Minería de datos: PEC3 - Clasificación con árboles de decisión

Autor: Gabriel Patricio Bonilla Sanchez

Diciembre 2020

Contents

Introducción

Presentación

Esta prueba de evaluación continua cubre los Módulos 3 (Clasificación: árboles de decisión) y el Módulo 8 (Evaluación de modelos) del programa de la asignatura.

Competencias

Las competencias que se trabajan en esta prueba son:

- Uso y aplicación de las TIC en el ámbito académico y profesional.
- Capacidad para innovar y generar nuevas ideas.
- Capacidad para evaluar soluciones tecnológicas y elaborar propuestas de proyectos teniendo en cuenta los recursos, las alternativas disponibles y las condiciones de mercado.
- Conocer las tecnologías de comunicaciones actuales y emergentes así como saberlas aplicar convenientemente para diseñar y desarrollar soluciones basadas en sistemas y tecnologías de la información.
- Aplicación de las técnicas específicas de ingeniería del software en las diferentes etapas del ciclo de vida de un proyecto.
- Capacidad para aplicar las técnicas específicas de tratamiento, almacenamiento y administración de datos.
- Capacidad para proponer y evaluar diferentes alternativas tecnológicas para resolver un problema concreto.

Objetivos

La correcta asimilación del Módulo 3. En esta PEC trabajaremos la generación e interpretación de un árbol de decisión con el software de prácticas. Seguiremos también con la preparación de los datos y la extracción inicial de conocimiento.

Descripción de la PEC a realizar

La prueba está estructurada en un total de un único ejercicio práctico.

Recursos Básicos

Material docente proporcionado por la UOC.

Módulo 3 y 8 del material didáctico.

Complementarios

- Los descritos para la anterior PEC.
- Fichero titanic.csv
- R package C5.0 (Decision Trees and Rule-Based Models): https://cran.r-project.org/web/packages/C50/index.html

Criterios de valoración

Todos los ejercicios deben ser presentados de forma razonada y clara, especificando todos y cada uno de los pasos que se hayan llevado a cabo para su resolución. No se aceptará ninguna respuesta que no esté claramente justificada.

Formato y fecha de entega

El formato de entrega es: usernameestudiant-PECn.html/doc/docx/odt/pdf. Se recomienda la entrega en formato html y también el Rmd que genera el html entregado. Fecha de Entrega: 18/12/2019. Se debe entregar la PEC en el buzón de entregas del aula.

Nota: Propiedad intelectual

> A menudo es inevitable, al producir una obra multimedia, hacer uso de recursos creados por terceras personas. Es por lo tanto comprensible hacerlo en el marco de una práctica de los estudios de Informática, Multimedia y Telecomunicación de la UOC, siempre y cuando esto se documente claramente y no suponga plagio en la práctica.
>
> Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se debe presentar junto con ella un documento en qué se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar dónde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por copyright.
>
> Deberéis, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.

Enunciado

En este ejercicio vamos a seguir los pasos del ciclo de vida de un proyecto de minería de datos, para el caso de un algoritmo de clasificación y más concretamente un árbol de decisión. Lo haremos con el archivo titanic.csv, que se encuentra adjunto en el aula. Este archivo contiene un registro por cada pasajero que viajaba en el Titanic. En las variables se caracteriza si era hombre o mujer, adulto o menor (niño), en qué categoría viajaba o si era miembro de la tripulación.

Objetivos:

- Estudiar los datos, por ejemplo: ¿Número de registros del fichero? ¿Distribuciones de valores por variables? ¿Hay campos mal informados o vacíos?
- Preparar los datos. En este caso ya están en el formato correcto y no es necesario discretizar ni generar atributos nuevos. Hay que elegir cuáles son las variables que se utilizarán para construir el modelo y cuál es la variable que clasifica. En este caso la variable por la que clasificaremos es el campo de si el pasajero sobrevivió o no.
- Instalar, si es necesario, el paquete C5.0 Se trata de una implementación más moderna del algoritmo ID3 de Quinlan. Tiene los principios teóricos del ID3 más la poda automática. Con este paquete generar un modelo de minería.
- ¿Cuál es la calidad del modelo?
- Generar el árbol gráfico.
- Generar y extraer las reglas del modelo.
- En función del modelo, el árbol y las reglas: ¿Cuál es el conocimiento que obtenemos?
- Probar el modelo generado presentándole nuevos registros. ¿Clasifica suficientemente bien?

Revisión de los datos, extracción visual de información y preparación de los datos

Carga de los datos:

```
data<-read.csv("./titanic.csv",header=T,sep=",")
attach(data)
```

Empezaremos haciendo un breve análisis de los datos ya que nos interesa tener una idea general de los datos que disponemos. Por ello, primero calcularemos las dimensiones de nuestra base de datos y analizaremos qué tipos de atributos tenemos.

Para empezar, calculamos las dimensiones de la base de datos mediante la función dim(). Obtenemos que disponemos de 2201 registros o pasajeros (filas) y 4 variables (columnas).

```
dim(data)
```

```
## [1] 2201    4
```

¿Cuáles son esas variables? Gracias a la función str() sabemos que las cuatro variables son categóricas o discretas, es decir, toman valores en un conjunto finito. La variable CLASS hace referencia a la clase en la que viajaban los pasajeros (1ª, 2ª, 3ª o crew), AGE determina si era adulto o niño (Adulto o Menor), la variable SEX si era hombre o mujer (Hombre o Mujer) y la última variable (SURVIVED) informa si el pasajero murió o sobrevivió en el accidente (Muere o Sobrevive).

```
str(data)
```

```
## 'data.frame':    2201 obs. of  4 variables:
##  $ CLASS   : chr  "1a" "1a" "1a" "1a" ...
##  $ AGE     : chr  "Adulto" "Adulto" "Adulto" "Adulto" ...
##  $ SEX     : chr  "Hombre" "Hombre" "Hombre" "Hombre" ...
##  $ SURVIVED: chr  "Sobrevive" "Sobrevive" "Sobrevive" "Sobrevive" ...
```

Es de gran interés saber si tenemos muchos valores nulos (campos vacíos) y la distribución de valores por variables. Es por ello recomendable empezar el análisis con una visión general de las variables. Mostraremos para cada atributo la cantidad de valores perdidos mediante la función summary.

```
summary(data)
```

```
##     CLASS               AGE                SEX              SURVIVED
##  Length:2201        Length:2201        Length:2201        Length:2201
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

Disponemos por tanto de un data frame formado por cuatro variables categóricas sin valores nulos. Para un conocimiento mayor sobre los datos, tenemos a nuestro alcance unas herramientas muy valiosas: las herramientas de visualización. Para dichas visualizaciones, haremos uso de los paquetes ggplot2, gridExtra y grid de R.

```
if(!require(ggplot2)){
    install.packages('ggplot2', repos='http://cran.us.r-project.org')
    library(ggplot2)
}
```

```
## Loading required package: ggplot2
```

```
if(!require(grid)){
    install.packages('grid', repos='http://cran.us.r-project.org')
    library(grid)
}
```

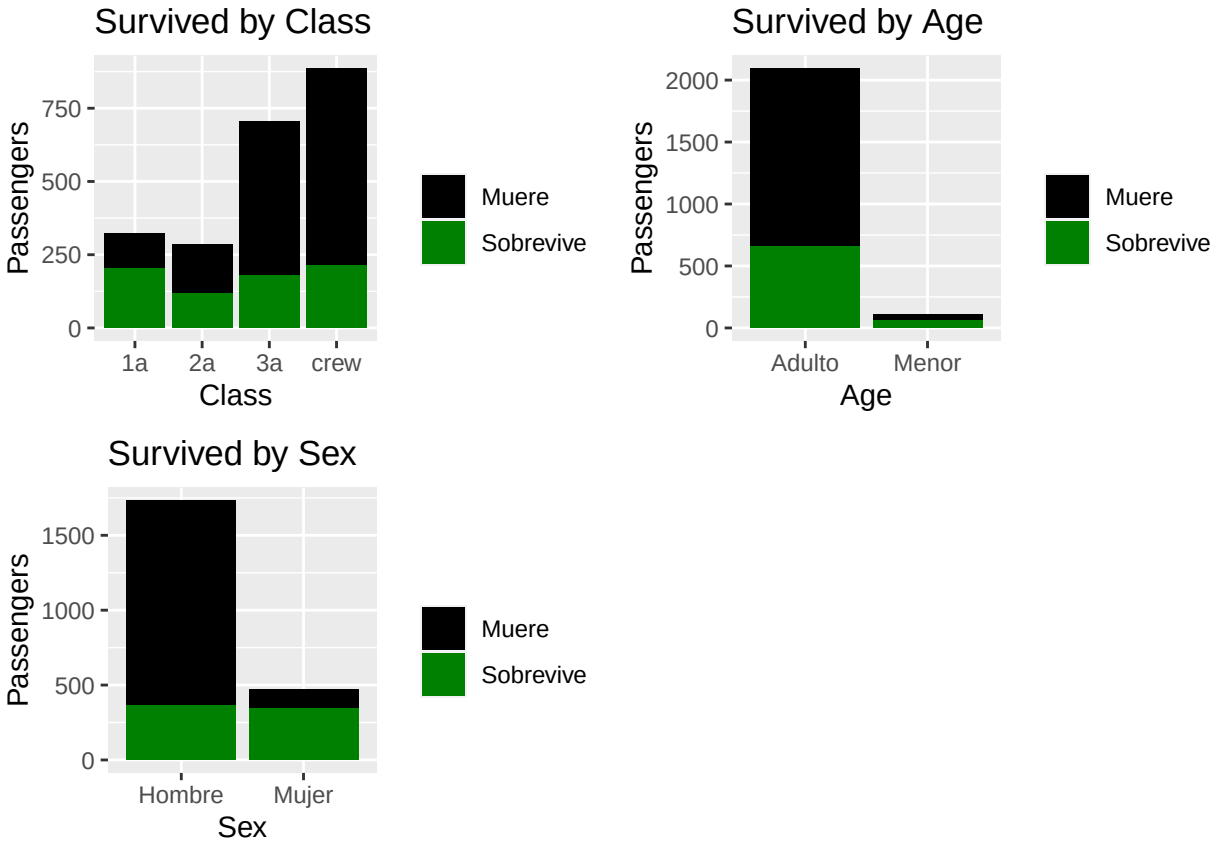
```
## Loading required package: grid
```

