Fernando Ferri

MBD – Lab 3 (CLUSTERING)

Descripción

El Grupo *Electronic-Media* es una de las empresas de comercio electrónico más grandes del mundo, con filiales en más de 20 países, incluyendo EE.UU., Alemania y Francia. Esta empresa vende millones de productos en todo el mundo y centenares de productos se incorporan a su catálogo diariamente.

Un análisis coherente del rendimiento de los productos es crucial. Sin embargo, debido a la dispar infraestructura global, muchos productos idénticos llegan a clasificarse de forma distinta. En consecuencia, la calidad del análisis del producto depende en gran medida de la capacidad de agrupar con precisión productos similares. Cuanto mejor sea la clasificación, mayor conocimiento se podrá generar sobre nuestra gama de productos.

El objetivo de esta práctica es:

1. **Generar una clasificación** de los productos procurando mantener las máximas similitudes entre grupos y la máxima heterogeneidad entre los mismos. Para llevarlo a cabo se deberán usar técnicas de *clustering (*k-means*)*→Sólo usar datos de entrenamiento

2. **Construir un modelo predictivo** que sea capaz de distinguir entre las principales categorías de productos existentes (en total, nueve categorías). Se deberán usar *conditional trees* para realizar la predicción→Usar datos de entrenamiento y test

Datos

Se entregaran 2 conjuntos de datos:

1. Datos de entrenamiento. Contendrá la variable respuesta (categoría actual del producto)

2. Datos test. No contendrán la variable respuesta y son sobre los que se hará la predicción de la segunda parte.

Las variables comunes que contienen ambos juegos son:

- *id*: identificador del producto
- *feat_1* a *feat_93*: características del producto
- *target* (entrenamiento): categoría al que pertenece el producto en cuestión.

A) Primera parte: Algoritmos no supervisados

A.1) Evaluación

Evaluación de la primera parte se basará en el estadístico de Inercia (variabilidad entre grupos dividida entre la variabilidad total):

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_j - \mu_i||^2)$$

Se obtendrá una puntuación más alta cuanto menor sea el estadístico con un número razonable de grupos. (Con un número infinito de grupos se consigue una inercia de 1)

A.2) Entrega para la primera parte:

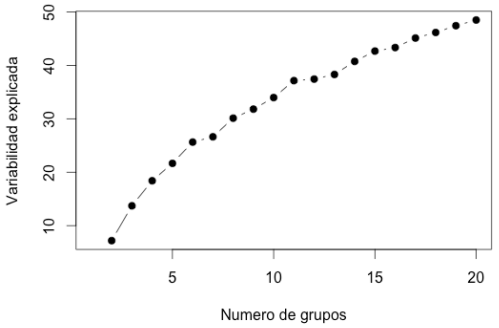
1. Una descripción del trabajo realizado y el proceso seguido para llegar a las conclusiones. Así mismo, debe especificarse el número de grupos idóneo y el valor de la inercia encontrado. Se valorará la presencia de gráficos que faciliten la comprensión. Extensión: 3 o 4 páginas. Formato: *.docx*

2. Fichero de texto con la clasificación final de los productos con dos columnas separadas por una coma: la primera debe contener el identificador del producto y la segunda el identificador del grupo al que es asignado (no tiene que tener ninguna relación con el grupo real). Formato: *.txt*

