Clustering en redes bayesianas. Implementación en R

Trabajo Fin de Máster

Autor:

Miguel Ángel Andrés Mañas

Tutor:

Rafael Rumí Rodríguez

Máster en Matemáticas

Septiembre, 2017

Universidad de Almería

Índice general

Abstract

The main subject of the present work is creating a non-supervised classification algorithm for bayesian networks using the free software R. This algorithm will perform a non-hierarchical clustering following the Naive-Bayes model. It will also introduce new ways of dividing groups and will use BIC as an optimality criterion.

First of all, the concept of classification will be introduced, paying special attention to non-supervised classification, also known as clustering. The concept of bayesian network will be presented too as a way of modelling reality from a probabilistic point of view.

This work is centered in mixing both concepts above, that is to say, creating groups in a situation modelled by a bayesian network, which can be a difficult problem in certain situations, such as having a big amount of data or a complex network structure. Therefore, some models which can simplify this problem has been created. In this work we study the Naive-Bayes model.

Once all concepts have been defined, the development of the function is explained, describing the code. It is divided into two iterative structures: the first one generates models with a set number of clusters and the second one increases this number.

Finally, once the function has been created, its performance and efficiency will be tested, not only amongst the different cluster splitting variations created, but also with the existing R functions *hclust* and *kmodes*.

Resumen

El tema principal de este trabajo es la creación de un algoritmo de clasificación no supervisada para redes bayesianas utilizando el software libre R. Con este algoritmo se llevará a cabo un clustering no jerárquico siguiendo el modelo Naive-Bayes, que además implementará nuevas formas de división de grupos y el uso del BIC como criterio de optimalidad.

Así pues, primero se introducen los conceptos de clasificación, prestando especial atención a la clasificación no supervisada o clustering, y de red bayesiana como forma de modelizar la realidad de forma probabilística.

El trabajo está centrado en la mezcla de los dos conceptos anteriores, es decir, en la creación de grupos dentro de una situación modelizada por medio de una red bayesiana, lo cual es un problema que puede ser difícil de abordar en casos de gran número de datos o de complejidad elevada de la estructura de la red. Por ello, existen algunos modelos de simplificación de este problema, estudiándose en este trabajo el modelo Naive-Bayes.

Una vez definidos todos los conceptos, se detalla la construcción de la función, exponiendo el funcionamiento del código, englobado en dos estructuras iterativas, una para generar los diferentes modelos para un determinado número de grupos y otra para aumentar dicho número.

Por último, cuando ya se ha creado la función, se comparan su rendimiento y eficiencia, tanto entre las propias variantes de la función como con los comandos *hclust* y *kmodes* implementados en R.

1 Objetivos

En el presente trabajo se pretende desarrollar, según el orden expuesto, los siguientes objetivos:

1. Diseñar un algoritmo, haciendo uso del software libre R con paquetes especializados en el tratamiento de redes bayesianas, que permita realizar la clasificación no supervisada de una serie de datos de carácter categórico suponiendo que los datos siguen el modelo Naive-Bayes. Además, se pretende crear las siguientes variantes de este algoritmo en cuanto se refiere a la forma de división de los grupos formados:
   - Comenzar el proceso para cada número de grupos desde el inicio, es decir, sin dividir ninguno.
   - Seleccionar el grupo a dividir aleatoriamente.
   - Dividir aquél con menor disimilaridad de Kullback-Leibler respecto a la distribución uniforme.
   - Dividir el que tenga la menor anchura media en su silueta, utilizando la distancia de Gower.
2. Comparar entre sí las variantes creadas, así como con las funciones *hclust* y *kmodes* que ya se encuentran implementadas en R, con objeto de determinar la capacidad de agrupación del nuevo método propuesto.

2 Introducción

El Análisis de Datos es la rama de las Matemáticas que se encarga de inspeccionar, limpiar, transformar y modelar una serie de datos con el objetivo de obtener información útil, sugerir conclusiones o ayudar en la toma de decisiones con respecto a los mismos.

En particular, dentro del Análisis de Datos se encuentra la Minería de Datos que puede definirse como el conjunto de técnicas de exploración y análisis de grandes cantidades de datos conducentes a encontrar patrones y reglas significativas en ellos.

Este término fue introducido en los años 90, si bien sus orígenes, es decir, los primeros métodos creados con el fin de observar patrones o reglas en los datos, se remontan a 1763 cuando fue publicado el artículo de Thomas Bayes [1] en el que se exponía el famoso Teorema de Bayes para probabilidades condicionadas, fundamental en el tratamiento de datos. Más tarde, en 1805, Adrien-Marie Legendre y Carl Friedrich Gauss aplicaron el análisis de regresión para la predicción de trayectorias de cometas y planetas. A lo largo del siglo XIX y parte del XX se desarrolló bastante el análisis de regresión, aunque fue a partir de 1936, con el inicio de la era computacional, cuando se hizo posible recoger, almacenar y tratar cantidades cada vez mayores de datos, desarrollándose así técnicas cada vez más complejas para su estudio, como por ejemplo el Análisis de Grupos, también llamado clasificación no supervisada, o las redes bayesianas, que son los temas principales de este trabajo.

Puede definirse de manera somera la clasificación como la tarea de agrupar un conjunto de objetos de tal manera que los miembros del mismo grupo sean lo más similares posible, en algún sentido u otro.

Los algoritmos de clasificación se dividen en dos grandes grupos:

- De clasificación supervisada. Cuentan con un conocimiento a priori, es decir, para la tarea de clasificar un objeto dentro de una clase se cuenta con modelos ya clasificados (objetos agrupados que tienen características comunes). En la primera fase del algoritmo se tienen un conjunto de entrenamiento o aprendizaje para el diseñar el modelo y otro de test o de validación para validarlo. La segunda fase es el proceso en sí de clasificar los individuos de los que se desconoce la categoría a la que pertenecen.

- De clasificación no supervisada o clustering. A partir de observaciones de un conjunto de variables se buscan grupos homogéneos sin que se conozcan a priori los grupos a los que pertenecen. Los individuos se agrupan de forma que en cada clase sean similares entre sí pero a la vez diferentes de los miembros de otros grupos. Dentro del clustering, los algoritmos pueden dividirse en dos tipos según su manera de proceder:

    - Jerárquicos. Siguen un orden y mantienen una estructura a la hora de construir los grupos. A su vez se dividen en aglomerativos (cada elemento forma

un grupo y se van uniendo) y divisivos (se comienza con el total de la muestra, que se va fracturando).

- No jerárquicos. Se elige el número de grupos y luego se van asociando individuos a cada grupo, pudiendo intercambiarse dichos individuos en cada paso, hasta que se alcance algún criterio de optimalidad.

Y según los resultados que producen pueden distinguirse:

- Difusos o probabilísticos. Cada individuo puede pertenecer a varias clases según una cierta distribución de probabilidad.
- Estrictos. Asignan cada elemento a un único grupo.

El término red bayesiana fue utilizado por primera vez en 1986 por Judea Pearl en su artículo [2].

Intuitivamente, una red bayesiana es una forma de representar y tratar un conjunto de datos teniendo en cuenta la relación que existe entre las diferentes variables de las que se obtiene información.

Como ventaja de las redes bayesianas destaca la simplicidad a la hora de exponer el modelo, ya que a través de un simple grafo se pueden resumir las dependencias entre las variables y así ofrecer una visión más clara y simplificada del problema. Además, mediante este sistema, a partir de los datos se puede obtener información tanto de todas las variables en su conjunto como de un segmento de ellas, dado que es posible calcular todas las distribuciones de probabilidad condicionadas asociadas.

Sin embargo, como inconveniente, la estructura de las redes bayesianas puede resultar bastante complicada a la hora de realizar ciertos tratamientos de los datos a medida que se aumenta el número de variables y de relaciones entre ellas. Por ello, existen una serie de modelos que simplifican dicha estructura y ofrecen un cálculo eficiente de las distribuciones de probabilidad, como se verá en secciones posteriores.

Por último, comentar que hoy en día las redes bayesianas son utilizadas en una gran cantidad de ámbitos científicos como puede observarse, por ejemplo, en [3].

3

Clustering y redes bayesianas

En este capítulo se ahondará en los conceptos generales de clustering y red bayesiana para posteriormente unirlos y estudiar en particular el problema del clustering en redes bayesianas. Además, se listarán los diferentes paquetes implementados y disponibles en R para estas tareas.

3.1 Clustering

En el capítulo anterior se introdujo la clasificación no supervisada o clustering. Ahora, se exponen con más detalle los algoritmos más populares para el tratamiento de este problema.

Dado que se trata de crear grupos homogéneos, se necesita medir de alguna manera la diferencia entre individuos. Para ello, se define el concepto de disimilaridad.

Definición 3.1. *Dado un conjunto X, una medida de disimiliaridad es una función $D : X \times X \to [0, +\infty]$ que verifica:*

- $D(x, y) \geq 0$, $\forall x, y \in X$.
- $D(x, x) = 0$, $\forall x \in X$.
- $D(x, y) = D(y, x)$, $\forall x, y \in X$.

Ejemplos de disimilaridades son: la distancia euclídea, la de Mahalanobis, la de Manhattan, la de Chebyshev o la de Gower. Éstas pueden consultarse en [4].

Dentro del clustering jerárquico, el algoritmo más utilizado es el siguiente, de tipo aglomerativo:

1. Cada tratamiento forma su propio grupo.
2. Se calcula la matriz de proximidades (la disparidad entre cada una de las observaciones) utilizando cualquier distancia o disimilaridad (que se denotará por d), como, por ejemplo, las listadas anteriormente.
3. Se determina la distancia entre los grupos a partir de la matriz de proximidades.

   Sean X un conjunto de datos con p variables, y A y B dos grupos con n_A y n_B elementos, respectivamente, los cuales se denotan por a_m y b_t, con $m = 1, \ldots, n_A$ y $t = 1, \ldots, n_B$. Además, cada una de las variables de cada elemento se denota como a_{mj} o b_{tj} para $j = 1, \ldots, p$.

   Una vez definida la notación, se presentan los criterios más utilizados a la hora de calcular la proximidad entre los grupos:

- Vecino cercano. La distancia entre dos grupos es la menor de todas las distancias entre los casos de un grupo y los casos del otro, esto es,

$$D_{AB} = \min_{m,t}\{d(a_m, b_t)\}$$

- Vecino lejano. Análogo al anterior, pero esta vez la distancia es la mayor de todas las distancias entre los casos de un grupo y los del otro.

$$D_{AB} = \max_{m,t}\{d(a_m, b_t)\}$$

- Promedio de grupo. Mide la distancia entre dos grupos como la media de todas las distancias entre los casos de un grupo y los del otro.

$$D_{AB} = \frac{1}{n_A n_B}\sum_{m=1}^{n_A}\sum_{t=1}^{n_B} d(a_m, b_t)$$

- Centroide. Se calcula el centroide para cada grupo (se denotará por $\overline{a}$, y cada una de sus componentes por $\overline{a}_j$), que es el vector formado por las medias de cada una de las variables y así, la distancia euclídea al cuadrado entre los centroides de los dos grupos es la que se toma como distancia entre ellos.

$$D_{AB} = \sum_{j=1}^{p}(\overline{a}_j - \overline{b}_j)^2$$

  La utilidad principal de este método es que se evita calcular todas las distancias entre todos los casos.

- Método de Ward. Mide el incremento de distancias euclídeas al cuadrado con respecto al centroide cuando se realiza la agrupación.

  Para el grupo A, las distancias respecto al centroide son

$$\phi_A = \sum_{m=1}^{n_A}\sum_{j=1}^{p}(a_{mj} - \overline{a}_j)^2$$

  y análogamente para B. Ahora, al juntar los dos grupos (se llamará AB), estas distancias quedan

$$\phi_{AB} = \sum_{m=1}^{n_A+n_B}\sum_{j=1}^{p}(ab_{mj} - \overline{ab}_j)^2$$

  y, por último, la proximidad entre A y B se mide como el incremento de estas distancias, es decir,

$$D_{AB} = \phi_{AB} - (\phi_A + \phi_B)$$

4. Mediante alguno de estos criterios, se determinan los dos grupos más próximos (menos distantes) y se unen creando un nuevo grupo.

