de atributos es mejor que otros para un clasificador dado. Además, sería interesante analizar el comportamiento del método de ranking a lo largo de la curva de aprendizaje descrita, y sacar conclusiones en función de la proporción de atributos utilizada.

Metodología propuesta

Una comparación más completa entre dos rankings de atributos sería comparar subconjunto a subconjunto, es decir, comparar los resultados de clasificación obtenidos con el primer (mejor) atributo de las dos listas, con los dos mejores, y así sucesivamente hasta los n atributos del ranking. Calculando el promedio de los resultados obtenidos con cada lista, se podría utilizar para comparar los rankings. Un estudio muy parecido sería el calcular el área bajo la curva que describen los resultados anteriores.

Antes de proseguir con el estudio de los rankings de atributos se darán algunas definiciones para describir formalmente los conceptos utilizados en esta sección, para ello, se parte de las definiciones enumeradas en la sección 2.2 (página 13). Se cuenta con las definiciones de

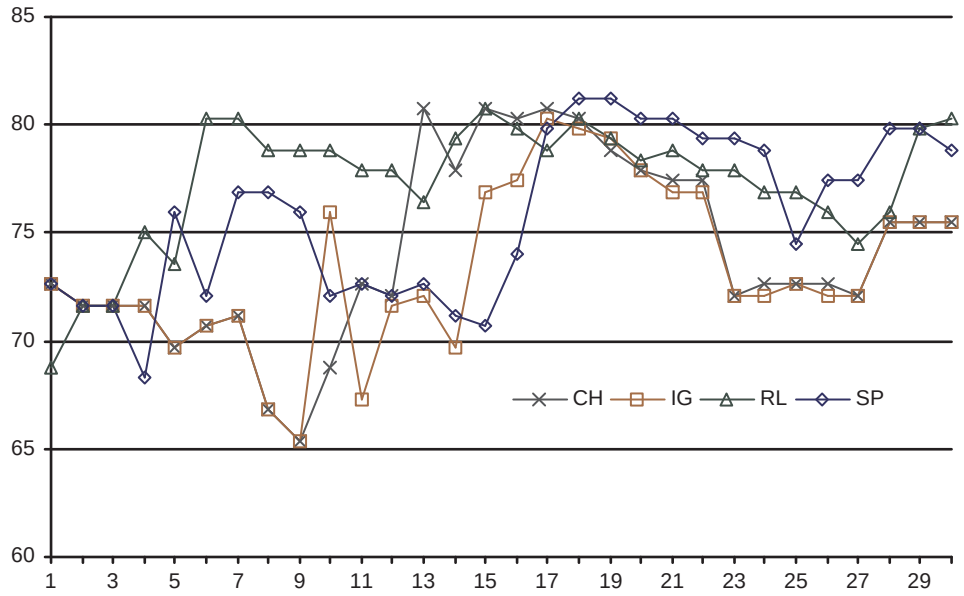


Figura 4.10: Curvas de aprendizaje obtenidas aplicando el clasificador C4 a diferentes rankings para la base de datos *Sonar*. En el eje de abscisas el nº de atributos utilizados en la clasificación y en el de ordenada la tasa de aciertos

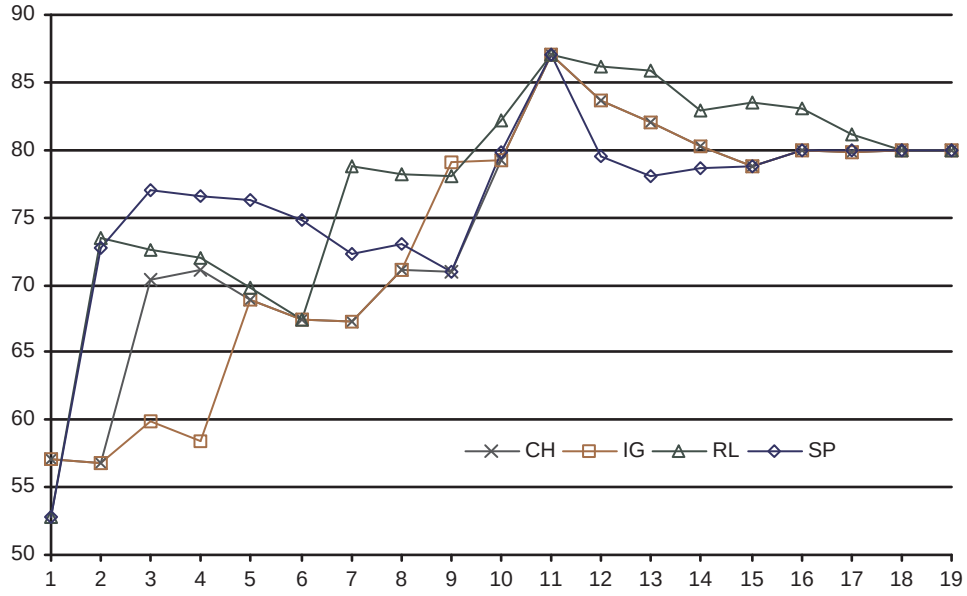


Figura 4.11: Curvas de aprendizaje obtenidas aplicando el clasificador NB a diferentes rankings para la base de datos *Segment*. En el eje de abscisas el nº de atributos utilizados en la clasificación y en el de ordenada la tasa de aciertos

clasificación 2.7 ($\mathcal{L}$) y de precisión 2.9 (Γ) dadas en el segundo capítulo (página 14 y 21 respectivamente). En el apartado 3.5.2 se definió el concepto de ranking de atributos (definición 3.11, página 61). Debe recordar, que un valor alto es indicativo de un atributo relevante y que los atributos están en orden decreciente según $r(a^*)$. Se considera definido el criterio de ordenación para atributos individuales, independientes del contexto de los demás, y nos limitaremos a criterios de aprendizaje supervisado.

Tener en cuenta que S_1^{F} es el primer (mejor) atributo del ranking elaborado por F; S_2^{F} son los dos primeros atributos, y así hasta n. Por tanto, la exactitud de la clasificación basada en ranking de $\mathcal{L}$, seleccionando los k primeros atributos proporcionados por el método de ranking F, será la siguiente:

$$\Gamma_k(\mathrm{F}, \mathcal{L}) = \Gamma(\mathrm{E}/S_k^{\mathrm{F}}, \mathcal{L})$$

El área que queda bajo una curva (*AUC-Area Under the Curve*) se calcula aplicando la fórmula del trapecio. En este caso se hace referencia a la curva de aprendizaje obtenida al ir añadiendo atributos según el orden asignado por el método de ranking, pudiéndose cuantificar su evolución y así comparar los resultados de los distintos rankings según su área.

$$\sum_{i=1}^{n-1} (x_{i+1} - x_i) * \frac{(y_{i+1} + y_i)}{2}$$

teniendo en cuenta que x e y son las coordenadas de un punto de la curva, donde x en el eje de de abscisas representa cada subconjunto de atributos del ranking e y el resultado de clasificación.

Definición 4.11 (AURC) *Dado un método de ranking de atributos* F *y un clasificador $\mathcal{L}$, se puede obtener el comportamiento del método de clasificación con respecto a la lista de atributos suministrada por el método de ranking, por medio del valor del área bajo la curva $AURC(\mathrm{F}, \mathcal{L})$. La curva se dibuja al unir cada dos puntos $(\Gamma_k(\mathrm{F}, \mathcal{L}), \Gamma_{k+1}(\mathrm{F}, \mathcal{L}))$, donde $k \in \{1, \ldots, n-1\}$ y n es el número de atributos. El área bajo la curva de comportamiento de clasificación basado en ranking (AURC–The Area Under Ranking Classification performance Curve) $AURC(\mathrm{F}, \mathcal{L})$ se calculará:*

$$AURC(\mathrm{F}, \mathcal{L}) = \frac{1}{2(n-1)} \sum_{i=1}^{n-1} (\Gamma_i(\mathrm{F}, \mathcal{L}) + \Gamma_{i+1}(\mathrm{F}, \mathcal{L}))$$

Con esta expresión, para cada par $(\mathrm{F}, \mathcal{L})$, el área bajo la curva $AURC(\mathrm{F}, \mathcal{L}) \in [0, 1]$ (en las tablas aparece multiplicado por cien para mayor claridad), lo que nos proporciona un buen método para comparar la calidad de los rankings de atributos con respecto a los algoritmos de clasificación. Una propiedad interesante observada en la curva, es que no es monótona creciente, es decir, para algunos i, sería posible que $\Gamma_i(\mathrm{F}, \mathcal{L}) > \Gamma_{i+1}(\mathrm{F}, \mathcal{L})$.

En la figura 4.11, tomando los valores de *AURC* como valor representativo de los ranking generados por los distintos métodos, CH:74.63, IG:73.79, RL:78.26 y SP:76.77, se comprueba que el resultado de la comparación es más fiel a la realidad que el obtenido mediante la comparación por el mejor subconjunto. Lo mismo ocurre en la figura 4.10, donde los valores de *AURC* son: CH:73.92, IG:73.71, RL:76.06 y SP:75.15, reflejando la diferencia de calidad entre ellos.

En la figura 4.11, donde el mejor *AURC* lo tiene el ranking elaborado por el método RL, se ve que el comportamiento de las dos mejores curvas (RL y SP) cambia a partir del séptimo atributo, siendo hasta ese instante SP el mejor método. Lo que nos lleva a realizar un estudio por porcentajes de atributos.

Definición 4.12 *El comportamiento de un ranking de atributos se mide como la evolución del AURC a lo largo del ranking de características, con paso δ %. Los puntos de la curva, para la que se calcula el AURC, se calculan para cada δ % de los atributos de la siguiente manera:*

$$AURC_{\delta}(\mathrm{F}, \mathcal{L}) = \frac{1}{2\delta(n-1)} \sum_{i=1}^{\delta(n-1)} (\Gamma_i(\mathrm{F}, \mathcal{L}) + \Gamma_{i+1}(\mathrm{F}, \mathcal{L}))$$

Continuando con el ejemplo de la curva de aprendizaje mostrada en la figura 4.11, si se comparan las medidas SP y RL mediante $AURC_{\delta}$ con varios valores para δ, se comprueba la evolución de los distintos rankings: el $AURC_{40}$ es favorable a la medida SP, con 73.30, siendo 70.17 para RL; el $AURC_{50}$ es casi idéntico en los dos casos, 74.34 y 74.23 para SP y RL respectivamente; mientras que con el total de los atributos, $AURC_{100}$, la evaluación es favorable a RL con 78.26, frente a 76.77 de SP.

Se debe tener en cuenta que la idea de todo método de selección de atributos, entre ellos los de ranking, es la de quedarse con el menor número de atributos posible, esto unido a la posibilidad de que dentro de la curva de aprendizaje se compensen exactitudes altas y bajas para el cálculo del *AURC*, nos lleva a realizar un estudio del comportamiento de los método con los primeros atributos y con porcentajes fijos.

Resultados experimentales

El proceso seguido en este estudio para medir la capacidad de un clasificador en predecir las nuevas instancias correctamente, es la validación cruzada *dejando uno fuera* o *leaving-one-out* (sección 2.4.2, página 22). Aunque este proceso sea computacionalmente muy costoso, produzca una varianza muy alta y los modelos generados están sobreajustados, se escoge esta validación cruzada por su ausencia de aleatoriedad (como ocurre en *k–fold cross–validation*), pues el objetivo de estudio no es la exactitud de los modelos, sino realizar una comparación

entre los rankings obtenidos por los distintos métodos y extraer reglas de comportamiento de las listas generadas en combinación con los clasificadores utilizados.

A cada base de datos se le aplica los cuatro métodos de ranking de atributos, y para cada ranking se calcula la curva de aprendizaje con cada uno de los tres clasificadores. Al aplicar validación cruzada *leaving-one-out* a cada subconjunto ordenado de atributos del ranking, la cantidad de modelos generados para cada selector-clasificador asciende a 286.009 (16 bd $\times$ n^{o}att's de cada una), como se tienen doce combinaciones selector–clasificador (4 selectores $\times$ 3 clasificadores), el total de modelos generados es de 3.432.108.

Área bajo la curva. La tabla 4.12 muestra, para cada conjunto de datos, el área bajo la curva de clasificación basada en ranking obtenida con cada combinación ranking-clasificador. Los valores en **negrita** son los mejores para los tres clasificadores, y los que aparecen subrayados son los mejores para el clasificador correspondiente. No se pueden extraer conclusiones claras, pero sí ciertas tendencias:

- Atendiendo a los promedios que aparecen en la última fila, los resultados son muy parecidos bajo cada clasificador, pero sí existen diferencias entre cada uno de ellos. El clasificador que ofrece mejor comportamiento con los cuatro métodos de ranking de atributos es NN, seguido muy de cerca por C4 y en último lugar NB con una diferencia significativa.

- Si se tiene en cuenta el mejor *AURC* para cada base de datos, se observa que en la mitad de las bases de datos, el mejor *AURC* se alcanza con el clasificador NN, lo que viene a reforzar la conclusión anterior.

- Por clasificadores, se comprueba que en los tres casos RL es el que más veces gana, por lo que se podría concluir que es mejor método de ranking.

Porciones de área bajo la curva. En la figura 4.12, siguiendo la Definición 4.12, se usa un paso de 5 % (δ=0.05). Aunque la evolución del *AURC* es más apropiada analizarla independientemente para cada base de datos, esta figura muestra el promedio del comportamiento del método de ranking de atributos para las dieciséis bases de datos. Se puede distinguir tres grupos de curvas, cada uno con cuatro curvas (una por método de ranking).

Al estudiar los resultados de clasificación incrementando el subconjunto de atributos en un porcentaje, en promedio, IG es el método de ranking de atributos que mejor comportamiento ofrece. Referente a los clasificadores, cuando se selecciona un número inferior al 15 % de los atributos de un ranking, C4 es el mejor clasificador. Sin embargo, por encima del 35 % 1–NN es

Tabla 4.12: Valor *AURC* para cada combinación ranking-clasificador. La última fila muestra el promedio de las AURCs de todas las bases de datos.

Datos	C4				NB				NN			
	CH	IG	RL	SP	CH	IG	RL	SP	CH	IG	RL	SP
ANN	97,30	97,12	96,90	97,11	85,82	86,30	86,50	86,47	**98,20**	98,09	97,54	97,71
BAL	68,61	68,61	68,83	68,61	**75,55**	**75,55**	72,56	**75,55**	72,77	72,77	69,79	72,77
CRG	72,39	72,39	71,71	72,31	**74,74**	**74,74**	73,89	74,22	70,16	70,16	70,38	66,83
DIA	72,85	72,89	73,30	72,52	75,36	**75,73**	75,68	75,15	68,87	69,52	68,09	67,87
GLA	64,57	66,09	67,09	67,32	49,15	49,85	47,34	51,37	63,49	67,67	68,17	**71,12**
GL2	76,65	77,11	74,35	79,03	63,27	62,50	63,50	62,27	79,41	79,64	**80,37**	78,91
HES	78,23	78,23	77,04	76,54	**83,09**	**83,09**	81,53	80,80	78,43	78,43	75,94	74,34
ION	88,69	89,18	**90,03**	86,94	84,52	85,11	85,66	80,23	88,57	88,36	88,57	86,92
IRI	95,11	95,11	95,22	95,22	95,56	95,56	95,56	95,56	95,00	95,00	95,56	95,56
KRV	95,22	95,13	**96,48**	95,56	87,47	87,47	89,81	86,99	93,97	93,89	95,79	94,63
LYM	74,84	75,97	75,66	75,42	78,76	80,25	80,37	80,09	76,61	81,28	**82,31**	80,56
SEG	92,23	92,15	93,37	92,96	74,63	73,79	78,26	76,77	92,78	93,11	**93,87**	93,26
SON	73,92	73,71	76,06	75,15	67,62	67,44	69,57	68,83	84,15	83,94	**84,41**	83,87
VEH	64,59	65,79	**68,10**	67,43	41,65	41,54	41,72	41,04	66,09	65,83	65,79	65,69
VOW	73,98	74,20	73,96	74,59	61,96	62,46	61,65	62,17	**90,67**	90,66	89,63	90,52
ZOO	88,18	87,56	86,88	88,27	88,95	88,21	86,42	89,36	**91,34**	90,84	87,69	90,87
Pr	79,83	80,08	80,31	80,31	74,25	74,35	74,37	74,18	81,91	**82,45**	82,12	81,96

CH–chi2, *IG*–Ganancia de Información, *RL–Relief* y *SP–SOAP*. Los valores en **negrita** son los mejores para los tres clasificadores, y los que aparecen subrayados son los mejores para el clasificador correspondiente.

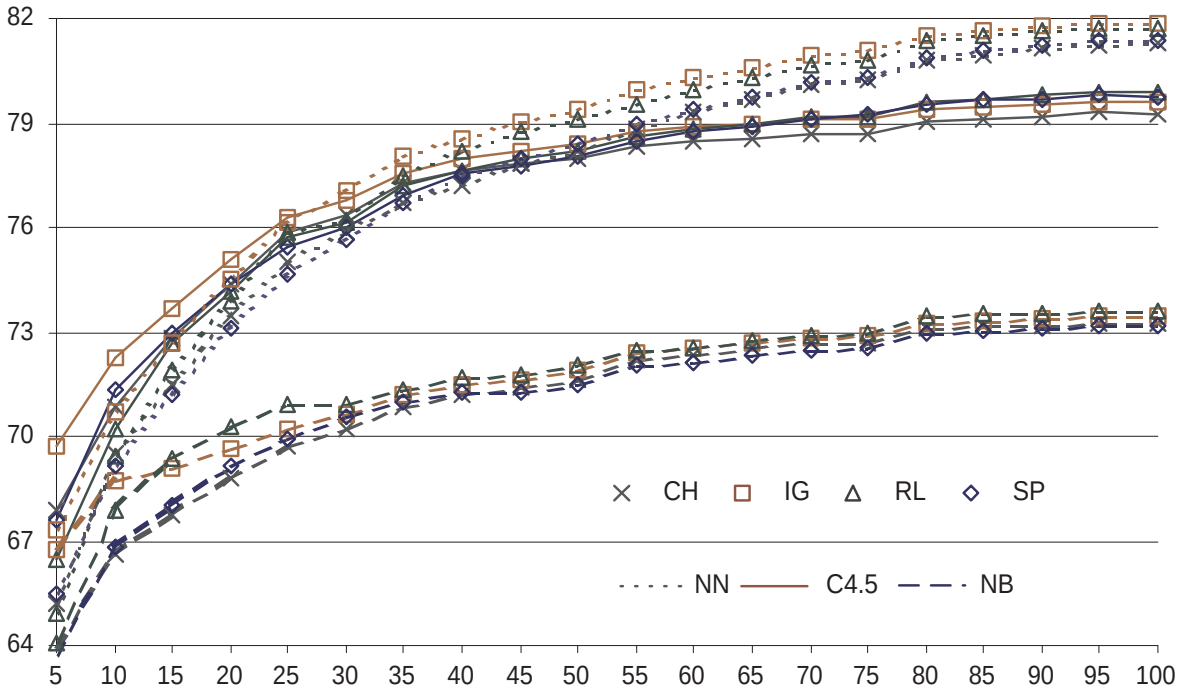


Figura 4.12: AURC utilizando porcentajes de atributos (con paso 5 %). En el eje de abscisas el porcentaje de atributos utilizados en la clasificación y en el de ordenadas el promedio de *AURC* para todas las bases de datos.

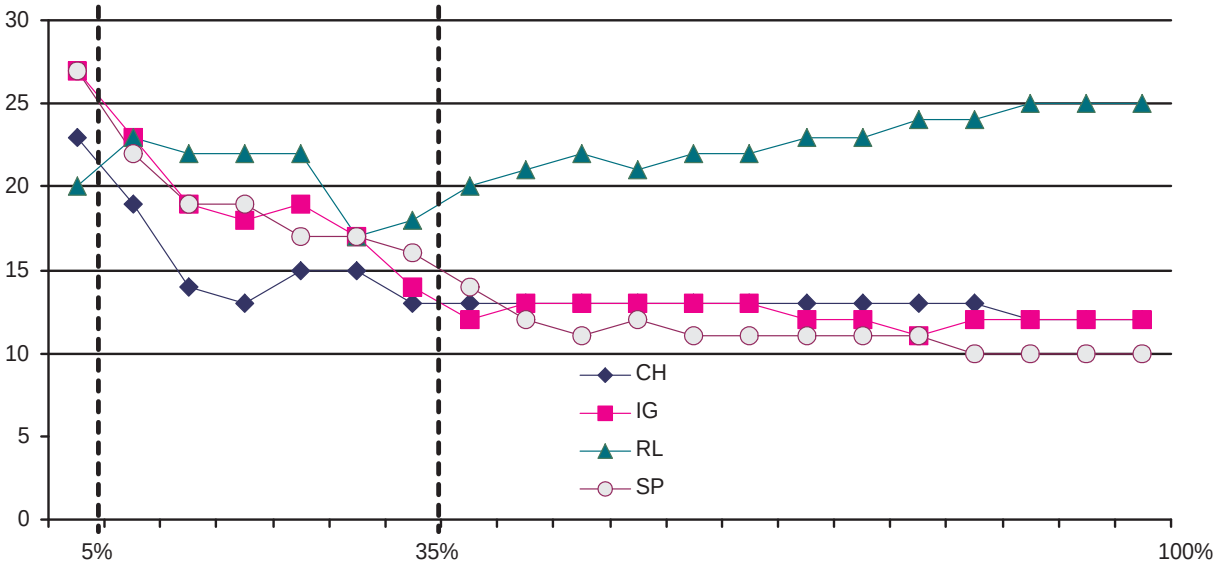


Figura 4.13: Relación entre el porcentaje de atributos seleccionado del ranking y la mejor clasificación obtenida. En el de ordenadas, el número de veces que un método de ranking fue mejor que los demás.

el mejor, especialmente con RL e IG (ver en la figura que las curvas de C4 interseccionan con las de NN aproximadamente en el 25 %). Excepto en muy pocos casos, especialmente cuando están involucrados muy pocos atributos relevantes, el clasificador NB no obtiene buenos resultados de comportamiento.

Si en vez de trabajar con los promedios de todas las bases de datos, donde se pueden solapar unos resultados buenos con otros, se hace contando el número de veces que un método es mejor que los demás, se obtiene la gráfica que aparece en la figura 4.13. Para menos del 5 % de los atributos, las mejores técnicas de ranking son IG y SP, las cuales mantienen una tendencia similar. También es obvio que RL es el más estable cuando el porcentaje de atributos está por encima del 35 % del total, quedando un intervalo entre ambas zonas (5 %–35 %) de cambios en las tendencias de las curvas.

Además, se quiere saber las exactitudes obtenidas con cada clasificador para los conjuntos de datos reducidos al primer atributo de las listas generadas por cada método de ranking. Además la *AURC* de las curvas obtenidas después de reducir los datos a dos, tres, cuatro y cinco atributos, puesto que se entiende que este será el objetivo de cualquier selector de atributos. Para terminar con este juego de comparaciones, se estudiará la *AURC* para los conjuntos obtenidos con el 25 %, 50 % y con el 100 % de los atributos de los rankings de cada método.

Se utilizan los resultados de clasificación con los algoritmos C4, NB y NN mediante validación cruzada *leaving-one-out*, después de reducir cada base de datos con los cuatro métodos

Tabla 4.13: Resumen de las veces que cada método de ranking ocupa la 1ª posición. Resultados agrupados por: los primeros atributos, porcentajes y clasificadores

Resultados por	CH	IG	RL	SP
Exactitud-1at:	26	28	20	**30**
AURC-2at:	18	22	16	**24**
AURC-3at:	15	17	15	**21**
AURC-4at:	14	15	16	**20**
AURC-5at:	15	**20**	18	17
AURC-25 %:	17	**22**	21	17
AURC-50 %at:	13	12	**21**	13
AURC-all at:	14	12	**25**	12
C4:	39	46	50	**51**
NB:	41	**59**	58	46
NN:	50	43	44	**55**
Total:	130	148	**152**	**152**

*CH–chi*2, *IG*–Ganancia de Información, *RL–Relief* y *SP–SOAP*. Se resalta en **negrita** los valores más altos.

de selección con las opciones antes descritas, se calcula el *AURC* según lo indicado en el párrafo anterior, disponiendo finalmente de mil quinientos treinta y seis valores (16bd. × 3cl. × 4ra. × 8 pruebas = 1536) para extraer conclusiones. Serán cuatro las posiciones posibles en cada comparación, interesándonos sólo la primera (en caso de empate cuenta para todos los implicados).

En la tabla 4.13 se dispone de un resumen de las veces que cada método de ranking ocupa la 1ª posición. Se establecen distintos grupos de comparaciones: En el primer bloque se encuentran los resultados obtenidos con los primeros atributos (con el primer atributo se compara la tasa de aciertos y el *AURC* con dos, tres, cuatro y cinco atributos); el segundo bloque muestra los resultados de las comparaciones por porcentajes (25, 50 y con todos los atributos); y en el último grupo se desglosa por clasificadores.

Si se atiende a las pruebas realizadas con los primeros atributos, destaca el método de ranking SP, sobre todo con los clasificadores C4 y NN, siendo IG el que mejor resultado ofrece con NB utilizando sólo los primeros atributos del ranking.

Al 25 % del ranking IG y RL obtienen mejores resultados (IG: 22, RL: 21 y CH, SP: 17). Parcialmente por clasificador, se mantiene esta posición con C4 y NB pero con NN la primera posición al 25 % es para *Relief*. Y a partir de aquí hasta el total del conjunto de atributos, RL es el que más veces ocupa la primera posición. Al 100 % del ranking, RL gana con diferencia 25 veces frente a CH y IG con 12 y SP con 10 e igualmente al 50 % de los atributos del ranking.

Los resultados se mantienen con estos porcentajes (50 y 100) para los tres clasificadores.

Si el estudio de las ocho pruebas se hace por clasificadores, no existen grandes diferencias. Se destaca con el clasificador C4 los métodos SP y RL, y con muy poca diferencia sobre IG. Con NB, los que ocupan los primeros puestos son IG y RL, mientras que con NN, es SP.

Los que más veces han quedado en primera posición en el conjunto de todas la pruebas (480) fueron SP y RF con 152, seguido de IG con 148 y CH con 130.

Con los resultados obtenidos en las tres pruebas anteriores (*AURC*, porcentajes de *AURC* y *AURC* con los primeros atributos de las listas ordenadas) se puede realizar las siguientes recomendaciones:

- El *AURC* da una idea más completa de la bondad del ranking que la exactitud obtenida con un subconjunto de atributos.
- La lista completa mejor valorada es la generada mediante el algoritmo *RL*, sin embargo, si se va a trabajar con los primeros atributos o con menos del 25 % de los atributos, los métodos *SP* e *IG* ofrecen mejores resultados en un tiempo más reducido.
- En general, los mejores resultados de clasificación se obtienen con NN, aunque cuando el número de atributos seleccionado es pequeño (inferior al 25 %) C4 se comporta mejor en los cuatro casos que los demás clasificadores.

Se concluye este apartado, indicando que el estudio del *AURC*, en sus distintas modalidades, es útil para investigar a fondo el comportamiento de clasificación de un ranking, teniendo en cuenta el orden existente en la lista de atributos generada para un conjunto de datos determinado. Es decir, para contar con las recomendaciones que nos pueda ofrecer este método de comparación, se debe realizar el estudio con cada uno de los rankings, y siempre depende de las necesidades del usuario. Además, el tiempo necesario para construir la curva de aprendizaje lo hace inviable en bases de datos medianas y grandes, así como para utilizarlo como ayuda rápida.

4.5. Conclusiones

Este capítulo presenta un criterio determinista para seleccionar atributos según su relevancia. Las principales ventajas de *SOAP* son la rapidez y la simplicidad en la evaluación individual de atributos. El método se utiliza en la fase de preprocesado para reducir de manera considerable el número de atributos. *SOAP* se basa en una nueva medida para la selección de atributos, *NLC*, calculada analizando las proyecciones de los elementos del conjunto de datos sobre cada

dimensión o atributo. La medida propuesta permite ordenar los atributos por orden de importancia en la determinación de la clase, todo ello sin la necesidad de realizar cálculos estadísticos que podrían ser muy costosos en tiempo (correlación, ganancia de información, etc.), siendo el tiempo necesario por *SOAP* el equivalente al método de ordenación más rápido $O(m \times n \times log\ n)$. Al mismo tiempo se mantienen las tasas de error, y se obtienen unos modelos de clasificación mucho más simples.

Generalizando, dados los datos mostrados en las tablas de resultados, se puede decir que no existen grandes diferencias entre los distintos algoritmos de ranking. El algoritmo *SOAP*, con la medida *NLC*, obtiene buenos rankings, estando entremedio sus resultados entre los obtenidos con los restantes métodos utilizados en las comparativas en cuanto a la precisión predictiva. La gran ventaja de *SOAP* es la rapidez, que se confirma en las cuatro comparativas analizadas (normalizando rankings y con los k primeros atributos, con bases de datos pequeñas y grandes), siendo la proporción de tiempo reducida muy parecida en los distintos casos.

Como resultado de esta parte de la investigación, se cumple con el primero de los objetivos de esta tesis, encontrar un medida para evaluar atributos que sea eficiente y eficaz. Los resultados obtenidos mediante la aplicación de *SOAP* a diversos grupos de bases de datos, fueron publicados en [140, 141, 144, 145, 146, 147, 148]. Dado que la intención de este trabajo de investigación es la aplicación de algoritmos de selección a datos de gran dimensionalidad, es previsible la necesidad de rapidez de los algoritmos para que puedan ser aplicables. El algoritmo de ranking propuesto, *SOAP*, puede ser un buen punto de partida para la construcción de un algoritmo más completo de selección de subconjuntos que se abordará en el siguiente capítulo.

Los trabajos tradicionales donde se realizan comparaciones de algoritmos de rankings de atributos, principalmente evalúan y comparan el método de selección de atributos, en vez del ranking. En este capítulo se presenta también una metodología para la evaluación de rankings, partiendo de la premisa de la no existencia de un único subconjunto ideal para todos los casos, y de que el mejor ranking va a depender de lo que el usuario esté buscando.

