UNIVERSIDAD MIGUEL HERNÁNDEZ
FACULTAD DE CIENCIAS SOCIALES Y JURÍDICAS

TRABAJO FIN DE GRADO
GRADO EN ESTADÍSTICA EMPRESARIAL
CURSO ACADÉMICO 2019-2020

Importancia de la selección de características en la agrupación de individuos

Autora
Marina Espinosa Fernández
Tutora
Mercedes Landete Ruiz

Índice

1 Introducción

El volumen de información del que se dispone actualmente, el cual experimenta un crecimiento constante, supone un reto a la hora de abordar cualquier tipo de análisis de datos. La existencia de gran cantidad de cantidad de variables y características puede resultar un problema de cara a la aplicación de técnicas estadísticas con diversos fines de análisis de dichos datos. Es en este punto, donde surgen como solución a esto, técnicas de preprocesamiento de datos como es la selección de características, que se trata de una técnica de búsqueda de las características más relevantes de un conjunto de datos.

El presente trabajo de fin de grado abarca la importancia de esta parte del preprocesamiento de datos conocida como la selección de características, dentro del contexto de la agrupación de individuos. La agrupación de individuos también conocida como clusterización se basa en la búsqueda de grupos homogéneos dentro de un conjunto de individuos y así poder segmentarlos, a través del uso de técnicas multivariantes. Esta técnica es implementada en la actualidad en una gran variedad de sectores por los numerosos beneficios que aporta como, identificación de patrones, mayor conocimiento de clientes lo que se traduce en una mejora en la experiencia del cliente, clasificación de enfermedades atendiendo a sintomatología, estudios de clasificación arqueológicos en base a su composición química y un gran número de aplicaciones exitosas en múltiples campos.

El análisis clúster es posible llevarlo a cabo mediante la aplicación de distintas técnicas o métodos, pero para el desarrollo del presente proyecto se plantea el uso del conocido algoritmo de Kruskal publicado en 1956 por Joseph Kruskal que permite encontrar el árbol de expansión mínima entre un conjunto de nodos de una red. Lo que se pretende es utilizar las bases de este algoritmo y conseguir con estas llevar a cabo la agrupación de individuos de forma muy similar al comportamiento de los métodos jerárquicos aglomerativos basados en la unión de mínima distancia, representados habitualmente mediante un dendograma, recurriendo frecuentemente a la comparativa en-

tre ambos métodos.

La importancia de la selección de características de forma previa a la agrupación de individuos se tratará mediante un caso práctico a través del cual se pretende ilustrar el impacto que puede llegar a tener el hecho de trabajar con unas u otras características y, consecuentemente las mejoras que se pueden obtener con una selección óptima de estas.

El objetivo del proyecto se encuentra desarrollado en tres secciones:

- En la primera sección se tratan los conceptos de árbol de expansión y de árbol de expansión mínima y su alcance a través del algoritmo de Kruskal, lo cual serán las bases para el posterior desarrollo del algoritmo de agrupación de individuos basado en este.

- En la siguiente sección se leva a cabo la búsqueda y el desarrollo del algoritmo de Kruskal con el fin de agrupar individuos, de forma similar a los métodos jerárquicos aglomerativos basados en la unión de mínima distancia. Se mostrará el funcionamiento de este y su interpretación, relacionando los resultados obtenidos con los resultados generados con métodos jerárquicos.

- En la última sección se trabaja con un caso práctico para ilustrar la importancia de la selección de características como parte del preprocesamiento previo a la agrupación de individuos. Y, a su vez, se plantea una alternativa de método de selección de características basado en una comparativa de resultados de agrupación, mediante el algoritmo de Kruskal, trabajando con distintas combinaciones de características.

La asignatura Minería de datos, impartida en segundo curso del grado, ha resultado ser una base primordial para el desarrollo del presente trabajo, proporcionando todos los conceptos, puestos en práctica, a cerca de la agrupación de individuos, concretamente el análisis jerárquico que es el más empleado en el estudio, lo que ha facilitado la adaptación del algoritmo de Kruskal a este fin, la agrupación. Para el desarrollo del algoritmo y para la totalidad del proyecto se ha utilizado como lenguaje de programación

R, bajo la interfaz gráfica de RStudio. El conocimiento de este ha sido adquirido a lo largo del grado ya que ha sido el lenguaje empleado mayoritariamente para llevar a cabo todas las prácticas de las distintas asignaturas. Como novedades y nuevos conocimientos adquiridos se ha tratado el tema de los árboles de expansión mínima y su alcance mediante el algoritmo de Kruskal, así como el desarrollo de este para la clusterización. Y, todas las relaciones establecidas entre el algoritmo, el análisis clúster y la selección de carcterísticas.

2 Árboles de expansión

Un árbol es denominado, en teoría de grafos (Alvarez Nuñez (2013)), a aquel grafo conexo en el que cualesquiera dos vértices están conectados por exactamente un camino.

Desarrollando este concepto, un árbol de expansión T, de un grafo conexo G, es un subgrafo formado por todos los vértices V y algunas de las aristas A de G de forma que todos los nodos de la red aparecen conectados, pero sin permitir la existencia ciclos. Un grafo puede tener múltiples árboles de expansión. De manera gráfica puede ser ejemplificado del siguiente modo, partiendo del siguiente grafo, se pueden generar distintos árboles de expansión (Graham and Hell (1985)).

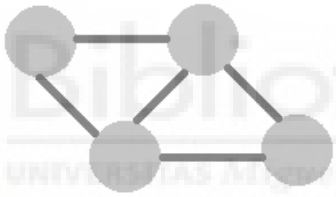

Figure 1: Grafo inicial

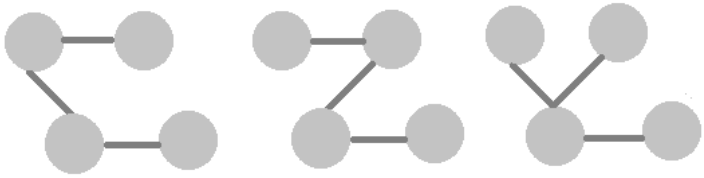

Figure 2: Árboles de expansión generados del grafo anterior

2.1 Árbol de expansión mínima

En ciertos campos de la teoría de grafos surge la necesidad de encontrar determinados modelos de optimización de redes que permitan abordar problemas como el árbol de expansión mínima de un grafo ponderado, lo cual puede

entenderse como coste o distancia, el máximo árbol de expansión, árbol de expansión de mínimo grado y del mismo modo que estos una gran cantidad de casuísticas que pueden derivarse de la búsqueda de la optimización de una determinada red.

El presente trabajo centrará su estudio en el caso concreto del árbol de expansión mínima o árbol recubridor mínimo (minimum spanning tree - MST) y concretamente su alcance mediante el algoritmo de Kruskal.