5. Se calcula la nueva matriz de proximidades, esta vez teniendo en cuenta la existencia de grupos.

6. Se vuelve al paso 3.

Este algoritmo termina cuando sólo queda una clase, es decir, cuando todos los casos se han unido. De esta forma, los estados interesantes son los intermedios, de donde se puede tomar una agrupación con sentido.

La variante divisiva del clustering jerárquico se obtiene de forma análoga a la aglomerativa sin más que partir de un solo grupo con todos los casos en lugar de uno para cada observación y separar, en cada iteración, aquél con mayor distancia intergrupal calculada con alguno de los criterios del punto 3.

Para el clustering no jerárquico, el algoritmo más conocido y utilizado es el de las k-medias, que realiza los siguientes pasos:

1. Fijar un número k de grupos a construir.

2. Asignar k casos iniciales, es decir, crear k grupos de un elemento. Esto puede hacerse seleccionando los elementos al azar o manualmente.

3. Calcular la distancia euclídea de cada observación a los centroides de los k grupos. Puede también llevarse a cabo con cualquier otra disimilaridad, aunque la euclídea es la más utilizada.

4. Reasignar cada caso al grupo cuyo centroide ha resultado ser el más cercano.

5. Recalcular los centroides una vez construidos los grupos.

6. Volver al paso 3.

Destacar que este tipo de algoritmo necesita también un criterio de parada, ya que, en cada iteración, los datos pueden cambiarse de grupo y podrían darse casos en los que esto ocurriera indefinidamente. Algunos criterios de parada podrían ser: un número máximo de iteraciones, un número de iteraciones seguidas en las que las agrupaciones coinciden, una medida que establezca la bondad del modelo (por ejemplo la verosimilitud), o incluso una combinación de éstas.

Ahora bien, para variables exclusivamente categóricas existe una variación del algoritmo de las k-medias, llamada k-modes, que funciona como éste pero utilizando la moda de cada grupo como punto de referencia en lugar del centroide y la distancia 'simple matching', que cuenta el número de elementos en los que se diferencian dos observaciones dividido entre el número total de variables, para calcular la proximidad de cada elemento hasta las modas. De esta forma, los elementos con menor distancia a cada una de ellas se añadirán a los grupos correspondientes a dichas modas.

Por lo que respecta a R, cuenta con varios paquetes enfocados exclusivamente a la clasificación, tanto supervisada como no supervisada. Destacan los siguientes:

- *stats*. Este paquete contiene principalmente funciones para cálculos estadísticos y números aleatorios. Además, en temas de clustering, contiene dos de las funciones más utilizadas:

  - *hclust*. Esta función lleva a cabo el cluster jerárquico descrito anteriormente. Para ello, se le han de pasar como entrada: la matrix de proximidad de los datos, calculados, por ejemplo, con la función *dist* dentro de este mismo paquete o con alguna otra de otro paquete, y el método de cálculo de la proximidad entre grupos, a seleccionar entre *single* (vecino cercano), *complete* (vecino lejano), *average* (promedio de grupo), *centroid* o *ward.D2*, además de otros menos comunes como *mcquitty* o *median* (utiliza la mediana del grupo como punto de referencia en lugar del centroide).

  - *kmeans*. Con *kmeans* se realiza el clustering no jerárquico mediante el método de las k-medias. La entrada de esta función consta del conjunto de datos a agrupar y el número de clusters que se crearán.

    Sin embargo, ésta función sólo esta definida para datos de carácter numérico, por lo que no será posible realizar comparaciones entre esta función y la que se creará en este trabajo. Por ello, se debe recurrir al paquete que comenta a continuación.

- *klaR*. Es un paquete destinado exclusivamente a clasificación, donde se destaca la función *kmodes*, que implementa el método de las k-modas. Como ya se comentó, esta función es similar a *kmeans* pero para variables categóricas, lo que supone que puede ser y será objeto de comparación con la función creada en este trabajo.

- *cluster*. Este paquete contiene otras formas de clustering, como PAM (*Partitioning Around Medoids*), que es una versión más robusta del k-medias, o CLARA (*Clustering LARge Applications*) para conjuntos de datos más grandes. Además, contiene una serie de funciones bastante útiles a la hora de trabajar el clustering.

  Para este trabajo, se utilizarán las funciones *daisy* y *silhouette* para calcular las disimilaridades entre elementos y la bondad del ajuste realizado, respectivamente.

3.2 Redes bayesianas

En el capítulo anterior se definió de manera simple el concepto de red bayesiana. En esta sección se definirá de manera formal y se comentarán las posibilidades que ofrece R para su manejo.

Así pues, una red bayesiana se define formalmente como:

Definición 3.2. *Una red bayesiana [5] es un modelo estadístico multivariante para un conjunto de variables* $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ *formado por dos componentes:*

1. *Un grafo dirigido acíclico $G = (\boldsymbol{X}, A)$ donde los vértices son el conjunto de variables del modelo y las aristas indican la existencia de dependencia estadística entre ellas.*

2. *Un conjunto de n distribuciones de probabilidad condicionada, una para cada variable, $P = \{P(X_1|\Pi_1), \ldots, P(X_n|\Pi_n)\}$, donde Π_i representa el conjunto de padres del nodo i en G, es decir, el conjunto de nodos desde los cuales sale una arista hacia X_i.*

   Así, la distribución de probabilidad global puede escribirse como:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(X_i = x_i|\Pi_i), \quad \forall (x_1, \ldots, x_n) \in \Omega_{X_1} \times \cdots \times \Omega_{X_n} \tag{3.1}$$

   donde Ω_{X_i} representa el conjunto de todos los posibles valores de X_i.

Para ilustrar el concepto, se utilizará un ejemplo sencillo obtenido de [5].

Ejemplo 3.1. *El señor Holmes está trabajando en su oficina cuando de repente recibe una llamada de su vecino, el señor Watson, avisándole de que la alarma antirobo de su casa está sonando. Convencido de que hay un ladrón en su casa, Holmes coge el coche y en el camino escucha en la radio que ha habido un pequeño terremoto en la zona. Como Holmes sabe que las alarmas pueden activarse debido a terremotos, vuelve al trabajo.*

Analizando este enunciado, puede construirse el grafo de la Figura 3.1, donde cada variable es de tipo binario (*Sí* o *No*, ó 1 ó 0).

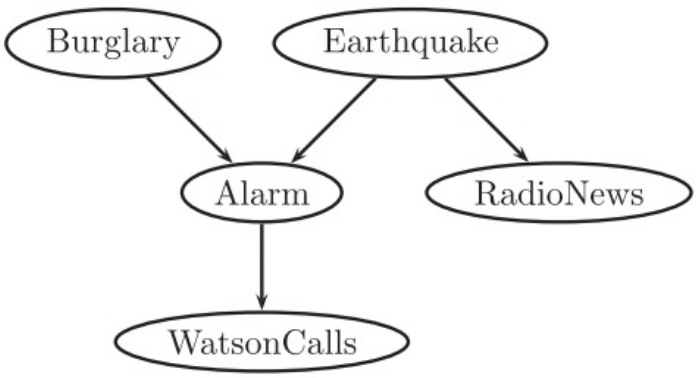

Figura 3.1: Grafo asociado al ejemplo del señor Holmes

En este grafo se observa cómo la llamada de Watson se encuentra condicionada únicamente a su visión de la alarma, pero ésta a su vez puede depender tanto de que haya habido un robo como de un terremoto. Además, otra variable importante es el hecho de que en la radio hayan informado de un terremoto, que obviamente sólo depende de la existencia o no de dicho terremoto.

Ahora, si se denota cada variable con su primera letra en el grafo, el componente probabilístico que dota a este grafo con estructura de red bayesiana sería el conjunto de distribuciones de probabilidad

$$\{P(B), P(E), P(A|B, E), P(R|E), P(W|A)\}$$

Y la distribución global quedaría:

$$P(B,E,A,R,W) = P(B)P(E)P(A|B,E)P(R|E)P(W|A)$$

donde, como ya se comentó, cada variable puede tomar el valor 0 ó 1.

Dos procedimientos necesarios a la hora de tratar con redes bayesianas son:

- Aprendizaje de la red. A su vez existen dos tipos:
  - Aprendizaje estructural. A partir del conjunto de datos se pretenden descubrir las dependencias existentes entre las variables para así poder construir el grafo representativo de la red.
  - Aprendizaje de parámetros. Una vez se tiene la estructura de la red, el objetivo es calcular las distribuciones de probabilidad asociadas a la misma. Existen dos formas de llevarlo a cabo: la estimación por máxima verosimilitud y la estimación bayesiana, que se explicarán un poco más adelante.
- Inferencia. Cuando ya se tiene la red bayesiana completamente definida, se busca obtener información de nuevos casos observados en la realidad modelada por esta red a partir de dichas evidencias observadas. Para ello, se pretende responde a preguntas de tipo: ¿cuál es la probabilidad de que X tome el valor x sabiendo que se ha observado que Y toma el valor y?

El principal paquete de R para creación y manipulación de redes bayesianas que se utilizará en este trabajo es *bnlearn* [6]. Se aplicarán las siguientes funciones presentes en este paquete:

- *empty.graph*. Genera un grafo a partir del conjunto de nodos de entrada.
- *set.arc*. Añade un arco dirigido a un grafo. Para ello, se deben especificar el grafo, el nodo de origen y el nodo destino.
- *bn.fit*. Estima los parámetros del modelo, es decir, calcula las distribuciones de probabilidad asociadas a éste, sin más que pasarle la estructura de la red y el conjunto de datos en que se basa.

  Es posible llevar a cabo este ajuste a través de dos métodos: *mle* y *bayes*. Con el primero, se calculan las distribuciones de probabilidad de manera puramente frecuentista, es decir, contando el número de casos en que se da un determinado suceso y dividiéndolo entre el número de observaciones total. Con el segundo, en cambio, se utiliza el enfoque bayesiano, que da una cierta probabilidad a priori a cada combinación de las variables, evitando así que existan configuraciones con probabilidad 0 de aparecer. Para esto, se añade la llamada *iss* (imaginary sample size) que, grosso modo, es un número determinado de casos ficticios añadidos a los datos precisamente para evitar la desaparición de un caso. Más concretamente, la probabilidad de que una variable X tome un cierto valor x se expresa como

$$P(X = x) = \frac{iss}{N + iss}\pi_x + \frac{N}{N + iss}p_x$$

donde N es el número de observaciones en los datos, $\pi_x = 1/n_X$ es la distribución *a priori* (que se considera uniforme), siendo n_X el número de valores que puede tomar la variable X, y $p_X = \frac{n}{N}$ es la probabilidad obtenida de los datos, con n el número de casos donde la variable X toma el valor x. De esta manera, si $n = 0$, la probabilidad frecuentista es 0, pero queda el sumando de la probabilidad *a priori* que da una cierta posibilidad (pequeña) de que se tome el valor x.

- *score*. Permite calcular medidas de bondad de ajuste de una red bayesiana partiendo del grafo que la define y de los datos. Estas medidas pueden la log-verosimilitud, el AIC (Akaike Information Criterion) o el BIC [7], siendo este último el que se utilizará en este trabajo y que se define como:

  Definición 3.3. *Dada una distribución de probabilidad P (o una densidad para el caso de funciones continuas), se define el BIC (Bayesian Information Criterion) como*

  $$BIC = \sum_{i=1}^{n} \log P(X_i|\Pi_{X_i}) - \frac{dim(P)}{2}\log(n)$$

  donde dim(P) es el número de parámetros libres.

  De esta manera, el BIC mide cómo de bueno es un modelo basándose en la verosimilitud del mismo y penalizando aquéllos más complejos (con más parámetros a estimar).

3.3 Clustering en redes bayesianas

Una vez definidos los conceptos de red bayesiana y clasificación no supervisada, se estudiará cómo unirlos, es decir, cómo abordar el problema de la agrupación de los datos sobre una estructura de red bayesiana.

Se tratará primero el caso general de la clasificación en redes bayesianas y posteriormente se expondrán algunos de los modelos utilizados para abordarlo. Además, se comentarán particularmente las posibilidades de tratamiento que ofrece el software R en este ámbito y las propuestas novedosas que se desarrollan en este trabajo.