3. El código utilizado para realizar esta parte con comentarios explicativos. Formato: *. R*

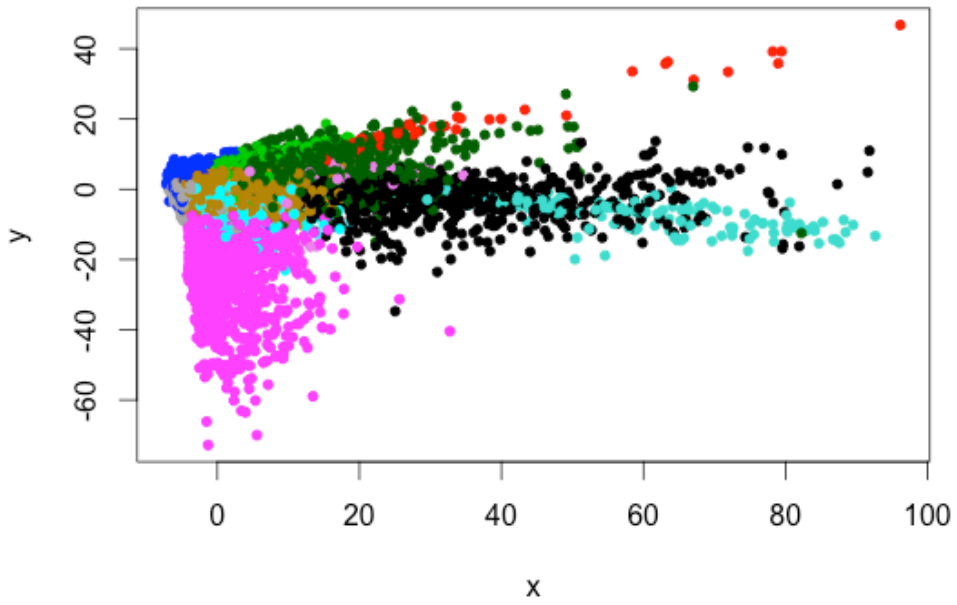
A.3) Solución:

```
##-- 1. Borramos todo lo que hay en memoria y leemos los datos de entrenamiento
rm(list=ls())
setwd('/Users/Fer/Development/MASTERBD/ESTADISTICA/E_CLUSTERING/practica/')
datos2 <- read.csv('p3_train.csv',sep=";")

##-- 2. Miramos los 100 primeros registros, haz la descriptiva de todas las variables y eliminamos las variables "id" y "target"
View(head(datos2,100))
summary(datos2)

#guardamos los ids de los productos antes de eliminarlos para cuando queramos guardar en que grupo ha sido clasificado cada producto
myIdsList <-datos2$id

datos2$id <- NULL
datos2$target <- NULL

################################################################
# Algoritmo de K-means
################################################################
##-- 3. Ponemos una semilla (un numero cualquiera) para siempre obtener los mismos resultados a partir de aqui
set.seed(12345)

##-- 4. Ejemplo de k-means. Usamos el algoritmo de k-means para hacer 2 grupos con 10 puntos de inicio
km2 <- kmeans(datos2,centers=2,nstart=10)

#Evaluamos la calidad del clustering con 2 grupos /clusters
inercia_km2 <- km2$tot.withinss  # Variabilidad intra-grupo (Inercia)
inercia_km2 #->21384032 a menor variabilidad intra-grupo más homogeneos son los grupos, mejor agrupación.

#EVALUACION – Variabilidad explicada (variabilidad entre grupos / total):
variabilidad_Explicada_km2 <-100*km2$betweenss/km2$totss
variabilidad_Explicada_km2 # 7.267007 % variabilidad explicada. Variabilidad entre grupos / variabilidad total

##-- 5. Calculamos el % de variabilidad explicada dividiendo la variabilidad entre grupos entre la variabilidad total
(I <-100*km2$betweenss/km2$totss)

##-- 6. Análisis de la variabilidad explicada para escoger el numero de grupo óptimo. Test de 2 a 20  grupos/clusters
set.seed(12345)
I <- c()                          # Vector donde se guardaran los porcentajes de variabilidad
for(g in 2:20){
  km <- kmeans(datos2,centers=g,nstart=5)  # grupos con 5 puntos de inicio
  I[g] <-100*km$betweenss/km$totss            # Guardar en vector el % de Variabilidad explicada
  print(g)                        # Printar progreso
}

##-- 7. Hacemos el grafico de las variabilidades explicadas en función del numero de grupos y decidimos con cuantos nos quedamos
plot(I, type="b", pch=19, xlab="Numero de grupos", ylab="Variabilidad explicada")
```



Solución: Elegimos 11 grupos siguiendo la regla "del codo", en la que parece que a partir de 11 grupos la variabilidad explicada aumenta mucho menos.

