Práctica 0

Guillermo Palomo y Miguel Díaz-Plaza

2022-10-15

Tabla de contenidos


```
library(mlr3data)
library(skimr)
library(corrplot)
```

EDA

```
data("ilpd", package = "mlr3data")

skim(ilpd)
```

Data summary

Name	ilpd
Number of rows	583
Number of columns	11

Column type frequency:	
factor	2

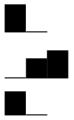
numeric	9

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	Mal: 441, Fem: 142
diseased	0	1	FALSE	2	yes: 416, no: 167

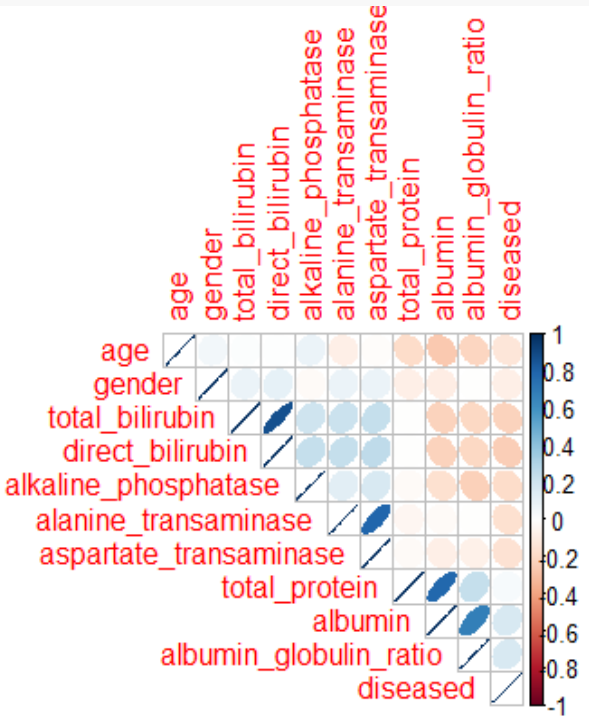
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	44.75	16.19	4.0	33.0	45.00	58.0	90.0	
total_bilirubin	0	1	3.30	6.21	0.4	0.8	1.00	2.6	75.0	
direct_bilirubin	0	1	1.49	2.81	0.1	0.2	0.30	1.3	19.7	
alkaline_phosphatase	0	1	290.58	242.94	63.0	175.5	208.00	298.0	2110.0	
alanine_transaminase	0	1	80.71	182.62	10.0	23.0	35.00	60.5	2000.0	
aspartate_transaminase	0	1	109.91	288.92	10.0	25.0	42.00	87.0	4929.0	
total_protein	0	1	6.48	1.09	2.7	5.8	6.60	7.2	9.6	
albumin	0	1	3.14	0.80	0.9	2.6	3.10	3.8	5.5	
albumin_globulin_ratio	0	1	0.95	0.32	0.3	0.7	0.95	1.1	2.8	

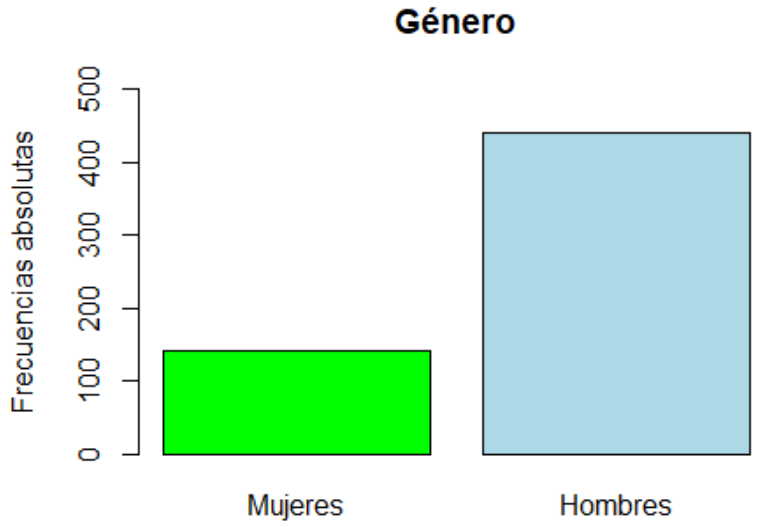
```
str(ilpd)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ age                   : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ gender                : Factor w/ 2 levels "Female","Male": 1 2 2 2
2 2 1 1 2 2 ...
##  $ total_bilirubin       : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7
...
##  $ direct_bilirubin      : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2
...
##  $ alkaline_phosphatase  : int  187 699 490 182 195 208 154 202 202
290 ...
##  $ alanine_transaminase  : int  16 64 60 14 27 19 16 14 22 53 ...
```