El problema de clasificación (tanto supervisada como no supervisada) en redes bayesianas se basa en agrupar las observaciones de una serie de datos en distintas clases (o clusteres) conociendo las dependencias entre las variables del modelo, dadas por la red. Para la clasificación supervisada, ya se dispone en los datos de una variable de clase, mientras que para la no supervisada, es necesario añadir manualmente una variable especial, que en adelante denotaremos por C, que asigne a cada elemento un valor (c_j) representativo del grupo al que pertenece. Así, los individuos con el mismo valor pertenecen a la misma clase, y se busca que, como en todo problema de clasificación, los objetos dentro del mismo grupo sean lo más parecidos posible entre sí y lo más diferentes posible del resto de los clusters. En otras palabras, se desea asignar cada elemento al grupo que mejor lo represente.

El clustering probabilístico se puede definir como una mixtura de modelos, donde los estados de la variable C, inicialmente desconocidos, se corresponden con los componentes de la mixtura [8]. La forma más usual de estimar este modelo es mediante el algoritmo EM [9], aunque en este trabajo se utilizará una versión estocástica del mismo (ver Sección 4).

En términos más matemáticos, si se tiene un vector de variables (o vector de características) $\mathbf{X} = \{X_1, \ldots, X_n\}$, el problema consiste en encontrar la clase c^* tal que

$$P(c^*|x_1, \ldots, x_n) = \max_j P(c_j|x_1, \ldots, x_n)$$

para todo $(x_1, \ldots, x_n) \in \Omega_{X_1} \times \cdots \times \Omega_{X_n}$.

Aplicando el Teorema de Bayes, cada una de las probabilidades se expresa como:

$$P(c_j|x_1, \ldots, x_n) = \frac{P(c_j)P(x_1, \ldots, x_n|c_j)}{P(x_1, \ldots, x_n)} \tag{3.2}$$

donde $P(c_j)$ es la probabilidad a priori de la clase.

Además, todas estas probabilidades han de verse en la forma de la Expresión 3.1, lo cual refleja la complejidad del problema si la estructura es excesivamente enrevesada (muchas dependencias entre las variables). Es por esto que, para abordar el problema, se han creado una serie de modelos que fijan una determinada estructura para la red bayesiana, como, por ejemplo, el modelo Naive-Bayes [10], que se estudia a continuación.

El modelo Naive-Bayes es aquél en torno al cual gira el cuerpo de este trabajo. Debe su nombre a la hipótesis ingenua (en inglés *naive*) sobre la que se desarrolla, y es que supone que todas las variables son condicionalmente independientes dada la variable clasificadora.

La estructura propuesta por este modelo es, por tanto, la de la Figura 3.2.

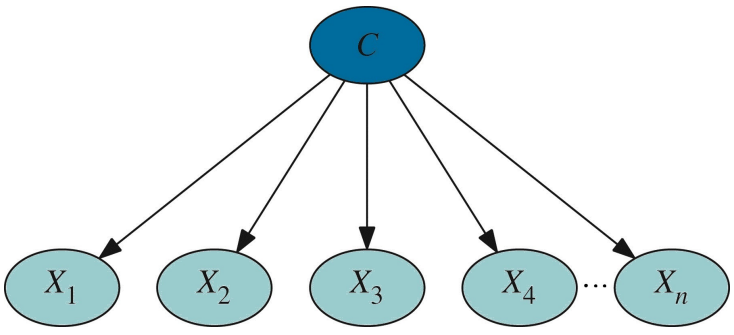

Figura 3.2: Grafo asociado al modelo Naive-Bayes

Esta disminución de los condicionamientos provoca que el cálculo de las probabilidades se simplifique notablemente. En efecto, partiendo de la Expresión 3.2 y aplicando en repetidas ocasiones la definición de probabilidad condicionada en el numerador:

$$P(c_j|x_1,\ldots,x_n) = \frac{P(c_j)P(x_1|c_j)P(x_2,\ldots,x_n|c_j,x_1)}{P(x_1,\ldots,x_n)} =$$

$$= \frac{P(c_j)P(x_1|c_j)P(x_2|c_j,x_1)P(x_3,\ldots,x_n|c_j,x_1,x_2)}{P(x_1,\ldots,x_n)} =$$

$$= \frac{P(c_j)P(x_1|c_j)P(x_2|c_j,x_1)\cdot\ldots\cdot P(x_n|c_j,x_1,x_2,\ldots,x_{n-1})}{P(x_1,\ldots,x_n)}$$

y en este punto entra en juego la independencia condicional de las variables, pues bajo esta hipótesis, $P(x_i|c_j,x_k) = P(x_i|c_j)$ para todo $i \neq k$ y entonces la Expresión 3.2 queda como

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

Pero, como se puede observar, el denominador sólo depende de los valores de X_i, es decir, de los datos de partida, no de la clasificación. Entonces, este valor es constante para cada combinación de las variables y, por tanto, puede simplificarse aún más la igualdad anterior, siendo ésta la que se utilizará en la práctica para programar el modelo:

$$P(c_j|x_1,\ldots,x_n) \propto P(c_j)\prod_{i=1}^{n} P(x_i|c_j) \tag{3.3}$$

Esto significa que las probabilidades de que un individuo se encuentre en un determinado grupo son proporcionales a estos productos, por lo que, al omitirse el factor que dependía sólo de los datos, las probabilidades deben ser normalizadas una vez se calculen, es decir, puestas en proporción respecto a 1.

A pesar de su simpleza estructural, es un modelo que trabaja bastante bien en aquellos casos donde las variables no están muy correlacionadas entre sí. Por el contrario, si la correlación entre las variables es fuerte la suposición de independencia supone la pérdida de información, aunque en estos casos de dependencia muy fuerte podría plantearse realizar un estudio de reducción de la dimensión para así eliminar variables que aporten información que ya puede deducirse a partir de las otras.

Las novedades que se pretenden introducir en este trabajo son, por una parte, la creación de cuatro formas de decisión a la hora de pasar de un número de clusters al siguiente (se detallará cada una en la sección siguiente), y por otra la utilización del BIC como método de comparación entre modelos. Destacar que, aunque existen funciones similares en R, ninguna de ellas tiene un método de selección del mejor modelo como se propone en este trabajo, es decir, se limitan a crear un modelo con un número de clusters establecido por el usuario.

Como se comentó anteriormente, el BIC es la función de log-verosimilitud penalizada por el número de parámetros del modelo.

Al incrementar el número de clusters, la función de verosimilitud mejora, pero también se incrementa la complejidad del modelo, lo cual puede llevar a casos de sobreajuste.

Por ello, se ha elegido el BIC como criterio de optimalidad, ya que permite conjugar precisión y simplicidad al regular el crecimiento de la log-verosimilitud a través del número de parámetros.

4 Desarrollo del algoritmo

En esta sección se expondrá de forma detallada la construcción del algoritmo en torno al cual gira este trabajo, que está basado en el trabajo [7], donde se utiliza el procedimiento *Data Augmentation* [11]. La principal diferencia con el algoritmo EM radica en que, en lugar de calcular el valor esperado para la variable oculta, el algoritmo *Data Augmentation* simula un valor para dicha variable a partir de su distribución a posteriori obtenida mediante un proceso de inferencia.

El código de la función creada puede consultarse en el Apéndice A y descargarse desde el sitio web `https://drive.google.com/file/d/0By8w3jW52EqeQkFVWl9oNWR0eG8/view?usp=sharing`.

Se trata de un algoritmo de clustering no jerárquico difuso para variables categóricas sobre un modelo Naive-Bayes en el cual se han implementado, además, varias maneras de aumentar el número de grupos, que se detallarán más adelante.

Grosso modo, el funcionamiento del algoritmo es el siguiente: se genera un modelo aleatorio para dos grupos, es decir, se da el valor 0 ó 1 a la variable de clase para cada individuo aleatoriamente; se calculan nuevos modelos con dos grupos a partir de éste teniendo en cuenta que se sigue el modelo Naive-Bayes; una vez alcanzado el criterio de optimalidad (mayor BIC), se elige una clase para ser dividida, se aumenta en 1 el número de grupos mediante el fraccionamiento de la clase elegida y se vuelve a generar un modelo inicial para dicho número de grupos, en el cual el nuevo cluster se compone de valores seleccionados aleatoriamente a partir de la clase dividida. Entonces, se vuelve al inicio.

Comenzando con la explicación detallada, la entrada de la función requiere tres argumentos: el conjunto de datos a clasificar; un número que indique el mínimo de iteraciones a realizar en cuanto al aumento del número de grupos, es decir, el mínimo de veces que se aumentará en 1 el número de clases; y el método que se utilizará para la elección del grupo a dividir, exponiéndose cada uno de ellos más adelante.

Al tratar con datos de carácter exclusivamente categórico, el primer paso que debe realizar el algoritmo es pasar estos datos a factores y prepararlos para que las funciones que se usen más adelante puedan leerlos correctamente.

Una vez hecho esto, se añade la variable de clase como una nueva columna dentro de los datos y se genera el modelo inicial asignando el valor 0 ó 1 a cada una de las observaciones. En este caso, se ha decidido que las probabilidades iniciales de cada valor sean aproximadamente 0.5.

El siguiente paso es crear el grafo representativo del modelo Naive-Bayes (similar al de la Figura 3.2) donde los nodos son las variables presentes en los datos junto con la variable clase y los arcos indican la dependencia estadística de éstas respecto de la variable clase. Para ello, se han utilizado los comandos *empty.graph* y *set.arc* del paquete *bnlearn*.

Ahora, se inicializan una serie de variables que estarán presentes en el cuerpo del algoritmo, entre las que destacan aquélla que contendrá los posibles valores que puede tomar la variable clase y las que se utilizarán para comparar los mejores modelos que se creen para cada número de grupos (a una de ellas se le da el valor inicial $-\infty$ para asegurar que en la primera iteración se entre en el bucle).

El cuerpo del algoritmo lo forman dos estructuras iterativas de tipo *while*, una dentro de la otra. A continuación se detalla el funcionamiento de cada uno de los bucles.

4.1 Bucle interior

El objetivo de esta estructura *while* es generar y comparar modelos para un número determinado de valores de la variable de clase siguendo el esquema Naive-Bayes (Expresión 3.3).

En primer lugar, se definen algunas variables de almacenamiento de resultados y conteo, así como las que almacenarán los BIC de los dos últimos modelos creados con el fin de poder compararlos.

La condición de entrada en este bucle establece que se realicen obligatoriamente 20 iteraciones, es decir, que se generen 20 nuevos modelos; a partir de entonces, sólo se continuará en el bucle si el BIC del modelo generado en una iteración es mayor, es decir, mejora, al del modelo de la iteración anterior. Además, se establece un límite superior de 150 pasadas para evitar un tiempo de procesamiento excesivo e incluso bucles infinitos.

Una vez dentro del bucle, se guarda el modelo sobre el que se trabajará (el grupo al que pertenece cada individuo) en la variable *lista.modelos*, que más tarde permitirá seleccionar el mejor de ellos.

Entonces, se procede a la creación de un nuevo modelo, para lo cual es necesario el cálculo de las distribuciones de probabilidad condicionadas $P(X_j = X_{jr}|C = c_i)$ para todos i ,j y r, así como de $P(C = c_i)$ para cada i. Todo esto se realiza con el comando de R *bn.fit* del paquete *bnlearn*, especificando que se calculen mediante el enfoque bayesiano para que, como ya se justificó en el capítulo anterior, no existan tratamientos con probabilidad 0. Este hecho provoca entonces que ningún caso tenga probabilidad nula de pertenecer a un grupo, es decir, que pueda ser asignados a cualquier clase aunque sea con una probabilidad pequeña. De esta manera se mantiene el carácter difuso del método. Destacar en este punto que, a nivel de código se han utilizado siempre que ha sido posible las funciones de tipo *apply*, que son más eficientes que un *for*.

Una vez obtenidas las probabilidades condicionadas de las variables dado el grupo, así como la probabilidad marginal de la variable clase, basta aplicar la Expresión 3.3 para calcular las probabilidades de que un cierto individuo pertenezca a un grupo o a otro. Para ello, se crea la matriz *px.c*, que se va renovando para cada valor de C y cuyas filas son las probabilidades de que cada variable X_j tome el valor x_{jr} para el valor del

grupo c_i. Así pues, en la variable *pc.x* se recoge el miembro derecho de la mencionada ecuación. Ahora bien, se vio que las probabilidades de pertenecer a un grupo u otro son proporcionales a los productos contenidos en *pc.x*, luego éstos deben ser normalizados, es decir, puestos en proporción respecto a 1.