```
if(!require(gridExtra)){
    install.packages('gridExtra', repos='http://cran.us.r-project.org')
    library(gridExtra)
}
```

```
## Loading required package: gridExtra
```

Nos interesa describir la relación entre la supervivencia y cada uno de las variables mencionadas anteriormente. Para ello, por un lado graficaremos mediante diagramas de barras la cantidad de muertos y supervivientes según la clase en la que viajaban, la edad o el sexo. Por otro lado, para obtener los datos que estamos graficando utilizaremos el comando table para dos variables que nos proporciona una tabla de contingencia.

```
grid.newpage()
plotbyClass<-ggplot(data,aes(CLASS,fill=SURVIVED))+geom_bar() +labs(x="Class", y="Passengers")+ guides(
plotbyAge<-ggplot(data,aes(AGE,fill=SURVIVED))+geom_bar() +labs(x="Age", y="Passengers")+ guides(fill=gu
plotbySex<-ggplot(data,aes(SEX,fill=SURVIVED))+geom_bar() +labs(x="Sex", y="Passengers")+ guides(fill=gu
grid.arrange(plotbyClass,plotbyAge,plotbySex,ncol=2)
```



De estos gráficos obtenemos información muy valiosa que complementamos con las tablas de contingencia (listadas abajo). Por un lado, la cantidad de pasajeros que sobrevivieron es similar en hombres y mujeres (hombres: 367 y mujeres 344). No, en cambio, si tenemos en cuenta el porcentaje respecto a su sexo. Es decir, pese a que la cantidad de mujeres y hombres que sobrevivieron es pareja, viajaban más hombres que mujeres (470 mujeres y 1731 hombres), por lo tanto, la tasa de muerte en hombres es muchísimo mayor (el 78,79% de los hombres murieron mientras que en mujeres ese porcentaje baja a 26,8%).

En cuanto a la clase en la que viajaban, los pasajeros que viajaban en primera clase fueron los únicos que el porcentaje de supervivencia era mayor que el de mortalidad. El 62,46% de los viajeros de primera clase sobrevivió, el 41,4% de los que viajaban en segunda clase mientras que de los viajeros de tercera y de la tripulación solo sobrevivieron un 25,21% y 23,95% respectivamente. Para finalizar, destacamos que la presencia de pasajeros adultos era mucho mayor que la de los niños (2092 frente a 109) y que la tasa de supervivencia en niños fue mucho mayor (52,29% frente a 31,26%), no podemos obviar, en cambio, que los únicos niños que murieron fueron todos pasajeros de tercera clase (52 niños).

```
tabla_SST <- table(SEX, SURVIVED)
tabla_SST
```

```
##          SURVIVED
## SEX       Muere Sobrevive
##   Hombre  1364       367
##   Mujer    126       344
```

```
prop.table(tabla_SST, margin = 1)
```

```
##          SURVIVED
```

```
## SEX          Muere Sobrevive
##   Hombre 0.7879838 0.2120162
##   Mujer  0.2680851 0.7319149
```

```
tabla_SCT <- table(CLASS,SURVIVED)
tabla_SCT
```

```
##       SURVIVED
## CLASS  Muere Sobrevive
##   1a     122       203
##   2a     167       118
##   3a     528       178
##   crew   673       212
```

```
prop.table(tabla_SCT, margin = 1)
```

```
##       SURVIVED
## CLASS      Muere Sobrevive
##   1a   0.3753846 0.6246154
##   2a   0.5859649 0.4140351
##   3a   0.7478754 0.2521246
##   crew 0.7604520 0.2395480
```

```
tabla_SAT <- table(AGE,SURVIVED)
tabla_SAT
```

```
##         SURVIVED
## AGE      Muere Sobrevive
##   Adulto  1438       654
##   Menor     52        57
```

```
prop.table(tabla_SAT, margin = 1)
```

```
##         SURVIVED
## AGE          Muere Sobrevive
##   Adulto 0.6873805 0.3126195
##   Menor  0.4770642 0.5229358
```

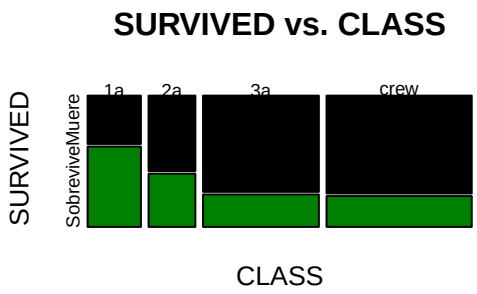
```
tabla_SAT.byClass <- table(AGE,SURVIVED,CLASS)
tabla_SAT.byClass
```

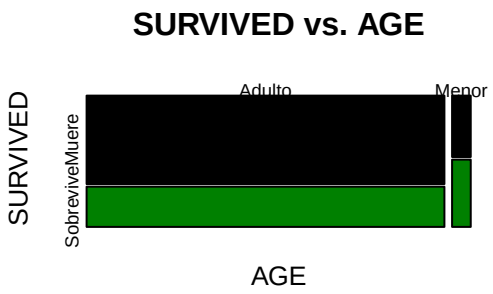
```
## , , CLASS = 1a
##
##         SURVIVED
## AGE      Muere Sobrevive
##   Adulto   122       197
##   Menor      0         6
##
## , , CLASS = 2a
##
```

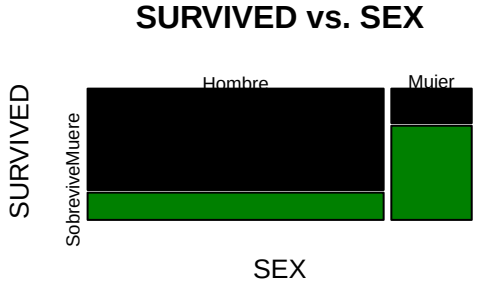
```
##          SURVIVED
## AGE       Muere Sobrevive
##   Adulto   167        94
##   Menor      0        24
##
## , , CLASS = 3a
##
##          SURVIVED
## AGE       Muere Sobrevive
##   Adulto   476       151
##   Menor     52        27
##
## , , CLASS = crew
##
##          SURVIVED
## AGE       Muere Sobrevive
##   Adulto   673       212
##   Menor      0         0
```

Una alternativa interesante a las barras de diagramas, es el plot de las tablas de contingencia. Obtenemos la misma información pero para algunos receptores puede resultar más visual.

```
par(mfrow=c(2,2))
plot(tabla_SCT, col = c("black","#008000"), main = "SURVIVED vs. CLASS")
plot(tabla_SAT, col = c("black","#008000"), main = "SURVIVED vs. AGE")
plot(tabla_SST, col = c("black","#008000"), main = "SURVIVED vs. SEX")
```







Nuestro objetivo es crear un árbol de decisión que permita analizar qué tipo de pasajero del Titanic tenía probabilidades de sobrevivir o no. Por lo tanto, la variable por la que clasificaremos es el campo de si el pasajero sobrevivió o no. De todas maneras, al imprimir las primeras (con head) y últimas 10 (con tail) filas nos damos cuenta de que los datos están ordenados.

```
head(data,10)
```

```
##    CLASS    AGE    SEX  SURVIVED
## 1     1a Adulto Hombre Sobrevive
## 2     1a Adulto Hombre Sobrevive
## 3     1a Adulto Hombre Sobrevive
## 4     1a Adulto Hombre Sobrevive
## 5     1a Adulto Hombre Sobrevive
## 6     1a Adulto Hombre Sobrevive
## 7     1a Adulto Hombre Sobrevive
## 8     1a Adulto Hombre Sobrevive
## 9     1a Adulto Hombre Sobrevive
## 10    1a Adulto Hombre Sobrevive
```

```
tail(data,10)
```

```
##      CLASS    AGE   SEX  SURVIVED
## 2192  crew Adulto Mujer Sobrevive
## 2193  crew Adulto Mujer Sobrevive
## 2194  crew Adulto Mujer Sobrevive
## 2195  crew Adulto Mujer Sobrevive
## 2196  crew Adulto Mujer Sobrevive
## 2197  crew Adulto Mujer Sobrevive
## 2198  crew Adulto Mujer Sobrevive
## 2199  crew Adulto Mujer     Muere
## 2200  crew Adulto Mujer     Muere
## 2201  crew Adulto Mujer     Muere
```

Nos interesará "desordenarlos". Guardaremos los datos con el nuevo nombre como "data_random".

