universidad del país vasco — euskal herriko unibertsitatea

Facultad de Informática — **Informatika Fakultatea**

Medidas de filtrado de selección de variables mediante la plataforma "Elvira"

Alumno: D. Rubén Armañanzas Arnedillo

Director: D. Iñaki Inza Cano

Proyecto Fin de Carrera, agosto de 2004

RESUMEN

Este proyecto presenta la selección de variables mediante técnicas indirectas o *filter*, dentro de la plataforma de redes Bayesianas *Elvira*. Tras un estudio detallado de la bibliografía disponible en el campo, se desarrolló un diseño completo para *Elvira*, implementando una batería de medidas de selección *filter* de variables. Han sido implementadas siete diferentes métricas univariadas, así como un método de selección del punto de corte en una métrica. También se ha implementado un método de selección de atributos denominado CFS o *Correlation-based Feature Selection*. También han sido realizadas aportaciones a la interfaz gráfica de la plataforma. Por último, se evaluan los métodos implementados mediante una amplia experimentación en bases de datos procedentes de diferentes campos, incluyendo bases de datos génicas provenientes de microarrays de ADN.

Palabras clave: *Aprendizaje automático, selección de variables, métricas filter, CFS*

ABSTRACT

This project presents the selection of variables by means of indirect or *filter* techniques, inside the *Elvira* platform for Bayesian networks. After a detailed study of the available bibliography in the field, a complete design was developed for *Elvira*, implementing a set of *filter* variable selection measures. Seven different univariate metrics have been implemented, as well as a selection method of the cut point in a metric. Besides, it has been implemented an attribute selection method named CFS or *Correlation-based Feature Selection*. Moreover, contributions to the graphical interface of the platform have been made. Finally, the implemented methods have been tested by means of a wide experimentation with databases from different fields, including databases of genetic information extracted from DNA microarrays.

Keywords: *Machine learning, feature selection, filter metrics, CFS*

LABURPENA

Proiektu honek aldagai aukeraketarako *filter* neurriak (edo ez-zuzenak) erabiltzen ditu, sare Baiesiarretarako pentsatuta dagoen *Elvira* plataformaren barruan. Eremuan dagoen bibliografiaren ikasketa eta gero, *Elvira*rentzat diseinu oso bat egin zen, aldagai aukeraketarako *filter* neurri multzo bat programatuz. Zazpi neurri unibariatu egikaritu dira, hala nola mozketa puntu bat ere. CFS edo *Correlation-based Feature Selection* deitutako aldagai aukeratarako metodoa programatu da. Elviraren interfaze grafikoari hobekuntzak ere egin zaizkio. Azkenik, egikaritutako metodoak esperimentaketa zabal baten bidez ebaluatu dira: eremu ezberdinetako datubasetaz aparte, ADN microarrayetako datuak ere erabiliak izan dira.

Hitz gakoak: *Ikasketa automatikoa, aldagai aukeraketa, filter neurriak, CFS*

Índice

Índice de cuadros

Índice de figuras

Parte I

Descripción y gestión del proyecto

1. Introducción

El presente proyecto se encuentra enmarcado dentro de la disciplina conocida como *Aprendizaje Automático* del campo de la *Inteligencia Artificial*. Realizar una definición completa de *Aprendizaje Automático* se torna una tarea compleja, debido al gran campo de actuación que comprende; podríamos asumir que abarca todas aquellas técnicas que utilicen conocimiento previo como base de posteriores razonamientos. Según la Real Academia de la lengua Española[1] el **aprendizaje** es la *"adquisición por la práctica de una conducta duradera"*. En cuanto al segundo término, define **automático** como *"Dicho de un mecanismo: Que funciona en todo o en parte por sí solo"*, aunque también incluye esta otra definición: *"Ciencia que trata de sustituir en un proceso el operador humano por dispositivos mecánicos o electrónicos"*.

El espíritu final que se esconde tras el proyecto es la evaluación de "cuántas cosas" hemos de conocer para poder llevar a cabo correctamente un proceso de *Aprendizaje Automático*. Esas "cuántas cosas" se refieren a qué cantidad de información necesitamos conocer sobre el dominio de estudio; en el ámbito de la *Inteligencia Artificial*, cuántas variables –según la RAE **variable** es la *magnitud que puede tener un valor cualquiera de los comprendidos en un conjunto*– de nuestro problema van a tomar real importancia.

Centrándonos en este objetivo, alumno y tutor de este proyecto, hemos intentado desarrollar un proceso completo, desde el estudio biliográfico inicial, familiarización con las herramientas a utilizar, diseño e implementación, y amplia validación de las técnicas

[1] Estas definiciones pueden consultarse en la web de la Academia en *http://www.rae.es*

en dominios totalmente diferenciados entre sí.

Por último, y a modo introductorio, hay dos conceptos utilizados en múltiples ocasiones a lo largo del texto, sobre los cuales también hemos preguntado a la Academia:

- **ranking** – *Clasificación de mayor a menor, útil para establecer criterios de valoración*.

- **clase** – *Orden en que, con arreglo a determinadas condiciones o calidades, se consideran comprendidas diferentes personas o cosas*.

1.1. Localización

Durante el curso 2001/2002 el alumno cursa la asignatura de *Métodos Matemáticos en Ciencias de la Computación* dentro del plan de estudios de la Ingeniería en Informática. Es entonces cuando toma contacto con un campo novedoso para él y por el que se interesa rápidamente. Gracias a los profesores de esta asignatura, Pedro Larrañaga e Iñaki Inza, conoce la realización de un nuevo curso de verano en la Universidad Complutense de Madrid[2] sobre *bioinformática*, nueva disciplina que auna las técnicas informáticas con novedosas técnicas genéticas. En el verano del 2002 supera con éxito dicho curso, obteniendo el título de "Especialista en Bioinformática: Genómica y Proteómica".

En Enero del año 2003 el alumno se incorpora a la plantilla del grupo de investigación *ISG* (Intelligent Systems Group[3]) de la Facultad de Informática de Donostia-San Sebastián. Beneficiándose de una beca con título "Redes genéticas: modelizando la interacción entre genes por medio de redes Bayesianas y Gaussianas", su tarea como becario de colaboración, entre otras, se centra en el análisis de datos génicos mediante técnicas de *Aprendizaje Automático*.

[2] *http://www.ucm.es*
[3] *http://www.sc.ehu.es/isg*

Tras unos meses de familiarización con dichas técnicas, software a utilizar, primeras implementaciones y contacto con el mundo investigador, comienza en Octubre del mismo 2003 el desarrollo del presente proyecto fin de carrera. Debido a este entroncamiento directo del proyecto con las tareas habituales del alumno dentro del grupo de investigación, partes del proyecto se encontraban ya finalizadas o en proceso de ser finalizadas cuando el proyecto se inicia oficialmente.

El objetivo del proyecto va a consistir en el estudio, implementación y validación en bases de datos reales, de varias medidas de filtrado para la selección de variables en problemas de clasificación supervisada; conceptos estos que serán ampliamente introducidos a lo largo del texto. El software desarrollado se implementará utilizando como base la plataforma "Elvira II", siendo el grupo de investigación *ISG* uno de los grupos desarrolladores. Las bases de datos sobre las que se validarán estas técnicas proceden de diferentes fuentes, siendo una de ellas el repositorio *UCI* de bases de datos de la Universidad de California - Irvine[4]. Una de las bases de datos génicas utilizadas procede del trabajo común entre el departamento de Genética de Leioa de la UPV-EHU y el grupo *ISG* de investigación.

1.2. Estructura del documento

El documento se encuentra divido en seis capítulos y cuatro apéndices. Cada capítulo responde a un ámbito de aplicación del proyecto, mientras que los apéndices aclaran y/o amplian contenidos auxiliares al texto de la memoria.

A lo largo del Capítulo I se introduce al lector en el campo de actuación del proyecto, haciendo una descripción del trabajo realizado y de la gestión que llevaron a cabo alumno y tutor del mismo. En el Capítulo II se presenta, con cierta profundidad, la base teórica sobre la que se sustentan las técnicas implementadas y utilizadas en el estudio. El

[4] *http://www.uci.edu*

Capítulo III está dedicado a una revisión descriptiva de la plataforma *Elvira II*, módulos principales que la componen con breves descripciones de sus puntos clave. En el Capítulo IV se recogen las implementaciones realizadas en el marco del proyecto dentro del sistema Elvira, especificando con cierto nivel de detalle su estructura interna y funcionamiento. Por último, el Capítulo V presenta todo el repertorio de experimentos realizados en base a las técnicas implementadas, así como un amplio discurso sobre los resultados, conclusiones y trabajo futuro que puede ser realizado.

En cuanto a los apéndices, el Apéndice A describe el contenido del CD que acompaña a la memoria, donde puede encontrarse el software y bases de datos utilizados, así como cualquier otro contenido de interés. En el Apéndice B se incluye una breve guía de uso de las implementaciones realizadas en Elvira en el marco de este proyecto. En los Apéndices C y D se recogen las publicaciones conocidas en las que estas técnicas ya han sido puestas en juego, así como los agradecimientos del autor.

1.3. Análisis de antecedentes

Los dos antecedentes directos en los que se ha basado el trabajo realizado son:

1. El proyecto fin de carrera del alumno *Antonio J. Cerrolaza*, con título *Algoritmos indirectos discretos para la selección de variables en clasificación supervisada sobre microarrays de ADN*, defendido en Septiembre del 2002. Dicho proyecto recoge parte de la fundamentación teórica utilizada en el presente proyecto.

2. El proyecto *Elvira I*[5] en el que se basa el proyecto *Elvira II*[6] sobre el que han sido implementadas las técnicas de selección de variables.

[5] TIC97-1135-C04, 1997-2000
[6] TIC2001-2973-C05, 2001-2004

2. Planificación

2.1. Objetivos del proyecto

Parte de los objetivos del proyecto han sido ya presentados en la sección anterior. A continuación se enumeran con mayor detalle los objetivos acotados por el proyecto:

1. **Introducir en la plataforma Elvira la selección de variables**. En el Capítulo III se recoge una amplia revisión sobre los ámbitos de actuación de la plataforma Elvira. Dentro del *Aprendizaje Automático*, la selección de variables es un campo con entidad propia hace ya años, de ahí la necesidad de hacer que Elvira dispusiera de una parte dedicada a ello.

2. **Dar acceso mediante la interfaz gráfica a las implementaciones realizadas**. Además de la implementación de las rutinas necesarias, era preciso que el usuario final pudiera acceder a la selección de variables en Elvira de una manera, rápida, sencilla y amigable.

3. **Estudiar la eficiencia de las técnicas de selección de variables implementadas**. Una vez implementadas las diferentes técnicas de selección, realizar un estudio comparativo de todas ellas. Utilizándolas en problemáticas diversas, estudiar su rendimiento en conjunción con técnicas clasificatorias de *Aprendizaje Automático*.

4. **Aplicar los métodos a problemas de selección de genes en bases de datos de microarrays de ADN**. Enlazando con el punto 3, aplicar dichas técnicas a problemas de selección de genes en problemas bioinformáticos en los que el grupo *ISG* trabaja.

5. **Sentar las bases de una posible tesis doctoral en base a los conocimientos y prácticas adquiridas**. El desarrollo de todo el proyecto llevará a la consecución de

una serie de conocimientos por parte del alumno que podrían sentar las bases para una futura formación pre-doctoral.

2.2. Estimación de tiempos

Tarea	*Tiempo estimado*
1. Aproximación al entorno ELVIRA II	**37:00 hrs**
◎ Familiarización con el software	20:00*
◎ Realización de diagramas UML	9:00*
◎ Familiarización con el entorno gráfico	8:00*
2. Estudio bibliográfico	**40:30 hrs**
◎ Revisión del trabajo previo	8:00*
◎ Revisión de artículos científicos sobre FSA	32:30
3. Diseño	**12:00 hrs**
◎ Diseño de la clase FilterMeasures	4:00*
◎ Diseño de la clase FeatureSelection	3:30
◎ Diseño de clases complementarias	1:30
◎ Diseño de los nuevos aspectos en la interfaz gráfica	3:00*
4. Implementación	**128:00 hrs**
◎ Implementación de la clase FilterMeasures	35:00*
◎ Implementación de la clase FeatureSelection	29:00
◎ Implementación de clases complementarias	4:00
◎ Implementación de las amplicaciones en la interfaz gráfica	45:00*
◎ Pruebas de las clases	10:00
◎ Corrección de errores	5:00
5. Experimentación	**34:00 hrs**
◎ Diseño de los experimentos	4:00
◎ Ejecución y análisis de resultados	5:00
◎ Generación de resultados gráficos	25:00
6. Documentación	**50:00 hrs**
◎ Familiarización con el entorno LaTeX	2:00
◎ Generación de la memoria del proyecto	40:00
◎ Revisión posterior y correcciones	8:00
7. Reuniones de control y seguimiento del proyecto	**30:30 hrs**
TIEMPO TOTAL ESTIMADO	**332:00 hrs**

Cuadro 1: Estimación de tiempos para las tareas identificadas en el proyecto.

La fecha de inicio del proyecto es el 10 de Octubre del 2003. El alumno realiza en

esa fecha la identificación de tareas y estimación de tiempos a invertir que se incluyen en el Cuadro 1. Los tiempos marcados con un asterisco (*) no son estimaciones, sino que corresponden a tareas que se encontraban completadas a la fecha de inicio del PFC, debido a la labor llevada a cabo por el alumno dentro del grupo de investigación.

2.3. Identificación de riesgos

Los riesgos identificados en el proyecto se centran en los aspectos más técnicos del mismo:

- *Posibles errores en la codificación*. Se han establecido tareas dedicadas específicamente a la corrección de errores, sin embargo, debido a que no existe un resultado único en el ámbito del proyecto, pueden detectarse fallos de codificación a lo largo de todo el desarrollo del trabajo. Al trabajar con probabilidades, un error de codificación puede enmascararse hasta ser detectado en etapas posteriores de ejecución. En función del fallo, y del momento de ejecución en que se encuentre el proyecto, estos errores podrían llevar a un retraso importante en la ejecución del mismo.

- *Trabajo concurrente en el sistema*. La implementación del sistema Elvira se realiza de forma concurrente contra un servidor de gestión de proyectos distribuidos. De esta forma, todos los desarrolladores pueden acceder y modificar el código de la plataforma desde sus respectivos lugares de trabajo. El peligro reside en que otra persona modifique parte del código ya implementado, dando como resultado un mal funcionamiento del sistema.

- *Tiempos de cómputo*. Al trabajar con bases de datos de gran dimensionalidad, el tiempo de cómputo que algunas técnicas puedan necesitar puede no ser viable. Si bien el alumno dispone de grandes recursos de cómputo al pertenecer al grupo de

investigación *ISG*, puede darse la circunstancia de que alguno de los experimentos previstos no sean realizables en la práctica.

- *Riesgos clásicos a un ámbito informático*. Comenzando con posibles fallos de *hardware* o *software* que deben ser tenidos en cuenta. Es habitual también que los tiempos estimados en un proyecto informático se amplien debido a diferentes factores, carga lectiva o de trabajo paralela del alumno, mal acotamiento de alguna de las tareas, o re-implementación de partes de código mal estructuradas.

3. Gestión y ejecución

3.1. Tiempos invertidos

Tarea	*Tiempo estimado*
1. Aproximación al entorno ELVIRA II	**37:00 hrs**
◎ Familiarización con el software	20:00
◎ Realización de diagramas UML	9:00
◎ Familiarización con el entorno gráfico	8:00
2. Estudio bibliográfico	**33:00 hrs**
◎ Revisión del trabajo previo	8:00
◎ Revisión de artículos científicos sobre FSA	25:00
3. Diseño	**7:00 hrs**
◎ Diseño de la clase FilterMeasures	4:00
◎ Diseño de la clase FeatureSelection	–
◎ Diseño de clases complementarias	–
◎ Diseño de los nuevos aspectos en la interfaz gráfica	3:00
4. Implementación	**90:30 hrs**
◎ Implementación de la clase FilterMeasures	35:00
◎ Implementación de la clase FeatureSelection	–
◎ Implementación de clases complementarias	–
◎ Implementación de las amplicaciones en la interfaz gráfica	45:00
◎ Pruebas de las clases	7:30
◎ Corrección de errores	3:00
5. Experimentación	**0:00 hrs**
◎ Diseño de los experimentos	–
◎ Ejecución y análisis de resultados	–
◎ Generación de resultados gráficos	–
6. Documentación	**2:00 hrs**
◎ Familiarización con el entorno LaTeX	2:00
◎ Generación de la memoria del proyecto	–
◎ Revisión posterior y correcciones	–
7. Reuniones de control y seguimiento del proyecto	**8:10 hrs**
TIEMPO CONSUMIDO	**177:40 hrs**

Cuadro 2: Tiempos invertidos en las tareas del proyecto a fecha 16-Feb-2004.

Aproximadamente en la mitad de la ejecución del proyecto, el tutor requiere al alumno una tabla de tiempos completados, tabla que se recoge en el Cuadro 2. Para esta fecha,

16 de Febrero del 2004, las tareas concernientes a estudio bibliográfico, diseño e implementación de la parte de medidas indirectas univariadas (ver Sección 6.2.1) se encuentran finalizadas. Restaba, por tanto, el diseño e implementación de las técnicas de selección de variables (ver Sección 6.2.3), así como los apartados posteriores de experimentación y documentación.

El tiempo total invertido a fecha 16 de Febrero es de 177:40 horas, considerándose un tiempo ajustado a las expectativas. Tutor y alumno acuerdan como fecha de cierre del proyecto el 2 de Septiembre del 2004, en el que la memoria completa debe estar impresa y depositada para su defensa.

El Cuadro 3 recoge los tiempos finales de ejecución del proyecto a su cierre. Se da la circunstancia de que el tiempo de ejecución ha sido levemente menor que el estimado inicialmente. Este hecho se debe a que la tarea de *Implementación de la clase FeatureSelection* fue cancelada (ver Sección 11) incrementándose, por contra, el tiempo necesario para la tarea *Implementación de la clase FilterMeasures*.

Respecto a las demás tareas, resaltar que el apartado dedicado a *Documentación* ha llevado un tiempo considerablemente más amplio que el estimado inicialmente.

3.2. Seguimiento y supervisión

A lo largo de la ejecución del proyecto, la supervisión por parte del tutor ha sido continuada. Se desarrollaban reuniones de control entre alumno y tutor cada semana, presentando el trabajo realizado durante esa semana y planificando el trabajo a realizar en la siguiente semana.

En el CD que acompaña a este texto se incluyen todas las actas de las reuniones de seguimiento a lo largo de la ejecución del proyecto (consultar Apéndice A). En estas actas se incluyen pormenorizadas el desarrollo de cada una de las tareas y las decisiones de gestión tomadas a lo largo de la ejecución.

Tarea	*Tiempo estimado*
1. Aproximación al entorno ELVIRA II	**37:00 hrs**
◎ Familiarización con el software	20:00
◎ Realización de diagramas UML	9:00
◎ Familiarización con el entorno gráfico	8:00
2. Estudio bibliográfico	**42:00 hrs**
◎ Revisión del trabajo previo	12:00
◎ Revisión de artículos científicos sobre FSA	30:00
3. Diseño	**16:30 hrs**
◎ Diseño de la clase FilterMeasures	4:00
◎ Diseño de la clase FeatureSelection	7:30
◎ Diseño de clases complementarias	2:00
◎ Diseño de los nuevos aspectos en la interfaz gráfica	3:00
4. Implementación	**116:00 hrs**
◎ Implementación de la clase FilterMeasures	50:00
◎ Implementación de la clase FeatureSelection	–
◎ Implementación de clases complementarias	3:30
◎ Implementación de las amplicaciones en la interfaz gráfica	48:00
◎ Pruebas de las clases	9:00
◎ Corrección de errores	5:30
5. Experimentación	**21:00 hrs**
◎ Diseño de los experimentos	2:30
◎ Ejecución y análisis de resultados	6:00
◎ Generación de resultados gráficos	12:30
6. Documentación	**71:00 hrs**
◎ Familiarización con el entorno LaTeX	2:00
◎ Generación de la memoria del proyecto	55:00
◎ Revisión posterior y correcciones	14:00
7. Reuniones de control y seguimiento del proyecto	**27:00 hrs**
TIEMPO TOTAL INVERTIDO	**330:30 hrs**

Cuadro 3: Tiempos invertidos en las tareas del proyecto a su cierre.

3.3. Gestión de riesgos

De los riesgos identificados en la Sección 2.3, dos de ellos aparecieron a lo largo del desarrollo del proyecto. El primero fue la detección de un error de codificación en la implementación de la medida indirecta *Kullback-Leibler* en su modo 2 (ver Sección 6.2.1). El fallo afectó a la batería de experimentos que ya habían sido realizados utilizando dicha medida. Debido a ello, el alumno tuvo que volver a ejecutar los experimentos afectados con la consiguiente demora en las tareas planificadas para dicha fecha.

El segundo fue la imposibilidad de ejecutar la selección de variables *CFS* (ver Sección 6.2.3) sobre las dos bases de datos de microarrays de ADN (ver Sección 14) debido a limitaciones físicas de memoria en su ordenador personal. Para acometer dichas tareas fue necesaria la utilización de los recursos tecnológicos disponibles por el alumno en el grupo de investigación *ISG* (uso de biprocesadores de cómputo dedicado, y cesión de una cuenta especial de trabajo en el Centro de Cálculo de la Facultad de Informática sobre una máquina paralela).

Con respecto a los otros riesgos, el alumno llevó una política constante de copias de seguridad para evitar posibles pérdidas de contenidos, así como, una dedicación constante de horas semanales a la ejecución del proyecto.

3.4. Especificaciones técnicas

El software utilizado para el desarrollo del presente proyecto ha sido:

- Sistema Operativo: Linux Mandrake 9.2 y 10.0[7]
- Entorno de programación: Oracle JDeveloper 9.0.3 y 10.0g[8]
- Versión de LaTeX: LaTeX 2_ε[9]

[7] *http://www.mandrakelinux.com*
[8] *http://www.oracle.com/tools/index.html*
[9] *http://www.latex-project.org* y *http://www.ctan.org*

- Editor de LaTeX: Kile 1.5.2 y 1.6.3[10]
- Diseño de gráficas: Matlab 6.3[11]
- Entorno de ofimática: Sun StarOffice 6 y 7[12]
- Análisis estadístico: SPSS 11.5[13]

[10] *http://kile.sourceforge.net*
[11] *http://www.mathworks.com*
[12] *http://wwws.sun.com/software/star/staroffice*
[13] *http://www.spss.com*

Parte II

Fundamentación teórica

4. Modelos gráficos probabilísticos

Los modelos gráficos probabilísticos vienen a aunar las posibilidades de expresión de los grafos en procesos de razonamiento complejos, y toda la teoría de la probabilidad ampliamente integrada en el campo científico.

Los modelos gráficos probabilísticos representan distribuciones de probabilidad conjuntas multivariante. Esta representación se expresa mediante un productorio de términos, cada uno de los cuales abarca sólo unas pocas variables de todo el conjunto de variables consideradas. La estructura del producto se representa mediante un grafo que relaciona variables que aparecen en un mismo término.

Será el grafo el que especifique la forma del producto de las distribuciones y provea, asímismo, de herramientas para el razonamiento sobre propiedades vinculadas a los términos de dicho producto (Lauritzen and Spiegelhalter, 1988). Para un grafo disperso, la representación será compacta y en muchos casos permitirá procesos eficientes de inferencia y aprendizaje.

Para una completa revisión sobre modelos gráficos probabilísticos puede ser consultado (Larrañaga, 2001). Debido a su implantación dentro de la comunidad investigadora y a sus contrastados resultados, las redes Bayesianas han sido el paradigma de modelo gráfico probabilístico seleccionado.

4.1. Redes Bayesianas

Sea un conjunto de variables aleatorias –atributos del problema– definido como

$$\boldsymbol{X} = (X_1, ..., X_n)$$

La distribución de probabilidades conjunta en una red Bayesiana (Pearl, 1988; Jensen, 2001; Neapolitan, 2003) se representa como un producto de probabilidades condicionadas. Cada variable aleatoria X_i tiene asociada una probabilidad condicional

$$p(X_i = x_i \mid \boldsymbol{Pa}_i = \boldsymbol{pa}_i)$$

donde $\boldsymbol{Pa}_i \subset \mathbf{X}$ es el conjunto de variables identificadas como padres de X_i. El producto resultado es de la forma:

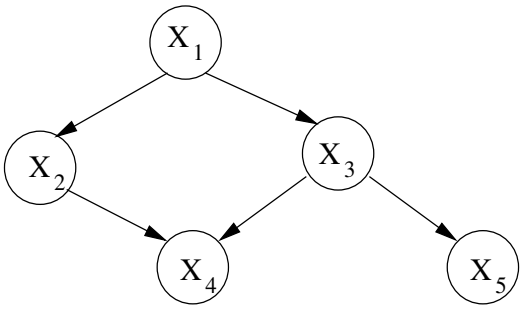
$$p(X_1 = x_1, ..., X_n = x_n) = \prod_{i=1}^{n} p(X_i = x_i \mid \boldsymbol{Pa}_i = \boldsymbol{pa}_i)$$

La representación gráfica es un grafo dirigido en el que se establecen uniones dirigidas entre los padres de $X_i(\boldsymbol{Pa}_i)$ y el propio X_i. Al tener en cuenta probabilidades condicionadas, el número de parámetros necesarios para determinar la distribución conjunta se reduce. En la Figura 1 podemos observar la reducción en ese número, pasando de 31 a 11 parámetros. La factorización conjunta para el modelo presentado en la Figura 1 se expresará como

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1) \cdot p(x_4 \mid x_2, x_3) \cdot p(x_5 \mid x_3)$$

Para obtener una red Bayesiana es necesario definir sus dos componentes:

Estructura – Es la parte cualitativa del modelo, la estructura se especifica mediante un



a. Estructura de la red Bayesiana

$p(X_1 = 0) = 0{,}20$
$p(X_2 = 0|X_1 = 0) = 0{,}80$
$p(X_2 = 0|X_1 = 1) = 0{,}80$
$p(X_3 = 0|X_1 = 0) = 0{,}20$
$p(X_3 = 0|X_1 = 1) = 0{,}05$
$p(X_4 = 0|X_2 = 0, X_3 = 0) = 0{,}80$
$p(X_4 = 0|X_2 = 1, X_3 = 0) = 0{,}80$
$p(X_4 = 0|X_2 = 0, X_3 = 1) = 0{,}80$
$p(X_4 = 0|X_2 = 1, X_3 = 1) = 0{,}05$
$p(X_5 = 0|X_3 = 0) = 0{,}80$

b. Parámetros

Figura 1: Ejemplo de red Bayesiana, estructura y parámetros.

grafo dirigido acíclico, o *DAG*, que refleje las dependencias condicionales entre las variables (Figura 1.a.). El concepto de independencia condicional entre tripletas de variables forma la base para entender e interpretar el campo de las redes Bayesianas.

Parámetros – Es la parte cuantitativa del modelo, y está formada por las probabilidades condicionales y no condicionales, o marginales, de las variables de la red (Figura 1.b.). Aquellos nodos que no tengan padres, *nodos raíz*, tendrán asociadas una serie de probabilidades marginales. El resto de nodos de la red, o nodos hijo, tendrán asociadas unas probabilidades condicionales en función de los valores que puedan tomar sus padres o antecesores.

Una vez que la red Bayesiana ha sido construída, constituye un eficiente medio para realizar tareas de inferencia probabilista (Lauritzen and Spiegelhalter, 1988). Dada una evidencia sobre el valor de ciertas variables en la red, puede ser calculada la distribución de probabilidad sobre otra serie de variables que puedan ser de interés. Sin embargo, el problema de construir la red Bayesiana se mantiene.

Este problema puede ser abordado desde dos vertientes diferenciadas; una, mediante la ayuda externa de expertos sobre el dominio a estudio –con el consiguiente gasto de tiempo y posibles errores en las aproximaciones–, o, dos, mediante el aprendizaje automático tanto de la estructura como de los parámetros en base a un conjunto de hechos conocidos

sobre el dominio, es decir, sobre una base de datos de casos.

Por otro lado, la tarea del aprendizaje puede separarse en dos subtareas: la identificación de la topología de la red Bayesiana, esto es, el aprendizaje de la estructura; y el aprendizaje paramétrico como estimación de parámetros numéricos (probabilidades condicionales e incondicionales) prefijada una topología de red Bayesiana.

Dentro del aprendizaje automático, existen dos grandes aproximaciones al problema de inducción de modelos Bayesianos: aproximaciones basadas en la detección de dependencias condicionales entre tripletas de variables; y, métodos de *score+search*.

Los algoritmos que intenta averiguar la estructura de una red Bayesiana mediante la detección de dependencias condicionales tienen como entrada algunas relaciones de dependencia condicional entre algún subconjunto de variables del modelo; y, devuelven como salida un grafo dirigido acíclico, que representa un alto porcentaje (incluso el 100 % si es posible) de esas relaciones. Una vez que la estructura ha sido aprendida, los parámetros necesarios son estimados a partir la base de datos. En (Spirtes et al., 1993) pueden consultarse más detalles sobre esta aproximación al aprendizaje de redes Bayesianas desde datos.

Un gran número de los algoritmos existentes en el campo de la modelización Bayesiana pertenecen a la categoría de métodos *score+search*. Para poder utilizar esta aproximación, necesitamos definir una métrica que mida la bonanza de cada red Bayesiana candidata con respecto al fichero de casos. Además, necesitamos también un procedimiento eficiente que guíe el proceso de búsqueda a través de todos los posibles grafos acíclicos dirigidos. Tres de los *scores* más habituales son: la penalización de la verosimilitud máxima, *scores* Bayesianos conocidos como de verosimilitud marginal, y *scores* basados en la teoría de la información. Con respecto al procedimiento de búsqueda existen muchas alternativas diferentes en la literatura: búsqueda exhaustiva, enfriamiento estadístico, algoritmos genéticos, búsqueda tabú, etc. Para una revisión sobre los méto-

dos de *score+search* en el aprendizaje de redes Bayesianas a partir de datos, el artículo de (Heckerman et al., 1995) puede ser consultado.

Los métodos de aprendizaje utilizados en el presente trabajo pertenecen a la categoría de detección de dependencias. Y, dentro de esa categoría, nos centraremos en modelos bayesianos que sean capaces de predecir una característica especial del problema.

5. Clasificación supervisada

Sea un conjunto de N hechos, cada uno caracterizado por $n+1$ variables. Las primeras n variables, $X_1, X_2, ..., X_n$, serán denominadas *variables predictoras*, y la variable de índice $n+1$, identificada como C, será denominada *clase*. Estos datos pueden representarse en forma de tabla –ver Cuadro 4–, utilizando la siguiente notación:

1. x_i^j será el valor que en el j-ésimo hecho toma la variable predictora i-ésima, $i = 1, ..., n$ y $j = 1, ..., N$

2. c^j será la clase a la que pertenece el j-ésimo hecho.

Los hechos pueden ser denominados también *casos* o *instancias*; y, a las variables se les conoce también como *atributos* del problema supervisado.

	X_1	X_2	$\dots$	X_i	$\dots$	X_n	C
1	x_1^1	x_2^1	$\dots$	x_i^1	$\dots$	x_n^1	c^1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
j	x_1^j	x_2^j	$\dots$	x_i^j	$\dots$	x_n^j	c^j
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
N	x_1^N	x_2^N	$\dots$	x_i^N	$\dots$	x_n^N	c^N

Cuadro 4: Elementos en una clasificación supervisada.

El objetivo es conseguir un *modelo clasificatorio* que posibilite la predicción del valor de la variable C ante la llegada de un nuevo caso, formado por n variables predictoras y del que se desconoce el valor de C. Es decir, clasificar correctamente un nuevo hecho basándonos en las evidencias anteriores.

Para poder acometer ese objetivo de la mejor manera posible, es necesario disponer de una base de conocimiento amplia y de calidad. Amplia, ya que cuanto más número de casos se disponga sobre el problema mejores serán las aproximaciones obtenidas, y, de calidad, ya que la presencia de casos repetidos, ruido o errores en los datos, variables

irrelevantes (que aporten poca información), o variables redundantes entre sí, puede hacer que el modelo no se ajuste correctamente a la realidad, siendo sus predicciones erróneas.

Es habitual que el número de variables predictoras sea elevado, o que no todas aporten la misma cantidad de información por lo que suelen aplicarse técnicas de *selección de variables* que ayuden a encontrar estas variables.

Las aplicaciones de este tipo de paradigmas clasificatorios son enormes ya que, en general, cualquier sistema cuyos resultados puedan ser agrupados en clases, y del que se disponga de una base de evidencias previas, es susceptible de ser analizado con esta técnica.

A la hora de resolver ese tipo de problemas de clasificación supervisada, los campos de la estadística y el aprendizaje automático han desarrollado diferentes técnicas: Análisis discriminante (Fisher, 1936), redes neuronales (McCulloch and Pitts, 1943), clasificadores K-NN (Cover and Hart, 1967), sistemas clasificadores (Holland, 1975), árboles de clasificación (Breiman et al., 1984), regresión logística (Hosmer and Lemeshow, 1989), inducción de reglas (Clark and T. Niblett, 1989) y máquinas de soporte vectorial (Cristianini and Shawe-Taylor, 2000), entre otras.

Los clasificadores Bayesianos son la apuesta realizada para resolver el problema supervisado en este texto. La teoría de redes Bayesianas nos proveé de la fundamentación matemática para abordar el problema clasificatorio.

5.1. Clasificadores Bayesianos

Las variables consideradas en este tipo de clasificadores toman un número finito de posibles estados, *variables discretas*. Para el caso de variables que tomen valores en un rango infinito, *variables continuas*, el paradigma de los clasificadores Gaussianos puede ser estudiado.

Los clasificadores Bayesianos están basados en la formulación del Teorema de Ba-

yes (Bayes, 1764)

$$p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)}$$

donde:

1. $p(A)$ es conocido como la probabilidad *a priori* de que el suceso *A* sea cierto.

2. $p(A|B)$ es conocido como la probabilidad *a posteriori*, o, la probabilidad de que el suceso *A* sea cierto tras considerar *B*.

3. $p(B|A)$ es conocido como verosimilitud o *likelihood*, e indica la probabilidad de que el suceso *B* sea cierto, asumiendo que *A* lo es.

4. $p(B)$ es la probabilidad *a priori* de que el suceso *B* sea cierto. Actúa de coeficiente normalizador o estandarizador en la fracción.

Este teorema no sólo puede ser aplicado a sucesos, sino también a variables aleatorias, tanto unidimensionales como multidimensionales. Su formulación general es:

$$p(Y = y|X = x) = \frac{p(Y = y) \cdot p(X = x|Y = y)}{\sum_{i=1}^{r_y} p(Y = y_i) \cdot p(X = x|Y = y_i)}$$

Aplicado al problema de clasificación supervisada, tenemos que $Y = C$ es una variable unidimensional; mientras que $\boldsymbol{X} = (X_1, X_2, ..., X_n)$ es una variable n-dimensional. $\boldsymbol{X}$ será la variable predictora, e Y la variable a predecir –la clase predicha por el modelo–.

Asumiendo una función de error 0/1, un clasificador Bayesiano $\gamma(\boldsymbol{x})$ asigna la clase con mayor probabilidad a posteriori dada una determinada instancia, es decir,

$$\gamma(\boldsymbol{x}) = \arg\max_{c} \; p(c \mid x_1, ..., x_n)$$

donde c representa la variable clase, y $x_1, ..., x_n$ son los valores de las variables predictoras. Podemos expresar la probabilidad a posteriori de la clase de la siguiente manera:

$$p(c \mid x_1, ..., x_n) \propto p(c) \cdot p(x_1, ..., x_n \mid c)$$

Asumiendo diferentes factorizaciones para $p(x_1, ..., x_n \mid c)$ se puede obtener una jerarquía de modelos de creciente complejidad dentro de los clasificadores Bayesianos, hasta ordenes exponenciales de $2^{m \cdot n}$ siendo m y n el número de dimensiones de las dos variables aleatorias. En el presente proyecto han sido considerados dos de los modelos ampliamente utilizados: naïve Bayes y tree augmented naïve Bayes.

5.1.1. Naïve Bayes

Naïve Bayes (Minsky, 1961) es un algoritmo Bayesiano de clasificación supervisada construído en base a la asunción de independencia condicional entre las variables predictoras, dado el valor de la clase.

Aceptando esta independencia condicional, la distribución de probabilidad a posteriori puede expresarse como

$$p(x_1, ..., x_n \mid c) = \prod_{i=1}^{n} p(x_i \mid c)$$

Una vez conocidas las variables predictoras $(x_1, x_2, ..., x_n)$ la determinación de la clase más probable c^* se reduce a encontrar el conjunto de variables predictoras que maximizan la probabilidad de pertenencia a una determinada clase:

$$c^* = \underset{c \in \{c^1, ..., c^m\}}{\arg\max} \; p(c|x_1, ..., x_n)$$

En base al supuesto de independencia condicional entre las variables, se verifica que

$$p(c|x_1, ..., x_n) \propto p(c) \prod_{k=1}^{n} p(x_k|c)$$

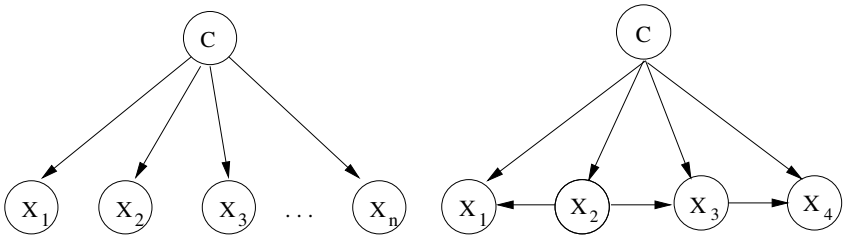
de manera que el clasificador naïve Bayes puede ser formulado como

$$c^* = \underset{nB}{\gamma}(\boldsymbol{x}) = \underset{c \in \{c^1,...,c^m\}}{\arg\max}\ p(c) \prod_{i=1}^{n} p(x_i \mid c)$$

donde $\boldsymbol{x}$ constituye el caso o instancia a clasificar.

En la Figura 2.a. se muestra la estructura gráfica del modelo. La gran ventaja del uso de este modelo es la sencillez en su formulación, y por tanto la de los cálculos a realizar. El tiempo de cómputo necesario para su aprendizaje y validación es lineal en el número de variables.

Pero existe un gran inconveniente, cuando la asunción de independencia condicional entre las variables predictoras no se cumple, la pérdida de precisión clasificatoria de este modelo puede ser grande. Este supuesto será violado cuando entre las variables predictoras existan redundancias, o bien, existan variables altamente correlacionadas unas con otras.



a. Naïve Bayes b. Tree augmented naïve Bayes

Figura 2: Estructuras de los clasificadores Bayesianos utilizados.

5.1.2. Tree augmented naïve Bayes

El modelo naïve Bayes no es capaz de tratar con dependencias entre las variables predictoras. En dominios donde la independencia condicional entre las variables predictoras dada la clase no se cumple, el rendimiento de un modelo naïve Bayes puede limitarse en gran manera.

El modelo bayesiano TAN o tree augmented naïve Bayes (Friedman et al., 1997) construye un clasificador donde existe una estructura de dependencias arborescente entre las variables predictoras. Basándose en un modelo de dependencias semejante al del naïve Bayes, añade dependencias condicionales entre los nodos, formando un árbol entre ellas. La mezcla de ambas estrategias hace posible la relajación de independencia entre las variables predictoras, en la Figura 2.b. se muestra un ejemplo de modelo TAN.

Este modelo, propuesto por Friedman et al. (1997), está basado en el cálculo de la información mutua condicionada entre pares de variables,

$$I(X_i, X_j \mid C) = \sum_{c} \sum_{x_i} \sum_{x_j} p(x_i, x_j, c) log \frac{p(x_i, x_j \mid c)}{p(x_i \mid c) \cdot p(x_j \mid c)}$$

y fuerza a construir una estructura conexa de árbol con todas las variables del dominio del problema.

5.2. Validación de un clasificador

Tras realizar el proceso de aprendizaje automático mediante un conjunto de datos de entrada, o conjunto de entrenamiento, se plantea la etapa de evaluar la calidad del modelo aprendido, para, posteriormente, utilizarlo como "predictor" ante nuevos casos.

5.2.1. Validación cruzada en *k*-hojas

La evaluación de la calidad de un modelo clasificatorio, o estimación de la bondad clasificatoria, se realiza clásicamente mediante un proceso de *validación cruzada*. El proceso recibe este nombre debido a los cruces que realiza entre aprendizaje y validación de modelos.

El fichero de entrenamiento va a ser dividido aleatoriamente en tantas partes, u hojas, como se le indique a la validación cruzada. El paso siguiente consiste en aprender, con todas las partes menos una, un modelo del tipo que estemos testeando. Por último, se clasifican los casos de la parte que no ha sido proporcionada al aprendizaje mediante el modelo aprendido con las demás. Este proceso se efectua tantas veces como hojas tenga la validación, dejando siempre como partición de validación una distinta. En la Figura 3 se muestra un esquema de una valicación cruzada en *k*-hojas.

Real	c_1	c_2	$\dots$	c_m
Asignado				
c_1	ε_{11}	ε_{12}	$\dots$	ε_{1m}
c_2	ε_{21}	ε_{22}	$\dots$	ε_{2m}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_m	ε_{m1}	ε_{m2}	$\dots$	ε_{mm}

Cuadro 5: Esquema de una matriz de confusión para un problema supervisado.

El resultado de una validación cruzada consiste en una matriz cuadrada en la que se van a recoger los casos correctamente predichos y los casos en los que los modelos se han equivocado. Esta matriz recibe el nombre de *matriz de confusión media* o *matriz de mala clasificación* y toma la estructura mostrada en el Cuadro 5. La dimensión de la matriz de confusión, m, la marcará el número de clases que haya en el problema, es decir, el número de estados que pueda tomar la variable C.