El algoritmo de Krukal fue publicado por primera vez en 1956 por Joseph B. Kruskal. Este busca enlazar todos los nodos de la red de forma que la longitud total de los arcos sea mínima, la longitud puede adoptar un valor de distancia o de unidad de medida, en función del contexto en el que se desarrolle el problema (Kershenbaum and Van Slyke (1972)).

El funcionamiento del algoritmo es el siguiente: en primer lugar dado un grafo ponderado G=(V,A) con V nodos y A aristas o arcos, el algoritmo parte de G'=(V,vacio), para ello se crea un conjunto T, el cual se inicializa vacío, en el que se almacenarán las aristas del árbol de expansión. Dado que si V es el número de vértices o nodos, la solución óptima debe contener V-1 arcos o aristas. Para ello, se creará un bucle hasta que T contenga n-1 aristas, esto se ha de cumplir ya que debe incluir todos los vértices del grafo de origen y, además, no deben existir ciclos. En dicho bucle se realizará lo siguiente, elegir la arista (v,w) de A con menor longitud, siempre que v y w pertenezcan a distintas componentes conexas, entendiendo por componente conexa a los subgrafos conexos más amplios posibles contenidos dentro de un grafo y que no están estrictamente contenidos en ningún otro, se trata de una partición conexa, y esta misma se borra de A para que no pueda ser seleccionada nuevamente. Para llevar a cabo el algoritmo es necesario ordenar las aristas de mayor a menor peso (Greenberg (1998)). Este algoritmo es de tipo greedy, ya que se basa en una estrategia de búsqueda por la cual se sigue una heurística consistente en elegir la opción óptima en cada paso con la finalidad de alcanzar una solución óptima, concretamente, a cada paso se selcciona la arista de longitud mínima y se añade al subgrafo.

El pseudocódigo del algoritmo es el siguiente:

Algorithm 1 Árbol de expansión mínima algoritmo Kruskal

```
1: procedure KRUSKAL (G)
2:     E(1) = 0
3:     E(2) = Arcos del grafo G
4:     while E(1) < n-1 arcos do
5:         De los arcos de E(2) seleccionar el de menor coste ← e(ij)
6:         E(2) = E(2) − e(ij)
7:         if V(i), V(j) no están en la misma componente conexa then
8:             Unir los nodos V(i) y de V(j)
9:             E(1) = E(1) ∪ {(i, j)}
10:        end if
11:    end while
12:    return E(1)
13: end procedure
```


A través de un ejemplo, es posible ver claramente el funcionamiento del algoritmo. Partiendo de la siguiente matriz de datos de pesos.

	A	B	C	D	E	F	G
A		5					7
B	5		7				13
C		7		9	17	2	
D			9		11		
E			17	11		2	
F			2		2		1
G	7	13					

Table 1: Matriz de datos del grafo

Se obtiene el siguiente grafo.

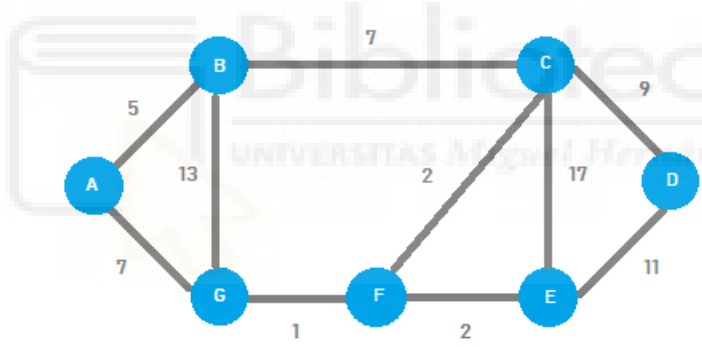

Figure 3: Grafo inicial

A continuación se comienzan a seleccionar las aristas que formarán el árbol de expansión. En este caso concreto es la arista **GF**, marcada en color rojo en la figura 4, la que tiene menor peso y, por tanto, la primera que se selecciona.

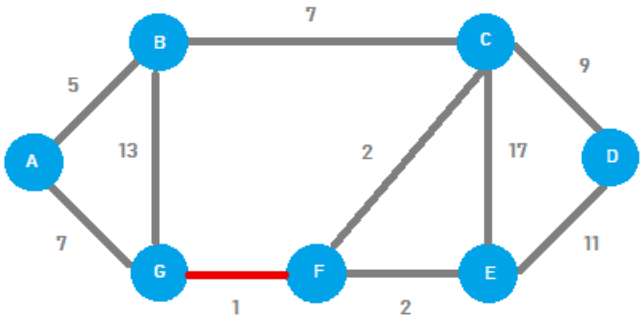

Figure 4: Selección aristas árbol de expansión mínima

A este conjunto de aristas se le suma la siguiente con menor tamaño, al existir un empate entre las aristas que unen los vértices **FE** y **FC** se elige una de forma arbitraria, para este ejemplo la arista **FC**, marcado en la figura 5. De forma paralela se ha de ir comprobando que los nodos pertenezcan a distintas componentes conexas como es el caso de estos ya seleccionados.

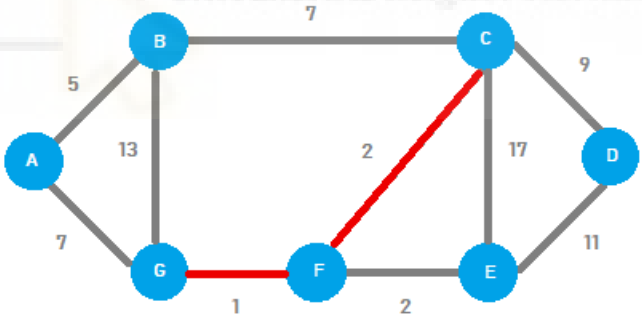

Figure 5: Selección aristas árbol de expansión mínima

Ahora es, sin embargo, la arista **FE**, comentada en el paso anterior y resaltada en la figura 6, la que contiene menor peso y no forma ciclos y que se añade al conjunto de aristas del árbol de expansión.

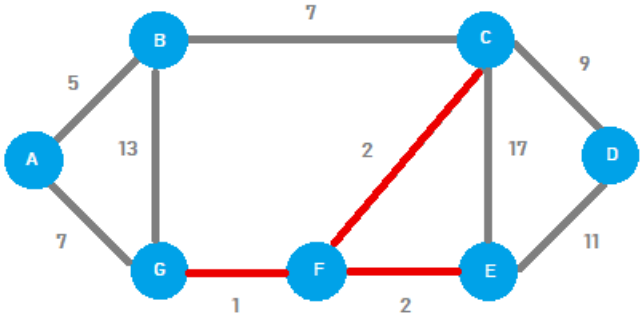

Figure 6: Selección aristas árbol de expansión mínima

La arista **AB** es la siguiente en orden ascendente de pesos y que no supone la formación de ciclos y, es por ello, la siguiente a seleccionar, marcada en la figura 7.