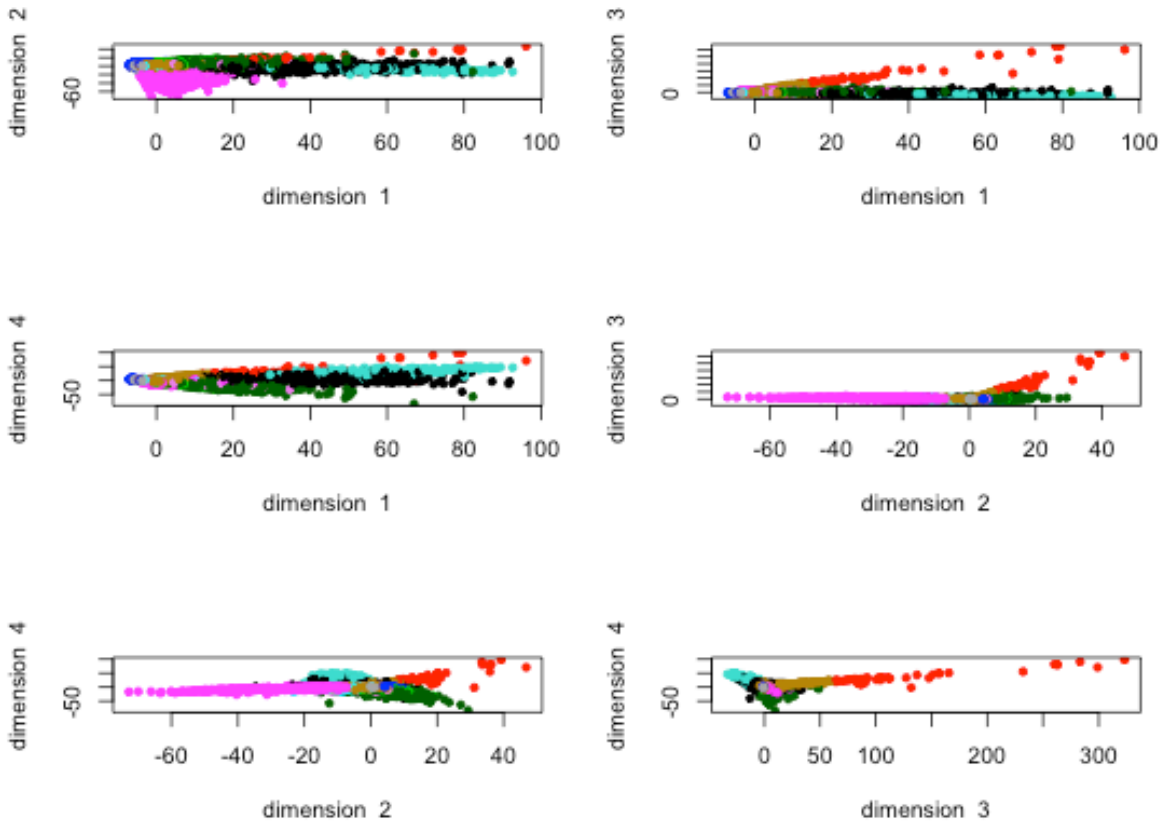
```
##-- 8. Para el numero de grupos escogido, ejecutamos el k-means con 10 puntos de inicio

set.seed(12345)                          # semilla
km.def <- kmeans(datos2,centers=11,nstart=10)  # aplicar kmeans para el numero escogido de grupos


##-- 9. Creamos una paleta con suficientes colores para representar los datos (tantos colores como grupos)
colors <- c("black","red","green3","blue","cyan","magenta","darkgray",
          "darkgoldenrod","darkgreen","violet","turquoise")

palette(colors)

##-- 10. La instruccion princomp realiza un analisis de componentes principales que reduce las dimensiones del conjunto
##-- de datos original. Representamos las 2 primeras dimensiones (scores) obtenidas con colores para los clusters
pc <- princomp(datos2)
par(mfrow=c(1,1))
x <- pc$scores[,1]                # Primera coordenada
y <- pc$scores[,2]                # Segunda coordenada
colores <- km.def$cluster                 # Haz names(km) y mira que objeto guarda el identificador de los grupos para ponerlo como color
plot(x,y,pch=19,col=km.def$cluster,cex=0.7) # cex=0.7 indica el tamanyo de los puntos
```



Así pues, tras representar el agrupamiento en clusters de las observaciones vemos que los grupos quedan bastante diferenciados en las dos primeras dimensiones.

##-- 10. Para finalizar el análisis visual de los clusters obtenidos representa todas las combinaciones de parejas de las 4 primeras dimensiones usando un bucle.
##-- (1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4). Marcamos los grupos con colores

```
par(mfrow=c(3,2))                # 6 ventanas en una
for(i in 1:3){
 for(j in (i+1):4){
  plot(pc$scores[,i], pc$scores[,j], pch=19, col=km.def$cluster, cex=0.7, xlab=paste("dimension ",i), ylab=paste("dimension ",j))
 }
}
```