En cada una de las hojas de la validación serán testeados N/k casos, siendo N el

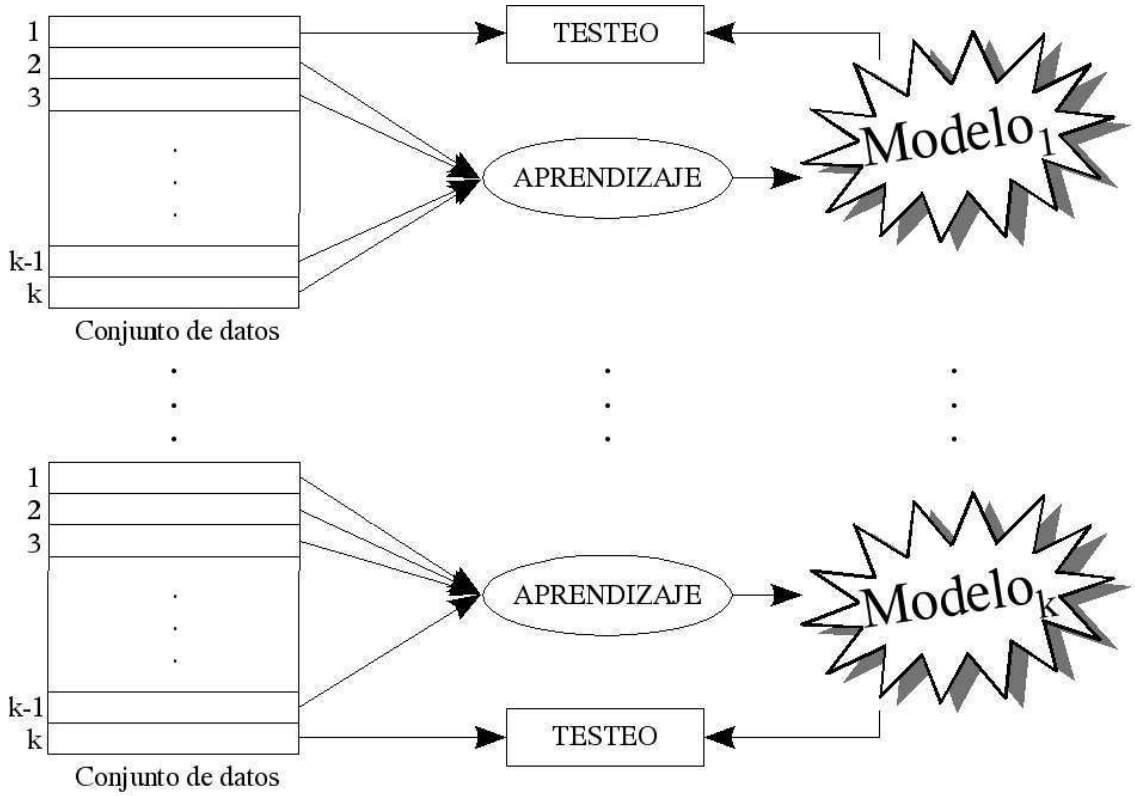


Figura 3: Esquema general de una *validación cruzada*.

número de casos totales en el conjunto de entrenamiento. La *matriz de confusión media* se obtiene como suma de matrices intermedias de cada una de las k hojas.

Los valores indicados como ε_{ij} corresponderán al número de casos que el modelo ha clasificado pertenecientes al tipo c_i, siendo su clase real la c_j. Un modelo óptimo será aquel que no "coloque" ningún caso en otra clase que no sea la suya, es decir $\forall i \neq j \in [1, ..., m]\ \varepsilon_{ij} = 0$.

Una medida clásica de la precisión media de un clasificador será el número de casos correctamente predichos en función del número total de casos clasificados. A este coeficiente se le denomina *precisión media* o *accuracy* del clasificador:

$$Accuracy(\,\%) = \frac{\sum_{i=1}^{m} \varepsilon_{ii}}{\sum_{i=1}^{m} \sum_{j=1}^{m} \varepsilon_{ij}} \times 100$$

La precisión media nos da una medida probabilista de, ante un nuevo caso de clase desconocida, cuál es el grado de fiabilidad de la predicción realizada.

5.2.2. Validación *leave-one-out*

La validación cruzada puede ser llevada hasta su extremo configurando el número de hojas k como el número de casos que conformen el conjunto de entrenamiento. En este caso, los modelos intermedios serían aprendidos con $N - 1$ casos de los N posibles, testeándolo contra el caso no utilizado. Este método recibe el nombre de validación por *leave-one-out*.

Esta técnica es la más próxima al paradigma clasificatorio que se ha ido presentando. Aprender un modelo con una serie de datos y confrontarlo contra un caso nuevo. Para ese caso nuevo se conoce su clase lo que hace que podamos estudiar el comportamiento de los modelos clasificatorios para todos los casos. Es el método de validación más exigente y que obtendrá unos resultados más reales.

Su gran inconveniente es que serán necesarios tantos procesos de aprendizaje como casos tenga el fichero de entrenamiento. En comparación con la *validación cruzada*, el tiempo de cómputo que puede demorar una *validación leave-one-out* puede llegar a ser muy elevado, incluso inviable.

6. Selección de variables

A lo largo de esta sección se va a presentar la problemática de la selección de variables predictoras en un modelo clasificatorio. Tras ello, se exponen en detalle varias de las técnicas más ampliamente reconocidas en este campo, técnicas implementadas a lo largo de la ejecución del presente proyecto y utilizadas en la realización de los experimentos abordados.

6.1. Introducción

El problema de la selección del subconjunto de variables para la inducción de un modelo clasificador, es denominado FSS (*Feature Subset Selection*) (Bell and Wang, 2000; Inza et al., 2000) y surge motivado por la *no monotocidad* de los modelos clasificatorios en relación con el número de variables predictoras, así como por la existencia de ciertas variables predictoras que pueden llegar a ser *irrelevantes* o incluso *redundantes*.

- La *no monotocidad* de la probabilidad de éxito de un sistema clasificador se debe al hecho –constatado empíricamente, y demostrado matemáticamente para algunos paradigmas– de que no por construir un modelo clasificatorio con una variable añadida a las ya existentes, la probabilidad de éxito que se va a obtener con este nuevo modelo clasificador deba superar a la del modelo actual.

- Se considera que una variable predictiva es *irrelevante* cuando el conocimiento del valor de la misma no aporta información alguna que despeje incertidumbre sobre la *variable clase*.

- Una variable predictiva se dice *redundante* cuando su valor puede ser determinado a partir de otras variables predictivas.

El objetivo al tratar de resolver el problema FSS es el de detectar aquellas variables que son irrelevantes y/o redundantes para un problema clasificatorio dado. Un ejemplo de dos variables redundantes sería tener en un mismo conjunto de datos el salario de una persona expresada en euros, y en pesetas. Un ejemplo de variable irrelevante sería el incluir las condiciones climatológicas del día en el que se recibe un correo basura, cuando lo que se persigue es la detección automática de ese tipo de correos.

Lo que se pretende es crear modelos parsimoniosos[14], guiados por el principio que en *Aprendizaje Automático* se conoce como KISS (*Keep It as Simple as possible, Idiot*). Es decir, la idea subyacente a la modelización parsimoniosa es que si se dispone de dos modelos que explican suficientemente bien los datos, se debe escoger el modelo más simple de los dos.

Beneficios directos derivados del FSS:

- *reducción en el costo de adquisición de los datos*, debido a que el volumen de información a manejar para inducir el modelo es menor
- *mejora en la comprensión del modelo clasificatorio*, ya que no vamos a inducir el modelo con un gran número de variables
- *inducción más rápida del modelo clasificatorio*, derivado de la no monotocidad explicada
- *posibilidad de indagar en la naturaleza de distintos casos*, debido al tamaño más manejable de cada instancia, la tarea de identificar patrones en ciertos subconjuntos se vuelve más fácil

Beneficios indirectos derivados del FSS:

[14] Según la *RAE*: escaso, ahorrativo

- *mejora en la precisión del modelo clasificatorio*, por la propia naturaleza de algunos de los modelos clasificatorios, el hecho de que no existan variables redundantes y/o variables irrelevantes hace que su comportamiento sea óptimo

- *abordar los modelos clasificatorios más complejos*, debido a la reducción de los datos, muchos de los modelos clasificatorios más complejos pueden ser aplicados cuando, antes de la selección, el proceso era inviable debido a las limitaciones computacionales actuales

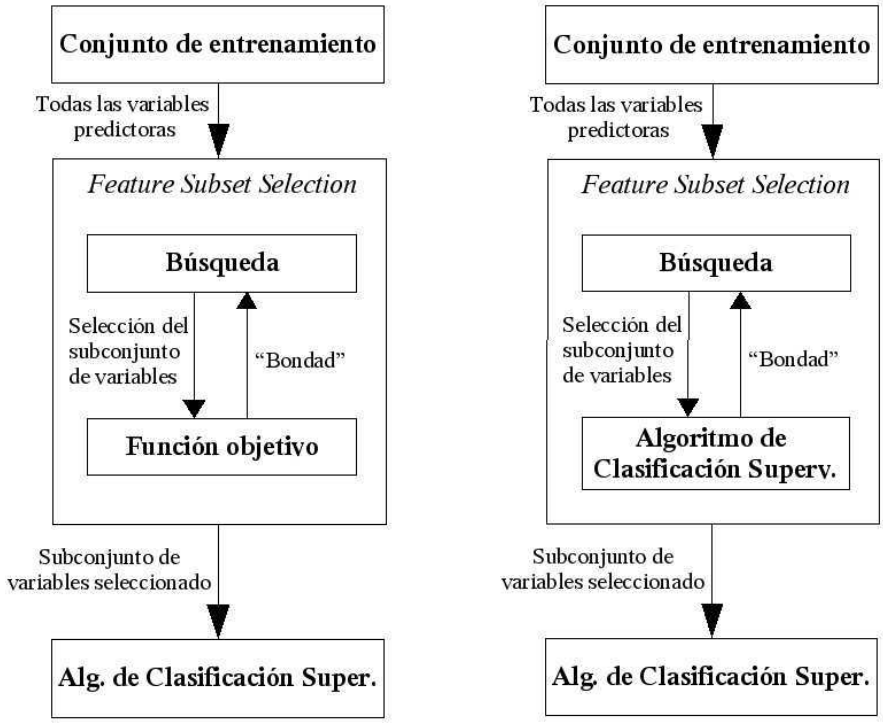
6.1.1. Filtrado y Envoltura

Dos son básicamente las aproximaciones al problema FSS: indirecta o *filter* y directa o *wrapper*.

La aproximación indirecta o *filter* (Ben-Bassat, 1982) va a hacer uso de *heurísticos* para determinar el subconjunto de atributos óptimo. Un *heurístico* es una regla matemática que es capaz de guiar un proceso de búsqueda hacia una solución. Una de sus principales ventajas es su rapidez de cálculo, hecho que hace que en conjuntos de datos con alta cardinalidad las aproximaciones *filter* sean consideradas como óptimas y utilizadas por los diferentes autores.

Además de la determinación de subconjuntos óptimos, dentro de la selección de variables tienen gran importancia los métodos que asignan coeficientes de relevancia a cada atributo, estableciendo un ranking entre ellos. A partir de dicha medida de relevancia, las variables predictoras quedan ordenadas –supongamos de mayor a menor relevancia respecto de la variable clase– seleccionándose las k, $k < n$, primeras para inducir con ellas el modelo clasificatorio.

Nótese que la medida de relevancia con la que se realiza la selección en las aproximaciones filter no hace uso de ningún modelo clasificatorio en el proceso de búsqueda, ver Figura 4.a.



a. Aproximación indirecta o *filter* b. Aproximación directa o *wrapper*

Figura 4: Esquemas de funcionamiento de los métodos directo e indirecto en selección de variables.

En la aproximación directa o *wrapper* (Kohavi and John, 1997) al FSS, cada posible subconjunto de variables candidato es evaluado por medio del modelo clasificatorio inducido –con el paradigma utilizado– a partir del subfichero que contenga exclusivamente las variables seleccionadas, junto con la variable clase, ver Figura 4.b.

Con esta aproximación *wrapper*, el problema FSS puede ser visto como un problema de búsqueda en un espacio de cardinalidad 2^n. En esta aproximación han sido utilizados diferentes heurísticos estocásticos de búsqueda –enfriamiento estadístico, algoritmos genéticos y algoritmos de estimación de distribuciones–, así como políticas clásicas de busqueda –*hill-climbing*, *best-first*, *branch& bound*, etc– con éxito desde esta perspectiva wrapper (Kohavi, 1995).

Existen técnicas mixtas o *híbridas* (Das, 2001) que intentan aunar las ventajas de una

y otra metodologías en una sola. Mezclan métodos directos e indirectos en los distintos subprocesos de la selección de atributos, aunque no existe ningún método híbrido que haya demostrado una mejora significativa con respecto a las otras dos.

6.2. Técnicas de filtrado

Debido a la rapidez de los metodos indirectos, el tratamiento de conjuntos de datos con alta cardinalidad se hace viable, cuando mediante los métodos directos, este tratamiento no era tecnológicamente abordable. Si añadimos a esto el hecho de que los rendimientos medios obtenidos por las aproximaciones indirectas se encuentran cercanas a las obtenidas por las directas, tendremos las explicaciones a la proliferación de técnicas *filter*, frente al estancamiento que han sufrido las técnicas *wrapper*.

6.2.1. Ranking de variables

Dentro de las aproximaciones *filter*, uno de los métodos más extendidos es la realización de clasificaciones o *rankings* de importancia de los atributos. Estos *rankings* son realizados de forma *univariada*, es decir, tan solo va a tenerse en cuenta la relación existente entre el atributo que está siendo analizado y la variable supervisada.

Mediante la aplicación de distintos heurísticos derivados de diferentes medidas de divergencia entre funciones de distribución, se va generando un coeficiente o peso para cada uno de los atributos. Una vez generados todos los coeficientes, los atributos son ordenados en función de éstos, obteniendo así el *ranking* que se buscaba. En función de cada métrica, los coeficientes de mayor valor serán asignados a los atributos más relevantes del problema, frente a otras métricas en las que el coeficiente más relevante será aquel que menor valor obtenga para la métrica. Existirán, por tanto, *rankings* ordenados de mayor a menor o al revés, aunque siempre los primeros atributos del *ranking* han de ser los más importantes dentro del conjunto de datos.

En las primeras posiciones de las "clasificaciones" de atributos tendremos atributos que despejan una cantidad grande de incertidumbre en el problema, mientras que en las zonas finales estarán aquellos atributos que parecen no tener apenas relación con el problema que se está abordando. Debemos tener en cuenta que al realizarse estas medidas de forma *univariada*, no se están teniendo en cuenta posibles relaciones o redundancias entre los propios atributos.

Una de las mayores aportaciones de este tipo de medidas es la rapidez con que realiza la clasificación. Conjugando esta técnica con alguna otra de determinación de punto de corte –presentada en el siguiente apartado– puede obtenerse un subconjunto de atributos "óptimo" para el problema.

En base al texto (Ben-Bassat, 1982) han sido extraídas siete métricas *filter* univariadas. Algunas de estas siete métricas planteaban un problema fundamental: en el texto se encontraban diseñadas para problemas donde la clase tan solo toma dos valores, es decir, para *problemas dicotómicos*. Para generalizar este esquema a *problemas multiclase* (más de dos estados para la variable clase), se optó por el cálculo de la métrica ponderando a cada clase mediante el producto de su probabilidad marginal. Es decir, para el cálculo del coeficiente global se calculan los coeficientes marginales dos a dos y se suman. En general,

$$Filter_{multiclase} = \sum_{i=1}^{r_c} \sum_{j=1}^{j<i} P(c_i) \cdot P(c_j) \cdot Filter(c_i, c_j)$$

donde $Filter(x, y)$ es la métrica original del problema *dicotómico*.

Información mutua. Una de las medidas de información más ampliamente utilizadas y contrastadas es el cálculo de la información mutua, o *mutual information* (Shannon, 1948). En base a la teoría de la información, calcula la relación que existe entre una variable y otra dada. La semántica de esta medida se centra en la cantidad de incertidumbre que el conocimiento de una variable es capaz de despejar con respecto al estado en el

que se encuentre la segunda. Esta medida se encuentra acotada en el intervalo [0,1]; valores cercanos a 1 marcan una alta correlación, mientras que, valores cercanos a 0 indican independencia entre las variables analizadas.

Su expresión matemática se presenta a continuación, donde: X es la variable a evaluar, C es la variable clase, r_x es el número de estados que X puede tomar, y r_c es el número de estados que C puede tomar.

$$I_p(X;C) = \sum_{i=1}^{r_x}\sum_{j=1}^{r_c} P(x_i,c_j)\log\frac{P(x_i,c_j)}{P(x_i)P(c_j)}$$

Distancia Euclídea. Partiendo de la formulación clásica de la distancia euclídea entre dos vectores, se generaliza la misma a un problema multiclase. En sí, esta métrica no deriva de ningún análisis de información o de dependencia probabilística, pero resulta un buen contraste con respecto a las demás técnicas.

Su expresión matemática, en base a los mismos parámetros que en el caso anterior, se formula como:

$$D_e(X;C) = \sqrt{\sum_{i=1}^{r_x}\sum_{j=1}^{r_c}\sum_{k=1}^{k<j} P(c_k)P(c_j) \;\mid P(x_i|c_k)^2 - P(x_i|c_j)^2 \mid}$$

Métrica Matusita. La formulación original de esta métrica mide la distancia entre dos distribuciones de probabilidad. Al generalizarse, se intenta medir la distancia media entre las diferentes distribuciones marginales de cada uno de los valores del atributo con los valores de la clase.

Su expresión matemática se formula como:

$$D_m(X;C) = \sum_{i=1}^{r_c}\sum_{j=1}^{j<i} P(c_i)P(c_j)\left[\sum_{k=1}^{r_x}\sqrt{P(x_k|c_i)P(x_k|c_j)}\right]$$

Divergencia de Kullback-Leibler. La divergencia de Kullback-Leibler es el método más conocido para la medición de distancias entre dos distribuciones de probabilidad. Su formulación genérica es:

$$D_{kl}\left(P(X), Q(X)\right) = \sum_{x_i} P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

Había que determinar cuáles eran las dos distribuciones que iban a ser utilizadas. Se tomaron dos posibilidades, comparar las probabilidades marginales a priori (modo 1), y las probabilidades condicionales a priori (modo 2). Primero debía ponderarse cada comparación con las probabilidades marginales de cada clase:

$$KL(X;C) = \sum_{i=1}^{r_c} \sum_{j=1}^{j<i} P(c_i)P(c_j)\ KL_{ij}(X;C)_1$$

La divergencia en modo 1 se formula como:

$$KL_{ij}(X;C)_1 = D_{kl}\left(P(X|c_i), P(X)\right) + D_{kl}\left(P(X|c_j), P(X)\right)$$

Y, en modo 2, como:

$$KL_{ij}(X;C)_2 = D_{kl}\left(P(X|c_i), P(X|c_j)\right) + D_{kl}\left(P(X|c_j), P(X|c_i)\right)$$

Entropía de Shannon. La entropía de Shannon es una de las medidas más conocidas y extendidas para la medición de la bondad de una variable determinada. En el presente trabajo, se adapta su clásica formulación dicotómica a una formulación multiclase:

$$Sh(X;C) = \sum_{i=1}^{r_c} \sum_{j=1}^{j<i} P(c_i)P(c_j)\ H_{ij}(X)$$

$$H_{ij}(X) = -\sum_{k=1}^{r_x} P(x_i|c_i)\log_2 P(x_i|c_j) + P(x_i|c_j)\log_2 P(x_i|c_i)$$

Métrica Bhattacharyya. Esta distancia mide la dependencia que existe entre dos distribuciones de probabilidad. Las distribuciones que van a ser comparadas son las probabilidades a priori de una variable, contra la condicionada a la clase. Intentamos de esta forma ver qué grado de dependencia encontramos entre ambas distribuciones; cuanto mayor sea ese grado, mayor será el peso de la variable analizada en el problema clasificatorio.

Su formulación, con los mismos parámetros que en los demás casos, es:

$$Bh(X;C) = \sum_{i=1}^{r_c} -\log\left[P(c_i)\sum_{j=1}^{r_x}\sqrt{P(x_j|c_i)P(x_j)}\right]$$

6.2.2. Umbrales de corte

El mayor problema que se nos presenta una vez realizado un ranking de atributos es decidir con cuántas de esas variables más importantes me quedo, empezando por la parte superior del ranking, y cuáles son las que rechazo por suponer que no aportan información alguna.

Existen diferentes alternativas para abordar el problema, pero todas ellas están basadas en el hecho de que, en algún momento, la ganancia en información que se produce por la adición de otra variable no es significativa.

Si se dibuja en una gráfica la cantidad de información acumulada frente al número de variables que se van añadiendo, se observa como esos puntos de inflexión crean un "codo" convexo. Será en el entorno de ese "codo" donde se encontrará el punto óptimo de corte de un ranking.

De las técnicas que intentan aproximar un resultado óptimo se ha seleccionado el método presentado en (Molina et al., 2002), al cual nos referiremos como *método del codo*, o *punto de corte óptimo*.

Este método va a intentar determinar automáticamente ese punto de corte fijando la posición de corte cuando exista una diferencia significativa entre la reducción de incertidumbre despejada por un atributo y el siguiente en la lista.

El algoritmo propuesto para ello se enuncia de la siguiente forma:

1. Eliminar los atributos con coeficientes más lejanos por abajo que dos veces la varianza total, desde el valor medio del estadístico que se esté computando.
2. Definir $s_i = w_i + w_{i-1}$ donde w_i es el valor del estadístico para el atributo i.
3. A partir de estos valores definir

$$\sigma_j = \sum_{i=2}^{j} s_i$$

4. El objetivo es buscar el atributo x_j tal que

$$K_{x_j} = 1 - \frac{\sigma_j}{\sigma_n}\frac{n-j}{n}$$

   sea máximo (n=número de atributos del conjunto).
5. El punto de corte estará entre el atributo x_j y x_{j+1}

Figura 5: Algoritmo utilizado en el *método del codo* para determinar el punto de corte.

Para ilustrar su funcionamiento, en el Cuadro 6 se incluye un breve ejemplo tomando una hipotética base de datos con nueve variables de la que se ha generado un ranking de atributos.

6.2.3. Selección de subconjuntos de variables

Mediante la aplicación de técnicas de punto de corte, un *ranking* de variables puede ser entendido como un subconjunto de atributos óptimo, es decir, un método FSS de selección de subconjuntos de atributos. Su gran inconveniente es que, al ser realizados estos rankings de forma univariada, pueden existir atributos redundantes entre sí. Para evitar este problema existen otra serie de aproximaciones *filter* al problema que lo abordan

Variable	*Métrica*	s_i	σ_j	K_j
Var1	1.7984809542417057	—	—	—
Var2	1.7941962125339819	3.592677167	3.592677167	0.9609580554
Var3	1.7297671578260516	3.523963370	7.116640537	0.9258852118
Var4	1.7252099049597018	3.454977063	10.57161760	0.8946908441
Var5	1.7111439076437815	3.436353813	14.00797141	0.8668023548
Var6	1.7076673894106351	3.418811297	17.42678271	0.8421846593
Var7	1.7012824251428578	3.408949815	20.83573253	0.8207478472
Var8	1.7010274222933854	3.402309847	24.23804237	0.8024522666
Var9	1.6967157792616443	3.397743202	27.63578557	0.7872729384
$\overline{w} =$	1.70045729870			
$\sigma^2 =$	0.000934090932			
$\overline{w} \pm 2 \times \sigma^2 =$	1.702325481 / 1.698589117			
Punto de corte $\rightarrow$	Var7 – Var8			

Cuadro 6: Ejemplo de funcionamiento del método del codo para determinar un umbral de corte en un ranking de variables.

desde un punto de vista multivariante.

Debido a su propia naturaleza, existen tantas técnicas de FSS *filter* como posibles heurísticos puedan encontrarse. Una de las medidas más utilizadas para calcular la bonanza de un subconjunto de atributos es la información mutua; encontramos ejemplos en (Torkkola, 2002; Xing et al., 2001). También pueden encontrarse artículos que utilizan como métrica heurística alguna otra de las presentadas anteriormente, por ejemplo, la métrica Bhattacharyya en (Aherne et al., 1997).

Existen otras aproximaciones que definen diferentes formalismos de relevancia para abordar el problema: MMR-FS (Maximal Marginal Relevance-FS) (Lee and Lee, 2003), RelFSS (Relevance-FSS) (Bell and Wang, 2000), Relief (Kira and Rendell, 1992), ReliefF (Kononenko, 1994), etc. Pero muchas de estas técnicas aplican fuertes restricciones al problema, como pueden ser el dar soporte sólo a atributos dicotómicos, o a problemas dicotómicos.

Una de las técnicas más recientes y que mejores resultados están obteniendo en diversos campos de aplicación, es el método descrito por Mark A. Hall de la Universidad de

Waikato[15] (Hall and Smith, 1997). Para una completa revisión sobre métodos de FSA el lector puede consultar el trabajo de (Molina et al., 2002), además de un número especial de la revista *JMLR*[16] sobre este campo (Guyon and Elisseeff, 2003).

CFS - Correlation-based Filter Selection. El método descrito en este punto (Hall and Smith, 1997) está basado en un heurístico muy intuitivo, combinado con una de las técnicas más sencillas dentro de las políticas clásicas de búsqueda, el ascenso de la colina o *hill-climbing*.

La descripción que se realiza en esta sección de la técnica se basa en la propuesta original de Hall. En base a esa propuesta los autores han realizado diferentes modificaciones que pueden ser consultadas en (Hall and Smith, 1999; Hall and Holmes, 2003). En la primera de ellas, la política de búsqueda utilizada pasa a ser *Best-First* e introducen búsquedas locales dentro de subconjuntos del conjunto de entrenamiento. En la segunda de ellas el criterio de parada en las políticas de búsqueda es eliminado. De esta forma, generan todo el árbol de búsqueda hasta incluir a todas las variables para, tras finalizar, escoger el subconjunto de atributos que mejor heurístico haya obtenido.

La mayoría de los algoritmos de selección de atributos realizan una búsqueda a través del espacio de posibles subconjuntos que puedan formarse. El tamaño de este espacio de búsqueda es exponencial sobre el número de atributos, de ahí que una búsqueda exhaustiva sea implanteable. Estrategias incrementales de *subida a la colina*, o *hill-climbing*, como son la selección hacia delante y la eliminación hacia atrás son utilizadas para realizar la búsqueda en un tiempo razonable. Aunque son simples, han demostrado obtener buenos resultados comparadas con técnicas más sofisticadas como *Best First* o *Beam Search*.

El *greedy hill-climbing* genera todas las posibles combinaciones a partir de un nodo y selecciona para continuar la búsqueda el hijo con mayor función de evaluación. Los

[15] *http://www.waikato.ac.nz*
[16] Journal of Machine Learning Research - *http://www.ai.mit.edu/projects/jmlr*

nodos son expandidos aplicándoles los operadores clásicos de búsqueda hacia delante o hacia atrás, ver Figura 6.

1. Sea $s \leftarrow$ el estado inicial
2. Expandir aplicando los operadores de búsqueda
3. Evaluar cada hijo t de s
4. Sea $s' \leftarrow$ el hijo t con mayor función de evaluación $e(t)$
5. Si $e(s') > e(s)$ entonces $s \leftarrow s'$, ir al paso 2
6. Devolver s

Figura 6: Pseudo código de una búsqueda *hill-climbing* hacia delante.

La selección hacia delante comienza con un conjunto vacío de atributos; la eliminación hacia atrás comienza con el conjunto completo de atributos. La selección hacia delante terminará cuando no haya ningún nodo hijo mejor que su padre, mientras que la eliminación hacia atrás seguirá iterando hasta que ningún hijo sea mejor que su padre.

Como la mayoría de técnicas de selección de atributos, CFS utiliza un algoritmo junto a una función de evaluación del mérito de los subconjuntos de atributos. El heurístico por el cual CFS mide la "bondad" de los subconjuntos tendrá en cuenta la usabilidad de los atributos individualmente para predecir la clase, junto con el nivel de intercorrelación bivariado (dos a dos) que haya entre ellos. La hipótesis sobre la cual se basa puede enunciarse como:

"Buenos subconjuntos de atributos contienen atributos altamente correlados (predictores de) la clase, y, a la vez, no relacionados (no predictivos) entre sí."

Los atributos seleccionados de esta forma serán atributos que formen "ejes perpendiculares", o relevantes, en el espacio probabilístico. Es decir, cada atributo seleccionado por el CFS formará un eje de discriminación distinto a los formados por el resto de atributos. La siguiente ecuación formaliza el heurístico:

$$G_s = \frac{k\overline{r}_{ci}}{\sqrt{k + k(k-1)\overline{r}_{ii'}}}$$

Donde k es el número de atributos en el subconjunto; $\overline{r}_{ci}$ es la correlación media con la clase, y $\overline{r}_{ii'}$ es la intercorrelación media entre ellos.

La expresión del heurístico es de hecho la *correlación de Pearson*, donde todas las variables han sido estandarizadas. El numerador puede ser visto como la medida de cuán predictiva de la clase puede ser un subconjunto de atributos dado; el denominador, como cuánta redundancia existe, bivariadamente, entre los atributos predictores. La bondad de este heurístico es que dejará fuera atributos irrelevantes, ya que serán malos predictores; y que los atributos redundantes serán ignorados ya que estarán altamente correlados con uno o más de los incluídos en el subconjunto.

La medida para medir estas correlaciones entre atributos y clases va a ser la *entropía condicional*. De tal forma, si X e Y son variables aleatorias discretas con rangos respectivos R_x y R_y, las siguientes ecuaciones nos presentan la entropía de Y antes, y después de haber observado a X.

$$H(Y) = -\sum_{y \in R_y} p(y)\log(p(y))$$

$$H(Y|X) = -\sum_{x \in R_x} p(x) \sum_{y \in R_y} p(y|x)\log(p(y|x))$$

Una medida de correlación o dependencia de Y sobre X, también llamado *coeficiente de incertidumbre* de Y, se define como:

$$C(Y|X) = \frac{H(Y) - H(Y|X)}{H(Y)}$$

Esta medida puede tomar valores entre 0 y 1. Un valor de 0 indica que X e Y no tienen asociación; el valor 1 indica que el conocimiento de X predice por completo a Y.

Si analizamos el algoritmo tal y como está enunciado por Hall, vemos que sus sucesivas iteraciones van a realizar un gran número de cálculos repetidos. Podemos simplificar la expresión de la correlación entre X e Y mediante el cálculo de su información mutua:

$$C(Y|X) = \frac{IM(X,Y)}{H(Y)} = \frac{IM(Y,X)}{H(Y)}$$

Asímismo, tras extender paso a paso las iteraciones de búsqueda hacia delante, se obtiene una expresión recurrente que minimiza el número de cálculos a realizar, lo que equivaldrá a una mayor eficiencia del algoritmo. La expresión de la correlación media entre los atributos seleccionados y la clase, $\bar{r}_{ci}$, puede expresarse como:

$$\overline{r}_{ci} = \frac{k-1}{k} \cdot \overline{r}'_{ci} + \frac{IM(C, A_{new})}{H(C) \cdot k}$$

La expresión de la intercorrelación media de los atributos, $\bar{r}_{ii'}$, puede expresarse también en función de la siguiente recurrencia:

$$\overline{r}_{ii'} = \frac{k-2}{k} \cdot \overline{r}'_{ii'} + \sum_{i=1}^{k-1} R(A_i, A_{new})$$

donde

$$R(A_i, A_j) = \frac{IM(A_i, A_j) \cdot [H(A_i) + H(A_j)]}{H(A_i) \cdot H(A_j)}$$

con A_{new} como el nuevo atributo incluído en el subconjunto, $\overline{r}'_{ci}$ y $\overline{r}'_{ii'}$ son los valores de las intercorrelaciones en la iteración anterior del bucle de búsqueda.

En la primera iteración de la búsqueda, la expresión del heurístico queda reducida al cálculo de la información mutua entre cada uno de los atributos y la clase, $IM(C, A_i)$; de forma que el conjunto inicial lo formará aquel atributo que mayor información mutua tenga con respecto a la clase.

Esta optimización del algoritmo ha sido producto del análisis teórico/práctico del

alumno durante el estudio bibliográfico efectuado a lo largo de la ejecución del proyecto.

Parte III

Elvira System

7. Introducción y revisión histórica

El sistema o entorno *Elvira* (Elvira Consortium, 2002) es una plataforma para la construcción y uso de modelos gráficos probabilísticos, que está siendo construída con el apoyo del *Ministerio de Ciencia y Tecnología*, actualmente integrado en el *Ministerio de Educación* español, bajo dos proyectos de investigación: Elvira (TIC97-1135-C04, 1997-2000) y Elvira II (TIC2001-2973-C05, 2001-2004).

La motivación original que hace surgir el proyecto es el objetivo de aunar los esfuerzos de los diferentes grupos de investigación españoles dentro del campo de los modelos gráficos probabilísticos. Debido a la amplitud de este campo, cada grupo de investigación estaba especializado en un tipo de problemática, teniendo sus algoritmos propios implementados en cada uno de los laboratorios. Esto hacía que existiera una gran cantidad de trabajo duplicado, y que la utilización de los métodos fuera casi de uso exclusivo de sus autores.

Por tanto, el principal objetivo del proyecto era la puesta en común de todos estos métodos en un entorno unificado. Este entorno debería incluir además, una interfaz de usuario amigable para la edición y uso de los diferentes modelos gráficos. El sistema Elvira surge *"from scratch"*, esto es, desde cero, ya que otro de los objetivos era que la herramienta fuera de libre distribución, anulando así la posibilidad de basarse en otro tipo de plataforma cerrada ya existente. Será Java[17] el lenguaje de programación utilizado para su implementación; su usabilidad, portabilidad y robustez se consideran características

[17] *http://java.sun.com*

apropiadas para llevar a cabo el proyecto.

Elvira es un proyecto abierto, no sólo porque su tiempo de vida aún no haya finalizado sino porque sus actualizaciones son muy frecuentes, incluso diarias en ciertas épocas del año. Actualmente se encuentran involucrados en este proyecto siete grupos de investigación de universidades españolas (UNED[18], UCLM[19], UPV-EHU[20], UG[21], UAL[22], UM[23], UJA[24]) y existen muchos otros grupos de investigación y particulares que aportan y colaboran en el mismo. La página web principal se encuentra en la dirección *http://leo.ugr.es/∼elvira*, donde puede ser descargado gratuitamente el código fuente, así como acceder a información más detallada sobre todos los aspectos del sistema.

[18]Universidad Española a Distancia - *http://www.uned.es*
[19]Universidad de Castilla-La Mancha - *http://www.uclm.es*
[20]Universidad del País Vaso - Euskal Herriko Unibersitatea - *http://www.ehu.es*
[21]Universidad de Granada - *http://www.ugr.es*
[22]Universidad de Almería - *http://www.ual.es*
[23]Universidad de Murcia - *http://www.um.es*
[24]Universidad de Jaén - *http://www.ujaen.es*

8. Dominios del sistema Elvira

8.1. Redes bayesianas

8.1.1. Aprendizaje

Elvira ofrece en su interfaz gráfico un soporte completo para la edición y uso de redes Bayesianas. De tal forma, el usuario puede crear y editar manualmente una red Bayesiana desde cero. La creación de una red desde cero puede resultar útil cuando el proceso se supervisa manualmente mediante conocimiento experto. Pero, como ya se comentó en la Sección 4, este método es lento, impreciso y obtiene resultados dispares.

Por contra, Elvira integra una colección de algoritmos de aprendizaje automático de redes Bayesianas desde datos. Presentados en la Sección 4, estos algoritmos pueden ser divididos en dos grandes grupos: los algoritmos de *score+search* que utilizan una estrategia de búsqueda guiada; y, los algoritmos que construyen la red en base a detecciones de independencias condicionales entre las variables.

Los algoritmos basados en funciones de *score+search* buscan el grafo que mejor modeliza a los datos de entrada, de acuerdo con un criterio específico. Todos ellos utilizan una función de evaluación junto con un método que mide la bondad de cada estructura explorada en el conjunto total de estructuras. Durante el proceso de exploración, la función de evaluación es aplicada para evaluar el ajuste de cada estructura candidata a los datos. Cada uno de estos algoritmos se caracteriza por la función de evaluación y el método de búsqueda utilizados. Actualmente, Elvira dispone de tres funciones de evaluación: K2 (Cooper and Herskovits, 1992), BIC (Schwarz, 1978) y BDeu (Heckerman et al., 1995). En cuanto a los métodos *score+search* que están implementados, nos encontramos con un algoritmo de Búsqueda Local (Heckerman et al., 1995), el algoritmo K2 (Cooper and Herskovits, 1992) y una versión distribuída de la Búsqueda en Vecindad

Variable (de Campos and Puerta, 2001). Con respecto al otro gran método de aprendizaje de estructuras, Elvira incluye el método PC (Spirtes et al., 1993), algoritmo ampliamente conocido.

Estas dos técnicas son utilizadas para la determinación de la estructura de la red Bayesiana y queda por tanto la estimación de los parámetros pendiente. Para esta tarea Elvira incluye dos opciones: procedimientos frecuenciales de máximos a posteriori, también llamados MLE (*Maximum Likelihood Estimation*) , y aproximaciones Bayesianas como la Aproximación de Laplace (Cooper and Herskovits, 1992).

Clasificadores Bayesianos Pero uno de los grandes campos dentro del aprendizaje de redes Bayesianas es su uso para resolver problemas de clasificación supervisada (ver Sección 5). En este campo Elvira contiene una batería de procedimientos y métodos de validación para dar un soporte global a la problemática supervisada.

En la familia de clasificadores "descendientes" del naïve Bayes, además del propio naïve Bayes (Minsky, 1961), encontramos el modelo selective-nB (Langley and Sage, 1994) y el modelo semi-nB (Pazzani, 1997). Elvira incluye también métodos de mayor complejidad, como son el *tree augmented naïve Bayes* o TAN (Friedman et al., 1997) y el *k-dependence Bayesian classifier* o kDB (Sahami, 1996). Dentro de cada uno de ellos podemos encontrar variaciones en las técnicas de aprendizaje de estructura y/o de parámetros, como técnicas directas e indirectas, o mediante tests de hipótesis.

8.1.2. Inferencia

Una tarea clásica dentro del campo de las redes bayesianas es la propagación de una evidencia o evidencias dada a lo largo de la red, pudiendo observar cómo varían los parámetros de dicha red en función de esos hechos o evidencias. Ésta va a ser la tarea de la *inferencia* dentro del sistema Elvira.

Prefijados una serie de hechos o evidencias, bien, mediante la interfaz gráfica de usuario, o bien, mediante un fichero de evidencias, el sistema propagará dichas evidencias a lo largo de la red, actualizando los parámetros de cada uno de los nodos. Para aquellos nodos que exista una observación directa, la actualización de los parámetros es directa, ya que con una seguridad total (P(hecho) = 1) se encuentran en un determinado estado.

Para el resto de los nodos existe un gran número de algoritmos que poder aplicar, dentro del sistema; destacamos tres de ellos, como son: *Hugin* (Jensen et al., 1990), *VariableElimination* (Zhang and Poole, 1996) y *Penniless Propagation* (Cano et al., 2000). Asímismo, el sistema permite al usuario la posibilidad de introducir múltiples hechos al mismo tiempo, visualizando conjuntamente las consecuencias de las diferentes propagaciones, que también podrán ser configuradas con algoritmos específicos para cada una de ellas.

8.1.3. Abducción

La abducción o inferencia abductiva puede definirse como "un proceso para obtener las explicaciones más plausibles a una serie de hechos observados". En sí, la abducción no es un proceso de inferencia sólido y las hipótesis obtenidas son sólo posibles explicaciones a los hechos observados.

En general, siempre se obtendrán varias explicaciones posibles; habrá que seleccionar las mejores. Una explicación debe "explicar" los hechos observados y, además, debe ser la más simple del conjunto de explicaciones generales. Una vez realizado este filtrado, deberemos seleccionar la más plausible de entre todas.

La abducción fue introducida en el campo de la Inteligencia Artificial por Pople en 1973 (Pople, 1973). En sistemas probabilísticos se plantea como la búsqueda de la configuración de valores para las variables no observadas que tenga máxima probabilidad. La mejor explicación será la configuración que maximiza la probabilidad P(configuración

| evidencia), siendo éste el criterio de selección de hipótesis utilizado. Al usar la regla de Bayes para realizar los cálculos, se utiliza la probabilida a priori de la explicación, así como la probabilidad de que la evidencia se produzca.

Dentro del sistema Elvira no sólo se puede buscar la explicación más probable (Nilsson, 1998), sino las k-primeras explicaciones más probables. También puede realizarse *abducción parcial* (de Campos et al., 2002), es decir, encontrar la explicación a un subconjunto de las variables no-observadas y no a todas a la vez.

En los procesos de abducción, el resultado es simplemente una "hipótesis", una posible explicación al hecho observado, y no una conclusión definitivamente cierta.

8.2. Diagramas de influencia

Los diagramas de influencia (IDs) aumentan las capacidades de las redes bayesianas con la posibilidad de tratar con eventos bajo el control del usuario (variables de decisión) y preferencias (funciones de utilidad). El sistema Elvira provee dos tareas fundamentales en el campo de los diagramas de influencia:

1. **Representación de IDs**: Los diagramas de influencia pueden ser creados utilizando la interfaz gráfica de usuario (GUI) o mediante la especificación detallada en la Sección 8.3. Existen métodos que automáticamente preparan los diagramas para poder ser evaluados, verificando las condiciones necesarias para ello.

2. **Evaluación de IDs**: Existen dos algoritmos para evaluar IDs: *ArcReversal* (Shachter, 1986) y *VariableElimination* (Jensen, 2001). No se encuentran disponibles desde la interfaz gráfica, pero pueden ser invocados desde la línea de comandos.

8.3. Formatos de ficheros

Dentro del sistema, las redes Bayesianas, los diagramas de influencia y las bases de datos de aprendizaje son definidos utilizando un *formato de ficheros* propio. Dicho formato está basado en texto plano o ASCII de forma que se utliza directamente para ser almacenadas con extensión `.elv` (redes y diagramas) o `.dbc` (bases de datos).