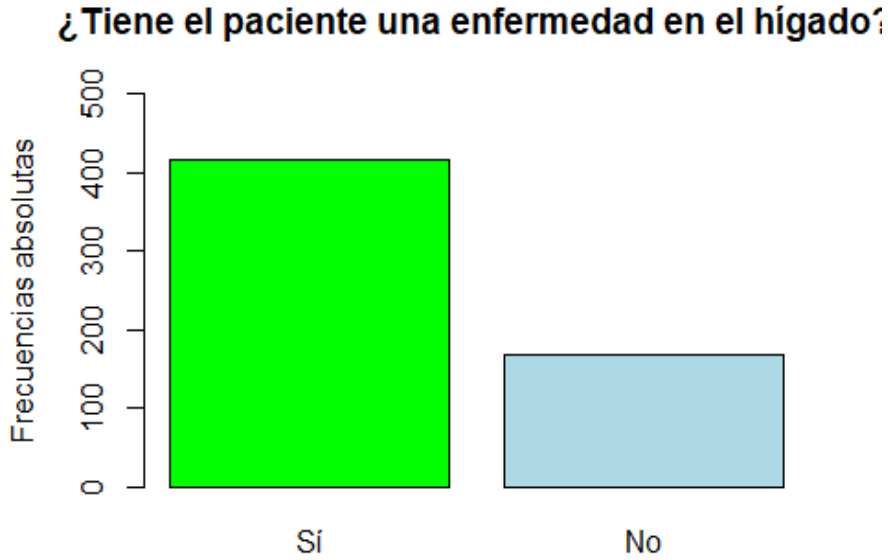
```
##  $ aspartate_transaminase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ total_protein         : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8
...
##  $ albumin               : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1
3.4 ...
##  $ albumin_globulin_ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1
...
##  $ diseased              : Factor w/ 2 levels "yes","no": 1 1 1 1 1 1
1 1 2 1 ...

#TRANSFORMAMOS VARIABLES PARA QUE FUNCIONE EL C50

ilpd$age <- as.numeric(ilpd$age)
ilpd$alkaline_phosphatase <- as.numeric(ilpd$alkaline_phosphatase)
ilpd$alanine_transaminase <- as.numeric(ilpd$alanine_transaminase)
ilpd$aspartate_transaminase <- as.numeric(ilpd$aspartate_transaminase)

ilpd1 <- ilpd


ilpd1$gender <- as.numeric(ilpd1$gender)
ilpd1$diseased <-as.numeric(ilpd1$diseased)


corrplot(cor(ilpd1),method = "ellipse", type = "upper")
```



```r
plot(ilpd$gender,names.arg = c("Mujeres","Hombres"),col =
c("green","lightblue"),ylim = c(0,500),main = paste("Género"),ylab =
"Frecuencias absolutas")
```




```r
plot(ilpd$diseased,names.arg = c("Sí","No"),col =
c("green","lightblue"),ylim = c(0,500),main = paste("¿Tiene el paciente
una enfermedad en el hígado?"),ylab = "Frecuencias absolutas")
```



Disponemos de un dataset de 583 **observaciones** x 11 **variables** sin ningún dato faltante.

Con las gráficas mostradas podemos ver como el género en la base de datos se divide entre 142 **mujeres** y 441 **hombres**. Además, el número de pacientes **con enfermedad de hígado es de** 416 y el de pacientes **sin enfermedad de hígado es de** 167.

La variable respuesta `diseased` es categórica por lo que tenemos un **problema de clasificación**, y vemos que tiene muchas más observaciones de enfermos que de sanos. La mayoría de variables son numéricas y enteros, excepto la variable `gender` que es categórica.