Se puede concluir que el área bajo la curva del ranking (*AURC*) refleja el comportamiento completo de la lista de atributos ordenada, indicando su poder predictivo en global. En base al análisis de la evolución del *AURC*, se ha recomendado el uso de los algoritmos *SOAP* (SP) y *ganancia de información* (IG) y del clasificador *C4.5* (C4) cuando se quiere clasificar con pocos atributos, y *Relief* (RL) y el clasificador *1–NN* (NN) en los demás casos. Los resultados obtenidos fueron publicados en [143].

Esta metodología proporciona recomendaciones después de realizar el estudio de la evolución del ranking, además demuestra que comparar algoritmos mediante un número de atributos de los primeros de un ranking, no es lo mismo que la comparación entre algoritmos que ofrecen

como salida un subconjunto. Por consiguiente, el estudio se traslada a encontrar la forma de seleccionar los atributos de un ranking y que ese subconjunto resuma las cualidades del ranking. Estas ideas se utilizan en el próximo capítulo en la construcción de algoritmos de selección de atributos orientados a bases de datos de gran dimensionalidad.

Capítulo 5

Búsqueda de subconjuntos en rankings de atributos

El enorme aumento de la dimensionalidad de una nueva generación de bases de datos hace que la tarea de encontrar un subconjunto óptimo de atributos sea de extrema dificultad. Aún escogiendo un criterio de evaluación de subconjuntos muy eficiente, las técnicas de selección tradicionales en aprendizaje automático no son aplicables, y mucho menos si la evaluación viene dada por un algoritmo de aprendizaje. En este capítulo, se proponen dos nuevas técnicas de selección de atributos que permiten utilizar cualquier evaluador de subconjuntos –incluido el modelo de evaluación *wrapper*– para buscar un grupo de atributos que permita distinguir entre las diferentes clases posibles. Los métodos, denominados *BIRS* (*Best Incremental Ranked Subset*) y *BARS* (*Best Agglomerative Ranked Subset*), se basan en la idea de relevancia y redundancia, en el sentido de que un atributo (o conjunto) ordenado se escoge si añade información al incluirlo en el subconjunto final de atributos elegidos. Estas heurísticas conducen a mejoras considerables en la exactitud obtenida, en comparación con el conjunto completo y frente a otros algoritmos de selección de atributos, junto a una notable reducción en la dimensionalidad.

El capítulo se organiza de la siguiente forma: tras una introducción a las particularidades de la selección de subconjuntos de atributos en bases de datos grandes, se revisan los conceptos generales de relevancia y redundancia. En la sección 5.3 se presenta nuestra propuesta de relevancia y redundancia, describiendo los algoritmos *BIRS* y *BARS* en las secciones 5.4 y 5.5, respectivamente, así como los resultados obtenidos con cada uno de ellos. Finalmente, en la sección 5.6 se recogen las conclusiones más interesantes.

5.1. Introduction

Como se describió en el capítulo 3, la mayoría de los algoritmos de selección de atributos enfocan esta tarea como un problema de búsqueda (ver apartado 3.4, página 37), donde cada uno de los estados en la búsqueda se corresponde con un subconjunto distinto de atributos [20]. El proceso de búsqueda se combina con un criterio para evaluar el mérito de cada uno de los subconjuntos de atributos candidatos. Existen numerosas combinaciones posibles entre las diferentes técnicas de búsqueda y cada una de las medidas de atributos [37, 103, 112]. Sin embargo, los métodos de búsqueda pueden ser demasiado costosos en bases de datos con dimensión alta, especialmente si se aplica un algoritmo de aprendizaje como criterio de evaluación.

En el apartado 3.5.2 (página 60) se mostró que los algoritmos de selección de atributos se pueden agrupar de dos maneras desde el punto de vista de la medida de evaluación escogida: dependiendo del modelo seguido (*filtro o wrapper*) o de la forma de evaluar los atributos (*individual o subconjuntos*). El modelo filtro evalúa los atributos de acuerdo con heurísticas basadas en características generales de los datos e independientes del método de clasificación a aplicar, mientras que el *wrapper* utiliza el comportamiento de un algoritmo de clasificación como criterio de evaluación de los atributos. El modelo *wrapper* escoge los atributos que demuestran mejor clasificación, ayudando a mejorar el comportamiento del algoritmo de aprendizaje. En contra tiene un coste computacional más elevado [98, 89] que el modelo filtro. Los métodos de ranking de atributos ($\mathcal{FR}$–*Feature Ranking*), evalúan los atributos individualmente, mientras que los algoritmos de selección de subconjunto de atributos ($\mathcal{FSS}$–*Feature Subset Selection*) evalúan la bondad de cada uno de los subconjuntos candidatos.

En la categoría de algoritmos $\mathcal{FR}$, los k primeros atributos formarán el subconjunto final. Ésta es una buena aproximación para bases de datos de dimensionalidad alta, dado su coste lineal con respecto al número de atributos. En algoritmos capaces de seleccionar subconjuntos de atributos, los subconjuntos candidatos se generan según alguna estrategia de búsqueda, existiendo diversas posibilidades. En la tabla 3.1 se hayan clasificados numerosos algoritmos de selección. Se pueden encontrar diferentes estrategias de búsquedas, exhaustivas, heurísticas y aleatorias, combinadas con distintos tipos de medidas para formar un gran número de algoritmos. La complejidad temporal es exponencial con respecto a la dimensionalidad de los datos en la búsqueda exhaustiva y cuadrática en la búsqueda heurística. En la búsqueda aleatoria, la complejidad puede ser lineal al número de iteraciones [108], pero la experiencia muestra que el número de iteraciones necesarias para encontrar un subconjunto óptimo es al menos cuadrático con respecto al número de atributos [39]. Los métodos de búsqueda más populares en aprendizaje supervisado no se pueden aplicar a este tipo de bases de datos debido al elevado número de atributos. Una de las pocas técnicas de búsqueda utilizadas en este dominio es la búsqueda

secuencial hacia adelante [184, 80, 176] (también denominada *hill-climbing* o *greedy*).

5.2. Trabajos relacionados

Las limitaciones de ambas aproximaciones, $\mathcal{FR}$ y $\mathcal{FSS}$, sugieren claramente la necesidad de un modelo híbrido. Recientemente, se utiliza un nuevo marco de trabajo para la selección de atributos, donde se combinan varias de las técnicas antes mencionadas, junto con los conceptos de relevancia y redundancia anteriormente estudiados (ver sección 3.3 en la página 33). Debido al gran número de atributos, el proceso de selección se descompone en dos fases: se empieza con una fase donde se evalúan los atributos individualmente, proporcionando un ranking ordenado según algún criterio filtro. En la segunda fase, se aplica un evaluador de subconjuntos de atributos (filtro o *wrapper*) a un número determinado de atributos del ranking anterior (los que superan un umbral, o los k primeros) siguiendo una estrategia de búsqueda. Xing et al. [175], Yu y Liu [184], Ding y Peng [47] y Guyon et al. [69] están entre los trabajos más referenciados que siguen este entorno de trabajo (ver descripción de los métodos en el apartado 3.5.1, página 46).

Los modelos híbridos propuestos en este capítulo intentan aunar las ventajas de las diferentes aproximaciones en dos pasos: primero, se genera una lista de atributos ordenada siguiendo un modelo filtro o *wrapper*, y segundo, los atributos ordenados se van seleccionando utilizando una evaluación de subconjuntos, asegurándose un buen comportamiento. Se resalta la validez de estos métodos de búsquedas para cualquier lista de atributos ordenada. Estas vías para buscar subconjuntos ofrecen la posibilidad de aplicar eficientemente un modelo *wrapper* en dominios de alta dimensión, mejorando los resultados obtenidos con el modelo filtro.

Este capítulo tiene como objetivo el estudio y propuesta de dos métodos de selección de atributos que se puedan aplicar a bases de datos de dimensión elevada en un marco de aprendizaje supervisado, en concreto para clasificación. Se utilizarán tres algoritmos de aprendizaje clasificadores para comparar los efectos de la selección de atributos, uno probabilístico (*Naïve Bayes*), otro basado en las técnicas de vecinos más cercanos (*ib1*) y un tercero basado en árboles de decisión (*C4.5*) (ver sección 2.3 en la página 14).

5.3. Búsqueda secuencial sobre un ranking

En esta sección se aportan nuevas nociones de relevancia y redundancia, teniendo en cuenta el objetivo fijado de permitir aplicar la utilización de cualquier tipo de criterio tanto para elaborar el ranking como para generar el subconjunto de atributos.

Como se ha señalado antes, se utilizará un evaluador de subconjunto de atributos, que denominaremos *SubEvaluador*, para seleccionar un grupo reducido de ellos. Así, dado un *SubEvaluador*, y dado un conjunto de atributos $\mathcal{X}$, se busca en el espacio de $\mathcal{X}$ el subconjunto de atributos que mejor resultado de evaluación presente, utilizando el valor para comparar el comportamiento del *SubEvaluador* sobre el subconjunto de prueba. El modelo *wrapper* es computacionalmente más costoso, sin embargo tiende a encontrar conjuntos de atributos mejor ajustados al algoritmo de aprendizaje destino, ofreciendo un comportamiento superior al modelo filtro. Entre las medidas más utilizadas en la evaluación de subconjuntos, ya comentadas en el apartado 3.4.3 (página 43), nos encontramos con la correlación (subconjuntos correlacionados con la clase y no correlacionados entre ellos), consistencia, divergencia y el resultado de aplicar un algoritmo de aprendizaje (*wrapper*).

La forma de realizar la búsqueda en el espacio formado por el conjunto de los atributos es un factor clave en los métodos *wrappers*. Entre otras, aparece junto a estrategias de búsquedas tales como búsqueda secuencial *greedy*, búsqueda *best-first*, y *algoritmos genéticos* [103], la mayoría de ellos tienen una complejidad temporal $O(n^2)$, no pudiéndose aplicar en bases de datos con decenas de miles de atributos. En algunos casos, no se puede prever el número de veces que se va a ejecutar el clasificador, pudiendo llegar a ser el tiempo requerido del orden de cientos o miles de horas, suponiendo que el método no caiga primero en algún mínimo local y se detenga prematuramente. Por ejemplo, en una base de datos con veinte mil atributos, suponiendo que el conjunto final esté compuesto por cincuenta atributos (0.0025 % del total), una búsqueda *greedy* realizaría aproximadamente un millon de comprobaciones con el algoritmo de aprendizaje (n veces para elegir el mejor atributo, siendo n el número total de atributos, $n-1$ veces para encontrar el siguiente mejor atributo junto al primero, de esta manera tendríamos aproximadamente $20000at. \times 50$ seleccionados), suponiendo un promedio de cuatro segundos por cada una de ellas, daría como resultado más de mil horas.

En este trabajo se proponen búsquedas rápidas sobre una parte mínima del espacio formado por el conjunto de los atributos. Comenzando por el primer atributo de una lista ordenada por algún criterio de evaluación, se va comprobando la aportación de los atributos al resultado del *SubEvaluador* añadiéndolos uno a uno en el primer método que comentaremos, y uniendo diferentes subconjuntos en el segundo. En el primer caso, el algoritmo de evaluación se ejecuta siempre tantas veces como atributos tenga la base de datos, teniendo en cuenta que normalmente el *SubEvaluador* se construirá con muy pocos atributos. En el segundo, no se puede prever el número de veces que se ejecutará, pero eso sí, seguro que con muy pocos atributos. Como ya ha sido comentado, el algoritmo de ranking de atributos utiliza una función de puntuación que relaciona los valores de cada atributo y la clase. Por convención, se asume que una puntuación

alta en un atributo es indicativa de una relevancia alta, y que los atributos se sitúan en orden decreciente de puntuación. Se considera definido el criterio de ranking para atributos individuales, indepedientemente del contexto de los demás.

Al realizar un ranking en bases de datos con muchos atributos, normalmente se tiene un elevado número de atributos con puntuaciones similares, y se critica la frecuente selección de atributos redundantes en el subconjunto final. Sin embargo, según Guyon y Elisseff [68], teniendo en cuenta atributos que son presumiblemente redundantes se puede reducir el ruido, y por consiguiente, obtener mejor separación entre las diferentes clases. Además, una correlación (en valor absoluto) muy alta entre variables no significa que no se complementen. En consecuencia, la idea de redundancia en este trabajo no se basa en medidas de correlación, sino en un criterio de evaluación de subconjuntos, pudiendo ser una aproximación filtro o *wrapper*, en el sentido de que un atributo (o conjunto) es seleccionado si se obtiene información adicional cuando se añade al subconjunto de atributos elegidos previamente.

5.4. Utilidad incremental ordenada

En la selección de subconjuntos de atributos, es un hecho que se perciban como innecesarios dos tipos de atributos: los que son irrelevantes al concepto destino y los que son redundantes con otros atributos dados. Partiendo de las definiciones 3.7 y 3.9, se define formalmente la *utilidad incremental ordenada* de manera que se identifiquen explícitamente los atributos relevantes y no se tengan en cuenta a los atributos redundantes, logrando así un aprendizaje más eficiente y efectivo.

Sea $\mathcal{E}$ un conjunto de datos etiquetados; sea $R = \{X_i\}$, $i = 1 \ldots n$ un ranking de todos los atributos de $\mathcal{E}$ ordenados decrecientemente (definición 3.11); sea $\mathcal{S}$ un subconjunto de atributos de los datos R; y sea $\Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L})$ el *valor de medición*, que según la definición 3.10 aplica el evaluador de subconjuntos $\mathcal{L}$ al subconjunto $\mathcal{S}$. Teniendo en cuenta la definición 3.8 de independencia condicional ya dada, entonces

Definición 5.1 *El atributo X_{i+1} en R es* incrementalmente útil *para $\mathcal{L}$ si no es condicionalmente independiente de la clase C dado $\mathcal{S}$, de manera que el valor de medición que $\mathcal{L}$ produce usando el conjunto de atributos $\{X_{i+1}\} \cup \mathcal{S}$ es significativamente mejor (indicada por $\succcurlyeq$) que el valor de medición obtenido utilizando sólo el conjunto de atributos $\mathcal{S}$.*

Por tanto, si $\Gamma(\mathcal{E}/\mathcal{S} \cup \{X_{i+1}\}, \mathcal{L}) \not\succcurlyeq \Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L})$, siendo X_{i+1} condicionalmente independiente de la clase C dado el subconjunto de atributos $\mathcal{S}$, se puede omitir X_{i+1} sin comprometer el resultado obtenido por el evaluador de subconjuntos. Para la definición anterior, como ya se ha

señalado, la expresión $\Gamma(\mathcal{E}/\mathcal{S}, \mathcal{L})$ engloba tanto al valor de precisión de un clasificador (*wrapper*) $\mathcal{L}$, como al valor de medición obtenido con una medida tipo filtro ($\Upsilon(\mathcal{E}/\mathcal{S}, Ł)$).

Una cuestión fundamental en la definición previa es cómo averiguar si el valor de medición de un conjunto es significativamente mejor ($\gtrdot$) que otro. Para ello, se distingue entre un *SubEvaluador* que siga el modelo filtro y otro que siga el modelo *wrapper*. En el primer caso, simplemente se comprueba si la mejora supera un umbral establecido. Sin embargo en el segundo caso, además se puede hacer uso de la técnica de evaluación denominada validación cruzada. Se aplica un test t–Student pareado para evaluar si la diferencia entre el conjunto candidato es estadísticamente significativa (al nivel 0.1) con respecto al mejor subconjunto anterior. Esta última definición nos permite seleccionar atributos del ranking, pero sólo aquéllos que mejoran la tasa de aciertos significativamente.

La búsqueda propuesta para el modelo *wrapper* puede utilizar un análisis estadístico aún cuando se es consciente de que el tamaño muestral (validación cruzada 5) no es significativo si fuera un análisis riguroso de la varianza entre poblaciones. Sin embargo, hay que tener en cuenta que es sólo una heurística basada en un criterio objetivo para obtener un criterio sobre qué es un aumento significativo de la exactitud. Por otro lado, el nivel de confianza se ha relajado del valor estándar de 0.05 a 0.1 porque con un tamaño muestral tan pequeño (5–CV) y el 0.05, el test difícilmente dará diferencias significativas de exactitud al aumentar el número de atributos. Evidentemente, aumentar el valor del nivel de confianza hace que el método de búsqueda seleccione más atributos y a la inversa, lo que puede proporcionar un mecanismo de ajuste al usuario.

Partiendo de la definición anterior se implementa el algoritmo en el siguiente apartado.

5.4.1. Algoritmo

Existen dos fases bien diferenciadas en el algoritmo 5.1. En primer lugar, los atributos se ordenan según alguna medida de evaluación (líneas 1–5). En segundo lugar, se tratará la lista de atributos una vez, recorriendo el ranking desde el principio hasta el último atributo ordenado (líneas 6–15). Se obtiene el valor de medición del evaluador de subconjuntos con el primer atributo de la lista (línea 10) y se marca como seleccionado (líneas 11–14). Se obtiene de nuevo el valor de medición del *SubEvaluador*, pero esta vez con el primer y segundo atributo. El segundo se marcará como seleccionado dependiendo de si el valor obtenido es significativamente mejor (línea 11). El siguiente paso es evaluar de nuevo con los atributos marcados y el siguiente de la lista, y así sucesivamente. Se repite el proceso hasta alcanzar el último atributo de la lista. Finalmente, el algoritmo devuelve el mejor subconjunto encontrado, y se puede afirmar que no contendrá atributos irrelevantes ni redundantes.

Algoritmo 5.1 *BIRS–Best Incremental Ranked Subset*.
Entrada: $\mathcal{E}$–conjunto de datos, $\mathcal{U}$–criterio ranking, $\mathcal{L}$–*SubEvaluador*.
Salida: *MejorSub*–subconjunto de atributos

```
1:  R ⇐ {}
2:  para i = 1 hasta n hacer
3:     Puntuacion ⇐ calcula(X_i, U, E)
4:     inserta X_i en R según Puntuacion
5:  fin para
6:  MejorEval ⇐ 0
7:  MejorSub ⇐ ∅
8:  para i = 1 hasta n hacer
9:     TempSub ⇐ MejorSub ∪ {X_i} (X_i ∈ R)
10:    TempEval ⇐ evaluar(TempSub, L)
11:    si TempEval ⩾ MejorEval entonces
12:       MejorSub ⇐ TempSub
13:       MejorEval ⇐ TempEval
14:    fin si
15: fin para
```

La eficiencia de la primera parte del algoritmo anterior es notoria, dado que sólo requiere el cálculo de n puntuaciones y su ordenación, mientras que en la segunda parte, la complejidad temporal depende del algoritmo de evaluación de subconjunto escogido, siendo, obviamente, más costoso en el caso de una aproximación *wrapper*. Se debe tener en cuenta que el *SubEvaluador* se ejecuta n (número total de atributos) veces con un número muy reducido de atributos, sólo los seleccionados. Por tanto, el tiempo de computación para la construcción del ranking es despreciable con respecto al proceso global y proporciona un buen punto de partida. De hecho, los resultados obtenidos a partir de un orden aleatorio de atributos (sin un ranking previo) presentaron tres inconvenientes: 1) inestabilidad, dado que la solución es no determinista; 2) mayor número de atributos seleccionados; y 3) mayor tiempo de computación al trabajar el algoritmo de clasificación con un mayor número de atributos desde las primeras iteraciones.

Considerar la situación ilustrada en la figura 5.1, un ejemplo del proceso de selección de atributos mediante *BIRS* utilizando el modelo *wrapper* como evaluador de subconjuntos. La primera línea muestra los atributos ordenados según alguna medida de evaluación. Se obtiene la exactitud de clasificación con el primer atributo de la lista (x_5-80 %). En el segundo paso, se ejecuta el clasificador con los dos primeros atributos del ranking (x_5,x_7-82 %), y se calcula el t–test pareado para determinar el grado de significación estadística de las diferencias entre las tasas de aciertos. Como es mayor que 0.1, x_7 no se selecciona. Lo mismo ocurre con los dos próximos subconjuntos (x_5,x_4-81 %, x_5,x_3-83 %), pero el atributo x_1 se añade, dado que la

Ranking	x_5	x_7	x_4	x_3	x_1	x_8	x_6	x_2	x_9

	Subconjunto	Ac	p-val	Ac–Mejor subconj.
1	x_5	80		**80** $\mathbf{x_5}$
2	$\mathbf{x_5}$,x_7	82		
3	$\mathbf{x_5}$,x_4	81		
4	$\mathbf{x_5}$,x_3	83		
5	$\mathbf{x_5}$,x_1	84	< **0.1**	**84** $\mathbf{x_5, x_1}$
6	$\mathbf{x_5, x_1}$,x_8	84		
7	$\mathbf{x_5, x_1}$,x_6	86		
8	$\mathbf{x_5, x_1}$,x_2	89	< **0.1**	**89** $\mathbf{x_5, x_1, x_2}$
9	$\mathbf{x_5, x_1, x_2}$,x_9	87		

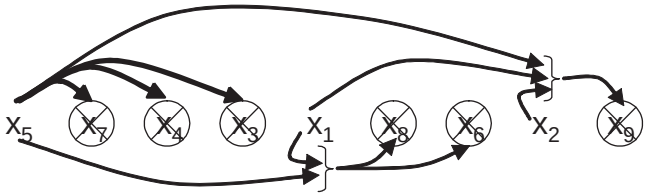


Figura 5.1: Ejemplo del proceso de selección de atributos mediante *BIRS*.

exactitud es significativamente mejor que la obtenida con sólo x_5 (x_5,x_1-84 %), y así sucesivamente. En resumen, el clasificador se ejecuta nueve veces para seleccionar, o no, los atributos ordenados (x_5,x_1,x_2-89 %): una vez con sólo un atributo, cuatro con dos atributos, tres con tres atributos y una vez con cuatro. La mayoría del tiempo, el algoritmo de aprendizaje se ejecuta con pocos atributos. De esta forma, esta aproximación *wrapper* necesita mucho menos tiempo que otras con una estrategia de búsqueda más amplia.

Como se puede observar en el algoritmo, el primer atributo es siempre seleccionado. Esto no debe representar un gran inconveniente en bases de datos de elevada dimensionalidad, basándonos en la probable existencia de diferentes conjuntos de atributos que contengan parecida información. Por consiguiente, *BIRS* se basa en la idea de que el mejor atributo de un ranking lleva a soluciones cercanas a la óptima. El principal inconveniente de hacer un recorrido siguiendo esta dirección (*hacia adelante*), es la posibilidad de no detectar interacciones básicas entre atributos que sean muy interesantes para la clasificación. Es decir, puede ocurrir que atributos que por separado son irrelevantes, el estar juntos en un determinado subconjunto les haga ser muy importantes. En el caso *hacia atrás*, se remedia en parte el inconveniente de la generación *hacia adelante*, aunque permanezcan interacciones ocultas, sin embargo necesita más tiempo de computación. Se debe tener en cuenta que dado el gran número de atributos de las bases de datos a las que se dirige este algoritmo, el coste computacional del *backward* es mucho mayor, y si se utiliza una aproximación *wrapper*, aún más.

5.4.2. Experimentos y resultados

En esta sección se pretende evaluar el método de búsqueda propuesto en términos de exactitud de clasificación, número de atributos seleccionados y eficiencia en la obtención del subconjunto final, para mostrar cómo *BIRS* se comporta en situaciones donde existen gran número de atributos. No obstante, la comparación se efectuó sobre los tres grupos de bases de datos cuyas propiedades se encuentran resumidas en las tablas A.1, A.2 y A.3. Se seleccionaron los tres algoritmos anteriores para evaluar la exactitud sobre los atributos seleccionados por cada uno de los algoritmos de selección de atributos. Los experimentos se llevaron a cabo en un Pentium IV a 3 Gh con 512 Mb de RAM, utilizando implementaciones de los algoritmos existente en el entorno WEKA [173], así como el método de búsqueda propuesto también fue implementado en dicho entorno.

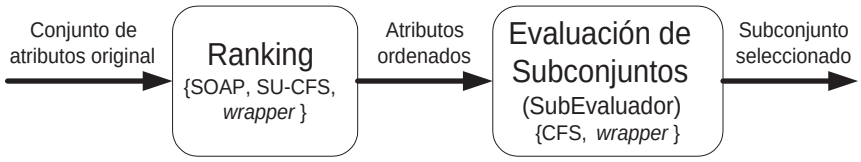


Figura 5.2: Partes del algoritmo de selección *BIRS*.

Considerar la figura 5.2, ilustrativa de los dos bloques que componen siempre el algoritmo *BIRS* propuesto en los apartados anteriores. El algoritmo de selección debe contar con una medida de evaluación que se pueda aplicar individualmente a los atributos para obtener un ranking, y además, en el proceso de búsqueda incremental se deberá emplear una medida capaz de evaluar subconjuntos de atributos (ver apartado 3.4.3, página 43). Se debe tener en cuenta que todas las medidas de evaluación de subconjuntos se pueden aplicar individualmente, pudiendo ser utilizadas para la elaboración de rankings de atributos. Se podrían formar numerosos algoritmos de selección combinando los criterios de cada grupo de medidas (individual y de subconjuntos), por tanto, siempre que aparezca el algoritmo *BIRS* en las tablas comparativas, se expresa con qué criterio se ha generado el ranking, y cómo se ha evaluado la calidad de los subconjuntos de atributos en el proceso de selección. Intentando esclarecer en todo lo posible los componentes que *BIRS* esté utilizando en cada caso, se antepone un subíndice a *BI–Incremental* que indica el método de ranking empleado y con un superíndice a continuación se señala el SubEvaluador, por ejemplo, ${}_{SP}BI^{CF}$ indica que se emplea SP como medida individual en la primera parte y *CFS* como evaluador de subconjuntos en la segunda parte, o con otras palabras, que tras elaborar un ranking con el criterio SP, la técnica *BIRS* realiza una búsqueda sobre la lista generada aplicando la medida *CFS* para evaluar los subconjuntos.

En el capítulo anterior, se pudo comprobar que no existían grandes diferencias entre los

criterios individuales de evaluación utilizados en las pruebas realizadas. Por consiguiente, en los experimentos realizados en este apartado, se emplea un grupo reducido de medidas de rankings: *SOAP* (SP), propuesta en el capítulo anterior, y en las tablas aparece ${}_{SP}BI$; incertidumbre simétrica (SU), que evalúa individualmente la correlación no lineal con la clase, y que coincide exactamente con la aplicación individual de la medida de subconjuntos *CFS* (*Correlation–based Feature Selection* [70], apartado 3.5.1, página 53), en las tablas se muestra ${}_{SU}BI$ ó ${}_{CF}BI$; además se puede utilizar como criterio de ranking el resultado de aplicar un clasificador a cada atributo, indicándose en las tablas con el nombre del clasificador como subíndice, por ejemplo, en ${}_{NB}BI$ se genera el ranking con NB. Las dos primeras medidas se consideran algoritmos de tipo filtro y la última *wrapper*. En el caso *wrapper*, se utiliza el mismo clasificador que posteriormente es utilizado en la búsqueda para evaluar subconjuntos, y con el que más tarde se obtendrán los resultados definitivos, por ejemplo, si se va a realizar una comparativa para comprobar los efectos de la reducción de datos en los resultados de clasificación con el algoritmo NB, en primer lugar se aplica individualmente a cada atributo dicho algoritmo de aprendizaje (NB) para generar un ranking, y se utiliza de nuevo en la búsqueda de subconjuntos.

Una vez obtenido el ranking, en el proceso de búsqueda llevado a cabo en la segunda parte (figura 5.2) para ver qué atributos añadir, se utiliza el resultado de una medida de evaluación de subconjuntos (*SubEvaluador*). En los experimentos realizados para comprobar la bondad de *BIRS* se emplean dos criterios de evaluación de subconjuntos representativos de los dos tipos de aproximaciones, siendo usados con frecuencia en los estudios comparativos: el resultado de clasificación o aproximación *wrapper* comentada en el párrafo anterior, que utiliza el algoritmo de aprendizaje objetivo para estimar la bondad del subconjunto de atributos, en las tablas se indica con el nombre del clasificador como superíndice, por ejemplo, en BI^{NB} se utiliza el clasificador NB como SubEvaluador en la segunda parte de *BIRS*; y como aproximación de tipo filtro una medida de correlación, *CFS*, que evalúa la bondad de un subconjunto de atributos basándose en la hipótesis de que un buen subconjunto contiene atributos muy correlacionados con la clase, pero poco correlados entre ellos, en las tablas se mostrará BI^{CF}. En definitiva, cada vez que se haga referencia al algoritmo *BIRS* se refleja la medidas utilizada en cada una de las partes del proceso de selección. Si se utiliza ^{WR} se refiere a las diferentes aproximaciones de tipo *wrapper*, por tanto *WR* abarca a los tres clasificadores (NB, C4 e IB).

Bases de datos pequeñas

Para demostrar la validez de *BIRS* en este grupo de bases de datos (tabla A.1), se han utilizado las medidas expuestas anteriormente (ver figura 5.2), es decir, se escogen tres criterios para generar una lista ordenada de atributos: *SOAP* (SP), indicado en la tabla como ${}_{SP}BI^{--}$,

incertidumbre simétrica (SU o CF), identificado como ${}_{CF}BI^{--}$, y ${}_{WR}BI^{--}$ indica que el ranking se ha elaborado con el clasificador que se va a aplicar posteriormente (*wrapper*); y en la segunda fase, se utiliza los dos *SubEvaluadores*, *wrapper* y *CFS* ($_{--}BI^{WR}$ y $_{--}BI^{CF}$), donde ^{WR} resume los resultados obtenidos con los tres clasificadores (NB, C4 o IB). Se debe tener en cuenta que en estos experimentos con bases de datos pequeñas, se añadirá al subconjunto final todo atributo candidato que mejore el resultado, sin utilizar umbral alguno.

