Ayudantía 5: Árboles de Clasificación (en R)

Profesor Nicolás Rojas
Francisca Ramírez - francisca.ramirez.12@sansano.usm.cl
Felipe Vega - felipe.vega.14@sansano.usm.cl

1. Antes de Comenzar

Es necesario instalar R antes de seguir con esta ayudantía, puedes encontrarlo en el siguiente link. También se recomienda el uso de R Studio como entorno de desarrollo.

También es necesario descargar el dataset a utilizar en esta ayudantía, el cual se encuentra en el siguiente link.

2. Librería

Antes de comenzar es necesario instalar el paquete a utilizar para crear los árboles de clasificación, utilizando el siguiente comando desde la terminal de R.

```
install.packages("tree")
```

O si utilizas *R Studio* puedes instalarlo seleccionando **Tools** ← **Install Packages** en la barra de herramientas. Una vez instalado, hay que importarlo con el siguiente comando.

```
library(tree)
```

3. Manos a la obra

3.1. Lectura de los datos

Primero debemos cargar los datos a utilizar en memoria, para esto utilizamos el siguiente comando.

```
data <- read.table("heart_disease_dataset.csv", header=TRUE, sep=",")
```

Lo anterior guarda en la variable *data* una tabla con el dataset. El primer parámetro es la ruta al archivo con los datos, mientras los otros dos le indican si el archivo tiene encabezados y el símbolo que utiliza para separar cada columna.

Para ver los datos de una determinada columna pueden utilizar el siguiente comando.

```
variable_con_los_datos$nombre_columna_a_visualizar
```

Y para ver el número de registros por cada valor de una columna se puede utilizar el siguiente comando.

```
xtabs(~ nombre_columna, data=variale_con_los_datos)
```

Por ejemplo:

```
xtabs(~ num, data=data)
```

Muestra lo siguiente

num	
0	1
164	139

Cuadro 1: Cantidad de registros por cada variable a predecir en el problema

También podemos verlo en un gráfico de barras.

```
    counts <- table(data$num)
    barplot(counts, main="Hearth Disease")
```

Figura 1: Gráfico de barras resultante.

3.2. Train - Test Split

Algo importante cuando se trabaja con técnicas de aprendizaje es separar nuestro conjunto de datos en al menos dos subconjuntos de datos: uno de entrenamiento,para entrenar nuestros modelos; y uno de pruebas, para probar el desempeño de este.

Eso lo realizamos de la siguiente manera.

```
# Queremos obtener siempre lo mismo
set.seed(42)
# En este caso utilizaremos un conjunto de entrenamiento del 80% de los datos
train_size <- floor(0.80 * nrow(data))
train_mask <- sample(seq_len(nrow(data)),size = train_size)
# Separamos nuestros conjuntos
train <- data[train_mask, ]
test <- data[-train_mask, ]
```

Lo anterior guarda en la variable *train* el 80 % de los datos y en *test* el 20 % restante.

3.3. Ármando el árbol

Es momento de armar el árbol, pero antes de eso hay que arreglar un detalle del cual me acabo de dar cuenta: hay que pasar la columna *num* (variable a predecir) a categórica.

```
    data$num = as.factor(data$num)
```

Ahora, utilizamos el método *tree* de la librería para construir nuestro árbol.

```
    arbol = tree(num ~ age + sex + cp + trestbps +
            chol + fbs + restecg + thalach + exang +
            oldpeak + slope + ca + thal, data = train)
```

Utilizaremos dos parámetros para el método *tree*. El primero se conoce como fórmula, donde la primera columna que aparece el la columna que queremos predecir, luego sigue el símbolo $\sim$ para luego colocar todas las columnas que queremos utilizar para construir el árbol. Si se quiere excluir una columna es cosa de no agregarla en la formula. El segundo parámetro es el conjunto de datos que se utiliza para entrenar el árbol.