Redes Bayesianas y diagramas de influencia. Estructuralmente ambos elementos está configurados por un grafo de dependencias. En cada nodo de ese grafo existirán diferentes tipos de nodos y parámetros en función del elemento, red o diagrama, que se esté manejando. Una red contiene tres tipos de elementos: un conjunto de nodos, un conjunto de enlaces y un conjunto de relaciones entre los nodos.

Un nodo contiene información sobre una variable de la red. Dichas variables pueden tener un número finito de estados posibles (variable discreta) o pueden definirse sobre un dominio continuo (variable continua). Un enlace se especifica mendiante los dos nodos a los que une; pudiendo ser dirigido o no-dirigido.

Una relación describe un conjunto de variables conectadas de alguna manera, e incluye información numérica sobre las variables mediante un *potencial*. Existen diferentes alternativas para definir un *potencial* dentro de Elvira, mediante tablas de probabilidad, árboles de probabilidad, o funciones.

Para cada uno de los elementos presentados pueden definirse una serie de propiedades. Por ejemplo, para cada nodo, puede incluirse un título, un comentario, una posición dentro de la interfaz gráfica de usuario, la tipología del mismo (aleatorio, de decisión o de utilidad). Si el nodo es de tipo discreto, cada estado puede identificarse con nombres descriptivos. Si es de tipo continuo, se definirá el mínimo y máximo que puede tomar. Para los enlaces, además de indicar si es de tipo dirigido o no, se puede incluir un comentario aclaratorio. Por último, en las relaciones, se puede incluir un comentario, el tipo de

relación que abarca y la información númerica correspondiente al *potencial* que la define.

Estas propiedades son inicializadas a valores por defecto si no se especifica lo contrario. Como ejemplo, se incluye la siguiente especificación completa de una red bayesiana para la detección de metástasis en un cáncer (Cooper, 1984):

```
bnet Cancer {
     title="Diagnosis of metastatic cancer";
     author="Greg Cooper";
     whochanged="Elvira Consortium";
     default node states=(absent present);
node A { title="Metastatic cancer";
     comment="It is ill or not"; }
node B { title="Serum calcium";
     states=(normal high); }
node C { title="Brain tumor";
     states=(present absent); }
node D; node E;
link A B; link A C; link B D;
link C D; link C E;
relation A {
     coment = "Probabilities for metastasis";
     kind-of-relation=conditional-prob;
     values= table (0.2 0.8); }
relation B A {
     values = table ([high,present]=0.8, [high,absent]=0.2,
     [normal,present]=0.2, [normal,absent]=0.8); }
relation C A {
     values = table (0.05 0.2 0.95 0.8); }
relation E C {
     values = table (0.8 0.6 0.2 0.4); }
relation D B C {
     values = tree (
```

```
          case D{
               present=case B {
                    high=0.8;
                    normal=case C{
                         present=0.8; absent=0.05; }
               }              absent=case B {
                    high=0.2;
                    normal=case C{
                         present=0.2; absent=0.95; }
               }
          }
     );
}
}
```

Además de las propiedades individuales para cada elemento, existen propiedades generales para la red, como pueden ser su título, comentario, autor, versión, etc.

Bases de datos. El formato expuesto hasta ahora varía un poco al tratar el sistema con bases de datos. Para las bases de datos no tiene sentido alguno el definir enlaces, potenciales o relaciones; pero sí, el contenido de información de dicha base de datos. Para una base de datos, la extensión por defecto del fichero tomará el nombre `dbc`, acrónimo en minúsculas de *Data Base Cases*.

Las propiedades generales serán semejantes a las definidas para las redes y diagramas. Al comienzo del fichero se definirá el conjunto de nodos o variables de las que consta el problema, tipología, comentarios e información adicional que pueda resultar de interés. También deberá incluirse el número de instancias o casos que existen en el fichero, número que tendrá que coincidir con el número de líneas que más tarde se incluyan como información.

Una vez definidas todas las variables se incluyen los casos que conforman el cuerpo de conocimiento de la base de datos. Cada instancia corresponderá con una línea, y, por cada línea, se incluirá el valor de esa instancia para cada una de las variables definidas en el preámbulo del fichero. Los valores que tome cada variable pueden ser separados bien por comas, o bien dejando un espacio en blanco.

En el caso de variables continuas se incluye el valor dentro del intervalo definido para ese nodo; en caso de ser continua, se incluirá el nombre del estado que toma para esa instancia. En el ejemplo siguiente se incluye una pequeña base de datos con tres variables, dos discretas y una continua:

```
data-base sampleDBC {
     number-of-cases = 7;
     node Var01(finite-states) {
          title = "Atributo 1";
          comment = "Equal Frequency Discretized [12.25, 13.28]";
          kind-of-node = chance;
          type-of-variable = finite-states;
          relevance = 10.0;
          purpose = ;
          num-states = 3;
          states = (low medium high);
     }
     node Var02(finite-states) {
          title = "Atributo 2";
          comment = "Equal Frequency Discretized [0.74, 1.51,
                     1.73, 2.13, 3.43]";
          kind-of-node = chance;
          type-of-variable = finite-states;
          relevance = 10.0;
          purpose = ;
          num-states = 5;
```

```
          states = (s0 s1 s2 s3 s4);
     }
     node Var03(continuous) {
          title = "Atributo 3";
          kind-of-node = chance;
          type-of-variable = continuous;
          relevance = 10.0;
          purpose = ;
          min = 278;
          max = 1680;
          precision = "1402";
     }
     relation {
          memory = true;
          cases = (
               [ high, s0, 1035 ]
               [ high, s2, 1015 ]
               [ medium, s4, 845 ]
               [ low, s2, 278 ]
               [ high, s3, 1680 ]
               [ medium, s4, 1515 ]
               [ medium, s1, 990 ]
          );
     }
}
```

9. Descripción de las clases principales

9.1. Diagrama de clases

La Figura 7 recoge el diagrama de las clases básicas que conforman el sistema Elvira. Las clases incluídas en la figura confeccionan el núcleo o "corazón" de la plataforma, sobre las que se han ido construyendo los diferentes paquetes.

9.2. Paquetes principales del sistema

A lo largo de esta sección se realiza una breve descripción de los paquetes que integran Elvira. Han sido detallados clases y componenetes de esas clases que resulten de especial importancia.

elvira Este es el paquete base sobre el que se encuentra construída toda la arquitectura de clases del sistema. Contiene entre otras, las clases que permiten almacenar en memoria una red Bayesiana o un diagrama de influencia. Entre las clases que contiene podemos destacar:

- Clase `Graph` – Representa un grafo (dirigido o no) con una lista de nodos y una lista de enlaces:

```
protected NodeList nodeList;
protected LinkList linkList;
private int kindOfGraph;
  (DIRECTED, UNDIRECTED, MIXED)
```

- Clase `Network` – Deriva de `Graph` y representa un grafo al que se le añade una lista de relaciones:

```
private Vector relationList;
  Vector de relaciones de la red
```

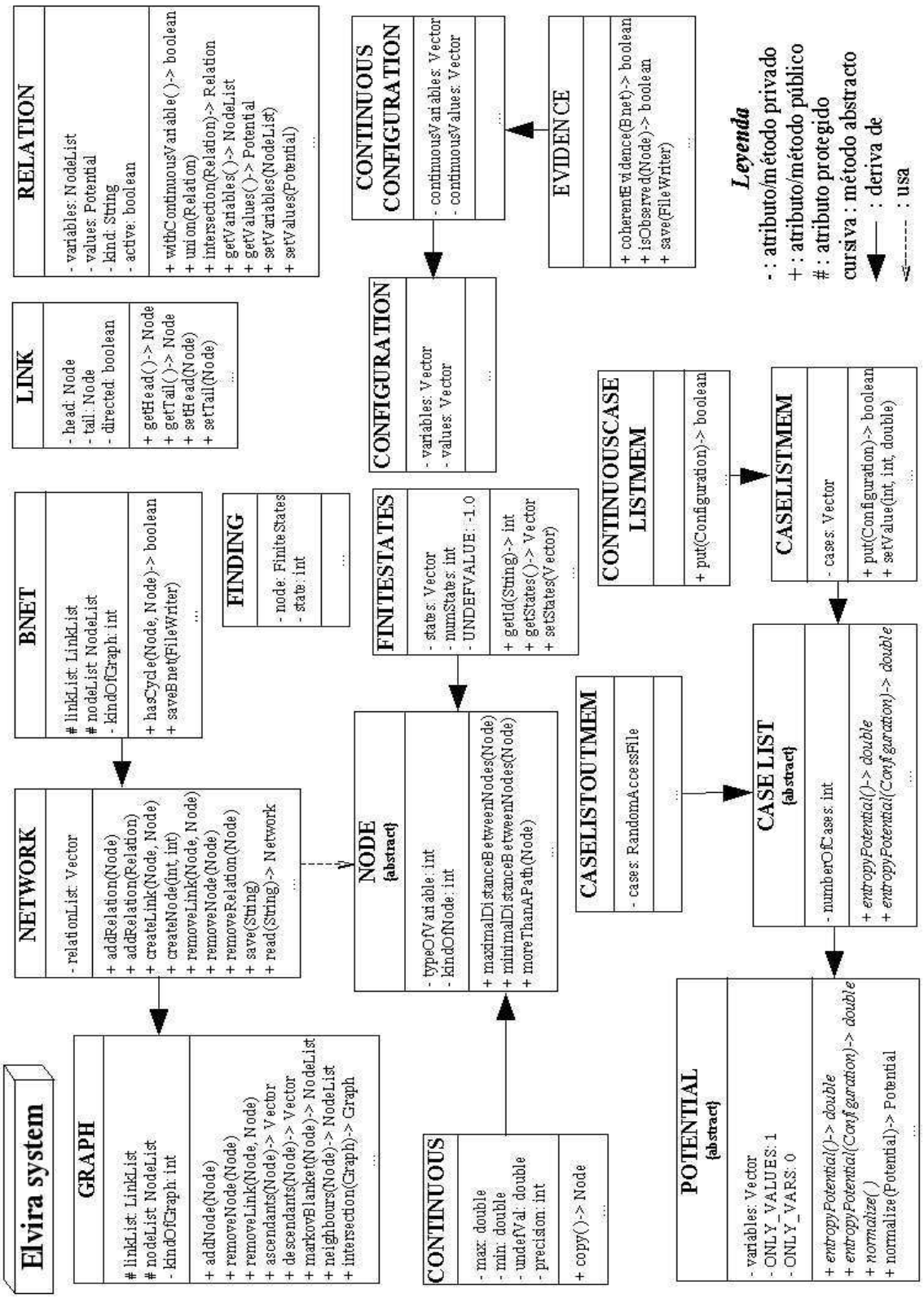


Figura 7: Clases y métodos principales de la plataforma Elvira.

- Clase `Bnet` – Deriva de `Network` y es la clase encargada de almacenar una red Bayesiana.
- Clase `Node` – Clase abstracta utilizada para almacenar un nodo de un objeto `Network`. Las subclases de ésta contienen los distintos tipos de nodos, `FiniteStates` y `Continuous`.

```
private int typeOfVariable;
  Puede ser CONTINUOUS, FINITE_STATES, o MIXED
private int kindOfNode;
  Puede ser CHANCE, DECISION o UTILITY
```

- Clase `FiniteStates` – Deriva de `Node` y es la encargada de almacenar nodos correspondientes a variables aleatorias (`kindOfNode = CHANCE`) con un número finito de posibles valores (`typeOfVariable = FINITE_STATES`); también es utilizada para nodos de decisión (`kindOfNode = DECISION`).

```
private Vector states;
  Vector de strings con los nombres de los estados
private int numStates;
  Número de estados
private static final double UNDEFVALUE;
  Valor de una variable discreta cuyo valor es
  desconocido. Todas las variables discretas usan
  el mismo valor (-1.0) para indicar valor desconocido
```

- Clase `Continuous` – Deriva de `Node`, encargada de almacenar los nodos correspondientes a variables aleatorias con dominio continuo (`kindOfNode = CONTINUOUS`); también es utilizada para nodos de utilidad (`kindOfNode = UTILITY`).

```
private double max;
private double min;
```

```
private double undefVal;
  Valor de esta variable continua cuyo valor es
  desconocido. Cada variable continua puede tener
  un valor diferente en este campo
```

- Clase `Link` – Almacena un enlace entre dos nodos de un grafo (`Graph`).

```
private Node head;
private Node tail;
private boolean directed;
  Flag para indicar si es dirigido o no
```

- Clase `Relation` – Representa una relación entre variables en la que aparecen las variables implicadas junto con información numérica representado con un `Potential`.

```
private NodeList variables;
  Lista de variables de la relación
private Potential values;
  Potencial de la relación
private String kind;
  Tipo de relación que se define
private boolean active;
```

- Clase `Configuration` – Implementa una configuración de variables discretas. O sea una lista de variables `FiniteState` y una lista de valores enteros (estados).

```
private Vector variables;
private Vector values;
```

- Clase `ContinuousConfiguration` – Deriva de `Configuration`, almacena una configuración de variables discretas y continuas.

```
private Vector continuousVariables;
private Vector continuousValues;
```

- Clase `Evidence` – Deriva de `ContinuousConfiguration` y representa un conjunto de evidencias (observaciones) en una red Bayesiana.

```
public boolean isObserved(Node node);
public save (java.io.FileWriter f);
```

- Clase `Finding` – Representa una observación para una sola variable `FiniteStates`. Es usada sobre todo en las clases del interfaz gráfico.

```
private FiniteStates node;
private int state;
```

- Clase `CaseList` – Deriva de `Potential`; se trata de una clase abstracta que representa una lista de casos de una red Bayesiana, definiendo como caso el conjunto formado por un valor para cada una de las variables.
- Clase `CaseListMem` – Deriva de `CaseList` y representa una lista de casos almacenados en memoria para variables discretas.

```
private Vector cases;
```

- Clase `ContinuousCaseListMem` – Deriva de `CaseListMem` y representa una lista de casos almacenados en memoria para variables tanto discretas como continuas.

elvira.potential Contiene las clases que permiten almacenar información numérica para un conjunto de variables. Entre las clases que contiene podemos destacar:

- Clase `Potential` – Clase abstracta que actúa como superclase de los distintos tipos de potenciales.

```
private Vector variables;
public Potential addVariables (FiniteStates var);
public Potential combine (Potential pot);
public Potential restrictVariable (Configuration conf);
```

- Clase `PotentialTable` – Deriva de `Potential` y representa el potencial para un conjunto de variables discretas, representado como una tabla, con un valor por cada configuración de las variables.

```
private double[] values;
```

- Clase `PotentialTree` – Deriva de `Potential` y representa el potencial para un conjunto de variables discretas, representado como un árbol de probabilidad.
- Clase `PotentialFunction` – Deriva de `Potential` y representa el potencial para un conjunto de variables discretas, representado mediante una función.

```
private Vector arguments;
  Vector de argumentos de la función. Un argumento
  puede ser un Double u otro Potential
private Function function;
  Es la función concreta usada para obtener los valores
  del potencial a partir del valor de las variables del
  potencial y de los argumentos de la función
public void setFunction(String s);
public double getValue(Configuration conf);
```

- Clase `ListPotential` – Deriva de `Potential` y representa un potencial a través de una lista de potenciales. Se supone que el potencial global se obtienen con la combinación de todos los de la lista.
- Clase `PotentialContinuousPT` – Deriva de `Potential` y representa una función de probabilidad que depende de variables discretas y continuas.

Es como un `PotentialTree` donde los nodos internos pueden representar variables discretas y continuas, y los nodos hoja son funciones de densidad *mixtura de exponenciales truncadas*, o MTE.

elvira.parser Contiene las clases para los métodos de entrada salida automática o "parsers" de ficheros Elvira, ficheros de evidencia, ficheros de bases de datos y ficheros de resultados de propagación. Estas clases son generadas con el programa `javacc` a partir de ficheros `.jj` que describen la sintaxis y semántiva de los diferentes formatos de ficheros.

Cuando el sistema *Elvira* lee un fichero propio `.elv` se crea un objeto de la clase `Bnet` o bien `IDiagram`. Al leer un fichero de evidencia `.evi` se crea un objeto `Evidence`. Si el fichero leído contiene una base de datos `.dbc` se creará un objeto `DataBaseCases`.

elvira.database Contiene la clase `DataBaseCases` que deriva de `Bnet`. Esta clase representa una base de datos de casos para una red Bayesiana. Las bases de datos pueden estar conformadas tanto por variables continuas como discretas. Además, es posible utilizar el símbolo `?` en el fichero fuente cuando no conocemos el valor de una variable para algún caso.

Los casos son almacenados encapsulados en una estructura `Bnet`. Los nodos de esta red serán las variables leídas en el fichero fuente, y el valor de los casos se encontrará encapsulado en una relación definida para la primera variable.

elvira.inference Contiene la clase `Propagation` que es la superclase de todos los métodos de propagación en redes Bayesianas y diagramas de influencia. Esta clase contiene métodos que permiten calcular el error de la propagación en métodos aproximados, siempre que se disponga de los exactos. Las propagaciones pueden también ser leídas y almacenadas en ficheros.

elvira.inference.elimination Contiene clases que implementan métodos de propagación por el método de eliminación de variables en redes Bayesianas y diagramas de influencia.

- Clases `VariableElimination` y `VEWithPotentialTree` – Clases que realizan la propagación mediante el método exacto de eliminación de variables en redes y diagramas.
- Clase `ArcReversal` – Método de inversión de arcos para diagramas de influencia.

elvira.inference.clustering Contiene las clases para almacenar en memoria un árbol de grupos o cliques, así como métodos de triangulación necesarios para construirlo.

- Clase `JoinTree` – Representa un árbol de grupos.
- Clase `NodeJoinTree` – Representa uno de los nodos (grupos) del árbol.

```
public NeighbourTreeList neighbourList;
  Representa la lista de vecinos del grupo
```

- Clase `Triangulation` – Contiene métodos de triangulación de grafos, usados en la construcción del árbol de grupos.

Además, contiene clases que implementan métodos exactos y aproximados de propagación en tales árboles de grupos o cliques.

- Clase `HuginPropagation` – Implementa el algoritmo de propagación Hugin. Puede usar, de forma opcional, árboles de probabilidad, permitiendo propagación exacta o aproximada.
- Clases `Penniless` y `SimplePenniless` – Implementan el algoritmo *penniless* de propagación aproximada. Este algoritmo puede realizar más de

dos recorridos por el árbol de grupos con la intención de mejorar cada vez más la aproximación conseguida. Utiliza árboles de grupos binario y árboles de probabilidad.

- Clases `LazyPenniless` y `SimpleLazyPenniless` – Variación sobre el algoritmo *penniless* utilizando listas de potenciales.

elvira.inference.approximate Contiene clases que implementan métodos aproximados de propagación en redes Bayesianas mediante algoritmos de *MonteCarlo*.

- Clase `SimulationProp` – Superclase de los siguientes métodos de propagación. Contiene datos y métodos comunes a todos.
- Clases `ImportanceSampling`, `ImportanceSamplingTable`, `ImportanceSamplingTree`, `ImportanceSamplingTreeAV` e `ImportanceSamplingFunctionTree` – Implementan distintos tipos del método de propagación de *muestreo por importancia* (semejante al algoritmo de eliminación de variables).
- Clases `MarkovChainMonteCarlo` y `ContinuousMCMC` – Implementan algoritmos de *MonteCarlo* con cadenas de Markov.

elvira.inference.abduction Contiene clases que implementan métodos exactos y aproximados de abducción total y parcial en redes Bayesianas.

elvira.inference.super_value Contiene clases que implementan los algoritmos de inferencia en Diagramas de influencia con nodos super valor (nodos que se generan como uniones o productos de otros nodos). Incluye los algoritmos de *VariableElimination*, y *ArcReversal* extendidos a nodos super-valor.

elvira.fusion Contiene clases para llevar a cabo fusión de redes Bayesianas.

elvira.translator Contiene diferentes clases para poder importar ficheros de otras aplicaciones sobre redes Bayesianas, o de otros formatos. Ejemplos son un traductor del formato `bif` o del formato `Hugin`.

elvira.tools Clases auxiliares para realizar monitorizaciones de tareas, estadísticas, definición de funciones complejas, etc.

elvira.learning Contiene las clases necesarias para el aprendizaje de redes Bayesianas mediante los algoritmos de *score+search*.

- Clase `Metrics` – Clase abstracta que declara los métodos que deben poseer todas las posibles métricas que vayan a evaluar, a partir de una base de datos, una red Bayesiana o un conjunto de nodos.

```
private DataBaseCases data;
  Contiene los datos contra los que se evaluará
  la métrica
public abstract double score (Bnet b);
  Método genérico que evaluará el ajuste de la reb b
  a los datos, en base a la métrica implementada
```

- Clases `BDeMetrics`, `BICMetrics`, `K2Metrics`, `KLMetrics` – Implementan la clase `Metrics`, contienen las métricas BDE, BIC, K2 y Kullback-Leibler para evaluar el ajuste de una red Bayesiana a un conjunto de datos dado.
- Clase `Learning` – Clase abstracta que deben implementar todos los algoritmos de aprendizaje *score+search*.

```
private Bnet output;
  Red Bayesiana resultado del aprendizaje
public abstract void learning();
  Método genérico que será el encargado de realizar el
  algoritmo de aprendizaje correspondiente
```

- Clases `DVNSSTLearning`, `PCLearning`, `K2Learning`, `K2GALearning`, `BICLearning`, `MTELearning` – Implementan la clase `Learning`, contienen los algoritmos de apendizaje por vecindad variable, PC, K2 y K2 guiado por algoritmo genético, BIC y aprendizaje utilizando mixturas de exponenciales truncadas.

elvira.learning.classification Contiene las clases básicas que un objeto que vaya a actuar como clasificador debe utilizar e implementar. Clases accesorias para calcular las probabilidades a priori de las variables de un fichero de datos, métodos de validación y proceso genérico de clasificación.

- Clase `AuxiliarPotentialTable` – Clase auxiliar que facilita el cálculo de las frecuencias relativas o probabilidades condicionadas de una variable cuando se estiman desde una base de datos.

```
private int nStatesOfVariable;
  Número de estados que puede tomar la variable
private int nStatesOfParents;
  Número de posibles configuraciones que pueden tomar
  sus padres
private double [][] numerator;
  numerator(i,j) almacena el número de casos para los
  que la variable toma su i-ésimo valor y sus padres
  toman su j-ésima configuración
private double [] denominator;
  denominator(j) almacena el número de casos en los
  que los padres de la variable toman su j-ésima
  configuración
public void addCase(int stateOfVariable,
      int stateOfParents, double quantity)
  quantity indica cuántas ocurrencias deben ser
```

```
    añadidas a la ocurrencia de los estados indicados
    por los parámetros
  public double getPotential(int stateOfVariable,
       int stateOfParents)
    Devuelve la probabilidad de que la variable tome su
    stateOfVariable-ésimo valor cuando sus padres toman
    su stateOfParents-ésima configuración
```

- Clase `Classifier` – *Interfaz* que se define para estandarizar las técnicas de clasificación que vayan a ser implementadas dentro del sistema. Consta de dos métodos que todo clasificador debe implementar, el algoritmo de aprendizaje, y la técnica de clasificación.

```
public void learn (DataBaseCases training,
     int classnumber);
  El parámetro classnumber indica el índice de la
  variable en el conjunto de atributos que conforman
  la base de datos training
public Vector classify (Configuration instance,
     int classnumber);
```

- Clase `ConfusionMatrix` – Clase dedicada a gestionar las matrices de confusión de los procesos de clasificación/validación de un clasificador (objeto que debe implementar la interfaz `Classifier`).

```
private double[][] confusionMatrix;
private double[][] confusionMatrixVariances;
```

- Clase `ClassifierValidator` – Clase encargada de realizar los procesos de validación de un clasificador (objeto que debe implementar la interfaz `Classifier`). Las validaciones recogidas actualmente es una validación en `k`-hojas, o una validación `leave-one-out`. La información de cada uno de los procesos de clasificación internos de cada hoja puede ser exportada mediante métodos accesores.

```
public ClassifierValidator(Classifier classifier,
      DataBaseCases dbc, int classvar)
public ConfusionMatrix kFoldCrossValidation (int k)
public ConfusionMatrix leaveOneOut()
```

elvira.learning.classificationtree Contiene la clase que implementa los árboles de clasificación en base a árboles de probabilidad en Elvira.

- Clase `ClassificationTree` – Implementa los árboles ID3, C4.5 y Dirichlet(probabilidades basadas en la distribución de Dirichlet). Sólo soporta atributos discretos.

```
public ClassificationTree(int bm, int pm, double cl)
  Constructor de la clase, bm indica el tipo de árbol
  a construir, pm el método de poda de ramas, cf es
  el nivel de confianza si se selecciona EBP como
  método de poda
public void learn (DataBaseCases training,
      int classnumber)
public Vector classify(Configuration instance,
      int classnumber)
  Aunque no implementa directamente la interface
  Classifier, sí que implementa dos métodos semejantes
  a los definidos en ella
```

elvira.learning.classification.supervised.discrete Contiene las clases necesarias para aprender modelos clasificatorios bayesianos supervisados. Los clasificadores disponibles son naïve Bayes, Selective naïve Bayes, Semi naïve Bayes, tree augmented naïve Bayes, k dependences Bayesian classifier y restricted partial DAG classifier. Todos ellos aplicados sólo sobre variables discretas.

- Clase `DiscreteClassifier` – Clase abstracta que implementa la interfaz `Classifier`, es la clase padre de todos los clasificadores discretos imple-

mentados en este paquete. Contiene los atributos y métodos principales de todos los modelos.

```
protected Bnet classifier;
  El modelo aprendido de la base de datos
public DiscreteClassifier(DataBaseCases data,
      boolean lap)
  Constructora de la clae, lap indica si debe aplicarse
  corrección de Laplace al cálculo de los parámetros
public abstract void structuralLearning()
  método abstracto que los clasificadores hijo de esta
  clase deben implementar para realizar el proceso de
  aprendizaje de la estructura Bayesiana
public void parametricLearning()
  Método encargado del aprendizaje paramétrico
public void categorize(String inputFile,
      String outputFile)
  Método que categoriza automáticamente un fichero
  entero de casos
```

- Clases `Naive_Bayes`, `SelectiveNaiveBayes`, `SemiNaiveBayes` – Derivan de `DiscreteClassifier` e implementa todos el aprendizaje estructural, el aprendizaje parámetrico es realizado por la clase padre al ser todos modelos Bayesianos.

- Clases `TAN`, `KDB` – Derivan de `DiscreteClassifier`, son las clases padres de todos los modelos TAN o kDB que vayan a ser implementados. El método de aprendizaje estructural está vacío de forma que serán las clases que deriven de ellos las que deban implementarlo en función del algoritmo que utilicen para realizar dicho aprendizaje.

- Clase `CMutInfTAN`, `CMutInfKDB` – Derivan de `TAN` y `KDB` e implementa

el aprendizaje paramétrico de ambos modelos mediante el cálculo de dependencias condicionales utilizando la información mutua como métrica.

- Clase `RPDAGClassifier` – Deriva de `MarkovBlanketLearning` y es la clase que implementa los clasificadores RPDAG. En base al *Markov blanket* realiza una búsqueda local en el espacio de los RPDAG.

elvira.learning.classification.supervised.mixed Contiene las clases correspondientes a los clasificadores mixtos que actualmente soporta Elvira. Un clasificador mixto se define como un modelo clasificatorio que posee variables predictoras continuas y dicretas, siendo su variable clase exclusivamente discreta.

- Clase `MixedClassifier` – Deriva de `DiscreteClassifier` y es la clase abstracta padre de todos los modelos clasificatorios mixtos implementados. En base a la estructura creada por la clase `DiscreteClassifier` realiza las modificaciones necesarias para dar soporte a las variables predictoras continuas.
- Clase `Mixed_Naive_Bayes` – Deriva de `MixedClassifier` y es la clase padre de todos los modelos mixtos en los que vaya a suponerse independencia condicional entre las variables predictoras dada la variable clase. Esta clase implementa el aprendizaje estructural del modelo dejando para sus subclases el aprendizaje paramétrico.
- Clases `Gaussian_Naive_Bayes`, `MTE_Naive_Bayes` – Derivan de `Mixed_Naive_Bayes` e implementa los aprendizajes paramétricos cuando las distribuciones de probabilidad de las variables continuas se suponen Gaussianas y cuando se suponen procedentes de mixturas de exponenciales.
- Clase `Selective_GNB` – Deriva de `Gaussian_Naive_Bayes` e implementa un clasificador mixto naïve Bayes Gaussiano con selección de varia-

bles. Sobreescribe los métodos de aprendizaje estructural y de evaluación de su super clase.

elvira.learning.classification.unsupervised.discrete Contiene las clases necesarias para aprender modelos clasificatorios no supervisados, o clusterings, con variables discretas. Actualmente puede aprenderse un modelo naïve-Bayes estimando los parámetros de la red mediante el algoritmo EM.

- Clase `UnsupervisedNBayes` – Clase abstracta que implementa un clasificador discreto no supervisado, utilizando una estructura de tipo naïve-Bayes.

```
protected int numberOfClusters;
  Número de particiones a realizar
public UnsupervisedNBayes(DataBaseCases dataCases,
      int numberOfClusters)
  Constructora que toma el fichero de datos
  y el número de particiones a efectuar en ellos
public abstract double learning();
  Método genérico de aprendizaje de los parámetros
```

- Clase `NBayesMLEM` – Deriva de `UnsupervisedNBayes` e implementa la estimación de los parámetros mediante el algoritmo EM o *Estimation Maximization*.

```
public double learning(boolean laplaceCorrection)
  Método que implementa el algoritmo EM, el
  parámetro indica si debe utilizar la corrección
   de Laplace al calcular las probabilidades
```

elvira.learning.constraints Contiene las clases necesarias para gestionar las restricciones aplicables a una red Bayesiana. Incluye algoritmos de aprendizaje 'score+search' que soporten restricciones en sus procesos de aprendizaje.

- Clase `ConstraintKnowledge` – Clase dedicada al almacenamiento de restricciones en tres grafos diferentes. Cada grafo contiene restricciones de diferente tipo: ausencia de enlaces, presencia de enlaces u orden parcial.

```
Graph existence;
Graph absence;
Graph order;
```

- Clase `PCLearningCK`, `DVNSSTLearningCK` – Derivan de la clase `Learning` e implementan los algoritmos de aprendiaje de redes Bayesianas mediante los algoritmos PC y vecindad variable con restricciones de ausencia, presencia y orden parcial.

elvira.learning.preprocessing Contiene las clases que implementan tareas que se realizan antes de un proceso de aprendizaje automático. Los métodos de este paquete no trabajan con redes, sino sobre objetos `dbc`, es decir, sobre los datos.

- Clase `Imputation` – Clase que implementa los diferentes algoritmos de imputación de valores perdidos/desconocidos sobre un conjunto de datos. Los algoritmos disponibles actualmente son: ceros (sustituye por 0 el valor perdido), media (calcula la media del atributo al que pertenece el valor perdido), media condicionada a la clase (idem que el anterior pero usando sólo las instancias de la misma clase, útil en problemas supervisados), árboles de clasificación (creará un árbol ID3, C4.5 o Dirichlet para calcular el valor perdidoe) y MPE (imputación basada en técnicas de abducción).

- Clase `Discretization` – Implementa todos los algoritmos de discretización de variables continuas disponibles en el sistema. Actualmente dispone de cinco algoritmos de discretización: misma frecuencia, misma anchura, suma de diferencias cuadradas, test monotético no-supervisado y k-Medias.

- Clase `FilterMeasures` – Clase correspondiente a la selección de variables, se detalla en la Sección 12.1.

elvira.localize Este paquete no contiene ninguna clase java, está formado por los ficheros que utiliza la interfaz gráfica para dar soporte multilingüe a la aplicación. En cada fichero se encuentran los strings utilizados en los diferentes componentes gráficos, botones, etiquetas, diálogos, etc. Actualmente se da soporte al lenguaje castellano e inglés.

elvira.gui El último de los paquetes raíces de la plataforma corresponde a toda su amplia interfaz gráfica. Debida a su complejidad y a que parte del trabajo desarrollado en este proyecto se centra sobre este paquete se incluye una sección específica dedicada a él –ver Sección 10–.

10. Interfaz gráfica de usuario

10.1. Introducción y ejemplos

Quizás la parte menos estructurada del sistema Elvira es su interfaz gráfica. La mayoría de desarrolladores del proyecto utilizan para sus experimentaciones la interfaz por línea de comandos, de forma que no se ha prestado la suficiente dedicación a una correcta implementación de la interfaz gráfica de usuario, *GUI*.

Ante un usuario que someta a pruebas no demasiado complejas a la interfaz, el sistema quedará bloqueado debido a la escasez de control de errores de la que dispone. Si bien no se espera un usuario "malintencionado" en este sentido y el trabajo a través de la interfaz es cada vez más amigable. Poco a poco, la interfaz va creciendo integrando cada un mayor número de los métodos disponibles en el sistema, y corrigiendo errores en su lógica de control.

El esquema de aplicación gráfica utilizado en Elvira es el MDI, o *Multiple Document Interface*, esquema que permite trabajar simultáneamente con varias redes o diagramas de influencia. La Figura 8 recoge un proceso de propagación de evidencias sobre una red bayesiana.

La interfaz gráfica puede trabajar en dos modos completamente diferenciados, en *modo Edición* o en *modo Inferencia*. En *modo Edición* puede editarse por completo el grafo que constituye la red, creando o borrando nodos, enlaces, o modificando los parámetros de probabilidades de cada nodo. Al pasar a *modo Inferencia* aparecen nuevos componentes en la barra de herramientas, y cambia el aspecto de los nodos, desplegándose los estados de cada uno. Este modo está destinado a realizar tareas de propagación y abducción presentadas en la Secciones 8.1.2 y 8.1.3.

El acceso a los métodos de *Aprendizaje Automático* se realiza a través del diálogo implementado por la clase `DataBaseMonitor.java`. En la Sección 12.2 se hace una

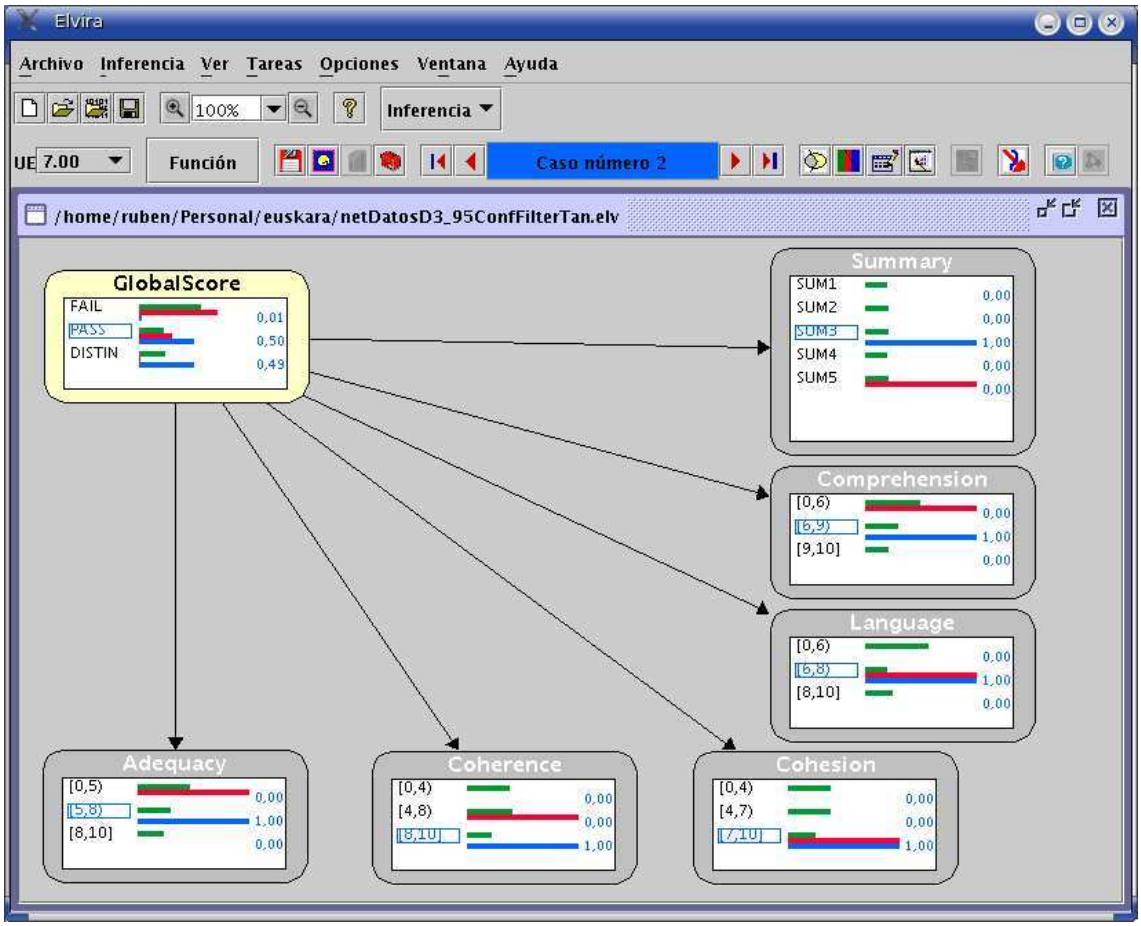


Figura 8: Interfaz Gráfica de Usuario (GUI) de Elvira: ejemplo 1.

breve revisión de las posibilidades ofrecidas por la interfaz mediante este cuadro diálogo. La Figura 9 muestra un clasificador de tipo TAN tras haber sido aprendido desde un fichero de casos.

10.2. Diagrama de clases de la interfaz

Dentro del paquete `elvira.gui` encontramos todas las clases utilizadas en la interfaz gráfica. El número de clases que contiene este paquete, así como los subpaquetes que contiene es bastante elevado debido al gran número de cuadros de diálogo, botones, imágenes y componentes gráficos que deben ser implementados. La mayoría de estos

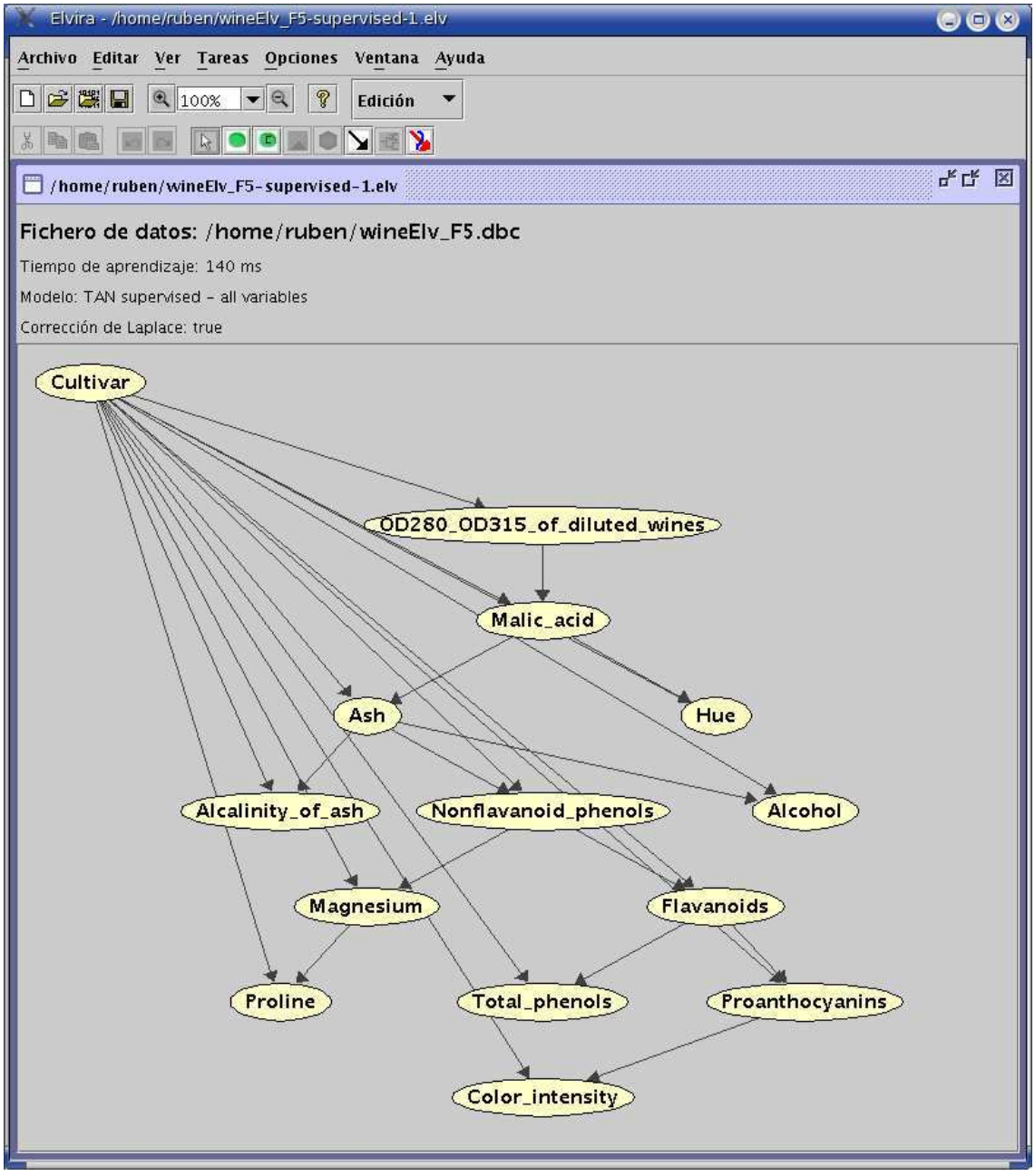


Figura 9: Interfaz Gráfica de Usuario (GUI) de Elvira: ejemplo 2.

componentes derivan de las clases que proporciona la plataforma Java para implementar interfaces gráficas, denominada JFC o *Java Foundation Classes*.

Si centramos el análisis en la interfaz gráfica principal del sistema, podremos destacar cuatro clases principales, dedicadas a gestionar la ventana de la aplicación.