Ahora que ya se tienen las probabilidades de que cada individuo pertenezca a un grupo u otro, se ha de generar el nuevo modelo de acuerdo a ellas. Para ello, se utilizará el paquete *stats*, que contiene funciones sobre cálculos estadísticos y distribuciones, en concreto la función *rmultinom*, con la cual se generarán tantos vectores aleatorios como individuos haya en los datos. Éstos son vectores nulos salvo en una posición donde se genera un 1 aleatoriamente según las probabilidades dadas por cada una de las filas de *pc.x*. Por último, se da el valor correspondiente a la variable de clase en función de en qué posición se hayan generado los unos. Nótese que en este punto es donde verdaderamente se aplica el carácter difuso del algoritmo, pues cada individuo puede ser asignado a un grupo u otro dependiendo de la probabilidad *pc.x*.

En este punto, se guardan el BIC del modelo anterior y el del modelo recién obtenido, calculándolo con la función *score* del paquete *bnlearn*.

Finalmente, se guardan todos los BICs en una variable destinada a ello, que posteriormente permitirá seleccionar el mejor modelo con cada número de clusters, y se aumenta en 1 el valor de la variable de conteo, volviéndose así al inicio del bucle donde se compararán los valores de los últimos BICs calculados.

4.2 Bucle exterior

Otra novedad implementada en este algoritmo es la posibilidad de seleccionar uno de los 4 métodos para dividir un grupo a la hora de crear uno nuevo. La función de esta estructura iterativa es, pues, aumentar el número de clusters, dividir uno a través de dichos métodos y generar un modelo inicial aleatorio que sirva de punto de partida en el bucle interior.

Se comienza de nuevo con la inicialización de variables, donde destacan la variable de conteo de este bucle, y las que almacenarán los BICs de los mejores modelos para un determinado número de grupos y para el número anterior con el fin de ver cuál de ellos es mejor.

La condición de entrada en este nuevo *while* es parecida a la del bucle anterior. Sin embargo, ahora se comparan únicamente los dos mejores modelos obtenidos de operar con las dos últimas longitudes del vector que contiene el número de clusters, es decir, con un cierto número de grupos y su anterior. Además, en este caso el número mínimo de iteraciones viene determinado por el usuario a través de la variable *n*.

Una vez dentro, se ejecuta la estructura de la sección anterior y, después de esto, de entre todos los modelos creados en ella, se elige aquél con el BIC mayor, es decir, el mejor según este criterio.

Una vez se tiene el mejor modelo para un cierto número de grupos, es hora de añadir un cluster más, para lo cual lo primero es seleccionar qué grupo se dividirá. Se han implementado para ello, como se comentó, cuatro métodos diferentes:

1. *Random*. Simplemente se selecciona al azar el cluster a dividir, es decir, en la variable *dividir* se guarda un número aleatorio elegido del vector que contiene el número de clusters.

   Como ventaja puede destacarse la simpleza del método, pero como inconveniente cabe mencionar que es un método que depende exclusivamente del azar y por tanto puede dividirse cualquier grupo, sea para mejorar o empeorar el modelo.

2. *KL*. Calcula la disimilaridad de Kullback-Leibler con respecto a la distribución uniforme dentro de cada grupo.

   Definición 4.1 (Divergencia de Kullback-Leibler para variables discretas)**.** *Sean P y Q dos distribuciones de probabilidad discretas. Se define la divergencia de Kullback-Leibler entre P y Q como:*

   $$d_{KL}(P,Q) = \sum_i P(i) \ln\left(\frac{P(i)}{Q(i)}\right)$$

   teniendo en cuenta que $0 * ln\left(\frac{0}{0}\right) = 0 * ln\left(\frac{0}{Q(i)}\right) = 0$ *y* $P(i) * ln\left(\frac{P(i)}{0}\right) = +\infty$.

   Esta medida no es simétrica, es decir, $d_{KL}(P,Q) \neq d_{KL}(Q,P)$, como puede comprobarse con este sencillo contraejemplo: sean $P = (0.5, 0.5)$ y $Q = (0.8, 0.2)$ dos distribuciones de probabilidad discretas. Entonces, $d_{KL}(P,Q) = 0.223144 \neq 0.192745 = d_{KL}(Q,P)$.

   Así pues, como lo que se desea considerar es una disimilaridad, se calcula la media de ambos valores, que ya sí es simétrica. En definitiva,

   Definición 4.2 (Disimilaridad de Kullback-Leibler)**.** *Dadas P y Q dos distribuciones de probabilidad discretas, se define la disimilaridad de Kullback-Leibler como*

   $$D_{KL}(P,Q) = \frac{d_{KL}(P,Q) + d_{KL}(Q,P)}{2}$$

   Proposición 4.1. *La función recién definida es una disimilaridad.*

   Demostración. Hay que ver que se verifican las tres propiedades de la Definición 3.1:

   a) $D_{KL}(P,Q) \geq 0$. Basta ver que $d_{KL}(P,Q) \geq 0$ para cualesquiera P y Q. Para ello, se hará uso del caso discreto de la desigualdad de Jensen, que establece que

si ϕ es una función convexa real definida en un conjunto X y a_i son pesos positivos, entonces

$$\varphi\left(\frac{\sum_{i=1}^{n} a_i x_i}{\sum_{i=1}^{n} a_i}\right) \leq \frac{\sum_{i=1}^{n} a_i \varphi(x_i)}{\sum_{i=1}^{n} a_i}$$

con $x_i \in X$ para todo i.

Como la función $-ln(x)$ es convexa para $x > 0$ (pues su derivada $-\frac{1}{x}$ es creciente para todo $x > 0$), se puede aplicar la desigualdad de Jensen con pesos $P(i)$. Así, como

$$d_{KL}(P,Q) = \sum_{i}^{n} P(i) \ln\left(\frac{P(i)}{Q(i)}\right) = -\sum_{i}^{n} P(i) \ln\left(\frac{Q(i)}{P(i)}\right)$$

y teniendo en cuenta que $\sum_{i=1}^{n} P(i) = \sum_{i=1}^{n} Q(i) = 1$,

$$-\sum_{i}^{n} P(i) \ln\left(\frac{Q(i)}{P(i)}\right) \geq -\ln\left(\sum_{i}^{n} P(i) \frac{Q(i)}{P(i)}\right) = -\ln\left(\sum_{i}^{n} Q(i)\right) = 0$$

b) $D_{KL}(P,P) = d_{KL}(P,P) = \sum_i P(i) \ln\left(\frac{P(i)}{P(i)}\right) = \sum_i P(i) \ln(1) = 0$

c) $D_{KL}(P,Q) = D_{KL}(Q,P)$. Es claro sin más que observar la definición de D_{KL}.

□

Esta disimilaridad mide la diferencia (o entropía) que existe entre las distribuciones P y Q, por lo que la idea plasmada en el algoritmo consiste en dar un valor a cada grupo que represente cuán representativas son o dejan de ser las variables dentro del grupo. Este valor es la suma de las disimilaridades de Kullback-Leibler entre cada variable dentro un grupo y la distribución uniforme. En efecto, si un valor de una variable toma una probabilidad alta dentro del grupo es señal de que dicho valor define bien el cluster al que pertenece, y por lo tanto aporta información sobre él. Al contrario, si la distribución es muy similar a la uniforme, ningún valor destaca sobre otro y no se puede sacar nada en claro del grupo en cuestión.

De esta manera, cuanto mayor sea la suma de estas disimilaridades, mejor explicado queda el grupo por una o varias variables, por lo que lo deseado es dividir aquél del que menos información se tenga, es decir, aquél con la menor disimilaridad de Kullback-Leibler.

A nivel de código, para computar las disimilaridades se utilizará la función *kl.dist* implementada en el paquete *seewave*, que es un paquete destinado al análisis, manipulación, edición y sintetización de ondas. En primer lugar, en la variable *grupo* se guardan todos los individuos que forman un cluster; a continuación, se toman cada una de las variables presentes en los datos para calcular su distribución de probabilidad y almacenarla en el vector *probs*. Por último, se compara

probs con la distribución de probabilidad de la uniforme (en la que todos los valores de una variable tienen la misma probabiidad) mediante la función *kl.dist*, para finalmente calcular la suma de las disimilaridades y dar a *dividir* el valor del cluster de mínima disimilaridad.

Como ventaja destaca que es un método de selección no arbitrario o aleatorio como lo era el anterior, con el hándicap de que debe realizarse el cálculo de bastantes disimilaridades, lo cual puede afectar al tiempo de procesamiento, que puede llegar a extenderse demasiado.

3. *Gower*. Se calcula la silueta media de cada cluster como medida de bondad de ajuste.

La silueta es un método de validación de la consistencia de los grupos. Fue descrito por Peter J. Rousseeuw en 1986 [12]. Mide cómo de parecido es un individuo a los de su propio cluster y en comparación con los de los demás. Valores altos indican que el individuo está bien clasificado en su grupo y mal clasificado en los demás.

Así pues, el valor de mayor interés es la silueta media de cada grupo, definida como:

Definición 4.3. *Partiendo de una disimilaridad, sean i un individuo, $a(i)$ la disimilaridad media de i con respecto a los otros elementos de su grupo y $b(i)$ el mínimo de las disimilaridades medias de i hasta los elementos de cada uno de los otros grupos. La silueta media $s(i)$ viene dada por:*

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Esta cantidad, como puede verse claramente, toma valores entre -1 y 1. Valores más cercanos a 1 representan que los individuos de ese grupo están bien clasificados, puesto que son similares entre sí y diferentes del resto de los grupos; lo contrario para valores cercanos a -1.

La silueta puede ser calculada con cualquier disimilaridad, eligiéndose para este trabajo la distancia de Gower por ser una de las pocas encontradas en R que trabaja exclusivamente con variables categóricas.

Definición 4.4 (Distancia de Gower)**.** *Sean x_i y x_j dos observaciones con n variables cada una (se denotarán como x_{ik}, x_{jk}). Se define la distancia de Gower como:*

$$d(x_i, x_j) = \frac{\sum_{k=1}^{n} c_k^{(ij)}}{n}$$

donde $c_k^{(ij)}$ es la contribución de la k-ésima variable a la distancia total. Para el caso de variables categóricas, $c_k^{(ij)}$ vale 1 si los valores de las variables x_{ik} y x_{jk} coinciden, y 0 en otro caso.

De esta manera, al tratar con variables categóricas en este trabajo, la distancia de Gower entre dos individuos coincide con la distancia 'simple-matching'.

La idea es, entonces, calcular la silueta media de cada uno de los grupos utilizando la distancia de Gower y seleccionar como grupo a dividir aquél con el menor valor, pues ese es el cluster cuyos elementos se encuentran peor clasificados.

Para llevar a cabo esto en R se utilizarán las funciones *daisy*, para calcular las distancias de Gower entre todos los individuos, y *silhouette*, para la silueta media de cada grupo, pertenecientes al paquete *cluster*.

Además, pueden obtenerse en formato pdf las representaciones gráficas de las siluetas medias en cada iteración del bucle exterior. Un ejemplo es la Figura 4.1, donde puede verse más intuitivamente cuál será el cluster que se dividirá; en este caso será el 1 (denotado por 0 en el código) por ser el de menor silueta media.

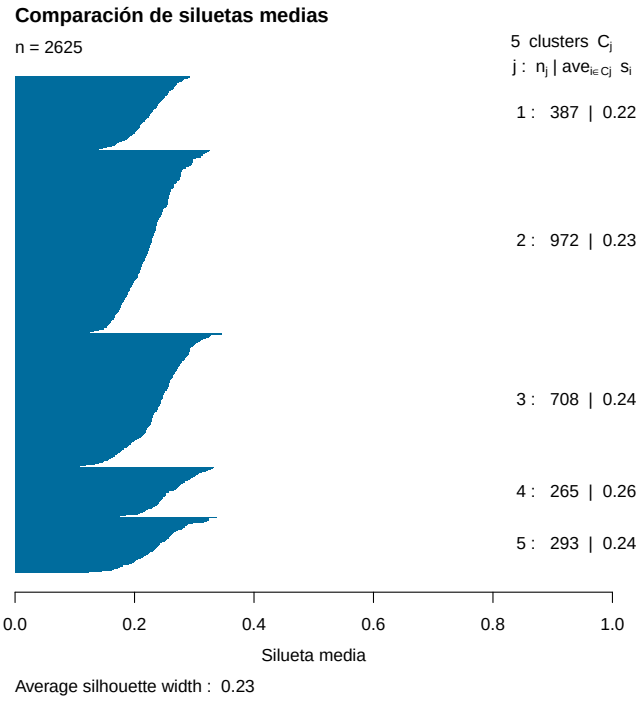

Figura 4.1: Ejemplo de gráfico de silueta

De nuevo, el inconveniente de este método es que puede traer consigo un tiempo de compilación algo grande, pero tiene la ventaja de que se está seleccionando para dividir el grupo peor clasificado y por tanto se aspira a que el modelo mejore en cada iteración.