En la comparativa se utilizaron técnicas secuenciales hacia adelante *SF* (*Sequential Forward*) y mejor primero *BF* (*Best First*) (ver sección 3.4.2, página 40), en combinación con los criterios de evaluación de subconjuntos antes mencionados, el resultado de clasificación y *CFS*, como aproximación *wrapper* y filtro respectivamente, indicándose con el superíndice correspondiente, por ejemplo SF^{WR} y BF^{CF}. También se compara con el método *FCBF* (*Fast Correlation–Based Filter* [184] apartado 3.5.1, página 52) que utiliza medidas de correlación para seleccionar el subconjunto. Para comparar los diferentes algoritmos de selección, se aplicaron los tres clasificadores (C4, IB y NB) sobre las bases de datos originales, así como sobre las obtenidas, con sólo los atributos seleccionados por cada uno de los algoritmos de selección, almacenando la precisión predictiva, el número de atributos seleccionados y el tiempo de ejecución. En cada caso, los resultados se obtuvieron calculando la media de diez ejecuciones de validación cruzada diez (10×10CV), y para que no se produzca un sobreajuste del algoritmo de selección a los datos utilizados, en cada ejecución se realizan diez reducciones, una por cada conjunto de entrenamiento (las Figuras 3.4 y 3.5 en la página 63 son ilustrativas del proceso a realizar con validación cruzada en la selección).

La tabla 5.1 proporciona para NB, C4, e IB respectivamente, el resumen de los resultados obtenidos con las treinta bases de datos que conforman el primer grupo de bases de datos (tabla A.1). En la primera línea de la tabla, se encuentra el promedio de las tasas de aciertos para cada clasificador con las bases de datos originales, siendo 75.47, 78.95 y 79.34, para NB, C4 e IB respectivamente. En las restantes líneas, se muestra para cada método combinado con la evaluación de subconjunto correspondiente, el promedio de los aciertos (Ac.), el porcentaje medio de atributos retenidos con respecto al original (%At) y el tiempo de ejecución en segundos (t(s)). En horizontal, los datos se agrupan por la medida de subconjuntos utilizada (*wrapper* y filtro–*CFS*), y dentro de cada una por método, es decir, cada línea de la tabla corresponde con el resultado promedio de aplicar un algoritmo a todas las bases de datos de la tabla; el grupo *wrapper* engloba cinco métodos (${}_{WR}BI^{WR}$, ${}_{SP}BI^{WR}$, ${}_{CF}BI^{WR}$, SF^{WR}, BF^{WR} ya comentados) que utilizan un clasificador como medida de evaluación de subconjuntos; a continuación, estos mismos cinco métodos realizan la búsqueda de subconjuntos utilizando *CFS* (excepto el cambio de ${}_{WR}BI^{WR}$ por ${}_{CF}BI^{CF}$); y la última línea muestra el resultado obtenido con el algoritmo *FCBF*.

Tabla 5.1: Comparativa de la técnica *BIRS*. Resultados obtenidos con los tres clasificadores, NB, C4 e IB, sobre los conjuntos de datos pequeños de *UCI* (tabla A.1). Aplicando evaluador de subconjuntos modelo *wrapper* y filtro (*CFS*). Ac–promedio de aciertos; %At–porcentaje medio de atributos retenidos con respecto al original; t (s)–tiempo de ejecución (en segundos).

		NB (75,47)			C4 (78,95)			IB (79,34)		
		Ac	%At	t (s)	Ac	%At	t (s)	Ac	%At	t (s)
wrapper	${}_{SP}BI^{WR}$	77,72	39,51	41,40	79,07	31,35	218,85	79,19	44,48	647,06
	${}_{CF}BI^{WR}$	77,48	37,90	44,31	79,20	32,03	227,89	79,71	43,86	720,55
	${}_{WR}BI^{WR}$	77,71	35,69	52,52	78,91	30,47	263,08	79,54	43,40	812,07
	SF^{WR}	77,80	29,75	185,69	78,64	23,61	905,16	79,08	35,16	3162,09
	BF^{WR}	78,37	40,45	437,38	79,24	31,57	1828,07	80,08	45,43	5993,27
CFS	${}_{SP}BI^{CF}$	75,28	36,55	0,98	78,89	=NB	=NB	76,38	=NB	=NB
	${}_{CF}BI^{CF}$	74,96	33,61	1,50	78,42	=NB	=NB	76,80	=NB	=NB
	${}_{WR}BI^{CF}$	75,26	37,51	10,82	78,57	37,99	38,60	76,96	35,70	114,85
	SF^{CF}	74,82	35,04	0,83	78,52	=NB	=NB	76,81	=NB	=NB
	BF^{CF}	75,22	36,33	1,06	78,64	=NB	=NB	77,11	=NB	=NB
FCBF		74,92	34,59	0,95	78,01	=NB	=NB	76,69	=NB	=NB

«=NB»–igual a los datos obtenidos con NB. ${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2; ^{WR} SubEvaluador *wrapper* y ^{CF} *CFS*; *SF*–búsqueda secuencial hacia adelante; *BF*–búsqueda *best first*; *FCBF–Fast Correlation-Based Filter*.

Hay que tener en cuenta que bajo la columna correspondiente al clasificador NB, los algoritmos del grupo *wrapper* emplean este mismo clasificador (NB) para obtener el subconjunto de atributos definitivo, así mismo, bajo la columna C4, se emplea este clasificador para extraer el subconjunto, y se aplica el clasificador IB bajo su columna correspondiente. Por utilizar distinto clasificador para obtener el subconjunto final, resultan subconjuntos con diferente número de atributos y en tiempos desiguales. No ocurre lo mismo al utilizar medidas de tipo filtro para generar el ranking y evaluar los subconjuntos de atributos, en este caso, el conjunto final seleccionado no dependerá del clasificador, por eso en la tabla se muestran sólo una vez en la columna correspondiente al clasificador NB. Nótese que para el caso ${}_{WR}BI$ con *SubEvaluador CFS*, al generarse el ranking con el clasificador destino, sí existe diferencia en los resultados.

Los resultados observados en la tabla para la tasa de aciertos son muy similares en todos los casos, no existen diferencias significativas entre los distintos algoritmos de selección, ni con respecto a las mediciones realizadas sobre las bases de datos originales. Se percibe una mejora de los resultados obtenidos con las aproximaciones *wrappers* con respecto a las filtros. No ocurre lo mismo si se comparan los porcentajes de atributos retenidos y el tiempo empleado. La mayores reducciones se consiguen con la búsqueda *SF* según el modelo *wrapper*. En cuanto al tiempo necesario para la reducción, en las aproximaciones de tipo filtro es despreciable, y en las *wrapper*, las versiones de *BIRS* son claramente mejores, siendo aproximadamente cinco veces más rápidas que *SF* y casi diez con respecto a *BF*.

Según los datos aportados por la tabla 5.1 se puede colegir que siendo *BIRS* un método orientado a bases de datos grandes, obtiene buenos resultados con las pequeñas y medianas, en un tiempo muy inferior al empleado por otros métodos muy populares. Además, los resultados obtenidos por *BIRS* no varían significativamente al partir de rankings elaborados con distintas medidas.

Bases de datos grandes

En este apartado, se realizaron las pruebas con los conjuntos de datos contenidos en las tablas A.2 y A.3. Para cada base de datos, se ejecutó la técnica de búsqueda propuesta elaborando el ranking con SP y el propio evaluador de subconjuntos; y de nuevo los dos *SubEvaluadores* (*wrapper* y filtro–*CFS*). Se nombra a los distintos algoritmos de la misma manera que la explicada en el apartado anterior (${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2). En estos experimentos con bases de datos grandes, para averiguar si el valor de medición de un conjunto es significativamente mejor ($\gg$) que otro al añadir un atributo, se distingue entre un *SubEvaluador* que siga el modelo filtro y otro que siga el modelo *wrapper*. En el primer caso, simplemente se comprueba si la mejora supera un umbral establecido en 0,005, sin embargo en el segundo caso,

se aplica un test t–Student pareado para evaluar si la diferencia entre el conjunto candidato es estadísticamente significativa (al nivel 0.1) con respecto al mejor subconjunto anterior.

Además, en la comparativa con las bases de datos de la primera tabla (A.2), se utilizará la técnica secuencial hacia adelante *SF* (*Sequential Forward*) en combinación con ambas medidas de subconjuntos (SF^{WR} y SF^{CF}) y el método *FCBF* (*Fast Correlation–Based Filter*). La búsqueda *BF* (*Best First*) se vuelve prohibitiva si se aplica en este tipo de bases de datos. Para comparar los diferentes algoritmos de selección, se aplicaron igualmente los tres clasificadores, C4, IB y NB, sobre las bases de datos originales, así como sobre las recién obtenidas, conteniendo sólo los atributos seleccionados por cada uno de los algoritmos de selección, almacenando la precesión predictiva, el número de atributos seleccionados y el tiempo de ejecución. En cada caso, los resultados se obtuvieron calculando la media de cinco ejecuciones de validación cruzada dos (5×2CV), y para que no se produzca un sobreajuste del algoritmo de selección a los datos utilizados, en cada ejecución se realizan dos reducciones, una por cada conjunto de entrenamiento (las Figuras 3.4 y 3.5 en la página 63 son ilustrativas del proceso a realizar con validación cruzada en la selección). Se cambia el método de validación de 10×10CV a 5×2CV, por ser demasiado costoso el aplicar la anterior validación a bases de datos grandes con aproximaciones wrappers.

Las tablas 5.2, 5.4 y 5.6 proporcionan para NB, C4, e IB respectivamente, las tasas de aciertos obtenidas con las doce bases de datos que conforman la tabla A.2. La tabla 5.2 muestra los efectos de la reducción de datos en la clasificación mediante NB. Tras la primera columna con los nombres de las bases de datos, los datos de las tres siguientes columnas que aparecen bajo la etiqueta *wrapper*, se corresponden con algoritmos que utilizan el método de aprendizaje como medida de evaluación de subconjuntos de atributos. Dentro de este grupo, en la primera columna ($_{NB}BI^{NB}$) la ordenación inicial del algoritmo *BIRS* se ha realizado utilizando el propio algoritmo de clasificación, en la segunda se utiliza *SOAP* ($_{SP}BI^{NB}$), y en la tercera se realiza una búsqueda secuencial (SF^{NB}). Del mismo modo, los métodos bajo el título de *CFS* ($_{CF}BI^{CF}$, $_{SP}BI^{CF}$ y SF^{CF}) utilizan esta medida para llegar al subconjunto final de atributos. Los resultados con *FCBF–* algoritmo *Fast Correlation-Based Filter*) se muestran en la columna siguiente, y la última columna de la tabla muestra el promedio de los datos originales sin reducir. Los símbolos ∘ y • respectivamente identifican si la diferencia con respecto a $_{NB}BI^{NB}$ es estadísticamente mejor o peor, al nivel 0.05 de confianza. Las tablas 5.4 y 5.6 mantienen la mis estructura.

Se puede comprobar en la tabla 5.3 que la distribución de los datos resultantes de los experimentos realizados con el clasificador NB es similar a la tabla anterior comentada (tabla 5.2), excepto que ahora por cada método se muestra el número de atributos seleccionados junto con el porcentaje de reducción respecto a la base de datos original. La última línea de la tabla con la

Tabla 5.2: Comparativa de la técnica *BIRS*. Tasas de aciertos obtenidas con el clasificador NB sobre grandes conjuntos de datos *UCI*.

	wrapper			*CFS*			*FCBF*	Orig
Datos	${}_{NB}BI^{NB}$	${}_{SP}BI^{NB}$	SF^{NB}	${}_{CF}BI^{CF}$	${}_{SP}BI^{CF}$	SF^{CF}		
ADS	95,42	95,84	95,83	95,38	95,20	95,81	95,64	96,38
ARR	66,99	68,10	67,70	66,50	66,37	68,05	63,98	60,13
HYP	95,10	95,01	95,32	94, 15•	94,61	94, 15•	94,90	95,32
ISO	83,30	81,35	82,28	77,61	76, 66•	80,79	74, 62•	80, 42•
KRV	94,27	94,27	94,32	90, 43•	90, 43•	90, 43•	92,50	87, 50•
LET	65,67	65,47	65,67	64, 28•	65,08	64, 28•	65,06	63, 97•
MUL	97,21	97,06	96,87	97,04	96,81	96,72	96,19	94,37
MUS	98,78	98,86	99,01	98,52	98,52	98,52	98,52	95, 10•
MUK	84,59	85,12	84,59	79,94	82,66	69, 78•	72,29	83,56
SIC	94,55	94,28	93,88	93,89	96,22	93,89	96,25	92,41
SPL	94,85	94,68	94,91	93, 63•	93, 63•	93, 60•	95,49	95,26
WAV	81,01	81,50	81,55	81,01	81,62	80,12	78, 42•	80,02

${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2; SF^{-2}–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*; Orig–Original. ◦, • mejora o empeora significativamente con respecto a ${}_{NB}BI^{NB}$.

Tabla 5.3: Comparativa de la técnica *BIRS*. Número de atributos seleccionados y porcentaje de reducción respecto a la base de datos original para las aproximaciones *wrappers* con NB y filtros.

	wrapper						*CFS*						*FCBF*	
	${}_{NB}BI^{NB}$		${}_{SP}BI^{NB}$		SF^{NB}		${}_{CF}BI^{CF}$		${}_{SP}BI^{CF}$		SF^{CF}			
Datos	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At
ADS	10,5	0,7	8,7	0,6	16,4	1,1	6,7	0,4	6,1	0,4	9,2	0,6	83,1	5,3
ARR	5,8	2,1	9,4	3,4	8,4	3,0	11,4	4,1	14,3	5,1	17,2	6,2	8,0	2,9
HYP	4,6	15,9	4,2	14,5	8,5	29,3	1,0	3,4	3,1	10,7	1,0	3,4	5,3	18,3
ISO	68,5	11,1	61,9	10,0	29,0	4,7	68,8	11,1	74,4	12,1	95,2	15,4	22,9	3,7
KRV	5,0	13,9	5,0	13,9	5,2	14,4	3,0	8,3	3,0	8,3	3,0	8,3	6,5	18,1
LET	11,0	68,8	12,0	75,0	11,6	72,5	9,0	56,3	10,0	62,5	9,0	56,3	10,3	64,4
MUL	22,2	3,4	32,7	5,0	15,3	2,4	28,0	4,3	18,4	2,8	90,3	13,9	121,3	18,7
MUS	2,1	9,5	2,4	10,9	3,0	13,6	1,0	4,5	1,0	4,5	1,0	4,5	3,6	16,4
MUK	1,0	0,6	1,7	1,0	1,0	0,6	6,5	3,9	4,5	2,7	16,3	9,8	2,9	1,7
SIC	2,4	8,3	1,9	6,6	1,0	3,5	1,0	3,4	3,1	10,7	1,0	3,4	4,8	16,6
SPL	13,1	21,8	13,6	22,7	14,8	24,7	6,0	10,0	6,0	10,0	6,1	10,2	21,8	36,3
WAV	9,4	23,5	9,3	23,3	12,9	32,3	12,4	31,0	11,5	28,8	14,8	37,0	6,1	15,3
Prom.		15,0		15,6		16,8		11,7		13,2		14,1		18,1

etiqueta *Prom.* muestra el promedio de los datos de la columna correspondiente. Las tablas 5.5 y 5.7 informan de la misma manera pero con los datos referidos a los algoritmos de clasificación C4 e IB. Hay que tener en cuenta, que al utilizar medidas de tipo filtro (${}_{CF}BI^{CF}$, ${}_{SP}BI^{CF}$, SF^{CF} y *FCBF*) para generar el ranking y evaluar los subconjuntos de atributos, el conjunto final seleccionado no dependerá del clasificador, por eso, en las tres tablas aparecen los mismos resultados para este tipo de algoritmos. Se muestran repetidos por facilidad y claridad en la interpretación de los resultados. Finalmente, en la tabla 5.8 se puede comprobar el tiempo empleado por los diferentes algoritmos, donde se puede observar, igualmente, que se dispone de un valor por base de datos y por algoritmo de tipo filtro, mientras que se cuenta con tres, uno por clasificador, para cada método *wrapper*.

Antes de comparar la técnica *BIRS* con las demás, se resalta la similitud entre los resultados obtenidos con sus diferentes versiones. Observando las tablas 5.2, 5.4 y 5.6 correspondientes a los tres clasificadores, se comprueba en las columnas de las dos versiones existentes de *BIRS* para cada aproximación (*wrapper* y *CFS*), que las diferencias entre las exactitudes no son estadísticamente significativas, excepto en un caso, las versiones wrappers con el clasificador C4. Además, el número de atributos seleccionado es similar, excepto para el clasificador IB con algunas bases de datos. En cuanto al tiempo, ${}_{SP}BI^{WR}$ es un poco más rápido que ${}_{WR}BI^{WR}$ debido al tiempo necesario para construir el ranking para la aproximación ranking–*wrapper*, y más rápido que ${}_{CF}BI^{WR}$ porque generar un ranking con SP es más rápido que generarlo con la correlación (ver experimentos del capítulo 4).

Además de la comparación anterior, se ha estudiado el comportamiento de ${}_{WR}BI^{WR}$ comparándolo en tres direcciones: con respecto al conjunto original de atributos, con respecto a la aproximación secuencial *wrapper* (SF^{WR}) y con respecto a otros métodos filtros.

Como se puede observar en la última columna de las tablas 5.2, 5.4 y 5.6, las exactitudes de clasificación obtenidas con la aproximación *wrapper* de *BIRS* (${}_{WR}BI^{WR}$) con respecto a las obtenidas con el conjunto total de atributos son estadísticamente mejores en 4 y 3 ocasiones para los clasificadores NB e IB respectivamente, y peores en dos para el clasificador C4. Llama la atención que el número de atributos seleccionados es drásticamente inferior al del conjunto original, reteniendo un promedio del 15 % (NB, tabla 5.3), 16,3 %(C4, tabla 5.5) y del 13,1 % (IB, tabla 5.7) de los atributos. Se puede observar que ${}_{WR}BI^{WR}$ escoge menos del 10 % de los atributos en más de la mitad de todos los casos estudiados en estas tablas.

No existen diferencias estadísticamente significativas, con los resultados obtenidos con los tres clasificadores, entre las exactitudes de la aproximación *wrapper* de *BIRS* (${}_{WR}BI^{WR}$) y las exactitudes obtenidas con el procedimiento *wrapper* secuencial hacia adelante (SF^{WR}), excepto para la base de datos MUS con el clasificador C4. Se debe resaltar que en dos casos con el

Tabla 5.4: Comparativa de la técnica *BIRS*. Tasas de aciertos obtenidas con el clasificador C4 sobre grandes conjuntos de datos *UCI*.

	wrapper			*CFS*			*FCBF*	Orig
Datos	${}_{C4}BI^{C4}$	${}_{SP}BI^{C4}$	SF^{C4}	${}_{CF}BI^{CF}$	${}_{SP}BI^{CF}$	SF^{CF}		
ADS	96,55	96,14	96,85	96,43	96,40	96,39	95,85	96,46
ARR	68,01	64,56	67,39	66,42	65,31	67,04	64,87	64,29
HYP	99,07	98,57	99,30	96, 56•	97,14	96, 56•	98,03	99,36
ISO	69,43	70,24	N/D	72,68	72,10	71,94	66,63	73,38
KRV	95,11	95,41	94,26	90, 43•	90, 43•	90, 43•	94,07	99, 07◦
LET	84,99	84,95	85,17	84, 21•	84,37	84, 21•	84,84	84,45
MUL	92,42	93,21	93,11	93,17	93,64	93,12	92,29	92,74
MUS	99,91	99,91	100, 00◦	98, 52•	98, 52•	98, 52•	98, 84•	100, 00◦
MUK	95,43	95,39	N/D	94,06	93,66	94,60	91, 19•	95,12
SIC	98,28	96, 26•	98,19	96, 33•	97,52	96, 33•	97,50	98,42
SPL	93,05	93,11	93,04	92,54	92,54	92,61	93,17	92,92
WAV	76,20	75,99	75,44	76,46	76,45	76,56	74,52	74,75

${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2; SF^{-2}–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*; Orig–Original. ◦, • mejora o empeora significativamente con respecto a ${}_{C4}BI^{C4}$. N/D–No Disponible.

Tabla 5.5: Comparativa de la técnica *BIRS*. Número de atributos seleccionados y porcentaje de reducción respecto a la base de datos original para las aproximaciones *wrappers* con C4 y filtros.

	wrapper						*CFS*						*FCBF*	
	${}_{C4}BI^{C4}$		${}_{SP}BI^{C4}$		SF^{C4}		${}_{CF}BI^{CF}$		${}_{SP}BI^{CF}$		SF^{CF}			
Datos	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At
ADS	8,5	0,5	9,8	0,6	12,4	0,8	6,7	0,4	6,1	0,4	9,2	0,6	83,1	5,3
ARR	6,7	2,4	13,4	4,8	8,6	3,1	11,4	4,1	14,3	5,1	17,2	6,2	8,0	2,9
HYP	4,2	14,5	4,5	15,5	5,9	20,3	1,0	3,4	3,1	10,7	1,0	3,4	5,3	18,3
ISO	22,5	3,6	49,5	8,0	N/D		68,8	11,1	74,4	12,1	95,2	15,4	22,9	3,7
KRV	6,2	17,2	6,6	18,3	4,9	13,6	3,0	8,3	3,0	8,3	3,0	8,3	6,5	18,1
LET	11,0	68,8	11,9	74,4	10,1	63,1	9,0	56,3	10,0	62,5	9,0	56,3	10,3	64,4
MUL	20,6	3,2	25,5	3,9	13,6	2,1	28,0	4,3	18,4	2,8	90,3	13,9	121,3	18,7
MUS	4,1	18,6	4,1	18,6	4,9	22,3	1,0	4,5	1,0	4,5	1,0	4,5	3,6	16,4
MUK	9,7	5,8	10,3	6,2	N/D		6,5	3,9	4,5	2,7	16,3	9,8	2,9	1,7
SIC	5,9	20,3	2,0	6,9	5,5	19,0	1,0	3,4	3,1	10,7	1,0	3,4	4,8	16,6
SPL	9,8	16,3	8,8	14,7	11,0	18,3	6,0	10,0	6,0	10,0	6,1	10,2	21,8	36,3
WAV	9,6	24,0	7,4	18,5	7,9	19,8	12,4	31,0	11,5	28,8	14,8	37,0	6,1	15,3
Prom.		16,3		15,9		18,2*		11,7		13,2		14,1		18,1

* Tener en cuenta que en el promedio no están incluidos los datos para ISO y MUK.

Tabla 5.6: Comparativa de la técnica *BIRS*. Tasas de aciertos obtenidas con el clasificador IB sobre grandes conjuntos de datos *UCI*.

	wrapper			*CFS*			*FCBF*	Orig
Datos	${}_{IB}BI^{IB}$	${}_{SP}BI^{IB}$	SF^{IB}	${}_{CF}BI^{CF}$	${}_{SP}BI^{CF}$	SF^{CF}		
ADS	95,28	95,85	N/D	95,93	95,74	96,07	95,75	95,95
ARR	62,74	62,21	57,12	61,37	60,35	61,06	58,67	54,12
HYP	83,66	94,31	83,57	85,75	88,92	85,75	94,88	90,85
ISO	80,64	80,68	78,61	79,37	78,97	80,28	72, 57•	77,58
KRV	92,27	94,26	94,24	90,43	82,97	90,43	93,85	89,21
LET	95,52	95,05	95,58	93, 62•	94, 46•	93, 62•	94,81	94, 23•
MUL	96,72	97,42	N/D	97,54	97,60	97,70	97,53	97,52
MUS	98,36	99,78	99,99	98,52	98,52	98,52	98,88	100,00
MUK	93,34	94,62	94,72	92,59	93,04	93,17	89, 04•	95,14
SIC	96,55	95,81	97,05	94,73	96,85	94,73	95,82	95,58
SPL	86,35	85,67	85,62	86,40	86,39	86,34	79, 21•	73, 74•
WAV	76,39	77,91	77,18	78, 89◦	78,60	78,72	71, 76•	73, 42•

${}_{-1}BI^{-2}$*–BIRS* con ranking 1 y SubEvaluador 2; SF^{-2}–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*; Orig–Original. ◦, • mejora o empeora significativamente con respecto a ${}_{IB}BI^{IB}$. N/D–No Disponible.

Tabla 5.7: Comparativa de la técnica *BIRS*. Número de atributos seleccionados y porcentaje de reducción respecto a la base de datos original para las aproximaciones *wrappers* con IB y filtros.

	wrapper						*CFS*						*FCBF*	
	${}_{IB}BI^{IB}$		${}_{SP}BI^{IB}$		SF^{IB}		${}_{CF}BI^{CF}$		${}_{SP}BI^{CF}$		SF^{CF}			
Datos	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At	#At	%At
ADS	5,2	0,3	15,2	1,0	N/D		6,7	0,4	6,1	0,4	9,2	0,6	83,1	5,3
ARR	14,1	5,1	24,5	8,8	12,7	4,6	11,4	4,1	14,3	5,1	17,2	6,2	8,0	2,9
HYP	1,0	3,4	7,7	26,6	1,0	3,4	1,0	3,4	3,1	10,7	1,0	3,4	5,3	18,3
ISO	35,5	5,8	43,7	7,1	29,4	4,8	68,8	11,1	74,4	12,1	95,2	15,4	22,9	3,7
KRV	6,5	18,1	12,6	35,0	10,0	27,8	3,0	8,3	3,0	8,3	3,0	8,3	6,5	18,1
LET	10,9	68,1	11,9	74,4	11,0	68,8	9,0	56,3	10,0	62,5	9,0	56,3	10,3	64,4
MUL	11,3	1,7	24,1	3,7	N/D		28,0	4,3	18,4	2,8	90,3	13,9	121,3	18,7
MUS	1,6	7,3	4,8	21,8	4,7	21,4	1,0	4,5	1,0	4,5	1,0	4,5	3,6	16,4
MUK	4,7	2,8	11,4	6,9	12,0	7,2	6,5	3,9	4,5	2,7	16,3	9,8	2,9	1,7
SIC	2,8	9,7	5,6	19,3	6,7	23,1	1,0	3,4	3,1	10,7	1,0	3,4	4,8	16,6
SPL	5,9	9,8	6,5	10,8	6,6	11,0	6,0	10,0	6,0	10,0	6,1	10,2	21,8	36,3
WAV	10,0	25,0	12,1	30,3	12,4	31,0	12,4	31,0	11,5	28,8	14,8	37,0	6,1	15,3
Prom.		13,1		20,5		20,3*		11,7		13,2		14,1		18,1

* Tener en cuenta que en el promedio no están incluidos los datos para ADS y MUL.

clasificador C4 (ISO y MUK) y dos con IB (ADS y MUL), SF^{WR} no aportó resultado alguno después de tres semanas ejecutándose, por tanto, no se dispone de atributos seleccionados ni tasa de aciertos. Sin tener en cuenta esta falta de resultados con SF^{WR}, los subconjuntos elegidos por *BIRS* son considerablemente menores con el clasificador IB, 13.1 % frente a 20 %, y algo menos diferencia con NB y C4, aunque es de suponer que los resultados que faltan favorecerían a *BIRS*, dado que si *SF* no ha terminado es por la inclusión de muchos atributos. Por otra parte, la ventaja de ${}_{WR}BI^{WR}$ frente a SF^{WR} es clara en cuanto al tiempo necesario para las ejecuciones, como se puede observar en la tabla 5.8, ${}_{WR}BI^{WR}$ es consistentemente más rápido que SF^{WR} (sin tener en cuenta el tiempo que habría que añadir a SF^{WR} por las bases de datos que faltan). El tiempo ahorrado por *BIRS* se hace más obvio cuando las necesidades de computación del algoritmo de aprendizaje se incrementan. En muchos casos, el tiempo necesario fue diez veces menor con *BIRS*. Estos resultados confirman que la eficiencia de la búsqueda incremental aplicada por *BIRS* es computacionalmente superior a la secuencial aplicada por SF^{WR}, con similar o menor número de atributos seleccionados y sin diferencias estadísticamente significativas entre exactitudes.

En general, el coste computacional de los procesos filtros se pueden considerar insignificantes frente a los wrappers (tabla 5.8). Sin embargo, las exactitudes obtenidas con la aproximación *wrapper* de *BIRS* (${}_{WR}BI^{WR}$) son mejores: gana significativamente a la primera versión filtro de *BIRS*, ${}_{CF}BI^{CF}$, en 4, 5 y 1 ocasiones para NB, C4 e IB respectivamente, y sólo pierde en una con IB; con respecto al filtro ${}_{SP}BI^{CF}$, gana en 3, 2 y 1 con NB, C4 e IB, respectivamente; a la versión secuencial SF^{CF}, *BIRS* gana en 5, 5 y 1 ocasiones para NB, C4 e IB respectivamente; y con respecto a *FCBF*, ${}_{WR}BI^{WR}$ *wrapper* es mejor en 2, 2 y 4 ocasiones con cada clasificador respectivo.

A diferencia de la aproximación *wrapper*, se dispone de un sólo subconjunto de atributos para cada algoritmo filtro. ${}_{CF}BI^{CF}$ retiene un 11,7 % del total de atributos de promedio de las doce bases de datos y ${}_{SP}BI^{CF}$ retiene un 13.2 %, SF^{CF} retiene el 14.1 % en el promedio de todas las bases de datos, mientras que *FCBF* es el filtro que más atributos retiene con un 18.1 %. Los tres primeros filtros retienen menos atributos en promedio que la versión *wrapper* de *BIRS*, y *FCBF* retiene más atributos que *BIRS* con los clasificadores NB e IB.

En estos experimentos se ha utilizado el algoritmo *FCBF* implementado en el entorno WEKA con los valores por defecto, sin embargo, si se modifica el umbral por el cual los atributos pueden ser descartados, los resultados obtenidos serán muy diferentes. Lo normal que ocurra es que disminuya el volumen de los conjuntos seleccionados, pero asociado a una pérdida de predicción muy considerable.