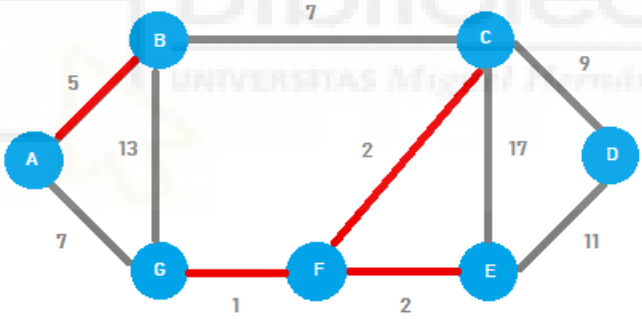

Figure 7: Selección aristas árbol de expansión mínima

De manera similar a la situación encontrada en la figura 5, las siguientes arista en orden de peso son las **AG** y **BC** y, por elección arbitraria, se elige la **AG**, resaltada en la figura 8, que no implica presencia de ciclos.

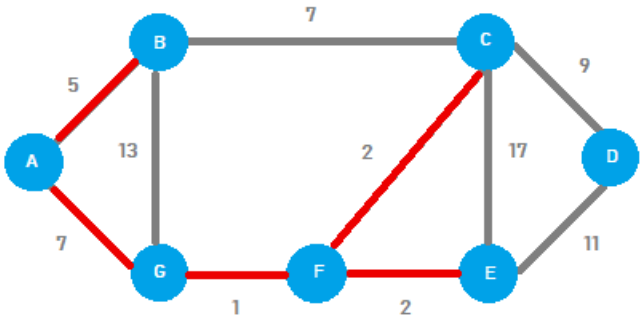

Figure 8: Selección aristas árbol de expansión mínima

La siguiente arista en orden de pesos es la mencionada anteriormente pero finalmente no elegida, la **BC**, pero tampoco será seleccionada ya que los vértices pertenecen a la misma componente conexa, lo que implica la creación de un subgrafo conexo o, dicho de otro modo, la existencia de un ciclo **ABCFG**. Por tanto, la arista elegida será la siguiente en orden de peso que corresponde con la **CD**, marcada en la figura 9.

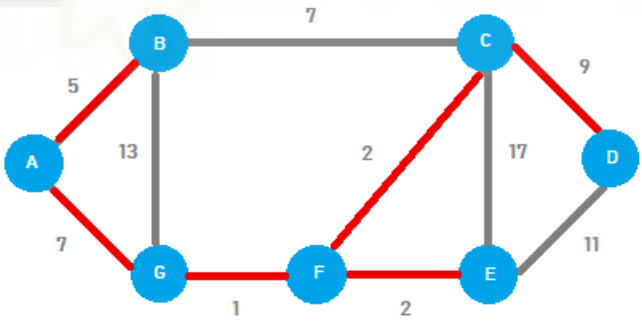

Figure 9: Árbol de expansión mínima

Con este conjunto de aristas resaltadas y comentadas se concluye el proceso, alcanzando un árbol de expansión mínima para el grafo inicial de la figura 3, que permite conectar todos los nodos con ausencia de ciclos. La forma de verificación mencionada anteriormente, la cual indica que para

que un árbol de expansión mínima sea válido ha de contener V-1 aristas, se cumple en este caso, el número de aristas es 6 y el número de nodos es 7.

Señalar que de forma alternativa existen otros algoritmos que también permiten hallar el árbol de expanisión mínima, como son el algoritmo de Prim, el algoritmo del borrador inverso y el algoritmo de Boruvka.

3 Árboles de expansión y dendograma

El algoritmo de Kruskal para la generación de árboles de expansión mínima resulta a su vez una opción interesante para la agrupación de individuos (Madrid (2009)). La sistemática que presenta el algoritmo recuerda a la forma utilizada por los métodos jerárquicos para agrupar individuos. Los métodos jerárquicos se basan en la agrupación de individuos en base a las distancias o similitudes entre cada uno de estos, se efectúa mediante un proceso de aglomeración o división, dependiendo de la dirección en la que se ejecute el algoritmo, de individuos de forma iterativa (Gallardo (2011)). El algoritmo de Kruskal funciona de modo similar al tipo aglomerativo, el cual parte de la suposición de que cada individuo forma un clúster o grupo y a cada paso se van agrupando pares de individuos que cumplen ciertos requisitos, los cuales vienen definidos por el método de unión seleccionado que podrá ser mínima distancia o máxima similitud, distancia máxima o similitud mínima o distancia/similitud promedio ponderada o no ponderada. El que se desarrollará en el presente estudio será el método de mínima distancia que se asemeja con el comportamiento del algoritmo de Kruskal. Al final se establece una jerarquía en la que cada nivel sustenta aquellos individuos similares y, por el contrario, a otros niveles los que presentan más diferencias. La forma natural de representar un proceso de clusterización jerárquico es a través de un dendograma en el que se ilustra el paso a paso de la creación de agrupaciones.

La principal diferencia entre el algoritmo de agrupamiento jerárquico aglomerativo con el método de unión de mínima distancia y el algoritmo de Kruskal para árboles de expansión mínima reside en que el método de clúster jerárquico centra su enfoque en el orden en que se van generando las agrupaciones en cada etapa recurriendo a la representación mediante un dendograma. Y, por otro lado, el algoritmo de Kruskal centra su enfoque en el conjunto de pares de puntos que conforman el árbol de expansión mínima.

El funcionamiento de ambos métodos de una forma breve y esquemática,

para reflejar las similitudes que comparten es el siguiente, para el algoritmo del árbol de expansión mínima de Kruskal:

1. Se parte de una matriz de distancias o similitudes (costes).
2. Se busca la menor de las distancias entre todas, individuos X e Y.
3. Se unen X e Y en un solo clúster y se actualiza la matriz de costes.
4. Se repiten los dos pasos anteriores hasta obtener N-1 uniones.

Y, para el algoritmo de agrupamiento jerárquico aglomerativo con el método de unión de mínima distancia:

1. Se parte de un conjunto de individuos que pertenecen cada uno a su propio clúster.
2. Se seleccionan los dos individuos o clústers que presenten menor distancia.
3. Se unen estos en un solo clúster y se actualiza la matriz de distancias.
4. Se repiten los dos pasos anteriores hasta agrupar en un solo clúester a todos los individuos.

La forma de presentar los resultados, dendograma y árbol de expansión, supone una diferenciación de los algoritmos, pero, como la información que representan es común para ambos, cabe la posibilidad de obtener un dendograma aplicando el algoritmo de Kruskal para el árbol de expansión mínima y, viceversa, obtener una representación en forma de árbol aplicando el método de agrupamiento jerárquico aglomerativo de mínima distancia.

3.1 Desarrollo del algoritmo de Kruskal en lenguaje de programación R

Para poder llevar acabo la agrupación de individuos mediante el algoritmo del árbol de expansión mínima de Kruskal se ha desarrollado este en lenguaje de programación R del siguiente modo.