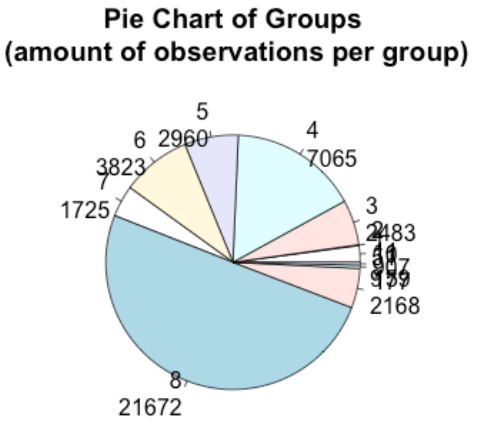


En el análisis bidimensional para cada combinación de dos dimensiones se observan diferentes perspectivas de los clusters. En algunos de los gráficos se aprecia más la diferenciación de los grupos, en cambio, como en la representación visual de la primera-tercera coordenada, puede ser más difícil ver la representación ya que parecen estar en el mismo plano y es más difícil observar la diferenciación de los grupos.
No obstante, se observan claramente los diferentes grupos en la mayoría de perspectivas obteniendo diferentes vistas dependiendo de las dimensiones analizadas.

```
#ANÁLISIS DEL MODELO FINAL OBTENIDO:
#####################################################################

## EVALUACION clustering: calculamos la inercia y variabilidad explicada

#print km.def para saber las variables disponibles
#km.def[1]                  # Clusters
#km.def[2]                  # Centos de los clusters
#km.def[3]                  # Variabilidad total
#km.def[4]                  # Variabilidad para cada grupo
#km.def[5]                  # Variabilidad intra-grupo (Inercia)
#km.def[6]                  # Variabilidad entre-grupo
#km.def[[6]]/(#km.def[[3]])          # Variabilidad explicada

inercia_km.def <- km.def$tot.withinss  # Variabilidad intra-grupo (Inercia)
inercia_km.def #->14373067. a menor variabilidad intra-grupo más homogéneos son los grupos, mejor agrupación.

#EVALUACION:
variabilidad_Explicada_km.def <-100*km.def$betweenss/km.def$totss
variabilidad_Explicada_km.def # 37.67 % variabilidad explicada. Variabilidad entre grupos / variabilidad total

# Para concluir el análisis de los clusters obtenidos con K-MEANS:
# * Vemos que hemos pasado de una variabilidad explicada del 7.27% con dos grupos a una variabilidad explicada del 37.67% con los 11 grupos escogidos. Como era de esperar, la variabilidad explicada ha aumentado al incrementar el número de grupos, siendo 100% el valor máximo que podría ser obtenido si incrementamos el numero de grupos hasta el número de observaciones.

# *Por el contrario, la variabilidad intra-grupo (inercia) ha disminuido. Ha pasado de 21384032 con 2 grupos a 14373067 con 11 grupos. Esto significa que el modelo obtenido con 11 grupos es mejor que el obtenido con dos grupos, siendo los grupos más homogéneos y con menos variabilidad intra-grupo, lo que significa que la variabilidad entre grupos se ha reducido cerca de un 33%.
```

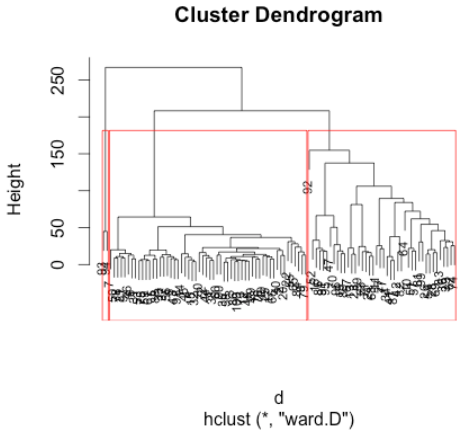
Nº grupos	Variabilidad intra-grupo	Variabilidad explicada
2 grupos	21384032	7.27%
11 grupos	14373067	37.67%

```
# Guartarmos en un fichero -> (id_producto, grupo asignado)
df <- data.frame(product_id = myIdsList , cluster=km.def$cluster)
write.table(df,'resultados_p3_partA.txt',sep=',',col.names=c("product_id","cluster"),row.names=TRUE,quote=FALSE,append=FALSE)
```




```
# Pie Chart from data frame with Appended Sample Sizes
mytable <- table(df$cluster)
lbls <- paste(names(mytable), "\n", mytable, sep="")
pie(mytable, labels = lbls,
main="Pie Chart of Groups\n (amount of observations per group)")
# Vemos que hay grupos con muy pocos elementos, lo que daría a pensar que se podría reducir el numero de grupos (lo cual ya sabemos que hay 9 grupos)
```