```
set.seed(1)
data_random <- data[sample(nrow(data)),]
```

Para la futura evaluación del árbol de decisión, es necesario dividir el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba. El conjunto de entrenamiento es el subconjunto del conjunto original de datos utilizado para construir un primer modelo; y el conjunto de prueba, el subconjunto del conjunto original de datos utilizado para evaluar la calidad del modelo.

Lo más correcto será utilizar un conjunto de datos diferente del que utilizamos para construir el árbol, es decir, un conjunto diferente del de entrenamiento. No hay ninguna proporción fijada con respecto al número relativo de componentes de cada subconjunto, pero la más utilizada acostumbra a ser 2/3 para el conjunto de entrenamiento y 1/3, para el conjunto de prueba.

La variable por la que clasificaremos es el campo de si el pasajero sobrevivió o no, que está en la cuarta columna.

```
set.seed(666)
y <- data_random[,4]
X <- data_random[,1:3]
```

Podemos elegir el subconjunto de entrenamiento y de prueba de diversas maneras. La primer opción consiste en calcular a cuántas filas corresponde dos tercios de los datos (2*2201/3=1467) y dividir "manualmente" el conjunto.

```
trainX <- X[1:1467,]
trainy <- y[1:1467]
testX <- X[1468:2201,]
testy <- y[1468:2201]
```

En la segunda opción podemos crear directamente un rango.

```
indexes = sample(1:nrow(data), size=floor((2/3)*nrow(data)))
trainX<-X[indexes,]
trainy<-y[indexes]
testX<-X[-indexes,]
testy<-y[-indexes]
```

Después de una extracción aleatoria de casos es altamente recomendable efectuar un análisis de datos mínimo para asegurarnos de no obtener clasificadores sesgados por los valores que contiene cada muestra.

Creación del modelo, calidad del modelo y extracción de reglas

Se crea el árbol de decisión usando los datos de entrenamiento (no hay que olvidar que la variable outcome es de tipo factor):

```
trainy = as.factor(trainy)
model <- C50::C5.0(trainX, trainy,rules=TRUE )
summary(model)
```

```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Dec 17 13:31:41 2020
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 1467 cases (4 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (1169/255, lift 1.2)
##  SEX = Hombre
##  ->  class Muere  [0.781]
##
```

```
## Rule 2: (20, lift 2.9)
##  CLASS in {2a, 1a}
##  AGE = Menor
##  ->  class Sobrevive  [0.955]
##
## Rule 3: (298/75, lift 2.3)
##  SEX = Mujer
##  ->  class Sobrevive  [0.747]
##
## Default class: Muere
##
##
## Evaluation on training data (1467 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##       3  317(21.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     914    75    (a): class Muere
##     242   236    (b): class Sobrevive
##
##
##  Attribute usage:
##
##  100.00% SEX
##    1.36% CLASS
##    1.36% AGE
##
##
## Time: 0.0 secs
```

Errors muestra el número y porcentaje de casos mal clasificados en el subconjunto de entrenamiento. El árbol obtenido clasifica erróneamente 304 de los 1467 casos dados, una tasa de error del 20.7%.

A partir del árbol de decisión de dos hojas que hemos modelado, se pueden extraer las siguientes reglas de decisión (gracias a rules=TRUE podemos imprimir las reglas directamente):

SEX = "Hombre" → Muere. Validez: 80,2%

CLASS = "3a" → Muere. Validez: 75.1%

CLASS "1ª", "2ª" o "Crew" y SEX = "Mujer" → Sobrevive. Validez: 90,5%

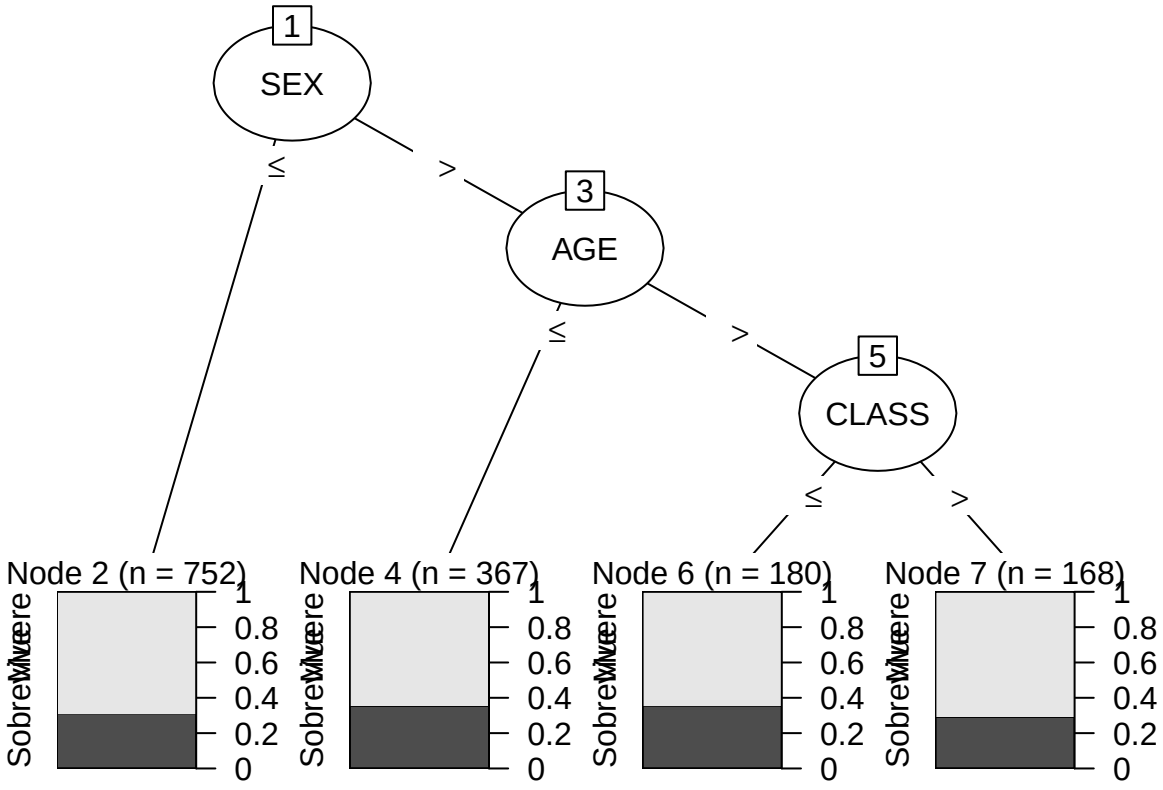
Por tanto podemos concluir que el conocimiento extraído y cruzado con el análisis visual se resume en "las mujeres y los niños primero a excepción de que fueras de 3ª clase".

A continuación mostramos el árbol obtenido.

```
model <- C50::C5.0(trainX, trainy)
plot(model)
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
```

```
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



Validación del modelo con los datos reservados

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
predicted_model <- predict( model, testX, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model == testy) / length(predicted_
```

```
## [1] "La precisión del árbol es: 78.2016 %"
```

Cuando hay pocas clases, la calidad de la predicción se puede analizar mediante una matriz de confusión que identifica los tipos de errores cometidos.

```
mat_conf<-table(testy,Predicted=predicted_model)
mat_conf
```

```
##            Predicted
## testy       Muere Sobrevive
##   Muere       450        51
##   Sobrevive   109       124
```

Otra manera de calcular el porcentaje de registros correctamente clasificados usando la matriz de confusión:

```
porcentaje_correct<-100 * sum(diag(mat_conf)) / sum(mat_conf)
print(sprintf("El %% de registros correctamente clasificados es: %.4f %%",porcentaje_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 78.2016 %"
```

Además, tenemos a nuestra disposición el paquete gmodels para obtener información más completa:

```
if(!require(gmodels)){
    install.packages('gmodels', repos='http://cran.us.r-project.org')
    library(gmodels)
}
```

```
## Loading required package: gmodels
```

```
CrossTable(testy, predicted_model,prop.chisq  = FALSE, prop.c = FALSE, prop.r =FALSE,dnn = c('Reality',
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  734
##
##
##              | Prediction
##      Reality |     Muere | Sobrevive | Row Total |
## -------------|-----------|-----------|-----------|
##        Muere |       450 |        51 |       501 |
##              |     0.613 |     0.069 |           |
## -------------|-----------|-----------|-----------|
##    Sobrevive |       109 |       124 |       233 |
##              |     0.149 |     0.169 |           |
```