Estas variables no parecen estar muy correladas en general según la matriz de correlaciones.

En los histogramas vemos que `age`,`total_protein` y `albumin` siguen unas distribuciones bastante normales, mientras que el resto acumulan la gran mayoría de sus observaciones en los valores más bajos.

En este apartado, hemos transformado las variables enteras a numéricas para poder utilizar más adelante el C5.0 sin problemas.

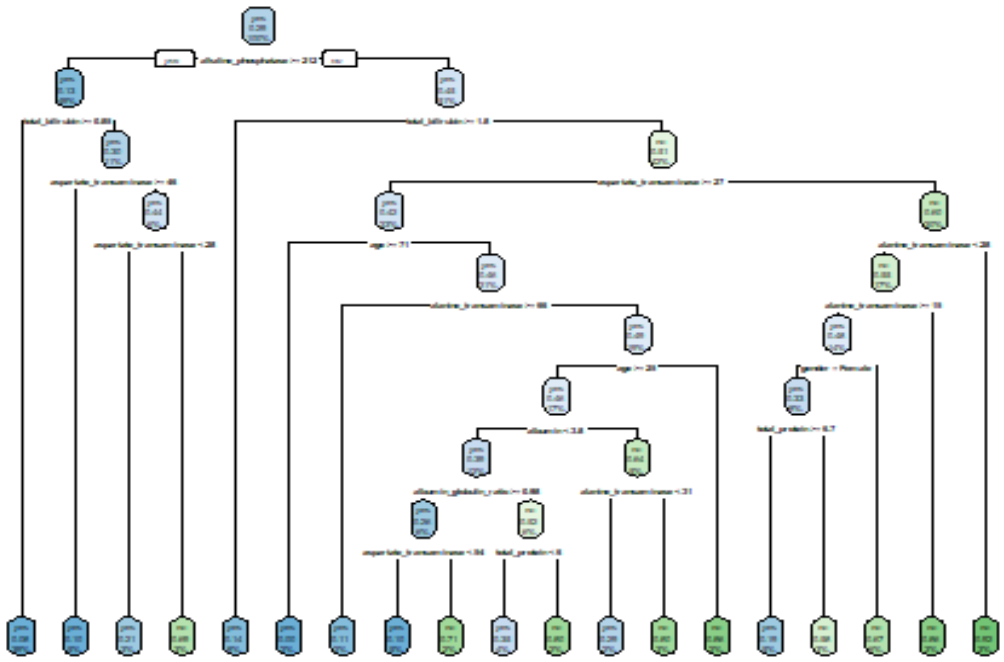
MÉTODOS

rpart con R

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

# Separamos en entrenamiento y test
set.seed(0)
indices_train_R_rpart <- sample(1:nrow(ilpd), nrow(ilpd)*3/4,
replace=FALSE) #1/4 test y 3/4 entrenamiento

ilpd_train_R_rpart <- ilpd[indices_train_R_rpart,]
ilpd_test_R_rpart <- ilpd[-indices_train_R_rpart,]

# Entrenamos modelo
set.seed(0)
R_rpart_model <-rpart(diseased~.,ilpd_train_R_rpart, method = "class")

# Obtenemos las predicciones
R_rpart_test <- predict(R_rpart_model, ilpd_test_R_rpart, type = "class")

# Calculamos la accuracy
R_rpart_accuracy <-
sum(ilpd_test_R_rpart$diseased==R_rpart_test)/length(R_rpart_test)
R_rpart_accuracy

