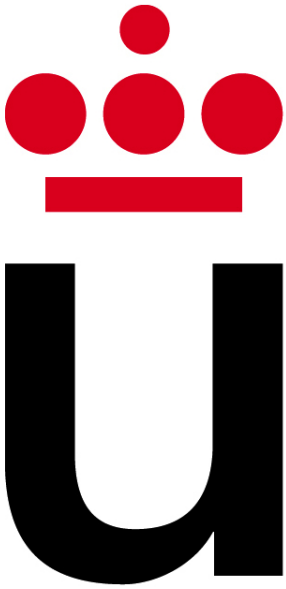
Caso Práctico IV

Minería de datos

Árboles y reglas de de decisión

José Ignacio Escribano



Móstoles, 8 de diciembre de 2015

Índice de figuras

Índice de tablas

Índice

1. Introducción

En este caso práctico utilizaremos los árboles como herramienta de clasificación. Utilizaremos dos bases de datos conocidas `iris` y `spam7`. La primera, muestra tres tipos de especies de iris a partir de distintas medidas de los pétalos y sépalos. La segunda, clasifica el spam a partir de distintas palabras o símbolos que se encuentran en los correos electrónicos.

2. Resolución de las cuestiones de evaluación

A continuación, resolvemos las cuestiones de evaluación planteadas.

2.1. Cuestión 1

En esta primera cuestión, utilizaremos la base de datos `iris`, que clasifica tres especies de iris de acuerdo a cuatro variables: longitud y anchura del sépalo, y longitud y anchura del pétalo.

Comenzamos representando el árbol de decisión para estos datos (Figura 1).

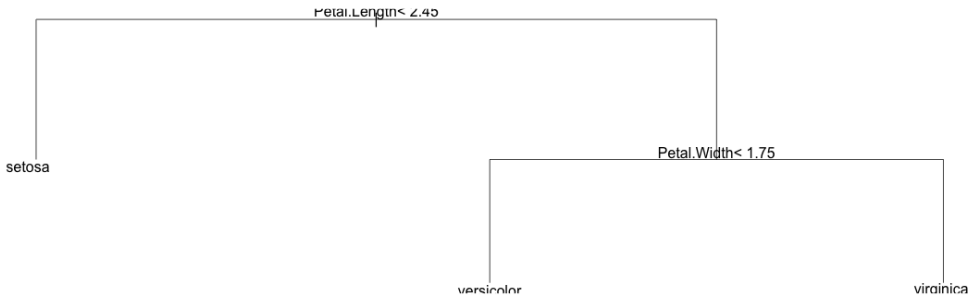


Figura 1: Árbol de decisión de la base de datos iris

Vemos que este árbol clasifica la especie setosa si la longitud del pétalo es menor que 2.45. Para las demás especies tiene en cuenta la anchura del pétalo: si es menor que 1.75 tendremos la especie versicolor, en caso contrario, la especie virginica.

Procedemos a podar este árbol. Para ello, representamos el gráfico donde se representa el número de nodos del árbol, el error cometido y la complejidad del árbol (Figura 2).

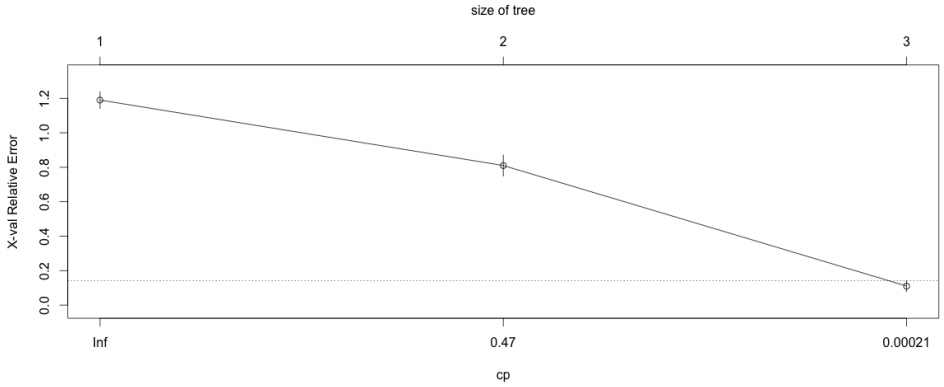


Figura 2: Error y complejidad del árbol de decisión antes de podar

Observamos que el error más pequeño se comete cuando el número de nodos del árbol es 3. Calculamos, usando R, el valor óptimo de este valor. Se tiene que el valor pedido es 10^{-7}. Con este valor ya estamos en disposición de podar el árbol (Figura 3).