Para poder implementar el algoritmo de Kruskal es necesaria la definición de funciones que simulen las bases del funcionamiento de este algoritmo, vistas en la sección anterior y, que son, la búsqueda de la menor de las distancias entre dos elementos, la comprobación de pertenencia de los elementos a distintas componentes conexas y, en caso de cumplir las condiciones, la unión de los elementos (Pan *et al.* (2009)).

La forma de trasladar esto a código se basa en la asignación de un valor raíz, en el código presente como *parent*, a cada uno de los vértices, con un valor único para cada cual al comenzar, este reflejará en que árbol o conjunto se encuentra cada vértice. Este valor se actualizará cuando se proceda a la unión de dos elementos que cumplan todos los requisitos, de este cambio de valor se encargará la función denominada *union* en el código, pasando a tomar el valor raíz del primer elemento x, el valor de la raíz del segundo elemento y, dejando ambos elementos con la misma raíz, indicando que se encuentran en la misma componente conexa.

Para la comprobación de la condición de no pertenencia a la misma componente conexa se han creado dos funciones, *find* y *componente*. La primera se encarga de devolver la raíz del árbol al cual pertenece cada uno de los elementos, de modo que hasta que no se alcance la igualdad entre el valor del vértice y el valor de la raíz no detendrá la búsqueda, para todos los casos en los que el árbol está compuesto por múltiples elementos y se hayan ido uniendo de forma jerárquica y sea necesario iterar hasta alcanzar el valor raíz.

La función *componente*, por otro lado, se trata de una función que devuelve un valor booleano, verdadero o falso, tras comprobar si dos elementos pertenecen a la misma componente conexa a través de la observación del valor raíz que indica a que árbol pertenecen, obtenido este a través de la función *find* descrita.

Para la búsqueda de los individuos que presenten la menor distancia, no ha sido necesario el desarrollo de una función, ya que es posible obtener este a partir de las funciones básicas ya existentes, como en este caso recurriendo

a la función *min*.

Una vez definidas todas estas funciones descritas, será posible la agrupación de individuos mediante el algoritmo del árbol de expansión mínima de Kruskal del siguiente modo.

En primer lugar, se parte de una matriz de distancias, la cual puede ser introducida manualmente o calculada a partir de un conjunto de individuos y sus respectivas variables dadas de una base de datos. Para este segundo caso, de forma previa al cálculo de la matriz de distancias basada en la información proporcionada habrá que considerar llevar a cabo una tipificación de los datos con el fin de garantizar que ha sido calculada con datos en una misma escala y, evitar errores por uso de distintas magnitudes. A esta matriz se le denominará matriz de costes.

A continuación, de forma iterativa se recorre la matriz buscando aquellos valores que presenten distancia 0, que serán los elementos de la diagonal principal, los cuales hacen referencia a la distancia de cada elemento consigo mismo, y estos se sustituyen por un valor significativamente elevado, ya predefinido al principio del código, con la finalidad de que estos no afecten en el desarrollo del algoritmo.

Con esta información ya será posible proceder a la ejecución del algoritmo. Se creará un bucle que recorrerá la longitud total de la matriz de coste, es decir, todas las distancias entre todos los individuos a estudiar y, dentro de este, una sentencia condicional *if* que permitirá la ejecución del algoritmo siempre y cuando el conjunto de aristas del árbol de expansión, almacenado en una variable denominada *ne*, en el código, e inicialmente tomando valor cero, sea menor al número de elementos N menos 1, lo que en el código funciona igual que la variable de conteo del número de aristas del árbol de expansión (*ne*) sea menor que N. Mientras se cumpla esta condición, se ejecutará el bloque de código encargado de lo siguiente.

Primero de todo identificar el menor de los elementos de la matriz de costes, que representa la menor distancia entre dos individuos y, a partir de este, definir variables que almacenen esta distancia y los elementos entre la

cual se produce.

En segundo lugar, una sentencia condicional *if* que compruebe que esta distancia seleccionada sea menor al valor que se ha definido al comienzo del código, para asignarle a los elementos de la diagonal principal de la matriz de costes y, asegurar que no se trate de esta y, en caso de cumplir la condición, se comprueba si los elementos pertenecen a la misma componente conexa, para ello se recurre nuevamente a una sentencia *if* que comprueba si el valor devuelto por la función componente es verdadero o falso.

En caso de ser verdadero y que, por tanto, pertenezcan a la misma componente conexa, se detendrá la unión de esos elementos y, comenzará de nuevo el algoritmo a buscar la siguiente menor distancia.

Y, en caso contrario, que no pertenezcan los individuos a la misma componente conexa, se llama a la función *union*, que se encarga de actualizar el valor raíz del primer elemento por el valor raíz del segundo elemento, quedando ambos en el mismo árbol o componente. De manera paralela, el valor *ne* que refleja el conjunto de aristas del árbol de expansión se incrementa en una unidad, a la variable que almacena el coste del árbol de expansión se le agrega la distancia entre los dos elementos unidos y en la posición de la matriz de costes correspondiente a esta distancia se le asigna un valor significativamente alto, para evitar que nuevamente vuelva a ser seleccionada.

En la parte final, se recoge el código que forma la salida del algoritmo, mediante la construcción de una tabla con la información almacenada en el proceso iterativo, se explicará el formato de la salida posteriormente.

Escrito en lenguaje de programación R, presenta la siguiente forma.

```
#PARAMETROS INICIALES
mincost=0;a=1;b=1;u=1;v=1;;ne=0;VAL=999999;min=VAL

#FUNCIONES
find <- function(x) {
if(x==parent[x]){x}
```

```
else{find(parent[x])}}

componente <- function(x,y) {
if(find( x ) == find( y )){TRUE}
else{FALSE}}

union <- function(x,y){
xr = find(x)
yr = find(y)
parent[xr] <- yr}

for(i in 1:ncol(cost)){
for(j in 1:nrow(cost)){
if(cost[i,j]==0){
cost[i,j]=VAL}}}

parent<-c()
for( i in 1:N){
parent[i]=i}

#VARIABLES VACIAS PARA ALMACENAMIENTO RESULTADOS
raices<-c();raices<-c();raices_ind<-c();elem<-c()
iteracion<-c();num_ar<-c();elem_a<-c()
elem_b<-c();dist<-c();componentes<-c()

#PREPARACIÓN RESULTADOS
for(i in 1:length(parent)){elem[i]<-paste('Ind',i)}
columnas<-c('Iteración','Nº aristas','Elemento a','Elemento b',
'Distancia','Componentes ',elem)

#ALGORITMO
```

```
for (i in 1:(length(cost))){
if(ne < N){
menor<-which(cost == min(cost), arr.ind = TRUE)
a=u=as.numeric(menor[1,1])
b=v=as.numeric(menor[1,2])
min=cost[a,b]

if(min < VAL){
if(componente(u,v)==FALSE){
union(u,v)
ne=ne+1
mincost = mincost+min
iteracion[i]<-i;num_ar[i]<-ne;elem_a[i]<-a;elem_b[i]<-b;dist[i]<-min
for(j in 1:length(parent)){raices_ind[j]<-find(parent[j]}
componentes[i]<-length(unique(raices_ind))
raices<-c(raices,raices_ind)}

cost[a,b]=cost[b,a]=999999}
iteracion<-iteracion[!is.na(iteracion)]
num_ar<-num_ar[!is.na(num_ar)];elem_a<-elem_a[!is.na(elem_a)
elem_b<-elem_b[!is.na(elem_b)];dist<-dist[!is.na(dist)]
componentes<-componentes[!is.na(componentes)]
tabla<-data.frame(iteracion,num_ar,elem_a,elem_b,dist,componentes)
individuos<-data.frame(matrix(unlist(raices),nrow=nrow(tabla),byrow=T))

tabla_final<-cbind(tabla,individuos)}}

colnames(tabla_final)<-columnas
print(noquote(paste('Coste total árbol de expansión:',mincost)))
```