## [1] 0.6712329
```

```
# Visualizamos modelo
R_rpart_model

## n= 437
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 437 127 yes (0.70938215 0.29061785)
##     2) alkaline_phosphatase>=211.5 216  28 yes (0.87037037 0.12962963)
##       4) total_bilirubin>=0.85 169  14 yes (0.91715976 0.08284024) *
##       5) total_bilirubin< 0.85 47  14 yes (0.70212766 0.29787234)
##        10) aspartate_transaminase>=45.5 20   2 yes (0.90000000
0.10000000) *
##        11) aspartate_transaminase< 45.5 27  12 yes (0.55555556
0.44444444)
##          22) aspartate_transaminase< 27.5 14   3 yes (0.78571429
0.21428571) *
##          23) aspartate_transaminase>=27.5 13   4 no (0.30769231
0.69230769) *
##     3) alkaline_phosphatase< 211.5 221  99 yes (0.55203620 0.44796380)
##       6) total_bilirubin>=1.75 36   5 yes (0.86111111 0.13888889) *
##       7) total_bilirubin< 1.75 185  91 no (0.49189189 0.50810811)
##        14) aspartate_transaminase>=26.5 99  42 yes (0.57575758
0.42424242)
##          28) age>=70.5 7   0 yes (1.00000000 0.00000000) *
##          29) age< 70.5 92  42 yes (0.54347826 0.45652174)
##            58) alanine_transaminase>=66 9   1 yes (0.88888889
0.11111111) *
##            59) alanine_transaminase< 66 83  41 yes (0.50602410
0.49397590)
##             118) age>=24.5 76  35 yes (0.53947368 0.46052632)
##               236) albumin< 3.75 54  21 yes (0.61111111 0.38888889)
##                 472) albumin_globulin_ratio>=0.98 27   7 yes
(0.74074074 0.25925926)
##                   944) aspartate_transaminase< 54 20   2 yes
(0.90000000 0.10000000) *
##                   945) aspartate_transaminase>=54 7   2 no (0.28571429
0.71428571) *
##                 473) albumin_globulin_ratio< 0.98 27  13 no
(0.48148148 0.51851852)
##                   946) total_protein< 5.95 17   6 yes (0.64705882
0.35294118) *
##                   947) total_protein>=5.95 10   2 no (0.20000000
0.80000000) *
##               237) albumin>=3.75 22   8 no (0.36363636 0.63636364)
##                 474) alanine_transaminase< 31 7   2 yes (0.71428571
0.28571429) *
##                 475) alanine_transaminase>=31 15   3 no (0.20000000
0.80000000) *
```

```
##             119) age< 24.5 7   1 no (0.14285714 0.85714286) *
##         15) aspartate_transaminase< 26.5 86  34 no (0.39534884
0.60465116)
##           30) alanine_transaminase< 27.5 74  33 no (0.44594595
0.55405405)
##             60) alanine_transaminase>=14.5 60  29 yes (0.51666667
0.48333333)
##               120) gender=Female 33  11 yes (0.66666667 0.33333333)
##                 240) total_protein>=6.65 21   4 yes (0.80952381
0.19047619) *
##                 241) total_protein< 6.65 12   5 no (0.41666667
0.58333333) *
##               121) gender=Male 27   9 no (0.33333333 0.66666667) *
##             61) alanine_transaminase< 14.5 14   2 no (0.14285714
0.85714286) *
##           31) alanine_transaminase>=27.5 12   1 no (0.08333333
0.91666667) *