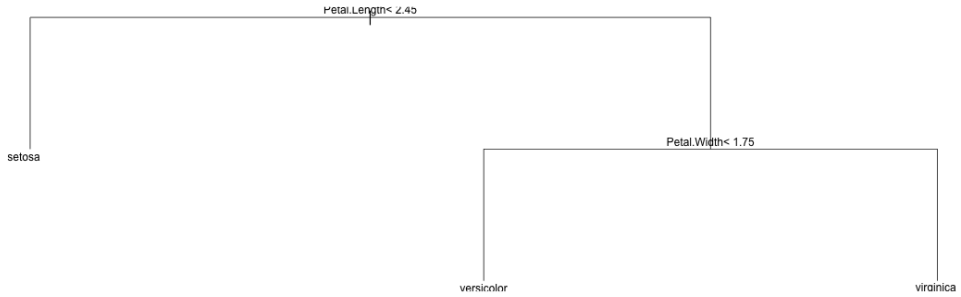


Figura 3: Árbol de decisión podado de la base de datos iris

Vemos que es el árbol podado es el mismo que sin podar, por lo que en este caso, podar no consigue un árbol más pequeño que el árbol inicial. Esta técnica de podado puede ser útil cuando el árbol tiene muchos nodos y ramas, y se quiere eliminar alguna para que sea más comprensible y además, evitar el sobreajuste.

2.2. Cuestión 2

En esta cuestión utilizaremos la base de datos `spam7` disponible en el paquete de DAAG de R. Esta base de datos consta de seis variables explicativas para medir la presencia de spam en emails: crl.tot (longitud de las palabras que están en mayúsculas), dollar (frecuencia del símbolo "$" en términos del porcentaje respecto de caracteres), bang

(frecuencia del símbolo "!" en términos del porcentaje respecto de caracteres), money (frecuencia de la palabra "money" en términos del porcentaje respecto de caracteres), n000 (frecuencia de la cadena "n000" en términos del porcentaje respecto de caracteres) y make (frecuencia de la palabra "make" en términos del porcentaje respecto de caracteres).

Calculamos el árbol de decisión usando el paquete tree de R. Un resumen del árbol se puede ver a continuación:

```
Classification tree:
tree(formula = yesno ~ ., data = spam7)
Variables actually used in tree construction:
[1] "dollar"  "bang"    "crl.tot" "money"
Number of terminal nodes:  8
Residual mean deviance:  0.7116 = 3268 / 4593
Misclassification error rate: 0.1445 = 665 / 4601
```

Vemos que este el árbol tiene 8 nodos terminales, con un error del 14.45 %, o lo que es lo mismo, clasifica de forma incorrecta 665 de los 4601 emails. La estructura del árbol es la siguiente:

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 4601 6170.00 n ( 0.60596 0.39404 )
   2) dollar < 0.0555 3471 3786.00 n ( 0.76491 0.23509 )
     4) bang < 0.0875 2407 1571.00 n ( 0.89946 0.10054 ) *
     5) bang > 0.0875 1064 1468.00 y ( 0.46053 0.53947 )
      10) crl.tot < 85.5 541  681.10 n ( 0.67652 0.32348 ) *
      11) crl.tot > 85.5 523  572.90 y ( 0.23709 0.76291 )
        22) bang < 0.5155 342  440.70 y ( 0.34503 0.65497 )
          44) money < 0.04 262  360.20 y ( 0.44656 0.55344 ) *
          45) money > 0.04 80   10.75 y ( 0.01250 0.98750 ) *
        23) bang > 0.5155 181   52.68 y ( 0.03315 0.96685 ) *
   3) dollar > 0.0555 1130  818.80 y ( 0.11770 0.88230 )
     6) bang < 0.0775 275  355.80 y ( 0.34909 0.65091 )
      12) money < 0.025 167  231.00 n ( 0.52695 0.47305 ) *
      13) money > 0.025 108   57.04 y ( 0.07407 0.92593 ) *
     7) bang > 0.0775 855  304.70 y ( 0.04327 0.95673 ) *
```

El árbol utiliza las variables dollar, bang, crl.tot y money. El árbol se puede ver en la Figura 4.

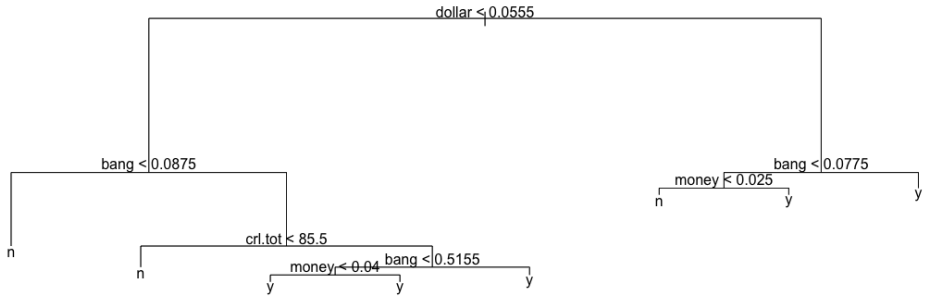


Figura 4: Árbol de decisión de la base de datos spam7

3. Conclusiones

En este caso práctico hemos visto cómo crear árboles de decisión y podarlos, si fuera necesario. Los árboles de decisión son una técnica de minería de datos muy potente, ya que es fácilmente entendible por las personas, y muy fácil de programar el modelo obtenido.

R nos ha permitido ahorrar mucho tiempo debido a las librerías que implementan árboles. Existen diferentes librerías que implementan los árboles de decisión, por lo que antes de utilizar una u otra será necesario conocer la documentación y los algoritmos que implementa cada una de ellas, de acuerdo al problema que estemos tratando.

4. Código R utilizado

```
#-----------------------------------------------------------------
# Cuestiones de evaluación
#-----------------------------------------------------------------

#-----------------------------------------------------------------
# Cuestión 1
#-----------------------------------------------------------------

# Cargamos las librerías
library(datasets)
library(rpart)

# Calculamos el árbol sin podar
iris.tree <- rpart(
  formula = Species ~ Sepal.Length + Sepal.Width +
    Petal.Length + Petal.Width, method="class", data=iris)

# Representamos el árbol sin podar
plot(iris.tree)
text(iris.tree)

# Calculamos el árbol ajustando el coeficiente de complejidad
iris.tree2 <- rpart(
  formula = Species ~ Sepal.Length + Sepal.Width +
    Petal.Length + Petal.Width, method="class",
    data=iris, cp=0.0000001)

# Representamos el gráfico con el error, el número de nodos
# y el coeficiente de complejidad
plotcp(iris.tree2)

# Calculamos el cp óptimo para del árbol, antes de podar
cpar <- iris.tree2$cptable[
  which.min(iris.tree2$cptable[,"xerror"]), "CP"]

# Podamos el árbol con el valor de cp que acabamos de calcular
iris.tree3 <- prune(iris.tree2, cp=cpar)

# Representamos el árbol
plot(iris.tree3, uniform=TRUE)
text(iris.tree3, cex=0.75)
```

```
#-----------------------------------------------------------------
# Cuestión 2
#-----------------------------------------------------------------

# Cargamos las librerías
library(tree)
library(DAAG)

# Calculamos el árbol
spam7.tr <- tree(yesno ~ .,  spam7)

# Representamos el árbol
plot(spam7.tr)
text(spam7.tr)

# Resumen del árbol calculado
summary(spam7.tr)
```