Otra comparación, no contemplada hasta el momento, puede ser entre las versiones filtros,

es decir, como se comporta la aproximación filtro de *BIRS* ($_{CF}BI^{CF}$) con respecto a la búsqueda secuencial SF^{CF} y al algoritmo *FCBF*. En cuanto a las exactitudes, los resultados obtenidos con los dos primeros (*BIRS* y *SF*) son similares y algo por encima de las obtenidas con *FCBF*. Sin embargo, los conjuntos más reducidos se obtienen con las versiones filtros de *BIRS*, y además, con un coste computacional inferior a los demás algoritmos.

Bases de datos *NIPS*. La tabla 5.9 muestra los resultados obtenidos con los clasificadores NB, C4 e IB sobre los datos generados para el congreso *NIPS–Neural Information Processing Systems* (ver tabla A.3 en la página 185) celebrado en el año 2003. La forma de nombrar los distintos algoritmos es la seguida en los apartados anteriores ($_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2). Por columnas se distinguen los resultados obtenidos con *BIRS* utilizando como evaluador de subconjuntos el propio clasificador (*wrapper*) y utilizando *CFS*, los resultados que devuelve el algoritmo *FCBF* y los obtenidos con la base de datos completa. Las columnas encabezadas por Ac y por #At muestran la tasa de aciertos y el número de atributos (el porcentaje de reducción es demasiado bajo para ser comparable) respectivamente.

Para los conjuntos de la segunda tabla de datos (A.3) no se muestran los resultados de *SF* dado que, en su aproximación *wrapper* es demasiado costoso en tiempo, y en su aproximación filtro, selecciona tantos atributos que se queda sin memoria fácilmente. Por otro lado, se ha modificado el algoritmo *CFS* para poder obtener resultados con las bases de datos DEX y DOR. Esta modificación hace que se necesite mucha menos cantidad de memoria, a costa de añadir coste temporal al algoritmo.

Las conclusiones más importantes que se pueden resaltar, son las siguientes:

- La técnica *BIRS* es un buen método para seleccionar atributos, dado que con un conjunto muy reducido de atributos, se mantiene, y en algunos casos se mejora, los resultados de clasificación de bases de datos grandes. En cuanto a la exactitud obtenida por *BIRS*, sobresale especialmente cuando se aplica el clasificador C4, ganando considerablemente en cuatro de las cinco bases de datos; con el clasificador NB, *BIRS* obtiene buenos resultados sobre la base de datos DEX; y aplicando IB, pierde en ARC y GIS, sin embargo gana aproximadamente por 20 puntos en las bases de datos DEX y MAD. En todos los casos, la reducción obtenida con respecto al conjunto original de datos es drástica, resaltando la conseguida con la base de datos DOR, donde se queda aproximadamente con el 0,01 % de los atributos (10 de 100000) en todos los casos.

- El comportamiento de las aproximaciones filtro de *BIRS* es excelente. Produce tasas de aciertos similares a las aproximaciones *wrappers*, incluso en algunos casos superiores

Tabla 5.8: Comparativa de la técnica *BIRS*. Tiempo (en segundos) al aplicar cada selector a grandes conjuntos de datos *UCI*.

Tiempo	NB			C4			IB			correlación*CFS*			
Datos	${}_{NB}BI^{NB}$	${}_{SP}BI^{NB}$	SF^{NB}	${}_{C4}BI^{C4}$	${}_{SP}BI^{C4}$	SF^{C4}	${}_{IB}BI^{IB}$	${}_{SP}BI^{IB}$	SF^{IB}	${}_{CF}BI^{CF}$	${}_{SP}BI^{CF}$	SF^{CF}	*FCBF*
ADS	217,5	75,7	2541,2	1895,2	1129,2	18465,9	13579,9	26293,9	N/D	7,0	7,2	14,3	37,6
ARR	53,6	72,4	370,7	119,3	251,1	730,0	102,3	127,8	716,7	0,4	0,3	0,8	0,2
HYP	9,9	4,3	67,6	14,6	7,3	69,0	128,0	215,9	51,1	0,1	0,1	0,1	0,1
ISO	4363,5	3641,1	37293,2	11632,0	10881,4	N/D	5968,4	6895,2	82241,8	21,1	13,9	58,2	9,5
KRV	2,5	1,5	8,9	7,3	8,6	19,3	261,6	319,2	1901,0	0,0	0,1	0,1	0,0
LET	134,4	120,6	629,6	540,0	627,7	2703,8	6565,8	5620,0	26829,9	1,5	1,1	1,3	0,9
MUL	1251,4	1677,8	8416,8	2503,6	4358,1	16975,3	4810,3	9905,9	N/D	12,0	7,0	49,7	15,9
MUS	4,2	2,7	9,4	5,8	4,4	18,4	541,1	430,5	1906,5	0,1	0,2	0,1	0,1
MUK	40,7	30,5	16,2	897,2	836,5	N/D	6563,0	21723,7	85601,6	5,5	3,6	7,0	2,6
SIC	2,3	1,3	1,1	12,3	1,8	66,3	189,6	145,6	1517,7	0,1	0,1	0,1	0,1
SPL	7,8	6,7	65,1	32,6	31,2	218,7	427,7	364,6	1678,9	0,1	0,2	0,2	0,3
WAV	24,5	20,0	200,4	253,6	193,7	830,9	1115,3	1190,0	8197,0	0,7	0,6	0,8	0,3
Total	6112,2	5654,5	49620,2	17913,6	18331,0	40097,6	40253,0	73232,4	210642,2	48,6	34,2	132,6	67,5

${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2; SF^{-2}–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*.

Tabla 5.9: Comparativa de la técnica *BIRS*. Resultados obtenidos con los clasificadores NB, C4 e IB sobre datos *NIPS*. Orig–Original; Ac–tasa de aciertos; #At–número de atributos;

		wrapper				*CFS*				*FCBF*		Orig
		${}_{WR}BI^{WR}$		${}_{SP}BI^{WR}$		${}_{CF}BI^{CF}$		${}_{SP}BI^{CF}$				
Datos		Ac	#At	Ac	#At	Ac	#At	Ac	#At	Ac	#At	
Clasif. NB	ARC	64,60	15,3	67,40	25,1	63,20	39,2	69,20	34,5	61,20	35,2	65,40
	DEX	81,33	30,2	79,47	32,4	82,47	11,3	81,20	12,4	85,07	25,1	86,47
	DOR	93,23	10,5	93,55	9,6	93,80	11,9	93,80	13,0	92,38	75,3	90, 68•
	GIS	92,66	35,3	92,66	38,4	90,83	11,6	88, 73•	12,3	87, 58•	31,2	91,88
	MAD	59,00	11,8	60,00	14,9	60,56	5,8	60,38	4,5	58,20	4,7	58,24
Clasif. C4	ARC	65,80	7,9	63,80	14,2	59,00	39,2	61,00	34,5	58,80	35,2	57,00
	DEX	80,27	18,9	79,53	19,4	81,47	11,3	83,47	12,4	79,00	25,1	73,80
	DOR	92,13	7,2	92,70	8,3	91,63	11,9	91,73	13,0	90,33	75,3	88,73
	GIS	93,29	26,9	91,98	5,8	90,92	11,6	91, 53•	12,3	90, 99•	31,2	92,68
	MAD	73,02	17,0	72,11	18,8	69,77	5,8	67,73	4,5	61,11•	4,7	57, 73•
Clasif. IB	ARC	69,00	15,1	68,80	14,0	68,60	39,2	72,60	34,5	62,00	35,2	78,00
	DEX	81,00	34,1	77,40	22,6	81,73	11,3	82,47	12,4	79,20	25,1	56, 67•
	DOR	92,18	3,5	92,22	4,2	90,98	11,9	91,70	13,0	90,35	75,3	90,25
	GIS	82,25	2,3	84,16	2,8	90,07	11,6	90,78	12,3	90,06	31,2	95,21
	MAD	74,92	14,4	85,76	11,0	71,59	5,8	73,94	4,5	56,90	4,7	54,39

${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2; ^{WR} SubEvaluador *wrapper* (NB, C4 e IB) y ^{CF} *CFS*; *FCBF–Fast Correlation-Based Filter*. ∘, • mejora o empeora significativamente con respecto a ${}_{WR}BI^{WR}$.

Tabla 5.10: Comparativa de la técnica *BIRS*. Tiempo (en segundos) al aplicar cada selector a grandes conjuntos de datos *NIPS*.

	NB		C4		IB		Filtros		
Datos	${}_{NB}BI^{NB}$	${}_{SP}BI^{NB}$	${}_{C4}BI^{C4}$	${}_{SP}BI^{C4}$	${}_{IB}BI^{IB}$	${}_{SP}BI^{IB}$	${}_{CF}BI^{CF}$	${}_{SP}BI^{CF}$	*FCBF*
ARC	266,4	281,2	384,5	511,3	336,8	242,2	96,0	74,6	2,8
DEX	1540,4	1877,9	4460,4	10847,8	8945,7	7719,8	259,7	109,4	8,9
DOR	7137,2	4389,0	16525,1	19894,7	42538,9	22930,0	7525,5	7778,8	1079,8
GIS	9075,2	8736,6	34630,7	9361,2	77951,8	32786,2	440,7	342,6	118,0
MAD	137,5	121,2	1269,1	1543,2	3863,4	6824,6	3,5	2,5	1,4
Total	18156,6	15405,9	57269,9	42158,2	13145,8	14786,6	8325,5	8308,0	1211,0

${}_{-1}BI^{-2}$–*BIRS* con ranking 1 y SubEvaluador 2; *FCBF–Fast Correlation-Based Filter*.

($_{SP}BI^{CF}$ en la base de datos DEX con el clasificador C4, y en ARC y DEX con IB), siendo el número de atributos igual o inferior a los *wrappers* (en las aproximaciones filtro, el número de atributos no depende del clasificador aplicado). El tiempo empleado por las versiones filtro está muy por debajo del utilizado por las versiones *wrappers*. Se puede comprobar que no existen diferencias considerables entre generar un ranking con el criterio SP o con el propio evaluador de subconjuntos *CFS*.

- Si se compara las diferentes aproximaciones del algoritmo *BIRS* con *FCBF*, se puede comprobar que, excepto con el clasificador NB sobre la base de datos DEX, las exactitudes obtenidas por *FCBF* están normalmente por debajo de las obtenidas con *BIRS*, destacando las diferencias existentes en el conjunto de datos MAD con el clasificador C4, y con los conjuntos ARC y MAD con el clasificador IB. Los subconjuntos seleccionados por *FCBF* son mayores que los elegidos por $_{SP}BI^{CF}$ en tres de los cinco conjuntos, sin embargo, el coste temporal es aproximadamente siete veces menos.

En resumen, aunque los datos reportados por el algoritmo *FCBF* son buenos, los obtenidos con la técnica *BIRS* son mejores, consiguiendo aumentar la tasa de aciertos con unos conjuntos de atributos más reducidos.

5.5. Utilidad aglomerativa ordenada

La solución propuesta anteriormente está condicionada en gran medida por el primer atributo del ranking obtenido en la fase inicial del algoritmo. Aunque poco frecuente, puede darse la situación en la que exista un subconjunto mejor en el que no se encuentre ese atributo.

Teniendo en cuenta los conceptos de relevancia y redundancia ya vistos (sección 3.3, página 33) y lo expuesto en la sección 5.3 (definiciones 3.10 y 3.11), proponemos una nueva versión de búsqueda incremental que salva el inconveniente de la dependencia del primer atributo, incrementando el espacio de búsqueda con los subconjuntos de atributos obtenidos con los primeros atributos del ranking.

Sea Ψ el conjunto que contiene la lista de subconjuntos de atributos candidatos, siendo $\Psi_1 = \{A^1_1, A^1_2, \ldots, A^1_n\}$ la lista inicial de subconjuntos, donde $A^1_i = \{X_i\}$, es decir, cada subconjunto de la lista contiene un atributo.

Sean dos secuencias ordenadas:

$\Psi^k_1 =< A^1_{(1)}, \ldots, A^1_{(k)} >$ los k primeros de Ψ_1 ordenados por $\mathcal{L}$ descendentemente y $\Psi^\epsilon_1 =< A^1_{(1)}, \ldots, A^1_{(\epsilon)} >$ los ϵ primeros de Ψ_1 ordenados igualmente por $\mathcal{L}$, siendo $k \leq \epsilon \leq n$, y n el número total de atributos.

Sea $T_1 = \Gamma(\mathcal{E}/A^1_{(1)}, \mathcal{L})$, el resultado de aplicar el evaluador de subconjuntos $\mathcal{L}$ al subconjunto mejor posicionado en la lista Ψ^k_1. Como ya se ha señalado, Γ engloba tanto al valor de precisión de un clasificador (*wrapper*) $\mathcal{L}$, como al valor de medición obtenido con una medida tipo filtro ($\Upsilon(\mathcal{E}/A^1_{(1)}, Ł)$). Se construyen conjuntos en base a las dos secuencias ordenadas anteriores, de manera que se une cada conjunto de la primera secuencia a cada uno de la segunda.

Sea $\Psi_2 = \{A^2_i | A^2_i = A^1_{(j)} \cup A^1_{(l)} \; \forall j : 1..k$ y $\forall l : 1..\epsilon$ con $A^1_{(j)} \in \Psi^k_1$ y $A^1_{(l)} \in \Psi^\epsilon_1$ y con $\Gamma(\mathcal{E}/A^2_i, \mathcal{L}) > T_1\}$.

Sean Ψ^k_2 y Ψ^ϵ_2 las secuencias iguales a las anteriores, en general, se define la lista de soluciones Ψ_p, como $\Psi_p = \{A^p_i | A^p_i = A^{p-1}_{(j)} \cup A^{p-1}_{(l)}\} y \Gamma(\mathcal{E}/A^p_i, \mathcal{L}) > T_{p-1}\}$, como el conjunto de subconjuntos de atributos candidatos, estando formado cada subconjunto por la unión de dos subconjuntos de la lista de soluciones anterior.

El proceso continua hasta que algún Ψ_p sea $\varnothing$ entonces $\mathcal{S} = A^{p-1}_{(1)}$. Es decir, hasta que no se generen nuevos subconjuntos, o los generados no superen el valor de bondad del mejor subconjunto de la lista de soluciones anterior.

Se le llama aglomerativo a este método por la forma en la que construye el subconjunto final de atributos seleccionados. Como el modelo anterior, comienza generando un ranking, pero en vez de obtener el conjunto final comparando la mejora obtenida al añadir elementos consecutivos del ranking, se obtendrán parejas de atributos formadas con los primeros atributos del ranking en combinación con cada uno de los restantes atributos de la lista. Se ordenan las parejas de atributos según el valor de la evaluación realizada, y se repite el proceso, es decir, se compararán los subconjuntos formados por los primeros conjuntos de la nueva lista combinados con el resto de conjuntos. El proceso terminará cuando sólo quede un subconjunto de atributos, o cuando no se produzca mejora al combinar los diferentes subconjuntos.

A diferencia de la búsqueda tradicional secuencial hacia adelante, donde se obtiene sucesivamente el mejor atributo, la mejor pareja, el mejor conjunto de tres,... así hasta que no se produzca mejoría, el método *BARS* realiza una búsqueda con un coste menor, dado que recorre una parte menor del espacio formado por los atributos. Además, el recorrido de la búsqueda que desarrolla *BARS* se hace entorno a los subconjuntos más relevantes en cada momento, escogiendo los mejores k subconjuntos en cada ciclo del algoritmo, y expandiendo la búsqueda, hacia otros subconjuntos del ranking (porcentaje del ranking ϵ) con un mínimo de relevancia.

5.5.1. Algoritmo

El procedimiento seguido hasta obtener el subconjunto final es el siguiente (ver algoritmo 5.2): en el primer paso se genera un ranking de atributos (línea 1) ordenado de mejor a peor según alguna medida de evaluación ($\mathcal{U}$); a continuación, se genera una lista de soluciones (lí-

Algoritmo 5.2 *BARS–Best Agglomerative Ranked Subset*.

Entrada: $\mathcal{E}$–conjunto de datos, $\mathcal{U}$–criterio ranking, $\mathcal{L}$–*SubEvaluador*, k–nº de conjuntos iniciales del ranking, ϵ–limita el número de subconjuntos del ranking.

Salida: $MejorSub$–subconjunto de atributos

1: $R \Leftarrow generaRanking(\mathcal{U}, \mathcal{E})$
2: $p \Leftarrow 1$
3: $ListaSoluciones\Psi_p \Leftarrow \emptyset$
4: **para** $i = 1$ hasta n **hacer**
5: $\quad ListaSoluciones\Psi_p \Leftarrow ListaSoluciones\Psi_p \cup \{A_i^1\}$ $(A_i^1 = \{X_i\} \ \wedge \ X_i \in R)$
6: **fin para**
7: **mientras** $\#ListaSoluciones\Psi_p > 1$ **hacer**
8: $\quad T \Leftarrow \Gamma(\mathcal{E}/A_1^p, \mathcal{L})$
9: $\quad p \Leftarrow p + 1$
10: $\quad ListaSoluciones\Psi_p \Leftarrow \emptyset$
11: $\quad i \Leftarrow 1$
12: $\quad$ **para** $j = 1$ hasta k **hacer**
13: $\quad\quad$ **para** $l = j + 1$ hasta ϵ **hacer**
14: $\quad\quad\quad A_i^p \Leftarrow A_{(j)}^{p-1} \cup A_{(l)}^{p-1}$
15: $\quad\quad\quad$ **si** $\Gamma(\mathcal{E}/A_i^p, \mathcal{L}) > T$ **entonces**
16: $\quad\quad\quad\quad ListaSoluciones\Psi_p \Leftarrow ListaSoluciones\Psi_p \cup A_i^p$
17: $\quad\quad\quad\quad i \Leftarrow i + 1$
18: $\quad\quad\quad$ **fin si**
19: $\quad\quad$ **fin para**
20: $\quad$ **fin para**
21: $\quad ordenar\ ListaSoluciones\Psi_p\ por\ \Gamma(\mathcal{E}/A_i^p, \mathcal{L})$
22: **fin mientras**
23: $MejorSub \Leftarrow A_1^{p-1}$

nea 2–6, $Listasoluciones\Psi_p$), de manera que se crea una solución por cada atributo individual y se mantiene el mismo orden del ranking; entre las líneas 7 y 22 se muestran los pasos necesarios para realizar la búsqueda aglomerativa; al final, el algoritmo devuelve el subconjunto mejor posicionado de todos los evaluados.

La búsqueda aglomerativa consiste en formar un subconjunto de atributos relevantes uniendo subconjuntos con un menor número de atributos.

En cada iteración de la estructura repetitiva *«mientras»* se genera una nueva lista de soluciones a partir de la anterior. Cada conjunto candidato, formado por la unión de dos conjuntos de la lista anterior de soluciones, formará parte de la próxima lista de soluciones si al aplicarle el evaluador de subconjuntos $\mathcal{L}$ devuelve un valor de medición (Γ) superior al obtenido con el mejor (o primer) subconjunto de la lista anterior de soluciones (T). Para que el algoritmo no tenga un coste temporal prohibitivo, se generan los nuevos conjuntos de atributos uniendo los primeros conjuntos con el resto de la lista de soluciones anterior, es decir, se empieza por el primer conjunto de la lista y se une al segundo, luego el primer conjunto con el tercero, y así hasta el final de la lista, a continuación se unen el segundo conjunto de la lista con el tercero, el segundo y el cuarto, así hasta el último conjunto de la lista. Este proceso de combinar un conjunto de atributos con el resto de conjuntos de la lista se realiza con los k mejores conjuntos de atributos de la lista de soluciones anterior.

Se tiene un parámetro de entrada ϵ que puede limitar el número de nuevos subconjuntos. En aquellas bases de datos donde la dimensionalidad sea extremadamente elevada, o cuando el evaluador de subconjunto sea muy costoso en tiempo, se indica hasta que porcentaje de la lista de soluciones anterior se llega a la hora de generar nuevos subconjuntos de atributos.

En la figura 5.3 muestra un ejemplo del proceso de selección de atributos mediante *BARS* utilizando la correlación no lineal *CFS* como medida de evaluación de subconjuntos ($\mathcal{L}$). El algoritmo se aplica a la base de datos *glass2* (ver tabla A.1, página 184) y la figura representa la evaluación de los subconjuntos de atributos en las diferentes iteraciones del algoritmo. Los números que se pueden ver en el eje de abscisas representan el orden del subconjunto en el ranking correspondiente, y en el eje de ordenadas se refleja la evaluación obtenida para cada subconjunto. Las líneas horizontales establecen el límite al finalizar cada etapa del algoritmo.

El proceso de reducción seguido, para k igual a tres y ϵ igual al 100 % del ranking, es el siguiente:

1. Se genera un ranking inicial de atributos: $[x_1, x_7, x_4]$, $x_5, x_2, x_3, x_6, x_9, x_8$ y el primer límite lo fija la evaluación del atributo x_1 con $\mathcal{L}$ (0,167).

2. Se forman subconjuntos de dos atributos con los tres primeros del ranking anterior ($[x_1, x_7, x_4]$),

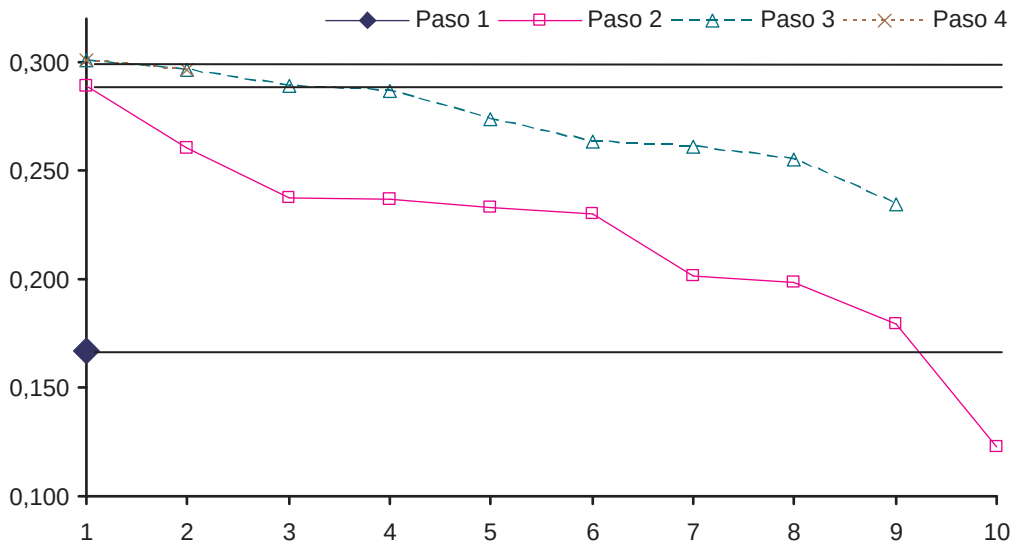


Figura 5.3: Ejemplo del proceso de reducción seguido por *BARS* al aplicarlo a la base de datos *glass2*. Las líneas horizontales representan los límites al finalizar cada fase de la heurística.

y se evalúan con $\mathcal{L}$. Los conjuntos que muestran su evaluación en negrita han sobrepasado el umbral fijado anteriormente con un atributo (0,167).

- Combinaciones con el atributo x_1: $(x_1, x_7 - \mathbf{0{,}261})$, $(x_1, x_4 - \mathbf{0{,}237})$, $(x_1, x_5 - 0,083)$, $(x_1, x_2 - 0,083)$, $(x_1, x_3 - \mathbf{0{,}202})$, $(x_1, x_6 - \mathbf{0{,}179})$, $(x_1, x_9 - 0,083)$, $(x_1, x_8 - 0,083)$
- Con el atributo x_7: $(x_7, x_4 - \mathbf{0{,}289})$, $(x_7, x_5 - 0,123)$, $(x_7, x_2 - 0,123)$, $(x_7, x_3 - \mathbf{0{,}234})$, $(x_7, x_6 - \mathbf{0{,}230})$, $(x_7, x_9 - 0,123)$, $(x_7, x_8 - 0,123)$
- Con el atributo x_4: $(x_4, x_5 - 0,101)$, $(x_4, x_2 - 0,101)$, $(x_4, x_3 - \mathbf{0{,}237})$, $(x_4, x_6 - \mathbf{0{,}198})$, $(x_4, x_9 - 0,101)$, $(x_4, x_8 - 0,101)$

Si se ordenan los subconjuntos que han mejorado al mejor subconjunto anterior queda el siguiente ranking:
[$(x_7, x_4 - 0,289)$, $(x_1, x_7 - 0,261)$, $(x_1, x_4 - 0,237)$**]**, $(x_4, x_3 - 0,237)$, $(x_7, x_3 - 0,234)$, $(x_7, x_6 - 0,230)$, $(x_1, x_3 - 0,202)$, $(x_4, x_6 - 0,198)$, $(x_1, x_6 - 0,179)$

3. Nuevamente, se forman subconjuntos a partir de los tres primeros conjuntos del último ranking generado y se evalúan con $\mathcal{L}$. Aparecerán resaltados aquellos subconjuntos que superen el nuevo límite (0,289), obtenido del conjunto con mejor evaluación hasta el momento, es decir, el primero del ranking anterior. Si en un subconjunto de atributos perteneciente a las próximas listas no aparece evaluación, significa que ese subconjunto ya ha sido evaluado.

- Combinaciones con el conjunto (x_7, x_4): $(x_7, x_4, x_1$ − **0,296**), (x_7, x_4, x_1), $(x_7, x_4, x_3$ − 0,289), (x_7, x_4, x_3), $(x_7, x_4, x_6$−0, 273), $(x_7, x_4, x_1, x_3$−**0,301**), (x_7, x_4, x_6), $(x_7, x_4, x_1, x_6$− 0, 287)
- Con (x_1, x_7): (x_1, x_7, x_4), (x_1, x_7, x_4, x_3), $(x_1, x_7, x_3$−0, 261), $(x_1, x_7, x_6$−0, 255), (x_1, x_7, x_3), (x_1, x_7, x_4, x_6), (x_1, x_7, x_6)
- Con (x_1, x_4): $(x_1, x_4, x_3$−0, 264), (x_1, x_4, x_7, x_3), (x_1, x_4, x_7, x_6), (x_1, x_4, x_3), $(x_1, x_4, x_6$− 0, 234), (x_1, x_4, x_6)

Ordenando los subconjuntos que superan el umbral actual: **[**$(x_7, x_4, x_1, x_3$−0, 301), $(x_7, x_4, x_1$− 0, 296)**]**

4. En el último paso del proceso se generan dos subconjuntos ya evaluados anteriormente (x_7, x_4, x_1, x_3) y (x_7, x_4, x_1), no mejorando el mejor subconjunto hasta el momento. Por tanto el subconjunto seleccionado es (x_7, x_4, x_1, x_3).

5.5.2. Experimentos y resultados

En esta sección se pretende evaluar el método de búsqueda aglomerativo propuesto en términos de exactitud de clasificación, número de atributos seleccionados y eficiencia en la obtención del subconjunto final, para mostrar como *BARS* se comporta en situaciones donde existen gran número de atributos. La comparación se efectuó sobre los dos grupos de bases de datos grandes cuyas propiedades se encuentran resumidas en las tablas A.2 y A.3 (página 185). Igualmente, se seleccionaron *C4.5* (C4), *ib1* (IB) y *Naïve Bayes* (NB), para evaluar la exactitud sobre los atributos seleccionados por cada uno de los algoritmos de selección de atributos. Los experimentos se llevaron a cabo en un Pentium IV a 3 Gh con 512 Mb de RAM, utilizando implementaciones de los algoritmos existente en el entorno WEKA [173], así como el método de búsqueda propuesto también fue implementado en dicho entorno.

Dado que el algoritmo *BARS* sigue el mismo entorno de trabajo que la técnica *BIRS*, se hereda la nomenclatura utilizada en la sección 5.4.2. Se puede considerar la misma figura 5.2 para ilustrar los dos bloques que también componen siempre el algoritmo *BARS* propuesto en los apartados anteriores. Por tanto, este algoritmo de selección igualmente necesita una medida de ranking y de subconjuntos de atributos. Igualmente, se podrían formar numerosas versiones de algoritmos de selección *BARS* combinando los criterios de cada grupo de medidas (individual y de subconjuntos). Siempre que aparezca el algoritmo *BARS* en las tablas comparativas, se expresa con qué criterio se ha generado el ranking, y cómo se ha evaluado la calidad de los subconjuntos de atributos en el proceso de selección. Intentando esclarecer en todo lo posible los

componentes que *BARS* esté utilizando en cada caso, se antepone un subíndice a *BA* que indica el método de ranking empleado y con un superíndice a continuación se señala el SubEvaluador. En los experimentos realizados para *BARS* no se han incluido tantas variaciones como en los realizados para *BIRS*, de hecho, para simplificar, se ha utilizado la misma medida de evaluación para elaborar el ranking que la empleada en la segunda parte del algoritmo en la búsqueda del subconjunto de atributos. Se emplean dos medidas de evaluación de subconjuntos, una por cada tipo de aproximación (*wrapper* y filtro–*CFS*). Por ejemplo, ${}_{CF}BA^{CF}$ indica que se empleará *CFS* como medida individual en la primera parte y *CFS* como evaluador de subconjuntos en la segunda parte.

En las pruebas realizadas, *BARS* generará subconjuntos con los tres mejores subconjuntos de la lista de soluciones que utiliza en cada paso, limitando el recorrido al 50 % de la lista de soluciones en las aproximaciones de tipo *wrapper*, y llegando hasta el final en las filtro. Es decir, k=3 y ϵ=50 % y 100 %, respectivamente.