# Representamos el árbol
rpart.plot(R_rpart_model)
```



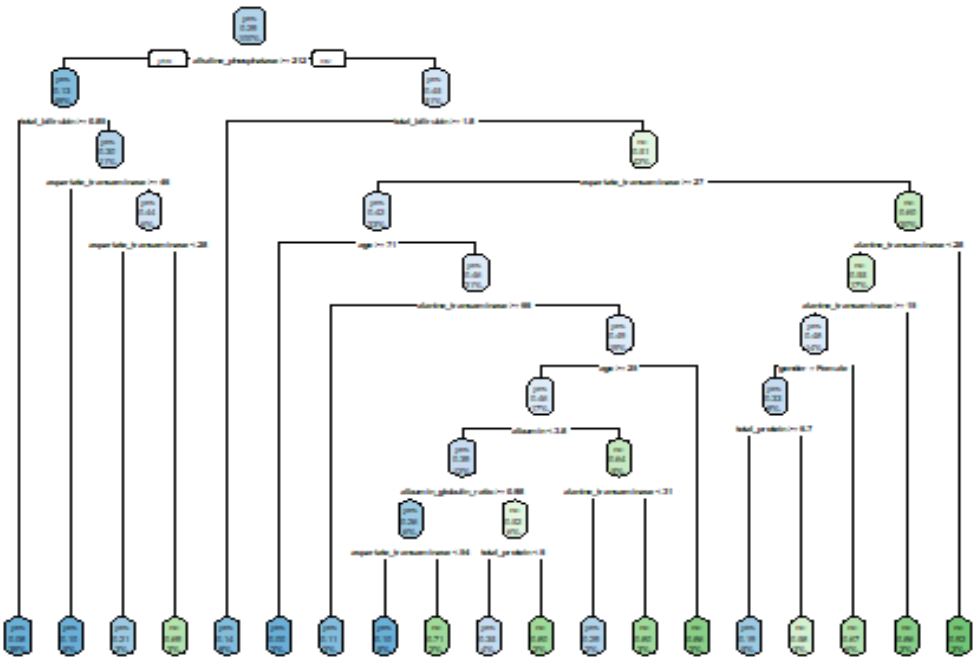
rpart CON MLR

```
library(mlr3)

library(mlr3learners)
remotes::install_github("mlr-org/mlr3extralearners")
```

```r
library(mlr3extralearners)


# Creamos la tarea
ilpd_task <- as_task_classif(ilpd, target="diseased", id="enfermos")

# Definimos un método de evaluación
res_desc_mlr3_rpart <- rsmp("holdout", ratio=3/4)
set.seed(0)
res_desc_mlr3_rpart$instantiate(ilpd_task)

# Definimos el método de aprendizaje
tree_learner_mlr3_rpart <- lrn("classif.rpart")

# Entrenamos y evaluamos el modelo con resample
set.seed(0)
tree_resample_mlr3_rpart <- resample(task=ilpd_task,
                         learner=tree_learner_mlr3_rpart,
                         resampling=res_desc_mlr3_rpart,
                         store_models = TRUE)

## INFO  [10:13:22.429] [mlr3] Applying learner 'classif.rpart' on task
'enfermos' (iter 1/1)

# Calculamos la accuracy con resample
tree_accuracy_rmse_mlr3_rpart <-
tree_resample_mlr3_rpart$aggregate(msr("classif.acc"))
tree_accuracy_rmse_mlr3_rpart

## classif.acc
##   0.6712329

# Visualizamos el modelo
tree_learner_mlr3_rpart <- tree_resample_mlr3_rpart$learners[[1]]
tree_learner_mlr3_rpart$model

## n= 437
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 437 127 yes (0.70938215 0.29061785)
##     2) alkaline_phosphatase>=211.5 216  28 yes (0.87037037 0.12962963)
##       4) total_bilirubin>=0.85 169  14 yes (0.91715976 0.08284024) *
##       5) total_bilirubin< 0.85 47  14 yes (0.70212766 0.29787234)
##        10) aspartate_transaminase>=45.5 20   2 yes (0.90000000
0.10000000) *
##        11) aspartate_transaminase< 45.5 27  12 yes (0.55555556
0.44444444)
##          22) aspartate_transaminase< 27.5 14   3 yes (0.78571429
0.21428571) *
```