ElviraFrame Deriva de `javax.swing.JFrame` (JFC) y corresponde con la ventana principal de la aplicación. Cuando la aplicación es ejecutada se instancia esta clase; al tratarse Elvira de una aplicación MDI sólo podrá existir una instancia de ella en ejecución. Recoge la barra de menús, y barras de botones necesarios en función del modo en el que se encuentre la aplicación.

MessageFrame Deriva de `javax.swing.JInternalFrame` (JFC) y será la subventana en la que se mostrarán los mensajes informativos auxiliares al usuario. Estos mensajes puede ser, por ejemplo, la correcta grabación de una red bayesiana en disco, la carga de un diagrama de influencia, o diferentes tipos de errores, entre otros.

DesktopPane Deriva de `javax.swing.JDesktopPane` (JFC) y corresponde con el panel transparente que vemos tras las subventanas de la aplicación. No tiene ninguna función especial en Elvira, tan solo el ser el medio de comunicación entre la aplicación y las subventanas que se encuentren abiertas en un momento determinado.

NetworkFrame Deriva de `javax.swing.JInternalFrame` (JFC) y corresponde con las ventanas de trabajo de la aplicación; éste será el componente clave de la interfaz gráfica. Se trata de un componente bastante “pesado” ya que va a contener en su interior un amplio reperterio de instancias de clases auxiliares, necesarias para realizar todas las opciones incluídas en el sistema.

En la Figura 10 se recoge la correspondencia visual de estas cuatro clases con los componentes que implementan.

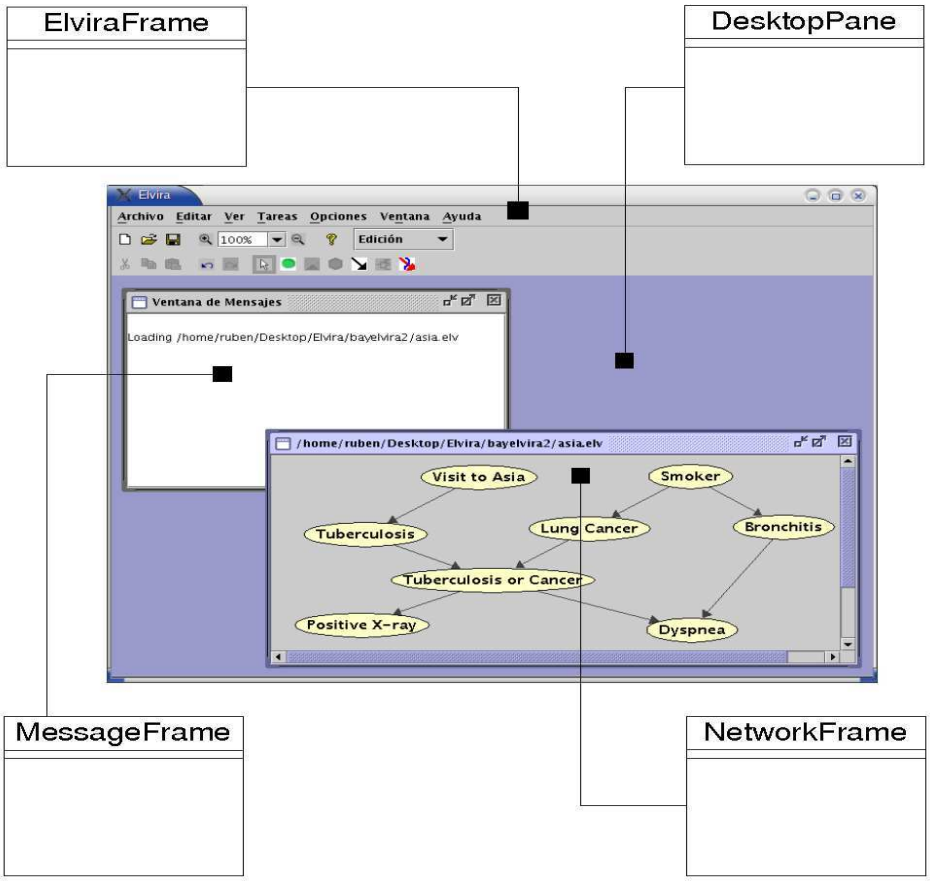


Figura 10: Clases principales de la interfaz gráfica de Elvira.

En función del modo, o de las operaciones a realizar sobre una ventana de trabajo, un `NetworkFrame`, éste dispone de diferentes paneles que encapsulan la información necesaria en cada momento para dichas tareas. Para ello, en Elvira, se define un tipo especial de panel genérico, denominado `ElviraPanel`. Los componentes claves que gestionan la edición de redes mediante la interfaz gráfica son lso siguientes:

- `protected Bnet bayesNet` – Red Bayesiana sobre la que se está trabajando actualmente con el panel.

- `protected Node currentNode` – Nodo actualmente seleccionado de la red; si no hay ningún nodo seleccionado, su valor será nulo (*null*).

- `protected Link currentLink` – Enlace actualmente seleccionado de la red; al igual que para los nodos, si no hay ninguno seleccionado su valor será nulo.

- `public Bnet` **`getBayesNet()`** – Método que al ser invocado devolverá el objeto `bayesNet` con la actual red Bayesiana de trabajo.

En base a este panel genérico se definen en Elvira cinco paneles diferentes, cada uno de ellos encargado de asumir diferentes tareas. En la Figura 11 se incluye un esquema de estas cinco clases.

InferencePanel Deriva de ElviraPanel y será el componente encargado de realizar todas aquellas tareas correspondientes a los procesos de inferencia. Cuando la interfaz principal funciona en *modo Inferencia*, éste es el panel que aparece visible.

EditorPanel Deriva de ElviraPanel y es el panel mostrado en el modo Edición de la interfaz principal. Es el panel encargado de la gestión de eventos del ratón, movimiento de nodos, selecciones, y demás tareas relacionados con la edición de las redes.

LearningPanel Deriva de ElviraPanel y encapsula toda la información y tareas necesarias para realizar el aprendizaje de redes Bayesianas en base a los métodos de *score+search*. Es un componente oculto, sin ningún tipo de comportamiento visual.

GenerateDBCPanel Deriva de ElviraPanel y su función es la de muestrear la red actual generando un fichero de casos que se ajuste a los parámetros de dicha red. Este panel no va a tener ningún comportamiento visual, siendo sólo utilizado cuando quiera realizarse un muestreo sobre la red de trabajo.

ConstraintKnowledgePanel Deriva de ElviraPanel y tampoco va a tener un comportamiento visual de cara al usuario. Almacena todos los datos sobre las restricciones que pueden ser configuradas para una red. Los métodos de aprendizaje que soporten restricciones en sus algoritmos de aprendizaje leerán la información de dichas restricciones en el panel. El usuario podrá configurar estas restricciones a través de la interfaz gráfica, o cargarlas de un archivo especial.

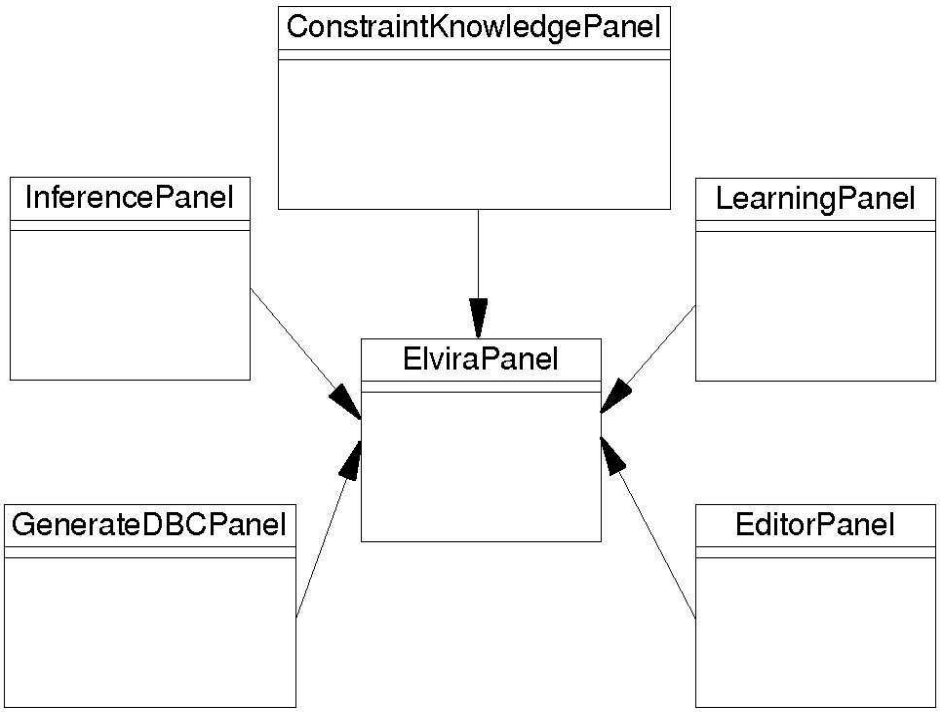


Figura 11: Tipos de paneles derivados de *ElviraPanel*.

Cada `NetworkFrame` contiene uno de estos cinco paneles, dándole así las capacidades para realizar las tareas correspondientes. Cuando se trabaja con la interfaz gráfica, el 95 % de la implementación gira entorno a este componente, ya que contiene todos los métodos y objetos necesarios para realizar casi todas las tareas soportadas por el interfaz gráfico de Elvira.

Cada vez que se instancia un nuevo `NetworkFrame` creamos una nueva ventana de trabajo, instanciándose los correspondientes paneles en su código:

```
private EditorPanel networkEditorPanel =
    new EditorPanel();
private InferencePanel networkInferencePanel =
    new InferencePanel();
private LearningPanel networkLearningPanel =
    new LearningPanel();
private GenerateDBCPanel GeneratePanel =
    new GenerateDBCPanel();
private ConstraintKnowledgePanel CKPanel =
    new ConstraintKnowledgePanel();
```

Parte IV

Implementaciones dentro del sistema Elvira

Como se comentó en la Sección 1.1, el presente proyecto se encuentra directamente relacionado con el trabajo habitual del alumno dentro del grupo de investigación. Debido a ello, las implementaciones recogidas a lo largo de este capítulo entroncan no sólo con la parte de selección de variables, sino con otras tareas paralelas llevadas a cabo, en especial en cuanto a la interfaz gráfica de usuario se refiere. Aún cuando puedan aparecer elementos paralelos, los contenidos desglosados en las siguientes secciones son aportaciones directas del presente proyecto al sistema Elvira.

11. Diseños realizados

Como ha podido ser constatado en la introducción a la plataforma Elvira a lo largo del Capítulo III, el sistema constituye una completa amalgama de clases y funcionanilidades que abarcan un amplio campo dentro de los modelos gráficos probabilísticos. A la hora de diseñar las nuevas funcionalidades que debían ser añadidas a la aplicación, dos premisas fueron fundamentales:

1. Conseguir la mayor generalidad posible, de forma que las capacidades de extensión del sistema no se vieran coartadas.

2. Respetando esa búsqueda de generalidad, intentar que el diseño fuera sencillo y claro.

Ambos objetivos son en cierta manera contrapuestos, aunque realmente son complementarios –cuanto más claro sea un diseño más fácilmente podrá ser ampliado y mejorado–.

11.1. Selección de variables

Hasta la iniciación de este proyecto, dentro del sistema Elvira, no existía soporte para la selección de variables. Si bien existían métodos que realizaban intrínsicamente en sus algoritmos algunos tipos de selecciones sencillas, no había una parte dedicada exclusiva e independientemente a ello.

La propuesta era clara, incluir dentro del paquete `elvira.learning.preprocessing` las clases necesarias para realizar estos procesos, de forma que pudieran ser utilizados de forma independiente o integrados en implementaciones futuras de nuevos métodos de aprendizaje.

Por tanto, el diseño o diagrama de clases a realizar debía ser lo más amplio posible, incluyendo métodos directos e indirectos, rankings de variables y selección de subconjuntos, políticas de búsqueda, etc. Tras un análisis de estos requisitos, se propuso el diseño reflejado en la Figura 12.

Este diseño inicial no se consideraba definitivo, sino un esqueleto de lo que podría ser el diseño final, una vez que hubiera sido consensuado entre todos los desarrolladores interesados en este campo. A continuación se incluye una breve descripción de cada clase:

FeatureSelection Clase abstracta padre de todas las demás clases de la jerarquía. El propósito de esta clase es servir de contenedor a todas los métodos de selección de variables. Debe especificarse el fichero de casos `dbc` que va a ser analizado. Está formada por un método abstracto que deberá ser implementado por todas sus subclases, `executeSelection()`. De esta forma, al instanciar a alguna de sus subclases, invocando este método, será realizada la selección de variables.

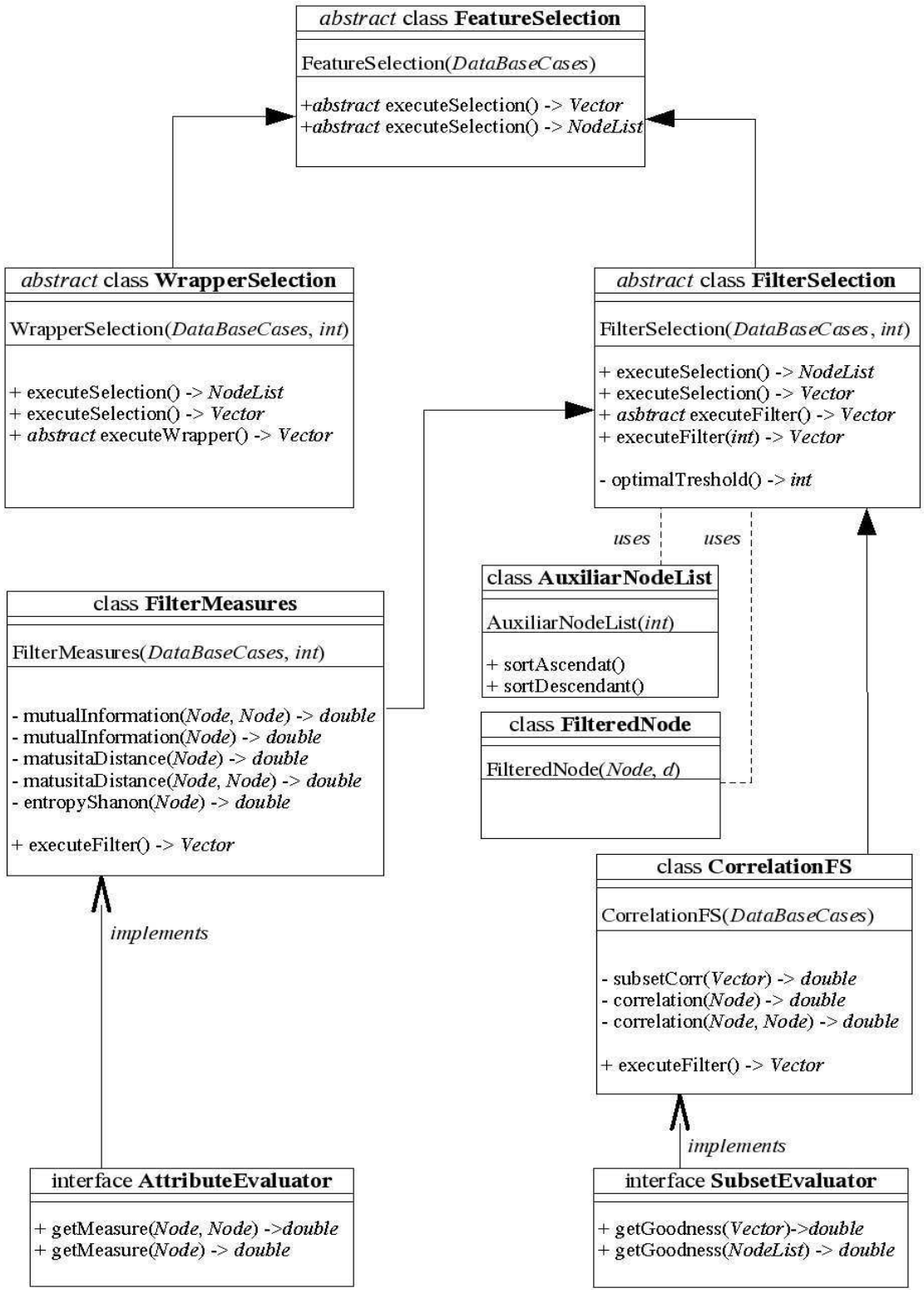


Figura 12: Diseño global propuesto para la selección de variables dentro del paquete `elvira.learning.preprocessing`.

```
public FeatureSelection(DataBaseCases);
public abstract NodeList executeSelection();
  NodeList contendrá los nodos seleccionados
```

WrapperSelection Deriva de `FeatureSelection` y es también una clase abstracta. Será la clase padre de todos los métodos que utilicen metodología directa o `wrapper` en la selección de atributos (ver Sección 6.1.1). Incluye un método abstracto que deberá ser implementado por sus subclases, y será el encargado de realizar las búsquedas.

```
public WrapperSelection(DataBaseCases, int);
  El parámetro entero identifica el método directo
  que instancia esta clase. Deberá ser indicado por
  la subclase
public abstract void executeWrapper();
```

FilterSelection Deriva de `FeatureSelection` y también es abstracta. Su estructura es semejante a la clase anterior, pero en el campo de las aproximaciones indirectas o *filter*. Podrían ser incluídas en su código las funciones de determinación de puntos de corte. Contiene también un método abstracto que deberá ser implementado por sus subclases.

```
public FilterSelection(DataBaseCases, int);
public abstract void executeFilter();
public int optimalThreshold();
```

FilteredNode, AuxiliarNodeList Clases auxiliares que pueden ser utilizadas por las diferentes clases como apoyo a las tareas que deben realizar. La primera trabaja con tuplas *(nodo, valor)*, y la segunda utiliza listas de tuplas, realizando ordenaciones.

FilterMeasures Deriva de `FilterSelection` y es la clase donde se encuentran las métricas indirectas univariadas. Implementa el método abstracto `executeFilter`

y, será la clase a instanciar para realizar un ranking de variables en problemas supervisados.

CorrelationFS Deriva de `FilterSelection` y viene a ser un ejemplo de cómo deberían ser implementados los métodos de selección de subconjuntos en esta arquitectura de clases. Se encarga de realizar la selección de atributos mediante el método *CFS*. Implementa también el método abstracto `executeFilter()`, a través del cual se invocará la búsqueda.

AttributeEvaluator Interfaz[25] que recoge los métodos que debe tener siempre una clase que utilice una métrica o distancia entre variables.

```
public double getMeasure(Node, Node);
  Calculará la métrica univariada ente dos nodos
public double getMeasure(Node);
  Calculará la métrica ente un nodo y el nodo clase
```

SubsetEvaluator Interfaz que recoge los métodos que debe tener siempre una clase que realice una selección de subconjuntos de atributos, mediante técnicas de búsqueda en el espacio de las variables. Dado un subconjunto de variables querríamos saber cuál es el grado de bonanza de dicho subconjunto.

```
public double getGoodness(NodeList);
```

Existían una serie de consideraciones de diseño sobre la propuesta realizada, que debían tenerse en cuenta :

- El diseño propuesto daba sólo soporte a atributos discretos, aunque podía ser fácilmente ampliado a atributos continuos.

[25] Las interfaces en Java son el medio de implementar herencia múltiple. Son clases vacías que tan solo definen métodos. Estos métodos deberán ser implementados por las clases que los utilicen.

- Podían ser calculadas métricas o distancias entre dos atributos de forma independiente a los problemas supervisados, siempre que dicha métrica pudiera ser aplicada en esos casos, por ejemplo, la información mutua entre dos variables.

- Las selecciones no sólo podían ser realizadas durante la ejecución del sistema, sino que podían ser guardadas en disco proyectando los ficheros de casos originales.

- Para los métodos de generación de rankings podían existir diferentes métodos automáticos que identificaran puntos de corte, generando así un subconjunto de atributos óptimos.

- Los métodos de selección de variables que utilicen búsquedas deberían implementar internamente los posibles métodos de búsqueda.

Tras la puesta en común de la propuesta realizada se decide "aparcar" su posible implementación. El número de horas necesarias para la completa implementación, o incluso para la implementación tan solo de la parte de medidas indirectas, se estima alto, escapándose por completo a los tiempos fijados para el proyecto. La propuesta es comunicada a los desarrolladores del sistema *Elvira* y queda pendiente de modificación y futura implementación.

De tal forma que la implementación de la selección de atributos va a realizarse de manera *monolítica*. Con este concepto queremos plasmar el hecho de que todo va a residir en una clase principal, `FilterMeasures`, que va a contener todas los componentes necesarios para una selección *filter*. Cualquier clase paralela que hiciera falta será implementada embebida dentro de ella. La propuesta final, junto con alguno de sus métodos y atributos, se recoge en la Figura 13. En la Sección 12.1 se presentan los métodos fundamentales de dicha clase.

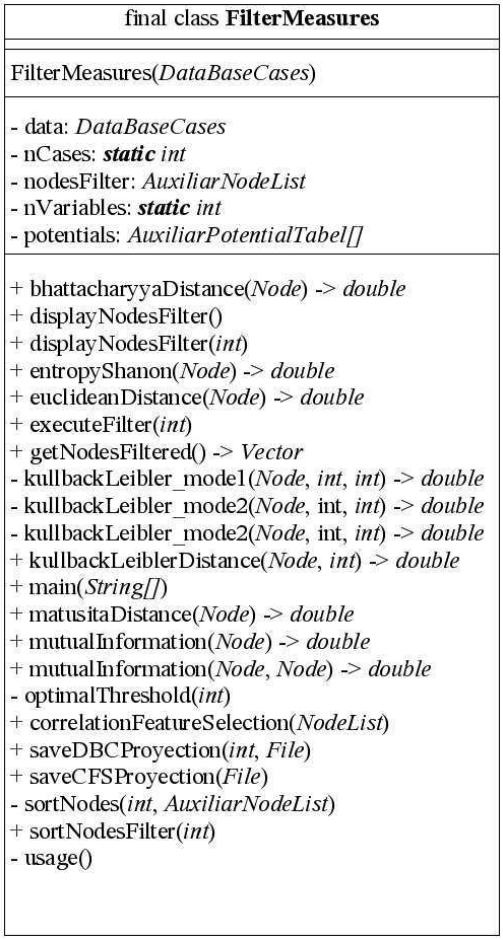


Figura 13: Diseño final implementado para la selección de variables del paquete `elvira.learning.preprocessing`.

11.2. Interfaz gráfico

A la hora de realizar el diseño de las clases que iban a integrar la selección de variables en la interfaz de Elvira tuvo que tenerse en cuenta el hecho que esta parte no podía ser separada de cualquier otro proceso relacionado con los ficheros de casos `dbc`.

El tratamiento de ficheros de casos se integraba en la interfaz de Elvira como una pequeña utilidad dentro de uno de los menús de la ventana principal. El objetivo propuesto era darle entidad propia a este tratamiento, ya que las técnicas que iban integrándose en Elvira para el aprendizaje desde datos crecía en gran manera. Junto a ello, uno de los puntos que más pesaban sobre las decisiones de diseño era el hecho de que quería realizarse un diseño que no supusiera un cambio grande en la estructura de clases ya creada para la interfaz gráfica.

De ahí que se decide crear un diálogo específico que reuna todas las opciones posibles en el tratamiendo de ficheros de casos. Este diálogo será accesible a través de una nueva entrada en el menú *Archivo* de la aplicación, llamado *Abrir fichero de casos*. De igual forma, se añade un nuevo botón en la barra de tareas principal que da acceso al mismo diálogo.

La clase que implementa este diálogo se llama `DataBaseMonitor` –monitor de bases de datos– ya que es el referente para todas las opciones relacionadas con las bases de datos. Al igual que el diseño para la selección de variables, su diseño interno es *monolítico*, embebiendo en su interior las clases necesarias para su funcionamiento. En la Figura 14 se muestran las clases de las que está compuesta.

Acotándonos al ámbito de la selección de variables, `DataBaseMonitor` incluye una pestaña en la parte de pre-procesamiento dedicada en exclusiva a ellas. Una vez seleccionado el fichero de casos, podremos seleccionar el algoritmo, visualizando en el mismo diálogo el estadístico que calcula, configurar las opciones disponibles (crear una proyección, utilizar el punto de corte, etc.) y ver los resultados obtenidos. Para un com-

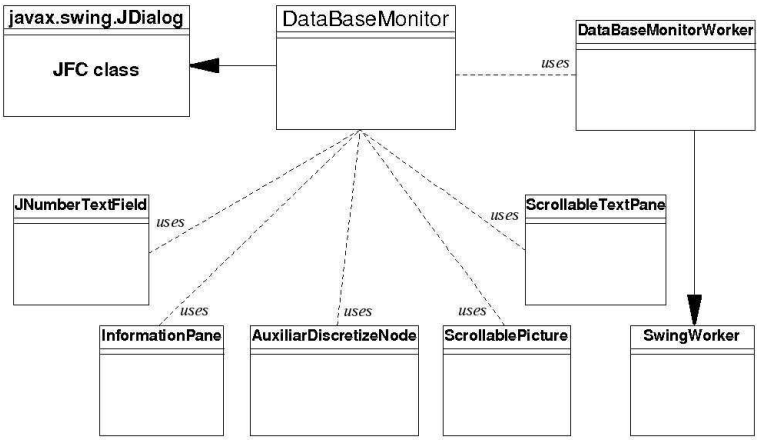


Figura 14: Clases integradas y utilizadas por `elvira.gui.DataBaseMonitor`.

pleto ejemplo visual del proceso se remite al lecto al Apéndice B que contiene un ejemplo completo del proceso.

12. Implementaciones

Para información más detallada sobre el funcionamiento interno de las clases enumeradas en esta sección, se remite al lector al código fuente del sistema, disponible en la dirección de Internet *http://leo.ugr.es/cgi-bin/cvsweb/bayelvira2/elvira/*.

12.1. La clase *FilterMeasures.java*

Esta clase está dedicada a realizar las tareas de selección de variables dentro del sistema *Elvira*. Pertenece, por tanto, al paquete `elvira.learning.preprocessing` al abarcar tareas de pre-procesamiento de bases de datos, previas a realizar un proceso de aprendizaje automático.

Su invocación puede ser realizada mediante la línea de comandos clásica, o desde la interfaz gráfica de usuario. Desde la línea de comandos dispone de una breve ayuda explicativa de cómo deben ser dispuestos los parámetros en la invocación a cada uno de los métodos. Es a través de la línea de comandos donde existe un mayor número de posibilidades en su ejecución. El acceso a través de la interfaz es realizado por la clase `DataBaseMonitor` que será detallada en la Sección 12.2.

Debido al diseño monolítico propuesto en la Sección 11, la clase `FilterMeasures` contiene una serie de clases auxiliares empotradas dentro de su propio código. A continuación se describen brevemente cada una de ellas:

- `FilteredNode` – Clase auxiliar que almacena parejas *nodo-valor*, donde *nodo* corresponde a la variable y *valor* a la métrica filter univariada computada para esa variable.
- `AuxiliarNodeList` – Clase que implementa una lista de `FilteredNode` y es la encargada de ordenarlos en función de la métrica seleccionada.
- `SearchState` – Clase encargada de realizar el seguimiento sobre el proceso de búsqueda de una selección *CFS*.

- `SearchParameters` – Clase que inicializa todos los parámetros necesarios para realizar una selección mediante la técnica *CFS*.

A continuación se detallan los métodos fundamentales que la clase `FilterMeasures` provee para llevar a cabo las diferentes selecciones de atributos ilustradas en la Sección 6.2:

- **`FilterMeasures`**`(DataBaseCases` *`cases`*`)` – Única constructora de la clase, toma como parámetros un objeto `DataBaseCases` en el que deberá haber sido cargado un fichero de casos `dbc`.

  La constructora supondrá que la variable clase del problema es la última que encuentra en la base de datos. Si la base de datos contiene algún elemento continuo o existen atributos con valores perdidos, la constructora retornará un error.

- `double` **`mutualInformation`**`(Node` *`variableX`*`, Node` *`variableY`*`)` – Método que computa la información mutua que existe entre el nodo representado por *variableX* y el nodo representado por *variableY* (ver Sección 6.2.1). Esta función puede ser utilizada independientemente a un proceso de selección de variables en un problema supervisado.

- `double` **`mutualInformation`**`(Node` *`variable`*`)` – Método que computa la información mutua entre el nodo representado por *variable* y el nodo clase de la base de datos configurada en la constructora de la clase. Es decir, el método calculará la información mutua entre el nodo *variable* y el nodo correspondiente al último en la base de datos, la clase.

- `double` **`euclideanDistance`**`(Node` *`variable`*`)` – Método que computa la distancia euclídea entre el nodo representado por *variable* y el nodo clase (ver Sección 6.2.1).

- `double` **`matusitaDistance`**`(Node` *`variable`*`)` – Método que computa la distancia Matusita entre el nodo representado por *variable* y el nodo clase (ver Sección 6.2.1).

- `double` **`kullbackLeiblerDistance`**`(Node` *`variable`*`, int` *`mode`*`)` – Método que computa las divergencias de Kullback-Leibler entre el nodo representado por *variable* y el nodo clase (ver Sección 6.2.1). El segundo parámetro de este método puede tomar los valores `1` o `2` en función del modo de divergencia que se quiera calcular.

- `double` **`entropyShanon`**`(Node` *`variable`*`)` – Método que computa la entropía de Shanon entre el nodo representado por *variable* y el nodo clase (ver Sección 6.2.1).

- `double` **`bhattacharyyaDistance`**`(Node` *`variable`*`)` – Método que computa la distancia Bhattacharyya entre el nodo representado por *variable* y el nodo clase (ver Sección 6.2.1).

- `void` **`executeFilter`**`(int` *`filterMeasure`*`)` – Método que genera internamente los rankings de todos los nodos del problema en base a la métrica indicada en el parámetro *filterMeasure*. Este método computa la métrica indicada para cada uno de los nodos, y, tras ello, los ordena en orden decreciente de importancia generando así el ranking.

  Hay que precisar que en algunas de las medidas la ordenación óptima busca incrementar el valor de la métrica, mientras que para otras el comportamiento es el contrario, será mejor aquel nodo que menor valor de su distancia tenga con la clase. Este método es la referencia básica a la hora de generar un ranking de variables.

- `int` **`optimalThreshold`**`(int` *`measure`*`)` – Método que calcula automáticamente el punto de corte dentro de un ranking de variables, en función del método del codo presentado en la Sección 6.2.2. Deberemos indicar al método qué medida filter debe ser aplicada mediante el parámetro *measure*.

- `void` **`saveDBCProyection`**`(int` *`cut`*`, java.io.File` *`file`*`)` – Método que almacena en el fichero configurado para el parámetro `file` la proyección del fichero original en función de un ranking y un punto de corte *cut*, fijado para dicho ranking. Dicha proyección contendrá los nodos que estén por encima de *cut* en el ranking de variables, y, además, como último nodo del fichero, el nodo clase del problema supervisado.

Este método es básico a la hora de realizar una selección de variables univariada, bien en función de un punto de corte fijado por el usuario, o bien mediante el determinado automáticamente. Facilita la labor de poder acceder a los resultados de las selecciones de atributos realizadas e ir estudiando los resultados obtenidos.

- `NodeList` **`correlationFeatureSelection`**`(NodeList` *`originalNodes`*`)` – Método principal que realiza la selección del subconjunto de atributos óptimo en función de la metodología *CFS* presentada en la Sección 6.2.3. Como parámetro toma un objeto `NodeList` que corresponde a los nodos del fichero de casos que haya sido configurado en la constructora de la clase.

  Una vez de realizado el proceso de búsqueda y selección, el método devuelve otro objeto de tipo `NodeList` en el que se incluyen sólo los atributos que han sido seleccionados como óptimos para ese conjunto de datos.

- `void` **`saveCFSProyection`**`(java.io.File` *`f`*`)` – Método que almacena en el fichero configurado para el parámetro *f* la proyección del fichero original incluyendo sólo aquellos nodos que han sido seleccionados previamente por el método *CFS*.

El proceso de búsqueda del método *CFS* puede ser planteado como una búsqueda hacia delante, o bien, hacia atrás (Hall and Smith, 1997). La actual implementación de éste método sólo abarca la búsqueda hacia delante; si bien, el esqueleto creado está preparado para soportar la búsqueda hacia atrás, pudiendo ser implementada sin mayores dificultades.

La documentación generada por el sistema automático de la plataforma Java, *javadoc*, puede ser consultada en la dirección *http://leo.ugr.es/∼elvira/devel/ElviraDoc2/*.

12.2. La clase *DataBaseMonitor.java*

Junto con la clase `ElviraFrame` correspondiente a la ventana principal de la interfaz gráfica del sistema, la clase `DataBaseMonitor` es una de las más grandes en

cuanto a líneas de código; actualmente, 3458 líneas. Pertenece al paquete `elvira.gui` y se trata de un gran panel en el que se encuentran encapsulados un gran número de componenetes y controles visuales, botones, barras, imágenes, textos, etc.

En ella se encuentran integradas todas las posibilidades que la plataforma *Elvira* dispone para el tratamiento de ficheros de bases de datos o `dbc`. Está dividida en tres grandes pestañas que contienen subdivisiones posteriores en función de las tareas a realizar. De tal forma, la primera división diferencia entre tareas de *Preprocesamiento*, tareas de *Aprendizaje automático*, o tareas de *Post-aprendizaje*.

Las medidas desarrolladas a lo largo del presente proyecto pertenecen al *Preprocesamiento*, como ya ha sido comentado con anterioridad. Dentro de las tareas de preprocesamiento existen otras dos subtareas disponibles, la *Imputación* de valores perdidos y la *Discretización* de variables continuas. Las opciones correspondientes a selección de atributos se encuentran bajo la subpestaña *Medidas filter*; en la Figura 15 se incluye una captura de pantalla de esta subpestaña.

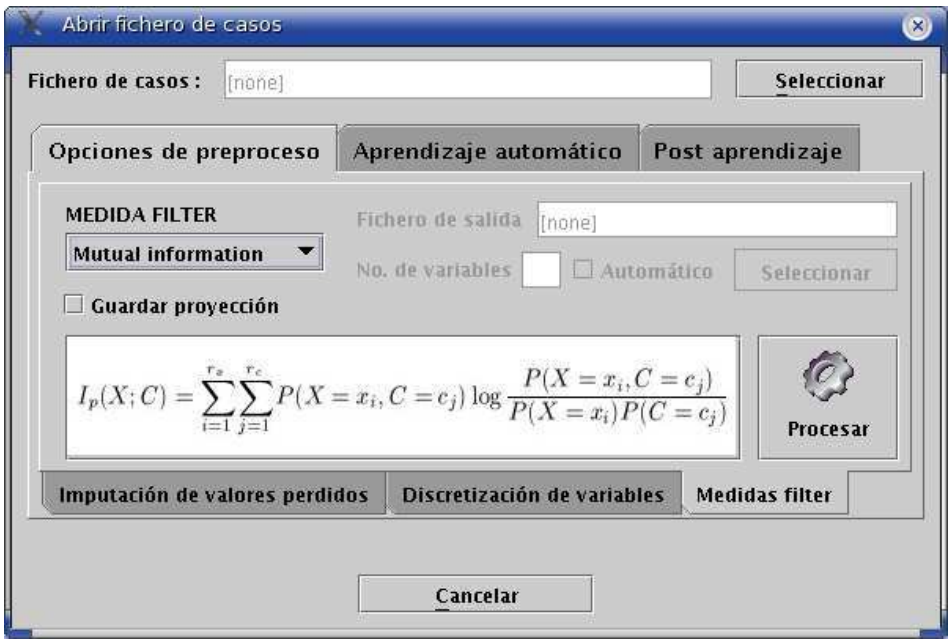


Figura 15: Cuadro de diálogo correspondiente a la selección de atributos dentro de la clase `DataBaseMonitor`.

Al formar parte del paquete de la interfaz gráfica del sistema, la mayoría de los elementos que incluye son componentes gráficos y métodos de `control de la lógica`, o métodos de control de errores y visualización. En particular, podríamos destacar tres de estos elementos, referentes a la parte de selección de variables:

- `tabFilter` – Objeto de tipo `JPanel` en el que se encuentran todas las opciones de selección de variables que han sido integradas en la interfaz gráfica. Corresponde a la captura presentada en la Figura 15.

- `tabFilterProcess_actionPerformed` – Método invocado al pulsar sobre el botón de *Procesar* del panel `tabFilter`. Tras una serie de verificaciones sobre los parámetros configurados por el usuario en el cuadro visual, será el encargado de invocar la tarea. Para ello, creará un "trabajador", o instancia de la clase `DataBaseMonitorWorker`, que será el que realice directamente el proceso. Mientras tanto, el panel general será desactivado mostrando una barra de progreso de la tarea; barra que indicará la tarea que está realizando el "trabajador" a cada segundo.

- `displayFilterResults` – Una vez finalizada la tarea por parte de la instancia de DataBaseMonitorWorker será éste el que invoque a este método. El método genera una segunda ventana en pantalla que recoge los resultados obtenidos para la tarea que había sido programada. Mostrará el método de selección aplicado, información sobre la base de datos, y el conjunto de atributos que hayan sido seleccionados, así como información dependiente de cada método.

La clase responde a un diseño *monolítico* ya que dispone, empotradas dentro de sí, de un conjunto de subclases necesarias para su correcto funcionamiento. Estas clases son utilizadas dependiendo de las tareas que se estén realizando y no todas están implicadas en los procesos de selección de variables. Una breve descripción de cada una de ellas sería:

- `JNumberTextField` – Clase utilizada como componente visual para los cuadros de texto numéricos. Incluye un control de errores si en el cuadro se teclean letras en vez de números.
- `InformationPane` – Clase correspondiente al panel informativo mostrado junto con la red cuando se aprende un clasificador Bayesiano.
- `AuxiliarDiscretizeNode` – Clase auxiliar utilizada para dar soporte a todas las opciones de discretización que existen en el sistema.
- `ScrollablePicture` y `ScrollableTextPane` – Ambas clases recogen componentes visuales utilizados para poder mostrar tanto imágenes como texto dentro de paneles, navegables mediante barras laterales.

Para una información más completa e información sobre una ejecución completa, se remite al lector al Apéndice B.

12.3. La clase *DataBaseMonitorWorker.java*

La tercera de las grandes clases incluídas en el sistema *Elvira* como parte del trabajo del presente proyecto, se encuentra dentro del paquete de la interfaz de usuario `elvira.gui`. Aunque en sí no posea ningun comportamiento visual, es la encargada de realizar las tareas que le sean invocadas desde la clase `DataBaseMonitor`. Además, deberá controlar la visualización de los resultados obtenidos en cada proceso, invocando a diferentes métodos de la clase `DataBaseMonitor`.

DataBaseMonitorWorker toma la función de un "trabajador" al que se le encarga una tarea a realizar, mientras tanto, la clase que la ha invocado esperará los resultados hasta que estén listos. Es por ello que el código específico para realizar cada una de las tareas que provee la interfaz gráfica de Elvira se encuentre incluído en esta clase.

La estructura de la clase es sencilla. Existen dos clases fundamentales embebidas dentro de su código, clases que serán las encargadas de realizar "el trabajo sucio":

- `SwingWorker` – Clase abstracta que crea un hilo de ejecución independiente con una tarea fijada por el usuario. La tarea será ejecutada al instanciar el hilo un objeto de tipo `ActualTask`.

- `ActualTask` – Clase sin métodos en cuya constructora se incluye el código necesario para las ejecuciones de las diferentes tareas abarcadas por la clase `DataBaseMonitor`. Cada tarea tiene asignado un identificador único y tomará distintos argumentos que debieron haber sido incluídos como parámetros en la llamada a la clase `DataBaseMonitorWorker`.

Tan solo se permite la existencia de una tarea en ejecución simultáneamente, es decir, el proceso ha sido considerado como "modal". Hasta que la tarea no haya finalizado y devuelto los resultados, el control sobre la aplicación del sistema no es devuelto al usuario.

Parte V

Experimentación y conclusiones

13. Diseño de los experimentos

La selección de las bases de datos de experimentación del presente proyecto se hizo de acuerdo a dos criterios fundamentales. Debido a que el principal objetivo era evaluar la bonanza de los métodos de selección de atributos/variables se debían seleccionar bases de datos con un amplio rango de atributos. Es decir, desde bases de datos con poco atributos (decenas) hasta bases de datos con muchos atributos (miles).

En segundo lugar, el tipo de técnicas utilizadas están demostrando una creciente utilidad en problemas bioinformáticos. Por ello, dos de las bases de datos utilizadas se enmarcan dentro de este campo, siendo los resultados obtenidos muy satisfactorios.

Las técnicas expuestas a lo largo de esta documentación están diseñadas para trabajar con bases de datos en las cuales los atributos o variables sean de carácter discreto, esto es, puedan tomar un número finito de estados. En algunas bases de datos existen atributos que inicialmente no cumplen esta premisa al tratarse de atributos continuos, bien por su propia naturaleza continua, o bien por ser el resutaldo de algún algoritmo. Para realizar la adaptación de un atributo continuo a otro discreto existen diferentes técnicas, de entre los cuales, el método de discretización en k intervalos de igual frecuencia ha sido el elegido.

El método de discretización por igual frecuencia (Catlett, 1991; Kerber, 1992; Dougherty et al., 1995) divide los valores ordenados de la variable en k intervalos de forma que cada intervalo contenga aproximadamente el mismo número de casos. Así, cada intervalo contendrá (idealmente) N/k valores adyacentes; donde k es un parámetro que el usuario debe prefijar. Este método se define como un método de discretización "ciego" ya que no

hace ningún estudio previo de los datos a discretizar, siendo el único parámetro fijado por el usuario. Siendo así, es uno de los métodos más ampliamente utilizados y contrastados para este tipo de tarea dentro del aprendizaje automático.