```
## -------------|-----------|-----------|-----------|
## Column Total |       559 |       175 |       734 |
## -------------|-----------|-----------|-----------|
##
##
```

Ejercicios

Ejercicio 1:

Partiendo del ejemplo mostrado, repetid el ejercicio con otro conjunto de datos. Pueden ser datos reales de vuestro ámbito laboral o de algún repositorio de datos de Internet. Mirad por ejemplo: http://www.ics.uci.edu/~mlearn/MLSummary.html y http://www.kaggle.com.

Es muy importante seleccionar correctamente el conjunto de datos y explicar de forma correcta la base de datos y la razón de su elección.

Podéis añadir o variar los puntos si lo consideráis necesario (por ejemplo, crear el modelo con todos los datos y validación cruzada, probar el boosting o variar el prunning ...) Recordad también que el ciclo de vida de los proyectos de minería contempla retroceder para volver a generar el modelo con datos modificados o parámetros del algoritmo variados si el resultado no es lo suficientemente bueno.

Respuesta 1:

15% Se explica de forma clara la base de datos seleccionada y la razón de su elección.

Para la respuesta de este ejercicio usaremos el siguienre dataset ubicado en http://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+. Contiene datos sobre estimaciones de niveles de obesidad en individuos de paises como Colombia, Perú y México, basado en sus hábitos alimenticios y condicion física.

Se eligió este dataset por las siguientes razones:

- Una mayor cantidad de variables categóricas o discretizadas.
- Variables numéricas que pueden ser discretizadas para elaborar el árbol.
- Cantidad de observaciones relativamente suficiente para dividir el dataset en un subconjunto de entrenamiento y prueba.
- No presenta valores nulos o datos faltantes.
- Dominio del tema. Se la usó en la PEC2, para aplicar otros métodos. Además el dominio del tema es de facil interpretación.
- Es conjunto de datos que permite obtener reglas sobre una variable que pueden reducirse a 2 clases: Tiene o no tiene sobrepeso.
- Suficiente cantidad de información o recursos en la web que permitan entender las variables y observaciones contenidas. [1] [2] [3]

Este dataset contiene 17 atributos/variables y 2111 instancias u observaciones, según el siguiente detalle:

> gender.- Cadena de texto que indica el genero del individuo: Male (Masculino) o Female (Femenino).

age.- Valor numérico que indica la edad del individuo. Entre 14 y 61.

height.- Valor numérico que indica la altura (en metros) del individuo.

weight.- Valor numérico que indica el peso (en kilogramos) del individuo.

family_history_with_overweight.- Valor booleano que indica si algún familiar ha sufrido o padece de sobrepeso (SI/NO).

FAVC.- Valor booleano que indica si el individuo consume alimentos ricos en calorías (SI/NO).

FCVC.- Valor numérico que indica la frecuencia en el consumo de vegetables por el individuo. Entre el 1 (Nunca) al 3 (Siempre).

NCP.- Número de comidas principales en el día. Entre 1 a 4.

CAEC.- Valor categórico que indica el consumo de alimentos entre comidas principales: No, A veces, Frecuentemente y Siempre.

SMOKE.- Valor booleano que indica si el individuo fuma (SI / NO).

CH2O.- Valor numérico que indica los litros de agua que el individuo consume. Entre 1 a 3.

SCC.- Valor booleano que indica si el individuo monitorea o no su consumo de calorías.

FAF.- Valor numérico que indica la frecuencia de actividad física. Entre el 0 a 3.

TUE.- Valor numérico que indica las horas que el individuo usa dispositivos de tecnología.

CALC.- Valor categórico que indica la frecuencia de consumo de alcohol por el individuo: No bebo, A veces, Frecuentemente, Siempre.

MTRANS.- Valor categórico que indica el método de transporte usado por el individuo: Automovil, Moto, Bicicleta, Transporte Público, Caminando.

NObeyesdad.- Valor de clase, donde se indica la estimación del nivel de obesidad en el individuo: Bajo peso, normal, Sobre Peso I, Sobre Peso II, Obesidad I, Obesidad II, Obesidad III.

Para el fin de la resolución de este ejercicio y el siguiente, vamos a usar los atributos categóricos y a discretizar las variables: **Age (Edad)**, **Height (Altura)** y **Weight (Peso)**, los cuales corresponden a la edad expresada en años, la altura de la persona en metros y el peso en kilogramos, respectivamente.

El archivo no contiene datos que deban ser tratados, como nulos o valores desconocidos. Los atributos que corresponden a los hábitos alimenticios y de condición física son categóricos.

Así mismo, la columna que corresponde a la variable de clasificación es: *NObeyesdad*, con 7 clases.

En el siguiente artículo [2] relacionado al dataset, encontramos la descripción de los atributos, las clases que corresponden a los niveles de obesidad y además establece los rangos de la clase NObeyesdad. Para efectos del análisis, en esta PEC vamos a considerar el índice dichos valores de la clase para definir o clasificar a los individuos por debajo o encima de ese rango, con lo cual, la variable *NObeyesdad* tendría 2 clases para este ejercicio, como se detalla a continuación:

- SI (Tiene sobrepeso)
- No (No tiene sobrepeso)

Para cargar la estructura de datos lo haremos a partir de una archivo CSV, el mismo que se descargó desde http://archive.ics.uci.edu/ml/machine-learning-databases/00544/ObesityDataSet_raw_and_data_sinthetic%20(2).zip y se ha descomprimido en la misma ruta del archivo .Rmd. Cabe indicar que la primera fila del archivo corresponde a los nombres de los atributos. Ahora procedemos a cargar el dataset y ver el resumen de los datos.

5% Se presenta el código y es fácilmente reproducible.

```
datos <- read.csv("ObesityDataSet_raw_and_data_sinthetic.csv", sep=",")
summary(datos)
```

```
##     Gender                Age              Height           Weight
##  Length:2111        Min.   :14.00   Min.   :1.450   Min.   : 39.00
##  Class :character   1st Qu.:19.95   1st Qu.:1.630   1st Qu.: 65.47
##  Mode  :character   Median :22.78   Median :1.700   Median : 83.00
##                     Mean   :24.31   Mean   :1.702   Mean   : 86.59
##                     3rd Qu.:26.00   3rd Qu.:1.768   3rd Qu.:107.43
##                     Max.   :61.00   Max.   :1.980   Max.   :173.00
##  family_history_with_overweight     FAVC               FCVC
##  Length:2111                    Length:2111        Min.   :1.000
##  Class :character               Class :character   1st Qu.:2.000
##  Mode  :character               Mode  :character   Median :2.386
##                                                    Mean   :2.419
##                                                    3rd Qu.:3.000
##                                                    Max.   :3.000
##       NCP            CAEC              SMOKE                CH2O
##  Min.   :1.000   Length:2111        Length:2111        Min.   :1.000
##  1st Qu.:2.659   Class :character   Class :character   1st Qu.:1.585
##  Median :3.000   Mode  :character   Mode  :character   Median :2.000
##  Mean   :2.686                                         Mean   :2.008
##  3rd Qu.:3.000                                         3rd Qu.:2.477
##  Max.   :4.000                                         Max.   :3.000
##      SCC                 FAF              TUE             CALC
##  Length:2111        Min.   :0.0000   Min.   :0.0000   Length:2111
##  Class :character   1st Qu.:0.1245   1st Qu.:0.0000   Class :character
##  Mode  :character   Median :1.0000   Median :0.6253   Mode  :character
##                     Mean   :1.0103   Mean   :0.6579
##                     3rd Qu.:1.6667   3rd Qu.:1.0000
##                     Max.   :3.0000   Max.   :2.0000
##     MTRANS           NObeyesdad
##  Length:2111        Length:2111
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

10% Hay un estudio sobre los datos de los que se parte y los datos son preparados correctamente.

```
# Discretizamos la variable Age y creamos una nueva variable discreta edad
datos["edad"] <- cut(datos$Age, breaks = c(0,20,30,40,50,61), labels = c("0-20", "21-30", "31-40", "41-5

# Discretizamos la variable Height y creamos una nueva variable discreta estatura
datos["estatura"] <- cut(datos$Height, breaks = c(0,1.60,1.75,1.98), labels = c("Baja", "Normal", "Alta

# Discretizamos la variable Weight y creamos una nueva variable discreta peso
datos["peso"] <- cut(datos$Weight, breaks = c(0,55,80,100,173), labels = c("0-54.99", "55-79.99", "80-99

# Discretizamos la variable FCVC y creamos una nueva variable discreta come_vegetal
datos["come_vegetal"] <- cut(datos$FCVC, breaks = c(0,1,2,3), labels = c("Nunca", "A veces", "Siempre"))

# Discretizamos la variable NCP y creamos una nueva variable discreta num_comidas
datos["num_comidas"] <- cut(datos$NCP, breaks = c(0,2.49,3.49,4), labels = c("Entre 1 y 2", "Tres", "+ 3

# Discretizamos la variable CH2O y creamos una nueva variable discreta bebe_agua
datos["bebe_agua"] <- cut(datos$CH2O, breaks = c(0,1,2,3), labels = c("- 1 litro", "Entre 1 y 2 litros"
```

No se ha considerado la discretización para la variable correspondiente a la frecuencia de actividad física, ya que no permite dividir sus valores en las categorías indicadas en la encuesta y además no se otorga información adicional en los valores que constan para la variable FAF (¿Con qué frecuencia haces actividad física?) del dataset, por lo tanto para evitar discretizar erroneamente esta variable la omitiremos por esta vez.

