# Lectura de datos

```
datos_train = read.csv("datos/datos_train_preprocesado.csv", stringsAsFactors = F)
datos_test = read.csv("datos/datos_test_preprocesado.csv", stringsAsFactors = F)
datos_total <- union(datos_train,datos_test)
```

## Árboles de decisión

El árbol de decisión es una técnica clásica de clasificación en la que se realizan particiones binarias de los datos de forma recursiva. El resultado se puede representar con un árbol.

Para la construcción del árbol también se puede utilizar la función **rpart**.El principal parámetro de esta función es la complejidad, **cp**. Éste permite simplificar los modelos ajustados mediante la poda de las divisiones que no merecen la pena. Otros parámetros importantes son **minsplit**, número mínimo de observaciones que debe haber en un nodo para que se intente una partición, y **minbucket**, número mínimo de observaciones de un nodo terminal. Por defecto minsplit= 20, minbucket=round(minsplit/3) y cp= 0,01.

A la hora de determinar los parámetros, el procedimiento habitual para determinar los parámetros del modelo consiste en dejar que el árbol crezca hasta que cada nodo terminal tenga menos de un número mínimo de observaciones, ya que en algunas divisiones puede que apenas mejore y en las siguientes sí lo haga. Para determinar *cp* se considera como 0, para que el árbol crezca lo máximo posible.

A continuación se va aplicar dicha técnica para realizar las siguientes predicciones:

### Predicción de efectivenessNumber en función del rating

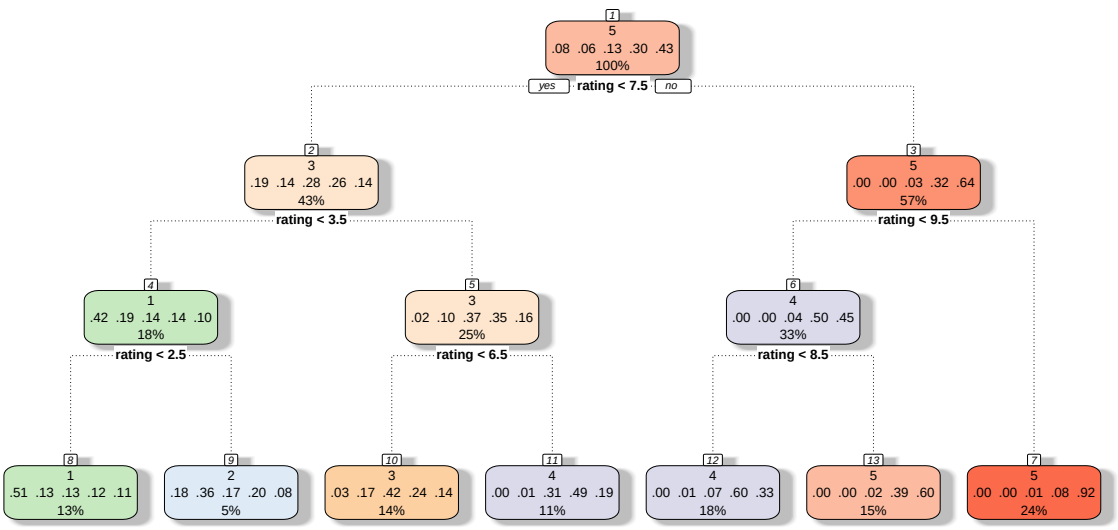
En esta sección se va a predecir la efectividad (de 1 a 5, siendo 1 menos efectiva y 5 más efectiva) en función del rating. Para ello se va a seguir con el procedimiento que se ha descrito en la introdución.

A continuación se va a mostrar el árbol de decisión que se ha generado utilizando **rpart**.

```
library("rattle")
```