La salida que proporcionará el algoritmo planteado será la siguiente, se

mostrará una tabla con una sección llamada resumen en la que se muestran a nivel iteración (solo para aquellas iteraciones en las que se haya producido una unión): la iteración en la que se ha producido la unión, se mostrará el valor de la variable que lleva el conteo del número de aristas del árbol de expansión mínima, los elementos que han sido unidos, la distancia existente entre estos dos y el número de componentes conexas distintas existentes en cada momento, es decir, en cuantas componentes distintas se encuentran los individuos en cada paso, este dato se obtiene observando el número de raíces distintas a las que pertenecen los individuos. Este último valor, será de gran utilidad a la hora de identificar como quedan los diferentes grupos una vez decidida la distancia, se explicará detalladamente en el próximo apartado. Por otro lado, la sección llamada raíz muestra el detalle a nivel individuo e iteración actualizado de la raíz, la cual permite conocer en que árbol o componente se encuentra cada uno de los individuos en cada momento y apreciar el paso a paso de como van variando estos.

Con la información disponible, una forma gráfica que ayude a sintetizar toda la información y sea útil y clara para tomar decisiones de agrupación es el dendograma, igual al que se obtiene con los métodos jerárquicos, por las grandes similitudes que comparten ambos métodos y el cual puede ser construido con la información proporcionada como salida del algoritmo. Otra opción que puede plantearse como representación gráfica del resultado es recurrir a la construcción de un árbol, la salida habitual del algoritmo de Kruskal, pero a pesar de ser muy similares, para este objetivo de agrupar resulta ser una opción más clara el dendograma.

3.2 Agrupación de individuos en base al algoritmo de agrupamiento jerárquico aglomerativo y de Kruskal

Una vez ejecutado el algoritmo de Kruskal sobre un conjunto de individuos y obtenido el paso a paso del proceso de unión de los distintos individuos, hasta alcanzar el árbol de expansión mínima, la decisión del número de grupos a distinguir sobre un conjunto de individuos vendrá dada en base a una

distancia D a seleccionar. A partir de esta información existe una multitud de formas tanto gráficas como no gráficas de ver como se forman los grupos, una de ellas, que resulta bastante simple partiendo de la salida obtenida en el código previamente explicado consiste en la observación de la segunda sección comentada, raíz, la cual indica a que árbol o componente pertenece cada uno de los individuos, para el valor de distancia dado, es decir, para la distancia x dada, en la cual se habrá producido una unión, como quedan los individuos agrupados en distintas componentes o, lo que resulta equivalente, clústeres. A su vez, se puede recurrir a la variable componentes del apartado de resumen para conocer directamente la cantidad de grupos que se formarán a ese nivel. De forma que, escogida una distancia D, habrá que identificar la inmediata anterior en la que se ha producido una unión y, observar cómo queda la distribución de los individuos en los distintos grupos o clústeres. Estos serán los grupos creados.

De forma más ilustrativa se representará este proceso de agrupación a través de un ejemplo, basado en la información que muestra como salida el algoritmo. Partiendo de una pequeña muestra de cinco individuos y su respectiva información atendiendo a dos variables, extraído de una base de datos, al aplicar el algoritmo de Kruskal, obtenemos la siguiente salida:

Resumen					
Iteración	Nº aristas	Elemento a	Elemento b	Distancia	Componentes
1	1	4	3	450.02	4
2	2	5	2	4800.00	3
3	3	5	1	12000.00	2
5	4	4	2	18300.00	1

	Raiz				
Iteración	Ind 1	Ind 2	Ind 3	Ind 4	Ind 5
1	1	2	3	3	5
2	1	2	3	3	2
3	1	1	3	3	1
5	1	1	1	1	1

Table 2: Salida algoritmo de Kruskal

Para proceder a agrupar, definiendo una distancia D de 5000, habría que detenerse en la iteración 2, por ser la inmediata inferior a esta y, los elementos quedarían agrupados en un total de tres grupos distribuidos tal que así, primer clúster con el individuo 1, segundo clúster los individuos (2,5), y tercer clúster los individuos (3,4). Si se selecciona una distancia D de 15000, la inmediata distancia de unión anterior es la correspondiente a la iteración 3 y, fijándose en el resultado de esta, los grupos formados serían dos, primer clúster individuos (1,2,5) y un segundo grupo con los individuos (3,4).

De otra forma, a modo comparativo, esta decisión de agrupación de individuos basada en la ejecución del algoritmo de agrupamiento jerárquico aglomerativo con el método de unión de mínima distancia y su correspondiente dendograma obtenido, será similar, partiendo de una distancia D elegida y de la observación del dendograma, se procederá a realizar lo que se conoce como corte del dendograma, que es la forma visual de ver como quedan repartidos los individuos, trazando una línea horizontal a la altura de la distancia D y viendo dónde se produce el corte y, por tanto, que grupos se generan.

Recurriendo al mismo conjunto de datos con el que ha sido ejemplificada la agrupación mediante el algoritmo de Kruskal, el dendograma que se obtiene es el siguiente.

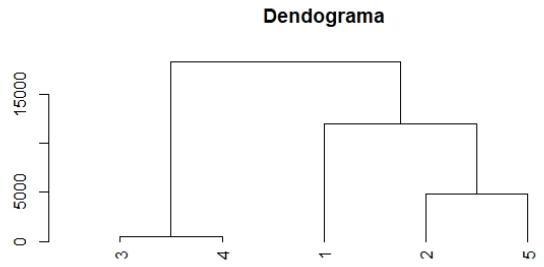

Figure 10: Dendograma obtenido a través del algoritmo de agrupamiento jerárquico aglomerativo con el método de unión de mínima distancia

Para proceder a agruparlos, asignamos los valores previamente escogidos para la distancia D, 5000 en primer lugar y 15000 a posteriori. Y, tal y como se ha mencionado previamente bastaría con trazar una línea horizontal en ambas alturas, ver dónde "cortan" y que conjuntos de elementos se crean. El resultado de forma visual quedaría del siguiente modo.