Los atributos continuos de las bases de datos *Image*, *Spambase* y *Cirrosis* fueron discretizados en cinco intervalos de igual frecuencia; una configuración muy habitual para bases de datos pequeñas y medianas. En el caso de los dos microarrays de ADN, *Lymphoma* y *Lupus*, las secuencias génicas fueron discretizadas en tres estados (intervalos), de forma que su tratamiento y posterior interpretación de resultados resultara más sencilla.

Una vez seleccionadas las bases de datos y preparadas para su tratamiento, se realizaron las siguientes experimentaciones cuyos resultados se muestran en la Sección 15:

1. Rankings de importancia de los atributos de cada base de datos en función de las siete medidas de filtrado presentadas en la Sección 6.2.1

2. Fijar manualmente puntos de corte en esos rankings y obtener la proyección de las bases de datos originales en función de los atributos incluídos antes del punto de corte.

3. Utilizando las bases de datos proyectadas, aprender modelos *nB* (naïve Bayes) y *TAN* (tree augmented naïve Bayes) validándolos con una validación cruzada en cinco hojas. Obtener los porcentajes de buena clasificación media, así como la desviación estándar producida en el proceso.

4. Utilizando el método de selección de un punto de corte óptimo, presentado en la Sección 6.2.2, se realiza el mismo proceso; se calcula la base de datos proyectada en función de esos atributos, se aprenden y validan los clasificadores con una validación cruzada en cinco hojas.

5. Por último, realizar una selección de atributos utilizando la técnica *CFS* (Sección

6.2.3) sobre las bases de datos. Proyectar las bases de datos con los atributos seleccionados, aprendizaje y validación de modelos nB y TAN.

14. Bases de datos

14.1. *Headache*

La primera de las bases de datos utilizadas en la experimentación del presente proyecto fue obtenida muestreando una red Bayesiana utilizada como ejemplo en muchos de los programas de redes Bayesianas existentes. La red se encuentra disponible "on-line" en el repositorio del proyecto Elvira en la dirección *http://leo.ugr.es/∼elvira*.

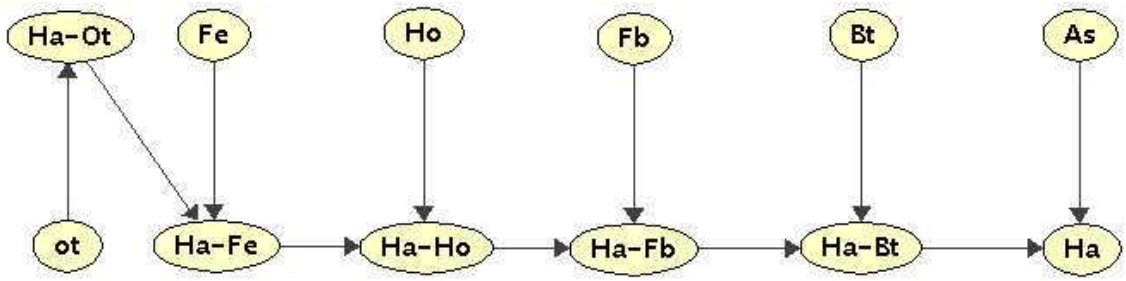


Figura 16: Red Bayesiana original de la base de datos *Headache*.

La estructura de dicha red se incluye en la Figura 16. Su objetivo es la predicción de una posible migraña, y de la intensidad de esta. Los posibles estados de las doce variables de la red están detallados en el Cuadro 7. Para abordar la red desde un punto de vista supervisado se tomó como variable clase el último de los nodos (`Ha`) y se muestrearon 10000 casos aleatorios generados en base a la red bayesiana.

14.2. *Image*

La base de datos identificada como *Image* procede del repositorio de bases de datos para *machine learning* de la Universidad de California-Irvine (Murphy and Aha, 1992). Su objetivo principal es aplicar métodos de aprendizaje automático tratando de identificar el material del que proceda una muestra segmentada de 3x3 pixels.

Variable	*Estados*
Fb	{present, absent}
Ho	{present, absent}
ot	{present}
Ha-Ot	{no, mild, moderate, severe}
Fe	{present, absent}
Ha-Fe	{no, mild, moderate, severe}
Ha-Ho	{no, mild, moderate, severe}
Ha-Fb	{no, mild, moderate, severe}
Bt	{present, absent}
Ha-Bt	{no, mild, moderate, severe}
As	{present, absent}
Ha	{no, mild, moderate, severe}

Cuadro 7: Variables de la base de datos *Headache*.

Los autores, el *Vision Group* de la Universidad de Massachusetts-Amherst[26], tomaron en 1990 muestras de pixels aleatorias provenientes de una base de datos con siete tipos distintos de imágenes exteriores. A partir de las muestras, cada instancia de esta base de datos se genera en función de la región de 3x3 pixels que rodea al pixel a clasificar. Una serie de parámetros son calculados y tomados como atributos de la base de datos; la especificación completa de los 19 atributos se incluye en el Cuadro 8.

La variable supervisada de esta base de datos puede tomar siete estados diferentes, siete materiales de los que proceden las muestras, los cuales son: *ladrillo*, *cielo*, *follaje*, *cemento*, *ventana*, *camino* o *hierba*. No existían valores perdidos y la distribución por clases es totalmente homogénea con 330 instancias para cada una de las clases.

14.3. *Insurance*

El término “insurance” significa literalmente “seguro”; concretamente, hace referencia al concepto de una póliza sobre algún objeto, animal o persona. En este caso, la base de datos denominada como *Insurance* es relativa a la problemática de los seguros de automóviles.

[26] *http://www.umass.edu*

Variable	*Explicación del parámetro*
region-centroid-col	la columna del pixel central de la región
region-centroid-row	la fila del pixel central de la región
region-pixel-count	el número de pixels por región = 9
short-line-density-5	resultado de un algoritmo de extracción de líneas que cuenta cuántas líneas de longitud 5 (cualquier orientación) de contraste bajo, menor o igual a 5, cruzan la región
short-line-density-2	idem que el anterior pero contando líneas de alto contraste, superior a 5
vedge-mean	media calculada a partir de la medida del contraste de los pixels horizontales adyacentes. Se utiliza como detector de bordes verticales
vegde-sd	desviación estándar de la medida anterior
hedge-mean	media calculada a partir de la medida del contraste de los pixels verticalmente adyacentes. Se utiliza como detector de líneas horizontales
hedge-sd	desviación estándar de la medida anterior
intensity-mean	la intensidad media de la región, calculada como (R + G + B)/3
rawred-mean	la media del componente R en la región
rawblue-mean	la media del componente B en la región
rawgreen-mean	la media del componente G en la región
exred-mean	media del exceso de rojo, calculada como: (2R - (G + B))
exblue-mean	media del exceso de azul, calculada como: (2B - (G + R))
exgreen-mean	media del exceso de verde, calculada como: (2G - (R + B))
value-mean	media del algoritmo de transformación 3-d no linear de un espacio de color RGB
saturatoin-mean	media de la saturación del algoritmo anterior
hue-mean	media de color del algoritmo anterior
Class	{brickface, sky, foliage, cement, window, path, grass}

Cuadro 8: Variables de la base de datos *Image*.

El trabajo de Binder (Binder et al., 1997) recoge en una de sus secciones una red bayesiana dedicada a estimar los costes previstos ante una demanda de un asegurado. En ella se recogen diferentes variables procedentes de conocimiento experto y que van a ser relevantes para el problema. La red bayesiana original puede consultarse en la Figura 17; los posibles estados para cada una de las variables se detallan en el Cuadro 9.

Inicialmente, esta red estaba pensada para un tipo de problema centrado en la propagación de evidencias y posterior estudio de los costes previstos por el proceso. El hecho de que este tipo de problemas sobre seguros sea muy corriente dentro del aprendizaje automático, hizo que se tomara como punto de partida para derivar de ella una base de datos con la que trabajar.

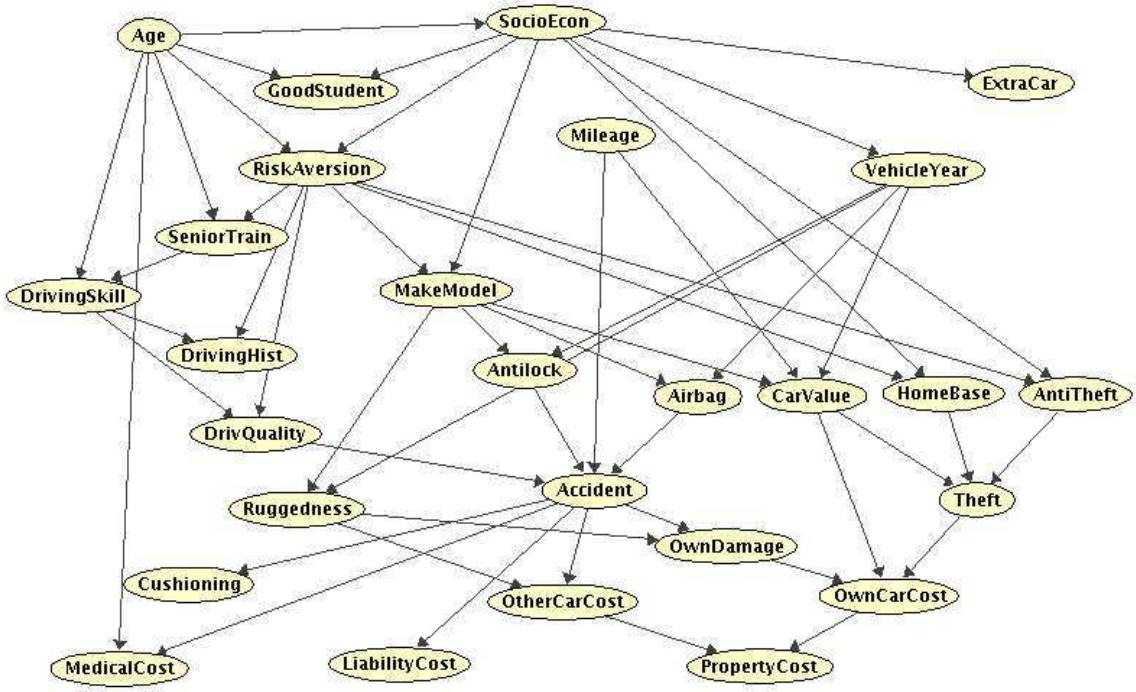


Figura 17: Red Bayesiana original de la base de datos *Insurance*.

Dejando de un lado su objetivo inicial, se fijó como variable clase la variable `DrivHist`. Con este enfoque se pretende que un modelo clasificatorio sea capaz de predecir, en base a los datos de un conductor determinado, qué futuros accidentes se esperan o no de él.

En base a la estructura presentada en la Figura 17 se muestrearon 10000 instancias aleatorias que constituirán la base de datos de la que partirán los análisis posteriores.

14.4. *Spambase*

Uno de los problemas más acuciantes en Internet actualmente es la proliferación incontralada de correos electrónicos publicitarios no deseados, o "spam". Cada usuario de Internet puede recibir al cabo de un día una cantidad considerable de correos electrónicos no deseados; en el ámbito de la empresa eso se traduce en una pérdida de tiempo de trabajo que tiene que ser dedicado a su detección y eliminación manual. Debido a ello las grandes empresas han volcado sus fuerzas en la automatización de este proceso, de forma

Variable	*Estados posibles*
Age	{Adolescent, Adult, Senior}
CarValue	{FiveThou, TenThou, TwentyThou, FiftyThou, Million}
SocioEcon	{Prole, Middle, UpperMiddle, Wealthy}
AntiTheft	{True, False}
GoodStudent	{True, False}
HomeBase	{Secure, City, Suburb, Rural}
RiskAversion	{Psychopath, Adventurous, Normal, Cautious}
Theft	{True, False}
VehicleYear	{Current, Older}
ThisCarCost	{Thousand, TenThou, HundredThou, Million}
MakeModel	{SportsCar, Economy, FamilySedan, Luxury, SuperLuxury}
OtherCarCost	{Thousand, TenThou, HundredThou, Million}
Antilock	{True, False}
PropCost	{Thousand, TenThou, HundredThou, Million}
Mileage	{FiveThou, TwentyThou, FiftyThou, Domino}
OtherCar	{True, False}
SeniorTrain	{True, False}
Airbag	{True, False}
DrivingSkill	{SubStandard, Normal, Expert}
Cushioning	{Poor, Fair, Good, Excellent}
DrivQuality	{Poor, Normal, Excellent}
MedCost	{Thousand, TenThou, HundredThou, Million}
Accident	{None, Mild, Moderate, Severe}
ILiCost	{Thousand, TenThou, HundredThou, Million}
RuggedAuto	{EggShell, Football, Tank}
DrivHist	{Zero, One, Many}
ThisCarDam	{None, Mild, Moderate, Severe}

Cuadro 9: Variables de la base de datos *Insurance*.

que no repercuta en el usuario final.

En los laboratorios de *Hewlett-Packard* en Palo Alto, California, se gestó entre junio y julio de 1999 la presente base de datos, conocida como *Spambase*. *Spambase* procede, al igual que *Image* del repositorio UCI (Murphy and Aha, 1992), al que fue donada por George Forman.

La base de datos está formada por el análisis de 4601 correos electrónicos, de los cuales 1813 eran “spam”. En base a los textos de estos correos se generaron 57 atributos continuos, 48 de ellos hacen referencia al porcentaje de ocurrencias de una determinada palabra o caracter. El resto evalúan ciertas características sintácticas habituales dentro de este problema. En el Cuadro 10 se detallan las variables y su significado. La variable clase

Variables	*Explicación*
0 .. 48	Porcentaje de ocurrencias de las palabras clave: *make*, *address*, *all*, *3d*, *our*, *over*, *remove*, *internet*, *order*, *mail*, *receive*, *will*, *people*, *report*, *addresses*, *free*, *business*, *email*, *you*, *credit*, *your*, *font*, *000*, *money*, *hp*, *hpl*, *george*, *650*, *lab*, *labs*, *telnet*, *857*, *data*, *415*, *85*, *technology*, *1999*, *parts*, *pm*, *direct*, *cs*, *meeting*, *original*, *project*, *re*, *edu*, *table*, *conference*
49 .. 54	Porcentaje de ocurrencias de los caracteres clave: *;*, *(*, *[*, *!*, *$*, *#*
55	Longitud media de las cadenas de caracteres en mayúsculas
56	Longitud de la mayor cadena de caracteres en mayusculas
57	Número total de caracteres en mayúsculas
58	{0, 1} El correo es considerado como “spam” (1), o no (0)

Cuadro 10: Variables de la base de datos *Spambase*.

será por tanto dicotómica, 0 en caso de no ser “spam”, y 1 en caso de serlo.

14.5. *Cirrosis*

Esta base de datos proviene del análisis realizado, entre Mayo de 1991 y Septiembre de 1998, en la Clínica Universitaria de Navarra a enfermos de cirrosis hepática. Este tipo de enfermos son altamente propensos a sufrir hipertensión portal, un incremento en la presión de la vena que comunica el sistema digestivo y bazo con el hígado. Dicha enfermedad conlleva en la mayoría de los casos la muerte debido a la aparición de varices esofágicas.

Existe una técnica denominada TIPS – *Transjugular Intrahepatic Portosystemic Shunt* – utilizada para descargar esa presión temporalmente, mientras el enfermo espera un transplante de hígado. La técnica TIPS consiste en la implantación de una vía artificial entre la vena portal y las venas suprahepáticas de forma que el flujo de sangre al hígado dañado se reduzca. Experimentalmente se ha comprobado que esta técnica puede conllevar una respuesta satisfactoria a largo plazo, o una rápida respuesta negativa – en un tiempo menor o igual a seis meses – acarreando la muerte del paciente.

El nombre de esta base de datos hace referencia al hecho de que todas las variables que se incluyen en ella (77) fueron evaluadas antes de implantar a los pacientes la vía

Variable	*Descripción*	*Tipo*	*Variable*	*Descripción*	*Tipo*
v0	Edad	C $(años)$	v4	Indicaciones de la DPPI	D (6)
v1	Altura	C (cm)	v5	Diabetes mellitus	D (3)
v2	Sexo	D (2)	v6	Peso	C (kg)
v3	Etiología de la cirrosis	D (5)			
v7	Índice CHILD	D (3)	v8	Índice PUGH	C
v9	Hemoglobina	C (g/dl)	v26	Sodio urinario	C (mEq/l)
v10	Hematocrito	C $(\%)$	v27	Potasio plasmático	C (mEq/l)
v11	Leucocitos	C $(\times 1000/mm^3)$	v28	Potasio urinario	C (mEq/l)
v12	Plaquetas	C $(/mm^3)$	v29	Osmolaridad en plasma	C $(mOsm/kg)$
v13	GOT	C (UI/l)	v30	Osmolaridad en orina	C $(mOsm/kg)$
v14	GPT	C (UI/l)	v31	Urea plasmática	C (g/l)
v15	Fosfatasa alcalina	C (UI/l)	v32	Creatinina en plasma	C (mg/dl)
v16	GGT	C (UI/l)	v33	Creatinina en orina	C (mg/dl)
v17	Bilirubina total en suero	C (mg/dl)	v34	Diuresis	C (ml)
v18	Bilirubina conjugada en suero	C (mg/dl)	v35	Aclaramiento de creatinina	C (ml/min)
v19	Actividad de protrombina	C $(\%)$	v36	Excrección fraccionada de sodio	C
v20	Albúmina en suero	C (g/dl)	v37	Aldosterona	C (pg/ml)
v21	Gammaglobulinas séricas	C (g/dl)	v38	Hormona antidiurética	C (pg/ml)
v22	Tiempo parcial de tromboplastina	C (seg)	v39	Dopamina	C (pg/ml)
v23	Proteínas totales	C (g/dl)	v40	Norepinefrina	C (pg/ml)
v24	FNG	C	v41	Epinefrina	C (pg/ml)
v25	Sodio plasmático	C (mEq/l)	v42	Renina plasmática	C $(ng/mlxh)$
v43	Tamaño de las varices esofágicas	C $(gradosI-IV)$	v45	Gastropatía portal	D (2)
v44	Presencia de varices gástricas	D (2)	v46	Hemorragia aguda	D (2)
v47	Número de hemorragias	C	v59	Peritonitis bacteriana espontánea	D (2)
v48	Origen de las hemorragias	D (4)	v60	Insuficiencia renal funcional	D (2)
v49	Escleroterapia previa	D (2)	v61	Nefropatía orgánica	D (2)
v50	Terapia con propranolol	D (3)	v62	Presión venosa hepática libre	C $(mmHg)$
v51	Intensidad de ascitis	D (3)	v63	Presión venosa hepática enclavada	C $(mmHg)$
v52	Número de paracentesis	C	v64	Gradiente de presión venosa hepática	C $(mmHg)$
v53	Volumen de paracentesis	C (l)	v65	Presión venosa central	C $(mmHg)$
v54	Dosis de furosemida	C $(mg/día)$	v66	Presión portal	C $(mmHg)$
v55	Dosis de espironolactona	C $(mg/día)$	v67	Gradiente de presión portosistémica	C $(mmHg)$
v56	Restricción de proteínas	C	v68	Gasto cardíaco	C (l/min)
v57	Número de encefalopatías hepáticas	C	v69	Presión arterial	C $(mmHg)$
v58	Tipo de encefalopatías hepáticas	D (3)	v70	Frecuencia cardíaca	C $(latidos/min)$
v71	Tamaño de la porta	C	v72	Velocidad de flujo de la porta	C (cm/seg)
v73	Trombosis portal	D (2)			
v74	Flujo portal derecho	D (2)	v76	Tamaño del bazo	C (cm)
v75	Flujo portal izquierdo	D (2)			
v77	Estado vital	D (2)			

Cuadro 11: Variables de la base de datos *Cirrosis*.

TIPS. El número de casos (pacientes) es de 107; 33 fallecieron antes de seis meses y 74 sobrevivieron por un período más largo.

En el Cuadro 11 se detallan las variables utilizadas procedentes de cuatro campos principales, historia clínica, análisis de laboratorio, índice de Child-Pugh, ecografías doppler, endoscopias, paramétros hemodinámicos y angiografías. Los indicadores *D* y *C* indican si dicha variable es de tipo discreto o continuo. En caso de ser discreto, entre paréntesis se incluye el número de estados que puede tomar.

14.6. *Lymphoma*

La base de datos identificada por *Lymphoma* proviene del trabajo realizado por Ash A. Alizadeh y Michael B.Eisen (Ash A. Alizadeh et al., 2000) y un amplio equipo de colaboradores sobre uno de los tipos más comunes de cáncer linfático.

El linfoma de células-B grandes, conocido como DLBCL *–Diffuse Large B-cell Lymphoma–* es el más común de los subtipos de linfomas no pertenecientes al patrón Hodgkin. Thomas Hodgkin fue el primer científico en identificar la patología de linfomas humanos en 1832. Basándonos en su primera clasificación, se distinguen hoy en día linfomas que se enmarcan en ella, y otros nuevos que no fueron identificados entonces.

Uno de los puntos que dio pie al estudio era el hecho de que la respuesta de los pacientes diagnosticados como DLBLC a los tratamientos era muy heterogénea. El 40 % de los pacientes respondían bien a la quimioterapia, teniendo una esperanza de vida alta; frente al resto que sucumbía a la enfermedad. Se ponía en entredicho el correcto diagnóstico de la enfermedad, o, que ésta pudiera ser dividida en subtipos más específicos.

En el estudio planteado se realizaron 128 microarrays en base a 96 casos o muestras de linfocitos normales y malignos. Se obtuvieron 1,8 millones de medidas de expresiones génicas que fueron agrupadas mediante varios algoritmos de cluster jerárquicos.

Tipología	*Número de casos*
DLBCL	46
Germinal centre B	2
NI. lymph node/tonsil	2
Activated blood B	10
Resting/activated T	6
Transformed cell lines	6
Follicular lymphoma	9
Resting blood B	4
Chronic lymphocytic leukaemia	11

Cuadro 12: Clases de la base de datos *Lymphoma*.

Las secuencias génicas incluídas en los microarrays fueron seleccionadas ad-hoc.

Existen librerías de genes en donde se identifican secuencias génicas involucradas, o sospechosas de estarlo, en problemas de este tipo de cánceres. El número de variables (secuencias génicas) que se incluyen en la base de datos presentada aquí es de 4026 sondas.

Las clases contenidas en el trabajo se presentan en el Cuadro 12, y corresponden a nueve posibles orígenes de diagnosis de un DLBLC. Debido a la complejidad sobre las nueve diferentes clases identificadas para el problema se remite al lector al trabajo original para una más amplia información de las mismas.

Debido a la repercusión del trabajo se creó una página web en la que se ha seguido todo el trabajo posterior, ampliaciones y material relacionado con el problema. Para cualquier tipo de consulta sobre el problema, la página web es *http://llmpp.nih.gov/lymphoma/*.

14.7. *Lupus*

La última base de datos, al igual que la anterior, procede de un problema biológico tratado con las últimas técnicas en bioinformática, los microarrays de ADN. La información procede del trabajo conjunto que se está llevando a cabo entre el grupo de investigación *ISG Group* de la Facultad de Informática y el grupo de investigaciones biológicas de Lejona, encabezado por Ana Zubiaga.

El nombre de esta base de datos –*Lupus*– procede del nombre de la enfermedad de la que deriva el estudio conjunto. El lupus eritematoso sistémico, también conocido como *LES* o simplemente *lupus*, es una enfermedad autoinmune que se caracteriza por episodios periódicos de inflamación y daño en articulaciones, tendones, otros tejidos conectivos y algunos órganos, incluyendo el corazón, los pulmones, los vasos sanguíneos, el cerebro, los riñones y la piel. El lupus es una sintomatología poco habitual que afecta a todas las personas de manera diferente y cuyos efectos oscilan desde leves a severos; potencialmente puede ser mortal.

Existe otra enfermedad también de carácter inmunológico y difícil de diagnosticar, muy relacionada en síntomas con la anterior, el denominado síndrome antifosfolipídico. El síndrome antifosfolípido, o *SAF*, es una enfermedad auto-inmune en la que el cuerpo produce grandes cantidades de anticuerpos antifosfolípidos. Los fosfolípidos son un tipo de grasas especiales que contienen el fosfato que constituye las paredes externas de las células del cuerpo. Los anticuerpos antifosfolípidos atacan a los fosfolípidos. Esto ocasiona diversos problemas incluyendo un aumento en la coagulación de la sangre. El *SAF* pudo definirse hace algunos años y en ocasiones se lo denomina *síndrome de Hughes* o *síndrome de la sangre pegajosa*.

El problema al que se enfrenta un médico ante el diagnóstico de estas dos enfermedades autoinmunes es que el *SAF* puede ocurrir junto con el *LES*. Esto hace que el diagnóstico de cada una, o de ambas conjuntamente sea complicado y muchas veces equivocado, provocando tratamientos incorrectos y/o graves efectos secundarios.

Por tanto, los dos grandes objetivos de este proyecto conjunto, enmarcado dentro del proyecto *Biogune–Genmodis* del Gobierno Vasco, se centran en:

1. Identificación de genes relevantes dentro de las sintomatologías LES y SAF.

2. Ayuda al facultativo en su tarea de diagnóstico de ambas cuando existan dudas razonables.

Los datos del trabajo provienen de 40 microarrays de ADN obtenidos a partir de pacientes sanos o contrastes, y de pacientes de *LES* y *SAF*. Los microarrays fueron obtenidos cruzando material genético de los contrastes contra el mismo material de los enfermos. De tal forma que, los casos en los que un enfermo de LES era cruzado con un contraste se "etiquetaban" como *LES*, ya que las diferencias en las expresiones génicas identificaban la diferencialidad entre el contenido genético de un enfermo de *LES* contra una persona que no lo sufriera.

Tipo	*Número de casos*	*Ids de los pacientes*
LES	20 instancias	04SE62HA
		04SE64HA
		04SE65HA
		04SE66HA
SAF	10 instancias	04SE63HA
		04SE67HA
SANE	10 instancias	04SE57HA
		04SE58HA
		04SE59HA
		04SE60HA
		04SE61HA

Cuadro 13: Casos de la base de datos *Lupus*.

El modelo de microarray utilizado para realizar el estudio es el denominado *U133A* de la compañía estadounidense *Affymetrix*. Se contaba con once muestras (4 *LES*, 2 *SAF* y 5 controles) de las que se obtuvieron un total de 40 cruces; los casos o instancias dentro del problema supervisado. En el Cuadro 13 se detallan todos los casos.

La tecnología de los arrays *U133A* es capaz de analizar un total de 22067 secuencias génicas simultáneamente. De éstas, la información que compone la base de datos es la expresión génica diferencial de la secuencia analizada, o *logRatio* de expresión. El *logRatio* es un indicador de cuántas veces más se ha expresado una secuencia con respecto a la expresión de la secuencia base o de contraste. De las 22067 secuencias analizadas se encontraron diferencialmente expresadas 8808 de ellas, que serán los atributos o variables de la base de datos.

15. Resultados

A lo largo de esta sección se muestran los resultados obtenidos para las bases de datos presentadas en la sección anterior. Durante todo el texto se utilizan acrónimos para identificar las distintas técnicas que se ilustran, de tal forma:

Acrónimo	*Técnica*	*Localización en el texto*
IM	Información mutua	pág. 33
DE	Distancia euclídea	pág. 34
MA	Métrica Matusita	pág. 34
KL-1	Divergencia de Kullback-Leibler modo 1	pág. 34
KL-2	Divergencia de Kullback-Leibler modo 2	pág. 34
SH	Entropía de Shannon	pág. 35
BH	Métrica Bhattacharyya	pág. 36
CFS	Correlation-based Feature Selection	pág. 39
Todas	Utilización de todas las variables originales	—

Cuadro 14: Acrónimos utilizados en la presentación de los resultados.

En las diferentes subsecciones dedicadas a cada base de datos se presentan dos resultados fundamentales. Por un lado, las gráficas obtenidas, para cada uno de los rankings efectuados, de las precisiones y desviaciones típicas obtenidas por los clasificadores nB y TAN en función de los diferentes puntos de corte (incluyendo el encontrado con el método automático). Debido al tamaño de estas gráficas se ha optado por la separación de ellas en dos figuras diferentes, la segunda de ellas marcada en su pie de gráfica como *(cont)*.

Junto a la batería de gráficas y sus comentarios se incluye, por cada una de las bases de datos, una tabla en la que se reflejan numéricamente los resultados más destacables de cada experimentación. De tal forma que se incluyen los puntos de corte para cada una de las medidas de ranking, las precisiones para ese punto de los modelos nB y TAN, y el valor de un test de hipótesis para dos muestras no pareadas. En esta tabla se recogen también los resultados obtenidos con el método *CFS*, para los modelos nB y TAN, y su valor en el test de hipótesis. Este test de hipótesis compara las posibles diferencias estadísticas

que puedan existir entre las precisiones obtenidas por los métodos filter y las precisiones obtenidas con todas las variables, es decir, sin ningún pre-procesamiento previo. Por ello, cada tabla incluye una fila en la que se recogen las precisiones obtenidas por los modelos cuando tienen en cuenta todas las variables de cada problema. Las precisiones obtenidas para cada una de las cinco hojas de la validación cruzada son comparadas con las obtenidas por el mismo proceso al tener en cuenta todas las variables. Para realizar esta comparación se utilizó el test no paramétrico de Mann-Whitney.

La prueba U de *Mann-Whitney* es la técnica más extendida de comparación de medias entre dos muestras independientes, o, no-pareadas. A groso modo, lo que hace este test es ordenar las puntuaciones de todos los sujetos en la variable considerada (comenzando desde 1 para el rango más bajo) y comparar el rango medio de los dos grupos.

El estadístico U de Mann-Whitney se calculará como $U = W - m(m+1)/2$ donde W es la suma de los rangos (valores en la ordenación) correspondientes de la segunda muestra y m es el número de observaciones de la misma.

Cuando existen en total menos de 20 observaciones para comparar, el estadístico U de contraste debe ser comparado con las tablas de la distribución apriori de U. Obtenemos por tanto un p-valor del test que indicará la significatividad estadística de dicho test. La hipótesis nula contempla que no existan diferencias significativas entre las medias; mientras que, la hipótesis alternativa afirma que sí existe significatividad estadística entre ambas medias.

Debido a que podemos fijar diferentes umbrales de confianza para aceptar o rechazar este test –un $p < 0{,}05$ para un 95 % de confianza, o un $p < 0{,}01$ para un 99 %– en las tablas de resultados se incluyen los valores obtenidos por el test. De esta forma el lector podrá valorar con más exactitud los resultados obtenidos.

Debe notarse que las variables obtenidas por el método *CFS* no tienen porqué coincidir con las variables seleccionadas en un corte automático de una técnica de ranking. En

aquellas bases de datos en las que el número no sea muy alto se indicarán cuáles han sido seleccionadas por el *CFS*.

15.1. *Headache*

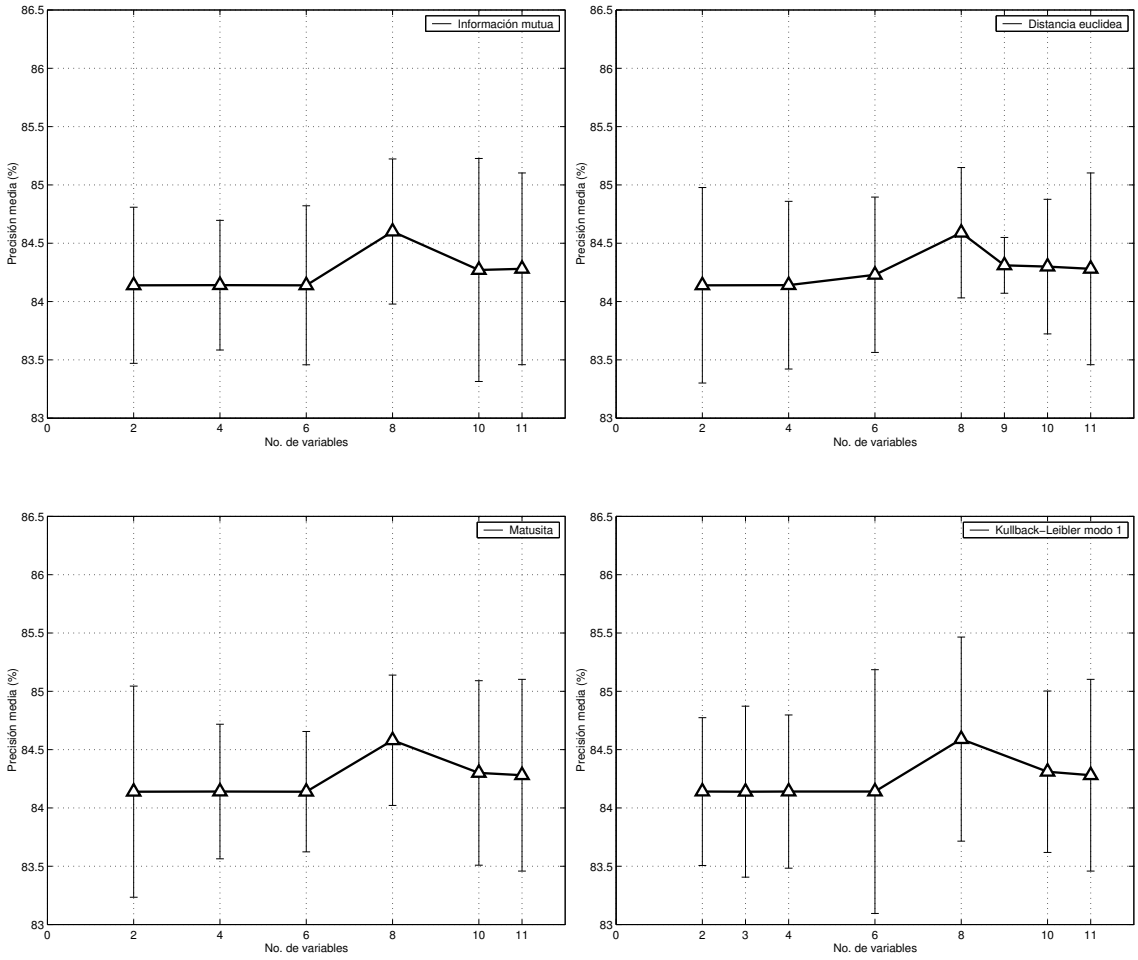


Figura 18: Resultados de las validaciones del modelo nB para la base de datos *Headache*.

Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 18 y 19. Las Figuras 20 y 21 muestran los resultados obtenidos sobre esos mismos datos habiendo utilizado un modelo clasificatorio TAN.

El Cuadro 15 comprende los resultados cuantitavos resumen de la información mos-

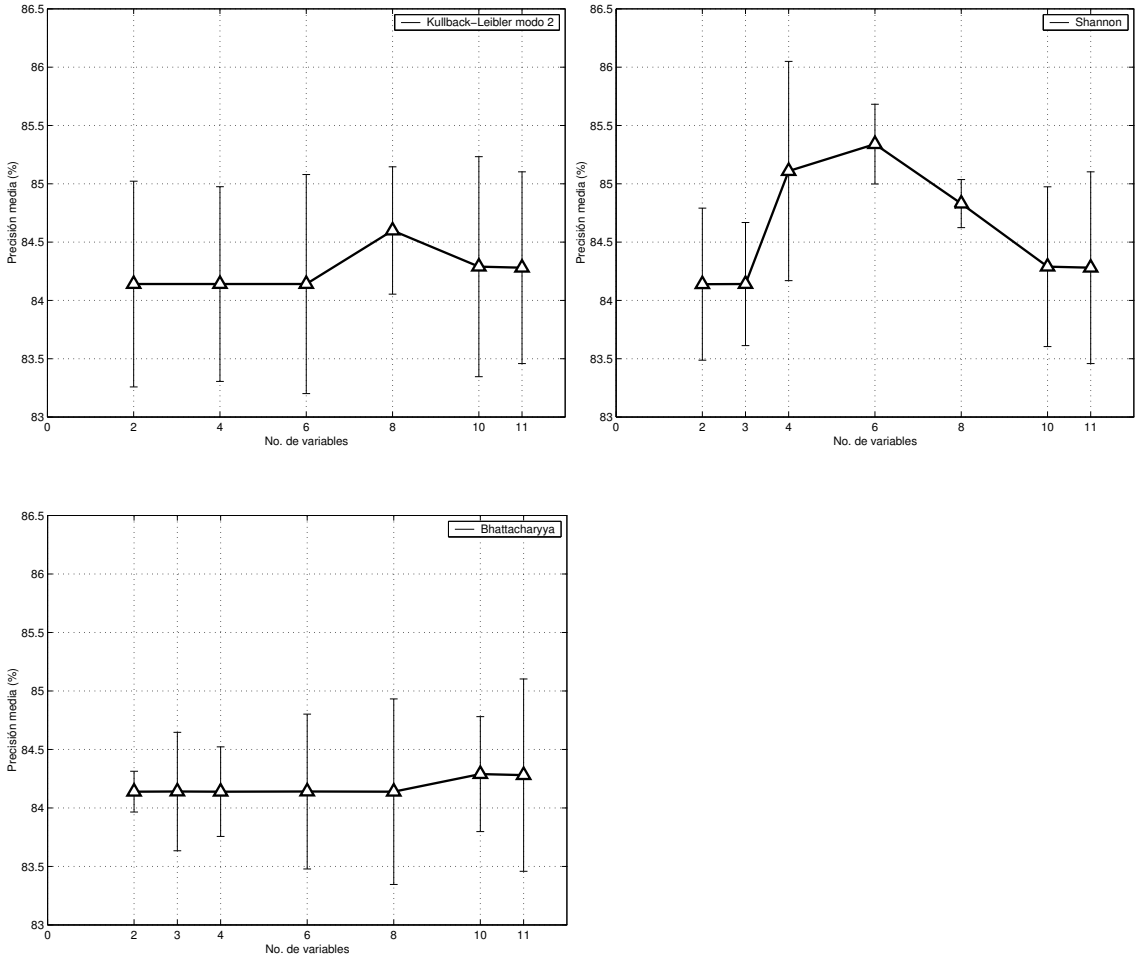


Figura 19: Resultados de las validaciones del modelo nB para la base de datos *Headache* (cont).

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*	*TAN + 5-cv*	*p-valor*
Punto de corte					
IM	4	$84{,}140 \pm 0{,}556$	0,84	$84{,}140 \pm 0{,}276$	0,01
DE	9	$84{,}310 \pm 0{,}239$	1	$86{,}440 \pm 0{,}524$	1
MA	4	$84{,}140 \pm 0{,}719$	0,84	$84{,}140 \pm 0{,}593$	0,01
KL-1	3	$84{,}139 \pm 0{,}733$	0,84	$84{,}139 \pm 0{,}682$	0,01
KL-2	10	$84{,}280 \pm 0{,}773$	1	$86{,}390 \pm 0{,}955$	1
SH	3	$84{,}140 \pm 0{,}528$	1	$83{,}070 \pm 1{,}015$	0,01
BH	3	$84{,}140 \pm 0{,}506$	0,84	$84{,}140 \pm 0{,}927$	0,02
CFS	2	$84{,}139 \pm 1{,}245$	0,84	$84{,}140 \pm 0{,}538$	0,02
Todas	11	$84{,}280 \pm 0{,}822$	—	$86{,}340 \pm 1{,}034$	—

Cuadro 15: Resultados de las validaciones en los valores críticos para la base de datos *Headache*.