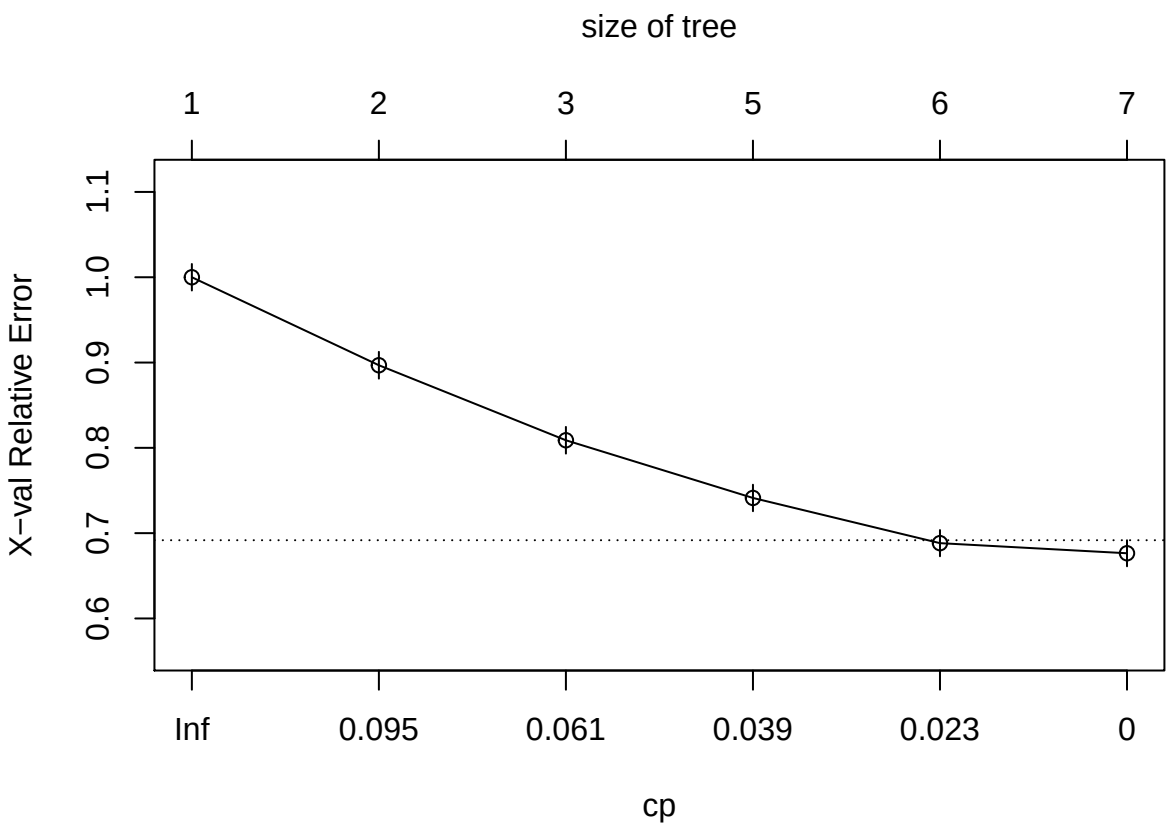
```
## Rattle: A free graphical interface for data science with R.
## Versión 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
library("rpart")
library("rpart.plot")

seed <- 42

set.seed(seed)

rpart <- rpart(effectivenessNumber ~ rating, data=datos_train,
                method="class",
                parms=list(split="information"),
                control=rpart.control(minsplit=30,
                minbucket=10,
                cp=0.00,
                usesurrogate=0,
                maxsurrogate=0)
              )

fancyRpartPlot(rpart, main="Decision Tree effectivenessNumber $ ratingLabel")
```




Como el árbol que se obtiene es muy grande, se podaría considerando una función de coste basada en la complejidad. Para esto, se calcula el error de la validación cruzada y se elige el menor.

```
plotcp(rpart)
```




Mostramos la tabla de resultados

```
printcp(rpart)
```

```
##
## Classification tree:
## rpart(formula = effectivenessNumber ~ rating, data = datos_train,
##     method = "class", parms = list(split = "information"), control = rpart.control(minsplit = 30,
##         minbucket = 10, cp = 0, usesurrogate = 0, maxsurrogate = 0))
##
## Variables actually used in tree construction:
## [1] rating
##
## Root node error: 1774/3104 = 0.57152
##
## n= 3104
##
##         CP nsplit rel error  xerror     xstd
## 1 0.103157      0   1.00000 1.00000 0.015541
## 2 0.087937      1   0.89684 0.89684 0.015698
## 3 0.042841      2   0.80891 0.80891 0.015658
## 4 0.034949      4   0.72322 0.74126 0.015519
## 5 0.014656      5   0.68828 0.68828 0.015342
## 6 0.000000      6   0.67362 0.67644 0.015294
```

Tal y como se puede ver en la tabla anterior, el mínimo error se alcanza en el nodo 6.

```
6 0.000000      6   0.67362 0.67644 0.015294
```

Típicamente se considera que hasta la línea discontinua de la figura anterior no hay diferencias significativas.