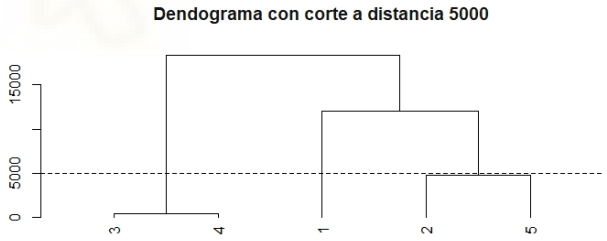

Figure 11: Agrupación con D 5000

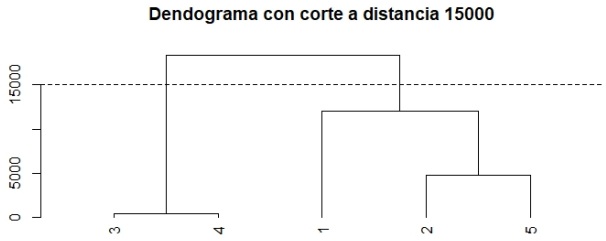

Figure 12: Agrupación con D 15000

Dando lugar a la misma cantidad de grupos diferenciados con el algoritmo de Kruskal, tres grupos para la distancia de 5000, el individuo 1, los individuos (2,5) y el (3,4). Del mismo modo sucede con la distancia de 15000, con la que se obtiene el mismo número de grupos que con el algoritmo desarrollado, dos grupos con los elementos (1,2,5) en un lado y los elementos (3,4) en otro grupo.

4 Técnicas de agrupación de individuos y la selección de características

La cantidad de información, cada vez mayor, de la que se dispone en la sociedad actual supone un reto a la hora de abordar tareas con datos de forma eficiente y satisfactoria, dando lugar a técnicas de preprocesamiento de datos como es la selección de características.

La selección de características es una técnica que se centra en la búsqueda del subconjunto de variables o características de una base de datos que resultan más relevantes (Herrera and Cano (2006)). Esto se lleva a cabo de manera previa a la aplicación de técnicas estadísticas que permitan determinar patrones de comportamiento entre los datos, realizar predicciones, agrupar individuos, etcétera. Evitando problemas de overfitting o el problema conocido como maldición de dimensión, así como conseguir una simplificación de los modelos y un menor tiempo de entrenamiento de estos.

A la hora de abordar una agrupación de individuos, basado por ejemplo en los métodos/algoritmos previamente desarrollados, es de vital importancia y, generará un impacto directo en el resultado, el conjunto de características seleccionadas. Pudiendo obtenerse de un mismo conjunto de individuos tantos resultados de agrupación como combinaciones posibles de las características disponibles, obteniendo así el subconjunto de características más relevante para agrupar a los individuos.

A su vez, las técnicas de agrupación de individuos pueden resultar de gran utilidad para llevar a cabo la selección de características. La forma de llevarlo a cabo sería realizar una comparativa entre los resultados obtenidos, para un número de clusters fijado, en base a las distintas combinaciones posibles con las características disponibles de la base de datos, de modo que el subconjunto de características que mejor agrupe a los individuos será aquel que presente una menor distancia para el número de clusters fijado.

4.1 Caso práctico con base de datos "empleados"

El objetivo de este apartado es, a través de una ejemplificación, poder ver el impacto y la importancia de la selección de características en la agrupación de individuos, así como poder llevar a cabo esta selección de características más relevantes a través de los resultados obtenidos, aplicando las técnicas y algoritmos de agrupación previamente detallados. Para conseguirlo se analizarán los resultados obtenidos de la aplicación de los algoritmos de agrupación a un conjunto de individuos, pero atendiendo a distintas combinaciones de características disponibles.

Se han seleccionado 20 individuos que serán objeto de estudio, de una base de datos denominada empleados la cual recoge información, tanto cuantitativa como cualitativa, acerca de su vida laboral. Las características que se estudiarán son las siguientes:

- C1: sexo
- C2: nivel educativo
- C3: categoría laboral
- C4: salario actual
- C5: salario inicial
- C6: tiempo contratado (contabilizado en meses)
- C7: experiencia previa (contabilizada en meses)
- C8: clasificación étnica

La agrupación de individuos, cuyos resultados permitan ilustrar el objetivo mencionado al comienzo del apartado, se realizará atendiendo a combinaciones de dos o tres características de entre las disponibles. Entre todas las combinaciones posibles de dos en dos y de tres en tres se seleccionarán aleatoriamente doce de forma que sea posible obtener una conclusión significativa.

Para la implementación del algoritmo de Kruskal, previamente desarrollado, se parte de la situación de información proveniente de una base de datos, sobre la cual habrá que calcular la matriz de distancias para las características que nos interesen en cada momento. Para el cálculo de esta será necesario en primer lugar realizar de forma previa una estandarización de las variables o características para poder combinarlas, debido a que hay presencia de variables tanto cuantitativas como cualitativas y, en diversas escalas numéricas. Una vez estandarizadas, a través de la función *scale* de R, con la que se consiguen todas las variables en una misma escala numérica, restando la media y dividiendo entre la varianza, se procede a obtener esta matriz de distancias a través de la función *dist*, seleccionando como método la distancia euclídea.

La comparación se llevará a cabo para un número fijado de tres grupos de individuos. El valor a comparar será la distancia mínima a partir de la cual se pueden distinguir tres grupos entre los individuos, basado en la información obtenida tras aplicar el algoritmo de Kruskal. Los resultados obtenidos de este ejemplo de estudio se recogen en la siguiente tabla, la cual indica en la primera columna la combinación de características analizada, y en la segunda columna la distancia mínima a partir de la cual se pueden diferencias los tres grupos.

Combinación características	Distancia
salario inicial - tiempo contratado (c5 y c6)	1.258
sexo - categoría laboral (c1 y c3)	1.989
sexo - categoría laboral - clasificación étnica (c1, c3 y c8)	2.729
nivel educativo - salario actual - exp previa (c2, c4 y c7)	1.753
tiempo contratado - exp previa (c6 y c7)	0.603
categoría laboral - salario actual (c3 y c4)	0.228
sexo - nivel educativo - categoría laboral (c1, c2 y c3)	1.641
tiempo contratado - exp previa - salario actual (c6, c7 y c4)	1.536
salario inicial - exp previa (c5 y c7)	0.843
tiempo contr - exp previa - clasificación étnica (c6, c7 y c8)	1.530
nivel educativo - clasificación étnica (c2 y c8)	2.729
cat laboral - salario inicial - clasificación étnica (c3, c5 y c8)	0.535

Table 3: Resultados obtenidos mediante algoritmo Kruskal (distancia mínima para distinción 3 grupos)

Mediante la observación de los distintos resultados obtenidos en base a las distintas combinaciones de características estudiadas, para esta pequeña muestra de 20 individuos y ocho características, se confirma el impacto que supone la selección de unas u otras características, consiguiendo una agrupación de los mismos individuos a distintos niveles de distancias, lo que implica una presencia de grupos más homogéneos que otros en función de una menor o una mayor distancia. Señalar que no es posible realizar una interpretación de las distancias y las características ya que han sido previamente escaladas para poder trabajar con ellas de manera conjunta.