4. *NoDiv*. No se divide ningún cluster, por lo que la variable *dividir* no toma ningún valor.

Este método es muy rápido, de hecho no se hace nada en este punto, aunque más adelante se verá cómo generar los modelos a partir de esto.

5. Otros. Se ha añadido una excepción para que, en el caso en que la variable *metodo* no tome ninguno de los cuatro valores anteriores, se avise al usuario que no es correcto con el mensaje 'Por favor inserte un método válido'.

Una vez se ha seleccionado el cluster a dividir, se crea un grupo nuevo, simplemente añadiendo un valor más a la variable que los cuenta.

Ahora, se genera un nuevo modelo inicial aleatorio con un grupo más que los anteriores, que servirá como comienzo del bucle interior en la siguiente iteración.

Si el método elegido ha sido *NoDiv*, simplemente se genera una muestra de los números presentes en la variable encargada de contar la cantidad de clases, es decir, se crea un modelo desde cero, sin tener en cuenta el proceso previo, siguiendo un proceso similar al realizado para 2 grupos.

En caso contrario, se selecciona el cluster a dividir y se sustituye aleatoriamente por, o bien él mismo, o bien el nuevo grupo creado, siendo el resultado una división de la clase seleccionada por uno de los métodos anteriores.

Ya que se ha generado el nuevo modelo, para poder compararlo con los anteriores y determinar el mejor de ellos, se guardan el valor del BIC del mejor modelo generado con un grupo menos y el del mejor modelo con el número de grupos con que se acaba de operar. Al inicio de la siguiente iteración se testeará si el segundo mejora al primero, y en caso positivo (o en caso de que no se haya alcanzado el número mínimo de iteraciones), se volverá a realizar todo el proceso descrito.

Finalmente, se aumenta en 1 la variable de conteo, lo que da fin al bucle exterior, volviéndose así al inicio de éste.

Por último, y fuera de ambas estructuras iterativas, se selecciona el modelo con el mejor BIC de todos y se crea la salida de la función *NaiveBayes*, similar a las de *hclust* o *kmodes*, que es una lista compuesta por:

- El recuento de los elementos de cada grupo del mejor modelo.
- Un vector, de longitud igual al número de individuos, con el grupo al que pertenece cada observación.
- Un vector que contiene el mejor BIC obtenido con cada número de grupos.

Para ilustrar mejor la salida del algoritmo, se expondrá un ejemplo con un conjunto de datos de prueba de 5000 observaciones con 6 variables. En primer lugar, se realiza la llamada a la función introduciendo como entrada el conjunto de datos (test2), el número mínimo de veces que se creará un nuevo cluster (por ejemplo 5) y uno de los 4 métodos creados (en este ejemplo se utilizará la variante *Gower*). Así, la llamada queda:

```
1 ejemplo=NaiveBayes(test2,5,"Gower")
```

Ahora, en la figura 4.2 se muestra la salida.

```
[[1]]
mejor.mejor
   0    1    2    3
1009 1907 1177  907

[[2]]
   [1] 3 1 1 1 1 2 2 0 3 1 0 3 3 2 0 1 1 3 1 0 2 3 2 0 1 0 1 3 2 2 1 2 0 3 2 0 1 2 2 2 2 1 1 2 1 3 3 2 1 2 0 2 1 0 2 2
  [57] 2 1 2 1 2 2 3 1 1 1 1 3 3 0 0 2 2 0 1 3 0 3 3 1 2 3 1 1 0 2 3 1 2 1 3 3 0 3 3 1 1 2 2 1 0 1 1 1 1 0 0 1 3 1 0 1
 [113] 3 1 0 2 0 3 0 1 0 3 3 1 3 0 2 0 3 1 1 0 2 0 2 2 0 2 2 1 1 2 3 2 2 1 0 1 2 2 1 1 2 0 0 2 3 1 0 0 1 2 1 2 1 1 2 1
 [169] 1 2 1 1 3 1 0 2 1 0 1 1 0 1 2 2 1 2 3 2 2 1 2 2 1 0 2 3 3 1 3 3 3 0 2 1 3 2 1 1 1 2 0 0 1 2 2 3 2 0 2 2 3 1 3 1
 [225] 0 3 2 0 1 2 0 1 0 1 3 0 2 0 1 0 0 0 2 0 3 2 0 1 1 2 3 1 1 1 2 0 2 0 2 0 1 0 0 2 0 2 1 0 3 1 1 2 2 1 2 1 1 3 1 1
 [281] 2 1 2 2 2 3 1 3 3 1 2 1 2 2 2 0 3 1 1 1 2 0 1 1 1 1 1 0 1 1 2 1 0 0 0 1 0 0 0 3 1 3 1 0 0 3 3 1 1 1 3 3 0 1 3 1
 [337] 3 0 0 1 2 1 2 3 1 1 1 2 1 2 1 1 1 3 0 1 1 1 1 2 1 2 1 2 1 2 0 1 2 0 1 3 3 0 1 2 1 3 1 2 1 1 1 0 1 1 1 2 1 0 2 3
 [393] 3 2 3 2 1 0 1 1 0 0 1 1 2 3 0 2 0 1 2 1 1 0 0 0 0 3 2 1 2 3 1 1 3 1 1 2 2 1 3 1 1 0 0 1 1 1 1 0 3 1 0 2 1 0 2 2
 [449] 1 1 2 0 2 0 3 2 2 2 0 3 0 1 0 2 1 2 2 3 1 1 0 1 1 3 2 0 0 3 1 0 1 1 0 1 1 2 3 1 3 1 3 3 2 3 0 1 1 2 0 1 2 1 1 0
 [505] 2 1 1 1 1 1 0 2 1 0 1 0 1 1 0 2 1 0 3 1 3 1 1 0 0 1 2 1 1 1 2 2 1 0 0 3 1 3 1 2 2 1 1 3 1 1 1 0 0 0 1 0 1 3 1 0
 [561] 1 3 1 1 1 2 1 1 1 3 1 1 0 3 3 1 1 0 2 1 1 1 0 3 2 1 1 2 2 0 1 1 0 1 1 2 2 1 3 3 2 0 3 1 0 1 2 1 1 1 1 1 2 1 2 2
 [617] 0 2 2 1 0 3 1 3 3 0 2 1 2 1 3 3 3 3 2 0 0 1 2 3 3 2 0 0 0 1 1 3 0 0 2 1 1 2 0 0 2 1 1 0 1 3 1 2 1 2 2 1 1 3 1 3
 [673] 1 1 1 3 0 2 3 2 1 2 0 1 0 2 1 1 3 1 0 0 0 3 2 0 1 1 1 3 1 1 3 1 0 1 0 1 2 1 1 1 3 0 2 1 0 2 1 2 2 2 0 0 3 2 1 3 3
 [729] 0 0 1 1 1 3 1 2 3 0 3 3 2 1 1 0 3 1 0 3 2 1 0 0 1 3 1 0 0 3 1 1 0 1 2 1 3 2 1 1 1 1 1 1 1 1 0 1 1 3 1 1 1 0 3 3 2
 [785] 1 2 2 1 2 3 0 2 2 2 1 2 1 1 0 2 0 2 0 2 1 1 1 3 2 0 3 3 1 2 0 1 1 0 3 1 0 1 0 2 2 0 0 1 1 3 1 0 1 0 0 2 1 1 3 1
 [841] 3 0 2 2 3 1 3 1 2 2 1 0 2 2 3 2 2 3 1 2 3 1 0 3 1 1 3 1 2 3 3 3 2 2 0 1 3 2 2 3 2 3 1 0 2 2 0 1 1 2 1 1 3 0 2 1
 [897] 0 1 1 2 1 2 3 1 3 1 0 1 3 0 2 2 2 1 1 3 2 1 2 2 0 1 2 1 0 1 1 2 3 0 1 3 1 1 1 3 2 2 1 0 3 1 1 2 0 1 3 3 3 0 0 1
 [953] 1 2 1 3 2 2 1 0 1 0 3 0 2 2 0 1 1 2 2 1 1 2 1 3 3 2 2 2 2 3 1 1 2 0 1 2 0 1 1 1 1 1 3 2 2 3 0 3 0
 [ reached getOption("max.print") -- omitted 4000 entries ]
Levels: 0 1 2 3

[[3]]
[1] -27079.60 -26960.04 -26490.39 -26508.91 -26511.58
```

Figura 4.2: Ejemplo de salida de la función

El primer elemento de la lista refleja que el mejor modelo creado se compone de 4 grupos y además se cuentan los casos que forman parte de cada uno. Puede verse que tres de los grupos tienen tamaños parecidos, mientras que la clase 1 es el grupo más numeroso.

En el segundo puede verse concretamente a qué clase pertenece cada elemento, luego los grupos quedan perfectamente determinados y pueden ser utilizados para, por ejemplo, realizar otros tratamientos con los elementos de una determinada clase.

Por último, se listan los BICs de los mejores modelos obtenidos, con el fin de ver cuán grande (o pequeña) es la diferencia entre ellos en cuanto a este criterio. En concreto, se ve cómo el tercer elemento del vector es el mayor de todos (correspondiente a los modelos de 4 grupos como se indicó con el primer elemento de la salida) y que no existe una diferencia grande entre los modelos.

Finalmente, a modo de resumen y para dar una expresión más clara del algoritmo, se representa el proceso en pseudocódigo. Se comienza con el bucle interior en el Algoritmo 1 y a continuación se exponen el bucle exterior (Algoritmo 2) y la unión de ambos, que da lugar a la función completa (Algoritmo 3).

Algorithm 1: Bucle interior

Input : Un modelo inicial M
Output: Una lista L de modelos generados con el método Naive-Bayes

```
1 L[1] = M
2 while (L[i] mejore a L[i − 1] o i ≤ 20 iteraciones) e i ≤ 150 do
3 |   P= Distribuciones de probabilidad de L[i].
4 |   Generar L[i + 1] dando valores a la variable clase en función de P.
5 |   B1 = BIC(L[i − 1])
6 |   B2 = BIC(L[i])
7 end
```

Algorithm 2: Bucle exterior

Input : Los datos y la estructura de la red bayesiana
Output: El mejor modelo generado

```
1  while (M[j] mejore a M[j − 1] o j ≤ n) y j ≤ 15 do
2  |   L=Bucle interior (Algoritmo 1).
3  |   M[j]=Mejor modelo de entre los de la lista L.
4  |   if metodo=Random then
5  |   |   dividir=cluster aleatorio
6  |   else if metodo=KL then
7  |   |   dividir=cluster con menor disimilaridad de Kullback-Leibler a la
   |   |     distribución uniforme.
8  |   else if metodo=Gower then
9  |   |   dividir=cluster con menor silueta media.
10 |   else if metodo=NoDiv then
11 |   |   dividir permanece vacío.
12 |   else
13 |   |   Error
14 |   Crear un nuevo cluster.
15 |   L[1] =Modelo inicial generado al fracturar el cluster de la variable dividir.
16 |   BIC1 =BIC(M[j − 1])
17 |   BIC2 =BIC(M[j])
18 end
```

Algorithm 3: función NaiveBayes

1 función NaiveBayes ($datos.ini, n, metodo$)
Input : El conjunto de datos, el número mínimo de grupos que se crearán y el método elegido.
Output: Una lista con el recuento de elementos en el mejor modelo, el grupo al que pertenece cada elemento y los BICs de los mejores modelos con cada número de grupos.
2 Datos=factor(datos.ini).
3 Añadir la variable de clase a Datos.
4 Generar el modelo inicial aleatoriamente.
5 Crear la estructura de grafo.
6 Ejecutar el bucle exterior (Algoritmo 2).
7 Crear la lista con la salida de la función.