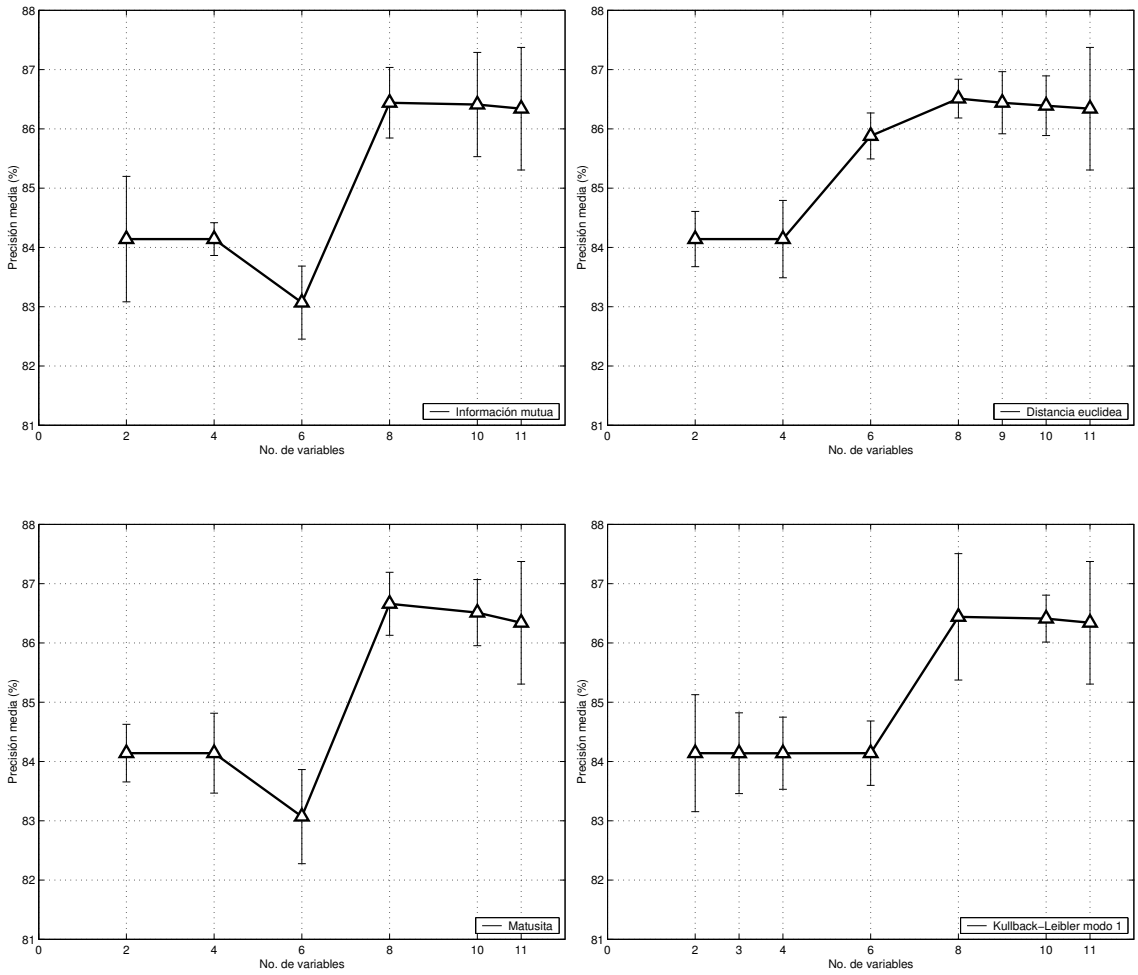


Figura 20: Resultados de las validaciones del modelo TAN para la base de datos *Headache*.

trada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

Las precisiones obtenidas por los modelos nB en base a las métricas *filter* son semejantes a las obtenidas en base a todas las variables. Ninguno de los valores supera el test de significación ni al 95 % ni al 99 %, lo cual indica que aquellas variables no incluídas en los subconjuntos seleccionados son descartables para mantener el nivel de precisión clasificatoria. Puede observarse que incluso con un número muy reducido de variables,

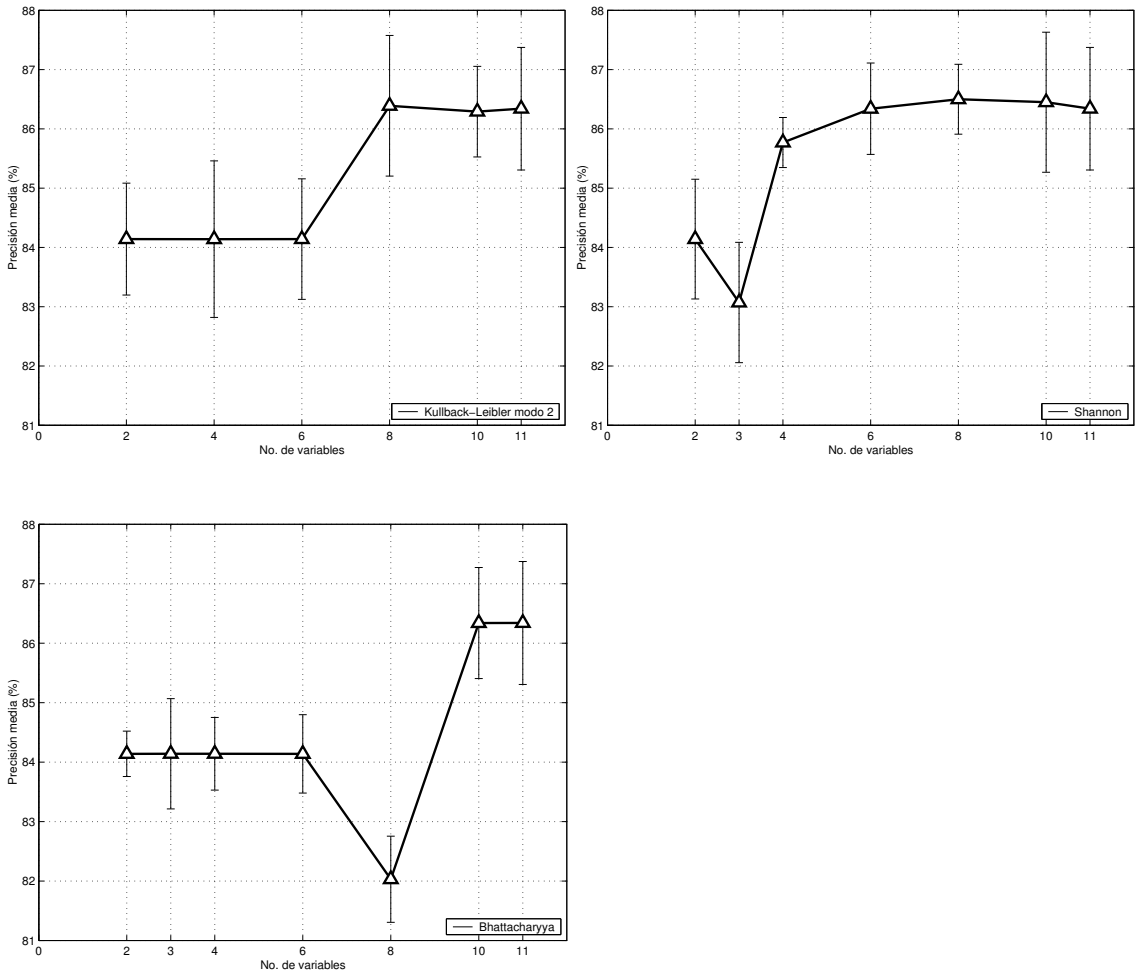


Figura 21: Resultados de las validaciones del modelo TAN para la base de datos *Headache* (cont).

tan sólo dos de las once originales, la precisión obtenida es casi igual a la obtenida con todas ellas.

Deteniéndonos en los resultados de los modelos TAN vemos cómo la tendencia cambia. Todas las técnicas salvo una (KL-2) obtienen peores resultados al utilizar los conjuntos reducidos. Es más, cinco de los ocho resultados son significativamente peores con un nivel de confianza del 99 % en el test, lo cual indica que realmente la exclusión de las variables ha hecho perder precisión al modelo. Una explicación a ello podría ser la

detección de alguna dependencia entre las variables predictoras que en el caso del nB no ha podido ser tratada.

El método *CFS* ha seleccionado dos atributos de los once, selección que implica una reducción del 81.82 % en el tamaño de la base de datos. Los dos atributos seleccionados son `Ha-Bt` y `Ha-Fb`.

15.2. *Image*

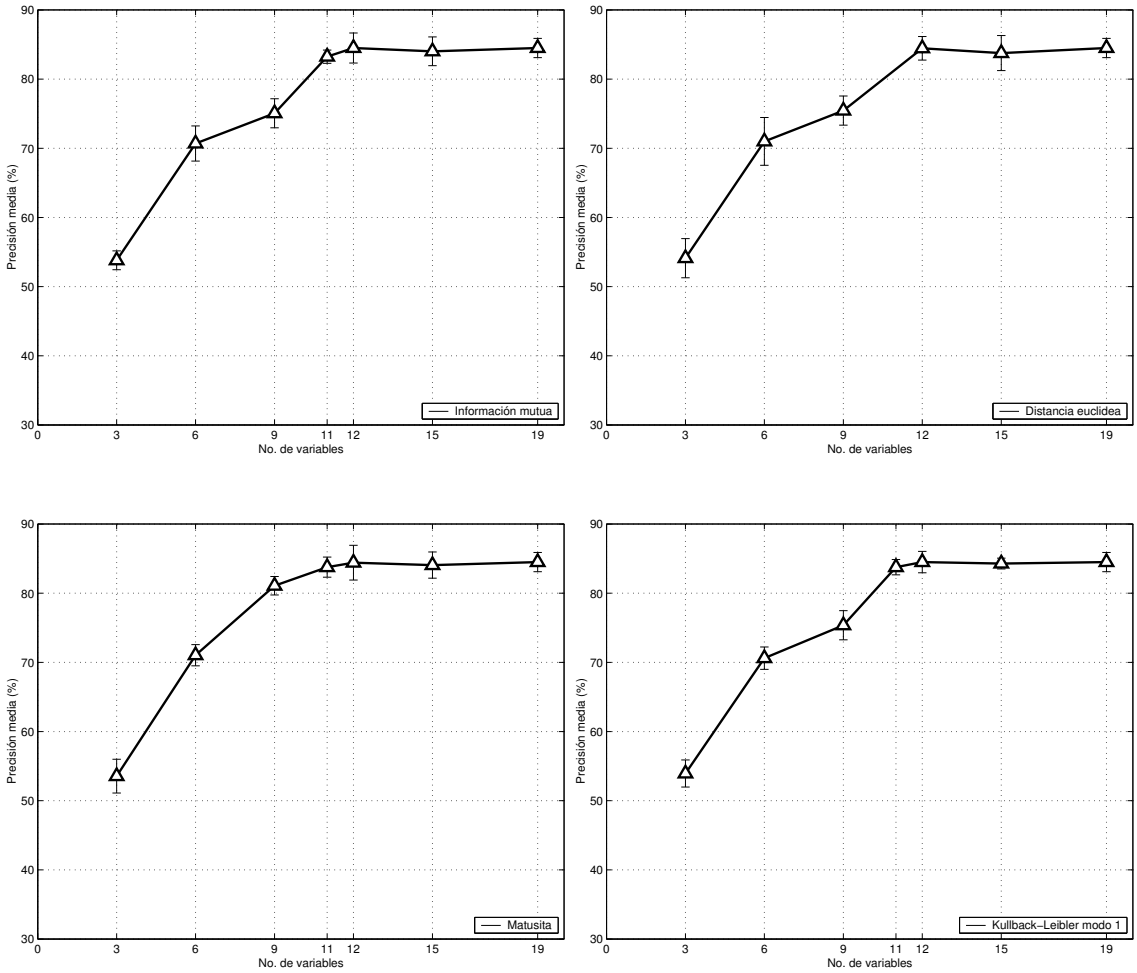


Figura 22: Resultados de las validaciones del modelo nB para la base de datos *Image*.

Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 22 y 23. Las Figuras 24 y 25 muestran los resultados obtenidos sobre esos mismos datos habiendo utilizado un modelo clasificatorio TAN.

El Cuadro 16 comprende los resultados cuantitavos resumen de la información mostrada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

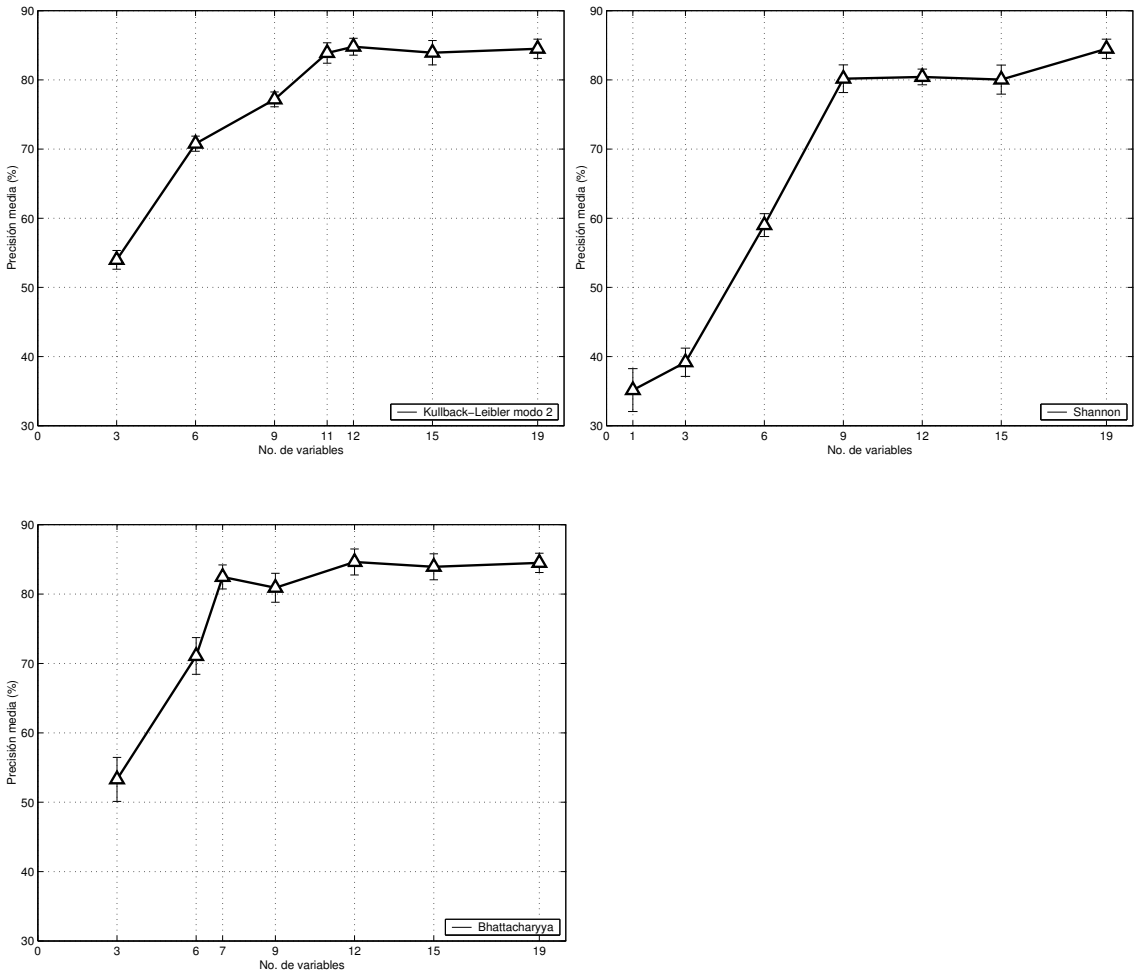


Figura 23: Resultados de las validaciones del modelo nB para la base de datos *Image* (cont).

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*	*TAN + 5-cv*	*p-valor*
Punto de corte					
IM	11	$83{,}250 \pm 0{,}954$	0,42	$90{,}610 \pm 1{,}723$	0,03
DE	12	$84{,}410 \pm 1{,}642$	1	$91{,}000 \pm 0{,}816$	0,06
MA	11	$83{,}770 \pm 1{,}455$	0,69	$90{,}650 \pm 0{,}779$	0,03
KL-1	11	$83{,}770 \pm 1{,}103$	0,84	$90{,}480 \pm 1{,}720$	0,06
KL-2	11	$83{,}900 \pm 1{,}478$	1	$90{,}700 \pm 1{,}348$	0,03
SH	1	$35{,}150 \pm 3{,}105$	0,01	$35{,}150 \pm 1{,}646$	0,01
BH	7	$82{,}470 \pm 1{,}731$	0,22	$92{,}030 \pm 1{,}090$	0,42
CFS	11	$83{,}809 \pm 1{,}942$	1	$90{,}562 \pm 0{,}621$	0,02
Todas	19	$84{,}110 \pm 1{,}244$	—	$93{,}07 \pm 1{,}239$	—

Cuadro 16: Resultados de las validaciones en los valores críticos para la base de datos *Image*.

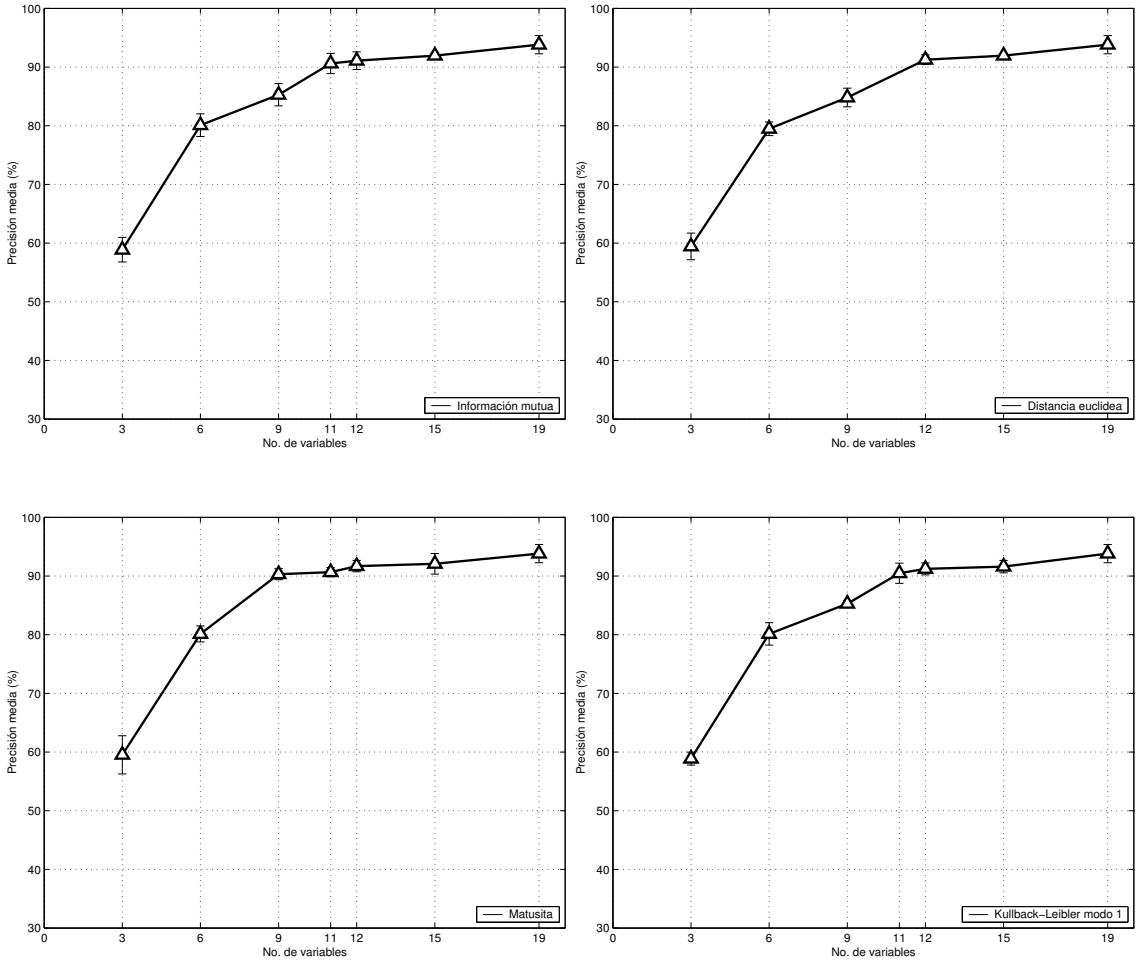


Figura 24: Resultados de las validaciones del modelo TAN para la base de datos *Image*.

Los resultados de este conjunto de datos responden a una base de datos en la que no hay aparentemente atributos descartables. Si vemos las tendencias de ambos modelos, nB y TAN, se observa cómo la tendencia en las gráficas es monótona creciente. A cada atributo de más que entra en la evaluación, más precisión obtienen los modelos. La precisión obtenida con un modelo nB y la obtenida con un TAN dista en nueve puntos porcentuales, distancia bastante grande, lo cual nos da la pista para afirmar que existen dependencias condicionales muy fuertes entre algunos de los atributos.

Analizando por separado los resultados vemos cómo las métricas *filter* en unión con

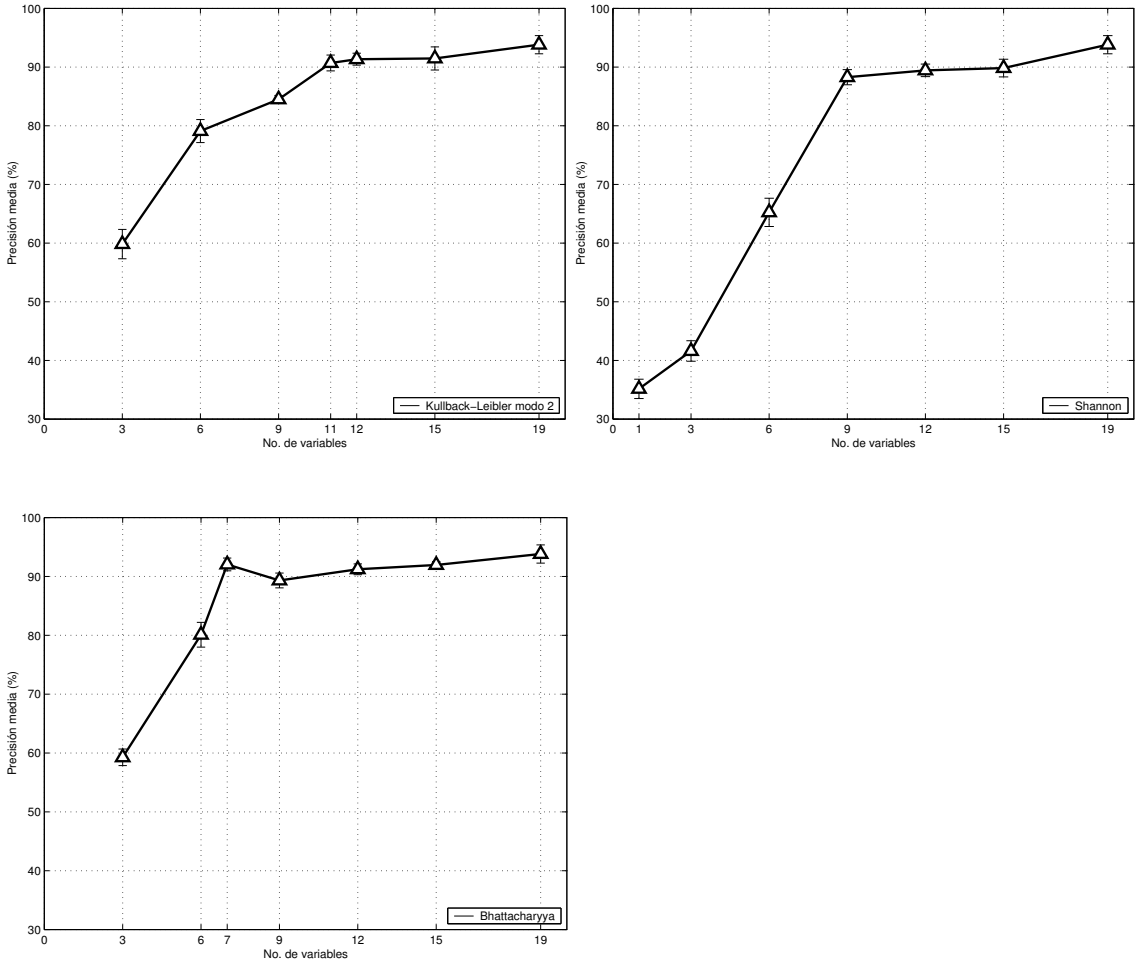


Figura 25: Resultados de las validaciones del modelo TAN para la base de datos *Image* (cont).

el corte del codo se comportan de forma óptima. Utilizando entre 11 o 12 de los 19 atributos se obtienen porcentajes que difieren en menos de un punto la precisión obtenida con los 19. El test de significancia rechaza la existencia de diferencias estadísticamente significativas, es decir, el uso de los atributos seleccionados no perjudicará los procesos de predicción, haciendo disminuir el tamaño de la base de datos considerablemente.

En el caso de los modelos TAN la tendencia de las gráficas es semejante, pero existe una gran diferencia: en este caso, el modelo TAN con todas las variables obtiene una precisión media de 93,07 %, precisión que se va a alejar de las obtenidas por el resto de

conjuntos. Los p-valores nos dan la pista para afirmar que las diferencias son significativas, los resultados son estadísticamente peores y debe recomendarse el uso de todas las variables para trabajar con modelos TAN sobre esta base de datos. Aunque existe una excepción a este comportamiento, la métrica *Bhattacharyya*, que en el caso de nB no obtenía un resultado destacable, mejora en este caso a las demás. Sigue quedando por debajo del mejor resultado, pero la diferencia ya no es significativa (93.07 % frente a 92.03 %).

En el caso de la métrica *Shannon* el punto de corte ha fallado, considerando tan sólo el primer atributo del ranking; de ahí los malos resultados obtenidos para los dos modelos clasificatorios.

CFS selecciona once atributos de los diecinueve originales, una reducción de un 42.10 % del tamaño original. Estos once atributos son: `hue-mean`, `rawred-mean`, `rawgreen-mean`, `region-centroid-row`, `intensity-mean`, `saturation-mean`, `exgreen-mean`, `value-mean`, `exblue-mean`, `exred-mean` y `rawblue-mean`.

Al igual que en la base de datos *Headache*, la precisión obtenida por estos once atributos como únicas variables predictoras es semejante a la obtenida con los diecinueve originales utilizando un modelo nB. En el caso de un TAN, como ya hemos comentado, la diferencia se torna significativa a favor de la utilización de la base de datos completa.

15.3. *Insurance*

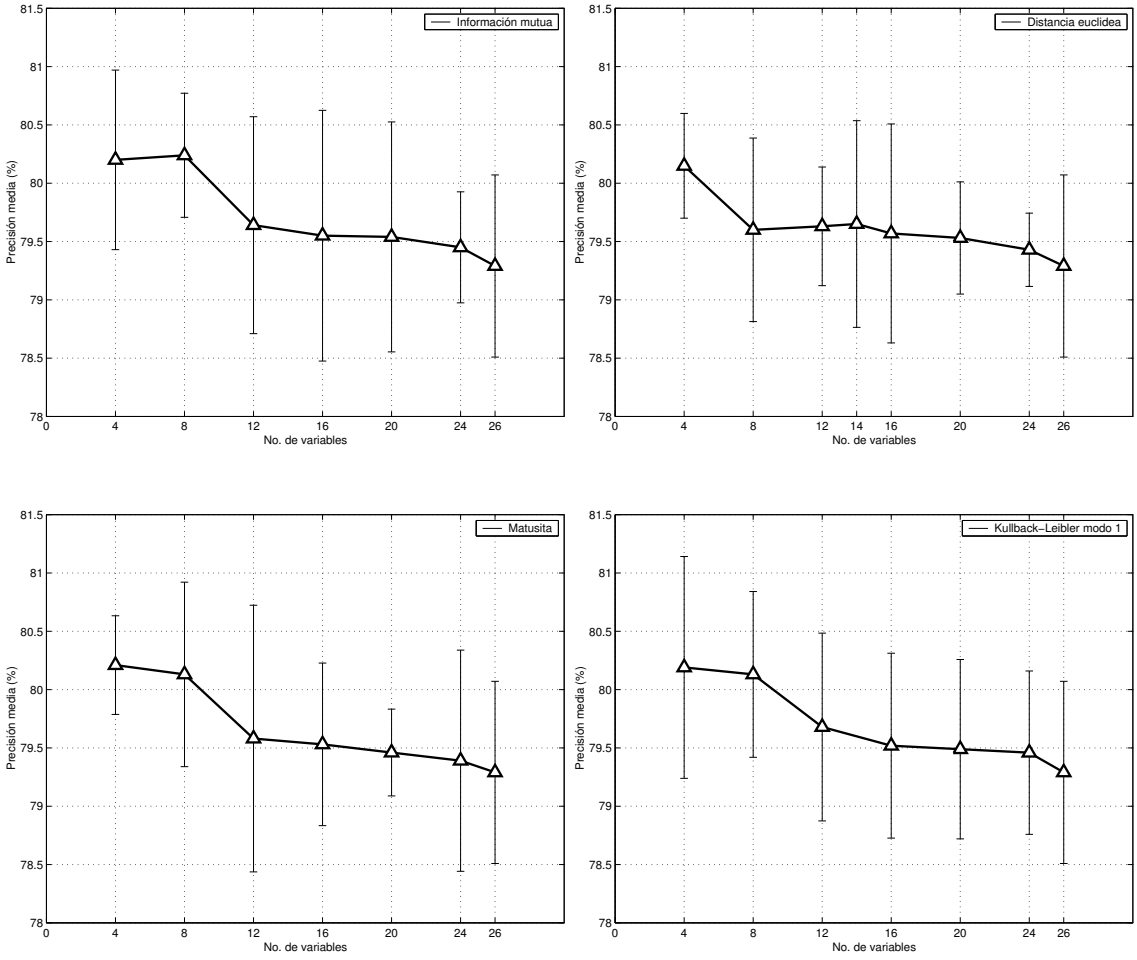


Figura 26: Resultados de las validaciones del modelo nB para la base de datos *Insurance*.

Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 26 y 27. Las Figuras 28 y 29 muestran los resultados obtenidos sobre esos mismos datos habiendo utilizado un modelo clasificatorio TAN.

El Cuadro 17 comprende los resultados cuantitavos resumen de la información mostrada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

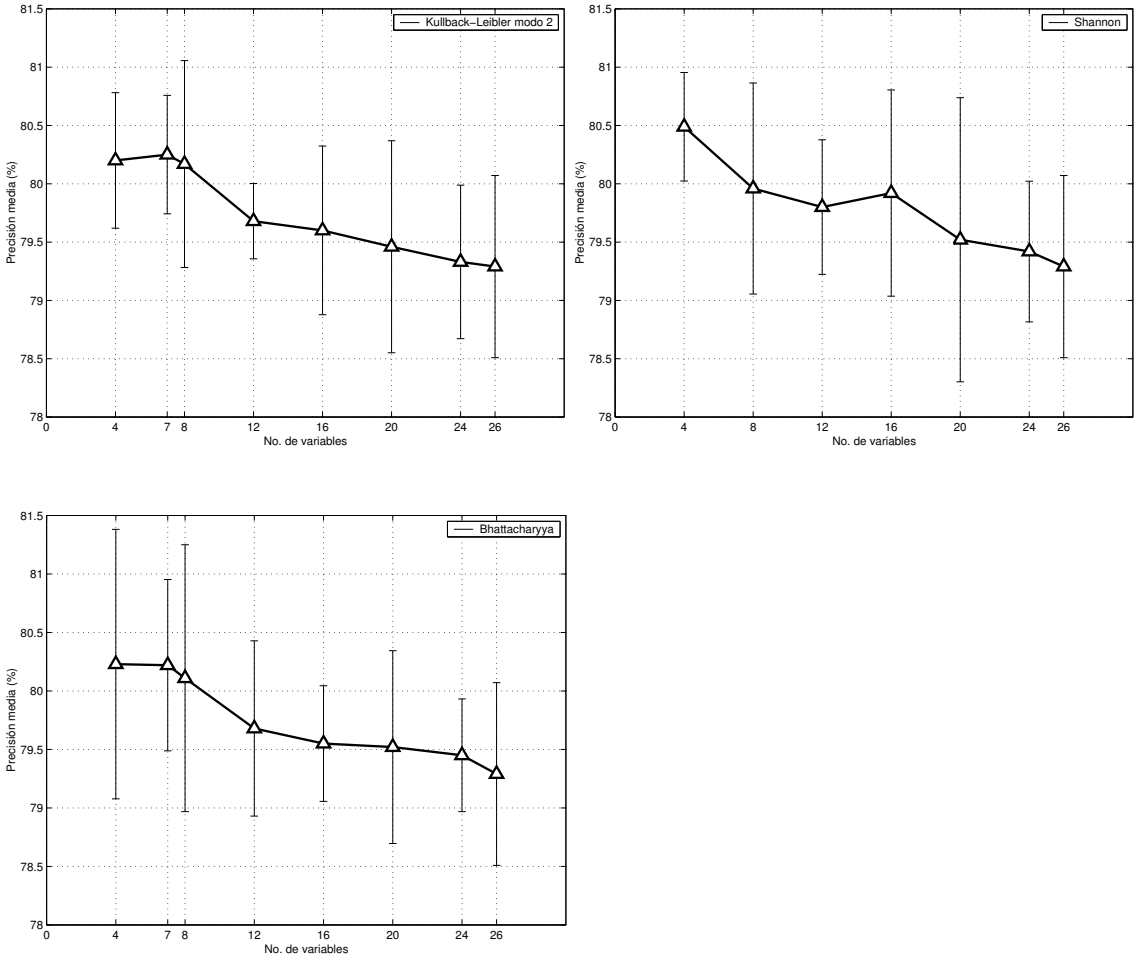


Figura 27: Resultados de las validaciones del modelo nB para la base de datos *Insurance* (cont).

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*	*TAN + 5-cv*	*p-valor*
Punto de corte					
IM	8	$80{,}240 \pm 0{,}532$	0,10	$80{,}140 \pm 0{,}884$	1
DE	14	$79{,}650 \pm 0{,}886$	0,69	$80{,}160 \pm 0{,}780$	1
MA	8	$80{,}130 \pm 0{,}791$	0,10	$79{,}990 \pm 1{,}080$	1
KL-1	8	$80{,}130 \pm 0{,}711$	0,15	$80{,}130 \pm 0{,}930$	1
KL-2	7	$80{,}250 \pm 0{,}508$	0,10	$80{,}300 \pm 0{,}532$	1
SH	16	$79{,}920 \pm 0{,}884$	0,22	$80{,}280 \pm 0{,}639$	1
BH	7	$80{,}220 \pm 0{,}732$	0,15	$80{,}210 \pm 0{,}466$	1
CFS	4	$80{,}749 \pm 0{,}894$	0,06	$81{,}329 \pm 0{,}660$	0,15
Todas	26	$79{,}290 \pm 0{,}781$	—	$80{,}150 \pm 0{,}995$	—

Cuadro 17: Resultados de las validaciones en los valores críticos para la base de datos *Insurance*.

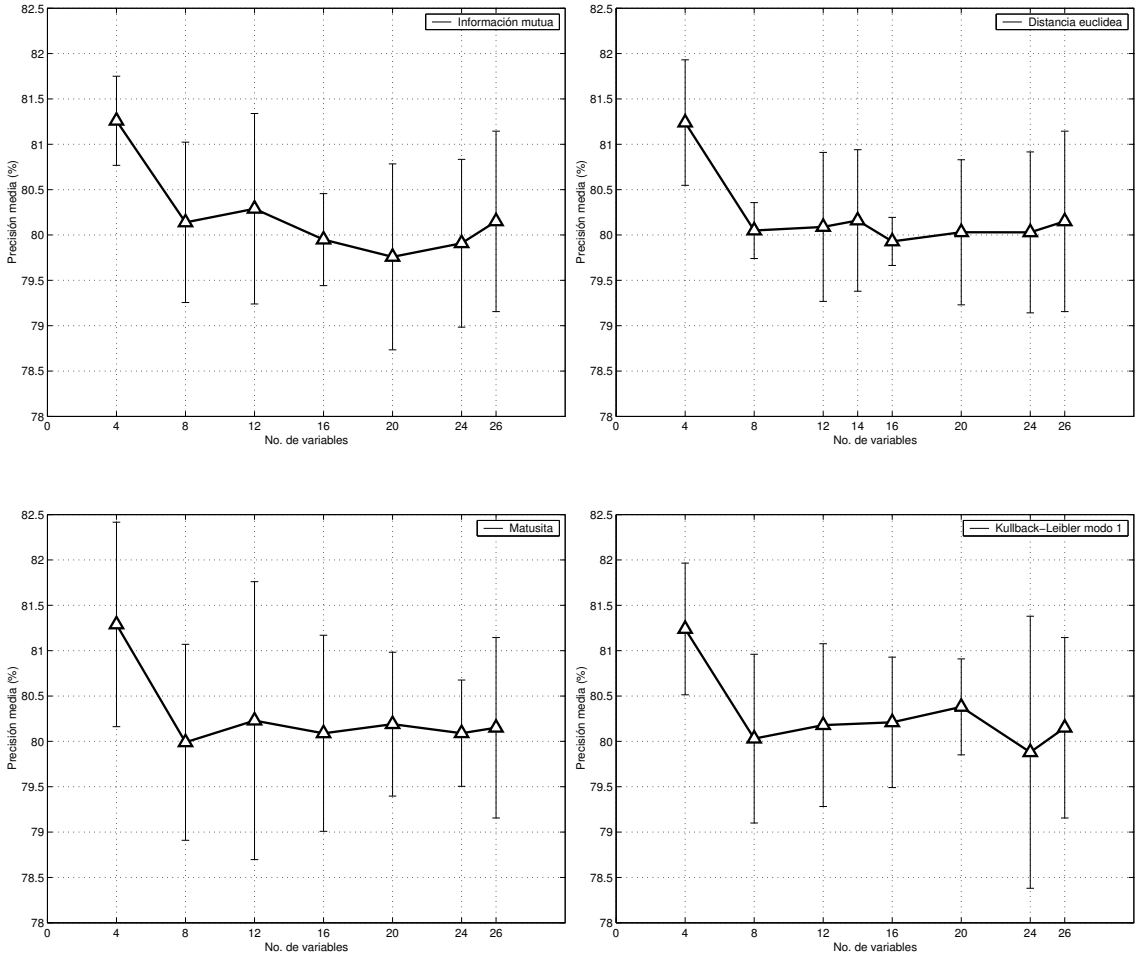


Figura 28: Resultados de las validaciones del modelo TAN para la base de datos *Insurance*.

En esta base los resultados obtenidos para ambos modelos clasificatorios son prácticamente iguales. La precisión media obtenida por un modelo nB que incluya todas las variables no llega a ser un punto porcentual menor que el correspondiente al TAN. Este hecho implica que las variables del problema parecen no encontrarse muy correladas entre sí.

Si fijamos la atención en las tendencias mostradas por las gráficas de ambos modelos vemos como para un subconjunto de variables muy pequeño, con cuatro son suficientes, los porcentajes de buena clasificación mejoran a todos los porcentajes de los demás sub-

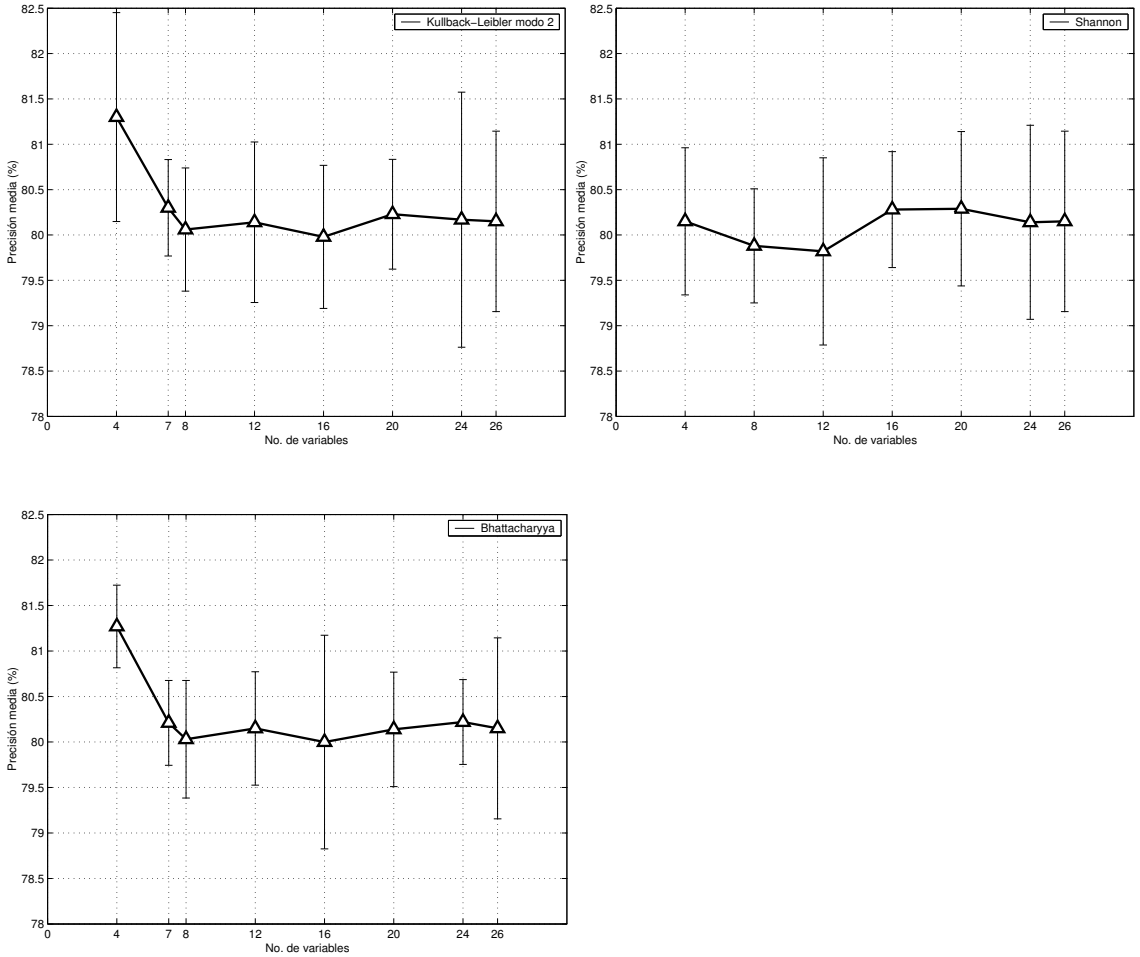


Figura 29: Resultados de las validaciones del modelo TAN para la base de datos *Insurance* (cont).

conjuntos. A partir del quinto de los rankings añadido a los subconjuntos evaluados la precisión no sólo mejora, sino que incluso empeora. Parece que existe una gran cantidad de información redundante, y/o irrelevante que podría ser descartada para el proceso clasificatorio.

El punto de corte se comporta correctamente, aunque no es capaz de detectar estos picos, quedándose en el entorno de siete u ocho atributos en los rankings. Con ese número de atributos, las precisiones medias obtenidas son significativamente iguales a las obtenidas con todos los atributos, y, en muchos casos, algo mejores.

En este contexto va a ser el *CFS* la medida que mejor comportamiento muestre. Es capaz de seleccionar este pequeño subconjunto de atributos realmente importantes y, con ellos, obtener precisiones, al aplicar los modelos clasificatorios, mejores que cualquiera de las obtenidas antes. *CFS* selecciona los siguientes cuatro atributos: `ILiCost`, `DrivingSkill`, `DrivQuality`, `Accident`, llegando la reducción a ser del 86.61 % con respecto al tamaño original de la base de datos.