A su vez, por otro lado, este análisis puede permitir realizar una selección de características, siendo las más relevantes de cara a llevar a cabo una agrupación aquellas que consigan agrupar los individuos en tres grupos, en este caso, con la menor de las distancias lo que se traduce en una mayor homogeneidad dentro de los grupos. Son en este caso las características

categoría laboral y salario actual (c3 y c4) las más relevantes por presentar la menor de las distancias (0.228), a partir de la cual es posible agrupar a los individuos en tres grupos, se presentará, a continuación, el detalle de los resultados para este caso en concreto comparado con el detalle del resultado de una de las combinaciones de características que presenta una mayor distancia, sexo, categoría laboral y clasificación étnica. También si, es posible, se puede alanzar en ocasiones una reducción de una colección de variables, como es el caso entre el conjunto de características (c1, c3 y c8) y el conjunto de características (c1 y c3) se elige el de dos variables por presentar una menor distancia, consiguiendo una reducción. Por otro lado, entre el conjunto de características (c6, c7 y c8) y el conjunto (c2 y c8), se elige el de tres variables, por tener un menor valor de distancia y, no se consigue, por tanto, una reducción.

Cabe señalar que este proceso de selección de características está planteado a modo representativo ya que para alcanzar un resultado completo sería necesario estudiar todas las combinaciones posibles, a pesar de que se ha analizado un volumen considerable para que el resultado sea significativo.

Analizando el detalle de la salida que se obtiene al proceder a agrupar a los individuos con la cobinación escogida como mejor, presente en la siguiente tabla, es posible apreciar el paso a paso de las uniones y como es posible distinguir tres grupos a partir de una distancia de 0.23.

Resumen					
Iteración	Nº aristas	Elemento a	Elemento b	Distancia	Componentes
1	1	13	9	0.01	19
2	2	4	3	0.02	18
3	3	8	3	0.02	17
4	4	12	9	0.02	16
5	5	15	13	0.02	15
6	6	16	2	0.03	14
9	7	14	7	0.05	13
10	8	17	5	0.05	12
11	9	20	15	0.06	11
14	10	19	16	0.08	10
16	11	11	6	0.10	9
17	12	12	11	0.11	8
18	13	10	4	0.11	7
22	14	20	10	0.12	6
26	15	19	5	0.15	5
27	16	14	6	0.16	4
36	17	7	2	0.23	3
153	18	18	1	2.54	2
154	19	17	1	3.30	1

Table 4: Salida del algoritmo de Kruskal para agrupación con características categoría laboral y salario actual

Por otro lado, de forma gráfica, a través de un dendograma la agrupación con una marca en la distancia mínima para diferenciar tres grupos en el caso de la combinación escogida como mejor y de la combinación escogida como peor, se muestra en las dos siguientes figuras. Gráficamente resulta de forma sencilla apreciar como para la formación de estos grupos para un mismo conjunto de individuos hay una clara diferencia de alturas de distancias.

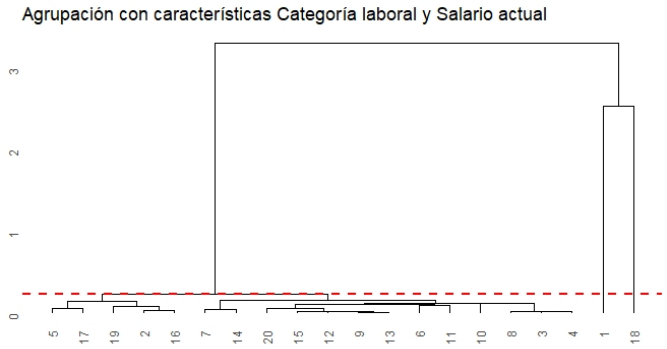

Figure 13: Dendograma con marca en distancia mínima formación 3 grupos

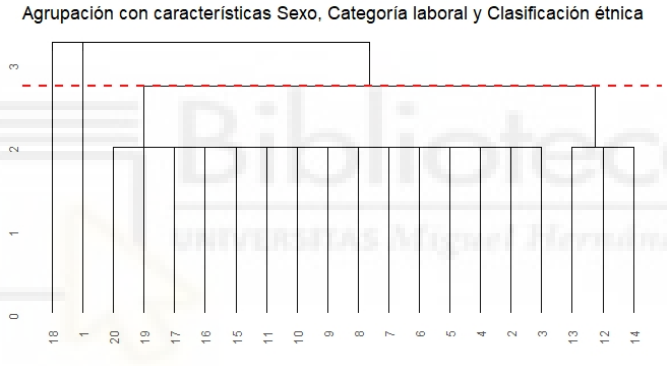

Figure 14: Dendograma con marca en distancia mínima formación 3 grupos

El código de lenguaje de programación R que recoge todo el procedimiento del análisis obtenido en este apartado de ejemplificación aparece indexado en el anexo.

5 Conclusiones

El fin de esta sección es sintetizar y destacar las conclusiones obtenidas como respuesta a los objetivos planteados al principio del proyecto. Se plantean también en este apartado futuras líneas de estudio que pueden resultar de gran interés para continuar la investigación en la línea de trabajo abierta.

El proyecto presentaba un doble objetivo, por un lado, ser capaces de llevar a cabo una agrupación de individuos basada en el algoritmo de Kruskal. Para ello una vez explicado y contextualizado el algoritmo de Kruskal con su fin natural de obtención de árboles de expansión mínima. Se ha construido un código que, basado en las bases de este algoritmo, sea capaz de llevar a cabo una agrupación de forma muy similar a los métodos jerárquicos. Una vez conseguido este código y una salida óptima que permita su uso con este fin, se ha mostrado su eficacia demostrando su funcionamiento con una pequeña muestra de datos y garantizando la veracidad de su resultado obteniendo la misma información que con la agrupación mediante métodos jerárquicos aglomerativos con unión basada mínima distancia.

Por otro lado, se pretendía demostrar la importancia de la selección de características a la hora de plantear una agrupación de individuos, para ello con el ejercicio planteado, de comparación de distancias mínimas para agrupar una muestra de 20 individuos de una base de datos dada en tres clústers, se obtiene una doble conclusión, por un lado, se reafirma el gran impacto que supone la elección de unas u otras características a la hora de agrupar a los individuos, observando cómo hay variaciones significativas entre la distancia mínima para diferenciar tres grupos en función de las características seleccionadas. Y, por otro lado, fruto de la comparativa llevada a cabo para demostrar el efecto de la selección, se consigue simular un proceso de selección de características, el cual aporta un resultado estimado debido a que solo se ha analizado una muestra relevante de combinaciones entre todas las posibles. Concluyendo con este, que las variables categoría laboral y salario actual, de entre la base de datos utilizada, son las más relevantes y que consiguen agrupar a la muestra en grupos más homogéneos.