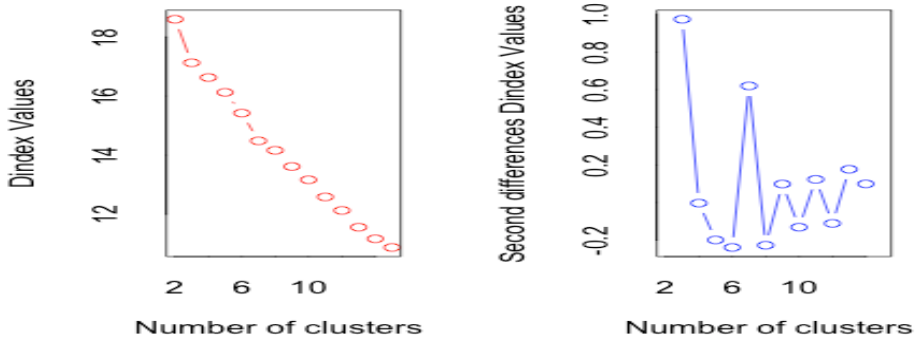
```
#################################################################
# Análisis con Clusterizacion jerarquica --> No es idonea para grandes conjuntos de datos
#################################################################
##
##
##-- 11. Calculamos las distancias con dist para los 100 primeros individuos y haz la jerarquia con hc
##-- Especifica los metodos que quieras mirando ?dist y ?hclust
##-- Recomendados: "euclidean" para dist y "ward.D" para hclust
nmax <- 100
d <- dist(datos2[1:nmax,], method = "euclidean")       # matriz de distancias
hc <- hclust(d,method = "ward.D")                  # jerarquizacion

##-- 12. Representamos graficamente la jerarquia
#windows(14,7)
quartz()
par(mfrow=c(1,1))
plot(hc,cex=0.7)
```



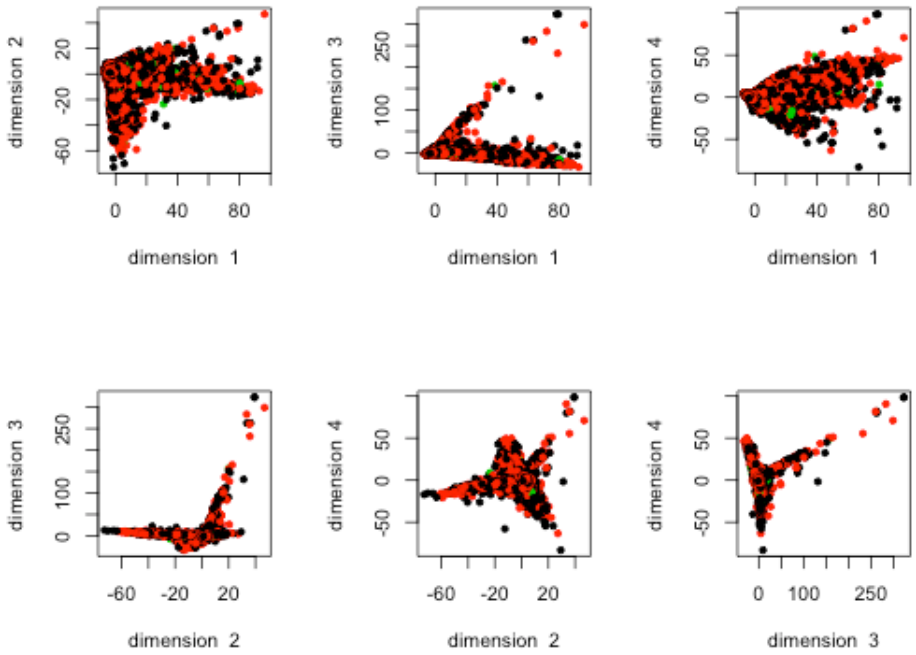
Es complicado decidir en cuantos grupos dividir el dentograma. No obstante, el paquete NbClust nos ayuda a decidir el numero de grupos óptimos.
En este ejemplo nos sugiere 3 grupos. Esto puede ser debido a que estamos analizando solo 100 observaciones.

Tradicionalmente buscamos por “el codo” (elbow). No obstante. el “second differences” plot (derecha) es más claro para la mayoría de observadores.

**Referencia: libro “Mastering Data Analysis with R” , pag 262.*



```
##-- 13. Viendo la jerarquizacion, escogemos el número de grupos idoneo
ct <- cutree(hc, k = 3)

##-- 14. Representa la clusterización hallada para todos los pares de las 4 primeras dimensiones
par(mfrow=c(3,2))
for(i in 1:3){
 for(j in (i+1):4){
  plot(pc$scores[,i],pc$scores[,j],pch=19,col=ct,cex=0.7,xlab=paste("dimension ",i), ylab=paste("dimension ",j))
 }
}
```



El análisis de las 4 primeras (Notese que sabemos que realmente hay 9 clases).