15.4. *Spambase*

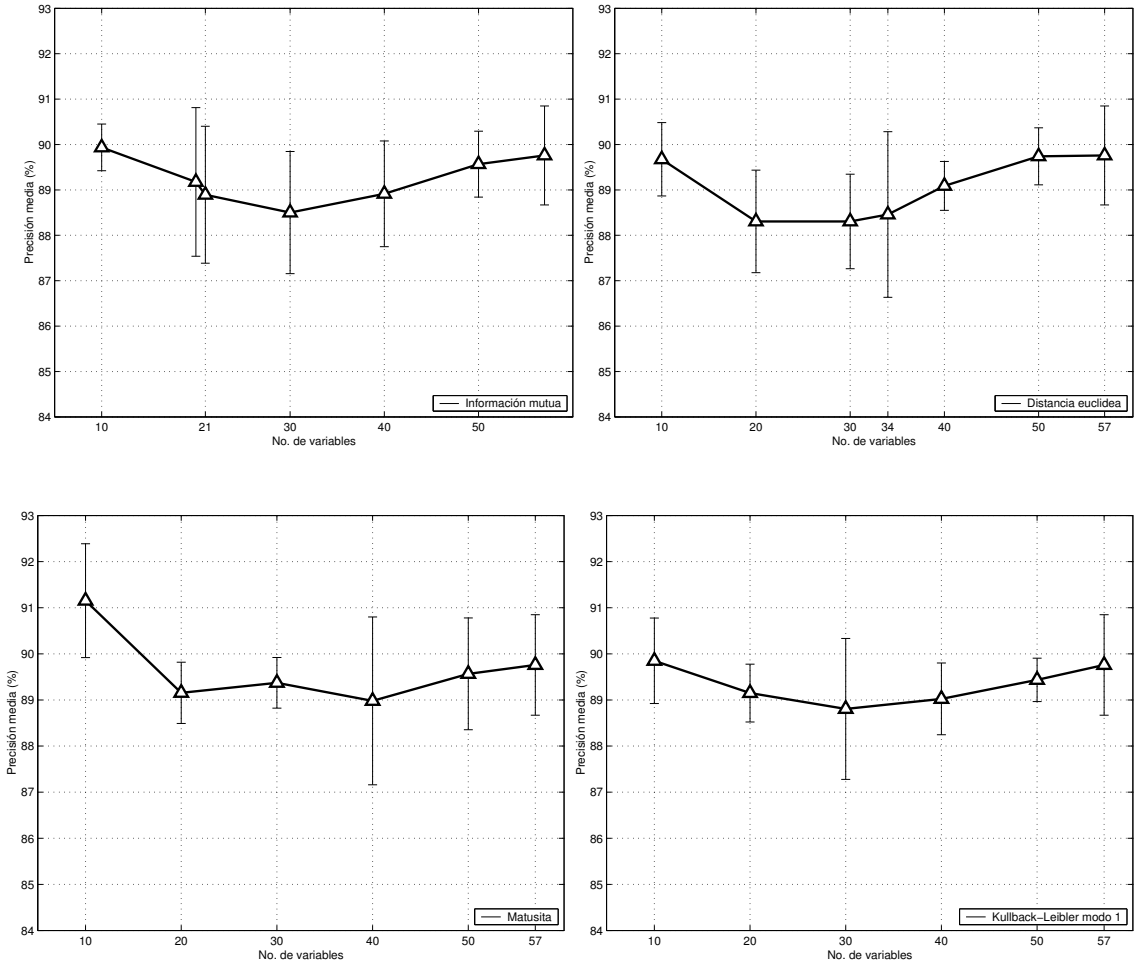


Figura 30: Resultados de las validaciones del modelo nB para la base de datos *Spambase*.

Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 30 y 31. Las Figuras 32 y 33 muestran los resultados obtenidos sobre esos mismos datos habiendo utilizado un modelo clasificatorio TAN.

El Cuadro 18 comprende los resultados cuantitavos resumen de la información mostrada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

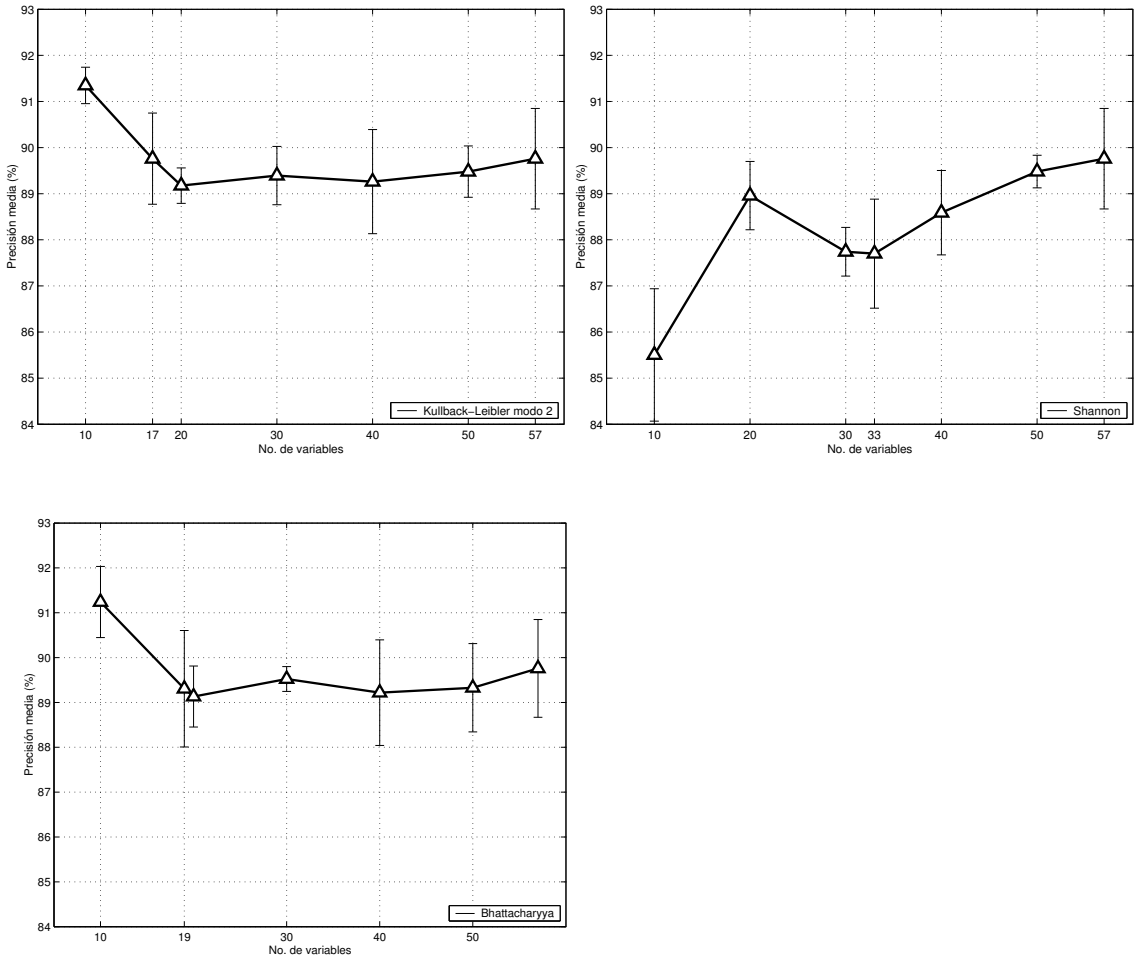


Figura 31: Resultados de las validaciones del modelo nB para la base de datos *Spambase* (cont).

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*	*TAN + 5-cv*	*p-valor*
Punto de corte					
IM	21	$88{,}890 \pm 1{,}508$	0,42	$91{,}980 \pm 0{,}391$	0,06
DE	34	$88{,}460 \pm 1{,}824$	0,55	$92{,}040 \pm 0{,}522$	0,31
MA	20	$89{,}150 \pm 0{,}664$	0,22	$92{,}070 \pm 0{,}603$	0,31
KL-1	20	$89{,}150 \pm 0{,}627$	0,42	$92{,}000 \pm 0{,}883$	0,42
KL-2	17	$89{,}760 \pm 0{,}988$	0,84	$92{,}110 \pm 0{,}640$	0,31
SH	33	$87{,}700 \pm 1{,}183$	0,06	$92{,}090 \pm 0{,}396$	0,06
BH	19	$89{,}300 \pm 1{,}298$	0,69	$91{,}800 \pm 1{,}906$	0,69
CFS	15	$90{,}349 \pm 0{,}430$	0,31	$92{,}023 \pm 0{,}488$	0,22
Todas	57	$89{,}760 \pm 1{,}090$	—	$92{,}780 \pm 0{,}645$	—

Cuadro 18: Resultados de las validaciones en los valores críticos para la base de datos *Spambase*.

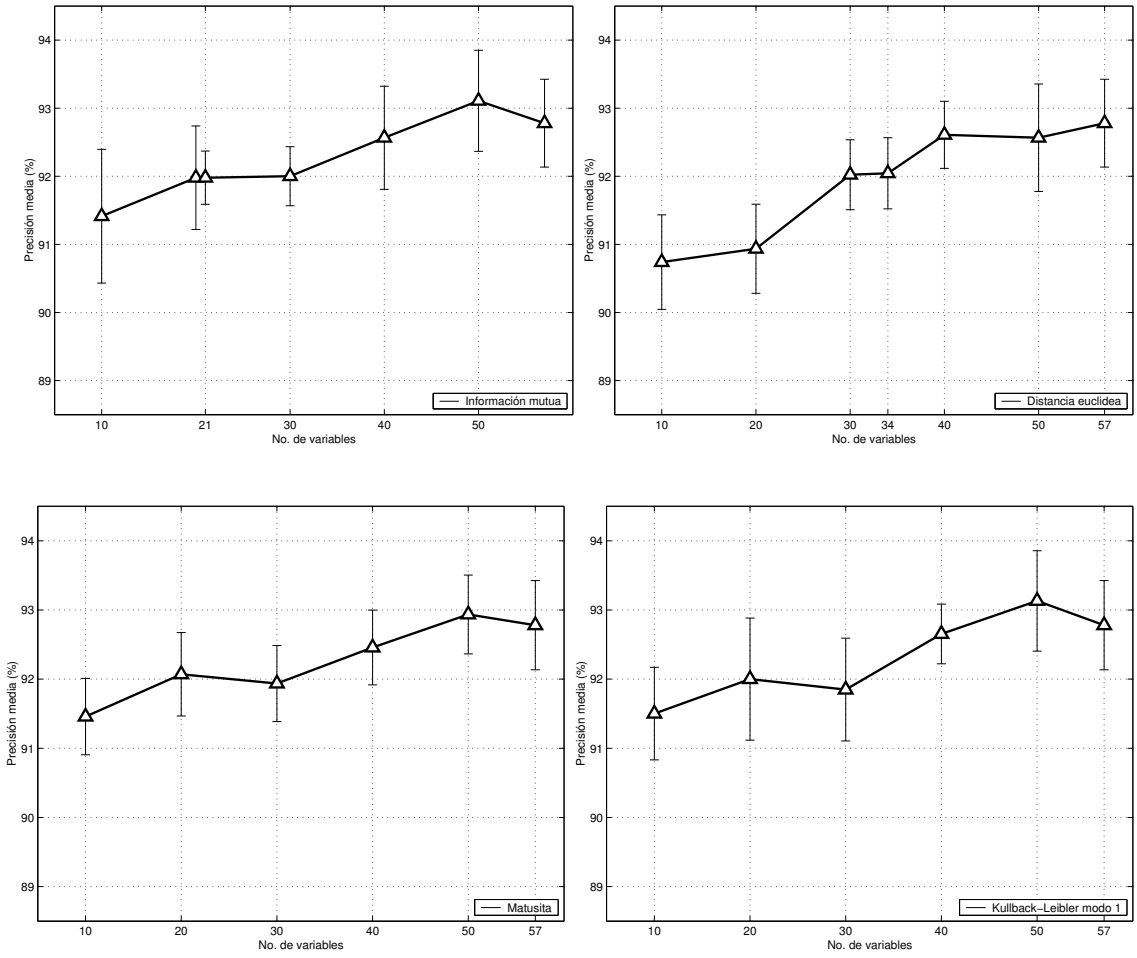


Figura 32: Resultados de las validaciones del modelo TAN para la base de datos *Spambase*.

En este conjunto de datos parece existir una gran correlación entre las variables que aparecen en los rankings a partir de las posiciones cercanas a la 20. Veamos cómo podemos llegar a esta conclusión a partir de los resultados obtenidos.

Si fijamos nuestra atención en los resultados en base al modelo nB vemos como para un número muy bajo de atributos, en el entorno de los diez primeros de los rankings, las precisiones obtenidas son mejores que en cualquier otro punto de la gráfica. Lo cual nos daría la pista para afirmar que existen variables redundantes en la base de datos, ya que al ir añadiendo variables a los subconjuntos, la precisión decae como tendencia general.

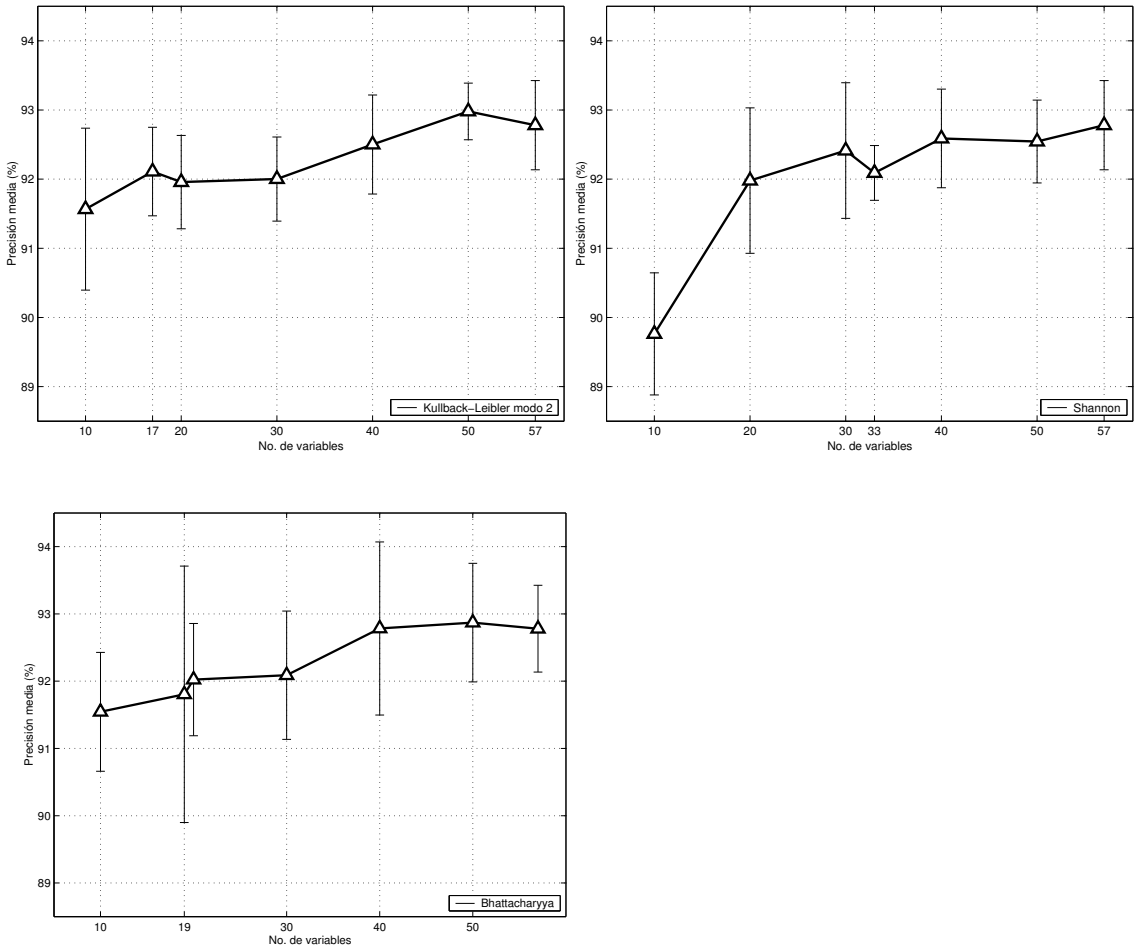


Figura 33: Resultados de las validaciones del modelo TAN para la base de datos *Spambase* (cont).

Se puede apreciar también que el método de corte no es capaz de detectar este punto de ruptura, devolviendo valores en el entorno de la posición 20 (17,19,20,21); posiciones en los que la precisión ha caído entre uno y dos puntos porcentuales. Comparando las precisiones obtenidas en estos puntos con la obtenida con todas las variables vemos que no existen diferencias significativas. En ese sentido las métricas se comportan bien, reducen el número de atributos y obtienen precisiones similares. El inconveniente es que sabemos que existen subconjuntos de atributos más pequeños para los que el modelo se comporta mejor.

Los resultados obtenidos en el caso de TAN nos dan la pista necesaria. La precisión en este caso cambia su tendencia y pasa a ser creciente en sus gráficas. Tomando como punto de partida las mismas precisiones sobre los diez primeros atributos de los rankings, al ir añadiendo más, las precisiones mejoran. Hecho que indica que los atributos no es que sean redundantes y dañen por ello la clasificación, sino que están muy correlados entre sí. Para los modelos nB estas dependencias condicionales producen caídas en su precisión, mientras que en los TAN, al ser detectadas, mejoran los porcentajes de clasificación.

Para el modelo TAN, el método de corte se muestra correcto. Los puntos fijados para evaluar los rankings retornan precisiones que no son significativamente diferentes a la evaluación del conjunto completo de atributos. A lo que debemos añadir que la disminución de atributos es considerable (de 59 a 20 en media).

CFS va a ser la técnica que, de nuevo, se comporte mejor. Es capaz de detectar de antemano esas dependencias y filtrar los atributos que realmente no aportan nada al problema. De ahí que seleccionando tan solo 15 de los 57 atributos sea capaz de obtener las mismas o incluso mejores (caso del nB) precisiones clasificatorias que en el caso general. Esta reducción lleva a evaluar un 73.68 % menos de información con respecto al conjunto inicial. Los quince atributos seleccionados son: `56` (longitud de la mayor cadena de caracteres en mayusculas); `55` (longitud media de las cadenas de caracteres en mayúsculas); `51`, `52`, `54` (porcentaje de ocurrencias de *[*, *!*, *#*); `4`, `6`, `15`, `18`, `20`, `22`, `23`, `24`, `25` y `26` (porcentaje de ocurrencias de *our*, *remove*, *free*, *you*, *your*, *000*, *money*, *hp*, *hpl*, *george*).

15.5. *Cirrosis*

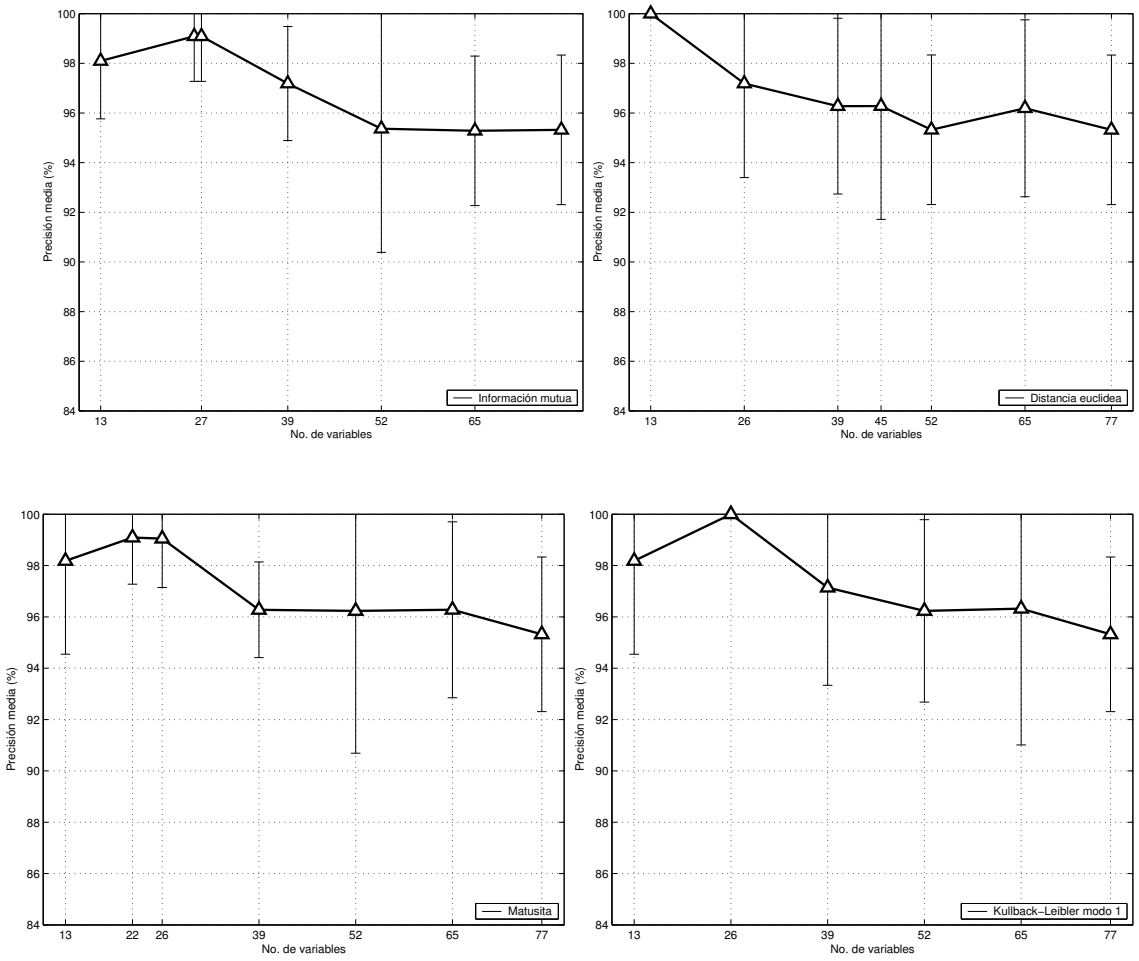


Figura 34: Resultados de las validaciones del modelo nB para la base de datos *Cirrosis*.

Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 34 y 35. Las Figuras 36 y 37 muestran los resultados obtenidos sobre esos mismos datos habiendo utilizado un modelo clasificatorio TAN.

El Cuadro 19 comprende los resultados cuantitavos resumen de la información mostrada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

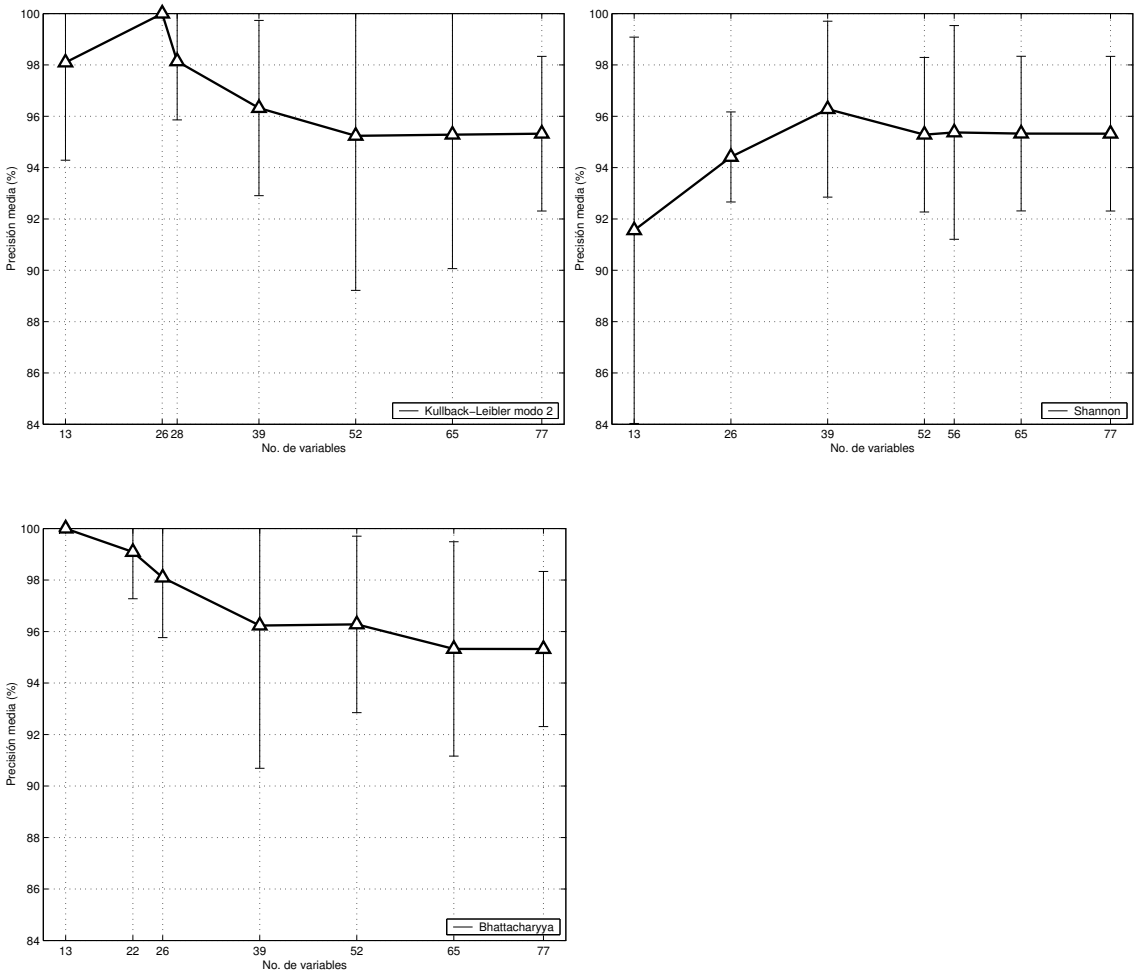


Figura 35: Resultados de las validaciones del modelo nB para la base de datos *Cirrosis* (cont).

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*	*TAN + 5-cv*	*p-valor*
Punto de corte					
IM	27	$99{,}090 \pm 1{,}818$	0,10	$96{,}230 \pm 5{,}546$	0,69
DE	45	$96{,}280 \pm 0{,}690$	0,69	$96{,}320 \pm 3{,}414$	0,69
MA	22	$99{,}090 \pm 1{,}818$	0,10	$97{,}190 \pm 2{,}298$	0,55
KL-1	26	$100{,}00 \pm 0{,}000$	0,03	$96{,}280 \pm 4{,}561$	1
KL-2	28	$98{,}140 \pm 2{,}281$	0,22	$98{,}140 \pm 2{,}281$	0,42
SH	56	$95{,}370 \pm 4{,}164$	0,84	$97{,}230 \pm 3{,}659$	0,55
BH	22	$99{,}090 \pm 1{,}818$	0,10	$97{,}230 \pm 2{,}263$	0,42
CFS	20	$99{,}047 \pm 1{,}904$	0,22	$97{,}229 \pm 3{,}659$	0,69
Todas	77	$95{,}320 \pm 3{,}013$	—	$95{,}320 \pm 4{,}164$	—

Cuadro 19: Resultados de las validaciones en los valores críticos para la base de datos *Cirrosis*.

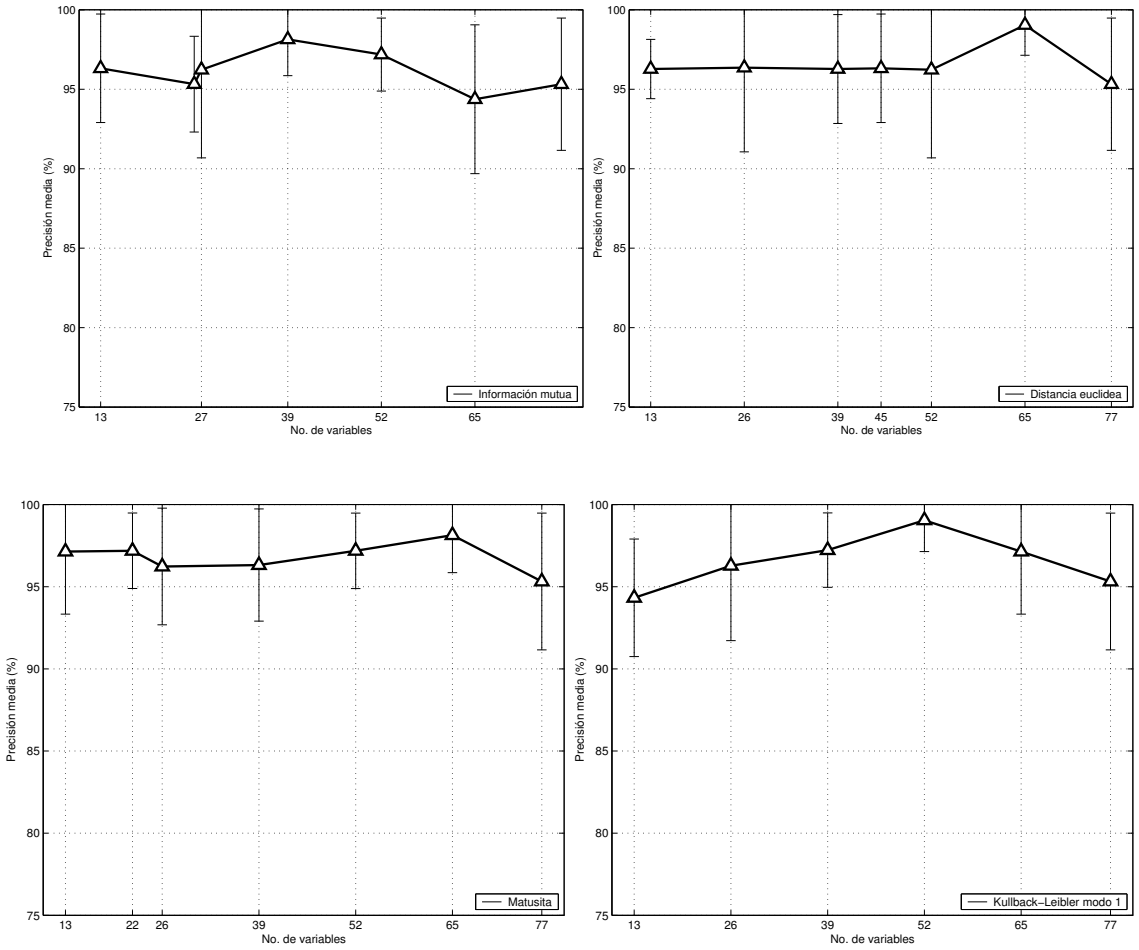


Figura 36: Resultados de las validaciones del modelo TAN para la base de datos *Cirrosis*.

Llama la atención en los resultados de esta base de datos como las precisiones obtenidas utilizando todas las variables predictoras en el caso de los modelos nB y en el caso de TAN son la misma, tan solo varía en un punto la desviación estándar del TAN (95.320 $\mp$ 3.013 frente a 95.320 $\mp$ 4.164). Cuando el modelo TAN no ha sido capaz de obtener mejores resultados, e incluso su resultado tiene mayor variabilidad, podemos asegurar que los modelos nB van a ser óptimos para realizar un proceso de clasificación en esta base de datos; hecho que se produce ya que, en base a los resultados, parece que las variables son condicionalmente independientes.

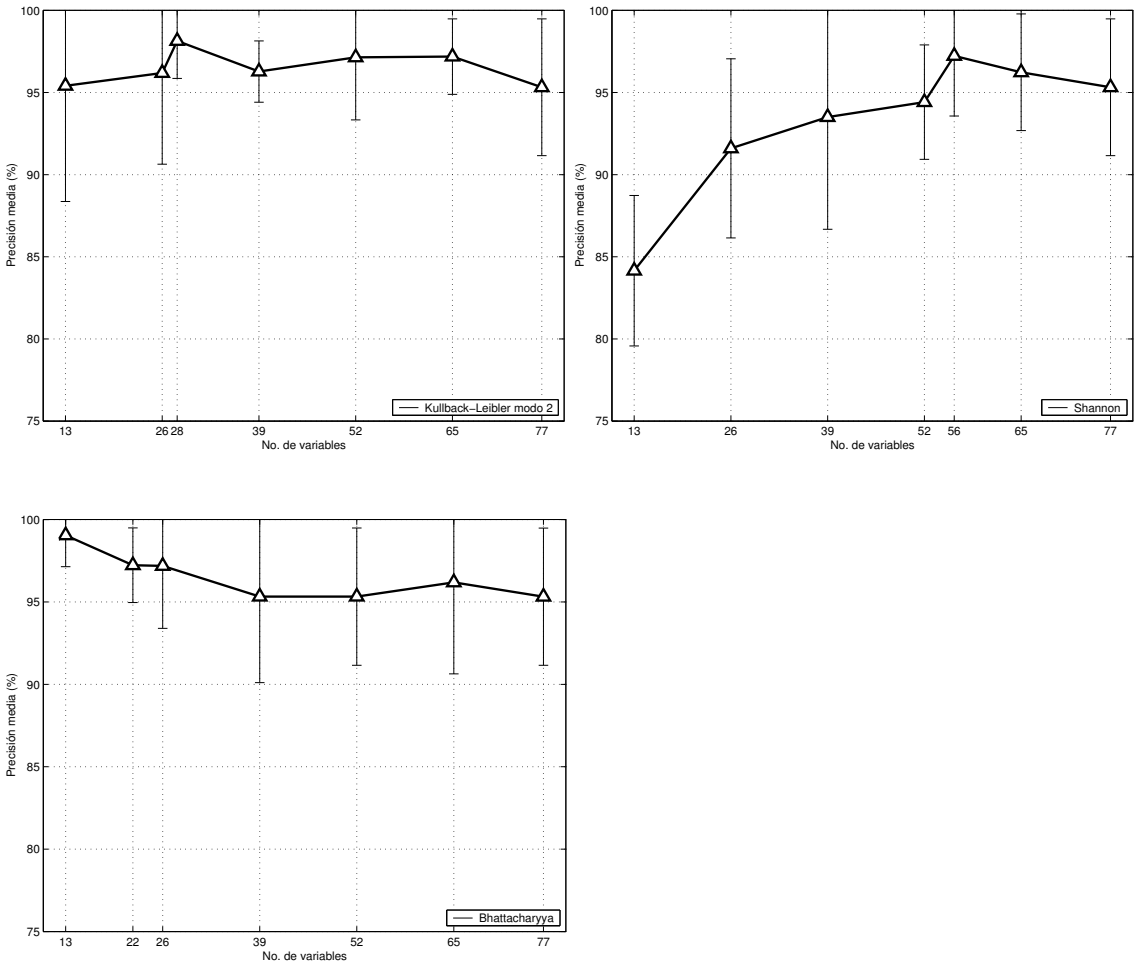


Figura 37: Resultados de las validaciones del modelo TAN para la base de datos *Cirrosis* (cont).

Los resultados obtenidos para las métricas univariadas y *CFS* utilizando TAN como modelo predictor dan como resultados precisiones que no son estadísticamente significativas respecto a la obtenida utilizando todas las variables. Lo cual ya es, de hecho, un buen resultado, ya que con subconjuntos de 20, 22, 26 o 27 variables se alcanza la misma precisión que con 77.

Los mejores resultados para esta base de datos se dan en la columna correspondiente al nB del Cuadro 19. En cinco de las ocho técnicas se alcanza una precisión del 99 %, y es para la métrica KL-1 junto con el punto de corte, para la que se llega a un 100 %

en la clasificación. Si analizamos las gráficas, vemos como las tendencias muestran que utilizando del orden de 26 atributos se obtienen los mejores resultados posibles (para KL-1 y KL-2 en ambos casos alcanzan el 100 %). Al ir añadiendo más atributos al modelo clasificatorio la precisión disminuye; parece que a partir de esas posiciones los atributos añadidos tienen un grado de redundancia con los demás.

Será *CFS* quien obtenga el subconjunto de atributos más refinado, con tan solo 20 de ellos. Las precisiones en ambos modelos son mejores a las generales, y son también de las más altas con respecto a las demás métricas (99.047 y 97.229 respectivamente). Las variables que forman este subconjunto de 20 son: `v24`, `v29`, `v30`, `v35`, `v36`, `v37`, `v38`, `v40`, `v41`, `v42`, `v62`, `v63`, `v64`, `v68`, `v69`, `v70`, `v71`, `v72`, `v76`. La reducción en la dimensionalidad de la base de datos va a ser de un 74 %. El subconjunto de atributos obtenidos del ranking de la métrica KL-1 en conjunción con el método de corte obtiene una reducción global del 66.23 %.

15.6. *Lymphoma*

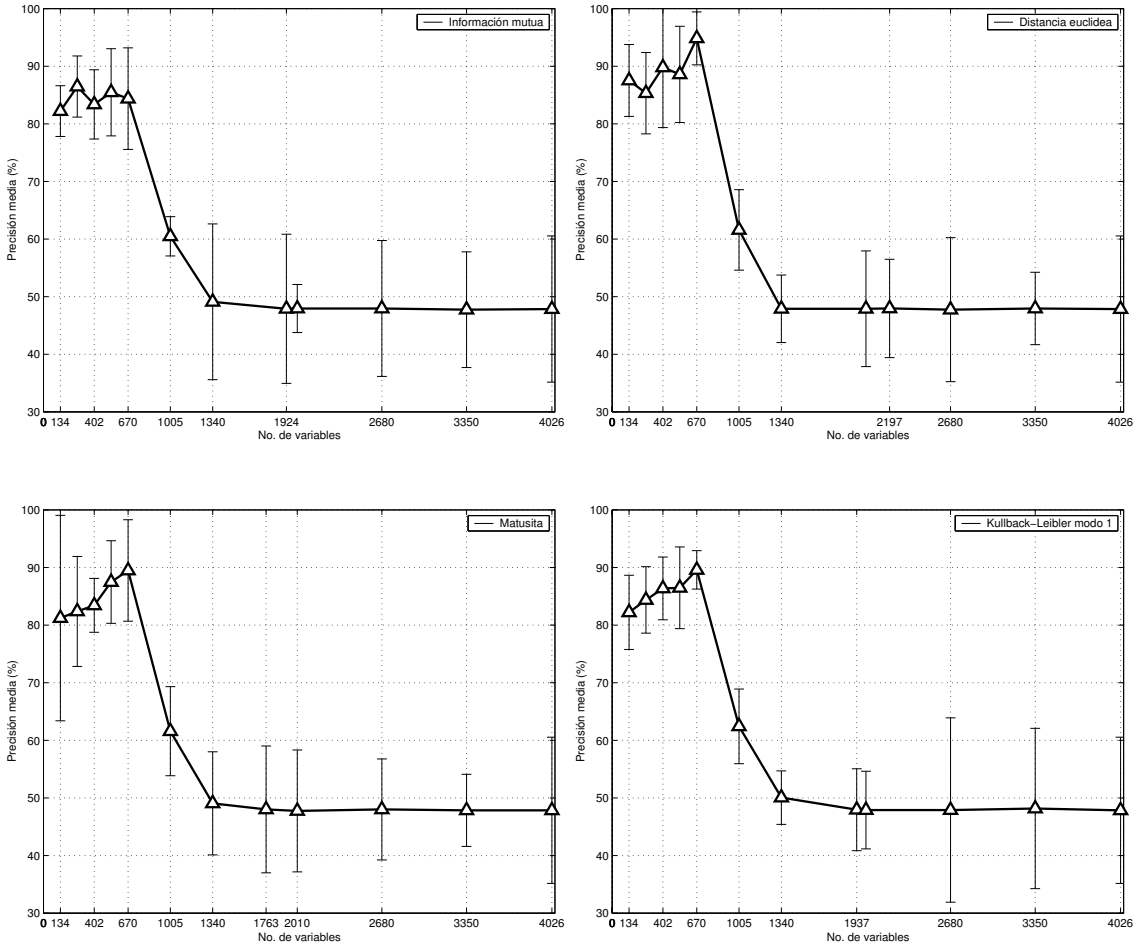


Figura 38: Resultados de las validaciones del modelo nB para la base de datos *Lymphoma*.

Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 38 y 39. Debido al número de variables a evaluar tanto en este conjunto de datos como en el siguiente microarray, *Lupus*, el tratamiento computacional de los modelos TAN era inviable desde el punto de vista de tiempo de cómputo. De tal forma que la experimentación en base a este modelo clasificatorio no ha podido ser realizada.

El Cuadro 20 comprende los resultados cuantitavos resumen de la información mos-

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*
Punto de corte			
IM	1924	$47{,}890 \pm 12{,}94$	1
DE	2197	$47{,}950 \pm 8{,}532$	0,84
MA	1763	$48{,}000 \pm 11{,}01$	1
KL-1	1937	$47{,}950 \pm 7{,}116$	0,84
KL-2	1448	$47{,}950 \pm 9{,}910$	0,84
SH	2535	$47{,}790 \pm 9{,}910$	1
BH	1322	$47{,}840 \pm 16{,}15$	1
CFS	525	$95{,}841 \pm 3{,}926$	0,01
Todas	4026	$47{,}841 \pm 12{,}689$	—

Cuadro 20: Resultados de las validaciones en los valores críticos para la base de datos *Lymphoma*.

trada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

La elección de los puntos de corte a evaluar para realizar las gráficas es asimétrico en este conjunto de datos. Al comenzar la experimentación con esta base de datos, siguiendo indicaciones y tendencias de la literatura relacionada, se sospechó que el uso de los primeros atributos de cada uno de los rankings obtendría resultados diferencialmente mejores que la división habitual en seis o siete intervalos de variables. De ahí que los puntos para los que se ha evaluado el modelo clasificatorio han sido (además de los indicados por método del codo): 134, 268, 402, 536, 670, 1005, 1340, 2010, 2680, 3350.

Como puede verse en el comportamiento de todas las gráficas, la sospechas se confirmaron. A partir del atributo 670 la precisión del modelo clasificatorio cae por debajo del 50 % (alrededor del 48 %), punto en el que se satura y que ya no es capaz de mejorar. Todos los atributos a partir del 670 no aportan ningún tipo de información y tan solo introducen ruido en el modelo: son atributos irrelevantes para el problema dados los ya seleccionados.