```
##          23) aspartate_transaminase>=27.5 13   4 no (0.30769231
0.69230769) *
##     3) alkaline_phosphatase< 211.5 221  99 yes (0.55203620 0.44796380)
##       6) total_bilirubin>=1.75 36   5 yes (0.86111111 0.13888889) *
##       7) total_bilirubin< 1.75 185  91 no (0.49189189 0.50810811)
##        14) aspartate_transaminase>=26.5 99  42 yes (0.57575758
0.42424242)
##          28) age>=70.5 7   0 yes (1.00000000 0.00000000) *
##          29) age< 70.5 92  42 yes (0.54347826 0.45652174)
##            58) alanine_transaminase>=66 9   1 yes (0.88888889
0.11111111) *
##            59) alanine_transaminase< 66 83  41 yes (0.50602410
0.49397590)
##             118) age>=24.5 76  35 yes (0.53947368 0.46052632)
##               236) albumin< 3.75 54  21 yes (0.61111111 0.38888889)
##                 472) albumin_globulin_ratio>=0.98 27   7 yes
(0.74074074 0.25925926)
##                   944) aspartate_transaminase< 54 20   2 yes
(0.90000000 0.10000000) *
##                   945) aspartate_transaminase>=54 7   2 no (0.28571429
0.71428571) *
##                 473) albumin_globulin_ratio< 0.98 27  13 no
(0.48148148 0.51851852)
##                   946) total_protein< 5.95 17   6 yes (0.64705882
0.35294118) *
##                   947) total_protein>=5.95 10   2 no (0.20000000
0.80000000) *
##               237) albumin>=3.75 22   8 no (0.36363636 0.63636364)
##                 474) alanine_transaminase< 31 7   2 yes (0.71428571
0.28571429) *
##                 475) alanine_transaminase>=31 15   3 no (0.20000000
0.80000000) *
##             119) age< 24.5 7   1 no (0.14285714 0.85714286) *
##        15) aspartate_transaminase< 26.5 86  34 no (0.39534884
0.60465116)
##          30) alanine_transaminase< 27.5 74  33 no (0.44594595
0.55405405)
##            60) alanine_transaminase>=14.5 60  29 yes (0.51666667
0.48333333)
##             120) gender=Female 33  11 yes (0.66666667 0.33333333)
##               240) total_protein>=6.65 21   4 yes (0.80952381
0.19047619) *
##               241) total_protein< 6.65 12   5 no (0.41666667
0.58333333) *
##             121) gender=Male 27   9 no (0.33333333 0.66666667) *
##            61) alanine_transaminase< 14.5 14   2 no (0.14285714
0.85714286) *
##          31) alanine_transaminase>=27.5 12   1 no (0.08333333
0.91666667) *
```

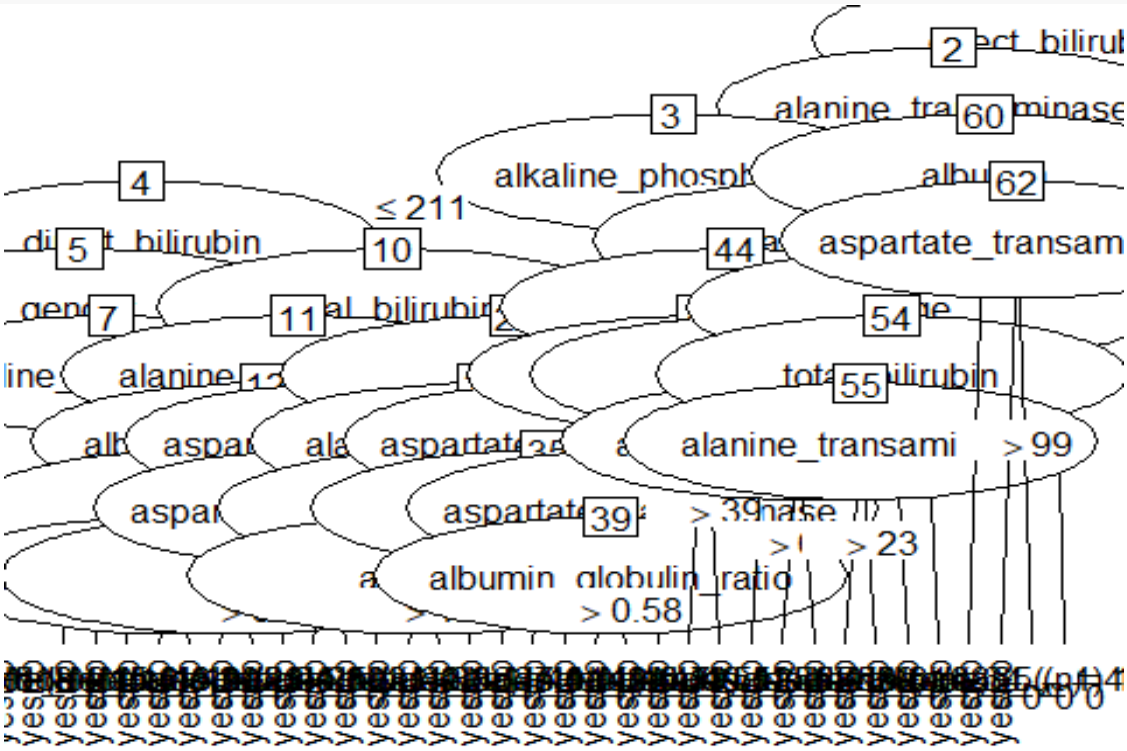
```
# Representamos el árbol
rpart.plot(tree_learner_mlr3_rpart$model)
```



C5.0 con R