En los resultados mostrados en las tablas 5.11 y 5.12, se agrupan los resultados por columnas. Se distinguen tres grupos, uno para cada clasificador, de manera que los resultados bajo el título NB que aparece en la primera línea indican que se ha estudiado la clasificación aplicando NB a las bases de datos reducidas con las dos versiones de *BARS* (${}_{NB}BA^{NB}$ y ${}_{CF}BA^{CF}$). Lo mismo ocurre en las columnas encabezadas por C4 e IB, sólo que cambiando de clasificador. Se comparan los resultados obtenidos con las dos versiones de *BARS* con los resultados de los demás algoritmos de selección mostrados en las tablas comparativas de la sección 5.4.2. Por lo tanto, se almacenan la precisión predictiva, el número de atributos seleccionados y el tiempo de ejecución (tabla 5.13). En cada caso, los resultados se obtuvieron de la misma forma que en las comparativas *BIRS* con bases de datos grandes, es decir, calculando la media de cinco ejecuciones de validación cruzada dos (5×2CV), y para que no se produzca un sobreajuste del algoritmo de selección a los datos utilizados, en cada ejecución se realizan dos reducciones, una por cada conjunto de entrenamiento (las Figuras 3.4 y 3.5 en la página 63 son ilustrativas del proceso a realizar con validación cruzada en la selección).

El análisis comparativo de los datos mostrados en la tabla 5.11 para bases de datos grandes del repositorio UCI, se puede realizar desde distintos puntos de vista: entre las diferentes versiones de *BARS*; frente a la heurística anterior propuesta, *BIRS*; con respecto al conjunto original de atributos; y comparándolo con los demás algoritmos de selección aplicados a estas bases de datos.

En general, los resultados obtenidos con los modelos *wrapper* y filtro–*CFS* de *BARS* son similares, aunque los subconjuntos extraídos por los primeros son algo más exactos y más reducidos que los segundos. Los promedios de exactitud y de porcentaje de reducción de atributos

Tabla 5.11: Comparativa de la técnica *BARS*. Resultados obtenidos con los clasificadores NB, C4 e IB sobre conjuntos de datos grandes *UCI*. Ac–tasa de aciertos; %At–porcentaje de atributos.

	NB				C4				IB			
	${}_{NB}BA^{NB}$		${}_{CF}BA^{CF}$		${}_{C4}BA^{C4}$		${}_{CF}BA^{CF}$		${}_{IB}BA^{IB}$		${}_{CF}BA^{CF}$	
Datos	Ac	%At	Ac	%At	Ac	%At	Ac	%At	Ac	%At	Ac	%At
ADS	95,8	0,5	94,6	0,3	96,4	0,5	95,3	0,3	96,2	0,6	95,0	0,3
ARR	68,9	2,3	67,3	4,1	67,9	2,0	66,5	4,1	58,2	2,6	61,4	4,1
HYP	94,9	10,3	94,2	3,4	98,9	10,7	96,6	3,4	83,7	3,4	85,8	3,4
ISO	77,4	3,4	67,0	3,8	68,2	2,7	67,3	3,8	75,1	3,5	72,9	3,8
KRV	94,1	11,4	84,4	7,2	94,1	11,1	84,4	7,2	92,6	11,4	83,5	7,2
LET	55,7	39,4	64,3	56,3	80,5	45,0	84,2	56,3	84,4	42,5	93,6	56,3
MUL	96,8	2,1	96,6	3,9	93,7	1,5	92,8	3,9	97,3	2,3	97,8	3,9
MUS	98,7	7,3	98,5	4,5	99,4	9,1	98,5	4,5	98,1	10,9	98,5	4,5
MUK	84,6	1,0	74,5	7,5	95,7	4,9	94,4	7,5	94,0	4,4	93,0	7,5
SIC	93,9	3,4	93,9	3,4	96,3	7,2	96,3	3,4	95,0	7,6	94,7	3,4
SPL	94,7	15,3	93,6	10,0	92,7	12,2	92,5	10,0	85,5	10,2	86,4	10,0
WAV	80,4	21,3	80,3	35,3	75,9	17,5	76,7	35,3	77,5	24,5	78,9	35,3
Prom.	86,3	9,8	84,1	11,6	88,3	10,4	87,1	11,6	86,5	10,3	86,8	11,6

${}_{-1}BA^{-2}$–*BARS* con la misma medida ranking y SubEvaluador.

mostrados en la última línea de la tabla confirman la afirmación realizada. Con el clasificador NB se producen las mayores diferencias, la versión *wrapper*, con un promedio de porcentaje de reducción mejor (9,8 % frente a 11,6 %) gana significativamente en cuatro ocasiones (ADS, HYP, ISO y SPL) y perdiendo sólo en una (LET). Aplicando el clasificador C4 a los subconjuntos elegidos por ambas versiones de *BARS*, se detectan diferencias significativas en dos conjuntos de datos, HYP y MUS, ganando nuevamente la versión *wrapper* con subconjuntos algo más reducidos (10,4 % frente a 11,6 %). No ocurre lo mismo con el clasificador IB, donde no existen diferencias significativas en las exactitudes, excepto para la base de datos LET que gana la versión *CFS*, e igual que antes, el subconjunto extraído por la versión wrapper es nuevamente menor (10,3 % frente a 11,6 %).

Si se compara la versión wrapper de las dos heurísticas presentadas en este capítulo, *BIRS* (ver tablas 5.2, 5.4 y 5.6, páginas 131, 133 y 134 respectivamente) gana significativamente a *BARS* (tabla 5.11) en dos (ISO y LET), dos (MUS y SIC) y una (LET) base de datos con NB, C4 e IB respectivamente (5 de 36 comparaciones). Sin embargo, los subconjuntos generados por *BARS* contienen menos atributos que los de *BIRS*, concretamente, los promedios de porcentajes de reducción de atributos del algoritmo *BARS* son 9,8 %, 10,4 % y 10,3 % para cada uno de los clasificadores, frente a 15,0 %, 16,3 % y 13,1 % del método *BIRS*. Comparando los resultados

Tabla 5.12: Comparativa de la técnica *BARS*. Resultados obtenidos con los clasificadores NB, C4 e IB sobre grandes conjuntos de datos *NIPS*. *wrapper–CFS*: criterio ranking y del *SubEvaluador*; Orig–Original; Ac–tasa de aciertos; At–número de atributos.

	NB				C4				IB			
	${}_{NB}BA^{NB}$		${}_{CF}BA^{CF}$		${}_{C4}BA^{C4}$		${}_{CF}BA^{CF}$		${}_{IB}BA^{IB}$		${}_{CF}BA^{CF}$	
Datos	Ac	At	Ac	At	Ac	At	Ac	At	Ac	At	Ac	At
ARC	65,4	4,6	66,0	22,5	63,6	2,7	61,6	22,5	62,8	3,5	66,8	22,5
DEX	79,1	15,3	80,7	7,5	78,3	5,8	80,4	7,5	79,6	12,4	79,0	7,5
GIS	91,2	9,2	87,3	8,6	93,0	11,5	89,6	8,6	89,7	6,8	88,1	8,6
MAD	61,0	4,9	60,4	6,3	68,4	4,3	69,3	6,3	78,4	6,1	76,1	6,3
Prom.	76,1		73,8		80,7		79,4		78,4		76,1	

${}_{-1}BA^{-2}$–*BARS* con la misma medida ranking y SubEvaluador.

de las versiones filtro aparecen tan sólo dos casos donde las diferencias son significativas a favor de *BIRS*, aplicando los clasificadores NB e IB a la base de datos ISO. En esta comparación, el promedio de reducción es equivalente, 11,6 % y 11,7 % para *BARS* y *BIRS* respectivamente.

Existen pocos casos donde las tasas de aciertos obtenidas con los subconjuntos seleccionados con la heurística *BARS* sean significativamente diferentes a las obtenidas con el conjunto completo de atributos (ver tablas 5.2, 5.4 y 5.6). De hecho, muestran casi los mismos promedios, sin embargo, *BARS* gana en tres ocasiones (KRV, MUL y MUS) y pierde en una (LET) con el clasificador NB, pierde en tres con C4 (HYP, MUS y SIC), y gana en dos (SPL y WAV) y pierde en una (LET) con IB. Se debe tener en cuenta que la aproximación *wrapper* de *BARS* retiene sólo entorno al 10 % de los atributos del conjunto de datos original, y la aproximación filtro, el 11,6 %.

El algoritmo filtro *FCBF* ofrece prácticamente los mismos resultados de clasificación, sin embargo, necesita casi el doble de atributos que la heurística *BARS*.

La tabla 5.12 sigue la misma estructura que la tabla 5.11, mostrando las tasas de aciertos obtenidas con los tres clasificadores sobre los datos *NIPS* (características en tabla A.3), además, dado que el promedio de reducción de atributos es un valor demasiado bajo, se indica el número medio de atributos seleccionado en cada caso. La última fila de la tabla refleja el promedio de aciertos de las cinco bases de datos. No están disponibles los resultados con la base de datos DOR por problemas de memoria.

El análisis de los resultados obtenidos con *BARS* arroja las siguientes conclusiones:

- No existen diferencias significativas entre las dos aproximaciones de *BARS*, excepto en la base de datos GIS con los clasificadores NB y C4. La diferencia mayor en el número de atributos seleccionados se da en el conjunto de datos ARC, donde la aproximación

wrapper se queda con 4,6, 2,7 y 3,5 atributos para NB, C4 e IB respectivamente, frente a 22,5 de la aproximación filtro.

- En la comparativa entre *BARS* y *BIRS* (tablas 5.12 y 5.9 respectivamente) se puede resaltar la igualdad en las tasas de aciertos obtenidas, tanto con las versiones *wrapper* como con las *CFS*, excepto con la base GIS con el evaluador de subconjuntos *CFS* (87,3 frente a 92,66). En cuanto al número de atributos escogidos para cada base de datos, se puede observar la notable reducción de *BARS* frente a *BIRS*. En la aproximación *wrapper* los subconjuntos seleccionados por *BARS* contienen entre dos y tres veces menos atributos que con *BIRS*, y en la aproximación *CFS* la diferencia es menor, pero favorable a *BARS*, excepto con el conjunto de datos MAD (6,3 frente a 5,8).

- Con respecto a los resultados con las bases de datos originales, lo más importante a resaltar es el bajísimo número de atributos con el que se queda el algoritmo *BARS*. Por ejemplo, la aproximación filtro retiene 22,5, 7,5, 8,6 y 6,3, frente a 10.000, 20.000, 500 y 5.000 atributos que tienen respectivamente cada base de datos. En las tasas de aciertos, las diferencias significativas se encontraron a favor de *BARS* en DEX, MAD y DEX, para NB, C4 e IB respectivamente, y en contra sólo en ARC con IB.

- Al compararlo con al algoritmo filtro *FCBF*, las dos únicas diferencias significativas son favorables a *BARS*, GIS y MAD con C4 e IB respectivamente. Además, estos resultados los consigue *BARS* con bastantes menos atributos que *FCBF*, excepto con el conjunto MAD. Por ejemplo, la aproximación *CFS* se queda con 22,5, 7,5, 8,6 y 6,3, atributos para cada base de datos frente a 35,2, 25,1, 31,2 y 4,7 con el algoritmo *FCBF*.

El tiempo total empleado por el método *BARS* (tabla 5.13) en reducir las bases de datos grandes del *UCI* es superior al del algoritmo *BIRS* (tablas 5.8 y 5.10) en la aproximación *wrapper*, sin embargo, si observamos los datos individualmente, en la mayoría de las ocasiones es más rápido *BARS*, salvo con las bases de datos ISO, MUL y MUK, donde la diferencia es importante. Este comportamiento es debido al incremento en el número de evaluaciones de *BARS*. No ocurre lo mismo con las bases de datos de *NIPS*, y la razón puede ser el escaso número de atributos seleccionado frente a la técnica *BIRS*, por lo que aunque se ejecute la evaluación de un subconjunto más veces, se hace que menos atributos. En las dos aproximaciones, *wrapper* y filtro, los resultados son levemente favorable a *BARS*.

Tabla 5.13: Comparativa de la técnica *BARS*. Tiempo (en segundos) al aplicar esta técnica a grandes conjuntos de datos *UCI* y *NIPS*.

Datos	${}_{NB}BA^{NB}$	${}_{C4}BA^{C4}$	${}_{IB}BA^{IB}$	${}_{CF}BA^{CF}$
ADS	165,1	838,3	17828,6	4,9
ARR	54,4	77,6	83,0	0,4
HYP	7,9	10,3	112,7	0,1
ISO	6761,2	20591,0	11625,2	17,4
KRV	4,0	4,5	254,9	0,0
LET	144,0	651,7	6129,1	1,5
MUL	2359,8	4727,7	11743,2	9,2
MUS	4,5	5,3	654,7	0,1
MUK	52,6	2038,8	22381,2	5,3
SIC	2,3	4,9	143,0	0,1
SPL	11,3	26,0	560,1	0,1
WAV	43,0	319,3	2093,7	0,7
Total 1	9610,2	29295,6	73609,3	39,7
ARC	191,1	267,2	225,0	70,0
DEX	714,1	683,1	1795,1	53,9
GIS	1511,2	5854,8	102847,1	296,7
MAD	76,6	87,9	1528,4	2,5
Total 2	2492,9	6893,0	3548,4	126,5

5.6. Conclusiones

El éxito de muchos esquemas de aprendizaje, en sus intentos para construir modelos de datos, pasa por la habilidad para identificar un subconjunto pequeño de atributos altamente predictivos. La inclusión de atributos irrelevantes, redundantes o con ruido en la fase del proceso de construcción del modelo puede provocar un comportamiento predictivo pobre y un incremento computacional. La aplicación de los métodos de búsqueda más populares en aprendizaje automático a bases de datos con un gran número de atributos puede ser prohibitivo. Sin embargo, en este capítulo, se han presentado dos nuevas técnicas de selección de atributos que permiten utilizar cualquier evaluador de subconjuntos –incluido el modelo de evaluación *wrapper*– para encontrar un buen conjunto de atributos para clasificación. Se utiliza la definición de Utilidad Incremental Ordenada para decidir, al mismo tiempo, si un atributo (o subconjunto) es relevante y no redundante o no (no relevante o redundante). Los métodos, denominados *BIRS* (*Best Incremental Ranked Subset*) y *BARS* (*Best Agglomerative Ranked Subset*), se basan en la idea de relevancia y redundancia, en el sentido de que un atributo (o conjunto) ordenado se escoge si añade información al incluirlo en el subconjunto final de atributos elegidos. Estas heurísticas

conducen a mejoras considerables en la exactitud obtenida, en comparación con el conjunto completo y frente a otros algoritmos de selección de atributos, junto a una notable reducción en la dimensionalidad.

De la aplicación del algoritmo *BIRS* vista en los experimentos de la sección 5.4.2 se pueden extraer las siguientes conclusiones:

- En principio, no se aprecia diferencias interesantes de resaltar al aplicar la búsqueda incremental partiendo de diferentes formas de generar rankings. La aplicación de *BIRS* a bases de datos grandes, hace que con unos subconjuntos de atributos muy reducidos, se obtengan resultados de clasificación iguales o incluso mejores (significativamente en algunos casos) que con las bases de datos originales.

- La precisión obtenida con la versión *wrapper* de *BIRS* y de *SF* es similar, sin embargo, *BIRS* extrae subconjuntos algo más reducidos y en un tiempo mucho menor. Además, *BIRS* se puede aplicar a bases de datos de cualquier tamaño, mientras que *SF* no.

- La versión *wrapper* de *BIRS* gana en precisión a todas las versiones filtro-*CFS* (*BIRS*, *SF* y *FCBF*), aunque para ello emplea un número mayor de atributos (excepto *FCBF*) y mucho más tiempo. Se puede entender que el tiempo obtenido con las distintas aproximaciones de tipo filtro es insignificante comparado con el calculado a partir de las aproximaciones *wrapper*.

- Para finalizar, resaltar los buenos resultados obtenidos con la aproximación *CFS* de *BIRS*, siendo las exactitudes similares a las de *SF* y algo por encima de las obtenidas con *FCBF*, sin embargo, los conjuntos más reducidos se obtienen con las versiones filtro de *BIRS*, y con un coste inferior a los demás filtros.

De la aplicación del algoritmo *BARS* analizada en los experimentos de la sección 5.5.2 se extraen algunas conclusiones más:

- Los resultados obtenidos con las aproximaciones de tipo *wrapper* y filtro del algoritmo *BARS* son muy parecidos, quizás, la versión *wrapper* ofrece algo más de exactitud con unos subconjuntos de atributos más pequeños.

- De la comparativa entre las dos versiones propuestas en el presente capítulo, *BIRS* y *BARS*, se puede extraer un balance ligeramente superior al primero en cuanto a exactitud (gana en cinco ocasiones de treinta y seis comparaciones), sin embargo, la reducción practicada en los conjuntos de datos es más favorable a *BARS* en su versión *wrapper*.

Se consigue una reducción drástica de los conjuntos originales, ofreciendo exactitudes equivalentes.

- Y en general, en comparación con el filtro *FCBF*, *BARS* ofrece unos resultados similares en precisión, pero con aproximadamente la mitad de los atributos, eso sí, necesitando mucho más tiempo de ejecución.

Como resultado de esta parte de la investigación, se cumple con otro de los objetivos importantes de esta tesis, poder aplicar una técnica de búsqueda en el espacio formado por los atributos de cualquier base de datos, ofreciendo la oportunidad de utilizar un clasificador (wrapper) como medida de evaluación de subconjuntos. Los resultados obtenidos mediante la aplicación de *BIRS* a diversos grupos de bases de datos, fueron publicados en [142, 149, 150].

Capítulo 6

Extracción de genes relevantes en bases de datos genómicas

La tecnología basada en microarray está fomentando el desarrollo de nuevos algoritmos de selección de atributos que, originalmente, se habían diseñado para bases de datos provenientes de otras fuentes. Ahora, con varios miles de genes, la extracción del mejor subconjunto de genes relevantes es un proceso computacionalmente muy costoso. La importancia de estos resultados es notoria cuando se trata de realizar diagnósticos de enfermedades graves, tales como el cáncer. En este capítulo se aplican las técnicas basadas en un ranking preliminar, presentadas en el capítulo anterior (denominados *BIRS–Best Incremental Ranked Subset* y *BARS–Best Agglomerative Ranked Subset*), para la selección de genes relevantes. En los experimentos se han utilizado varias bases de datos reales, obteniendo resultados de clasificación mejores con los genes seleccionados que con todo el conjunto inicial.

El capítulo se organiza de la siguiente forma: en la siguiente sección se realiza una introducción a la bioinformática, describiendo algunos de los conceptos básicos; en la sección 6.2 se describe el origen de las bases de datos genómicas y sus posibles aplicaciones; en la sección 6.4 se describen brevemente otros trabajos relacionados con la selección de genes; en la sección 6.5, se aplican los algoritmos *BIRS* y *BARS* a bases de datos genómicas. Finalmente, en la sección 6.6 se recogen las conclusiones más interesantes.

6.1. Introducción

En los últimos años se ha producido un importante crecimiento de las bases de datos en todas las áreas del conocimiento humano, de las que podríamos destacar especialmente las biológicas. Este incremento es debido, principalmente, al progreso en las tecnologías para la adquisición

de datos y, en particular, los relacionados con el genoma de diferentes organismos.

La *bioinformática* es una disciplina científica emergente que utiliza tecnología de la información para organizar, analizar y distribuir información biológica con la finalidad de responder preguntas complejas en biología. La bioinformática es un área de investigación multidisciplinaria impulsada por la incógnita del genoma humano y la promesa de una nueva era en la cual la investigación genómica puede ayudar espectacularmente a mejorar la condición y calidad de la vida humana. Los avances en la detección y tratamiento de enfermedades es uno de los beneficios mencionados más frecuentemente. Por lo tanto, la informática se utiliza desde tres perspectivas: como herramienta tecnológica que agiliza los procedimientos de adquisición de datos; como soporte para la gestión y organización de tales cantidades ingentes de información; y como metodología para el diseño de aplicaciones capaces de gestionar los datos, solucionar problemas complejos y extraer conocimiento útil a partir de la información. En el primer caso podríamos hablar de tecnología informática; en el segundo caso, de sistemas de gestión de bases de datos; y, en el tercero, de ingeniería del software y minería de datos.

Según la definición del Centro Nacional para la Información Biotecnológica (*National Center for Biotechnology Information–NCBI*): «*Bioinformática* es un campo de la ciencia en el cual confluyen varias disciplinas tales como: biología, computación y tecnología de la información. El fin último de este campo es facilitar el descubrimiento de nuevas ideas biológicas así como crear perspectivas globales a partir de las cuales se puedan discernir principios unificadores en biología».

Al comienzo de la «revolución genómica» el concepto de bioinformática se refería sólo a la creación y mantenimiento de bases de datos donde almacenar información biológica, como secuencias de nucleótidos y aminoácidos. El desarrollo de este tipo de bases de datos no sólo implicaba el diseño de la misma, sino también el desarrollo de interfaces complejas donde los investigadores pudieran acceder a los datos existentes y suministrar o revisar datos. Luego, toda esa información debía ser combinada para formar una idea lógica de las actividades celulares normales, de tal manera que los investigadores pudieran estudiar cómo estas actividades se veían alteradas en estados de una enfermedad. A partir de este origen, la bioinformática ha ido evolucionando y ahora es el campo más popular en el análisis e interpretación de varios tipos de datos, incluyendo secuencias de nucleótidos y aminoácidos, dominios de proteínas y estructura de proteínas. El proceso de analizar e interpretar los datos es conocido como *biocomputación*. Dentro de la bioinformática y la biocomputación existen otras sub-disciplinas importantes: El desarrollo e implementación de herramientas que permitan el acceso, uso y manejo de varios tipos de información. El desarrollo de nuevos algoritmos y estadísticos con los cuales se pueda relacionar partes de un conjunto enorme de datos, como por ejemplo métodos para localizar

un gen dentro de una secuencia, predecir estructuras o funciones de proteínas y poder agrupar secuencias de proteínas en familias relacionadas.

Las tecnologías de la información jugarán un papel fundamental en la aplicación de los desarrollos tecnológicos en el campo de la genética a la práctica médica, como refleja la presencia de la bioinformática médica y la Telemedicina dentro de las principales líneas en patología molecular. La aplicación de los conocimientos en genética molecular y las nuevas tecnologías son necesarios para el mantenimiento de la competitividad del sistema sanitario no sólo paliativo sino preventivo. La identificación de las causas moleculares de las enfermedades, junto con el desarrollo de la industria biotecnológica en general y de la farmacéutica en particular, permitirán el desarrollo de mejores métodos de diagnóstico, la identificación de dianas terapéuticas, el desarrollo de fármacos personalizados y una mejor medicina preventiva.

Nuestro trabajo de investigación se centra en el análisis de datos de expresión genética desde la perspectiva de la minería de datos. La expresión de un gen se obtiene cuando una porción de ADN se transcribe para crear una molécula de ARN como primer paso en la síntesis de proteínas. Del numeroso grupo de genes que se encuentran en el núcleo y que codifican proteínas, sólo un subconjunto de ellos se expresará en un momento determinado. Esta expresión selectiva viene regulada por distintos aspectos: tipo de célula, fase de desarrollo del ser vivo, estimulo interno o externo, enfermedades, etc. Estos datos, por simplicidad, suelen representarse en tablas (o vistas) formadas por un número finito de genes (columnas) y un número finito de condiciones (filas). En suma, cada fila representa diferentes condiciones a las que se somete un gen y, cada columna, la respuesta que un gen experimenta ante una determinada condición (temperatura, excitación, etc.). Estos experimentos, por ejemplo, pueden determinar si un gen se expresa o no ante una determinada condición, que puede estar relacionada con la presencia o ausencia de un tipo de cáncer. De esta forma, podemos estudiar qué genes pueden ayudarnos a pronosticar si pacientes aún no tratados son propensos a desarrollar la enfermedad.

La información almacenada en microarrays suele ser numerosa respecto del número de genes, es decir, son miles de genes los que se estudian, aunque ciertamente sólo unos pocos tendrán relación directa con la enfermedad en cuestión. Estos genes más relevantes, posteriormente, son estudiados con mayor detalle por el biólogo/médico, que intentará determinar el comportamiento específico, el proceso regulador, o los biomarcadores involucrados en el desarrollo de una enfermedad. Por lo tanto, es de especial interés la extracción de genes relevantes para un proceso biológico específico.

Para la extracción de genes, desde la perspectiva de la minería de datos, se aplican técnicas de selección de atributos, para cuya tarea se han desarrollado algoritmos válidos que son capaces de abordar problemas que involucran a varias decenas de atributos. No obstante, ahora nos

enfrentamos a varios millares de genes, por lo que dichos algoritmos no son tan efectivos. Esta razón ha promovido en los últimos años el desarrollo de algunas técnicas específicas de extracción de genes relevantes.

En este capítulo abordamos el problema de la extracción de genes relevantes utilizando las técnicas novedosas descritas en el capítulo anterior, basadas en un ranking preliminar. Además, se describen y discuten los experimentos realizados con bases de datos reales, mostrando en todos los casos un rendimiento muy satisfactorio con respecto a la calidad de los genes extraídos. Esta calidad se cuantifica con el porcentaje de clasificación correcta que estos genes, por sí solos, generan.

6.2. Bases de datos genómicas

A finales de los años ochenta los científicos, Stephen Fodor, Michael Pirrung, Leighton Read y Lubert Stryer, desarrollaron una revolucionaria tecnología para la determinación y cuantificación de ADN en una muestra. Esta tecnología desembocaría posteriormente en la primera plataforma de microarrays de ADN, denominada entonces *GeneChip* de *Affymetrix*. La principal ventaja de esta novedosa tecnología frente a los métodos tradicionales residía en la alta integración (cantidad) de material biológico que se consigue inmovilizar, es decir, la posibilidad de analizar simultáneamente miles de genes.

Los microarrays o biochips se definen como una matriz bidimensional de material genético, que permite la automatización simultánea de miles de ensayos encaminados a conocer en profundidad la estructura y funcionamiento de nuestra dotación genética, tanto en los distintos estados de desarrollo como patológicos del paciente. Normalmente consiste en una fina capa de cristal o nilón donde un dispositivo robótico imprime muestras de algunos genes específicos. De esta forma, se mide la abundancia relativa de *m-RNA* entre dos muestras, las cuales son etiquetadas con colorantes fluorescentes rojo y verde (ver figura 6.1). La abundancia relativa de la mancha o gen se mide como la tasa logarítmica entre las intensidades de los colorantes y constituye el dato de expresión del gen.

La nomenclatura empleada para referirse a estas nuevas tecnologías es diversa y comienza por el término más general que es el de *Biochip* y hace referencia al empleo de materiales biológicos sobre un chip. Otros términos más específicos son: *DNA chip*, *RNA chip* (según el material empleado) y *Oligonucleotide chip* o *DNA Microarray*, que hacen referencia al material y a la forma en la que se construye el chip. Existen también unos términos comerciales con los que referirse a los biochips que varían dependiendo de la tecnología empleada.

La minería de datos es una de las técnicas empleadas para analizar la información genética.

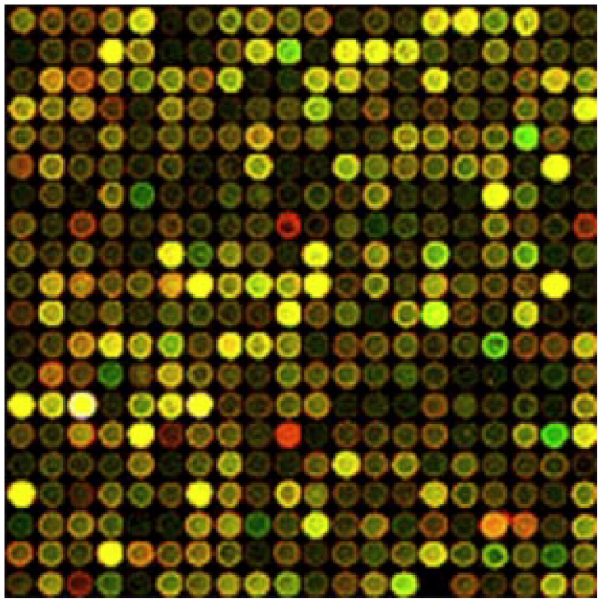


Figura 6.1: Ejemplo de microarray.

Partiendo de bases de datos de expresión genética, se pueden aplicar una amplia variedad de algoritmos: clasificación, regresión, selección de atributos (en este caso se denomina selección de genes), clustering, detección de outliers, análisis de componentes principales, descubrimiento de relaciones entre los genes, etc.

A pesar de ser una tecnología muy reciente y que, por lo tanto, está aún en vías de experimentación, actualmente los microarrays están siendo aplicados en:

- Monitorización de expresión genética: permite determinar cuál es el patrón de expresión genética y cuantificar el nivel de expresión de manera simultánea para un elevado número de genes. Esto permite realizar estudios comparativos de activación de determinados genes en tejidos sanos y enfermos y determinar así la función de los mismos.

- Detección de mutaciones y polimorfismos: Permite el estudio de todos los posibles polimorfismos y la detección de mutaciones en genes complejos.

- Secuenciación: habiéndose diseñado algunos biochips para secuenciación de fragmentos cortos de ADN, no existe aún en el mercado ningún biochip que permita secuenciar de nuevo secuencias largas de ADN.

- Diagnóstico clínico y detección de microorganismos: posibilitan la identificación rápida empleando unos marcadores genéticos de los patógenos.

- *Screening* y toxicología de fármacos: El empleo de los biochips permite analizar los cambios de expresión génica que se dan durante la administración de un fármaco de forma rápida, así como la localización de nuevas posibles dianas terapéuticas y los efectos toxicológicos asociados.

- Seguimiento de terapia: los biochips permiten valorar rasgos genéticos que pueden tener incidencia en la respuesta a una terapia.

- Medicina preventiva: el conocimiento y posible diagnóstico de ciertos caracteres genéticos asociados a determinadas patologías permiten la prevención de las mismas antes de que aparezcan los síntomas.