5 Experimentos

Una vez implementado el algoritmo, se probará mediante una serie de conjuntos de datos la eficiencia y el buen funcionamiento de éste. Además, se compararán entre sí las diferentes formas de dividir los grupos para determinar, si fuera posible, cuál de ellas puede considerarse la mejor, basándose en varios criterios. Después de esto, se comparará la mejor variante del algoritmo creado con las funciones de R *kmodes* y *hclust* ya comentadas anteriormente; con esto se pretende ver cómo se comporta el algoritmo creado frente a otras funciones enfocadas de clustering.

Los conjuntos de datos con los que se trabajará han sido obtenidos tanto de los propios paquetes de R como del sitio web `http://archive.ics.uci.edu/ml/index.php`, que es un repositorio con gran cantidad de conjuntos de datos destinados a la experimentación en el análisis y tratamiento de los mismos. Cabe mencionar que a algunos de ellos se le han añadido observaciones de forma aleatoria para dar sentido al problema de clustering en ellos, ya que en un principio sólo había una observación para cada combinación de variables. En concreto, los 6 conjuntos seleccionados han sido:

- *Test1*. Es un 'toy problem' utilizado como ejemplo en la documentación de la función *kmodes*. Está compuesto por 100 observaciones de 5 variables que toman los valores 0, 1 ó 2.
- *Test2*. Es otro 'toy problem', esta vez un poco más amplio, obtenido del paquete *bnlearn*. Consta de 5000 tratamientos con 6 variables, que toman los valores *a*, *b* o *c*, salvo la última variable que únicamente puede ser *a* o *b*.
- *Balance*. Este conjunto de datos recoge observaciones sobre los pesos y las distancias puestos en una balanza de platos. Hay 2625 casos para 4 variables (peso y distancia en el plato izquierdo y en el derecho) que toman valores del 1 al 5, siendo 1 el menor peso o la menor distancia y 5 los mayores.
- *Heart*. Se recogen datos de 1267 pacientes sobre 22 variables relacionadas con el diagnóstico de enfermedades cardíacas mediante imágenes SPECT. Todas las variables son binarias.
- *Poker*. En esta base de datos se tienen 10000 observaciones y 10 variables que representan posibles manos obtenidas en una partida de póker (número y palo de cada una de las 5 cartas).
- *Chess*. Conjunto de 36 variables que describen diferentes posiciones de un tablero de ajedrez en el cual sólo se encuentran rey y peón blancos frente a rey y torre negros. Se recogen 3196 observaciones para su estudio.

En primer lugar, como ya se comentó, se llevan a cabo las comparaciones entre las 4 variantes de la función creada. Como se ha introducido en el algoritmo un componente aleatorio, tanto en la generación de los modelos como en la variante *Random*, se han realizado 5 ejecuciones para cada variante y conjunto de datos con el fin de atenuar dicha aleatoriedad. Así pues, como criterios de evaluación se utilizan: la mediana del

número de clusters creados, el BIC medio conseguido por cada método, la silueta media (que, recuérdese, expresaba cómo de bien clasificados se encuentran los datos) y el tiempo medio de ejecución.

No se han dividido las bases de datos en conjuntos de aprendizaje y test (como suele ser habitual en otros procedimientos de clasificación) por dos motivos: primero, para que la comparación que se hará posteriormente con los métodos *hclust* y *kmodes* sea justa, puesto que estos dos métodos no hacen tal división, y segundo, porque en el caso concreto del clustering no interesa tanto que el modelo sea genérico como que sea capaz de encontrar la estructura de grupos más representativa del total de los datos.

Los resultados se muestran en la Tabla 5.1.

Conjunto de datos		*NoDiv*	*Random*	*KL*	*Gower*
Test1	#Clusters	2	2	2	2
	BIC	−580.8095	−530.8907	−529.3408	−529.5514
	Silueta	0.5963	0.5984	0.5978	0.5994
	Tiempo (s)	0.6692	0.6852	0.7334	0.8190
Test2	#Clusters	3	5	4	5
	BIC	−26755.48	−26671.62	−26502.02	−26462.89
	Silueta	0.4423	0.4326	0.4551	0.4507
	Tiempo (s)	15.2979	14.2983	21.5878	36.5365
Balance	#Clusters	2	2	2	2
	BIC	−18712.14	−18690.98	−18668.18	−18673.32
	Silueta	0.3899	0.3936	0.3922	0.3935
	Tiempo (s)	7.0120	7.0398	7.0182	11.0564
Heart	#Clusters	8	9	13	11
	BIC	−13518	−13548.01	−13237.97	−13359.74
	Silueta	0.3401	0.3375	0.3683	0.3594
	Tiempo (s)	9.2655	10.6306	20.9088	20.2214
Poker	#Clusters	2	2	2	2
	BIC	−204367.4	−204383	−204373.6	−204358.1
	Silueta	0.2081	0.2085	0.2082	0.2090
	Tiempo (s)	48.3221	52.3244	35.8352	121.65
Chess	#Clusters	8	9	12	11
	BIC	−39953.22	−39988.35	−39562.28	−39465.99
	Silueta	0.2556	0.2487	0.2594	0.2597
	Tiempo (s)	24.6508	30.6226	20.8723	58.8788

Tabla 5.1: Comparación entre los diferentes métodos creados

Se comienza analizando el número de clusters junto al BIC. Obsérvese cómo con los métodos *KL* y *Gower* se suelen obtener los mejores valores del BIC, además de un número de grupos parecido, mientras que para *NoDiv* y *Random* se suele conseguir un número de clusters menor pero para valores menores del BIC.

Para mayor claridad, se exponen de manera gráfica los resultados para el BIC, habiendo normalizado previamente los valores dividiendo por el elemento de mayor valor absoluto de los BICs de cada conjunto de datos. Así pues, al ser los datos originales negativos, cuanto más cercanos a 0 mejor será el valor del BIC. En la Figura 5.1 se recoge esta comparación.

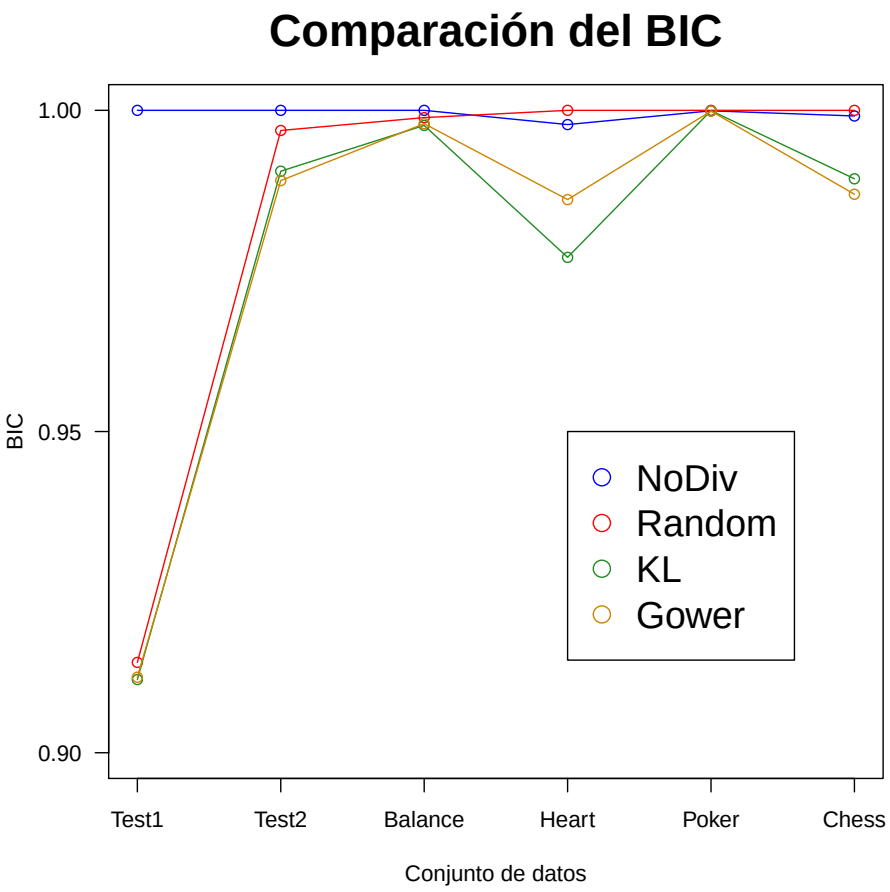

Figura 5.1: Gráfico de comparación de los BIC

Se observa claramente que las líneas verde y amarilla se encuentran siempre por debajo de las otras dos, como ya se adelantó. Además, la correspondiente al método de *Gower* suele encontrarse al mismo nivel de la de *Kullback-Leibler*, excepto para el conjunto *heart* donde existe una diferencia algo mayor.

Para verificar matemáticamente las diferencias entre los BICs, se llevará a cabo la comparación de los valores para cada uno de los métodos mediante el test de Friedman (puede consultarse [13]), con el cual se determina si un conjunto de muestras pueden considerarse estadísticamente iguales o no. En caso negativo, se realizan además los contrastes dos a dos. Se toma el nivel de confianza habitual del 95 %, que significa que la hipótesis de igualdad se acepta si el P-valor obtenido es mayor que 0.05. El P-valor general ha sido 0.0369, por lo que existe diferencia significativa entre los métodos y es necesario realizar los contrastes dos a dos para determinar entre cuáles de ellos. Los

P-valores se recogen en la Tabla 5.2.

	NoDiv	Random	KL
Random	0.9961		
KL	0.1137	0.0665	
Gower	0.0665	0.0366	0.9961

Tabla 5.2: P-valores de los tests de Friedman dos a dos para los BICs

Puede verse entonces que la igualdad estadística con respecto al BIC se da muy fuertemente para los métodos *Random* y *NoDiv*, y *KL* y *Gower*. Además, los únicos considerados estadísticamente diferentes son *Gower* y *Random*.

Por último, en la Figura 5.2 se muestran las comparaciones entre cada uno de los métodos, donde los coloreados significan que existe diferencia estadística entre ellos. La notación usada en el eje de abscisas es la siguiente:

1. *KL - Gower*
2. *NoDiv - Gower*
3. *Random - Gower*
4. *NoDiv - KL*
5. *Random - KL*
6. *Random - NoDiv*

Se realiza ahora un estudio similar para las siluetas medias de cada método.

En primer lugar, se grafican los datos y se muestran en la figura 5.3, observándose que, de nuevo, para los métodos *KL* y *Gower* se obtienen valores más altos que para los otros dos, esto es, las primeras variantes clasifican mejor los datos.

Como antes, se lleva a cabo un test de Friedman para determinar si los vectores de las siluetas pueden considerarse iguales estadísticamente o no. El P-valor general ha sido de 0.0365, luego existen diferencias significativas. Los P-valores para los contrastes dos a dos se dan en la Tabla 5.3.

	NoDiv	Random	KL
Random	0.8078		
KL	0.2786	0.8078	
Gower	0.0367	0.2786	0.8078

Tabla 5.3: P-valores de los tests de Wilcoxon para la comparación de las siluetas

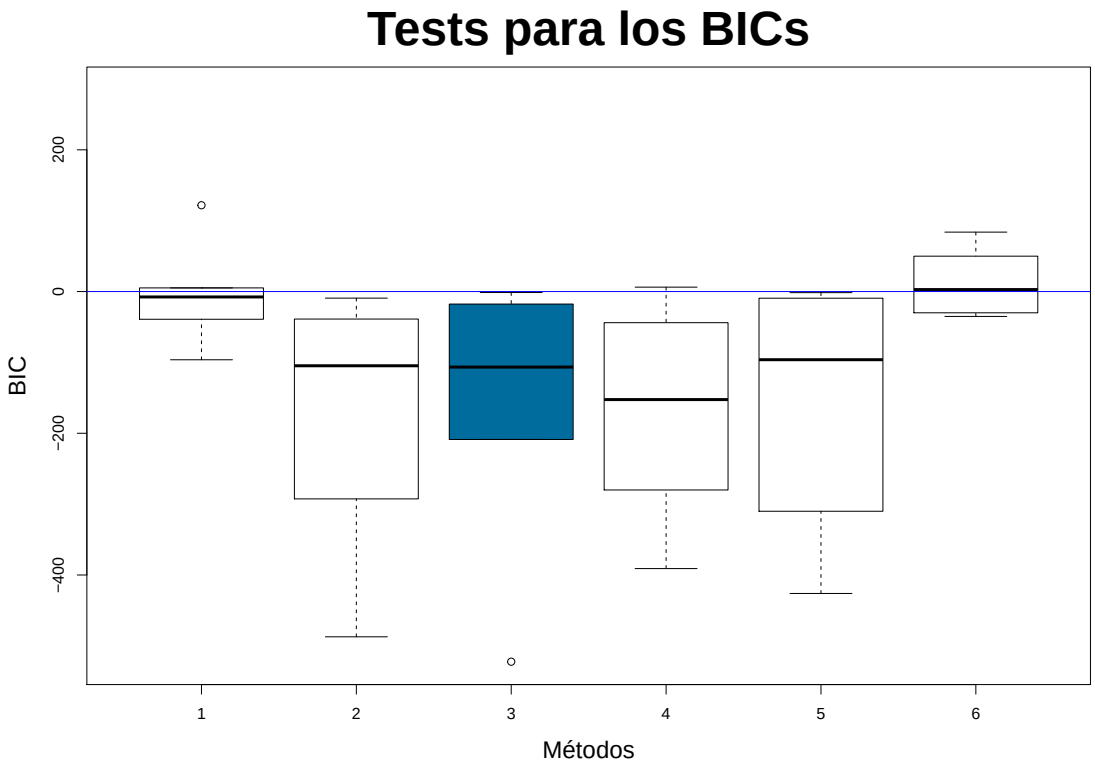

Figura 5.2: Gráfico de comparación de los BICs dos a dos

Para este criterio sólo se han considerado diferencias estadísticas significativas entre los métodos *NoDiv* y *Gower*. Esto se muestra de forma gráfica en la Figura 5.4, donde la notación es la misma que antes.