Esta línea es la suma del mínimo error y la desviación típica, es decir:

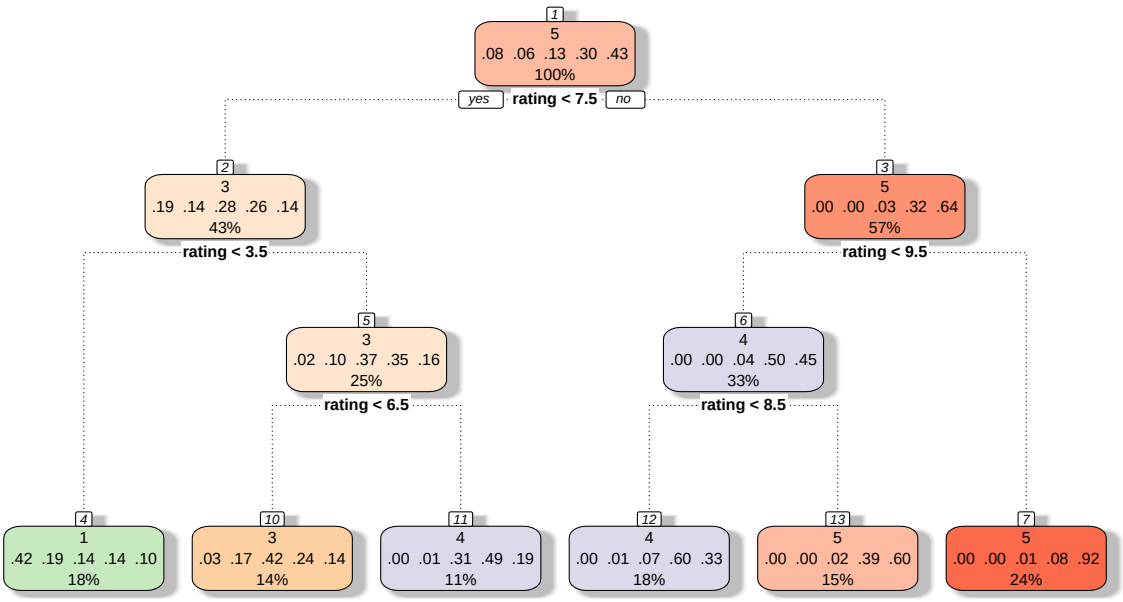
```
0.67644 + 0.015294 = 0.691734
```

Observamos en el gráfico anterior que dicho error tiene un cp de 0.023, por lo que se va a volver a generar el árbol con rpart utilizando ahora como valor de cp 0,023.

```
set.seed(seed)
rpart <- rpart(effectivenessNumber ~ rating, data=datos_train,
               method="class",
               parms=list(split="information"),
               control=rpart.control(minsplit=30,
               minbucket=10,
               cp=0.023,
               usesurrogate=0,
               maxsurrogate=0)
              )
# Vista textual del modelo árbol de decisión:
print(rpart)
```

```
## n= 3104
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3104 1774 5 (0.08 0.06 0.13 0.3 0.43)
##    2) rating< 7.5 1324  959 3 (0.19 0.14 0.28 0.26 0.14)
##      4) rating< 3.5 553  318 1 (0.42 0.19 0.14 0.14 0.099) *
##      5) rating>=3.5 771  485 3 (0.016 0.096 0.37 0.35 0.16)
##       10) rating< 6.5 422  245 3 (0.028 0.17 0.42 0.24 0.14) *
##       11) rating>=6.5 349  178 4 (0 0.0086 0.31 0.49 0.19) *
##    3) rating>=7.5 1780  632 5 (0 0.0028 0.028 0.32 0.64)
##      6) rating< 9.5 1038  518 4 (0 0.0039 0.044 0.5 0.45)
##       12) rating< 8.5 558  224 4 (0 0.0072 0.068 0.6 0.33) *
##       13) rating>=8.5 480  194 5 (0 0 0.017 0.39 0.6) *
##      7) rating>=9.5 742   62 5 (0 0.0013 0.0054 0.077 0.92) *
```

```
fancyRpartPlot(rpart, main="Árbol de decisión effectivenessNumber $ rating")
```




A continuación se lleva a cabo la predicción del conjunto de prueba y se calcula la matriz de confusión.

Para cada uno de los datos del conjunto de prueba se sigue el árbol según el valor de sus variables hasta llegar a los nodos terminales, allí, se clasifica con la categoría más probable de ese nodo terminal, como se vio en el árbol anterior.