6.3. Aprendizaje supervisado en bioinformática

En aprendizaje automático existen algunos aspectos sobre la calidad de las bases de datos que se deben tener en cuenta durante el proceso de la construcción del modelo de aprendizaje, y que se vulneran con facilidad en este tipo de aplicaciones. En primer lugar, la ausencia de valores, por lo que se debe plantear el uso de métodos de preprocesado para estimar esta falta de información. En segundo lugar, la posibilidad de sesgo de los datos debido a diversas razones, tales como las tecnológicas, aspectos del entorno, o factores humanos. Finalmente, es de esperar la presencia de ruido en este tipo de datos.

En las bases de datos provenientes de microarrays, los genes son los atributos y las muestras biológicas a estudiar son los ejemplos o condiciones. Lo normal en estas bases de datos es tener un número reducidos de ejemplos (menos de cien), y un elevado número de genes (varios miles). Este bajo número de muestras hace probable el sobreajuste del modelo de predicción a los datos disponibles. Esto es debido a que el espacio de búsqueda no está uniformemente representado por los datos.

En la parte final del proceso de minería de datos, llamada validación, no están claras las técnicas que pueden o no aplicarse a este tipo de dato. Entre las más utilizadas está la técnica tradicional de reservar una parte de los datos para prueba (*hold out*), la validación cruzada con k carpetas, siendo frecuente el caso especial de validación cruzada dejando uno fuera (*leave–one–out*), o también se utilizan métodos de *boostrap*. Sea cual sea la técnica utilizada, es esencial para comprobar el comportamiento de los algoritmos de selección estimar la bondad del clasificador con un conjunto de muestras independientes, es decir, que no hayan formado parte en el proceso de selección.

La tarea de la clasificación se ha investigado intensivamente y se han realizado numerosas

propuestas en las últimas dos décadas. En la actualidad, una de las principales aplicaciones en bioinformática es la de clasificar datos de expresión de genes o proteínas. El hecho de que en los datos de expresión genética se dé un elevado número de genes ligado a un tamaño pequeño de muestra puede causar el problema conocido como «curso de la dimensionalidad», sobreajustando los datos de entrenamiento [52]. Esto se traduce en la necesidad de disponer de una enorme cantidad de observaciones para obtener una buena estimación de una función de los genes. Por tanto, seleccionar un número pequeño de genes discriminativos de entre miles de ellos es esencial para el éxito en la clasificación de muestras [162, 175]. Este es el motivo principal por el que la selección de genes para clasificación de muestras está en auge en el estudio sobre la selección de atributos.

Con este tipo de datos se pueden encontrar aplicaciones de los algoritmos de aprendizaje supervisado más populares. Además de las técnicas estadísticas clásicas (por ejemplo, análisis discriminante, clasificadores gaussianos y logísticos, etc.) [66], el componente principal de esta clase de aplicaciones incluyen *Naïve Bayes*[80, 185], vecinos más cercanos [80, 65, 185], árboles de decisión [80, 185], redes neuronales [18], algoritmos genéticos [41, 42], máquinas de soporte vectorial [102, 69] y *rough sets* [116]. En principio, no existe un algoritmo que sea aceptado mejor que el resto.

6.4. Trabajos relacionados

Como se ha señalado, lo normal en las bases de datos de procedentes de experimentos con microarrays es tener un número reducidos de ejemplos (menos de cien), y un elevado número de genes (varios miles), donde muchos de ellos pueden ser redundantes o irrelevantes. Teniendo en cuenta tales genes durante la clasificación, se añaden dificultades computacionales y se introduce ruido innecesario al proceso. Otro aspecto importante cuando se tiene un elevado número de genes es la interpretabilidad de los resultados. En muchas aplicaciones de análisis funcional sería deseable reconocer y entender el significado biológico de un conjunto reducido de genes, por ejemplo para obtener fármacos específicos. Sin embargo, el elevado número de genes hace difícil (o prohibitivo) buscar subconjuntos de genes óptimos.

Los métodos de selección de genes tradicionales frecuentemente eligen los primeros genes de una lista ordenada según su poder predictivo. Esta aproximación es eficiente para bases de datos con un elevado número de atributos debido al coste temporal lineal en términos de dimensionalidad. Sin embargo éstos sólo pueden captar la relevancia de los genes con respecto al concepto objetivo o clase, y no descubren redundancia e interacciones básicas entre genes. Como ya se expuso en el capítulo anterior, los métodos de búsqueda de subconjuntos más

populares en aprendizaje supervisado no se pueden aplicar a este tipo de bases de datos, y se hace uso de un nuevo marco de trabajo híbrido donde se combinan técnicas de rankings y de selección de subconjuntos de atributos.

Entre las medidas propuestas para ordenar los genes por su relevancia en el campo de los microarrays, están: PS (*Prediction Strength*) propuesta por Golub [65]; TNoM (*Threshold Number of Misclassification*) Ben-Dor et al. [17]; ganancia de información [175]; t-score [177]; y LDA (*Linear Discriminant Analysis*), LR (*Logistic Regression*) y SVM (*Support Vector Machine*) en [176].

Algunas aplicaciones emplean métodos para eliminar redundancia en el conjunto de genes seleccionado, por ejemplo, genes que se comporten de manera muy parecida y proporcionen la misma información [175, 185]. Estos métodos se basan en el concepto de *Markov blanket*, según el cual, se eliminan gradualmente los atributos redundantes con respecto a un número prefijado de genes (*Markov blanket*) de una lista ordenada de genes. En este contexto, RFE (*Recursive Feature Elimination*) [69], descrito brevemente en el apartado 5.2, selecciona un conjunto de genes eliminando de manera consecutiva aquellos que aportan menos a la clasificación. En [179] se muestra una aplicación multivariante de selección de atributos de RFE. Las SVM´s son particularmente efectivas en manejar grandes espacios de atributos e identificar outliers, lo que les permite modelar eficientemente datos de alta dimensionalidad.

Los algoritmos genéticos han sido bien utilizados para clasificar bases de datos de expresión con sólo unos cuantos genes [43]. La identificación de subconjuntos de genes para clasificar datos con dos clases también ha sido modelado como un problema de optimización multiobjetivo [42, 41]. La teoría del *rough set* también ha sido utilizada para generar subconjuntos de atributos que definen la misma partición que la base de datos original [12].

Además de los algoritmos de selección anteriores, en la bibliografía especializada se pueden encontrar otros procedimientos para reducir la dimensionalidad de los datos de expresión genética, tales como el análisis de componentes principales o los mínimos cuadrados parciales [124]. Estas aproximaciones, consideradas como procesos de extracción de características, combinan información de atributos/genes individuales en componentes. y definen cada una de las muestras en base a las nuevas componentes en vez de los atributos originales individuales.

6.5. Evaluaciones experimentales

En esta sección se pretende analizar el comportamiento de las algoritmos *BIRS* y *BARS* ante conjuntos de datos de gran tamaño, formados por la expresión genómica extraída de los microarrays de ADN. La finalidad es evaluar las técnicas en términos de exactitud en la clasi-

ficación, velocidad y grado de reducción. Para ello, se comparan los resultados de clasificación obtenidos con las bases de datos originales y los obtenidos con las bases de datos reducidas.

6.5.1. Descripción de las bases de datos

Para analizar el comportamiento del método de selección de genes, se han elegido bases de datos muy conocidas relacionadas con el cáncer:

Colon Esta base de datos [10] está formada por 62 muestras (ejemplos) de células epiteliales de *colon*. Cada muestra cuenta con 2000 genes (atributos). Las muestras se recogieron de pacientes con cáncer de colon, y la variable clase puede tomar dos valores según el resultado de la biopsia practicada: 22 muestras recogidas de partes del colon con tumores y 40 muestras recogidas de partes sanas del colon de los mismos pacientes. La asignación final del estado de la biopsia de las muestras se hizo por examinación patológica. Los niveles de expresión del gen en las 62 muestras se midieron utilizando arrays de oligonucleótidos de alta densidad. De los $\approx$ 6000 genes representados en el array, se seleccionaron 2000 genes basándose en la confianza de los niveles de expresión medidos.

Leukemia Esta base de datos [65] contiene 72 pacientes con esta enfermedad. Estas muestras se dividen en dos variantes de leucemia: 25 muestras de *acute myeloid leukemia* (AML) y 47 muestras de *acute lymphoblastic leukemia* (ALL). Las mediciones de expresión de los genes fueron tomadas de 63 muestras de médula ósea y 9 muestras de sangre periférica. Los niveles de expresión del gen en las 72 muestras se midieron utilizando arrays de oligonucleótidos de alta densidad. Se presentan los niveles de expresión de 7129 genes.

Lymphoma Esta base de datos [6] está formada por 96 muestras descritas por los valores de expresión de 4026 genes. Existen en este conjunto de datos nueve subtipos diferentes de linfomas.

GCM (Global Cancer Map) Esta base de datos [135] está compuesta por 190 muestras y 16063 genes analizados. Clasifica las muestras en 14 tipos diferentes según el cáncer diagnosticado.

Las dos primeras aparecen en numerosos estudios, sin embargo, no ocurre lo mismo con las dos últimas, *lymphoma* y *GCM*, quizás por la dificultades de clasificación que plantean.

6.5.2. Descripción de los experimentos

Como se comentó en el capítulo anterior, en la primera fase de los algoritmos se elabora una lista con todos los genes ordenados según su relevancia de acuerdo con el criterio elegido, y en una segunda fase se realiza una búsqueda sobre el ranking según el modelo envoltorio (wrapper) o filtro escogido. Para demostrar la validez de las técnicas, las listas ordenadas de atributos se generan con dos criterios diferentes, SOAP (SP) y la medida de evaluación de subconjunto que se va a utilizar en la segunda fase del algoritmo. Como criterio de evaluación de subconjuntos, se utilizó el propio clasificador destino y una medida de correlación (utilizada por el algoritmo *CFS–Correlation–based Feature Selection* [70]) como aproximación *wrapper* y filtro respectivamente.

Debido a la alta dimensionalidad de los datos se limita la comparativa a técnicas secuenciales, en concreto a la búsqueda secuencial hacia adelante (*SF–Sequential Forward*) en combinación con las medidas de evaluación de subconjuntos, y al algoritmo *FCBF* (*Fast Correlation–Based Filter* [184]).

Los métodos de selección de genes, así como los clasificadores, se han implementado utilizando el entorno Weka [173], ejecutándose en un Pentium IV a 3 Gh con 512 Mb de RAM.

Para comparar los diferentes algoritmos de selección se aplicaron los tres clasificadores, C4, IB y NB (ver sección 2.3 en la página 14), sobre las bases de datos originales, así como sobre las recién obtenidas, conteniendo sólo los genes seleccionados por cada uno de los algoritmos de selección, almacenando la precesión predictiva, el número de atributos seleccionados y el tiempo de ejecución. En cada caso, los resultados se obtuvieron calculando la media de una ejecución de validación cruzada diez (1×10CV), y para que no se produzca un sobreajuste del algoritmo de selección a los datos utilizados, se realizan diez reducciones, una por cada conjunto de entrenamiento (las Figuras 3.4 y 3.5 en la página 63 son ilustrativas del proceso a realizar con validación cruzada en la selección).

6.5.3. Análisis de los resultados

La tabla 6.1 muestra las tasas de aciertos y el número de genes seleccionado obtenidos con las cuatro bases de datos descritas anteriormente. La tabla se encuentra dividida en distintos grupos, en horizontal por los clasificadores NB, C4 e IB, y en vertical por cada conjunto de datos tratado. Dentro de cada grupo horizontal (por clasificador), se encuentran los resultados de todos los algoritmos de selección participantes en la comparativa y los obtenidos con el conjunto original de los datos (última línea de cada grupo). Excepto del algoritmo filtro *FCBF–Fast Correlation-Based Filter* (penúltima línea de cada grupo), por cada técnica de búsqueda se

tiene una aproximación *wrapper* y una filtro, según la medida de evaluación de subconjuntos utilizada sea el propio clasificador o *CFS*. Los métodos utilizados son los dos presentados en esta tesis en el capítulo anterior y la búsqueda secuencial hacia adelante (*SF*). La manera de nombrar los algoritmos será la usada en el capítulo anterior. Para cada modelo, *wrapper* o *CFS*, *BIRS* y *BARS* contemplan dos versiones de algoritmos, según sea generado el ranking inicial con la propia medida de evaluación de subconjuntos (${}_{WR}BI^{WR}$, ${}_{CF}BI^{CF}$, ${}_{WR}BA^{WR}$ y ${}_{CF}BA^{CF}$) o con el criterio SP presentado en el capítulo 4 (${}_{SP}BI^{WR}$ y ${}_{SP}BA^{CF}$), englobando WR a los tres clasificadores, NB, C4 e IB.

Los símbolos $\overset{\circ}{}$, $^{\circ}$ y $\overset{\bullet}{}$, $^{\bullet}$, respectivamente identifican si la diferencia frente al conjunto de datos original es estadísticamente mejor o peor, al nivel de confianza 0,05 o 0,1. Al utilizar medidas de tipo filtro para generar el ranking y evaluar los subconjuntos de atributos (grupos *CFS* y algoritmo *FCBF*), el conjunto final seleccionado no dependerá del clasificador. Por eso, en la tabla 6.1 aparecen los mismos números de genes para este tipo de algoritmos. Por ejemplo, el número medio de genes seleccionados con el algoritmo ${}_{CF}BA^{CF}$ con el clasificador NB, C4 e IB es 7,3. Se muestran repetidos por facilidad y claridad en la interpretación de los resultados. En estos experimentos, los parámetros prefijados para los algoritmos *BIRS* y *BARS* (apartados 5.4.2 y 5.5.2) son: para *BIRS* un umbral de 0,005, y un nivel de confianza en el t-test de 0,1, según sea la aproximación filtro o *wrapper*; mientras que *BARS* genera los subconjuntos con los tres mejores subconjuntos de la lista de soluciones que utiliza en cada paso, limitando el recorrido al 50 % de la lista en ambas aproximaciones. En la tabla 6.2 se muestra el porcentaje retenido del subconjunto de genes seleccionado con respecto al conjunto de datos original (teniendo en cuenta que los datos mostrados para las aproximaciones *wrappers* son los promedios sobre los tres clasificadores). Finalmente, en la tabla 6.3 se puede comprobar el tiempo empleado por los diferentes algoritmos, mostrando tres resultados diferentes para cada aproximación *wrapper*, dependiendo del algoritmo de aprendizaje utilizado, y obviamente, uno para cada aproximación filtro.

El análisis comparativo de los datos mostrados en la tabla 6.1 se puede realizar desde distintos puntos de vista: entre las diferentes versiones de *BIRS* y *BARS*; con respecto al conjunto original de atributos; y comparándolo con los demás algoritmos de selección aplicados a estas bases de datos.

BIRS versus *BARS*

En primer lugar se comparan los resultados obtenidos con las diferentes versiones de las técnicas propuestas en el capítulo anterior, *BIRS* y *BARS*. En la tabla 6.1 se observa el excelente resultado conseguido con todas las aproximaciones realizadas para estos métodos. Se destaca

Tabla 6.1: Aplicación de las técnicas *BIRS* y *BARS* a datos *genómicos*. Comparativa con los clasificadores NB, C4 e IB.

		colon		*leukemia*		*lymphoma*		*GCM*		
	Alg.	Ac	#g	Ac	#g	Ac	#g	Ac	#g	
wrapper	${}_{NB}BI^{NB}$	85,48$^{\circ}_{\circ}$	3,5	93,04	2,5	82,44	10,3	67,37	44,0	Clasif. NB
	${}_{SP}BI^{NB}$	85,48$^{\circ}_{\circ}$	7,4	93,04	2,8	82,22	10,1	65,26	40,9	
	${}_{NB}BA^{NB}$	83,81$^{\circ}_{\circ}$	4,0	91,43	3,0	83,67	7,7	62,11	19,3	
	${}_{SP}BA^{NB}$	83,81$^{\circ}_{\circ}$	4,0	94,46	2,7	84,44°	7,0	63,16	20,0	
	SF^{NB}	84,05$^{\circ}_{\circ}$	5,9	87,32$^{\bullet}$	3,2	83,56°	7,1	N/D		
CFS	${}_{CF}BI^{CF}$	80,95$^{\circ}_{\circ}$	15,8	94,46	6,7	84,33	57,6	70,53	56,1	
	${}_{SP}BI^{CF}$	80,95$^{\circ}_{\circ}$	22,1	94,46	7,3	83,33	67,5	66,84	65,7	
	${}_{CF}BA^{CF}$	79,05°	7,3	93,04	5,6	85,56	30,5	62,11	32,1	
	${}_{SP}BA^{CF}$	77,86$^{\circ}_{\circ}$	12,5	94,04	5,7	83,56°	35,1	62,63	32,2	
	SF^{CF}	82,62$^{\circ}_{\circ}$	22,1	91,43	40,3	75,11	153,2	N/D		
FCBF		77,62$^{\circ}_{\circ}$	14,6	95,89	45,8	78,22	290,9	68,95	60,9	
Orig		53,33		98,57		75,11		65,79		
wrapper	${}_{C4}BI^{C4}$	83,81	2,9	88,57	1,2	80,00	8,8	46,84$^{\bullet}$	9,8	Clasif. C4
	${}_{SP}BI^{C4}$	83,81	2,9	84,64	1,1	81,33	6,6	44,21$^{\bullet}$	24,8	
	${}_{C4}BA^{C4}$	82,14	2,5	83,21	1,9	68,78	5,2	50,00$^{\bullet}$	9,1	
	${}_{SP}BA^{C4}$	75,48	4,2	87,50	1,9	76,33	6,2	46,84$^{\bullet}_{\bullet}$	7,5	
	SF^{C4}	80,71	3,3	87,32	1,6	73,00	8,2	N/D		
CFS	${}_{CF}BI^{CF}$	85,24	15,8	85,89	6,7	85,56	57,6	52,63	56,1	
	${}_{SP}BI^{CF}$	86,67	22,1	83,21	7,3	74,22	67,5	54,21	65,7	
	${}_{CF}BA^{CF}$	83,57	7,3	87,32	5,6	80,33	30,5	53,68	32,1	
	${}_{SP}BA^{CF}$	81,90	12,5	87,32	5,7	79,33	35,1	50,00$^{\bullet}$	32,2	
	SF^{CF}	86,90	22,1	84,82	40,3	79,22	153,2	N/D		
FCBF		88,33	14,6	83,21	45,8	78,22	290,9	52,63	60,9	
Orig		82,14		82,14		81,44		60,00		
wrapper	${}_{IB}BI^{IB}$	79,76	6,3	93,04	3,3	85,56	16,1	58,95	37,0	Clasif. IB
	${}_{SP}BI^{IB}$	79,05	7,6	88,75	4,2	83,44	19,6	60,00	33,1	
	${}_{IB}BA^{IB}$	78,33	4,2	88,93	2,5	81,22	8,4	55,26	13,8	
	${}_{SP}BA^{IB}$	77,38	4,3	93,21	2,3	85,44	9,5	54,21	13,4	
	SF^{IB}	66,67	4,8	88,93	2,3	80,11	8,4	N/D		
CFS	${}_{CF}BI^{CF}$	80,48	15,8	88,75	6,7	90,78	57,6	64,74	56,1	
	${}_{SP}BI^{CF}$	78,10	22,1	88,75	7,3	93,67	67,5	61,05	65,7	
	${}_{CF}BA^{CF}$	80,48	7,3	87,32	5,6	89,44	30,5	57,89	32,1	
	${}_{SP}BA^{CF}$	79,05	12,5	85,89	5,7	87,44	35,1	57,89	32,2	
	SF^{CF}	80,71	22,1	90,18	40,3	92,78	153,2	N/D		
FCBF		80,71	14,6	94,46	45,8	91,89	290,9	61,05	60,9	
Orig		77,62		86,25		84,33		57,37		

wrapper–CFS: medida del *SubEvaluador*. Ac–tasa de aciertos; #g–número de genes. ${}_{-1}BI^{-2}$–*BIRS*; ${}_{-1}BA^{-2}$–*BARS*; medida 1 ranking y 2 SubEvaluador; *SF*–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*; Orig–Original; $\circ$ y $\bullet$, mejora o empeora significativamente con respecto al original. N/D–No Disponible.

Tabla 6.2: Aplicación de las técnicas *BIRS* y *BARS* a datos *genómicos*. Porcentaje de atributos seleccionados respecto al conjunto original.

		colon	*leukemia*	*lymphoma*	*GCM*	Promedio
wrapper	${}_{WR}BI^{WR}$	0,21	0,03	0,29	0,19	0,18
	${}_{SP}BI^{WR}$	0,30	0,04	0,30	0,21	0,21
	${}_{WR}BA^{WR}$	0,18	0,03	0,18	0,09	0,12
	${}_{SP}BA^{WR}$	0,21	0,03	0,19	0,08	0,13
	SF^{WR}	0,23	0,03	0,20	N/D	0,15*
CFS	${}_{CF}BI^{CF}$	0,79	0,09	1,43	0,35	0,67
	${}_{SP}BI^{CF}$	1,11	0,10	1,68	0,41	0,82
	${}_{CF}BA^{CF}$	0,37	0,08	0,76	0,20	0,35
	${}_{SP}BA^{CF}$	0,63	0,08	0,87	0,13	0,43
	SF^{CF}	1,11	0,57	3,81	N/D	1,83*
	FCBF	0,73	0,64	7,23	0,38	2,24

wrapper–CFS: medida del *SubEvaluador*; ${}_{-1}BI^{-2}$*–BIRS*; ${}_{-1}BA^{-2}$*–BARS*; medida 1 ranking y 2 SubEvaluador; *SF*–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*. N/D–No Disponible. * Tener en cuenta que en el promedio no están incluidos los datos para GCM.

la similitud entre todas las exactitudes obtenidas, y no sólo entre las dos versiones de cada algoritmo para cada modelo (*wrapper* o *CFS*), sino entre los ocho resultados por clasificador que se muestran para cada base de datos. Entre los resultados que no se aproximan a la media de los demás están:

- Por encima de la media:
  - Con el conjunto de datos *leukemia* se dan dos casos, correspondientes a las versiones *wrappers* de *BIRS* y de *BARS* con ranking *wrapper* y SP respectivamente (${}_{IB}BI^{IB}$ y ${}_{SP}BA^{IB}$) con el clasificador IB. Las dos sobresalen positivamente con 93,04 % y 93,21 %, estando los seis restantes por debajo del 88,93 %.
  - En *lymphoma* y el clasificador IB, siguiendo la aproximación filtro de *BIRS* con ranking SP (${}_{SP}BI^{IB}$);
- Y por debajo de la media:
  - Un caso en *colon*, ${}_{SP}BA^{C4}$, la versión *wrapper* de *BARS* con ranking SP para el clasificador C4, con una tasa de acierto de 75,48 % frente a las otras siete restantes que están por encima del 81,90 %.

- Por último, en la base de datos *lymphoma*, se dan diferencias en tres ocasiones: la primera con la versión *wrapper* de *BIRS* y ranking SP para el clasificador NB (${}_{SP}BI^{NB}$); la segunda y tercera, en la versión *wrapper* de *BARS* con ranking *wrapper* (${}_{C4}BA^{C4}$) y en la aproximación filtro–*CFS* de *BIRS* con ranking SP (${}_{SP}BI^{C4}$), ambas aplicando el clasificador C4.

Sin embargo, si existen diferencias al comparar los subconjuntos de genes seleccionados con las diferentes versiones de *BIRS* y *BARS*. En todos los casos, los algoritmos con un modelo *wrapper* ofrecen subconjuntos más reducidos que sus respectivas versiones que siguen un modelo filtro. Tomando la base de datos *colon* como ejemplo, la aproximación *CFS* de *BIRS* con ranking *CFS* (${}_{CF}BI^{CF}$) se queda con un subconjunto de 15,8 genes (es el mismo dato con los tres clasificadores), mientras que las tres versiones *wrappers* de este mismo método (BI_{WR}) retienen 3.5, 2.9 y 6.3 genes con NB, C4 e IB respectivamente. Si tomamos el número de genes modificando el ranking a SP (línea siguiente de los resultados anteriores) tenemos, 22.1 genes frente a 7.4, 2.9 y 7.6 genes con las respectivas aproximaciones *wrappers*. Este comportamiento se da de manera general en el resto de casos. En la tabla 6.2 se pueden comprobar los promedios de retención respecto del original. Las distintas versiones de *BIRS* y *BARS* se quedan con 0,18 %, 0,21 %, 0,12 % y 0,13 % de los atributos, para ${}_{WR}BI^{WR}$, ${}_{SP}BI^{WR}$, ${}_{WR}BA^{WR}$ y ${}_{SP}BA^{WR}$ respectivamente, en las aproximaciones *wrappers*, mientras que sube aproximadamente tres veces en las aproximaciones *CFS*: 0,67 %, 0,82 %, 0,35 % y 0,43 %.

Otra comparación posible entre los subconjuntos obtenidos por *BIRS* y *BARS* resulta de analizar los resultados dentro de cada grupo, *wrapper* y filtro–*CFS*, por cada clasificador. Los subconjuntos obtenidos con *BARS* contienen un número de genes menor. Empezando por las aproximaciones filtros (disponemos de un resultado por cada base de datos), tenemos: para *colon* 7,3 y 12,5 de ${}_{CF}BA^{CF}$ y ${}_{SP}BA^{CF}$ frente a 15,8 y 22,1 genes de las versiones *BIRS* respectivas; con *leukemia* existen menos diferencia, 5,6 y 5,7 genes con *BARS* y 6,7 y 7,3 genes para *BIRS*; para *lymphoma*, 30,5 y 35,1 de ${}_{CF}BA^{CF}$ y ${}_{SP}BA^{CF}$ frente a 57,6 y 67,5 de ${}_{CF}BI^{CF}$ y ${}_{SP}BI^{CF}$; y para *GCM*, con *BARS* 32,1 y 32,2 genes, mientras que con *BIRS* se tiene aproximadamente el doble, 56,1 y 65,7. Estos resultados se repiten con menor diferencia en las aproximaciones *wrappers*. Si observamos los promedios de genes retenidos mostrados por la tabla 6.2, y generalizando para todas las versiones del mismo algoritmo, *BIRS* retiene en promedio el 0,47 % de los genes para las cuatro bases de datos, mientras que *BARS* el 0,26 %.

Tabla 6.3: Comparativa de las técnicas *BIRS* y *BARS*. Tiempo (en segundos) al aplicar cada selector a datos genómicos.

		colon	*leuk*	*lymp*	*GCM*
wrapper	${}_{NB}BI^{NB}$	27,6	105,5	309,4	11060,9
	${}_{C4}BI^{C4}$	56,3	165,6	579,9	32328,5
	${}_{IB}BI^{IB}$	42,9	145,9	235,8	8669,7
	${}_{SP}BI^{NB}$	19,9	71,3	304,6	10626,8
	${}_{SP}BI^{C4}$	55,1	118,8	428,4	25624,3
	${}_{SP}BI^{IB}$	33,0	99,3	202,4	7888,3
	${}_{NB}BA^{NB}$	31,8	271,1	271,1	4645,8
	${}_{C4}BA^{C4}$	44,5	165,3	752,1	14961,5
	${}_{IB}BA^{IB}$	84,4	183,7	312,8	2347,7
	${}_{SP}BA^{NB}$	38,3	323,7	487,3	5066,2
	${}_{SP}BA^{C4}$	88,9	223,0	772,7	7379,5
	${}_{SP}BA^{IB}$	91,4	138,2	326,7	2546,5
	SF^{NB}	127,1	226,1	1357,7	N/D
	SF^{C4}	156,9	189,7	4540,7	N/D
	SF^{IB}	120,1	228,3	1032,8	N/D
CFS	${}_{CF}BI^{CF}$	3,7	21,9	33,8	1770,6
	${}_{SP}BI^{CF}$	3,6	16,0	30,6	2041,0
	${}_{CF}BA^{CF}$	2,4	18,5	20,6	208,9
	${}_{SP}BA^{CF}$	1,7	12,3	19,7	154,2
	SF^{CF}	13,7	498,9	1570,7	N/D
FCBF		0,4	2,0	8,7	26,3

wrapper–CFS: medida del *SubEvaluador*. ${}_{-1}BI^{-2}$–*BIRS*; ${}_{-1}BA^{-2}$–*BARS*; medida 1 ranking y 2 SubEvaluador; *SF*–búsqueda secuencial hacia adelante; *FCBF–Fast Correlation-Based Filter*. N/D–No Disponible.

BIRS y *BARS* versus datos originales

En la tabla 6.1 se puede ver que entre las exactitudes de clasificación obtenidas con los conjuntos de datos originales y los reducidos según *BIRS* o *BARS* no existen diferencias significativas, excepto con la base de datos *colon* y el clasificador NB, donde en todos los casos ganan *BIRS* y *BARS*, y con el conjunto *GCM* y C4, donde pierden en algunas de las aproximaciones. Por base de datos se puede observar que:

- Con la base de datos *colon*, además del resultado con NB comentado anteriormente, los clasificadores C4 e IB obtienen mejores resultados sobre los conjuntos de datos reducidos mediante *BIRS* y *BARS*.
- Para *leukemia* se obtiene un resultado similar con los clasificadores C4 e IB pero no con NB, donde se llega a la mejor exactitud con el conjunto original de datos.
- En *lymphoma* se aprecia que la versión *wrapper* de *BARS* gana en dos ocasiones con NB, y se obtiene el peor resultado sobre el conjunto original junto con la versión *CFS* de *SF*. Aplicando C4 e IB a la base de datos completa el resultado es intermedio.
- Finalmente para *GCM*, aparte del resultado ya comentado favorable al conjunto original con C4, no existen diferencias considerables.

Se debe resaltar la reducción tan drástica realizada por los algoritmos de selección propuestos en comparación con el conjunto original de datos (ver tabla 6.2), sobretodo con las versiones *wrappers*, y dentro de éstas, las practicadas con el método *BARS* (0,12 % y 0,13 %).

BIRS y *BARS* versus *SF*

No se muestran diferencias estadísticamente significativas entre las exactitudes obtenidas con *BIRS–BARS* y la exactitud obtenida según el procedimiento secuencial hacia adelante (SF_W), excepto para la base de datos *colon* aplicando el clasificador IB, donde las aproximaciones *wrappers* de *BIRS* y *BARS* ganan (tabla 6.1).