B) Segunda parte: Algoritmos supervisados

B.1) Evaluación de la segunda parte

Se usará la función de pérdida logarítmica multi-clase. Cada producto pertenece a una categoría real de la empresa. Para cada uno de ellos, se debe presentar un conjunto de probabilidades predichas (una por cada categoría). La fórmula de la función de pérdida es entonces,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

donde *N* es el número de productos en la prueba, *M* es el número de clases, *log* es el logaritmo natural, y_{ij} es 1 si la observación i está en la clase *j* y 0 en caso contrario, y p_{ij} es la probabilidad predicha de que la observación i pertenezca a la clase *j*.

Las probabilidades presentadas para un determinado producto deben sumar uno. Con el fin de evitar valores extremos de la función de registro, las probabilidades predichas son reemplazadas por max(min (p, 01/10^{-15}), 10^{-15}).

Se obtendrá una puntuación más alta cuanto mayor sea este estadístico. En ambas partes, también se valorará:

1. Los criterios de selección del modelo final
2. La presentación de los resultados

B.2) Entrega para la segunda parte:

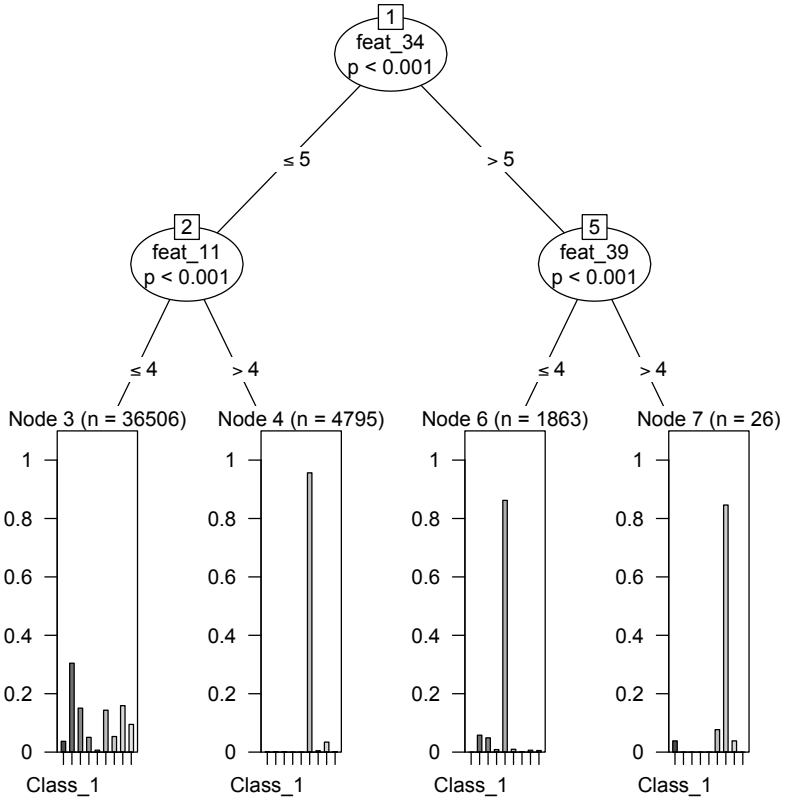
1. Una descripción del trabajo realizado y el proceso seguido para llegar a las conclusiones. Debe explicar el proceso utilizado para calcular las probabilidades y los parámetros del árbol condicional. Se valorará la presencia de gráficos que faciliten la comprensión. Extensión: 2 o 3 páginas. Formato: *.docx*

2. Documento de texto que contenga las probabilidades predichas de pertenencia de los productos a cada grupo en un documento con 10 columnas separadas por una coma. La primera debe contener el identificador del producto y las 9 restantes las probabilidades de pertenecer a cada grupo. Formato: *.txt*