La denominada matriz de confusión permite la visualización de los resultados del clasificador. Las filas muestran los valores reales y las columnas las predicciones del modelo. A partir de esta tabla se calcula la precisión del modelo.

```
pred <- predict(rpart, newdata=datos_test, type="class")

# Matriz de confusión
tabla <-table(datos_test$effectivenessNumber, pred, useNA="ifany",
dnn=c("Real", "Predicho" ))

tabla
```

```
##     Predicho
## Real   1   2   3   4   5
##    1  74   0   5   0   2
##    2  46   0  28   2   0
##    3  34   0  65  56   2
##    4  28   0  41 170  70
##    5  14   0  30  82 285
```

Esta tabla es la denominada matriz de confusión, que permite la visualización de los resultados del clasificador. Las filas muestran los valores reales y las columnas las predicciones del modelo. A partir de esta tabla se calcula la precisión del modelo.

```r
sum(diag(tabla))/nrow(datos_test)
```

```
## [1] 0.5744681
```

La precisión de este modelo es del **57,44%** aproximadamente.

El poder de predicción de los árboles de decisión no suele ser muy bueno, pero el algoritmo es sencillo y los modelos resultantes tienen una fácil interpretación.

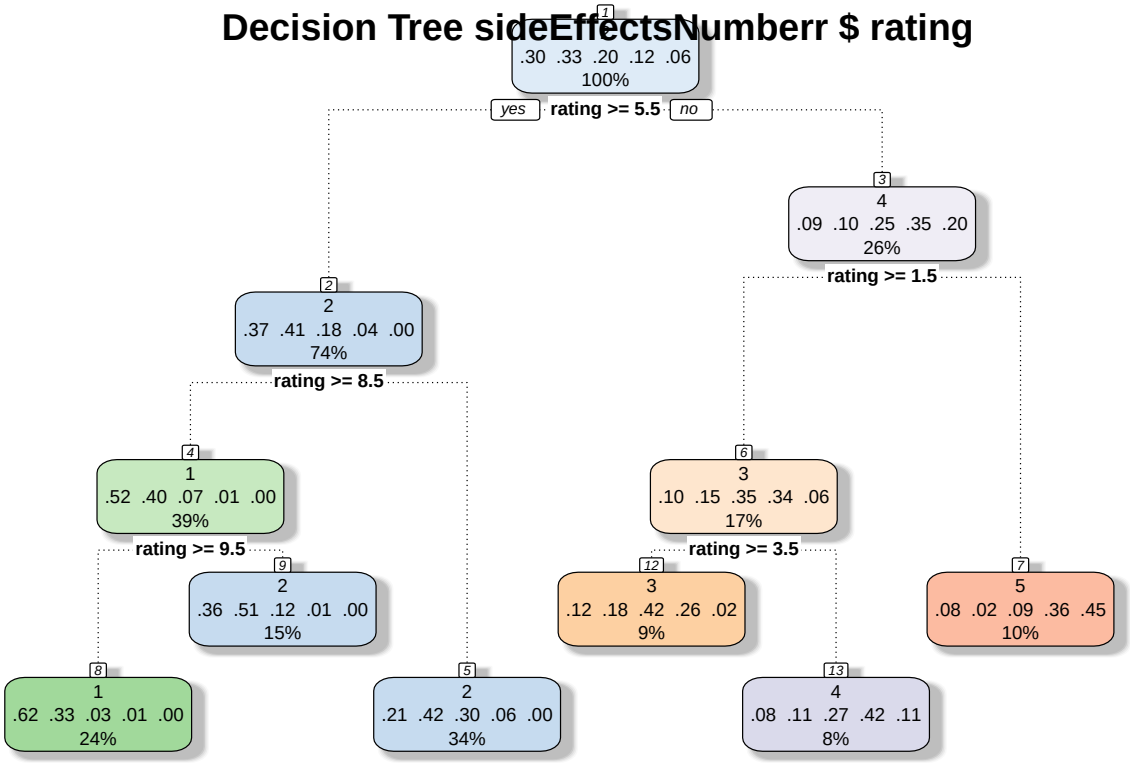
### Predicción de sideEffectsNumber en función del rating

Siguiendo el mismo proceso que se ha seguido para la obtención del árbol generado anteriormente, se va a predecir el valor de efectos secundarios en función del rating del medicamento.

```r
seed <- 42

set.seed(seed)

rpart <- rpart(sideEffectsNumber ~ rating, data=datos_train,
               method="class",
               parms=list(split="information"),
               control=rpart.control(minsplit=30,
               minbucket=10,
               cp=0.01,
               usesurrogate=0,
               maxsurrogate=0)
              )


print(rpart)
```