Asimismo las variables TUE, NCP, para reducir el dataset con el que se elaborará el juego de datos inicial para elaboración del árbol. También vamos a excluir las variables *SMOKE*, *CALC* y *SCC*. Por esta vez solo vamos a considerar las variables que permitan establecer las reglas según la ingesta de comida y ciertos hábitos alimenticios.

Según lo indicado previamente respecto a las clases que se considerarán en la variable *NObeyesdad*:

- SI (Tiene sobrepeso)
- No (No tiene sobrepeso)

Procedemos a generar una nueva variable que contendrá las 2 clases antes mencionadas, para lo cual vamos a proceder a agruparlas de acuerdo al siguiente detalle:

- SI (Tiene sobrepeso)
  - Overweight_Level_I
  - Overweight_Level_II
  - Obesity_Type_I
  - Obesity_Type_II
  - Obesity_Type_III
- NO (No Tiene sobrepeso)
  - Insufficient_Weight
  - Normal_Weight

```
# Agrupamos las clases
datos$sobrepeso[datos$`NObeyesdad` %in% c("Overweight_Level_I", "Overweight_Level_II", "Obesity_Type_I"

datos$sobrepeso[datos$`NObeyesdad` %in% c("Insufficient_Weight", "Normal_Weight")] = "NO"
```

Finalmente definimos el dataset que será usado como base para aplicar el árbol de decisión.

```
# Agrupamos las clases
data_tree = datos[, c(1, 5:6, 9, 16, 18:24)]

str(data_tree)
```

```
## 'data.frame':    2111 obs. of  12 variables:
##  $ Gender                         : chr  "Female" "Female" "Male" "Male" ...
##  $ family_history_with_overweight: chr  "yes" "yes" "yes" "no" ...
##  $ FAVC                           : chr  "no" "no" "no" "no" ...
##  $ CAEC                           : chr  "Sometimes" "Sometimes" "Sometimes" "Sometimes" ...
##  $ MTRANS                         : chr  "Public_Transportation" "Public_Transportation" "Public_Trans
##  $ edad                           : Factor w/ 5 levels "0-20","21-30",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ estatura                       : Factor w/ 3 levels "Baja","Normal",..: 2 1 3 3 3 2 1 2 3 2 ...
##  $ peso                           : Factor w/ 4 levels "0-54.99","55-79.99",..: 2 2 2 3 3 1 1 1 2 2 .
##  $ come_vegetal                   : Factor w/ 3 levels "Nunca","A veces",..: 2 3 2 3 2 2 3 2 3 2 ...
##  $ num_comidas                    : Factor w/ 3 levels "Entre 1 y 2",..: 2 2 2 2 1 2 2 2 2 2 ...
##  $ bebe_agua                      : Factor w/ 3 levels "- 1 litro","Entre 1 y 2 litros",..: 2 3 2 2 2
##  $ sobrepeso                      : chr  "NO" "NO" "NO" "SI" ...
```

```
summary(data_tree)
```

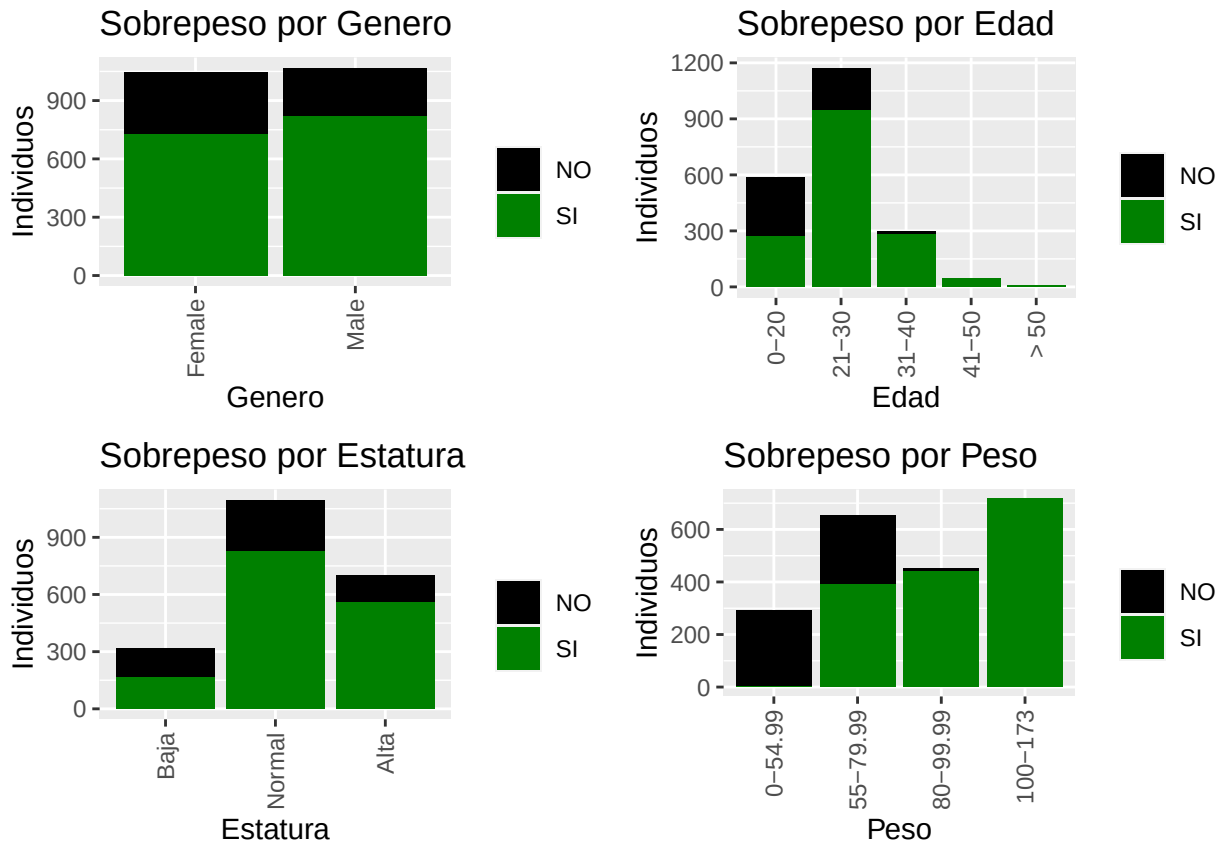
```
##     Gender          family_history_with_overweight     FAVC
##  Length:2111        Length:2111                    Length:2111
##  Class :character   Class :character               Class :character
##  Mode  :character   Mode  :character               Mode  :character
##
##
##      CAEC              MTRANS             edad        estatura
##  Length:2111        Length:2111        0-20 : 585   Baja  : 316
##  Class :character   Class :character   21-30:1170   Normal:1096
##  Mode  :character   Mode  :character   31-40: 299   Alta  : 699
##                                        41-50:  47
##                                        > 50 :  10
##        peso      come_vegetal       num_comidas                bebe_agua
##  0-54.99 :292   Nunca  :  33   Entre 1 y 2: 491   - 1 litro         : 211
##  55-79.99:653   A veces: 769   Tres       :1470   Entre 1 y 2 litros:1006
##  80-99.99:450   Siempre:1309   + 3        : 150   + 2 litros        : 894
##  100-173 :716
##
##   sobrepeso
##  Length:2111
##  Class :character
##  Mode  :character
##
##
```

Revisión de los datos, extracción visual de información y preparación de los datos

Nos interesa describir la relación entre el sobrepeso y algunas de las variables mencionadas anteriormente. Para ello, por un lado graficaremos mediante diagramas de barras la cantidad de individuos con sobrepeso

y sin sobrepeso según el genero, edad, estatura, peso. Por otro lado, para obtener los datos que estamos graficando utilizaremos el comando table para dos variables que nos proporciona una tabla de contingencia.

```
grid.newpage()
plotbyGenero<-ggplot(data_tree,aes(Gender,fill=sobrepeso))+geom_bar() + theme(axis.text.x = element_tex
plotbyEdad<-ggplot(data_tree,aes(edad,fill=sobrepeso))+geom_bar() + theme(axis.text.x = element_text(an
plotbyEstatura<-ggplot(data_tree,aes(estatura,fill=sobrepeso))+geom_bar() + theme(axis.text.x = element_
plotbyPeso<-ggplot(data_tree,aes(peso,fill=sobrepeso))+geom_bar() + theme(axis.text.x = element_text(an
grid.arrange(plotbyGenero, plotbyEdad, plotbyEstatura, plotbyPeso, ncol=2)
```



De estos gráficos obtenemos información muy valiosa que complementamos con las tablas de contingencia (listadas abajo):

- La cantidad de individuos que tienen sobrepeso es similar en hombres y mujeres.
- Por otro lado consideramos el porcentaje respecto a su edad, obtenemos que el mayor porcentaje de sobrepeso está en los individuos del rango entre 21 a 30 años.

