Introducción al Aprendizaje Estadístico - Tareas de Clase 5 - Resolución
Ana M. Bianco - Paula M. Spano

Ejercicio Datos Carseats:
Este es un conjunto de datos simulados, incluido en la librería ISLR. El mismol contiene las ventas de butacas infantiles para automóviles, proveniente de 400 comercios diferentes.
El objetivo es predecir si el volumen de ventas superará o no las 8000 unidades. Para ello se reemplazó la variable "Sales" por "High", la cual indica con "Yes", si las unidades superan las 8000 y "No", si no superan esa cantidad. El archivo modificado lo pueden descargar AQUí.

1. Explorar el conjunto de datos y convertir en factores las variables que deban llevar ese tratamiento.

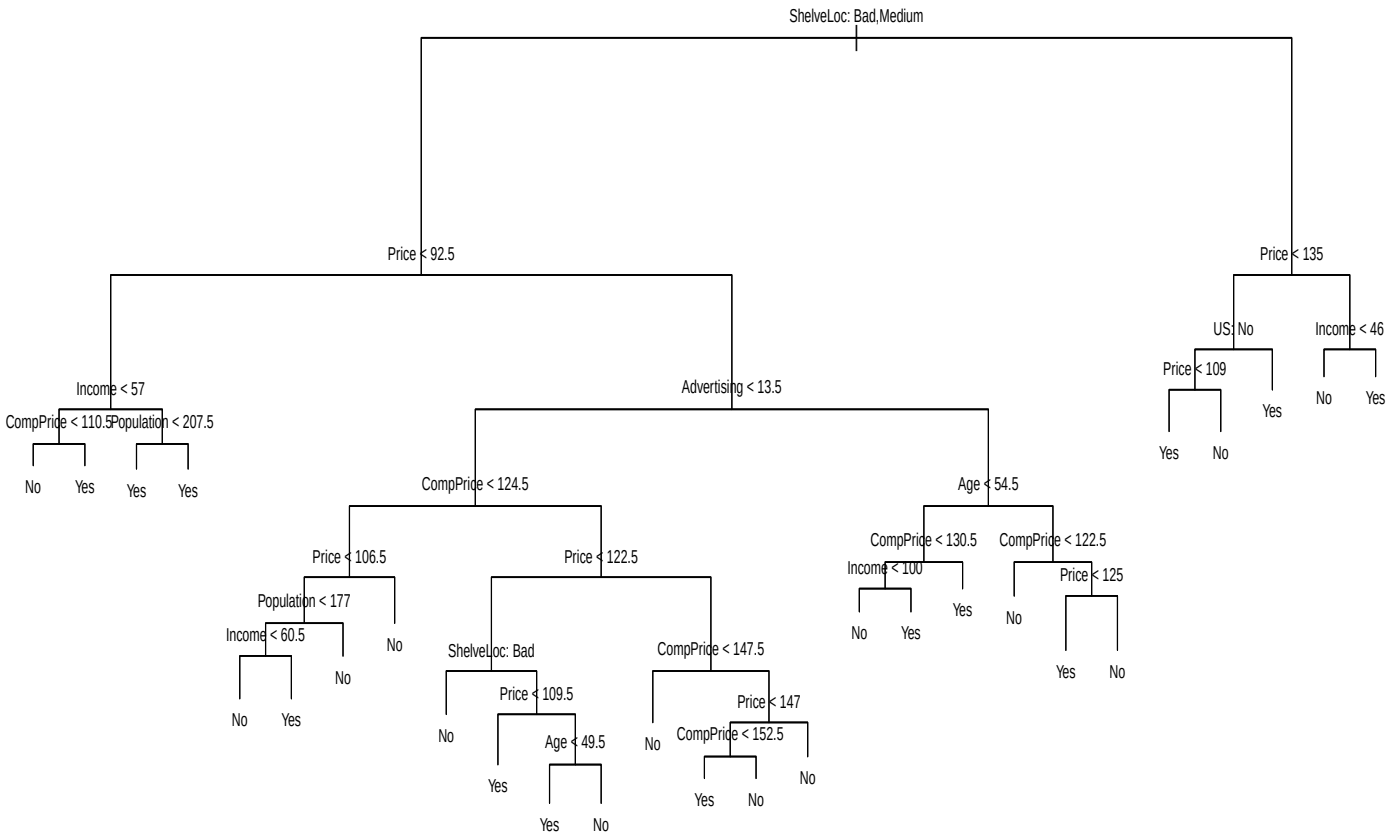
```
library (ISLR)

butacas <- read.csv("carsH.csv",sep=",",header=T)
```