```
library(C50)

# Separamos entre entrenamiento y test
set.seed(0)
indices_train_R_C5.0 <- sample(1:nrow(ilpd), nrow(ilpd)*3/4,
replace=FALSE) #1/4 test y 3/4 entrenamiento

ilpd_train_R_C5.0 <- ilpd[indices_train_R_C5.0,]
ilpd_test_R_C5.0 <- ilpd[-indices_train_R_C5.0,]

# Entrenamos modelo
set.seed(0)
R_C5.0_model <-C5.0(diseased~.,ilpd_train_R_C5.0)

# Obtenemos las predicciones
R_C5.0_test <- predict(R_C5.0_model, ilpd_test_R_C5.0, type = "class")

# Calculamos la accuracy
R_C5.0_accuracy <-
sum(ilpd_test_R_C5.0$diseased==R_C5.0_test)/length(R_C5.0_test)
R_C5.0_accuracy
```

```
## [1] 0.6986301

# Visualizamos modelo
R_C5.0_model

##
## Call:
## C5.0.formula(formula = diseased ~ ., data = ilpd_train_R_C5.0)
##
## Classification Tree
## Number of samples: 437
## Number of predictors: 10
##
## Tree size: 33
##
## Non-standard options: attempt to group attributes

# Representamos el árbol
plot(R_C5.0_model)
```



C5.0 con MLR

```
# Creamos la tarea
ilpd_task <- as_task_classif(ilpd, target="diseased", id="enfermos")

# Definimos un método de evaluación
```

```
res_desc_mlr3_C5.0 <- rsmp("holdout", ratio=3/4)
set.seed(0)
res_desc_mlr3_C5.0$instantiate(ilpd_task)

# Definimos el método de aprendizaje
tree_learner_mlr3_C5.0 <- lrn("classif.C50")

# Entrenamos y evaluamos el modelo con resample
set.seed(0)
tree_resample_mlr3_C5.0 <- resample(task=ilpd_task,
                          learner=tree_learner_mlr3_C5.0,
                          resampling=res_desc_mlr3_C5.0,
                          store_models = TRUE)

## INFO  [10:13:26.957] [mlr3] Applying learner 'classif.C50' on task
'enfermos' (iter 1/1)

# Calculamos el error con resample
tree_accuracy_rmse_mlr3_C5.0 <-
tree_resample_mlr3_C5.0$aggregate(msr("classif.acc"))
tree_accuracy_rmse_mlr3_C5.0

## classif.acc
##   0.6986301

#Visualizamos el modelo
tree_learner_mlr3_C5.0 <- tree_resample_mlr3_C5.0$learners[[1]]
tree_learner_mlr3_C5.0$model