- Respecto a la estatura el sobrepreso se concentra en los individuos con estatura media (normal) o alta.
- El porcentaje de sobrepeso en individuos que sobrepasan los 80 kg es del 100 por ciento en los 2 rangos que se establecieron

```
tabla_SGT <- table(data_tree$Gender, data_tree$sobrepeso)
tabla_SGT
```

```
##
```

```
##           NO  SI
##   Female 314 729
##   Male   245 823
```

```
prop.table(tabla_SGT, margin = 1)
```

```
##
##                 NO        SI
##   Female 0.3010547 0.6989453
##   Male   0.2294007 0.7705993
```

```
tabla_SET <- table(data_tree$edad, data_tree$sobrepeso)
tabla_SET
```

```
##
##          NO  SI
##   0-20  312 273
##   21-30 226 944
##   31-40  19 280
##   41-50   0  47
##   > 50    2   8
```

```
prop.table(tabla_SET, margin = 1)
```

```
##
##                 NO         SI
##   0-20  0.53333333 0.46666667
##   21-30 0.19316239 0.80683761
##   31-40 0.06354515 0.93645485
##   41-50 0.00000000 1.00000000
##   > 50  0.20000000 0.80000000
```

```
tabla_SEST <- table(data_tree$estatura, data_tree$sobrepeso)
tabla_SEST
```

```
##
##           NO  SI
##   Baja   150 166
##   Normal 269 827
##   Alta   140 559
```

```
prop.table(tabla_SEST, margin = 1)
```

```
##
##                 NO        SI
##   Baja   0.4746835 0.5253165
##   Normal 0.2454380 0.7545620
##   Alta   0.2002861 0.7997139
```

```
tabla_SPT <- table(data_tree$peso, data_tree$sobrepeso)
tabla_SPT
```

```
##
##             NO  SI
##   0-54.99  289   3
##   55-79.99 261 392
##   80-99.99   9 441
##   100-173    0 716
```

```
prop.table(tabla_SPT, margin = 1)
```

```
##
##                     NO         SI
##   0-54.99  0.98972603 0.01027397
##   55-79.99 0.39969372 0.60030628
##   80-99.99 0.02000000 0.98000000
##   100-173  0.00000000 1.00000000
```

```
tabla_SEPT.byPeso <- table(data_tree$edad, data_tree$sobrepeso, data_tree$peso)
tabla_SEPT.byPeso
```

```
## , ,  = 0-54.99
##
##
##          NO  SI
##   0-20  177   3
##   21-30 105   0
##   31-40   6   0
##   41-50   0   0
##   > 50    1   0
##
## , ,  = 55-79.99
##
##
##          NO  SI
##   0-20  132  94
##   21-30 116 186
##   31-40  12  93
##   41-50   0  18
##   > 50    1   1
##
## , ,  = 80-99.99
##
##
##          NO  SI
##   0-20    3 103
##   21-30   5 241
##   31-40   1  73
##   41-50   0  17
##   > 50    0   7
##
```

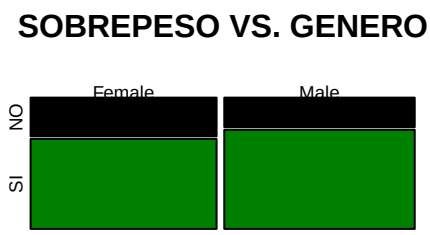
```
## , ,  = 100-173
##
##
##          NO  SI
##   0-20    0  73
##   21-30   0 517
##   31-40   0 114
##   41-50   0  12
##   > 50    0   0
```

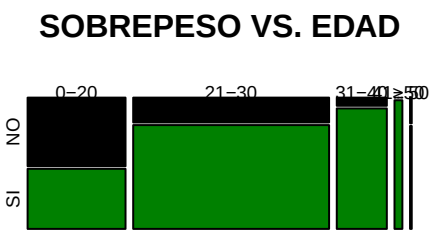
```
tabla_SPET.byEstatura <- table(data_tree$peso, data_tree$sobrepeso, data_tree$estatura)
tabla_SPET.byEstatura
```

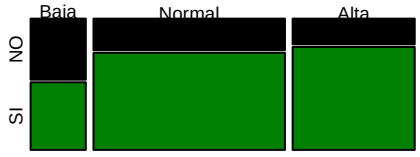
```
## , ,  = Baja
##
##
##             NO  SI
##   0-54.99  125   3
##   55-79.99  25 137
##   80-99.99   0  15
##   100-173    0  11
##
## , ,  = Normal
##
##
##             NO  SI
##   0-54.99  138   0
##   55-79.99 131 231
##   80-99.99   0 269
##   100-173    0 327
##
## , ,  = Alta
##
##
##             NO  SI
##   0-54.99   26   0
##   55-79.99 105  24
##   80-99.99   9 157
##   100-173    0 378
```

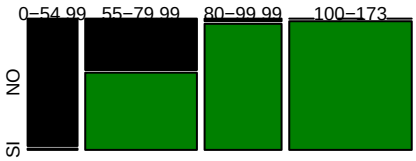
Una alternativa interesante a las barras de diagramas, es el plot de las tablas de contingencia. Obtenemos la misma información pero para algunos receptores puede resultar más visual.

```
par(mfrow=c(2,2))
plot(tabla_SGT, col = c("black","#008000"), main = "SOBREPESO VS. GENERO")
plot(tabla_SET, col = c("black","#008000"), main = "SOBREPESO VS. EDAD")
plot(tabla_SEST, col = c("black","#008000"), main = "SOBREPESO VS. ESTATURA")
plot(tabla_SPT, col = c("black","#008000"), main = "SOBREPESO VS. PESO")
```









Nuestro objetivo es crear un árbol de decisión que permita analizar si hay o no sobrepeso en los individuos encuestados. Por lo tanto, la variable por la que clasificaremos es el campo de si el pasajero tiene sobrepeso o no. De todas maneras, al imprimir las primeras (con head) y últimas 10 (con tail) filas nos damos cuenta de que los datos están ordenados.

```
head(data_tree,10)
```

```
##    Gender family_history_with_overweight FAVC      CAEC                MTRANS
## 1  Female                             yes   no Sometimes Public_Transportation
## 2  Female                             yes   no Sometimes Public_Transportation
## 3    Male                             yes   no Sometimes Public_Transportation
## 4    Male                              no   no Sometimes               Walking
## 5    Male                              no   no Sometimes Public_Transportation
## 6    Male                              no  yes Sometimes            Automobile
## 7  Female                             yes  yes Sometimes             Motorbike
## 8    Male                              no   no Sometimes Public_Transportation
## 9    Male                             yes  yes Sometimes Public_Transportation
## 10   Male                             yes  yes Sometimes Public_Transportation
##     edad estatura     peso come_vegetal num_comidas          bebe_agua
## 1  21-30   Normal 55-79.99      A veces        Tres Entre 1 y 2 litros
## 2  21-30     Baja 55-79.99      Siempre        Tres         + 2 litros
## 3  21-30     Alta 55-79.99      A veces        Tres Entre 1 y 2 litros
## 4  21-30     Alta 80-99.99      Siempre        Tres Entre 1 y 2 litros
## 5  21-30     Alta 80-99.99      A veces Entre 1 y 2 Entre 1 y 2 litros
## 6  21-30   Normal  0-54.99      A veces        Tres Entre 1 y 2 litros
## 7  21-30     Baja  0-54.99      Siempre        Tres Entre 1 y 2 litros
```

```
## 8  21-30   Normal  0-54.99      A veces        Tres Entre 1 y 2 litros
## 9  21-30     Alta 55-79.99      Siempre        Tres Entre 1 y 2 litros
## 10 21-30   Normal 55-79.99      A veces        Tres Entre 1 y 2 litros
##    sobrepeso
## 1         NO
## 2         NO
## 3         NO
## 4         SI
## 5         SI
## 6         NO
## 7         NO
## 8         NO
## 9         NO
## 10        NO
```

```
tail(data_tree,10)
```

```
##      Gender family_history_with_overweight FAVC      CAEC                MTRANS
## 2102 Female                            yes  yes Sometimes Public_Transportation
## 2103 Female                            yes  yes Sometimes Public_Transportation
## 2104 Female                            yes  yes Sometimes Public_Transportation
## 2105 Female                            yes  yes Sometimes Public_Transportation
## 2106 Female                            yes  yes Sometimes Public_Transportation
## 2107 Female                            yes  yes Sometimes Public_Transportation
## 2108 Female                            yes  yes Sometimes Public_Transportation
## 2109 Female                            yes  yes Sometimes Public_Transportation
## 2110 Female                            yes  yes Sometimes Public_Transportation
## 2111 Female                            yes  yes Sometimes Public_Transportation
##       edad estatura    peso come_vegetal num_comidas          bebe_agua
## 2102 21-30   Normal 100-173      Siempre        Tres         + 2 litros
## 2103 21-30   Normal 100-173      Siempre        Tres         + 2 litros
## 2104 21-30   Normal 100-173      Siempre        Tres Entre 1 y 2 litros
## 2105 21-30   Normal 100-173      Siempre        Tres Entre 1 y 2 litros
## 2106 21-30   Normal 100-173      Siempre        Tres Entre 1 y 2 litros
## 2107 21-30   Normal 100-173      Siempre        Tres Entre 1 y 2 litros
## 2108 21-30   Normal 100-173      Siempre        Tres         + 2 litros
## 2109 21-30     Alta 100-173      Siempre        Tres         + 2 litros
## 2110 21-30   Normal 100-173      Siempre        Tres         + 2 litros
## 2111 21-30   Normal 100-173      Siempre        Tres         + 2 litros
##      sobrepeso
## 2102        SI
## 2103        SI
## 2104        SI
## 2105        SI
## 2106        SI
## 2107        SI
## 2108        SI
## 2109        SI
## 2110        SI
## 2111        SI
```

20% Se aplica un árbol de decisión de forma correcta y se obtiene una estimación del error.