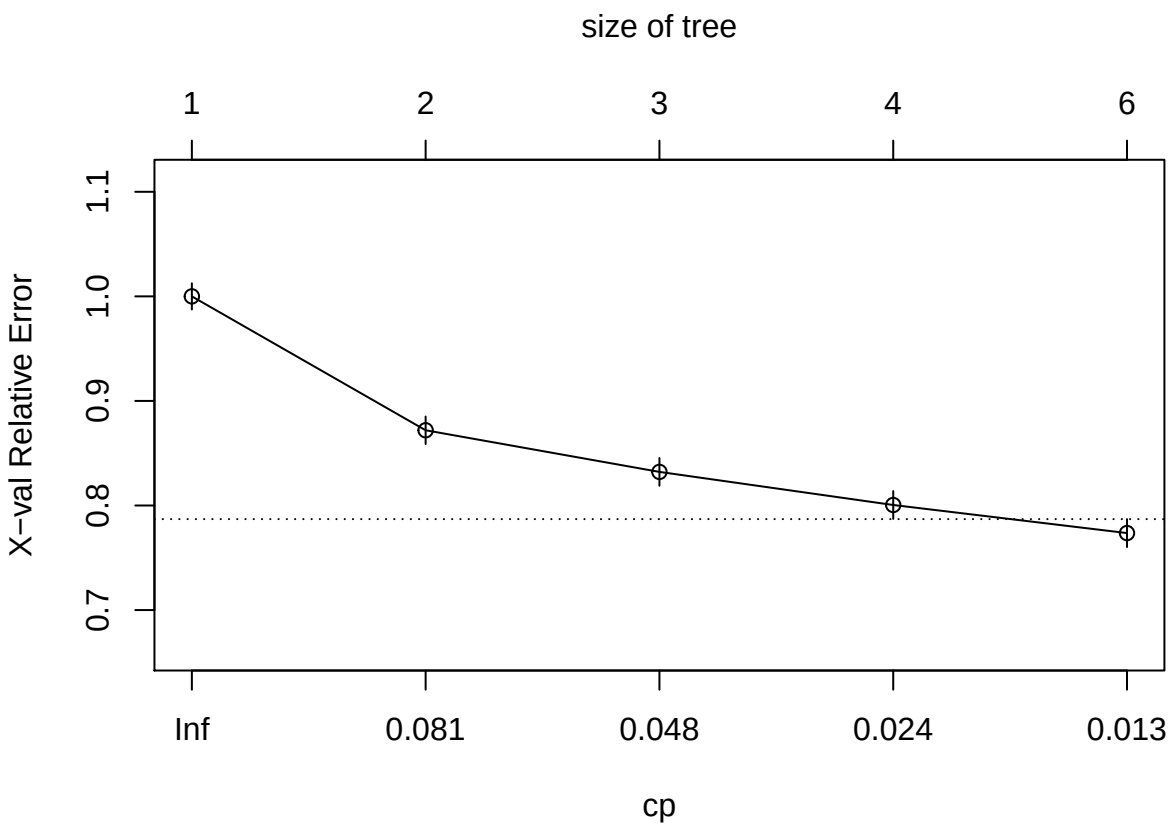
```
## n= 3104
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3104 2085 2 (0.3 0.33 0.2 0.12 0.056)
##    2) rating>=5.5 2286 1349 2 (0.37 0.41 0.18 0.036 0.0035)
##      4) rating>=8.5 1222  591 1 (0.52 0.4 0.067 0.012 0.0025)
##        8) rating>=9.5 742  282 1 (0.62 0.33 0.03 0.013 0.004) *
##        9) rating< 9.5 480  236 2 (0.36 0.51 0.12 0.01 0) *
##      5) rating< 8.5 1064  618 2 (0.21 0.42 0.3 0.064 0.0047) *
##    3) rating< 5.5 818  533 4 (0.093 0.1 0.25 0.35 0.2)
##      6) rating>=1.5 513  333 3 (0.1 0.15 0.35 0.34 0.06)
##       12) rating>=3.5 265  153 3 (0.12 0.18 0.42 0.26 0.015) *
##       13) rating< 3.5 248  144 4 (0.085 0.11 0.27 0.42 0.11) *
##      7) rating< 1.5 305  169 5 (0.075 0.023 0.092 0.36 0.45) *
```

```
fancyRpartPlot(rpart, main="Decision Tree sideEffectsNumberr $ rating")
```




Mostramos la gráfica del error de la validación cruzada.

```
plotcp(rpart)
```




Generamos la predicción

```
pred <- predict(rpart, newdata=datos_test, type="class")

# Matriz de confusión
tabla <-table(datos_test$sideEffectsNumber, pred, useNA="ifany",
dnn=c("Real", "Predicho" ))

tabla
```

```
##     Predicho
## Real   1   2   3   4   5
##    1 141 109   7   7   4
##    2  74 219  24  10   2
##    3   9 143  48  26  10
##    4   2  24  30  31  35
##    5   0   6   2   8  63
```

Obtenemos el resultado

```
sum(diag(tabla))/nrow(datos_test)
```

```
## [1] 0.4854932
```

La precisión de este modelo es del **50,58%** aproximadamente.