5.1 Líneas futuras del estudio

En este último apartado se plantea una línea de trabajo que es posible desarrollar en base a los análisis llevados a cabo en el proyecto y que surge como conclusión de estos y podría resultar de interés.

Esta oportunidad parte del ejercicio planteado para demostrar la importancia de la selección de características para la agrupación de individuos, mediante la cual se plantea una reflexión a través de la comparación de las formas de agrupar un mismo conjunto de individuos, pero atendiendo a distintas combinaciones de variables. La oportunidad que, por tanto, surge aquí es la opción de optimizar un proceso mediante el cual se produjese una comparación de todas las combinaciones posibles con las características de cierta base de datos de manera que se consiguiese obtener un preprocesamiento de los datos para seleccionar las características más relevantes. Esta decisión, de que combinación sería la más adecuada viene dada mediante la observación de valor de la distancia a la cual se consigue diferenciar un determinado número de grupos, siendo el menor el óptimo, consiguiendo así grupos más homogéneos y que presenten un valor de la suma de cuadros dentro de los grupos menor.

Referencias

Marcelino Felipe Alvarez Nuñez. Teoría de grafos. 2013.

J Gallardo. Métodos jerárquicos de análisis cluster. *Disponible desde Internet en: www. ugr. es/˜ gallardo/pdf/cluster-3. pdf.(Con acceso el 22/02/2017)*, 2011.

Ronald L Graham and Pavol Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.

Harvey J Greenberg. Greedy algorithms for minimum spanning tree. *University of Colorado at Denver*, 1998.

Francisco Herrera and JR Cano. Técnicas de reducción de datos en kdd. el uso de algoritmos evolutivos para la selección de instancias. *Actas del I Seminario sobre Sistemas Inteligentes (SSI_06)*, pages 165–181, 2006.

Aaron Kershenbaum and Richard Van Slyke. Computing minimum spanning trees efficiently. In *Proceedings of the ACM annual conference-Volume 1*, pages 518–527, 1972.

UCI d Madrid. Análisis de cluster y arboles de clasificación. *Universidad Carlos III, Madrid, España*, 2009.

Da-Zhi Pan, Zhi-Bin Liu, Xian-Feng Ding, and Qin Zheng. The application of union-find sets in kruskal algorithm. In *2009 International Conference on Artificial Intelligence and Computational Intelligence*, volume 2, pages 159–162. IEEE, 2009.

Anexo

```
#Paráméros iniciales
mincost=0
a=1;b=1;u=1;v=1;
ne=0
VAL=999999;min=VAL

#Funciones
find <- function(x) {
if(x==parent[x]){
x}
else{
find(parent[x])}}

componente <- function(x,y) {
if(find( x ) == find( y )){
TRUE}
else{
FALSE}}

union <- function(x,y){
xr = find(x)
yr = find(y)
parent[xr] <<- yr}

#Lectura de la base de datos
#EMPLEADOS <- read_sav("EMPLEADOS.sav")

#Selección del bote de individuos a analizar y de las características
(esta parte del código es la que se irávariando de un caso a otro
para elegir las características)
```

```
individuos<-EMPLEADOS[1:20,c(5,7,10)]

#Estandarización de los datos
individuos_std<- scale(individuos,center=T,scale=T)

#Obtención y preparación de la matriz de costes
cost <-dist(individuos_std, method = "euclidean",upper=TRUE,diag=TRUE)
cost <- as.matrix(cost)

for(i in 1:ncol(cost)){
for(j in 1:nrow(cost)){
if(cost[i,j]==0){
cost[i,j]=VAL}
}
}
N<-sqrt(length(cost))

parent<-c()
for( i in 1:N){
parent[i]=i
}

#Agrupación mediante dendograma(basado método jerárquico
aglomerativo con unión basada en distancia mínima)
por si se quiere recurrir a una comparación
library('dplyr');library('ggplot2')
hc<-hclust(d = as.dist(cost), method = "single")
hc<-as.dendrogram(hc)
ggdendrogram(hc) + geom_hline(yintercept=0.228,
 color="red",linetype=2, size=1)
```

```
#Creación de variables vacias para almacenamiento de resultados
raices<-c();raices_ind<-c();elem<-c();iteracion<-c()
num_ar<-c();elem_a<-c();elem_b<-c()
dist<-c();componentes<-c()

#Preparación tabla resultados
for(i in 1:length(parent)){elem[i]<-paste('Ind',i)}
columnas<-c('Iteración','Nº aristas','Elemento a','Elemento b',
'Distancia','Componentes ',elem)

#Algoritmo
for (i in 1:(length(cost))){
if(ne < N){
menor<-which(cost == min(cost), arr.ind = TRUE)
a=u=as.numeric(menor[1,1])
b=v=as.numeric(menor[1,2])
min=cost[a,b]

if(min < VAL){
if(componente(u,v)==FALSE){
union(u,v)
ne=ne+1
mincost = mincost+min
iteracion[i]<-i
num_ar[i]<-ne
elem_a[i]<-a
elem_b[i]<-b
dist[i]<-min
for(j in 1:length(parent)){
raices_ind[j]<-find(parent[j])}
```

```
componentes[i]<-length(unique(raices_ind))
raices<-c(raices,raices_ind)}
cost[a,b]=cost[b,a]=999999}

#Construcción tabla resultados
iteracion<-iteracion[!is.na(iteracion)]
num_ar<-num_ar[!is.na(num_ar)]
elem_a<-elem_a[!is.na(elem_a)]
elem_b<-elem_b[!is.na(elem_b)]
dist<-dist[!is.na(dist)]
componentes<-componentes[!is.na(componentes)]
tabla<-data.frame(iteracion,num_ar,elem_a,elem_b,dist,componentes)
individuos<-data.frame(matrix(unlist(raices),nrow=nrow(tabla),byrow=T))

tabla_final<-cbind(tabla,individuos)}
}
colnames(tabla_final)<-columnas
tabla_final
print(noquote(paste('Coste total árbol de expansión:',mincost)))

library('kableExtra')
kable(tabla_final,format='latex',digits=2)

kable(tabla_final[,1:6],format='latex',digits=2) %>%
add_header_above(c("Resumen" = 6)) %>%
kable_styling(latex_options = c("repeat_header"))

kable(tabla_final[,c(1,7:(6+N))],format='latex',digits=2) %>%
add_header_above(c("" ,"Raiz" = N)) %>%
kable_styling(latex_options = c("repeat_header"))
```