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US	High
1	138	73	11	276	120	Bad	42	17	Yes	Yes	Yes
2	111	48	16	260	83	Good	65	10	Yes	Yes	Yes
3	113	35	10	269	80	Medium	59	12	Yes	Yes	Yes
4	117	100	4	466	97	Medium	55	14	Yes	Yes	No
5	141	64	3	340	128	Bad	38	13	Yes	No	No
6	124	113	13	501	72	Bad	78	16	No	Yes	Yes
7	115	105	0	45	108	Medium	71	15	Yes	Yes	No
8	136	81	15	425	120	Good	67	10	Yes	Yes	Yes
9	132	110	0	108	124	Medium	76	10	No	No	No
10	132	113	0	131	124	Medium	76	17	No	Yes	No
11	121	78	9	150	100	Bad	26	10	No	Yes	Yes
12	117	94	4	503	94	Good	50	13	Yes	Yes	Yes
13	122	35	2	393	136	Medium	62	18	Yes	No	No
14	115	28	11	29	86	Good	53	18	Yes	Yes	Yes
15	107	117	11	148	118	Good	52	18	Yes	Yes	Yes

```
butacas$ShelveLoc <- as.factor(butacas$ShelveLoc)
butacas$Urban <- as.factor(butacas$Urban )
butacas$US <- as.factor(butacas$US)
butacas$High <- as.factor(butacas$High)
```

2. Utilizar el comando **tree** para crear un árbol con el objetivo de clasificar el volumen de ventas (High) en función de las otras variales y graficarlo.

```
library (tree)

ventas.tree<- tree(High~.,butacas)

plot(ventas.tree);text(ventas.tree,pretty =0,cex=0.5)
```



3. Determinar las variables utilizadas en los nodos internos, la cantidad de nodos terminales, el error de clasificación total y el error de clasificación del primer y segundo nodo.

```
summary(ventas.tree)
```

```
Classification tree:
tree(formula = High ~ ., data = butacas)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"  "Advertising" "Age" "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

```
err1<-butacas$High[butacas$ShelveLoc!='Good'&butacas$Price<92.5&butacas$Income<57&butacas$CompPrice<110.5]
mean(err1=='Yes')
sum(err1=='Yes')
length(err1)

err2<-butacas$High[butacas$ShelveLoc!='Good'&butacas$Price<92.5&butacas$Income<57&butacas$CompPrice>=110.5]
```

```
mean(err2=='No')
sum(err2=='Yes')
length(err2)
```

```
[1] 0
[1] 0
[1] 5

[1] 0.4
[1] 2
[1] 5
```

Calcular "a mano" el error de clasificación total:

```
ventas.pred <- predict(ventas.tree, butacas, type ="class")
Highnum <- (butacas$High=='Yes')*1
sum(abs(Highnum-(ventas.pred=='Yes')*1))
```