Al existir tantas variables en este tipo de problemas, el punto de corte automático

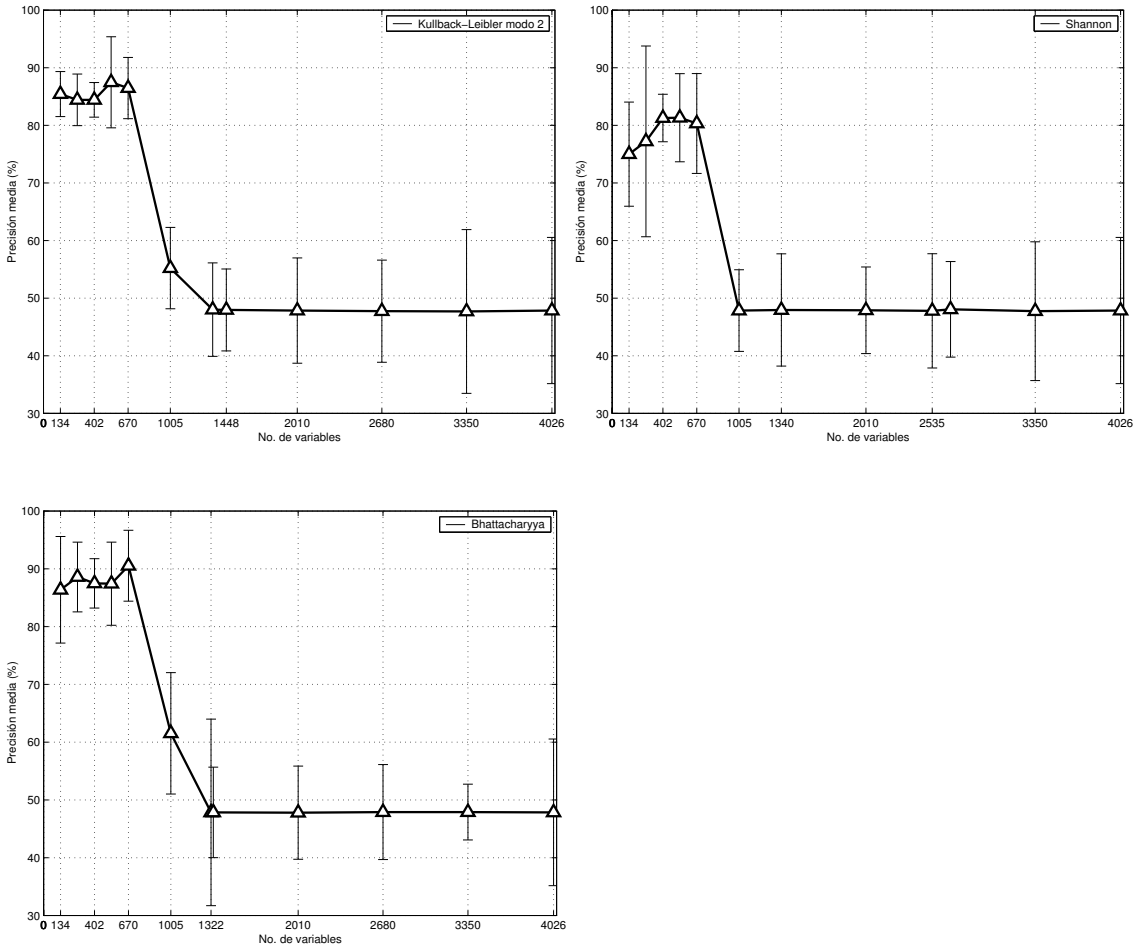


Figura 39: Resultados de las validaciones del modelo nB para la base de datos *Lymphoma* (cont).

realiza "correctamente" su función ya que selecciona subconjuntos de menor cardinalidad que obtienen clasificaciones semejantes al uso de todos ellos. Pero debido a la naturaleza del problema, aun cuando esto se cumple, los resultados obtenidos tanto por unos como por otros, no son los óptimos.

Este es el campo idóneo en el que aplicar *CFS*, por su propia definición matemática, será capaz de quitar todo ese ruido y realmente seleccionar esos atributos iniciales que, como se ve en las gráficas, sí tienen buenas precisiones clasificatorias. El resultado tras su ejecución no puede ser mejor, seleccionando 525 atributos, un 87 % de reducción

del fichero de datos, siendo capaz de alcanzar un 95.841 % de precisión media, con una desviación estándar pequeña en comparación con el resto de resultados.

Debemos notar, además, que el problema tenía siete clases diferentes, con dos instancias tan solo para algunas de ellas. Esos casos son complicados de clasificar al existir un número tan bajo de representantes, de ahí que el resultado obtenido por *CFS* sea aún más plausible.

15.7. *Lupus*

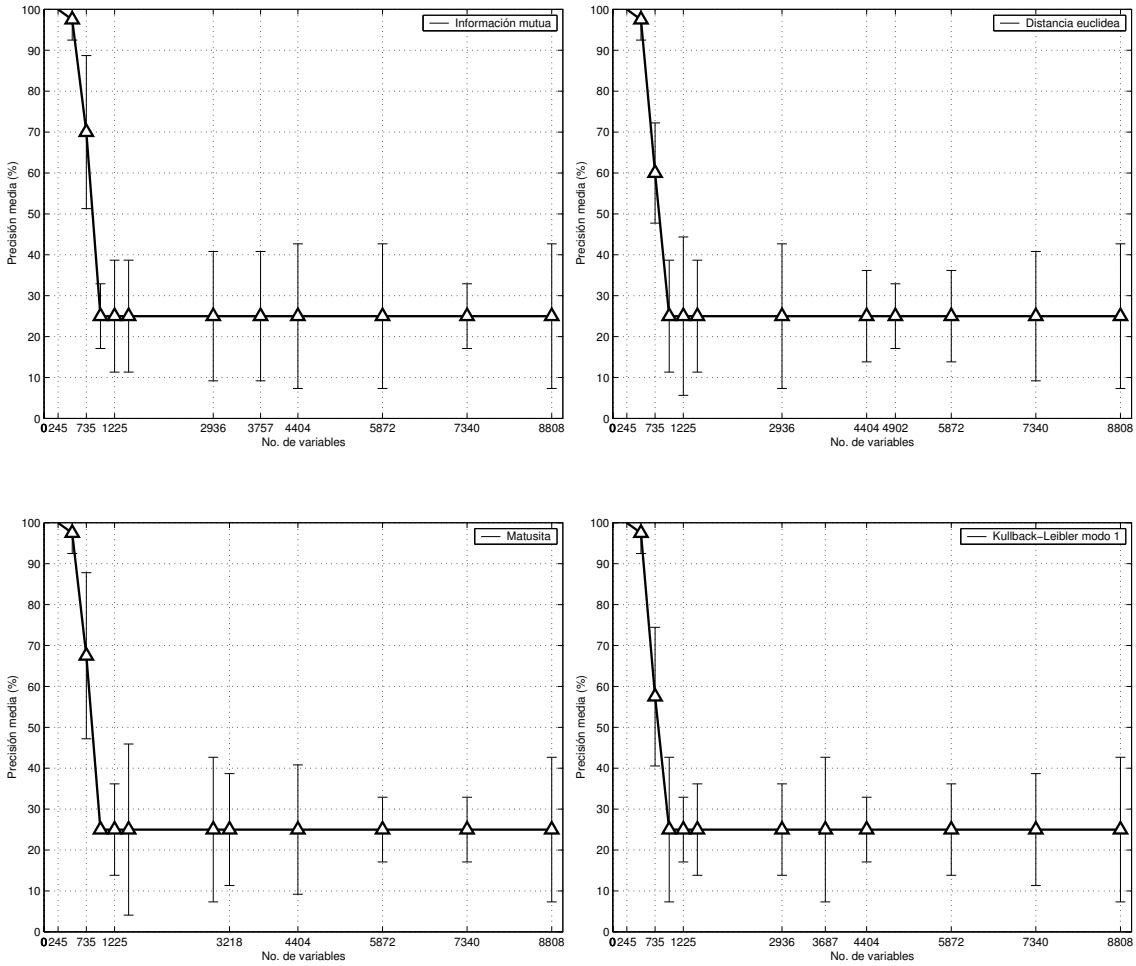


Figura 40: Resultados de las validaciones del modelo nB para la base de datos *Lupus*.

Al igual que en el caso de *Lymphoma*, la experimentación en base al modelo TAN no ha podido realizarse. Los resultados de los modelos nB, aplicados junto con las métricas univariadas, se incluyen en las Figuras 40 y 41.

El Cuadro 21 comprende los resultados cuantitavos resumen de la información mostrada en las diferentes gráficas, junto a los valores del test de significatividad entre los resultados obtenidos mediante las técnicas de selección de variables, y los resultados obtenidos utilizando todas las variables predictoras originales.

Análisis	*No. de variables*	*NB + 5-cv*	*p-valor*
Punto de corte			
IM	3757	$25{,}000 \pm 15{,}811$	1
DE	4902	$25{,}000 \pm 7{,}905$	1
MA	3218	$25{,}000 \pm 13{,}693$	1
KL-1	3687	$25{,}000 \pm 17{,}677$	1
KL-2	3761	$25{,}000 \pm 11{,}180$	1
SH	5676	$25{,}000 \pm 20{,}916$	1
BH	2922	$25{,}000 \pm 7{,}905$	1
CFS	31	$100{,}00 \pm 0{,}000$	0,01
Todas	8808	$25{,}000 \pm 17{,}677$	—

Cuadro 21: Resultados de las validaciones en los valores críticos para la base de datos *Lupus*.

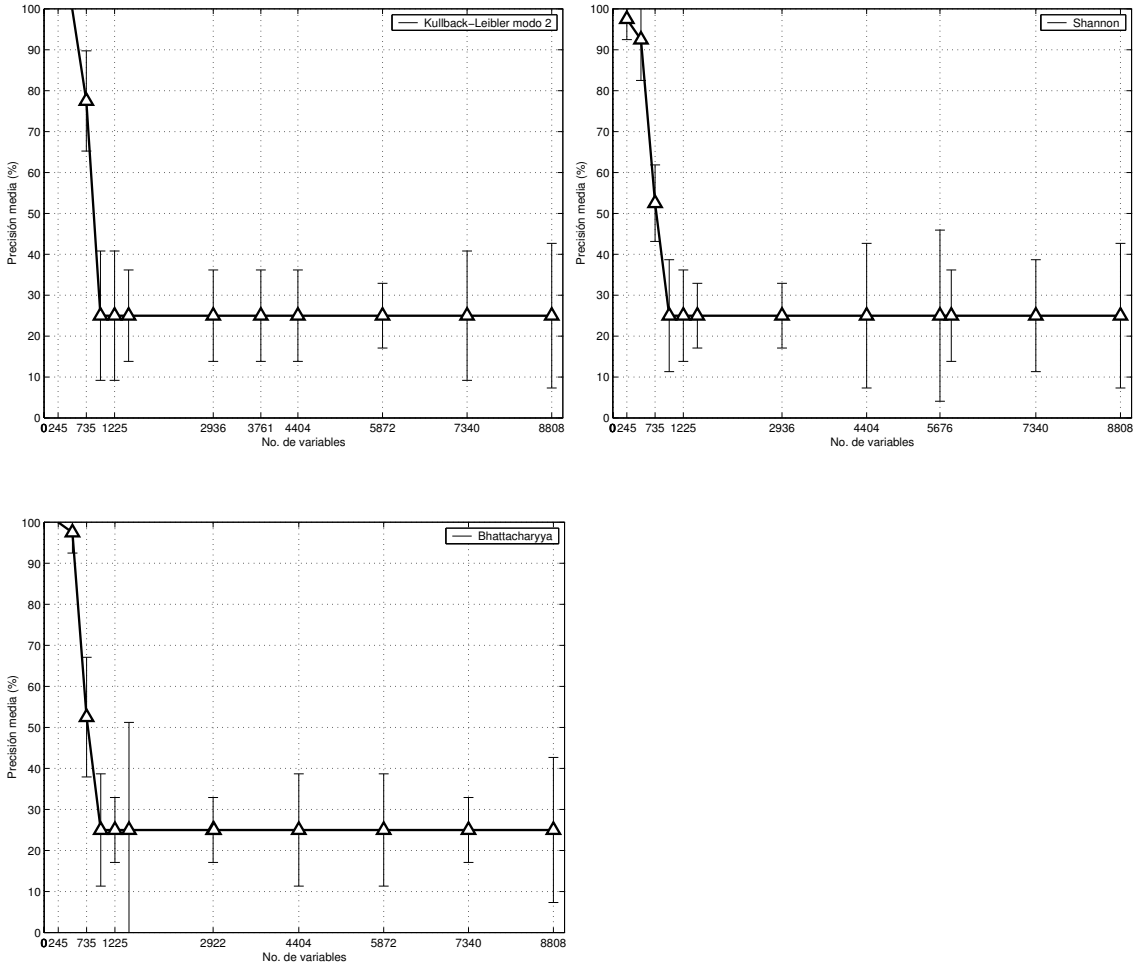


Figura 41: Resultados de las validaciones del modelo nB para la base de datos *Lupus* (cont).

Los puntos de corte para este conjunto de datos han sido (además de los automáticos recogidos en el Cuadro 21: 245, 490, 735, 980, 1225, 1468, 2936, 4404, 5872, 7340. El discurso sobre los resultados es análogo al realizado para el caso de *Lymphoma*. Experimentalmente se comprueba como con unos pocos atributos de los identificados como más relevantes en los rankings, las precisiones clasificatorias son altas, frente a la saturación del modelo clasificatorio al ir añadiendo atributos a partir del 490.

En esta base de datos se da la circunstancia de que el número de casos es extremadamente bajo, de ahí que con los primeros atributos seamos capaces de alcanzar un 100 % de precisión en la clasificación. *CFS* destaca 31 atributos del fichero, llegando la reducción a ser de un 99.65 % del tamaño original. En base a esos 31 atributos, un modelo nB alcanza el 100 % de precisión clasificatoria.

16. Conclusiones

Con referencia a los objetivos fijados inicialmente para el proyecto, las conclusiones de los resultados obtenidos son básicamente positivas. La realización de las tareas ha sido posible, integrando en el sistema Elvira los procesos de selección de variables. Procesos que en la actualidad se encuentran en ampliación por parte de otros desarrolladores de la plataforma.

En las reuniones del proyecto Elvira realizadas en el año 2003 y 2004[27] fueron presentadas las mejoras de la interfaz gráfica, así como las técnicas de FSS implementadas, teniendo una amplia aceptación y generando nuevas colaboraciones entre el alumno y los desarrolladores de otras universidades.

En cuanto a la aplicación de estas técnicas sobre bases de datos génicas, el grupo de investigación continúa su colaboración con el grupo de Biología de Lejona, habiendo llegado ya por medio de las técnicas implementadas en el marco de este proyecto, a resultados que están siendo estudiados por los biólogos.

A modo de conclusiones generales al trabajo realizado, podemos enumerar:

1. **Posibilidad de identificar patrones en los datos**. El aplicar técnicas de selección de variables sobre un conjunto de datos más o menos "denso" hace que esos datos "se aclaren" al reducir los atributos y ser la cardinalidad de los casos mucho menor. Este proceso puede conllevar la detección de patrones ocultos en los datos mediante un estudio directo de ellos, estudio que antes del proceso de selección era del todo imposible.

2. **Rapidez en los procesos clasificatorios**. Al ser esta cardinalidad baja, los tiempos de cómputo para procesos clasificatorios se reducen también drásticamente. Existen bases de datos para las cuales la aplicación de un modelo muy complejo no es viable

[27] *http://leo.ugr.es/∼elvira/Meetings/meetings.html*

de inicio, pero que tras los procesos de selección de variables permiten la aplicación de dichas técnicas más complejas.

3. **Mejorar la *calidad* de los atributos**. Entendemos por *calidad* el grado de relevancia de un atributo en el dominio en que está siendo evaluado. Mediante las técnicas presentadas a lo largo de este proyecto pueden ser identificados atributos que en el dominio de estudio no aporten apenas información. Cuando ese dominio está bajo el control del investigador, siendo éste el que selecciona qué atributos debe evaluar en el problema, está obteniendo una prueba inicial de relevancia. De tal forma que podrá ir seleccionando diferentes atributos a evaluar, descartando aquellos que se muestren como poco relevantes, adecuándose al problema sobre el que trabaja.

4. **Reducción de costes**. En problemas de índole clínico, el hecho de reducir las pruebas necesarias para evaluar una determinada dolencia puede revertir en un ahorro considerable de dinero, así como en una disminución de las molestias que algunas técnicas causan a los pacientes.

5. **Novedad en microarrays de ADN**. Las técnicas informáticas aplicadas al análisis de microarrays de ADN hasta la fecha se han centrado siempre en el aspecto no supervisado, calculando particiones de genes. Es una novedad el hecho de tratarlos desde un punto de vista supervisado, aplicando las técnicas de selección de variables de forma que los modelos clasificatorios realicen correctamente su labor.

Como conclusión general a las técnicas presentadas hay que hacer hincapié en la recomendación de aplicar este tipo de preprocesamiento siempre que vaya a trabajarse sobre procesos de clasificación supervisada. El coste computacional de las técnicas es muy reducido, mientras que los beneficios que se derivan de ellas pueden llegar a ser muy grandes.

Uno de los futuros estudios derivados del trabajo realizado a lo largo de este proyecto será el análisis de los comportamientos de las técnicas de selección de variables cuando los conjuntos de selección y entrenamiento son diferentes. Tanto en toda la literatura consultada, como en los diseños realizados en el marco del proyecto, el conjunto sobre el que se realiza la selección de variables es el mismo que luego es usado para el aprendizaje automático. Queda por estudiar la eficiencia de estos datos cuando, por ejemplo dividiendo la base de datos original, se utilice una partición para el preprocesamiento y otra diferente para el aprendizaje. Podrían plantearse técnicas de validaciones cruzadas sobre la selección de variables, como existen para las validaciones de los algoritmos clasificatorios.

16.1. Métodos de ranking

Centrando el análisis en las técnicas univariadas de generación de rankings debemos recalcar el punto tercero de las conclusiones generales: se trata de técnicas rápidas en tiempo de cómputo, orden lineal $O(n)$, que pueden dar a simple vista una idea de la relevancia que cada atributo o variable tiene en un problema supervisado.

No puede afirmarse que exista alguna de ellas que sea mejor que las demás y se considera que deben ser aplicadas al menos tres o cuatro de ellas por cada problema. Otra técnica recomendable es el cálculo de rankings *consensuados*; a partir de todos los rankings se ponderan las posiciones de los atributos en cada uno de ellos y se genera un ranking unificado o *consensuado*, que refleje qué atributos han obtenido mayores coeficientes de relevancia y cuáles menores.

16.2. CFS - Correlation-based Feature Selection

Dentro de las técnicas que seleccionan subconjuntos de atributos, *CFS* viene demostrándose como una de las más ampliamente utilizadas en la investigación. Es una

técnica *filter*, rápida, aunque el evaluar correlaciones dos a dos hace que su tiempo de cómputo sea de orden cuadrático $O(n^2)$ haciéndola más lenta que las técnicas de ranking junto a un método de corte.

Pero el aumento del tiempo de cómputo está de sobra justificado ante los resultados obtenidos. En todas las bases de datos utilizadas en el proyecto, ha obtenido las reducciones más drásticas en tamaño, consiguiendo precisiones clasificatorias siempre estadísticamente iguales que el caso general, al utilizar un clasificador naïve Bayes.

Su eficiencia es especialmente resaltable cuando la base de datos es de una cardinalidad alta; a partir de 100 atributos. Es en estos casos cuando está especialmente recomendada ya que las precisiones obtenidas por el subconjunto de atributos que selecciona son casi siempre mejores que cualquier otra de las técnicas, o que con el uso de todos los atributos.

Parte VI

Apéndices

A. Contenido del CD

El CD que acompaña a la memoria tiene cuatro directorios principales: El directorio `jvm` contiene los instaladores de la plataforma de ejecución Java 1.4.2 para sistemas operativos *Windows* o *Linux*. El directorio `elvira` contiene el código fuente de Elvira descargado de su sitio oficial a fecha 28-Ago-2004; se incluyen también subdirectorios con el código fuente ya compilado y empaquetado en un fichero Java ejecutable. El tercer directorio, `datasets`, incluye las bases de datos con las que se ha realizado la experimentación del proyecto.

Por último, en el directorio `minutes` se encuentran las actas de seguimiento del proyecto, y en el directorio raíz del CD se incluye una copia digital de esta memoria, en formato PS y en formato PDF.

jvm	\j2re-1_4_2_05-linux-i586.bin	Java Runtime 1.4.2 para sistemas *Linux x86*
	\j2re-1_4_2_05-windows-i586.exe	Java Runtime 1.4.2 para sistemas *Microsoft Windows x86*
elvira	**src**	Código fuente del sistema Elvira
	src-compiled	Código fuente compilado del sistema Elvira
	jar-package	Fichero Java ejecutable, *Elvira.jar*
datasets		Bases de datos del proyecto en formato `dbc`
minutes		Actas de seguimiento del proyecto
memoria.ps		Memoria del proyecto en formato *PostScript*
memoria.pdf		Memoria del proyecto en formato *PDF*

B. Ejecución del sistema

Como se ha expuesto en los capítulos dedicados a la explicación del sistema Elvira, éste posee dos modos de funcionamiento: mediante la tradicional línea de comandos, o mediante el uso de la interfaz gráfica. En este apéndice se recogen explicaciones básicas de cómo invocar los métodos de selección de variables implementados en el marco del proyecto.

Los requisitos de *hardware* y *software* necesarios para utilizar el sistema Elvira son:

1. **Java Virtual Machine 1.4.2 o superior**. Elvira precisa de una máquina virtual Java instalada en el equipo sobre el que se vaya a ejecutar. En el CD que se adjunta a la memoria se incluyen los ficheros de instalación para plataformas *Windows* y *Linux*.

2. **256 Mb de RAM**. Por defecto Java reserva 64 Mb para la ejecución de sus procesos; cuando la base de datos que se utilice tenga una dimensión grande, Elvira necesitará más memoria para poder cargarla completa. En la Sección B.1 se indica como ampliar el tamaño de la memoria reservada para la ejecución del sistema.

3. **CPU de 1 Ghz. o superior**. Elvira puede trabajar con sistemas más lentos, pero, en función de la tarea que realicemos, el tiempo de cómputo puede ser muy alto en sistemas inferiores a 1 Ghz. de CPU.

B.1. Línea de comandos

La mayoría de clases que integran el sistema disponen de métodos ejecutables que permiten la invocación de tareas desde la línea de comandos. Debemos tener presente que para poder invocar una clase Java, ésta ha debido ser previamente compilada. En el CD se ha incluído el código fuente del sistema, así como la compilación de dicho código fuente. La línea de comandos a utilizar para invocar la clase `FilterMeasures` es:

$ java elvira.learning.preprocessing.FilterMeasures

```
[ruben@sipi42 classes]$ java elvira/learning/preprocessing/FilterMeasures

Usage: FilterMeasures input.dbc filterOption [noVar] [output.dbc]
       FilterMeasures input.dbc 7 [output.dbc]

filterOption: 0 - Mutual information
              1 - Euclidean distance
              2 - Matusita distance
              3 - Kullback-Leibler mode 1
              4 - Kullback-Leibler mode 2
              5 - Shanon entropy
              6 - Bhattacharyya metric
              7 - Correlation-based Feature Selection

noVar:        number of variables to be displayed/saved
              0 - the number of variables will be determined automatically

output.dbc: file where the 'noVar' filtered variables will be saved, including the class one (the last of them)
            in CFS case 'output.dbc' file will include the selected variables by the method and the class
```

Figura 42: Invocación de la clase `FilterMeasures` sin parámetros.

La salida producida por una invocación sin parámetros muestra las opciones de las que disponemos; en la Figura 42 se recogen estas opciones. Los parámetros del método son:

- **input.dbc** – fichero de casos de entrada.
- **filterOption** – mediante un índice se indica el método *filter* a utilizar. Entre 0 y 6 para las métricas, y 7 para el método CFS.
- **noVar** *(parámetro opcional)* – número de variables incluídas en la salida de la selección. Un valor de 0 en este parámetro hará que sea utilizado el método de codo de detección automática del punto de corte. Si hemos configurado el método *CFS* en el parámetro anterior, éste no debe ser indicado.
- **output.dbc** *(parámetro opcional)* – fichero de salida en el que se proyectará la base de datos original en función a los atributos seleccionados, incluyendo como última variable del fichero a la variable clase.

Una ejecución utilizando como métrica la información mutua y el punto de corte del codo sería:

$ java elvira.learning.preprocessing.FilterMeasures headache.dbc 0 0

Podemos guardar la proyección de la selección en el fichero `head-proy.dbc` invocando a la clase:

$ java elvira.learning.preprocessing.FilterMeasures headache.dbc 0 0 head-proy.dbc

```
Node                          Filter metric
-----------------------------------------------
node10                        0.376886604015881
node8                         0.21664010847041915
node6                         0.1285196121289348
node4                         0.06180905519954172
File head-proy.dbc correctly written
```

Figura 43: Invocación de la clase `FilterMeasures` proyectando una base de datos.

La salida de los procesos de selección muestran, para cada uno de los nodos, el ranking de esas variables en función de la métrica utilizada. En el caso de *CFS*, el valor de la métrica indica el valor del heurístico que guía el proceso de búsqueda. En la Figura 43 puede verse la salida de la última invocación.

Para poder asignar más de 64 Mb de memoria a la ejecución de la clase, por ejemplo, al trabajar con microarrays de ADN, el modificador que debemos indicar a Java es `-Xmx[no. de Mb]m`. El ejemplo anterior, usando 256 Mb de memoria quedaría:

$ java -Xmx256m elvira.learning.preprocessing.FilterMeasures headache.dbc 0 0

B.2. Interfaz gráfica de usuario

La interfaz gráfica de Elvira puede ser invocada desde la línea de comandos al igual que cualquier otra de las clases que dispongan de un método ejecutable:

$ java elvira.Elvira

Si disponemos de un fichero autoejecutabe de Java, un fichero de tipo `jar`, podemos ejecutarlo directamente. La clase por defecto que se ejecutará será la que invoca la interfaz gráfica:

$ java -jar Elvira.jar

En los sistemas *Windows*, si la instalación de la máquina virtual Java ha sido configurada con las opciones por defecto, podremos lanzar la interfaz gráfica haciendo doble *click* sobre el fichero `jar`. Para los tres casos, y tras la pantalla de progreso incial, obtendremos una pantalla similar a la mostrada en la Figura 44.

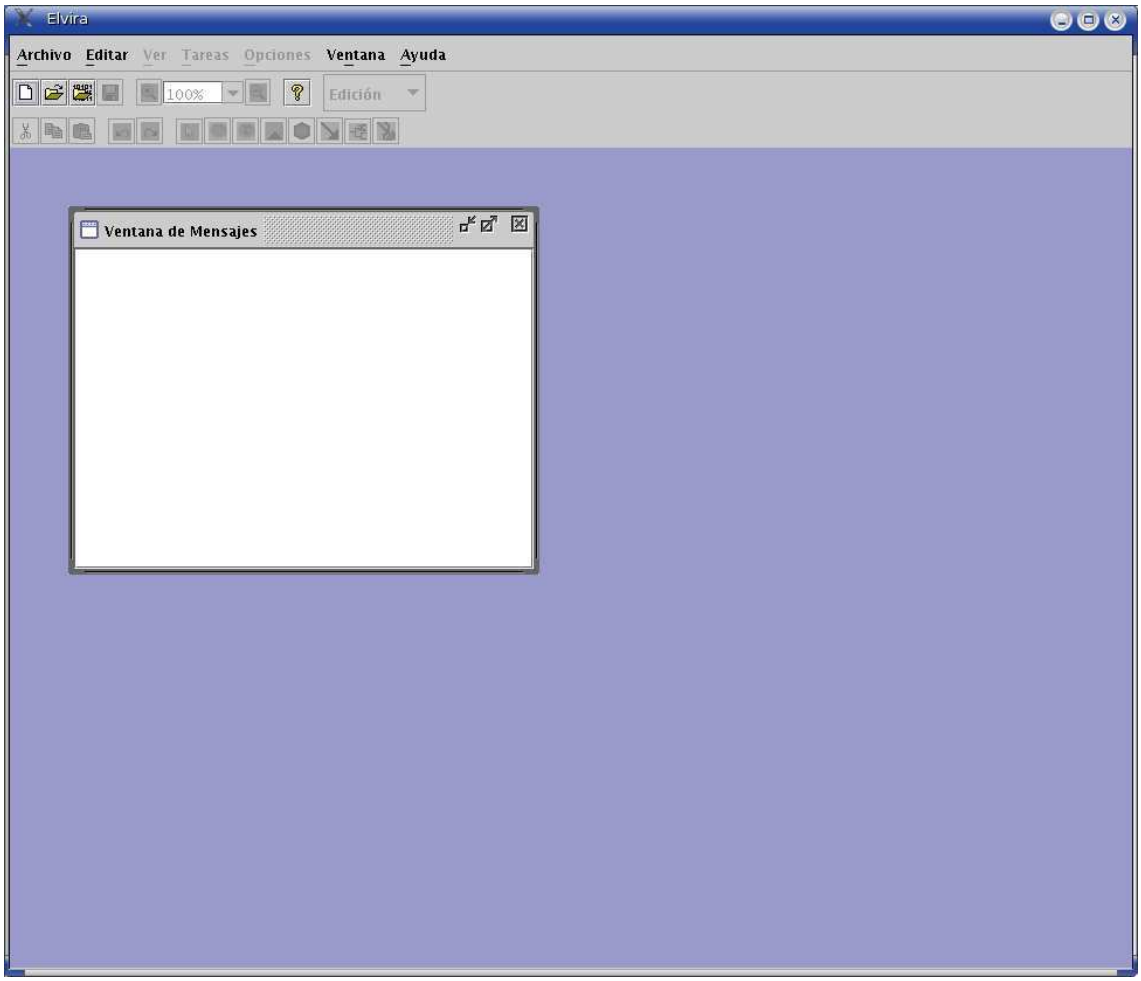


Figura 44: Pantalla inicial de la interfaz gráfica de Elvira.

Para acceder al diálogo que integra el tratamiento de ficheros de casos –Figura 45–, seleccionaremos la entrada *Abrir fichero de casos ...* de la entrada *Archivo* de la barra de menús. El acceso rápido al diálogo lo forman la combinación de teclas Control y D pulsadas simultáneamente.

Lo primero que debemos hacer es seleccionar el fichero de casos sobre el que vamos

Figura 45: Diálogo para el manejo de ficheros de casos.

a trabajar. Pulsando el botón de *'Seleccionar'* tendremos acceso a nuestro disco, seleccionando el fichero `dbc` de trabajo. Por defecto, el diálogo aparece incialmente con las opciones de imputación de valores perdidos. Suponiendo que el fichero no tiene casos perdidos y todos los atributos son discretos, podemos pasar a la selección de variables. Para ello, cambiamos a la pestaña llamada *'Medidas filter'* –Figura 46–.

En el menú desplegable seleccionamos la técnica que deseamos utilizar. En función de la técnica que seleccionamos, el cuadro informativo mostrará el estadístico que es calculado en esa medida *filter*. En el ejemplo utilizaremos la métrica *Bhattacharyya* para calcular el ranking de la base de datos seleccionada. Tras pulsar en *'Procesar'* y haber calculado los resultados, el sistema nos muestra el cuadro de la Figura 47 con los resultados.

Desde la interfaz también disponemos de la posibilidad de guardar en disco las proyecciones de los ficheros de casos originales, en función de las selecciones que realicemos. Para ello debemos seleccionar la opción *'Guardar proyección'*, una vez hecho esto, se habilitarán las opciones de proyección. Podemos indicar el número de variables que

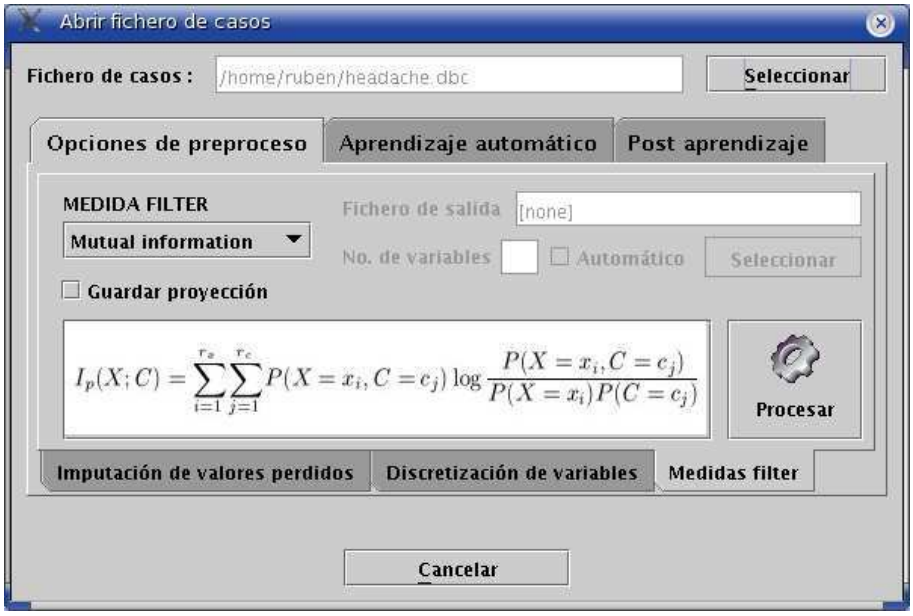


Figura 46: Pestaña de medidas de filtrado con un fichero de casos ya seleccionado.

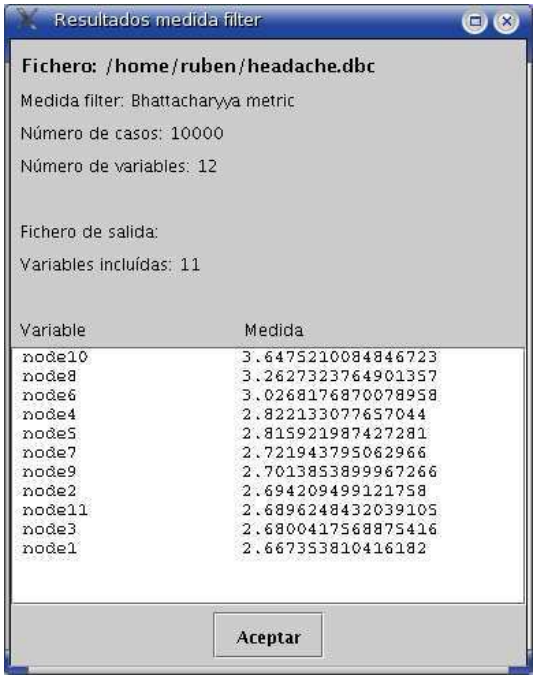


Figura 47: Ranking de las variables de la base de datos *Headache* en base a la métrica *Bhattacharyya*.

queremos seleccionar de un ranking, comenzando desde la primera en el cuadro *'No. de variables'*. Si queremos utilizar el método del codo del punto de corte debemos seleccionar la opción *'Automático'*. Deberemos, también, indicar mediante el botón *'Seleccionar'* el nombre del fichero destino donde almacenar la base de datos proyectada.

Figura 48: Resultados de una selección CFS sobre la base de datos *Headache*, almacenando el fichero proyección en disco.

En caso del *CFS*, se guardará el suconjunto de atributos calculado por el método. En la Figura 48 se incluye el resultado producido por esta configuración para la base de datos *Headache*.

C. Aportaciones del proyecto a la comunidad

El proyecto Elvira no tiene ningún tipo de licencia de uso, aunque se encuentra en estudio aplicar alguna de las licencias existentes de código libre (GPL, LGPL, etc.). Actualmente, el código del sistema está disponible para descarga gratuita en la página principal del proyecto en *http://leo.ugr.es/∼elvira*

No existe ningún tipo de control sobre esas descargas así que no existen estadísticas de cuántas personas pueden haber probado el sistema o estén trabajando con él. En recientes fechas ha llegado a conocimiento del alumno encargado de este proyecto la existencia del siguiente artículo:

Título:	Feature Selection-Ranking Methods In a Very Large Electric Database
Autor(es):	Manuel Mejía-Lavalle, Guillermo Rodríguez-Ortiz, Gustavo Arroyo, Eduardo F. Morales
Publicación:	Third edition of the Mexican International Conference on Artificial Intelligence
Fecha:	April 26-30, 2004
Lugar:	Mexico City, Mexico

En el artículo, se analiza la información de una central eléctrica de la capital Mejicana, utilizando selección de variables. Las siete métricas implementadas en este proyecto son comparadas con otras dos técnicas de selección de variables. Empirícamente demuestran que el uso de las métricas mejora los procesos de detección de fugas eléctricas en la central.

D. Agradecimientos

Desde aquí me gustaría agradecer el apoyo y colaboración de :

- Mis padres, Félix y Mari Carmen, por haberme apoyado en todo lo que necesité durante estos años y haber tenido la suficiente paciencia como para ver llegar este momento.
- Mi hermana Puy y cuñado Jose, por sus sabios consejos y ánimos en los momentos difíciles, y por ser en parte responsables de que esta memoria se haya podido escribir.
- Mi tutor de proyecto, Iñaki Inza, por haber sabido guiarme en el trabajo a realizar, frenando mi ansia cuando debía.
- A Pedro Larrañaga, Jose Antonio Lozano e Iñaki Inza por haber confiado en mí y en el trabajo que realizo.
- A todos los miembros del laboratorio 310; Rosa, Guzmán, Jose Luis, Roberto, Javi, Ramón y Aritz, por su colaboración y paciencia conmigo.

Referencias

Aherne, F. J., Thacker, N. A., and Rockett, P. I. (1997). The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 32(4):001–007.

Ash A. Alizadeh et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511.

Bayes, T. (1764). Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53.

Bell, D. A. and Wang, H. (2000). A formalism for relevance and its application in feature subset selection. *Machine Learning*, 41(2):175–195.

Ben-Bassat, M. (1982). Use of distance measures, information measures and error bounds in feature evaluation. In Krishnaiah, P. R. and Kanal, L.Ñ., editors, *Handbook of Statistics*, volume 2, pages 773–791. North-Holland Publishing Company.

Binder, J., Koller, D., Russell, S. J., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2–3):213–244.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.

Cano, A., Moral, S., and Salmerón, A. (2000). Penniless propagation in join trees. *International Journal of Intelligent Systems*, 15:1027–1059.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In *Proceedings of the European Working Session on Learning*, pages 164–178.

Clark, P. and T. Niblett (1989). The CN2 induction algorithm. *Machine Learning*, 3:261–283.

Cooper, G. F. (1984). Nestor: A computer-based medical diagnostic that integrates causal and probabilistic knowledge. HPP 84-48, Standford University, Standford, California.

Cooper, G. F. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–348.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, 13:21–27.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press.

Das, S. (2001). Filters, wrappers, and a boosting-based hybrid for feature selection. In *Proceedings of the Eighteenth International Congress of Machine Learning*, pages 74–81, Williams College, Massachusetts.

de Campos, L. M., Gámez, J. A., and Moral, S. (2002). On the problem of performing exact partial abductive inference in bayesian networks using junction trees. In Bouchon-Meunier, B., Gutierrez-Rios, J., Magdalena, L., and Yager, R., editors, *Technologies for Constructing Intelligent Systems 2: Tools*, Physica Verlag, pages 289–302. Springer Verlag.

de Campos, L. M. and Puerta, J. M. (2001). Stochastic local and distributed search algorithms for learning belief networks. In *Proceedings of the Third International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, pages 109–115.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202.

Elvira Consortium (2002). Elvira: An environment for probabilistic graphical models. In Gámez, J. A. and Salmerón, A., editors, *Electronic Proceedings of the First European Workshop on Probabilistic Graphical Models*, Cuenca, Spain.

Fisher, R. A. (1936). The use of multiple measurements. *Annals of Eugenics*, 7:179–188.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2):131–164.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research: Special Issue on Variable and Feature Selection*, 3:1157–1182.

Hall, M. A. and Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Konwledge and Data Engineering*, 15(3):1437–1447.

Hall, M. A. and Smith, L. A. (1997). Feature subset selection: A correlation based filter approach. In et al., N. K., editor, *Proceedings of the Fourth International Conference on Neural Information Processing and Intelligent Information Systems*, pages 855–858, Dunedin.

Hall, M. A. and Smith, L. A. (1999). Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *Proceedings of the Florida Artificial Intelligence Research Symposium*, pages 235–239, Orlando, Florida.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Hosmer, D. and Lemeshow, S. (1989). *Applied Logistic Regression*. John Wiley and Sons.

Inza, I., Larrañaga, P., Etxeberria, R., and Sierra, B. (2000). Feature subset selection by Bayesian networks based optimization. *Artificial Intelligence*, 123(1-2):157–184.

Jensen, F. (2001). *Bayesian Networks and Decision Graphs*. Springer Verlag.

Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4:269–282.

Kerber, R. (1992). Chimerge: Discretization for numeric attributes. In Press, A., editor, *National Conference on Artificial Intelligence*, pages 123–128.

Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 249–256, Abeerden, Scotland.

Kohavi, R. (1995). *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Department of Computer Science, Standford University.

Kohavi, R. and John, G. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. In *Proceedings of the Fifth European COnference on Machine Learning*, pages 171–182, Heraclion, Crete, Greece.

Langley, P. and Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Morgan Kaufmann.

Larrañaga, P. (2001). An introduction to probabilistic graphical models. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of distribution algorithms – A New Toold for Evolutionary Computation*. Kluwer Academic Publishers.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society Series B*, 50(2):157–224.

Lee, C. and Lee, G. G. (2003). Information gain and divergence-based feature selection for machine learning-based text categorization. *Information processing and management: Special issue for Bayesian networks and information retrieval*.

McCulloch, W. S. and Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Minsky, M. (1961). Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers*, 49:8–30.

Molina, L. C., Belanche, L., and Àngela Nebot (2002). Feature selection algorithms: A survey and experimental evaluation. In Society, I. C., editor, *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 306–313, Maebashi City, Japan.

Murphy, P. M. and Aha, D. W. (1992). *UCI Repository of Machine Learning Databases*. www.ics.uci.edu. Department of Information and Computer Science, University of California-Irvine.

Neapolitan, R. (2003). *Learning Bayesian Networks*. Prentice Hall.

Nilsson, D. (1998). An efficient algorithm for finding the m most probable configurations in bayesian networks. *Statistics and Computing*, 9:159–173.

Pazzani, M. J. (1997). Searching for dependencies in Bayesian classifiers. In *Artificial Intelligence and Statistics IV, Lecture Notes in Statistics*, New York. Springer-Verlang.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Pople, H. E. (1973). On the mechanization of abductive logic. In *Proceedings of the Third International Joint Conference on Artifical Intelligence*.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6:461–464.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, 34:871–882.

Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*. Lecture Notes in Statistics 81, Springer-Verlag.

Torkkola, K. (2002). On feature extraction by mutual information maximization. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 821–824.

Xing, E. P., Jordan, M. I., and Karp, R. M. (2001). Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 601–608, Williamstown, MA.

Zhang, N. L. and Poole, D. (1996). Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328.