##
## Call:
## C50::C5.0.formula(formula = f, data = data, control = ctrl)
##
## Classification Tree
## Number of samples: 437
## Number of predictors: 10
##
## Tree size: 35
##
## Non-standard options: attempt to group attributes
```

CONCLUSIONES

Diferencias fundamentales entre el código R para rpart y para C5.0

`rpart` y `C5.0` son dos librerías distintas, que se cargan, respectivamente, con las librerías `library(rpart)` y `library(C50)`.

En ambas separamos la muestra entre entrenamiento y test de la misma forma.

La primera diferencia viene obviamente en cómo entrenamos el modelo. En `rpart` utilizamos la función `rpart()`, y por su parte, en `C5.0` utilizamos `C5.0()`.

Con respecto a las predicciones, aquí observamos una gran diferencia en los resultados, ya que no coinciden.

Esto se ve reflejado en la accuracy, ya que no coincide. De hecho, es más alta con `C5.0` que es de 0.6986301, mientras que la accuracy con `rpart` es de 0. 6712329 . El código de R no presenta cambios.

Observamos que el modelo que genera R de `C5.0` es mucho más sencillo y la información viene más resumida que en el modelo de `rpart` (por ejemplo, en el de `C5.0` aparece el tamaño del árbol a simple vista, que es de 33, y en el de `rpart` no). También, el método `C5.0` carga más rápido.

Por último, la representación del árbol sí que es más visible e intuitiva en `rpart`, aunque hemos tenido que instalar la librería `rpart.plot` y utilizar la función `rpart.plot()`.

Desarrollar las diferencias entre el código MLR (rpart y C5.0) y el código R (rpart y C5.0)

El código MLR necesita de la instalación de las librerías `library(mlr3)`, `library(mlr3learners)` y `library(mlr3extralearners)`.

En MLR, creamos una tarea en la que indicamos la variable respuesta y en la que especificamos que es un problema de clasificación.

Consideramos que la principal diferencia es cómo definimos el método de evaluación. Para ello utilizamos `holdout`, que es una forma de resamplear que nos separa entre entrenamiento y test. Además, con MLR, definimos un método de aprendizaje con `lrn("classif.rpart")` y `lrn("classif.C50")` y entrenamos y evaluamos el modelo con resample. Esto hace más sencillo separar entre entrenamiento y test.

La visualización del modelo con MLR (para `rpart`) nos sale igual a la que hacemos con el código R con `rpart`. Por su parte, el modelo con MLR de `C5.0` (tamaño del árbol 35) aparece similar que con código R de `C5.0` (tamaño del árbol 33). La única pega al MLR de `C5.0` es que no hay cómo graficar el árbol como en el resto de los métodos.

Por su parte, la accuracy con MLR no varía de la que se obtiene con código R.

Por tanto, podemos decir que MLR nos aporta simplicidad en el código con respecto a R y hace mucho más automáticas algunas tareas, que pueden ser tediosas sin usar estas librerías.

Las accuracies(o ce) de test. ¿Coinciden en R y en MLR? ¿Qué método es mejor?

Las accuracies y errores de test **coinciden en R y MLR**, pero son **diferentes entre `rpart` y `C5.0`**, siendo **mejor** la librería **`C5.0`** según esta métrica.

```
set.seed(0)
Accuracies<-data.frame(c(R_rpart_accuracy,R_C5.0_accuracy,tree_accuracy_r
mse_mlr3_rpart,tree_accuracy_rmse_mlr3_C5.0))
```

```
rownames(Accuracies)=c("R rpart","R C5.0","mlr3 rpart","mlr3 C5.0")
colnames(Accuracies)=c("Accuracies")
Accuracies
```

```
##            Accuracies
## R rpart     0.6712329
## R C5.0      0.6986301
## mlr3 rpart  0.6712329
## mlr3 C5.0   0.6986301
```

Al ser un caso de detección de enfermos, preferimos que el modelo prediga falsos positivos a negativos. Para hallar qué método es mejor según este punto de vista, utilizamos las matrices de confusión.

```
table(ilpd_test_R_rpart$diseased,R_rpart_test)
```

```
##      R_rpart_test
##       yes no
##   yes  84 22
##   no   26 14
```

```
table(ilpd_test_R_C5.0$diseased,R_C5.0_test)
```

```
##      R_C5.0_test
##       yes no
##   yes  92 14
##   no   30 10
```

Según las tablas, el mejor método es el C5.0 ya que proporciona menos falsos negativos 14 < 22.

Los 5 primeros índices usados en la partición de entrenamiento y los 5 primeros de la partición de test. ¿Coinciden en R y en MLR?

Sí coinciden en todos. Tanto con código R (para `rpart` y `C5.0`), como con MLR (para `rpart` y `C5.0`) ya que hemos usado para todos la misma semilla, y si fueran diferentes, los modelos entre R y MLR no coincidirían como hemos visto.

```
head(indices_train_R_rpart,5)
```

```
## [1] 398 129 509 471 299
```

```
head(indices_train_R_C5.0,5)
```

```
## [1] 398 129 509 471 299
```

Las 5 primeras predicciones de los datos de test. ¿Coinciden en R y en MLR?

Las predicciones de los datos de test **coinciden en R y MLR**, pero son **diferentes entre `rpart` y `C5.0`**

```
head(R_rpart_test,5)

##   4   9  11  12  13
##  no  no yes yes yes
## Levels: yes no

head(R_C5.0_test,5)

## [1] no  yes yes yes yes
## Levels: yes no
```