```
[1] 36
```

```
length(Highnum)
[1] 400
```

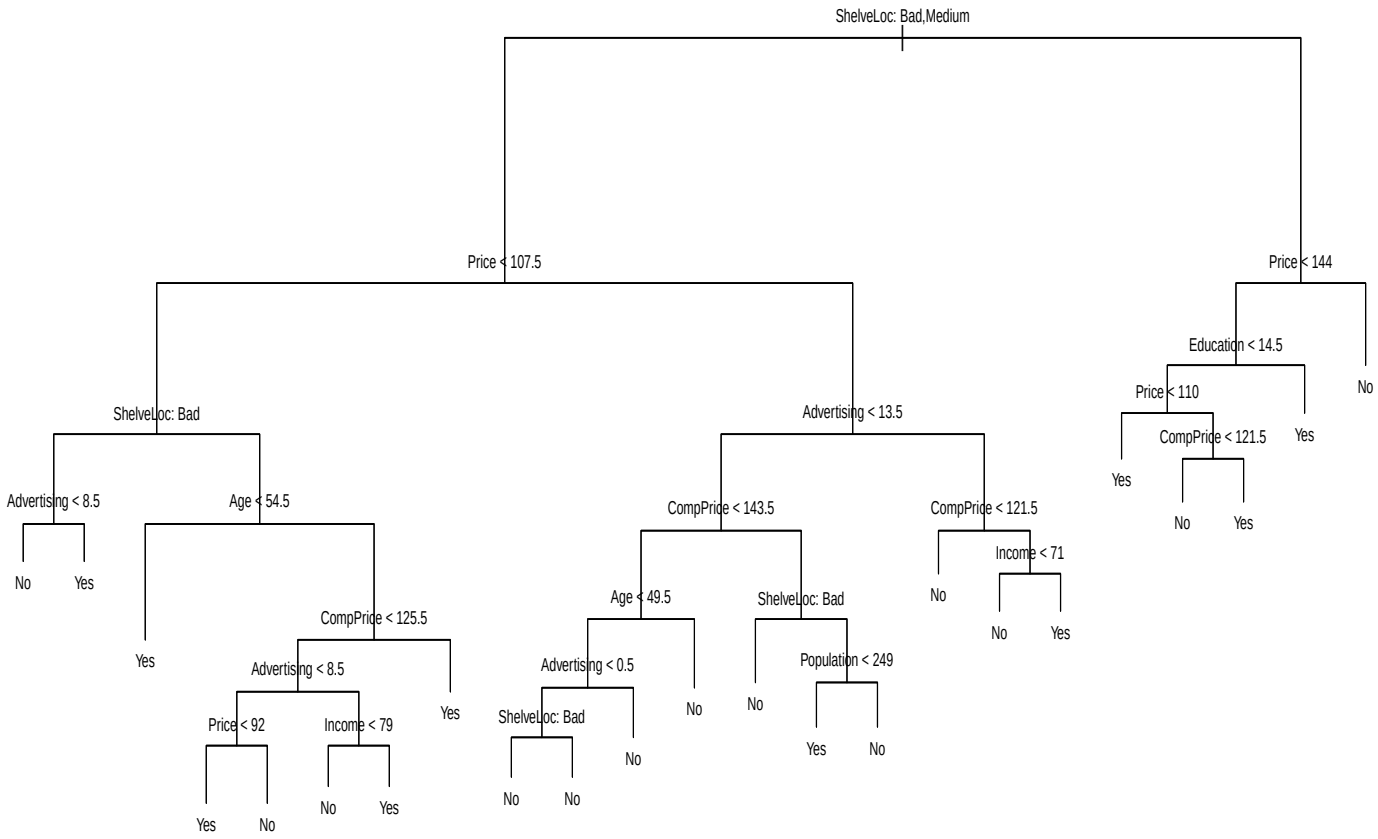
```
36/400
[1] 0.09
```

4. Con el objetivo de tener una medida confiable de la bondad del ajuste del árbol encontrado, dividir las observaciones en dos muestras, una de entrenamiento (250) y la otra para testear (150). Obtener el árbol de decisión y calcular la proporción de predicciones bien hechas.

```
set.seed (123)
entrenam <-  sample (1: nrow(butacas),250)
butacas.test <- butacas[-entrenam,]

ventas.tree2 <- tree(High~.,butacas,subset=entrenam)
summary(ventas.tree2)
```

```
Classification tree:
tree(formula = High ~ ., data = butacas, subset = entrenam)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Advertising" "Age"   "CompPrice"   "Income"  "Population"  "Education"
Number of terminal nodes:  23
Residual mean deviance:  0.4295 = 97.51 / 227
Misclassification error rate: 0.108 = 27 / 250
```

```
plot(ventas.tree2);text(ventas.tree2,pretty =0,cex=0.7)
```



Calculemos ahora la proporción de predicciones bien hechas en base a la muestra para testear (exactitud o accuracy)

```
ventas.test.pred <- predict(ventas.tree2, butacas.test, type ="class")
aux
```

```
                High
ventas.test.pred No Yes
             No  66  18
             Yes 23  43
```