Procedemos a "desordenar" el dataset. Guardaremos los datos en un nuevo dataset: "data_random".

```
set.seed(1)
datos_random <- data_tree[sample(nrow(data_tree)),]
```

La variable que usaremos para la clasificación es el campo de SI tiene o NO sobrepeso, que está en la duodécima columna del conjunto de datos.

```
set.seed(1234)
y <- datos_random[,12]
X <- datos_random[,1:11]
```

Podemos elegir el subconjunto de entrenamiento y de prueba de diversas maneras. La primer opción consiste en calcular a cuántas filas corresponde dos tercios de los datos (2*2111/3=1407) y dividir "manualmente" el conjunto.

```
trainX <- X[1:1407,]
trainy <- y[1:1407]
testX <- X[1408:2111,]
testy <- y[1408:2111]
```

Creación del modelo, calidad del modelo y extracción de reglas

Se crea el árbol de decisión usando los datos de entrenamiento (no hay que olvidar que la variable outcome es de tipo factor):

```
trainy = as.factor(trainy)
model <- C50::C5.0(trainX, trainy,rules=TRUE )
summary(model)
```

```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Dec 17 13:31:42 2020
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 1407 cases (12 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (193/2, lift 3.6)
##  peso = 0-54.99
##  ->  class NO  [0.985]
##
## Rule 2: (78/6, lift 3.3)
##  Gender = Male
##  estatura = Alta
##  peso = 55-79.99
##  ->  class NO  [0.913]
```

```
##
## Rule 3: (54/5, lift 3.3)
##  FAVC = no
##  CAEC in {Frequently, Always}
##  ->  class NO  [0.893]
##
## Rule 4: (72/8, lift 3.2)
##  CAEC in {Frequently, Always}
##  edad in {0-20, 21-30}
##  peso = 55-79.99
##  ->  class NO  [0.878]
##
## Rule 5: (6, lift 3.2)
##  CAEC = no
##  estatura in {Baja, Normal}
##  bebe_agua = Entre 1 y 2 litros
##  ->  class NO  [0.875]
##
## Rule 6: (5, lift 3.1)
##  family_history_with_overweight = yes
##  CAEC = Sometimes
##  estatura in {Baja, Normal}
##  come_vegetal = Nunca
##  num_comidas = Tres
##  ->  class NO  [0.857]
##
## Rule 7: (21/3, lift 3.0)
##  FAVC = no
##  MTRANS in {Public_Transportation, Automobile}
##  edad = 21-30
##  estatura = Normal
##  peso = 55-79.99
##  ->  class NO  [0.826]
##
## Rule 8: (56/10, lift 3.0)
##  family_history_with_overweight = no
##  CAEC = Sometimes
##  edad in {0-20, 21-30, 41-50}
##  estatura = Normal
##  ->  class NO  [0.810]
##
## Rule 9: (14/3, lift 2.7)
##  CAEC = Sometimes
##  MTRANS in {Walking, Bike}
##  estatura = Normal
##  peso = 55-79.99
##  ->  class NO  [0.750]
##
## Rule 10: (761/5, lift 1.4)
##  peso in {80-99.99, 100-173}
##  ->  class SI  [0.992]
##
## Rule 11: (37, lift 1.3)
##  edad = 41-50
```

```
##  ->  class SI  [0.974]
##
## Rule 12: (69/2, lift 1.3)
##  CAEC = Sometimes
##  edad = 31-40
##  estatura = Normal
##  ->  class SI  [0.958]
##
## Rule 13: (21, lift 1.3)
##  CAEC = no
##  peso = 55-79.99
##  bebe_agua = + 2 litros
##  ->  class SI  [0.957]
##
## Rule 14: (21, lift 1.3)
##  family_history_with_overweight = yes
##  FAVC = no
##  CAEC = Sometimes
##  MTRANS in {Public_Transportation, Automobile}
##  edad in {0-20, 31-40, 41-50}
##  estatura = Normal
##  ->  class SI  [0.957]
##
## Rule 15: (177/7, lift 1.3)
##  FAVC = yes
##  edad = 31-40
##  ->  class SI  [0.955]
##
## Rule 16: (18, lift 1.3)
##  family_history_with_overweight = yes
##  FAVC = no
##  CAEC = Sometimes
##  MTRANS in {Public_Transportation, Automobile}
##  peso = 55-79.99
##  num_comidas = Entre 1 y 2
##  ->  class SI  [0.950]
##
## Rule 17: (546/32, lift 1.3)
##  family_history_with_overweight = yes
##  FAVC = yes
##  CAEC in {Sometimes, no}
##  MTRANS in {Public_Transportation, Automobile}
##  estatura in {Baja, Normal}
##  come_vegetal in {A veces, Siempre}
##  ->  class SI  [0.940]
##
## Rule 18: (102/9, lift 1.2)
##  CAEC = Sometimes
##  estatura = Baja
##  peso = 55-79.99
##  ->  class SI  [0.904]
##
## Rule 19: (54/6, lift 1.2)
##  Gender = Female
```

```
##  CAEC = Sometimes
##  estatura = Alta
##  ->  class SI  [0.875]
##
## Default class: SI
##
##
## Evaluation on training data (1407 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      19   56( 4.0%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     353    33    (a): class NO
##      23   998    (b): class SI
##
##
##  Attribute usage:
##
##   88.13% peso
##   62.26% CAEC
##   60.20% estatura
##   52.31% FAVC
##   45.49% family_history_with_overweight
##   43.57% MTRANS
##   39.16% come_vegetal
##   26.58% edad
##    9.38% Gender
##    1.92% bebe_agua
##    1.63% num_comidas
##
##
## Time: 0.0 secs
```

En el resumen, en la sección de evaluación en los datos de entrenamiento, el valor *Errors* muestra una tasa de error de 4.3% para las reglas obtenidas a partir del dataset de entrenamiento (trainx), la cual es muy aceptable. El árbol obtenido clasifica erróneamente 61 de los 1407 casos presentes.

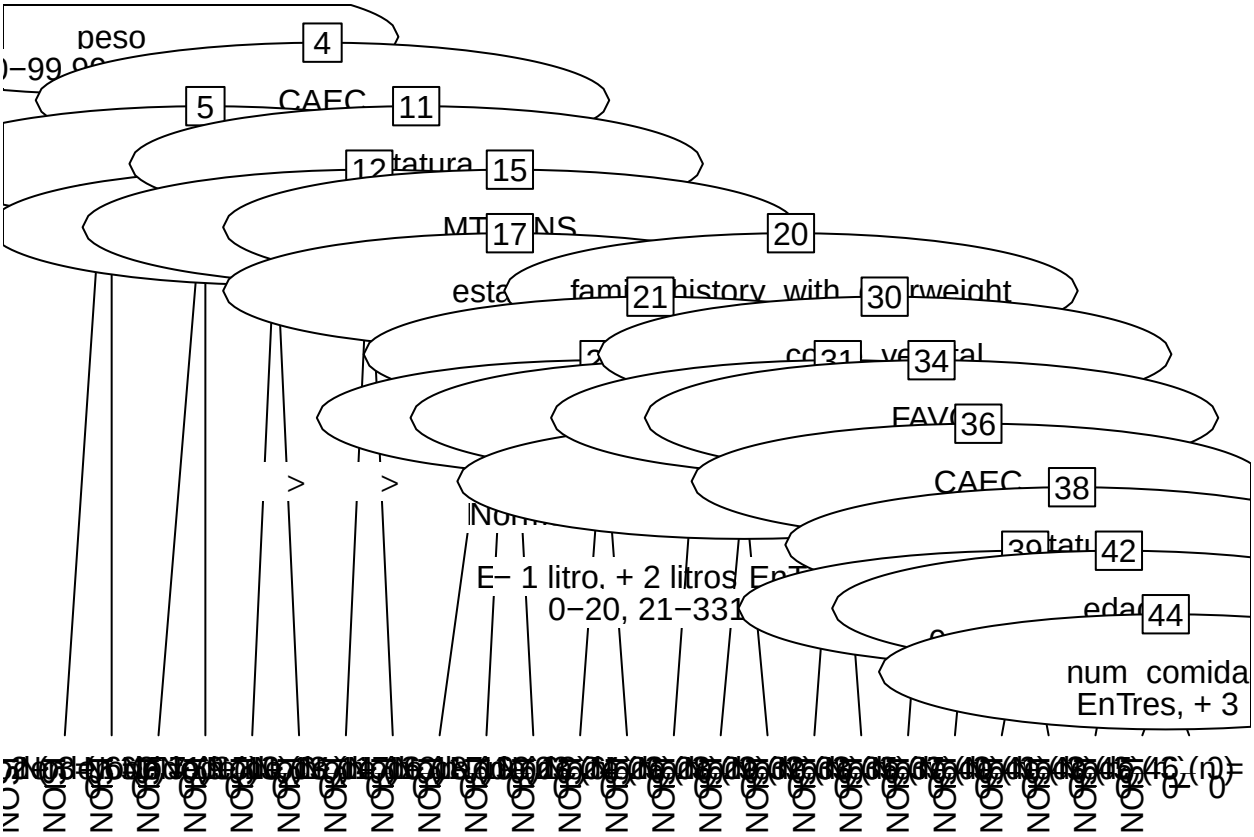
5% Se muestra de forma gráfica el árbol obtenido.