Luego, podemos ver un resumen del árbol de la siguiente manera.

```
    summary(arbol)
```

```
Classification tree:
tree(formula = num ~ age + sex + cp + trestbps + chol + fbs +
    restecg + thalach + exang + oldpeak + slope + ca + thal,
    data = train)
Variables actually used in tree construction:
[1] "thal"     "cp"       "age"      "sex"      "oldpeak"  "trestbps" "ca"
[8] "chol"     "thalach"
Number of terminal nodes:  18
Residual mean deviance:  0.485 = 108.6 / 224
Misclassification error rate: 0.1322 = 32 / 242
```

Figura 2: Resumen del árbol

El cual muestra la fórmula del árbol, las variables que se utilizaron para realizar las particiones, el error de clasificación, entre otras.

Finalmente, podemos ver una imagen de nuestro árbol con el siguiente comando.

```
plot(arbol)
text(arbol, pretty = 1)
```

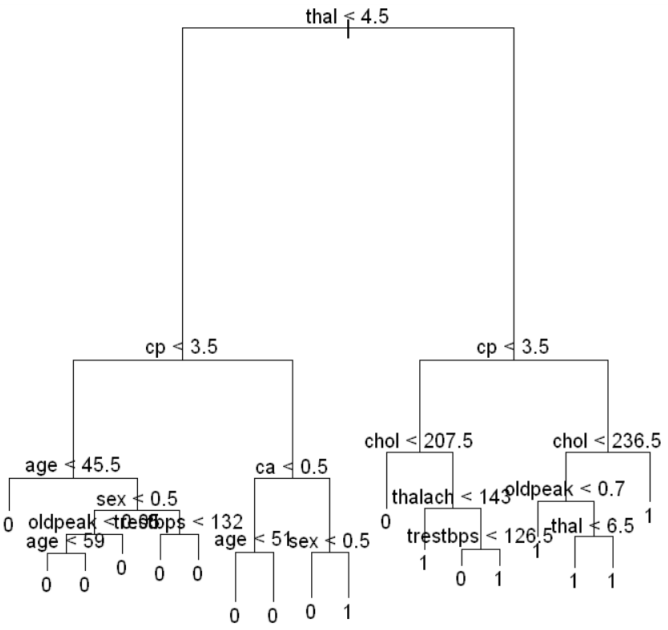

Figura 3: Árbol generado

En cada nodo, si se cumple la condición se sigue la rama de la izquierda, caso contrario la de la derecha.

3.4. Discretización

Si se fijan, el árbol anterior tiene una partición que separa según cuando el sexo sea mayor o menor a 0,5, lo cual es incorrecto, pues solo tiene valore 0 o 1, el por esto que debemos cambiar el tipo de las variables para que efectivamente sean categóricas (ver la sección Dataset para ver las distintas columnas).

```
data$sex = as.factor(data$sex)
data$cp = as.factor(data$cp)
data$fbs = as.factor(data$fbs)
data$restecg = as.factor(data$restecg)
data$exang = as.factor(data$exang)
data$slope = as.factor(data$slope)
data$ca = as.factor(data$ca)
data$thal = as.factor(data$thal)
```

Si repetimos el procedimiento se genera el siguiente árbol.

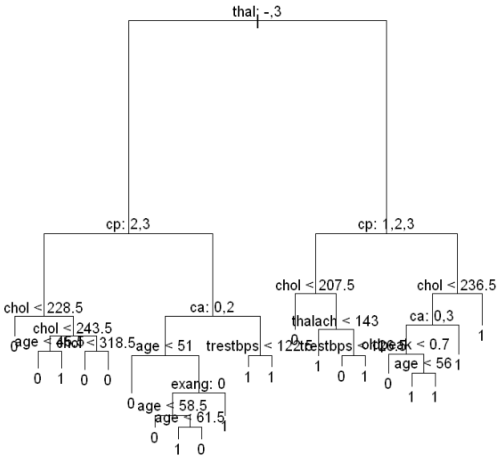

Figura 4: Árbol generado

3.5. Viendo el árbol más a fondo

Con los siguientes comandos pueden ver más detalles del árbol, incluyendo cuantos datos existen en cada partición, la probabilidad de cada clase en cada partición y la predicción que se haría en cada nodo de ser un nodo terminal.

```
arbol
```

```
arbol$frame
```