```
(aux[1,1]+aux[2,2])/sum(aux)
```

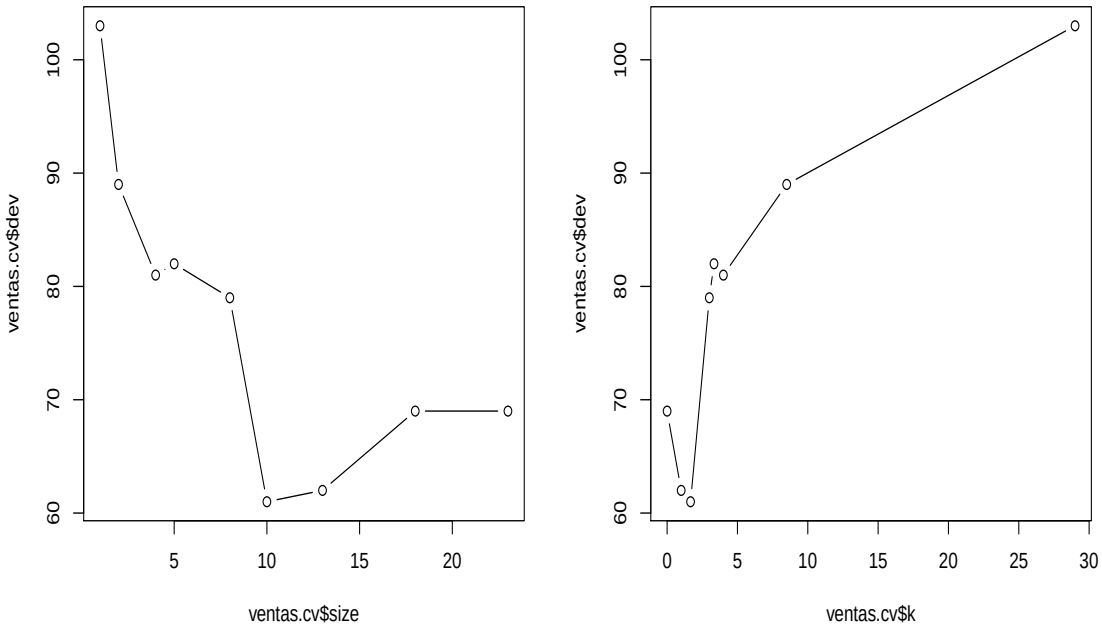
```
[1] 0.7266667
```

5. Consideremos ahora la posibilidad de encontrar un árbol con menos nodos terminales, es decir podar el árbol del ítem anterior, usando convalidación cruzada en base al error de clasificación total. Calcular su exactitud.

```
set.seed(123)
ventas.cv <- cv.tree(ventas.tree2,FUN=prune.misclass)
ventas.cv
```

```
$size
[1] 23 18 13 10  8  5  4  2  1
$dev
[1]  69  69  62  61  79  82  81  89 103
$k
[1]      -Inf  0.000000  1.000000  1.666667  3.000000  3.333333  4.000000  8.500000 29.000000
$method
[1] "misclass"
attr(,"class")
[1] "prune"         "tree.sequence"
```

```
par ( mfrow =c(1 ,2) )
plot(ventas.cv$size,ventas.cv$dev ,type ="b")
plot(ventas.cv$k ,ventas.cv$dev ,type ="b")
```