Por otro lado, la ventaja de los algoritmos *BIRS* y *BARS* con respeto a *SF* es clara con todos los clasificadores. Se puede observar (ver tabla 6.3) que *BIRS* y *BARS* son consistentemente más rápidos que *SF*. Esto es debido a que la función de evaluación de subconjuntos se ejecuta menos veces en los primeros. Por ejemplo, con el conjunto de datos *lymphoma* y el clasificador C4, ${}_{C4}BI^{C4}$ y SF^{C4} retienen de media un número similar de genes, 8,80 y 8,20 respectivamente. Para obtener estos subconjuntos, el primero evaluó individualmente 4026 genes (para generar el ranking) y 4026 subconjuntos, mientras que el segundo evaluó 32180 subconjuntos (4026 genes

+ 4025 pares de genes + ... + 4019 conjuntos de ocho genes). El tiempo ahorrado por *BIRS* y *BARS* se hace más obvio conforme aumentan las necesidades computacionales debidas a la base de datos utilizada y a la evaluación de subconjuntos practicada para obtener el subconjunto final de genes. En muchos casos el tiempo necesario para seleccionar el subconjunto con *SF* es varias veces el empleado por *BIRS* o *BARS*. Por ejemplo con la base de datos *lymphoma*, se tiene 1570 segundos de SF^{CF} frente a 33, 30, 20 y 19 segundos de las aproximaciones *CFS* de *BIRS* y *BARS*; y aplicando el clasificador C4 como *wrapper*, 4540 segundos de SF^{C4} frente a 579, 428, 752 y 772 segundos de ${}_{C4}BI^{C4}$, ${}_{SP}BI^{C4}$, ${}_{C4}BA^{C4}$ y ${}_{SP}BA^{C4}$ respectivamente. Además se debe tener muy en cuenta que las dos aproximaciones de *SF* (*wrapper* y *CFS*) no reportan resultado alguno con el conjunto de datos *GCM* después de cuatro semanas de ejecución.

En cuanto a la reducción practicada en las bases de datos (tablas 6.1 y 6.2), si bien en las aproximaciones *wrappers* es muy parecida, estando *SF* situado entre los algoritmos *BIRS* y *BARS*, con las versiones *CFS* el número de genes retenidos con *BIRS* y *BARS* está muy por debajo de los genes retenidos por *SF* (especialmente con las bases de datos *leukemia* y *lymphoma*). Concretamente *SF* retiene un 1,37 % de los genes, frente a 0,18 %, 0,21 %, 0,12 % y 0,13 % de los genes con las distintas versiones de *BIRS* y *BARS*. *BARS*.

Estos resultados verifican la eficiencia computacional de la búsqueda incremental aplicada por *BIRS* y *BARS* sobre la búsqueda secuencial utilizada por *SF*, con un número similar de genes (o inferior en el caso de los filtros) y sin diferencias estadísticas significativas en exactitud.

BIRS y *BARS* versus *FCBF*

Las exactitudes obtenidas con el algoritmo *FCBF* son muy similares a las obtenidas con *BIRS* y *BARS*, pero con un número de genes más elevado (tabla 6.1). *FCBF* retiene de media un 2,24 % de los genes para las cuatro bases de datos (tabla 6.2), mucho más que cualquiera de las versiones de *BIRS* y *BARS*. Las aproximaciones *CFS*, ${}_{CF}BI^{CF}$, ${}_{SP}BI^{CF}$, ${}_{CF}BA^{CF}$ y ${}_{SP}BA^{CF}$, retienen respectivamente, 0,67 %, 0,72 %, 0,35 % y 0,43 % de los genes para las cuatro bases de datos y los tres clasificadores, mientras que las aproximaciones *wrappers* utilizan el 0,18 %, 0,21 %, 0,12 % y 0,13 % de los genes de las bases de datos originales.

El coste computacional de *FCBF* es muy bajo (ver tabla 6.3). Los tiempos de computación resultantes de las aproximaciones filtros no son considerables en comparación con los empleados por las aproximaciones *wrappers*, y en este caso, *FCBF* es algo más rápido que cualquiera de las versiones filtro de *BIRS* y *BARS*.

En estos experimentos se ha utilizado el algoritmo *FCBF* implementado en el entorno WEKA con los valores por defecto. Sin embargo, si se modifica el umbral por el cual los atributos pueden ser descartados, los resultados obtenidos serán muy diferentes. Lo normal es que dis-

minuya el volumen de los conjuntos seleccionados junto con una pérdida de predicción muy considerable. Resumiendo los resultados con *FCBF* mostrados en la tabla 6.1 para todas las bases de datos y todos los clasificadores, se obtiene una exactitud del 79,27 % reteniendo un 2,24 % del conjunto original de genes (tabla 6.2). Si se establece el umbral en 0,25, se baja a 77,35 % de exactitud con 1,56 % de los genes. Y si se fija el parámetro en 0,5, la tasa de aciertos promedio baja a 59,81 %, reteniendo el 0,11 % de los genes. Este porcentaje es similar al obtenido con la versión wrapper de *BARS*, 0,12 %, aunque este último proporciona un promedio mayor de exactitud, 78,32 %.

Interpretación biológica

Para completar el estudio, se reduce cada base de datos aplicando los métodos de selección presentados, *BIRS* y *BARS*, a las bases de datos originales. En el proceso seguido por *BIRS* siempre se elige el primer gen de cada ranking, sin embargo, generalmente el resto de los genes seleccionados no se localizan en posiciones consecutivas del ranking.

Para la base de datos *colon* y el clasificador NB, *BIRS* escoge los genes M63391 (human desmin gene), H25136 (Inositol 1,4,5-Trisphosphate-binding protein type 2 receptor), M64231 (Human spermidine synthase gene) y R80427 (C4-Dicarbosylate transport sensor protein) localizados en las posiciones 1, 127, 159 y 160, respectivamente. Para IB, H77597 (H.sapiens mRNA for metallothionein), X12671 (Human gene for heterogeneous nuclear ribonucleoprotein (hnRNP) core protein A1) y H24956 (Proto–oncogene tyrosine–protein kinase receptor ret precursor) en las posiciones 1, 6 y 215, respectivamente. Y J02854 (Myosin regulatory light chain 2), D00860 (Ribose–phosphate pyrophosphskinase I) y M26383 (Human monocyte-derived neutrophil-activating protein mRNA) en las posiciones 1, 6 y 1033, respectivamente, para el clasificador C4. El primer gen de cada subconjunto aparece en la lista de genes relevantes detectados en estudios previos realizados sobre estas bases de datos [80, 73, 17].

Un comportamiento similar se da en el resto de conjuntos de datos. Para *leukemia* y el clasificador NB, *BIRS* selecciona los genes M84526, M27891, M31523 y M23197 entre los primeros veinte genes del ranking y M36652. Para IB, M23197, M27891, M31523 y M11722, todos ellos de entre los veinte primeros. Y sólo un gen, el primero (M27891) para el clasificador C4. El primer gen de cada subconjunto aparece en la lista de genes relevantes detectadas en estudios previos realizados sobre estas bases de datos [80, 73, 17]. Para *lymphoma* y el clasificador NB, *BIRS* selecciona tres genes de entre los veinte primeros y cinco más lejanos en la lista. Para IB, siete y tres respectivamente. Y dos genes y seis para el clasificador C4 [174]. Finalmente, para *GCM*, *BIRS* escoge tres, tres y cinco genes para NB, IB y C4, respectivamente, de entre los veinte primeros genes del ranking [183, 134].

Como se puntualizó anteriormente, los subconjuntos de genes seleccionados por el algoritmo *BARS* suelen contener menos genes que los subconjuntos seleccionados con *BIRS*, y se debe al proceso seguido en la obtención del subconjunto final. *BIRS* empieza a crear el subconjunto candidato con los genes más relevantes. A medida que avanza en el ranking va añadiendo aquellos genes que aporten información al subconjunto final, sin posibilidad de volver a probar con los genes que se han dejado atrás sin añadir. Sin embargo, en la búsqueda aglomerativa empleada en *BARS* se va ampliando la cardinalidad del subconjunto a partir de los conjuntos más relevantes. Lo normal es que los genes que componen el subconjunto final sean también relevantes individualmente, alcanzando resultados similares con menos genes que *BIRS*.

De las dos primeras bases de datos (*colon* y *leukemia*) se encuentra mucha más información par poder comparar subconjuntos o rankings de genes, mientras que con las dos restantes (*lymphoma* y *GCM*) las comparaciones están más limitadas, seguramente debido al hecho de que ambas poseen más de dos etiquetas dificultando así la tarea de clasificación. Realmente existen bastantes más algoritmos para trabajar sobre conjuntos de datos con dos etiquetas. Esta diferencia en el tipo de base de datos se refleja en los resultados obtenidos con *BARS*, de manera que con las dos primeras los subconjuntos de genes seleccionados están compuestos en su totalidad por genes que individualmente también son relevantes, encontrándose en las primeras posiciones de sus respectivos rankings iniciales, así como en las listas de genes más relevantes detectadas en estudios previos realizados sobre estas bases de datos [80, 73, 17]. Sin embargo, los genes seleccionados con *lymphoma* y *GCM* no se sitúan todos entre los primeros puestos de los rankings, tan sólo un pequeño porcentaje.

A continuación se muestran los conjuntos de genes seleccionados para las cuatro bases de datos con la aproximación filtro de *BARS*, de manera que se utiliza la medida *CFS* tanto para generar el ranking inicial como para obtener el subconjunto final. Estos conjuntos de genes elegidos no dependerán del clasificador aplicado. Los genes que componen el subconjunto extraído de la base de datos *colon*, M63391 (human desmin gene), M26383 (Human monocyte-derived neutrophil-activating protein mRNA), J02854 (Myosin regulatory light chain 2), H08393 (45395 Collagen alpha 2(XI) chain (Homo sapiens)) y X12671 (Human gene for heterogeneous nuclear ribonucleoprotein (hnRNP) core protein A1), se encuentran localizados entre los siete primeros genes del ranking inicial. Algo similar ocurre con el conjunto de genes seleccionado para *leukemia*, M23197, M27891, M92287, U46499, X95735, L09209-s y M31523, localizados entre los once primeros. No ocurre lo mismo con *lymphoma*, se escogen treinta y seis genes, GENE2536X, GENE3520X, GENE3795X, GENE3760X, GENE3734X, GENE2568X, GENE2495X, GENE2387X, GENE2400X, GENE2166X, GENE3274X, GENE3315X, GENE3251X, GENE3388X, GENE2668X, GENE2761X, GENE2758X, GENE-

1312X, GENE1339X, GENE1333X, GENE1192X, GENE1178X, GENE275X, GENE641X, GENE655X, GENE711X, GENE717X, GENE1323X, GENE405X, GENE1818X, GENE1730-X, GENE1602X, GENE1672X, GENE1673X, GENE1621X y GENE1719X, donde sólo diez de ellos están entre los cincuenta primeros genes del ranking. De *GCM* se obtiene un conjunto de veinte y ocho genes, D79205, M12529, M18728, U42390, U77180, U77594, X14046, X69699, Z84721-cds2, AB006781-s, D42040-s, X74929-s, X89399-s, L05144, AA095885, AA490691, D79819, R87834, W27827, X04476-s, RC-AA025296, RC-AA034364, RC-AA084286, RC-AA136474, RC-AA189083, RC-AA233541, RC-AA425309 y RC-AA425782, con tan sólo once entre los cincuenta primeros genes del ranking. Para obtener buenos subconjuntos con las dos últimas bases de datos no es suficiente con los genes más relevantes, se debe realizar una búsqueda más amplia intentando localizar las interacciones entre genes y desechando además la redundancia. *BARS* y *BIRS* lo realizan de una manera muy eficiente.

6.6. Conclusiones

Las técnicas tradicionales de selección de atributos se han desarrollado para un número de atributos no mayor de una centena. Las características propias de los datos provinientes de microarrays han fomentado el interés por desarrollar técnicas específicas de selección de genes que sean capaces de procesar, de manera efectiva, varios millares de genes.

Una de las principales ventajas de aplicar algoritmos de reducción de bases de datos en la fase de preprocesado es la reducción en el tiempo de procesamiento de los algoritmos de aprendizaje. En nuestro caso, las aplicaciones de los tres clasificadores sobre todas las bases de datos reducidas mediante nuestros algoritmos se realizan en un tiempo inferior al segundo. Sin embargo, algunas pruebas que se realizan sobre las bases de datos originales tardan más de un minuto.

Computacionalmente es inviable abordar el problema de la selección de un subconjunto óptimo de genes. Los métodos basados en ranking proporcionan una forma rápida de conocer la relevancia de los genes, aunque no elimina las interrelaciones con ellos. Los algoritmos presentados en el capítulo anterior y aplicados a este tipo de datos tratan directamente ese problema. Los resultados, tal como muestran los experimentos, son buenos, consiguiendo iguales o mejores tasas de clasificación con el subconjunto de genes seleccionados que con todos los genes iniciales.

BIRS y *BARS* utilizan una búsqueda muy rápida a través del espacio formado por los genes, y cualquier función de evaluación de subconjuntos puede ser utilizada para seleccionar el subconjunto final de genes, incluidos los algoritmos de aprendizaje. Cuando se utiliza una función

de evaluación *wrapper* en bases de datos de tan alta dimensionalidad se necesita una gran cantidad de recursos computacionales. *BIRS* y *BARS* reducen el espacio de búsqueda al trabajar directamente sobre el ranking, transformando la combinatoria de la búsqueda secuencial hacia adelante en búsqueda cuadrática. Sin embargo, la evaluación es mucho menos costosa dado que sólo unos cuantos de genes son seleccionados, y por tanto, la evaluación del subconjunto no tiene un coste computacionalmente elevado en comparación con otros métodos que tratan metodologías *wrappers*.

El análisis se ha realizado sobre cuatro bases de datos muy conocidas: *colon*, *lymphoma*, *leukemia*, y *global cancer map*, y todos los experimentos se han llevado a cabo utilizando la técnica de validación cruzada en la selección con diez carpetas.

En resumen, para bases de datos de muy elevada dimensionalidad, como las procedentes de experimentos con microarrays, los métodos basados en aproximaciones *wrappers* podrían ser prohibitivos. En cambio *BIRS* y *BARS* son técnicas rápidas que pueden aplicarse junto a cualquier modelo, filtro o *wrapper*, proporcionando buenos resultados en la exactitud predictiva y escogiendo subconjuntos con un número muy reducido de genes (0,12 %–0,82 % de media). Los resultados obtenidos mediante la aplicación de la aproximación *wrapper* de *BIRS* a bases de datos genómicas están aceptados para su publicación en [150].

Conclusiones y Trabajos Futuros

El éxito de muchos esquemas de aprendizaje, en sus intentos para construir modelos de datos, pasa por la habilidad para identificar un subconjunto pequeño de atributos altamente predictivos. La inclusión de atributos irrelevantes, redundantes o con ruido en la fase del proceso de construcción del modelo puede provocar un comportamiento predictivo pobre y un incremento computacional. La selección de atributos en campos como el aprendizaje automático sigue siendo un tema de interés, lo que cambia es el dominio de los datos donde se aplica. Para comprobarlo es suficiente con revisar los trabajos publicados en los últimos años sobre el tema. En estos trabajos se intenta encontrar métodos que permitan detectar aquellos atributos que aporten la mayor información posible al proceso de aprendizaje y cuya obtención implique un coste computacional abordable. De esta manera, se pueden aplicar a bases de datos de gran dimensionalidad, como es la tendencia actual.

Como se ha comentado a lo largo del documento, los métodos de selección se pueden clasificar en general como filtros, que son independientes del clasificador utilizado, y los *wrappers*, que basan la búsqueda en el rendimiento del clasificador. Además, también se pueden clasificar en métodos rankings o de subconjuntos, según devuelvan una lista ordenada o un grupo de atributos. Por consiguiente, la primera propuesta realizada en este trabajo de investigación, *SOAP*, se encuadra dentro de los métodos como filtro y ranking, con la introducción de la medida *NLC* que está basada en la consistencia de los datos y que permite obtener los atributos más relevantes para los algoritmos de clasificación. Mientras que las propuestas *BIRS* y *BARS* se encasillan entre las técnicas que devuelven un subconjunto de atributos, siendo posible utilizar medidas de evaluación tanto de tipo filtro como *wrapper*.

El propósito inicial de esta tesis doctoral fue encontrar formas de seleccionar atributos en entornos altamente dimensionales. La búsqueda de este objetivo general nos ha llevado a desarrollar diferentes propuestas, cuyas conclusiones se resumen en la siguiente sección. Además, en la última sección se enumeran los temas sobre los que seguimos trabajando y a los que se les dedicará nuestra atención en el futuro.

Conclusiones

A continuación se exponen las conclusiones que presentamos como resultado de la realización de esta tesis.

1. El problema de la selección de atributos en aprendizaje automático es un tema abierto a pesar del tiempo que se lleva tratando, a tenor de la bibliografía encontrada en la revisión realizada. También se puede constatar que no es un problema que se pueda enfocar desde un único marco de trabajo. En efecto, distintos autores han abordado este problema dándole diferentes enfoques. Este hecho precisa la clasificación de los métodos en función de diferentes parámetros. En concreto en este documento se ha utilizado la clasificación propuesta por Liu en sus diferentes trabajos [37, 103, 111, 112].

2. La calidad del algoritmo *SOAP* para obtener atributos relevantes se evalúa de forma empírica. Para comprobar diferentes aspectos de la medida *NLC* se diseñó y ejecutó un conjunto de experimentos. La medida muestra un buen comportamiento en bases de datos reales, donde la dependencia con los atributos no es conocida a priori y por tanto la bondad del método se estima en base a la tasa de acierto obtenida por los clasificadores utilizando el conjunto de atributos seleccionados. En la comparativa con otros cuatro métodos de selección y con tres clasificadores diferentes, la medida *NLC* da resultados similares a los otros métodos, pero con un coste computacional mucho menor.

3. Los trabajos tradicionales donde se realizan comparaciones de algoritmos de rankings de atributos, principalmente evalúan y comparan el método de selección de atributos en vez del ranking. En este trabajo se presenta una metodología para la evaluación del ranking, partiendo de la premisa de la no existencia de un único subconjunto ideal para todos los casos, y de que el mejor ranking va a depender de lo que el usuario esté buscando. Se puede concluir que el área bajo la curva del ranking (*AURC*) refleja el comportamiento completo de la lista de atributos ordenada, indicando su poder predictivo en global. En base al análisis de la evolución del *AURC*, se ha recomendado el uso de los algoritmos *SOAP* (SP) y *ganancia de información* (IG) y del clasificador *C4.5* (C4) cuando queremos clasificar con pocos atributos, y *Relief* (RL) y el clasificador *1–NN* (NN) en los demás casos.

4. La aplicación de los métodos de búsqueda más populares en aprendizaje automático a bases de datos con un gran número de atributos puede generar costes computacionales prohibitivos. Sin embargo, en esta tesis se han presentado dos nuevas técnicas de selección de atributos que permiten utilizar cualquier evaluador de subconjuntos –incluido el

modelo de evaluación *wrapper*– para encontrar un buen conjunto de atributos para clasificación. Se utiliza la definición de Utilidad Incremental Ordenada para decidir, al mismo tiempo, si un atributo (o subconjunto) es relevante y no redundante o no (no relevante o redundante). Los métodos, denominados *BIRS* (*Best Incremental Ranked Subset*) y *BARS* (*Best Agglomerative Ranked Subset*) se basan en la idea de relevancia y redundancia, en el sentido de que un atributo (o conjunto) ordenado se escoge si añade información al incluirlo en el subconjunto final de atributos elegidos. Estas heurísticas conducen a mejoras considerables en la exactitud obtenida, en comparación con el conjunto completo y frente a otros algoritmos de selección de atributos, junto a una notable reducción en la dimensionalidad.

De la aplicación del algoritmo *BIRS* vista en los experimentos de la sección 5.4.2 se pueden extraer algunas conclusiones. En principio, no se aprecia diferencias interesantes de resaltar al aplicar la búsqueda incremental partiendo de diferentes formas de generar rankings. La aplicación de *BIRS* a grandes bases de datos implica que con subconjuntos de atributos muy reducidos se obtengan resultados de clasificación iguales o incluso mejores (significativamente en algunos casos) que con la bases de datos originales. La precisión obtenida con la versión *wrapper* de *BIRS* y de *SF* es similar, sin embargo, *BIRS* extrae subconjuntos algo más reducidos y en un tiempo mucho menor. Además, *BIRS* se puede aplicar a bases de datos de cualquier tamaño, mientras que *SF* no. La versión *wrapper* de *BIRS* gana en precisión a todas las versiones filtro-*CFS* (*BIRS*, *SF* y *FCBF*), aunque para ello emplea un número mayor de atributos (excepto *FCBF*) y mucho más tiempo. Se puede entender que el tiempo obtenido con las distintas aproximaciones de tipo filtro es insignificante comparado con el calculado a partir de las aproximaciones *wrapper*. Para finalizar, resaltar los buenos resultados obtenidos con la aproximación *CFS* de *BIRS*, siendo las exactitudes similares a las de *SF* y algo por encima de las obtenidas con *FCBF*, sin embargo, los conjuntos más reducidos se obtienen con las versiones filtro de *BIRS*, y con un coste inferior a los demás filtros.

De la aplicación del algoritmo *BARS* analizada en los experimentos de la sección 5.5.2 se extraen algunas conclusiones más. Los resultados obtenidos con las aproximaciones de tipo *wrapper* y filtro del algoritmo *BARS* son muy parecidos, quizás, la versión *wrapper* ofrece algo más de exactitud con unos subconjuntos de atributos más pequeños. De la comparativa entre las dos versiones propuestas en el capítulo 5, *BIRS* y *BARS*, se puede extraer un balance ligeramente superior al primero en cuanto a exactitud (gana en cinco ocasiones de treinta y seis comparaciones), sin embargo, la reducción practicada en los conjuntos de datos es más favorable a *BARS* en su versión *wrapper*. Se consigue una

reducción drástica de los conjuntos originales, ofreciendo exactitudes equivalentes. Y en general, en comparación con el filtro *FCBF*, *BARS* ofrece unos resultados similares en precisión, pero con aproximadamente la mitad de los atributos, eso sí, necesitando más tiempo de ejecución.

5. Abordar el problema de la selección de un subconjunto óptimo de genes es computacionalmente muy costoso. Los métodos basados en ranking proporcionan una forma rápida de conocer la relevancia de los genes, aunque no elimina las interrelaciones con ellos. Los algoritmos presentados, *BIRS* y *BARS*, aplicados a este tipo de datos tratan directamente ese problema, y los resultados, tal como muestran los experimentos, son buenos, consiguiendo iguales o mejores resultados de clasificación con el subconjunto de genes seleccionados que con todos los genes iniciales.

6. Con estas propuestas de selección de atributos que se puedan aplicar a bases de datos de elevada dimensión en el marco del aprendizaje supervisado se cumple el objetivo de este trabajo de investigación. Además se consigue sin un coste computacional excesivo, que irá en función del método de evaluación escogido, al mismo tiempo que se reduce el conjunto de datos original a subconjuntos mínimos con muy pocos atributos.

Trabajos Futuros

En esta sección se indican algunas ideas que pueden continuar el trabajo presentado en este documento. A este respecto, actualmente estamos avanzando en las siguientes direcciones:

1. Construcción de atributos realizando transformaciones de atributos simples con la finalidad de mejorarlos para clasificación. Para ello, se toman los atributos como operandos y se relacionan mediante operadores simples, como sumar, restar, multiplicar, dividir, etc., para formar los nuevos atributos. Si tenemos en cuenta que de una base de datos con cien atributos, realizando operaciones con dos operandos (atributos) y cuatro operadores (+, -, $\times$ y $\div$), se transformaría en otra con casi dos mil quinientos, la medida de evaluación para saber como de buenos son debe ser muy rápida, y *SOAP* puede ser una buena opción.

2. De las bases de datos de microarrays se pueden extraer relaciones entre genes, de manera que cuando uno de ellos, o combinación de varios mediante transformaciones, toma ciertos valores, determina los valores de otros genes. Si mediante el algoritmo *SOAP* se consigue que un gen, simple o transformado mediante las transformaciones simples antes

mencionadas, al ordenarlo sea capaz de encontrar un relación con otro gen, o transformación de varios, se podría formar una red de genes donde estudiar las distintas relaciones de unos con otros. Para que este tipo de tareas se pueda aplicar a microarrays, donde la dimensionalidad es muy alta, el tipo de búsqueda debe ser muy rápido, y ahí es donde entra en juego *SOAP*.

3. Como se ha expuesto repetidamente, *SOAP* es un algoritmo que evalúa individualmente los atributos de una base de datos y devuelve un ranking. Pensamos que si esta misma idea se aplica a la evaluación de subconjuntos, los resultados deben ser buenos. Consistiría en, igual que se estudia el cambio de etiquetas (*NLC*) en una dimensión, analizar el comportamiento de una región con las colindantes en el espacio, es decir, estudiar los cambios de etiquetas que se producen en una zona del espacio formada por las secuencias de valores de un subconjunto de atributos, añadiéndoles los cambios que se producirían con las zonas que la rodean. De esta forma se mediría no sólo la consistencia existente dentro de una región, sino también con su entorno.

4. Con respecto al area bajo la curva de comportamiento al clasificar un ranking de atributos (*AURC*), se puede seguir avanzando. Los futuros trabajos irán encaminados a confirmar si se pueden extrapolar estos resultados a otras bases de datos con mucho mayor tamaño, así como en profundizar si existe alguna relación entre el método de ranking y el clasificador elegido. Además, se pretende ampliar el estudio con otras medidas de evaluación de atributos. Un tema muy relacionado sería evaluar los rankings mediante curvas *ROC* y estudiar qué subconjuntos son los mejores según las condiciones existentes en cada instante, y en consecuencia elegir el conjunto más adecuado.

5. Las más recientes publicaciones apuntan que la manera de resolver el problema de la selección de atributos en bases de datos altamente dimensionales es mediante algoritmos híbridos en distintos sentidos, combinando la ventajas de los métodos rankings y que evalúen subconjuntos de atributos, así como mezclando algoritmos de tipo filtro y *wrapper*. Se ha confirmado mediante los experimentos realizados con las dos propuestas desarrolladas en este documento, que algoritmos con las cuatro aproximaciones anteriores son muy válidos en estos entornos de elevada dimensionalidad. Por tanto, se puede seguir investigando técnicas de búsqueda sobre rankings, o utilizando listas ordenadas de subconjuntos en el proceso de búsqueda.

   En cuanto al algoritmo *BIRS*, se están probando diferentes modificaciones de la misma técnica: en primer lugar, en vez de emprender la búsqueda en el ranking hacia adelante, realizar el recorrido hacia atrás (*backward*), es decir, ver si eliminando el atributo menos

relevante del ranking la evaluación del subconjunto resultante es igual o mejor, y así sucesivamente hasta el primer atributo del ranking; en segundo lugar, se aplica *BIRS* tal como está, y a continuación realizar una eliminación (*backward*) de los atributos que no aporten información alguna; y en tercer y último lugar, ir realizando un recorrido hacia adelante y uno hacia atrás consecutivamente, es decir, una vez que obtenemos la evaluación del subconjunto añadiendo el siguiente elemento del ranking, se prueba también eliminando sucesivamente uno de los ya existentes en el subconjunto candidato, de esta forma se abarca un espacio de búsqueda más extenso. Los resultados obtenidos con la primera modificación son sorprendentemente buenos, sin embargo se puede prever su principal inconveniente, fruto del recorrido hacia atrás que realiza, que es el elevado coste computacional que supondría en bases de datos con una dimensionalidad muy alta, y utilizando una medida *wrapper* para evaluar los subconjuntos. En la segunda modificación se obtienen subconjuntos igualmente predictivos, pero con menor cardinalidad, sin apenas coste añadido. Con la última modificación, el coste computacional sería mucho más elevado, dado el incremento en el número de evaluaciones de subconjuntos que se realizaría, sin embargo, se podría probar con medidas de tipo filtro únicamente.

En cuanto al método aglomerativo *BARS*, también es susceptible de modificaciones. La primera modificación se da realizando una poda al generar la nueva lista de soluciones. Actualmente siguen adelante aquellos subconjuntos que superan al mejor subconjunto de la fase anterior, sin embargo, como se buscan soluciones con los mejores k subconjuntos, se pueden eliminar aquellos que no mejoren al mejor subconjunto evaluado de uno de las k pasadas, reduciendo el número total de evaluaciones. Existe otra modificación más profunda de esta técnica aglomerativa, de manera que en vez de generar subconjuntos de atributos con los k primeros conjuntos de las listas respectivas, se cree una lista con los k mejores subconjuntos hasta el momento, y se generan nuevos subconjuntos con el primero de la lista y el resto de atributos tomados individualmente. Una vez evaluado el nuevo subconjunto se comprueba si está dentro de la lista, ocupando su lugar si corresponde y echando hacia atrás los subconjuntos en la lista. El último de la lista dejaría de tenerse en cuenta. Además, para reducir el coste computacional, se podrían mezclar evaluaciones de tipo filtro y *wrapper* de manera que tan sólo aquellos subconjuntos que mejoren los subconjuntos con la primera medida serían tratados con la segunda.

Además de las anteriores propuestas, algunas de las cuales ya están siendo abordadas, seguimos trabajando con los algoritmos presentados en este documento analizando conjuntos de datos de otros dominios. A. Troncoso et al. [166] presentan una técnica para predecir los precios del mercado de la electricidad basada en la metodología de los vecinos más cercanos. En este

tipo de bases de datos con series temporales se espera que *SOAP* ofrezca buenos resultados, descubriendo los atributos que sean más influyentes. Otro tipo de datos donde puede ser muy conveniente el estudio de los atributos más relevantes es el de los flujos de datos o *stream*.

Apéndice A

Conjuntos de Datos

En este apéndice se muestran las características de las bases de datos utilizadas en los diversos experimentos realizados a lo largo de este trabajo de investigación. Todas las bases de datos utilizadas son públicas.