3. El código utilizado para realizar esta parte con comentarios explicativos.

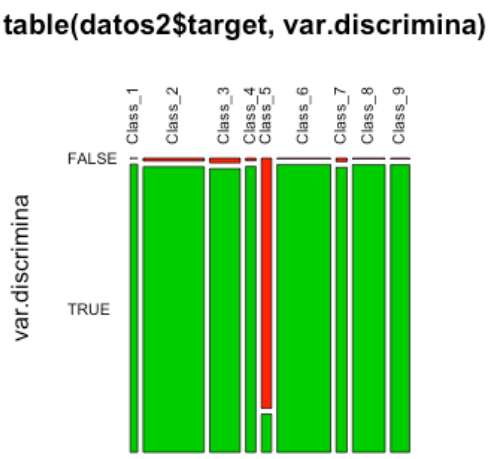
B.3) Solución

```
#############################################################
# Conditional trees
#############################################################
##-- 15. Se vuelven a leer los datos sin eliminar la respuesta y eliminando la variable id
setwd('/Users/Fer/Development/MASTERBD/ESTADISTICA/E_CLUSTERING/practica/')
datos2 <- read.table('p3_train.csv',header=TRUE,sep=';')
datos2$id <- NULL

##-- 16. Instalamos y cargamos el paquete party para calcular arboles condicionales
#install.packages('party')
library("party")

##-- 17. TEST con profundidad 2: Calculamos el arbol con los parametros dados
ct <- ctree(target~.,datos2,control = ctree_control(mincriterion = 0.95,   # pvalue<0.05 para hacer una particion
                                    minsplit = 50,    # minimo tamanyo de un nodo para hacer una particion
                                    minbucket = 20,   # minimo tama?o de un nodo (minbucket<minsplit/2)
                                    maxdepth = 2))    # Profundidad del arbol

##-- 18. Dibujamos el arbol
#windows(30,10,rescale="fixed")  # rescale="fixed" no adapta la ventana al tamanyo de la pantalla
quartz()
plot(ct)                  # arbol
```



##-- 19. Para la variable que mas discrima (raiz del arbol), creamos una variable segun este criterio de discriminacion
##-- Hacemos un mosaicplot de esta variable y la respuesta. Cual es la categoria de la respuesta que predice esta variable?
#Respuesta: Esta variable representa si la "feat_34"<=5 de cada observación, o sea si esta a la izquierda del árbol.

```
graphics.off()
var.discrimina <- datos2$feat_34 <= 5
mosaicplot(table(datos2$target,var.discrimina),col=2:3,las=2)
```



##-- 20. Leemos el fichero que contiene la muestra test

```
test <- read.table('p3_test (con respuesta).csv',header=TRUE,sep=';')
```

##-- 21.Miramos el % de aciertos que se obtiene con este arbol

```
pr <- predict(ct,test)        # Predicciones de las clases
(t <- table(pr,test$target))        # Tabla de prediciciones vs valor real
```

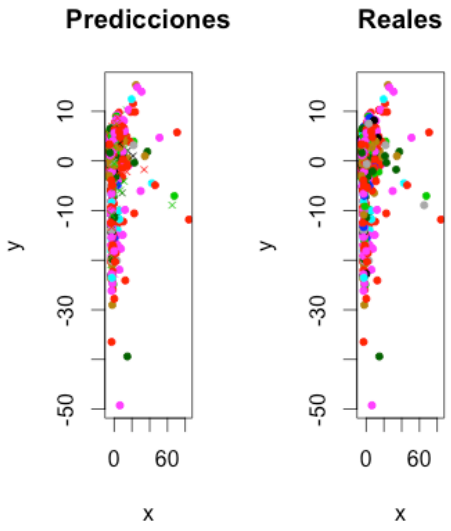
```
pr       Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
  Class_1     0     0     0     0     0     0     0     0     0
  Class_2   568  4836  2380   828   100  2244   828  2409  1470
  Class_3     0     0     0     0     0     0     0     0     0
  Class_4     0     0     0     0     0     0     0     0     0
  Class_5     1    58    25     2   788    11     0     2     5
  Class_6     5     1     5     0     0  2035     5    70     0
  Class_7     0     0     0     0     0     0    12     0     0
  Class_8     0     0     0     0     0     0     0     0     0
  Class_9     0     0     0     0     0     0     0     0     0
```

```
100*sum(diag(t))/sum(t)        # SOLUTION: 41.04773% de aciertos
```