```
ventas.cv$size[which(ventas.cv$dev==min(ventas.cv$dev))]
```

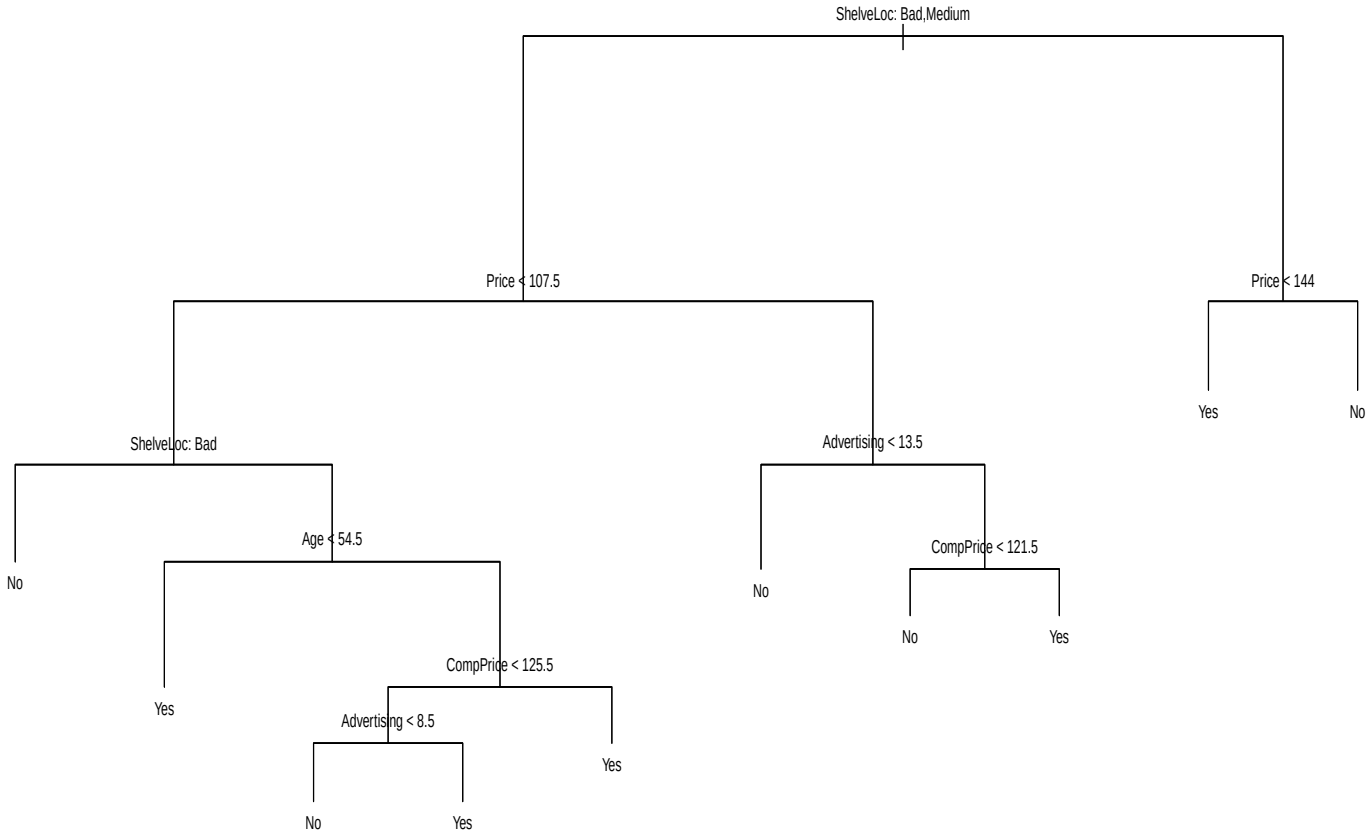
```
[1] 10
```

Podamos el árbol para que tenga 10 nodos terminales:

```
ventas.poda <- prune.misclass(ventas.tree2,best =10)
summary(ventas.poda)
```

```
Classification tree:
snip.tree(tree = ventas.tree2, nodes = c(8L, 23L, 6L, 76L, 77L, 10L))
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Age"         "CompPrice"   "Advertising"
Number of terminal nodes:  10
Residual mean deviance:  0.8251 = 198 / 240
Misclassification error rate: 0.148 = 37 / 250
```