Así, teniendo en cuenta todo lo anterior, los métodos más eficientes son *KL* y *Gower*, que además han sido considerados estadísticamente iguales para ambos criterios. Sin embargo, *Gower* ha sido el único con diferencia significativa respecto a *NoDiv* y *Random*, lo que hace ver que los resultados obtenidos con *Gower* son mejores que los de *KL*.

Con respecto a *NoDiv* y *Random* (también considerados estadísticamente iguales en ambos casos), obtienen valores peores por tratarse de modelos generados sin ningún criterio de mejora en particular.

Como se puede observar claramente en las Figuras 5.2 y 5.4, en las comparaciones de los métodos *KL* y *Gower* con *NoDiv* y *Random*, prácticamente siempre, tanto para las siluetas como para el BIC, salen beneficiados los primeros, aunque solamente una de esas comparaciones en cada caso es estadísticamente significativa. No parece descabellado suponer que en un estudio más amplio todas estas diferencias sí sean significativas. Asimismo, en tres de las seis bases de datos, el número óptimo de grupos seleccionado por los cuatro métodos es de 2, lo que resta influencia al método de división con respecto al resultado final, como se puede ver en las Figuras 5.1 y 5.3.

En el tiempo de compilación para cada método y cada conjunto de datos existe cla-

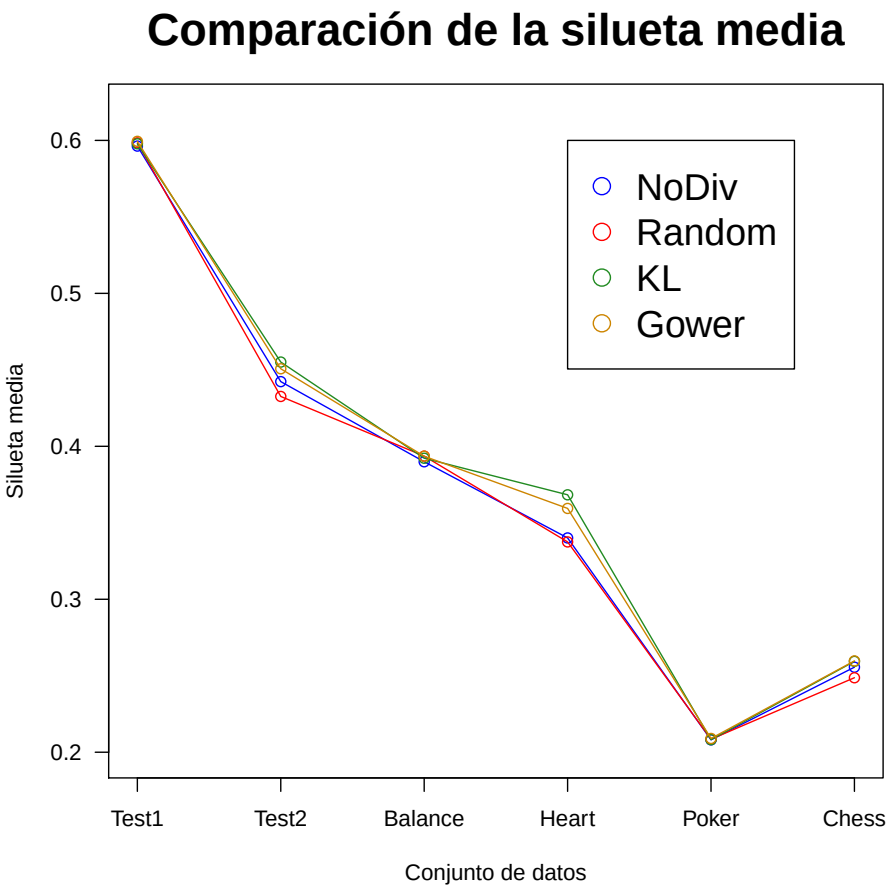

Figura 5.3: Gráfico de comparación de las siluetas medias

ramente una gran diferencia, siendo mayor el tiempo de *Gower* por tener que calcular las siluetas de cada observación en cada iteración. Aún así, la distinción frente al resto de métodos hace que merezca la pena esa espera.

En definitiva, de entre las 4 variantes del algoritmo creadas, la que ofrece mejores resultados en un tiempo razonable es la variante *Gower*.

Una vez elegida la mejor variante, se estudia el comportamiento de ésta frente a las funciones *hclust* y *kmodes*. Como estas dos no ofrecen ningún criterio de elección de modelos, no proporcionan un número óptimo de clusters, luego se confrontará con la cantidad de grupos dada por la función *NaiveBayes* en su versión *Gower*. Además, tampoco suponen una estructura en el modelo ni distribución de probabilidad alguna, luego no existe la opción de comparar a través del BIC y, por tanto, sólo se tendrán en cuenta la silueta media y el tiempo de procesamiento. Los datos se muestran en la Tabla 5.4 y la representación gráfica de las siluetas medias se puede ver en la Figura 5.5.

Nótese cómo, de las líneas gruesas dentro de cada caja, que simbolizan la media de las siluetas medias, la del método *Gower* se encuentra por encima de las demás, lo que

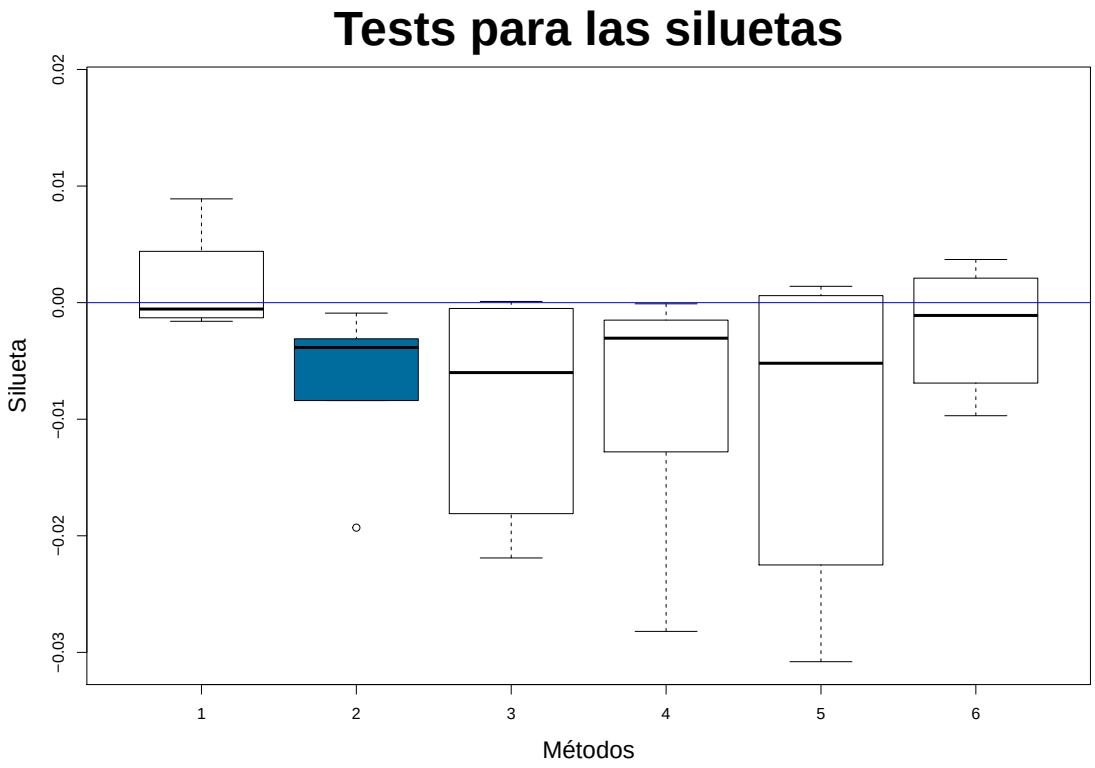

Figura 5.4: Gráfico de comparación de las siluetas medias dos a dos

Conjunto de datos		*Gower*	*hclust*	*kmodes*
Test1	Silueta	0.5994	0.6010	0.6040
	Tiempo (s)	0.8190	0.014	0.7710
Test2	Silueta	0.4507	0.0587	0.3655
	Tiempo (s)	36.5365	4.6429	45.0496
Balance	Silueta	0.3935	0.5085	0.4203
	Tiempo (s)	11.0564	1.1201	13.6188
Heart	Silueta	0.3594	0.2780	0.2734
	Tiempo (s)	20.2214	0.2250	40.1863
Poker	Silueta	0.2090	0.2316	0.2329
	Tiempo (s)	121.65	19.6301	200.0614
Chess	Silueta	0.2597	0.1176	0.1207
	Tiempo (s)	58.8788	2.3311	472.38

Tabla 5.4: Comparación entre el método *KL* y los implementados en R

sugiere que los datos se han clasificado mejor con dicho método.

Para verificarlo, se realiza de nuevo un test de Friedman con las siluetas, siendo en este caso el P-valor general de 0.8322, luego no existe diferencia significativa entre las funciones comparadas y por tanto los datos se clasifican de igual manera con cualquiera de ellas.

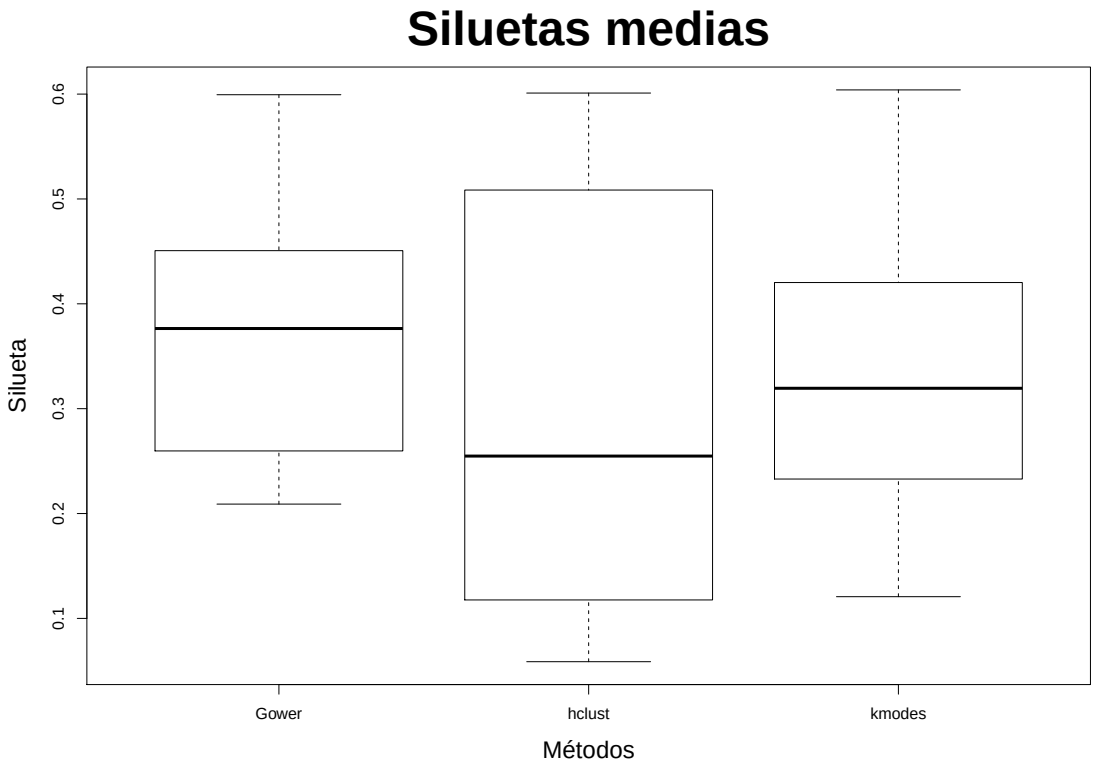

Figura 5.5: Gráfico de comparación de las siluetas con las funciones de R

En cuanto al tiempo, cabe mencionar la enorme duración de la función *kmodes*, especialmente para conjuntos de datos con un número elevado de variables. Por el contrario, *hclust* es una función muy rápida a la hora de realizar la agrupación de los elementos.