### Predicción de effectivenessNumber en función del weightedRating

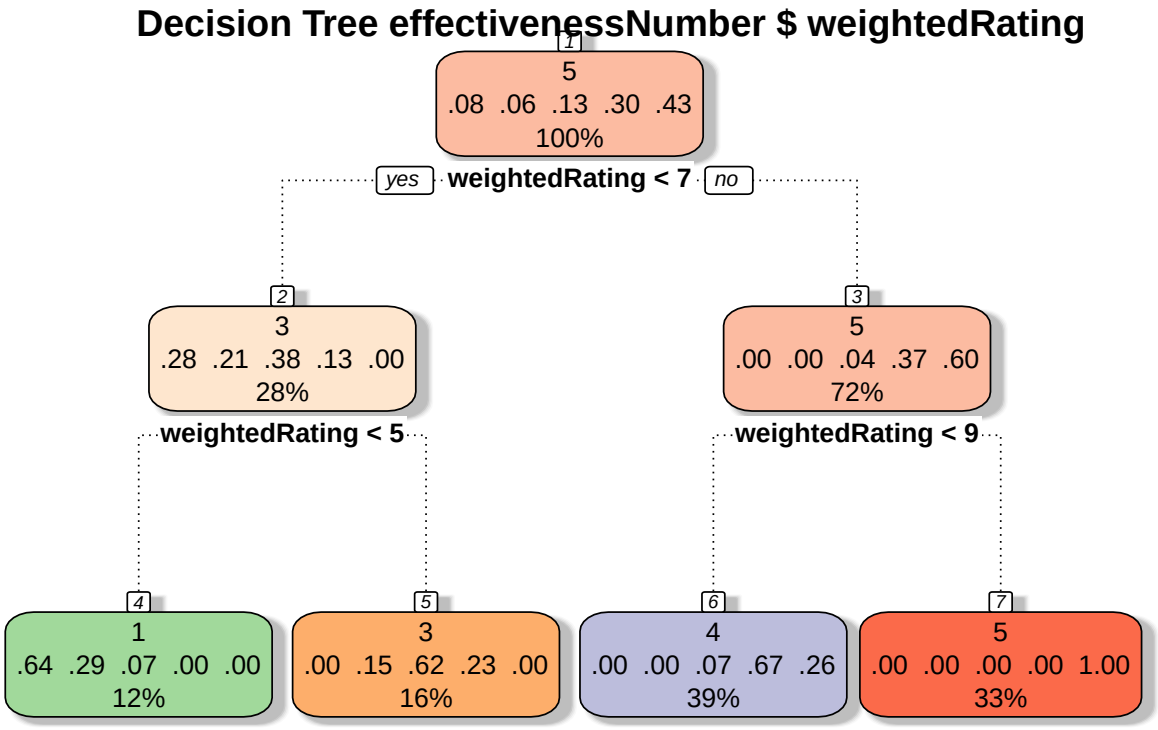
Siguiendo el mismo proceso que se ha seguido para la obtención del árbol generado anteriormente, se va a predecir el valor de la efectividad del medicamento en función del rating ponderado que se ha generado

durante el procesamiento ( 70% efectos secundarios, 30% efectividad).

```
seed <- 42

set.seed(seed)

rpart <- rpart(effectivenessNumber ~ weightedRating, data=datos_train,
                method="class",
                parms=list(split="information"),
                control=rpart.control(minsplit=30,
                minbucket=10,
                cp=0.03,
                usesurrogate=0,
                maxsurrogate=0)
              )


print(rpart)
```