```
plot(ventas.poda);text(ventas.poda,pretty =0,cex=0.7)
```



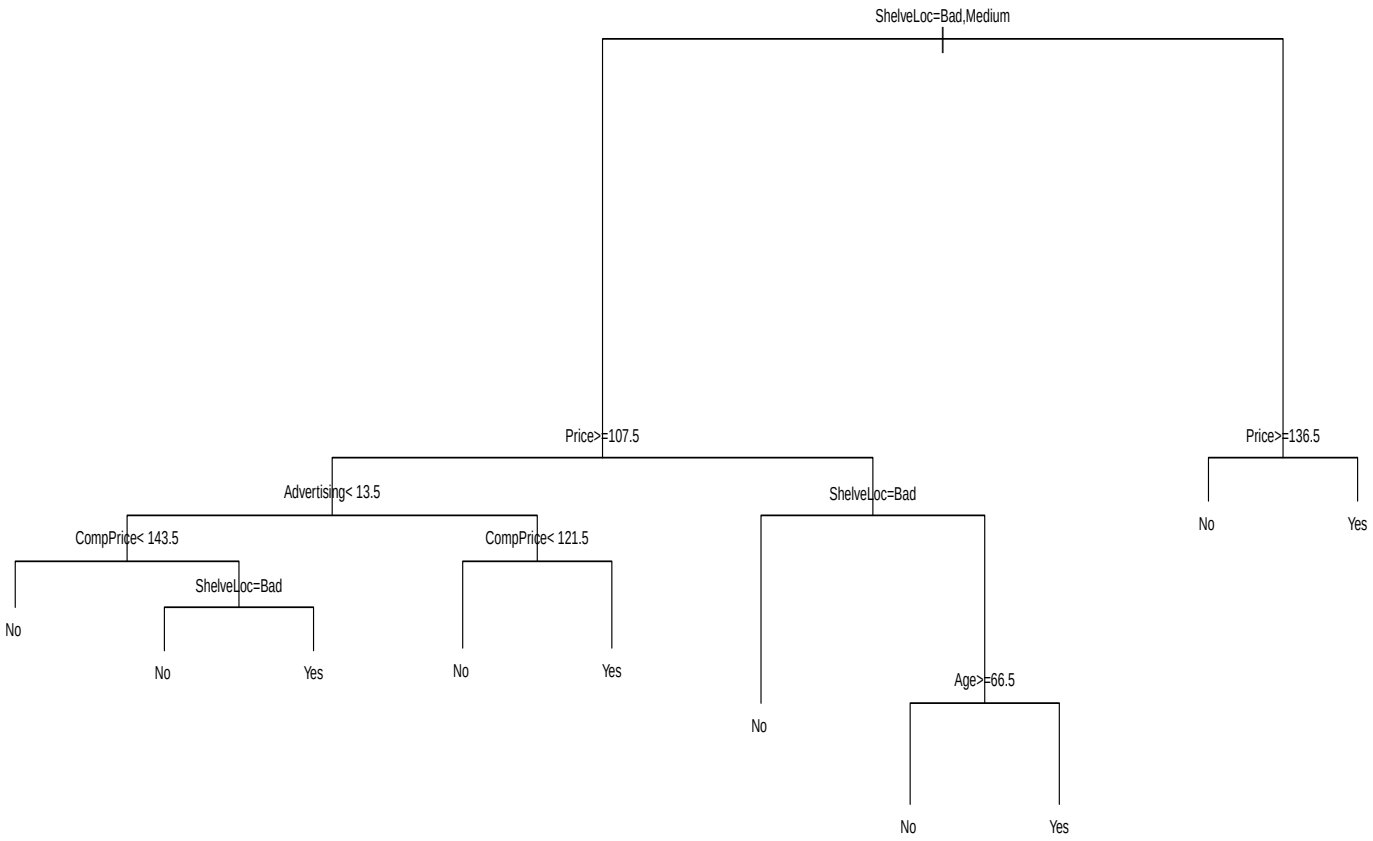
Calculemos la exactitud para compararla con el árbol maximal:

```
ventas.test.pred.poda <- predict(ventas.poda, butacas.test, type ="class")
```

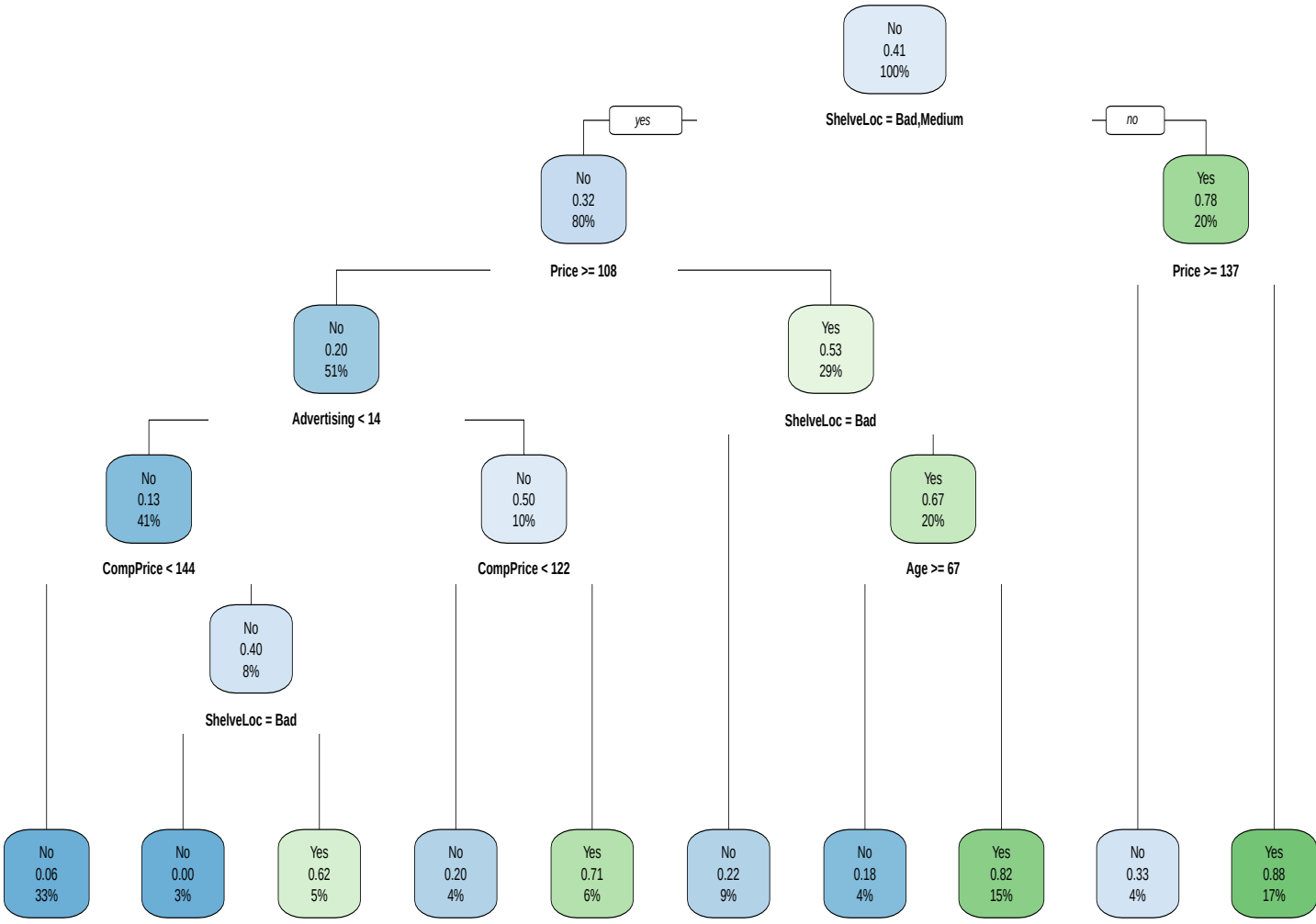
```
[1] 0.7733333
```

6. Vamos a explorar el comando, **rpart** perteneciente a la librería con el mismo nombre.

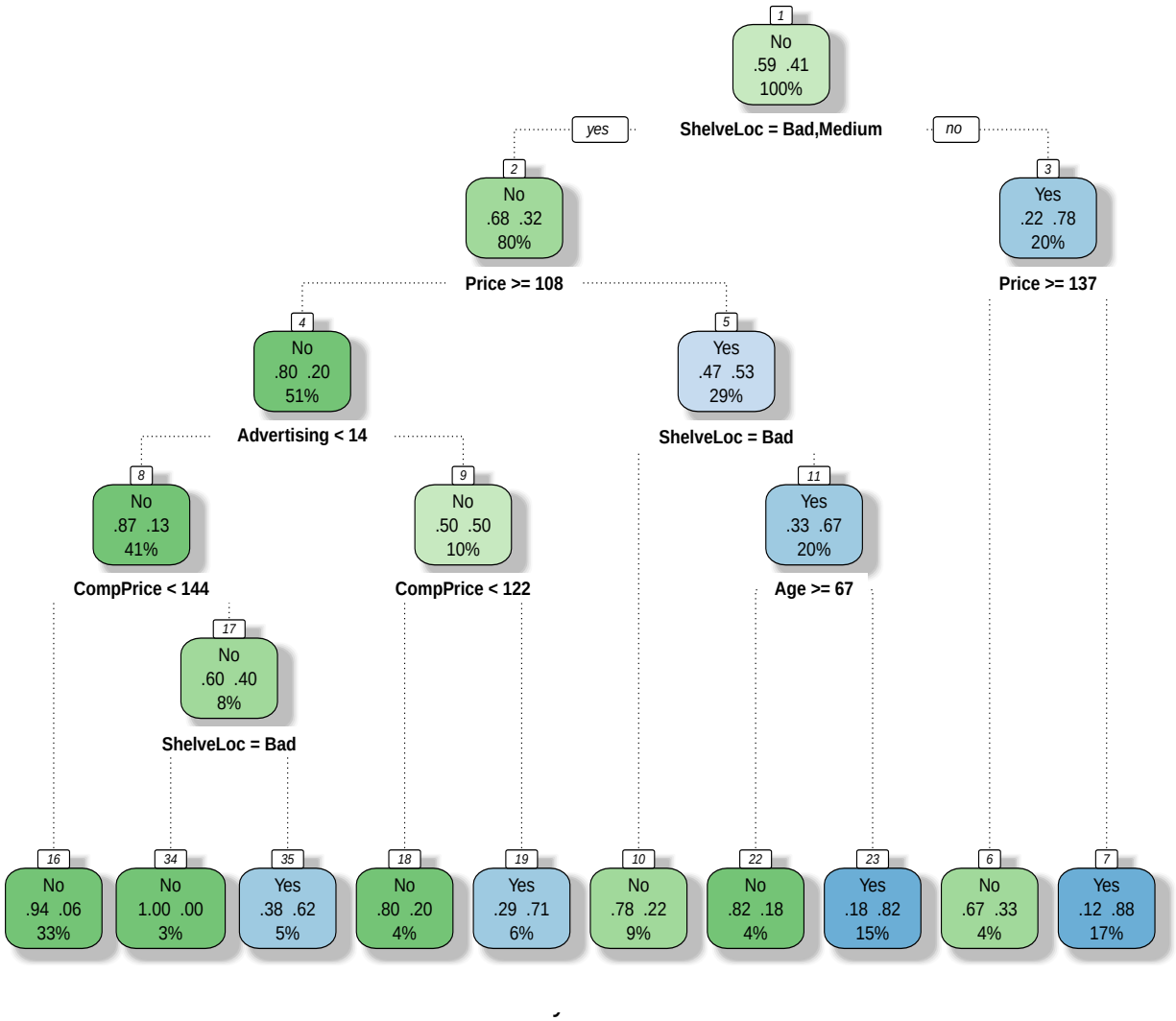
```
library(rpart)
butacas.ent <- butacas[entrenam,]