##-- 22. Creamos otro arbol condicional en el que buscamos maximizar el porcentaje de aciertos.

```
# El procedimiento a seguir ha sido probar adaptando el minsplit y minbucket hasta obtener el mejor valor observado, una vez
# seleccionado el valor para estos dos parámetros intentar optimizar el maxdepth (profundidad del árbol).
# Parece ser que con profundidad 50 se obtienen los mejores resultados

ct1 <- ctree(target~.,datos2,control = ctree_control(mincriterion = 0.95,
                                    minsplit = 14,
                                    minbucket = 7,
                                    maxdepth = 50))
```

```
##-- 23. Realizamos la predicción sobre la muestra test y comprueba si mejoras el nivel de aciertos
pr1 <- predict(ct1,test)        # Predicciones de las clases (clase predicha)
(t1 <- table(pr1,test$target))        # Tabla de prediciciones vs valor real

pr1      Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
 Class_1    128     3     4     2     0    23    13    52    27
 Class_2     77   4147   1512   435    60    96   164    80   117
 Class_3      9   518    714    78     1    13    57    28    23
 Class_4      7    77    54    250     2    26    18     3    11
 Class_5      2    13     1     3    819     5     1    10     6
 Class_6     59    30    17    35     4   3857    75   106    58
 Class_7     27    33    54    20     0    63   414    52    20
 Class_8    123    48    34     1     1   118    89   2060   121
 Class_9    142    26    20     6     1    89    14    90   1092

100*sum(diag(t1))/sum(t1)       # 72.1372 % de aciertos

##-- 24. Representamos en dos gráficos las dos primeras dimensiones del princomp para las 1000 primeras observaciones
##-- En el primero, con colores según las predicciones y con un símbolo diferente dependiendo de si se acertado o no
##-- En el segundo con los colores según los valores reales
x <- pc$scores[1:1000,1]            # Primera coordenada
y <- pc$scores[1:1000,2]            # Segunda coordenada
colores1 <- as.numeric(pr1)          # predicciones pasadas a números
colores2 <- as.numeric(test$target)          # Valores reales pasados a números
par(mfrow=c(1,2))
plot(x,y,col=colores1,cex=0.7,pch=ifelse(pr1==test$target,19,4),main="Predicciones") # pch=19 (punto); pch=4 (cruz)
plot(x,y,col=colores2,cex=0.7,pch=19,main="Reales")
```



Se observa los valores reales a la derecha y las predicciones a la izquierda con una “x” cuando una predicción no coincidió con la realidad.

```
##-- 25. Calculamos las predicciones de las probabilidades para cada categoría y mira las 6 primeras predicciones
pr.prob0 <- predict(ct1,test,type="prob")
head(pr.prob0,6)
```

#Hemos obtenido una lista que contiene en cada posición un vector con 9 elementos.
#Cada elemento es la probabilidad de esa categoría, siendo la suma total de las nueve categorías 1 (100%).
 # i.e: sum(pr.prob0[[1]]) → 1, sum(pr.prob0[[2]]) → 1 …, sum(pr.prob0[[n]]) →1

```
##-- 26. Pasamos las probabilidades de una lista a una matriz y miramos las 6 primeras filas de predicciones
pr.prob <- matrix(unlist(pr.prob0),ncol=9,byrow=TRUE)
head(pr.prob,6)

##-- 27. Escogemos una observacion al azar en que no acierte la prediccion
##-- Miramos que probabilidad tenia la respuesta real en este caso
fallos <- which(pr1!=test$target)       # fallos (no coincide predicción con valor real)
set.seed(12345)
sel <- sample(fallos,1)         # un fallo al azar
test$target[sel]              # valor real (clase de pertinencia)  del elemento escogido → Class_2
pr.prob[sel,]           # probabilidades del elemento escogido → 0.040, 0.255, 0.245, 0.355, 0.000, 0.010, 0.085, 0.010, 0.000
```

#En este ejemplo vemos que el valor real era Class_2, pero la predicción le daba una probabilidad de 2.5%. En cambio fue predicho Class_4 con un 3.55%. pr1[sel] → Class_2

```
##-- 28. Escribimos las probabilidades con el identificador en un fichero separado por comas
df <- data.frame(id=test$id,pr.prob)
write.table(df,'resultados_parteB.txt',sep=',',dec='.',row.names=FALSE,quote=FALSE,append=FALSE)

# Para finalizar, visualizamos las 3 primeras filas que se almacenan en el fichero. Cada fila una observación con la probabilidad de pertenencia a cada una de las 9 clases.
> df[1:3,]
  id         X1       X2       X3       X4        X5         X6         X7       X8         X9
1  1 0.0833333333 0.0000000 0.00000000 0.00000000 0.055555556 0.1111111111 0.0000000000 0.16666667 0.5833333333
2  2 0.0008787346 0.9165202 0.05536028 0.02460457 0.000000000 0.0008787346 0.0008787346 0.00000000 0.0008787346
3  4 0.0088888889 0.5466667 0.20444444 0.08444444 0.008888889 0.0577777778 0.0666666667 0.02222222 0.0000000000
```

 # CALCULATING LOGLOSS (EVALUATION OF CLASSIFICATION ALGORITHM)

```
handle_extreme_values <- function(p){
  res <- max(min(p,1-exp(-15)),exp(-15))
  return(res)
}

logloss <- 0
for(i in 1:nrow(df))
  for(j in 1:9)
    if(!is.element(i, fallos))
      logloss <- logloss + log(handle_extreme_values(df[i,j]))

logloss <- logloss*(-1/nrow(df))
logloss
```

#OBTAINED LOGLOSS VALUE: 53.44 (i.e, for deep 15 the logloss value was much lower)