Entre las colecciones más conocidas están las bases de datos para aprendizaje automático de la Universidad de California Irvine (*UCI Repository of Machine Learning Databases*) [19][1]. En este apéndice, los conjuntos de datos de esta colección se encuentran divididos en dos grupos, tablas A.1 y A.2, en función del tamaño (teniendo en cuenta el número de atributos y el número de instancias) de la base de datos. Esta clasificación se realiza para favorecer la diversidad de procedimientos de evaluación, dado que algunas pruebas serían demasiado costosas llevarla a cabo con grandes bases de datos, ya sea en horizontal o en vertical.

Las cinco bases de datos que muestra la tabla A.3 fueron generadas para el congreso *NIPS* (*Neural Information Processing Systems*)[2] celebrado en el año 2003. Todas son supervisadas con dos etiquetas, y se eligieron para abarcar distintos dominios y dificultades. Los atributos son continuos o binarios, y la representación de los ejemplos en el espacio formado por los atributos puede ser densa o dispersa. Una base de datos (Madelon-MAD) fue artificialmente creada para ilustrar una dificultad en particular: seleccionar un conjunto de atributos cuando estos no aportan información por si solos. A las cuatro restantes se añadió un porcentaje de atributos aleatorios sin información sobre las etiquetas de la clase. Los detalles de la preparación de los datos se pueden encontrar en un informe técnico.

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html
[2] http://clopinet.com/isabelle/Projects/NIPS2003/

Tabla A.1: Conjuntos de datos pequeños de la Universidad de California Irvine (*UCI Repository of Machine Learning Databases*).

Datos	Acrón.	Ats	Instancias	Clases	Num	Sim	Missing
anneal Orig.	ANN	38	898	6	6	32	SI
audiology	AUD	69	226	24	0	69	SI
autos	AUT	25	205	7	15	10	SI
breast cancer	BCA	9	286	2	0	9	SI
br. wisconsin	BCW	9	699	2	9	0	SI
horse colic	COL	22	368	2	7	15	SI
horse Orig.	COO	27	368	2	7	20	SI
credit rating	CRR	15	690	2	6	9	SI
german credit	CRG	20	1000	2	7	13	NO
pima diab.	DIA	8	768	2	8	0	NO
glass	GLA	9	214	7	9	0	NO
glass2	GL2	9	163	2	9	0	NO
heart clev.	HEC	13	303	5	6	7	SI
heart hung.	HEH	13	294	5	6	7	SI
heart stat.	HES	13	270	2	13	0	NO
hepatitis	HPT	19	155	2	6	13	SI
ionosphere	ION	34	351	2	34	0	NO
labor	LAB	16	57	2	8	8	SI
lung cancer	LUC	56	32	3	56	0	SI
lymphography	LYM	18	148	4	3	15	NO
primary tumor	PRT	17	339	22	0	17	SI
segment	SEG	19	2310	7	19	0	NO
sonar	SON	60	208	2	60	0	NO
soybean	SOY	35	683	19	0	35	SI
vehicle	VEH	18	846	4	18	0	NO
vote	VOT	16	435	2	0	16	SI
vowel	VOW	13	990	11	10	3	NO
wine	WIN	13	178	3	13	0	NO
zoo	ZOO	17	101	7	1	16	NO

Tabla A.2: Grandes conjuntos de datos de la Universidad de California Irvine (*UCI Repository of Machine Learning Databases*).

Datos	Acrón.	Ats	Instancias	Clases	Num	Sim	Missing
ads	ADS	1558	3279	2	3	1555	SI
arrhythmia	ARR	279	452	16	279	0	SI
hypothyroid	HYP	29	3772	4	7	22	SI
isolet	ISO	617	1559	26	617	0	NO
kr vs kp	KRV	36	3196	2	0	36	NO
letter	LET	16	20000	26	16	0	NO
multi feat.	MUL	649	2000	10	649	0	NO
mushroom	MUS	22	8124	2	0	22	SI
musk	MUK	166	6598	2	166	0	NO
sick	SIC	29	3772	2	7	22	SI
splice	SPL	60	3190	3	0	60	NO
waveform	WAV	40	5000	3	40	0	NO

Tabla A.3: Conjuntos de datos *NIPS*–03.

Datos	Acrón.	Dominio	Tipo	Atributos	Inst.	%Aleat
Arcene	ARC	Espectrometría	Densa	10000	100	30
Dexter	DEX	Clasificación texto	Dispersa	20000	300	50
Dorothea	DOR	Descubrim. fármacos	Disp. binaria	100000	800	50
Gisette	GIS	Reconocim. dígitos	Densa	5000	6000	30
Madelon	MAD	Artificial	Densa	500	2000	96

Bibliografía

[1] J. Aguilar-Ruiz, J. Riquelme, and M. Toro. Data set editing by ordered projection. In *Proceedings of the* 14^{th} *European Conf. on Artificial Intelligence*, pages 251–55, Berlin, Germany, August 2000.

[2] J. Aguilar-Ruiz, J. Riquelme, and M. Toro. Evolutionary learning of hierarchical decision rules. *IEEE Systems, Man and Cibernetics Part B*, 33(2):324–331, 2003.

[3] D. Aha and R. Bankert. Feature selection for case-based classification of cloud types. In *Working Notes of the AAAI-94, Workshop on Case-Based Reasoning*, pages 106–112, 1994.

[4] D. Aha and R. Bankert. A comparative evaluation of sequential feature selection algorithms. In *5th Int. Workshop on Artificial Intelligence ans Statistics*, pages 1–7, Lauderdale, FL, 1995.

[5] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[6] A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu, J. Powell, L. Yang, G. Marti, T. Moore, J. H. Jr, L. Lu, D. Lewis, R. Tibshirani, G. Sherlock, W. Chan, T. Greiner, D. Weisenburger, J. Armitage, R. Warnke, R. Levy, W. Wilson, M. Grever, J. Byrd, D. Botstein, P. Brown, and L. Staudt. Distinct types of diffuse large b–cell lymphoma identified by gene expression profiling. *Nature*, 403:503–11, 2000.

[7] H. Almuallim and T. Dietterich. Learning with many irrelevant features. In *9th National Conf. on Artificial Intelligence*, pages 547–552, 1991.

[8] H. Almuallim and T. Dietterich. Efficient algorithms for identifying relevant features. In *9th Canadian Conf. on Artificial Intelligence*, pages 38–45, 1992.

[9] H. Almuallim and T. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1–2):279–305, 1994.

[10] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96:6745—-50, 1999.

[11] E. Alpaydin. Combined 5x2 cv f test for comparing supervised classification learning algorithms. *Neural Computation*, 11:1885–92, 1999.

[12] M. Barnerjee, S. Mitra, and H. Banka. Evolutionary-rough feature selection in gene expression data. *IEEE Transaction on Systems, Man and Cybernetics*, Part C: Applications and Reviews(aceptado), 2006.

[13] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. on Neural Networks*, 5(4):537–550, 1994.

[14] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, 53:370–418, 1763.

[15] D. Bell and H. Wang. A formalism for relevance and its application in feature subset selection. *Machine Learning*, 41(2):175–195, 2000.

[16] M. Ben-Bassat. *Handbook of statistics-II*, chapter Pattern recognition and reduction of dimensionality, pages 773–791. Prentice Hall, London, GB, 1982.

[17] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. *Proc. Natl. Acad. Sci. USA*, 98(26):15149–54, 2001.

[18] S. Bicciato, M. Pandin, G. Didone, and C. D. Bello. Pattern identification and classification in gene expression data using an autoassociative neural network model. *Biotechnology and Cioengineering*, 81:594–606, 2003.

[19] C. Blake and E. K. Merz. Uci repository of machine learning databases, 1998.

[20] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[21] L. Bobrowski. Feature selection based on some homogeneity coefficient. In *9th Int. Conf. on Pattern Recognition*, pages 544–546. IEEE Press, 1988.

[22] G. Brassard and P. Bratley. *Fundamentals of Algorithms*. Prentice Hall, 1996.

[23] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regresion trees*. Wadsworth Int. Group, Belmont, CA, 1984.

[24] J. Callan, T. Fawcett, and E. Rissland. An adaptative approach to case-based search. In *12th Int. Joint Conf. on Artificial Intelligence*, pages 803–808. Morgan Kaufmann, 1991.

[25] J. R. Cano, F. Herrera, and M. Lozano. Using evolutonary algorithms as instance selection for data reduction in kdd: an experimental study. *IEEE Trans. Evolutionary Computation*, 7(6):561–575, 2003.

[26] C. Cardie. Using decision trees to improve case-based learning. In *10th Int. Conf. on Machine Learning*, pages 25–32, 1993.

[27] C. Cardie and N. Howe. Empirical methods in information extraction. In *14th Int. Conf. on Machine Learning*, pages 65–79. Morgan Kaufmann, 1997.

[28] R. Caruana and D. Freitag. Greedy attribute selection. In *11th Int. Conf. on Machine Learning*, pages 28–36, 1994.

[29] R. Caruana and D. Freitag. How useful is relevance? In *Working notes of the AAAI fall symp. on relevance*, pages 25–29, 1994.

[30] C. Chang. Finding prototypes for nearest neighbor classifier. *IEEE Transactions on Computers*, 23(11):1179–1184, 1974.

[31] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[32] T. M. Cover. Estimation by nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14:50–55, 1968.

[33] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, 1967.

[34] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *18th Int. Conf. on Machine Learning*, pages 74–81. Morgan Kaufmann Publishers Inc., 2001.

[35] B. Dasarathy. Nearest neighbor (nn) norms: Nn pattern classification techniques. IEEE Computer Society Press, 1995.

[36] M. Dash. Feature selection via set cover. In *IEEE Knowledge and Data Engineering Exchange Workshop*, pages 165–171, 1997.

[37] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analisys*, 1(3):131–56, 1997.

[38] M. Dash and H. Liu. Hybrid search of feature subsets. In H. Lee and H. Motoda, editors, *15th Pacific Rim Int. Conf. on Artificial Intelligence*, pages 22–27, 1998.

[39] M. Dash, H. Liu, and H. Motoda. Consistency based feature selection. In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 98–109, 2000.

[40] S. Davies and S. Russell. Np-completeness of searches for smallest possible feature sets. In *AAAI Fall Symposium on Relevance*, pages 37–39, N. Orleans, LA, 1994.

[41] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[42] K. Deb and A. R. Reddy. Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems*, 72:111–129, 2003.

[43] J. Deutsch. Evolutionary algorithms for finding optimal gene sets in microarray prediction. *Bioinformatics*, 19:45–52, 2003.

[44] P. Devijver and J. Kittler. *Statistical Pattern Recognition*. Prentice Hall, London, GB, 1982.

[45] T. Dietterich. An experimental comparison of nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning*, 19(1):5–28, 1995.

[46] T. Dietterich. Approximate statistical test for comparing supervised classification learning alforithms. *Neural Computation*, 10(7):1895–1924, 1998.

[47] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *IEEE Computer Society Bioinformatics*, pages 523–529, 2003.

[48] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical Report CSE-92-18, University of California, Department of Computer Science, Davis, CA, 1992.

[49] J. Doak. An evaluation of search algorithms for feature selection. Technical report, Los Alamos National Laboratory, 1994.

[50] P. Domingos. Rule induction and instance-based learning: A unified approach. In *Int. Conf. on Artificial Intelligence*, 1995.

[51] P. Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997.

[52] E. Dougherty. Small sample issue for microarray-based classification. *Comparative and Functional Genomics*, 2:28–34, 2001.

[53] R. Duda and P.Hart. *Pattern classification and scene analysis*. John Willey and Sons, 1973.

[54] S. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(4):325–327, 1975.

[55] B. Efron. Bootstrap methods: another look at the jacknife. *Anals of Statistics*, 7(1):1–26, 1979.

[56] B. Efron. Estimating the error rate of a prediction rule: some improvements on cross-validation. *Journal of the American Statistical Association*, 78:316–331, 1983.

[57] B. Efron and R. Tibshirani. *An intoduction to the Bootstrap*. Chapman and Hall, London, UK, 1993.

[58] U. Fayyad and K. Irani. The attribute selection problem in decision tree generation. In *10th Nat. Conf. on Artificial Intelligence*, pages 104–110. MIT Press, 1992.

[59] E. Fix and J. Hodges. Discriminatory analysis, nonparametric discrimination consistency properties. Technical Report 4, US Air Force, School of Aviation Medicine, Randolph Field, TX, 1951.

[60] E. Fix and J. Hodges. Discriminatory analysis, nonparametric discrimination: small sample performance. Technical Report 11, US Air Force, School of Aviation Medicine, Randolph Field, TX, 1952.

[61] D. Foley. Consideration of sample and feature size. *IEEE Trans. Information Theory*, 18:618–626, 1972.

[62] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[63] I. Foroutan and J. Sklansky. Feature selection for automatic classification of non-gaussian data. *IEEE Transaction on System Man, and Cybernetics*, 17:187–198, 1987.

[64] D. Gamberger and N. Lavrac. Conditions for ocam's razor applicability and noise elimination. In *9th European Conf. on Machine Learning*, 1997.

[65] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–37, 1999.

[66] L. Grate, C. Bhattacharyya, M. Jordan, and I. Mian. Simultaneous relevant feature identification and classification in high-dimensional spaces. In *Lecture Notes in Computer Sciences, WABI'02*, pages 1–9. Springer–Verlag, 2002.

[67] C. Guerra-Salcedo, S. Chen, D. Whitley, and S. Smith. Fast and accurate feature selection using hybrid genetic strategies. In *Congress on Evulotionary Computation*, pages 177–184. IEEE Press, 1999.

[68] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[69] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machine. *Machine Learning*, 46(1-3):389–422, 2002.

[70] M. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *17th Int. Conf. on Machine Learning*, pages 359–366. Morgan Kaufmann, San Francisco, CA, 2000.

[71] M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Eng.*, 15(3), 2003.

[72] P. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, May 1968.

[73] T. Hellem and I. Jonassen. New feature subset selection procedures for classification of expression profiles. *Genome Biology*, 3(4):0017.1–0017.11, 2002.

[74] R. Holte. Very simple classification rules perform well on most commonly use datasets. *Machine Learning*, 11:63–91, 1993.

[75] E. Hunt, J. Marin, and P. Stone. *Experiments in induction*. Academis Press, New York, 1966.

[76] M. Ichino and J. Sklansky. Feature selection for linear classifier. In *7th Int. Conf. on Pattern Recognition*, pages 124–127, 1984.

[77] M. Ichino and J. Sklansky. Optimum feature selection by zero–one programming. *IEEE Transactions on Systems, Man and Cybernetics*, SMC–14(5):737–746, 1984.

[78] I. Iman and H. Vafie. An empirical comparison between global and greedy–like search for feature selection. In *Proceedings of the Florida AI Research Symposium (FLAIRS-94)*, pages 66–70, 1994.

[79] I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by bayesian networks based optimization. *Artificial Intelligence*, 123(1-2):157–184, 2002.

[80] I. Inza, P. L. naga, R. Blanco, and A. Cerrolaza. Filter versus wrapper gene selection approaches in dna microarray domains. *Artificial Intelligence in Medicine*, 31:91–103, 2004.

[81] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 22(1):4–37, 2000.

[82] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 19(2):153–158, 1997.

[83] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *11th Int. Conf. on Machine Learning*, pages 121–129, 1994.

[84] W. Kim, B. Choi, E.-K. Hong, and S.-K. Kim. A taxonomy of dirty data. *Data Mining and Knowledge Discovery*, 7:81–99, 2003.

[85] K. Kira and L. Rendell. A practical approach to feature selection. In *9th Int. Conf. on Machine Learning*, pages 249–256, 1992.

[86] V. Klee. On the complexity of d-dimensional voronoi diagrams. *Arch. Math.*, 34:75–80, 1980.

[87] R. Kohavi. *Wrapper for performance enhancement and oblivious decision graphs*. PhD thesis, Standfor University, Dpt. of Computer Science, 1995.

[88] R. Kohavi and B. Frasca. Useful feature subsets and rough set reducts. In *3rd Int. Workshop on Rough Set and Soft Computing (RCSSC-94)*, pages 310–317, 1994.

[89] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 1-2:273–324, 1997.

[90] R. Kohavi, P. Langley, and Y. Yun. The utility of feature weighting in nearest neighbor algorithms. In *Poster Papers: 9th European Conf. on Machine Learning*, Prague, Czech Republic, 1997.

[91] D. Koller and M. Sahami. Toward optimal feature selection. In *13th Int. Conf. on Machine Learning*, pages 284–292, 1996.

[92] I. Kononenko. Estimating attributes: Analysis and estensions of relief. In *European Conf. on Machine Learning*, pages 171–182, Vienna, 1994.

[93] M. Kudo and J. Sklansky. A comparative evaluation of medium and large–scale feature selectors for pattern classifiers. In $1^s t$ *Int. Workshop on Statistical Techniques in Pattern Recognition*, pages 91–96, Prague, Czech Republic, 1997.

[94] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 2000.

[95] M. Kudo, P. Somol, P. Pudil, M. Shimbo, and J. Sklansky. Comparison of classifier-specific feature selection algorithms. In *SSPR/SPR*, pages 677–686, 2000.

[96] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[97] P. Lachenbruch. An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis. *Biometrics*, pages 639–645, 1967.

[98] P. Langley. Selection of relevant features in machine learning. In *Procs. Of the AAAI Fall Symposium on Relevance*, pages 140–144, 1994.

[99] P. Langley and S. Sage. Oblivious decision tree and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 113–117, Seattle, WA, 1994.

[100] P. Larrañaga and J. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[101] W. Lee, S. Stolfo, and K. Mok. Adaptive intrusion detection: A data mining approach. *AI review*, 14(6):533–567, 2000.

[102] Y. Lee and C. Lee. Classification of multiple cancer types by multicategory support vector machines using gene expression data. *Bioinformatics*, 19:1132–1139, 2003.

[103] H. Liu and H. Motoda. *Feature Selection for Knowlegde Discovery and Data Mining*. Kluwer Academic Publishers, London, UK, 1998.

[104] H. Liu, H. Motoda, and M. Dash. A monotonic measure for optimal feature selection. In *European Conf. on Machine Learning*, pages 101–106. Springer Verlag, 1998.

[105] H. Liu, H. Motoda, and L. Yu. Feature selection with selective sampling. In *19th Int. Conf. on Machine Learning*, pages 395–402, 2002.

[106] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *7th IEEE Int. Conf. on Tools with Artificial Intelligence*, 1995.

[107] H. Liu and R. Setiono. Feature selection and classification: a probabilistic wrapper approach. In *9th Int. Conf. on Industrial and Eng. Aplications of AI and ES*, pages 129–135. Morgan Kaufmann, 1996.

[108] H. Liu and R. Setiono. A probabilistic approach to feature selection: a filter solution. In *13th Int. Conf. on Machine Learning*, pages 319–327. Morgan Kaufmann, 1996.

[109] H. Liu and R. Setiono. Incremental feature selection. *Applied Intelligence*, 9(3):217–230, 1998.

[110] H. Liu and R. Setiono. Some issues on scalable feature selection. *Expert Systems with Application*, 15:333–339, 1998.

[111] H. Liu and L. Yu. Feature selection for data mining. Technical report, Department of Computer Science and Eng., Arizona State University, Temp, Arizona, 2002.

[112] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Eng.*, 17(3):1–12, 2005.

[113] J. Lorenzo. *Selección de atributos en aprendizaje automático basada en teoría de la información*. PhD thesis, U. de Las Palmas de Gran Canaria, Dpto. de Informática y Sistemas, 2001.

[114] H. Mamitsuka. Iteratively selecting feature subsets for mining from high–dimensional databases. In *European Conf. on Principles and Practice of Knowledge Discovery in DB (PKDD)*, pages 361–372. IEEE Press, 2002.

[115] R. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2):111–161, 1983.

[116] H. Midelfart, J. Komorowski, K. Norsett, F. Yadetie, A. Sandvik, and A. Laegreid. Learning rough set classifiers from gene expression and clinical data. *Fundamenta Informaticae*, 53:155–183, 2003.

[117] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[118] M. Modrzejewski. Feature selection using rough sets theory. In *European Conf. on Machine Learning*, volume 667, pages 213–226. Springer Verlag, 1993.

[119] L. Molina, L. Belanche, and A. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Int. Conf. on Data Mining, ICDM-02*. IEEE Computer Society, 2002.

[120] A. Moore and M. Lee. Efficient algorithms for minimizing cross validation error. In *11th Int. Conf. on Machine Learning*, pages 190–198, 1994.

[121] A. Mucciarde and E. Gose. A comparison of seven techniques for choosing subsets of pattern recognition properties. *IEEE Transactions on Computer*, C-20(9):1023–1031, 1971.

[122] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computer*, C-26(9):917–922, 1977.

[123] K. Ng and H. Liu. Customer retention via data mining. *AI review*, 14(3):569–590, 2000.

[124] D. Nguyen and D. Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, 2002.

[125] A. Oliveira and A. Sangiovanni-Vicentelli. Constructive induction using a non-greedy strategy for feature selection. In *9th Int. Workshop on Machine Learning*, pages 354–360, 1992.

[126] P. Pudil and J. Novovicová. *Feature extraction, construction and selection*, chapter Novel methods for feature subset selection with respect to problem knowledge, pages 102–116. Kluwer Academic Publishers, 1998.

[127] P. Pudil, J. Novovicová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[128] D. Pyle. *Data preparation for data mining*. Morgan Kaufmann Publishers, 1999.

[129] C. Queiros and E. Gelsema. On feature selection. In *7th Int. Conf. on Pattern Recognition*, pages 128–130, 1984.

[130] J. Quinlan. Discovering rules by induction from collections of examples. In *Expert System in the Micro-Electronic Age*, pages 168–201, Edinburgh, 1979.

[131] J. Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine Learning: An Artificial Intelligence Approach*, Palo Alto, Tioga, 1983.

[132] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[133] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, California, 1993.

[134] S. Ramaswamy, K. Ross, E. Lander, and T. Golub. A molecular signature of metastasis in primary tomors. *Nature genetics*, 33:49–54, 2003.

[135] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. Mesirov, T. Poggio, W. Gerald, M. Loda, E. Lander, and T. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *J Comp Biol*, 7(3–4):559–84, 2000.

[136] J. Riquelme, J. Aguilar-Ruiz, and M. Toro. Finding representative patterns with ordered projections. *Pattern Recognition*, 36(4):1009–18, 2003.

[137] J. Riquelme, F. Ferrer, and J. Aguilar-Ruiz. Búsqueda de un patrón para el valor de k en k-nn. In *IX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'01)*, pages 63–72, Gijon, Noviembre 2001.

[138] G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

[139] Y. Rui, T. Huang, and S. Chang. Image retrieval: Current techniques, promising directions and open issues. *Visual Communication and Image Representation*, 10(4):39–62, 1999.

[140] R. Ruiz, J. Aguilar-Ruiz, and J. Riquelme. Soap: Efficient feature selection of numeric attributes. In *8th Ibero-American Conf. on AI (IBERAMIA)*, pages 233–242. Springer Verlag, 2002.

[141] R. Ruiz, J. Aguilar-Ruiz, and J. Riquelme. Reducción de bases de datos. In *VII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pages 469–478, 2003.

[142] R. Ruiz, J. Aguilar-Ruiz, and J. Riquelme. Wrapper for ranking feature selection. In *Intelligent Data Engineering and Automated Learning, 5th Int. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL'04)*, pages 384–389. Springer Verlag, 2004.

[143] R. Ruiz, J. Aguilar-Ruiz, J. Riquelme, and N. Diaz-Diaz. Analysis of feature rankings for classification. In *Advances in Intelligent Data Analysis VI (IDA 2005)*, pages 362–372. Springer Verlag, 2005.

[144] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Projection-based measure for efficient feature selection. *Journal of Intelligent and Fuzzy System*, 12(3–4):175–183, 2002.

[145] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Fast feature ranking algorithm. In *7th International Conference on Knowledge-Based Intelligent Information and Engineering System (KES'03)*, pages 325–331. Springer Verlag, 2003.

[146] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Fast feature selection by means of projections. In *16th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems (IEA/AIE'03)*, pages 461–470. Springer Verlag, 2003.

[147] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Nlc: A measure based on projections. In *14th International Conference on Database and Expert Systems Applications (DEXA'03)*, pages 907–916. Springer Verlag, 2003.

[148] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Databases reduction. In *6th International Conference on Enterprise Information System (ICEIS'04)*, pages 98–103, 2004.

[149] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Heuristic search over a ranking for feature selection. In *Computational Intelligence and Bioinspired Systems, 8th International Work-Conference on Artificial Neural Networks (IWANN'05)*, pages 742–749. Springer Verlag, 2005.

[150] R. Ruiz, J. Riquelme, and J. Aguilar-Ruiz. Incremental wrapper-based gene selection from microarray expression data for cancer classification. *Pattern Recognition, (aceptado)*, 2005.

[151] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309, 1991.

[152] M. Scherf and W. Brauer. Improving rbf networks by the feature selection approach eubafes. In *7th Int. Conf. on Artificial Neural Networks (ICANN-97)*, pages 391–396, Laussane, Switzerland, 1997.

[153] J. Schlimmer. Efficiently inducing determinations: A complete and efficient search algorithm tat uses optimal pruning. In *10th Int. Conf. on Machine Learning*, pages 284–290, 1993.

[154] J. Segen. Feature selection and constructive inference. In *7th Int. Conf. on Pattern Recognition*, pages 1344–1346. IEEE Press, 1984.

[155] R. Setiono and H. Liu. Improving backpropagation learning with feature selection. *Applied Intelligence*, 6:129–139, 1996.

[156] R. Setiono and H. Liu. Neural–network feature selector. *IEEE Trans. on Neural Networks*, 8(3):654–662, 1997.

[157] J. Sheinvald, B. Dom, and W. Niblack. A modelling approach to feature selection. In *10th Int. Conf. on Pattern Recognition*, volume 1, pages 535–539. IEEE Press, 1990.

[158] W. Siedlecki and J. Sklansky. On automatic feature selection. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220, 1988.

[159] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large–scale feature selection. *Pattern Recognition Letters*, 10:335–347, 1989.

[160] D. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *11th Int. Conf. on Machine Learning*, pages 293–301, 1994.

[161] N. Slonim, G. Bejarano, S. Fine, and N. Tishbym. Discriminative feature selection via multiclass variable memory markov model. In *19th Int. Conf. on Machine Learning*, pages 578–585, Sydney, Australia, 2002.

[162] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardinand, and S. Levy. A comprensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–643, 2005.

[163] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

[164] I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, June 1976.

[165] G. T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–68, 1980.

[166] A. Troncoso, J. M. Riquelme, A. G. J. L. Martínez, and J. C. Riquelme. Electricity market price forecasting based on weighted nearest neighbors techniques. *IEEE Transactions on Power Systems (aceptado)*.

[167] H. Vafaie and K. D. Jong. Robust feature selection algorithms. In *5th Conf. on Tools for Artificial Intelligence*, pages 356–363, 1993.

[168] H. Vafie and I. Iman. Feature selection methods: Genetic algorithms vs. greedy–like seach. In *3rd Int. Conf. on Fuzzy System and Intelligence Control*, 1994.

[169] H. Wang, D. Bell, and F. Murtagh. *Feature extraction, construction and selection*, chapter Relevance approach to feature subset selection, pages 85–97. Kluwer Academic Publishers, 1998.

[170] H. Wang, D. Bell, and F. Murtagh. Axiomatic approachto feature subset selection based on relevance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(3):271–277, 1999.

[171] K. Wang and S. Sundaresh. *Feature extraction, construction and selection*, chapter Selecting features by vertical compactness of data, pages 71–84. Kluwer Academic Publishers, 1998.

[172] D. Wilson. Asymtotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–21, July 1972.

[173] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.

[174] D. Wolowiec, F. Berger, P. Ffrench, P. Bryon, and M. Ffrench. Cdk1 and cyclin a expression is linked to cell proliferation and associated with prognosis in non-hodgkin's lymphomas. *Leuk Lymphoma*, 1(2):147–57, 1999.

[175] E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *Proc. 18th Int. Conf. on Machine Learning*, pages 601–608. Morgan Kaufmann, San Francisco, CA, 2001.

[176] M. Xiong, X. Fang, and J. Zhao. Biomarker identification by feature wrappers. *Genome Res*, 11:1878–87, 2001.

[177] M. Xiong, L. Jin, W. Li, and E. Boerwinkle. Computatinal methods for gene expression-based tumor classification. *BioTechniques*, 29:1264–70, 2000.

[178] L. Xu, P. Yan, and T. Chang. Best first strategy for feature selection. In *9th Int. Conf. on Pattern Recognition*, pages 706–708. IEEE Press, 1988.

[179] M. Xu and R. Setiono. Gene selection for cancer classification using a hybrid of univariate and multivariate feature selection methods. *Applied Genomics and Proteomics*, 2:79–91, 2003.

[180] H. Yang and J. Moody. Feature selection based on joint mutual information. In *International ICSC Symposium on Advances in Intelligent Data Analysis*, 1999.

[181] J. Yang and V. Honavar. *Feature extraction, construction and selection*, chapter Feature subset selection using a genetic algorithm, pages 117–136. Kluwer Academic Publishers, 1998.

[182] Y. Yang and J. Pederson. A comparative study on feature selection in text cateforization. In *14th Int. Conf. on Machine Learning*, pages 412–420. Morgan Kaufmann, 1997.

[183] C. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub. Molecular classification of multiple tumor types. *Bioinformatics*, 17(1):S316–22, 2001.

[184] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5:1205–24, 2004.

[185] L. Yu and H. Liu. Redundancy based feature selection for microarry data. In *10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2004.

[186] S. Zhang, C. Zhang, and Q. Yang. Data preparation for data mining. *Applied Artificial Intelligence*, 17(5–6):375–381, 2003.

[187] S. Zilberstein. Using anytime algorithms in intelligence systems. *AI magazine*, Fall:73–83, 1996.