set.seed(123)
ventas.tree3 <- rpart(High~.,butacas,subset=entrenam,method="class")

plot(ventas.tree3);text(ventas.tree3,pretty =0,cex=0.7)
```




```
library(rpart.plot)
rpart.plot(ventas.tree3)
```




```
library(rattle)
fancyRpartPlot(ventas.tree3)
```



Obtengamos su exactitud:

```
ventas.test.pred.rpart <- predict(ventas.tree3, butacas.test, type ="class")
```

```
[1] 0.7133333
```

Variables de control para **rpart(formula, data, method, parms, control, ...)**:

- *formula*: permite especificar la respuesta y las variables predictoras de la forma habitual, se suele establecer de la forma respuesta $\sim$. para incluir todas las posibles variables explicativas.
- *data*: data.frame con la muestra de entrenamiento.
- *method*: método empleado para realizar las particiones, puede ser "anova" (regresión), **"class" (clasificación)** (nuestro caso de estudio), "poisson" (regresión de Poisson) o "exp" (supervivencia).
- *parms*: lista de parámetros opcionales para la partición en el caso de clasificación (o regresión de Poisson). Puede contener los componentes prior (vector de probabilidades previas; por defecto las frecuencias observadas), loss (matriz de pérdidas; con ceros en la diagonal y por defecto 1 en el resto) y split (criterio de error; por defecto "gini" o alternativamente "information").
- *control*: lista de opciones que controlan el algoritmo de partición, por defecto se seleccionan mediante la función **rpart.control**, aunque también se pueden establecer en la llamada a la función principal, y los parámetros que más no pueden llegar a interesar son: **rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01, xval = 10, maxdepth = 30, ...)**

- *cp*: es el parámetro de complejidad α para la poda del árbol, de forma que un valor de 1 se corresponde con un árbol sin divisiones y un valor de 0 con un árbol de profundidad máxima.
- *maxdepth*: es la profundidad máxima del árbol (la profundidad de la raáz sería 0).
- *minsplit y minbucket*: son, respectivamente, los números mínimos de observaciones en un nodo intermedio para particionarlo y en un nodo terminal.
- *xval*: es el número de grupos (folds) para validación cruzada.

7. Vamos a calificar mediante el método de Bagging usando el comando **randomForest**

```
library(randomForest)
set.seed(1)
ventas.bag <- randomForest(High~.,butacas,subset=entrenam,mtry=9,importance=TRUE)
ventas.bag
```

```
Call:
 randomForest(formula = High ~ ., data = butacas, mtry = 9, importance = TRUE,      subset = entrenam)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 20.4%
Confusion matrix:
     No Yes class.error
No  127  20   0.1360544
Yes  31  72   0.3009709
```

```
ventashat.bag <-  predict(ventas.bag,newdata=butacas.test)
```

```
[1] 0.82
```

Vemos que es más exacto que los anteriores.