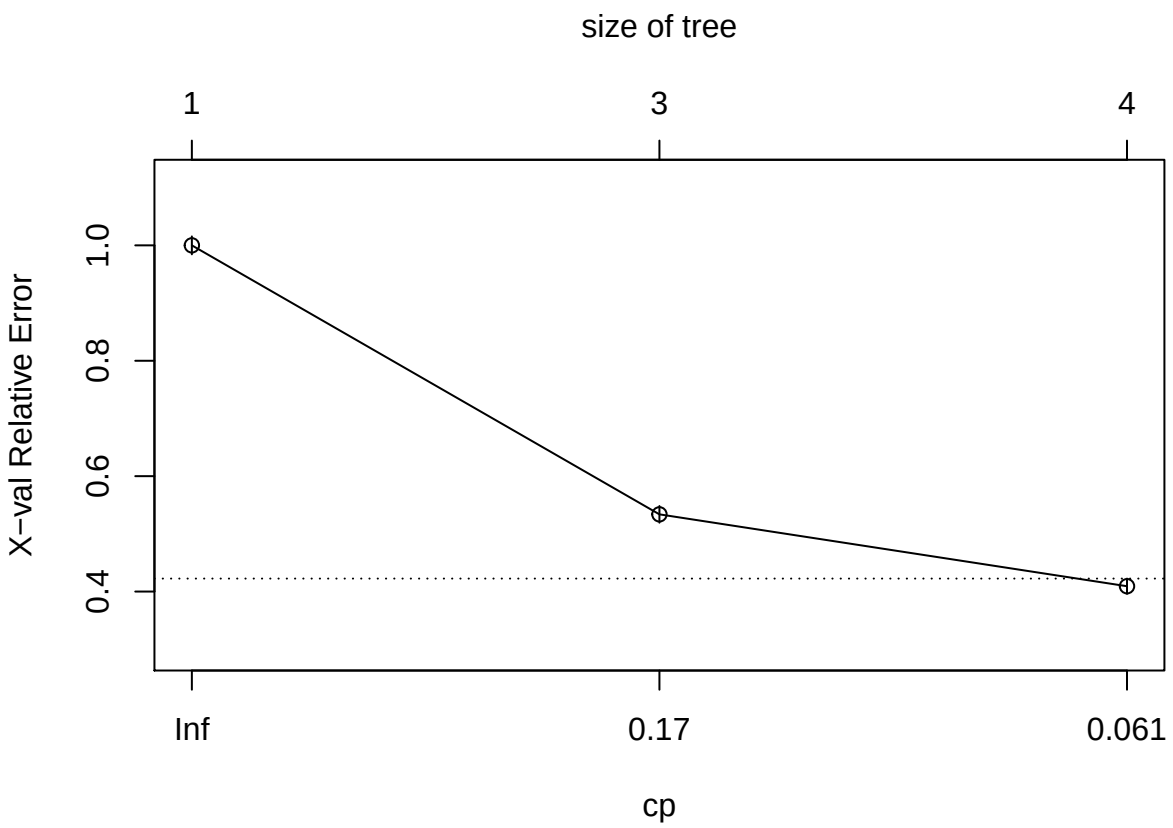
```
## n= 3104
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3104 1774 5 (0.08 0.06 0.13 0.3 0.43)
##   2) weightedRating< 7 881  547 3 (0.28 0.21 0.38 0.13 0)
##     4) weightedRating< 5 387  140 1 (0.64 0.29 0.067 0 0) *
##     5) weightedRating>=5 494  186 3 (0 0.15 0.62 0.23 0) *
##   3) weightedRating>=7 2223  893 5 (0 0 0.036 0.37 0.6)
##     6) weightedRating< 9 1212  400 4 (0 0 0.067 0.67 0.26) *
##     7) weightedRating>=9 1011    0 5 (0 0 0 0 1) *
```

```
fancyRpartPlot(rpart, main="Decision Tree effectivenessNumber $ weightedRating")
```




Mostramos la gráfica del error de la validación cruzada.

```
plotcp(rpart)
```




Generamos la predicción

```
pred <- predict(rpart, newdata=datos_test, type="class")

# Matriz de confusión
tabla <-table(datos_test$effectivenessNumber, pred, useNA="ifany",
dnn=c("Real", "Predicho" ))

tabla
```

```
##     Predicho
## Real   1   2   3   4   5
##    1  63   0   9   9   0
##    2  25   0  43   8   0
##    3  42   0  53  62   0
##    4  14   0 108 123  64
##    5  12   0  25 210 164
```

Obtenemos el resultado

```
sum(diag(tabla))/nrow(datos_test)
```

```
## [1] 0.3897485
```

La precisión de este modelo es del **53,57%** aproximadamente.