Como conclusión, no se puede decir estrictamente que una función sea mejor que otra, ya que cada una tiene su propio modo de operar, pero, dentro de la base del clustering, se ha observado que la función creada en este trabajo ofrece resultados acordes a los de las otras ya implementadas en R. Destacar también de la función *NaiveBayes* que, mientras que las otras se limitan a calcular un modelo con el número de grupos dado por el usuario, ésta crea como mínimo 20 para cada número de clusters y selecciona el mejor de todos ellos en cada caso, y todo ello en un tiempo razonable. De esta forma se asegura obtener un número de grupos que puede considerarse el más adecuado.

6 Conclusiones

Una vez realizado todo el estudio, decir que se han cumplido todos los objetivos propuestos para el trabajo al haberse construido la función según lo planeado, es decir, una función que realiza el clustering no jerárquico de un conjunto de datos a través de uno de los cuatro métodos de división de grupos creados.

En cuanto al segundo objetivo, en la comparación entre las cuatro variantes, la que ha obtenido mejores resultados en cuanto a BIC, silueta y tiempo medios ha sido la de Kullback-Leibler, considerándose entonces ésta como la más eficiente de las cuatro.

Destacar nuevamente que esta función genera una gran cantidad de modelos y de entre todos ellos selecciona el mejor según el criterio del BIC, a diferencia de las funciones ya existentes en R que sólo calculan un determinado modelo para el número de grupos dado. Así, además de seleccionarse un número de clases adecuado, se permite ver cómo evoluciona el clustering según aumenta esta cantidad.

Con respecto a la comparación con *hclust* y *kmodes*, se pone de manifiesto que *NaiveBayes* ofrece resultados a la altura de estas dos funciones. Es por ello que se pretende enviar la función creada al repositorio oficial de R `https://cran.r-project.org/` para así compartir el código creado con cualquier usuario de la comunidad de R.

Por último, como posibles opciones de ampliación de este trabajo destacan el uso no exclusivo de variables categóricas o la introducción de otros modelos de clustering en redes bayesianas, como por ejemplo el TAN (Tree Augmented Naive-Bayes)[14] para el cual el conjunto de padres de cada una de las variables predictoras X_i contiene necesariamente a la variable de clase y además puede contener también otra variable X_j, pero sólo una. Un ejemplo de tal estructura puede verse en la Figura 6.1.

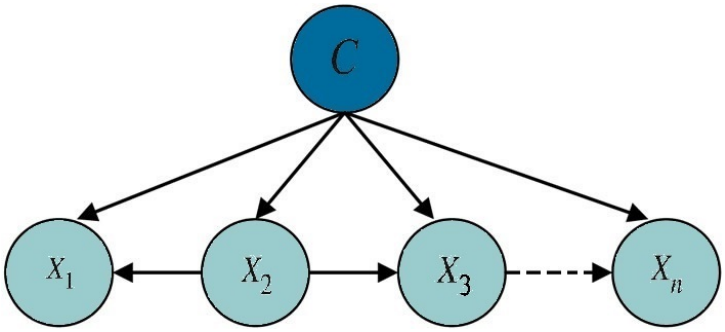

Figura 6.1: Ejemplo de grafo del modelo TAN

Código en R

A

```
1  NaiveBayes=function(datos.ini,n,metodo){
2
3    #Se transforma la matriz de datos en factores.
4    datos.ini=as.data.frame(datos.ini)
5    datosfac=as.data.frame(lapply(datos.ini, factor))
6
7
8    #Se anade la variable de clase y se genera el primer modelo al azar
       .
9    datos=cbind(datosfac,as.factor(rbinom(length(datosfac[,1]),1,0.5)))
10   names(datos)[length(names(datos))]="C"
11
12
13   #Se crea el grafo con la estructura del modelo Naive-Bayes.
14   variables=colnames(datos)
15   g = empty.graph(variables)
16   for(i in 1:(length(variables)-1)){
17     g=set.arc(g,"C",variables[i])
18   }
19
20
21   #Inicializacion de variables
22   c.valores=c(0,1)
23   seguridad.exterior=1
24   mejor.modelo=list()
25   bic.mej.ant=-Inf
26   bic.mej.post=0
27   bic.mejores=vector()
28
29
30   #Inicio del bucle exterior
31   while((bic.mej.ant<bic.mej.post||seguridad.exterior<=n)&&seguridad.
       exterior<=15){
32
33
34   #Inicializacion de variables
35   seguridad.interior=1
36   lista.modelos=list()
37   bic.antiguo=-Inf
38   bic.nuevo=score(g,data=datos,type = "bic")
39   bic.grupos=bic.nuevo
40
41   #Inicio del bucle interior
42   while((bic.antiguo<bic.nuevo||seguridad.interior<=20)&&seguridad.
       interior<=150){
43
44   lista.modelos[[seguridad.interior]]=datos[,length(variables)]
```

```
45
46  # Calculo de probabilidades en el modelo
47  bn=bn.fit(g,data=datos,method="bayes",iss=10)
48  pc=bn[[length(variables)]]$prob
49  px.c=matrix(nrow=length(datos[,1]),ncol =length(variables)-1 )
50  pc.x=matrix(nrow=length(datos[,1]),ncol = length(levels(datos[,
      length(variables)])))
51
52  for(k in 1:length(c.valores)){
53    for(j in 1:(length(variables)-1)){
54      px.c[,j]=bn[[j]]$prob[datos[,j],k]
55    }
56    pc.x[,k]=pc[k]*apply(px.c,1,prod)
57  }
58  pc.x=pc.x/apply(pc.x,1,sum)
59
60
61  #Generacion del modelo nuevo a partir de pc.x
62  rand=rmultinom(1,1,pc.x[1,])
63  for(i in 2:length(datos[,1])){
64    rand=cbind(rand,rmultinom(1,1,pc.x[i,]))
65  }
66  for(l in 1:length(c.valores)){
67    datos[which(rand[l,]==1),length(variables)]=l-1
68  }
69
70
71  #Calculo del BIC.
72  bic.antiguo=bic.nuevo
73  bic.nuevo=score(g,data=datos,type = "bic")
74  bic.grupos=append(bic.grupos,bic.nuevo)
75
76  seguridad.interior=seguridad.interior+1
77  }
78  #Final del bucle interior.
79
80
81  #Seleccion del mejor modelo
82  bic.mejores[seguridad.exterior]=max(bic.grupos)
83  mejor.modelo[[seguridad.exterior]]=lista.modelos[[min(which(bic.
      grupos==bic.mejores[seguridad.exterior]))]]
84
85
86  #Seleccion del cluster a dividir
87  dividir=-1
88  sil=0
89  if(metodo=="Random"){
90    dividir=sample(c.valores,1)
91  }
92
```

```
93   else if(metodo=="KL"){
94     disim=vector(length=length(variables)-1)
95     kl.total=vector(length = length(c.valores))
96     probs=vector()
97     grupo=0
98
99     for(i in 1:length(c.valores)){
100      grupo=datos[which(datos$C==i-1),]
101      for(q in 1:(length(variables)-1)){
102        probs=vector()
103        variab=grupo[,q]
104        for(m in 1:length(levels(datos[,q]))){
105          probs[m]=nrow(grupo[which(variab==(levels(variab)[m])),])/
               length(variab)
106        }
107        disim[q]=kl.dist(probs,rep(1/length(probs),length(probs)),
             base = exp(1))[[3]]
108      }
109      kl.total[i]=sum(disim)
110    }
111    dividir=which.min(kl.total)-1
112  }
113
114  else if(metodo=="Gower"){
115    sil = silhouette(as.integer(datos$C),daisy(datos,metric="gower"))
116    dividir=which(summary(sil)[[2]]==min(summary(sil)[[2]]))-1
117    outputfile = paste("~/R/plots/Grafico ",seguridad.exterior,".pdf"
         , sep="")
118    pdf(file=outputfile)
119    plot(sil,main="Comparacion de siluetas medias",xlab = "Silueta
         media",col="#006C9D")
120    dev.off()
121  }
122
123  else if(metodo=="NoDiv"){
124
125  }
126
127  else{
128    stop("Por favor inserte un metodo valido")
129    }
130
131
132  #Adicion del nuevo cluster
133  c.valores=append(c.valores,length(c.valores))
134  levels(datos$C)=c.valores
135
136
137  #Generacion del modelo inicial con el cluster nuevo
138  if(metodo=="NoDiv"){
```

```
139     datos$C=as.factor(sample(c.valores,length(datos$C),replace = TRUE
            ))
140   }
141   else{
142     datos[datos$C==dividir,"C"]=as.factor(sample(c(dividir,length(c.
            valores)-1),length(datos[datos$C==dividir,"C"]),replace = TRUE
            ))
143   }
144
145
146   #Extraccion del BIC de los mejores modelos.
147   if(seguridad.exterior==1){
148
149   }
150   else{
151     bic.mej.ant=bic.mejores[seguridad.exterior-1]
152     bic.mej.post=bic.mejores[seguridad.exterior]
153   }
154
155
156   seguridad.exterior=seguridad.exterior+1
157   }
158   #Final del bucle exterior.
159
160
161   #Creacion de la salida.
162   mejor.mejor=mejor.modelo[[which(bic.mejores==max(bic.mejores))]]
163
164   salida=list(table(mejor.mejor),mejor.mejor,bic.mejores)
165   return(salida)
166 }
```

Bibliografía

[1] T. Bayes, *An Essay towards solving a Problem in the Doctrine of Chances*, Phil. Trans., **53**, (1763), 370-418.

[2] J. Pearl, *A constraint propagation approach to probabilistic reasoning*, Uncertainty in Artificial Intelligence Annual Conference on Uncertainty in Artificial Intelligence, **85**, (1986), 357-369.

[3] O. Pourret et al., *Bayesian Networks: A Practical Guide to Applications*, Wiley, 2008.

[4] M. M. Deza, E. Deza, *Encyclopedia of Distances*, Springer, 2009.

[5] P. A. Aguilera et al., *Bayesian Networks in Environmental Modelling*, Environmental Modelling & Software, **26**, (2011), 1376-1388.

[6] M. Scutari, J. B. Denis, *Bayesian Networks With Examples in R*, CRC Press, 2015.

[7] A. Fernández et al., *Data clustering using hidden variables in hybrid Bayesian networks*, Prog Artif Intell,**2**, (2014), 141–152.

[8] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, Wiley Interscience, 2001.

[9] A.P. Dempster, N.M. Laird, D. B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, J. R. Stat. Soc. B, **39**, (1977), 1-38.

[10] J. A. Gámez, J. M. Puerta, *Sistemas expertos probabilísticos*, Ediciones de la Universidad de Castilla-La Mancha, 1998.

[11] W. R. Gilks, S. Richardson, D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman and Hall, 1996.

[12] P.J. Rousseeuw, *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*, Computational and Applied Mathematics, **20**, (1987), 53–65.

[13] M.Friedman, *The use of ranks to avoid the assumption of normality implicit in the analysis of variance*, Journal of the American Statistical Association, **32**, (1937), 675–701.

[14] N. Friedman et al., *Bayesian Networks Classifiers*, Machine Learning, **29**, (1997), 131–163.