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

  1) root 242 335.200 0 ( 0.51653 0.48347 )
    2) thal: -100000,3 134 149.600 0 ( 0.75373 0.24627 )
      4) cp: 2,3 76  41.980 0 ( 0.92105 0.07895 )
        8) chol < 228.5 31   0.000 0 ( 1.00000 0.00000 ) *
        9) chol > 228.5 45  35.340 0 ( 0.86667 0.13333 )
         18) chol < 243.5 12  15.280 0 ( 0.66667 0.33333 )
           36) age < 45.5 5   0.000 0 ( 1.00000 0.00000 ) *
           37) age > 45.5 7   9.561 1 ( 0.42857 0.57143 ) *
```

Figura 5: Resultado (incompleto).

	var	n	dev	yval	splits	yprob
	<fct>	<dbl>	<dbl>	<fct>	<chr[,2]>	<dbl[,2]>
1	thal	242	335.218724	0	:ab , :cd	0.51652893, 0.48347107
2	cp	134	149.597223	0	:bc , :ad	0.75373134, 0.24626866
4	chol	76	41.981020	0	<228.5, >228.5	0.92105263, 0.07894737
8	<leaf>	31	0.000000	0	,	1.00000000, 0.00000000

Figura 6: Resultado (incompleto).

3.6. Evaluación

Finalmente, evaluamos el comportamiento de nuestro árbol en el conjunto de pruebas. Para esto, primero necesitamos predecir las clases en dicho conjunto, para luego crear una matriz de confusión.

```r
#Predecimos los valores
pred = predict(arbol,test,type="class")
#Matriz de confusion
conf_matrix <- with(test,table(pred,test$num))
conf_matrix
```

```
pred  0  1
   0 32  6
   1  7 16
```

Figura 7: Matriz de Confusión

Esta matriz nos muestra los valores predichos (filas) y los valores reales (columna). En este caso, cuando el modelo dijo que la clase era 0 (hay enfermedad) acertó 32 veces y se equivocó 6, y cuando dijo que la clase era 1 acertó 16 veces y falló 7.

Con esto es posible calcular el *accuracy*, sumando la diagonal (aciertos del modelo) y dividiendolo por la cantidad de datos del conjunto de pruebas. Finalmente, el error de clasificación se puede calcular como $1 - accuracy$.

```r
acc <- sum(diag(conf_matrix))/nrow(test)
```

```
acc
miss_class <- 1 - acc
miss_class
```

PD: Si corren este último código y lo comparan con el error de clasificación en el set de entrenamiento se darán cuenta que el error en el conjunto de test es mucho mayor. Esto último indica que el modelo está sobreajustado.

4. Dataset

El dataset que se utilizará en esta ayudantía consiste en 303 datos sobre pacientes que pueden o no padecer una enfermedad al corazón. Está compuesto por 14 atributos los cuales se detallan a continuación.

- **age**: Edad del paciente.
- **sex**: Sexo del paciente. 1 = masculino; 0 = femenino.
- **cp**: Tipo de dolor al pecho, tiene 4 posibles valores enteros describiendo tipos de dolores posibles.
- **trestbps**: Presión de la sangre en reposo.
- **chol**: Medida del colesterol.
- **fbs**: Azucar en la sangre. 1 si es mayor a 120 $\frac{mg}{dl}$; 0 en caso contrario.
- **restecg**: Resultados de un electrocardiograma, tiene 4 posibles valores enteros.
- **thalach**: Ritmo cardiaco máximo registrado.
- **exang**: 1 si presenta *exercise induced angina*; 0 si no.
- **oldpeak**: *ST depression*: Valor que aparece de *ST depression* en un electrocardiograma.
- **slope**: Forma del peak del *ST depression*, tiene 3 posibles valores enteros.
- **ca**: Número de *major vessels*.
- **thal**: Variable categórica con 3 posibles valores: 3 = normal; 6 = defecto irreversible; 7 = defecto reversible.
- **num**: Variable a precedir: 0 = enfermedad cardiaca; 1 = no hay enfermedad.

PD: Los términos que aparecen en inglés fueron cosas médicas que no supe traducir.