### Conclusión

Haciendo un breve resumen de lo obtenido, tenemos lo siguiente:

- Porcentaje de acierto en la predicción de la efectividad en función del rating: **57,44%**
- Porcentaje de acierto en la predicción de los efectos secundarios en función del rating: **50,58%**
- Porcentaje de acierto en la predicción de la efectividad en función del rating ponderado: **53,57%**
- Porcentaje de acierto en la predicción de los efectos secundarios en función del rating ponderado: **73,21%**

Como se puede observar, el porcentaje de acierto de la efectividad teniendo en cuenta el rating (dado por los usuarios) y el rating ponderado es muy similar, sin embargo la predicción para los efectos secundarios difiere en un 23% a favor del rating ponderado. Esto es debido a que en el rating ponderado se ha tenido más en cuenta el impacto de los efectos secundarios y esto hace que la predicción de dichos efectos secundarios sea más precisa.

### Predicción del rating en función del comentario de beneficios

En esta sección se va a realizar una predicción de la efectividad en base a los comentarios aportados por los usuarios (benefitsReview).

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

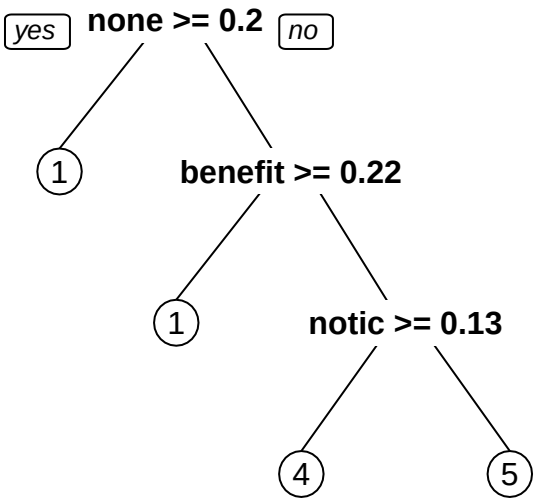
```
library(tm)
```

```
## Loading required package: NLP
```

```
library(SnowballC)

datos_train = read.csv("datos/datos_train_preprocesado.csv", stringsAsFactors = F)
datos_test = read.csv("datos/datos_test_preprocesado.csv", stringsAsFactors = F)
datos_total <- bind_rows(datos_train,datos_test)
corpus = Corpus(VectorSource(datos_total$benefits_preprocesado))
# Creamos matriz de términos con las palabras de los comentarios. Entonces cada fila, va a estar formad
tdm <- tm::DocumentTermMatrix(corpus)
tdm.tfidf <- tm::weightTfIdf(tdm)
reviews = as.data.frame(cbind(datos_total$effectivenessNumber, as.matrix(tdm.tfidf)))
preparado_train <- reviews[1:1500,]
preparado_test <- reviews[-(1:1000),]
```

A continuación se va a generar el modelo utilizando como variable a predecir el effectivenessNumber con el resto de palabras que se ha almacenado en preparado_train.

```
reviews.tree = rpart(V1~ .,  method = "class", data = preparado_train );
prp(reviews.tree)
```




En realidad, es lógico que si utilizamos un conjunto de datos mayor, sobretodo en el caso de los textos, que haga predicciones muy muy restringidas, ya que al haber tanta cantidad de palabras tiene que generalizar un montón y al final, esa generalización se reduce a utilizar pocas palabras para clasificar en las distintas etiquetas. Por ello, hemos ido probando distintos tamaños hasta dar con uno que visualmente nos compense. Pero obviamente, la importancia no está en que visualmente nos parezca un árbol representativo y que de buenos resultados, sino en que a la hora de predecir el conjunto de test, nos de resultado realmente representativos.

Por ello vamos a hacer predict con el árbol que hemos generado anteriormente. Simplemente le pasamos a la función predict de la librería `rattle` el árbol que hemos entrenado.

```
library(rattle)
pred <- predict(reviews.tree, newdata=preparado_test, type="class")
table(pred)
```

```
## pred
##    1    2    3    4    5
##  224    0    0  112 2802
```

```
table(preparado_test$V1)
```

```
##
##    1    2    3    4    5
##  244  205  438  933 1318
```

```
# Matriz de confusión
tabla <-table(preparado_test$V1, pred, useNA="ifany",
dnn=c("Real", "Predicho" ))
tabla
```

```
##     Predicho
## Real    1    2    3    4    5
##    1  116    0    0    3  125
##    2   32    0    0   15  158
##    3   20    0    0   21  397
##    4   27    0    0   45  861
##    5   29    0    0   28 1261
```

Obtenemos el resultado

```
sum(diag(tabla))/nrow(datos_test)
```

```
## [1] 1.375242
```

Con el fin de mejorar esta predicción, se puede seguir la idea del bagging de combinar muchos métodos sencillos, como se verá en el método de bosques aleatorios.