A continuación mostramos el árbol obtenido.

```
model <- C50::C5.0(trainX, trainy)
plot(model)
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
```

```
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



10% Se explican las reglas que se obtienen.

A partir del árbol de decisión de dos hojas que hemos modelado, se pueden extraer las siguientes reglas de decisión:

- Si Peso es menor a 55 kg -> NO Tiene Sobrepeso. **Validez = 98.5%**
- Si Gender = Hombre (Male) y, estatura = Alta y, peso = 55-79.99 kg -> NO Tiene Sobrepeso. **Validez = 91.3%**
- Si no ingiere comida altamente calórica y, come algo entre comidas está en {Frequently, Always} -> NO Tiene Sobrepeso. **Validez = 89.3%**
- Si come algo entre comidas está en {Frequently, Always} y, edad in {0-20, 21-30} y, peso = 55-79.99 kg -> NO Tiene Sobrepeso. **Validez = 87.8%**
- Si NO come algo entre comidas y, estatura está en {Baja, Normal} y, bebe agua Entre 1 y 2 litros diariamente -> NO Tiene Sobrepeso. **Validez = 87.5%**
- Si Tien historial de familia con sobrepeso (family_history_with_overweight = yes) y, come algo entre comidas a veces (CAEC = Sometimes) y, estatura está en {Baja, Normal} y, nunca come vegetles (come_vegetal = Nunca) y, el número de comidas son 3 (num_comidas = Tres) -> NO Tiene Sobrepeso. **Validez = 85.7%**
- Si no ingiere comida altamente calórica (FAVC = no) y, se transporta en automóvil y transporte público (MTRANS in {Public_Transportation, Automobile}) y, la está en edad entre 21-30 años y, la estatura es Normal y, su peso está entre 55-79.99 kg -> NO Tiene Sobrepeso. **Validez = 82.6%**

- Si NO tiene historial de familia con sobrepeso (family_history_with_overweight = no) y, come algo entre comidas a veces (CAEC = Sometimes) y, su edad está en los rangos {0-20, 21-30, 41-50} y, la estatura es Normal -> NO Tiene Sobrepeso. **Validez = 81%**
- Si come algo entre comidas a veces (CAEC = Sometimes) y, se transporta {Walking, Bike} y, su estatura es Normal y, su peso está en el rango 55-79.99 kg -> NO Tiene Sobrepeso **Validez = 75%**
- Si el peso está en los rangos {80-99.99, 100-173} kg. -> SI tiene sobrepeso. **Validez = 99.2%**
- Si la edad está entre los 41-50 años -> SI tiene sobrepeso. **Validez = 97.4%**
- Si come algo entre comidas a veces (CAEC = Sometimes) y, la edad está entre 31-40 años y, la estatura es Normal -> SI tiene sobrepeso. **Validez = 95.8%**
- Si NO come algo entre comidas (CAEC = no) y, su peso está entre 55-79.99 kg y, bebe más de 2 litros de agua diariamente -> SI tiene sobrepeso. **Validez = 95.7%**
- Si Tiene historial de familia con sobrepeso (family_history_with_overweight = yes) y, no ingiere comida altamente calórica (FAVC = no) y, come algo entre comidas a veces (CAEC = Sometimes) y, se transporta en automóvil y transporte público (MTRANS in {Public_Transportation, Automobile}) y, la edad está en los rangos {0-20, 31-40, 41-50} y, la estatura es Normal -> SI tiene sobrepeso. **Validez = 95.7%**
- Si ingiere comida altamente calórica (FAVC = yes) y, la edad está entre 31-40 años -> SI tiene sobrepeso. **Validez = 95.5%**
- Si tiene historial de familia con sobrepeso (family_history_with_overweight = yes) y, no ingiere comida altamente calórica (FAVC = no) y, come algo entre comidas a veces (CAEC = Sometimes) y, se transporta en automóvil y transporte público (MTRANS in {Public_Transportation, Automobile}) y, el peso está en el rango de 55-79.99 kg y, el número de comidas diarias son entre 1 y 2 (num_comidas = Entre 1 y 2) -> SI tiene sobrepeso. **Validez = 95%**
- Si tiene historial de familia con sobrepeso (family_history_with_overweight = yes) e, ingiere comida altamente calórica (FAVC = yes) y, come algo entre comidas a veces o no (CAEC {Sometimes, no}) y, se transporta en automóvil y transporte público (MTRANS in {Public_Transportation, Automobile}) y, la estatura está en {Baja, Normal} y, come vegetales A veces o Siempre -> SI tiene sobrepeso. **Validez = 94%**
- Si come algo entre comidas a veces (CAEC = Sometimes) y, la estatura es Baja y el peso está entre 55-79.99 kg -> SI tiene sobrepeso. **Validez = 90.4%**
- Si es mujer (Gender = Female) y, come algo entre comidas a veces (CAEC = Sometimes) y, la estatura es Alta -> SI tiene sobrepeso. **Validez = 87.5%**

10% Se usa el modelo para predecir con muestras no usadas en el entrenamiento y se obtiene una estimación del error.

Validación del modelo con los datos reservados

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
predicted_model <- predict( model, testX, type="class")
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model == testy) / length(predicted
```

```
## [1] "La precisión del árbol es: 95.3125 %"
```

15% Se prueba otro modelo de árbol o variantes diferentes del C50 obteniendo mejores resultados.

Obtenemos un nuevo dataset más reducido en base al resumen del árbol anterior, con las variables que tienen un uso superior al 50%, las cuales son: *peso* (88.13%), *CAEC* (62.26%), *estatura* (60.20%) y *FAVC* (52.31%).

```
# Obtenemos un nuevo dataset desordenado con menos variables. Definimos las variables
variables = c("peso", "CAEC", "estatura", "FAVC")
```

Vamos a aplicar una variación del c50 [4]. Usaremos la opción trials = 3, el cual permite obtener un clasificador árbol de decisión y mejorarlo con la técnica de boosting, mediante la función C5.0.

```
# Obtenemos un nuevo arbol con boosting
tree_boost <- C50::C5.0(x = trainX[, variables], y = trainy, trials = 3, rules=TRUE)

summary(tree_boost)
```

```
##
## Call:
## C5.0.default(x = trainX[, variables], y = trainy, trials = 3, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Dec 17 13:31:43 2020
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 1407 cases (5 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Rules:
##
## Rule 0/1: (193/2, lift 3.6)
##  peso = 0-54.99
##  ->  class NO  [0.985]
##
## Rule 0/2: (198/29, lift 3.1)
##  CAEC in {Frequently, Always}
##  ->  class NO  [0.850]
##
## Rule 0/3: (761/5, lift 1.4)
##  peso in {80-99.99, 100-173}
##  ->  class SI  [0.992]
##
## Rule 0/4: (1209/217, lift 1.1)
##  CAEC in {Sometimes, no}
##  ->  class SI  [0.820]
##
## Default class: SI
##
## -----  Trial 1:  -----
##
## Rules:
##
```

```
## Rule 1/1: (154.6/6, lift 2.3)
##  peso = 0-54.99
##  ->  class NO  [0.956]
##
## Rule 1/2: (536.6/149.7, lift 1.7)
##  peso = 55-79.99
##  estatura in {Normal, Alta}
##  ->  class NO  [0.720]
##
## Rule 1/3: (603.1/14.9, lift 1.7)
##  peso in {80-99.99, 100-173}
##  ->  class SI  [0.974]
##
## Rule 1/4: (112.8/31.5, lift 1.2)
##  peso = 55-79.99
##  estatura = Baja
##  ->  class SI  [0.717]
##
## Default class: SI
##
## -----  Trial 2:  -----
##
## Rules:
##
## Rule 2/1: (119.7, lift 3.1)
##  peso = 0-54.99
##  ->  class NO  [0.992]
##
## Rule 2/2: (111.2/12.2, lift 2.8)
##  CAEC in {Frequently, Always}
##  estatura in {Baja, Normal}
##  ->  class NO  [0.883]
##
## Rule 2/3: (177.5/40.2, lift 2.4)
##  peso = 55-79.99
##  estatura = Alta
##  ->  class NO  [0.771]
##
## Rule 2/4: (473.7, lift 1.5)
##  peso in {80-99.99, 100-173}
##  ->  class SI  [0.998]
##
## Rule 2/5: (853.3/198.7, lift 1.2)
##  CAEC in {Sometimes, no}
##  estatura in {Baja, Normal}
##  ->  class SI  [0.767]
##
## Default class: SI
##
##
## Evaluation on training data (1407 cases):
##
## Trial            Rules
## -----      ----------------
```

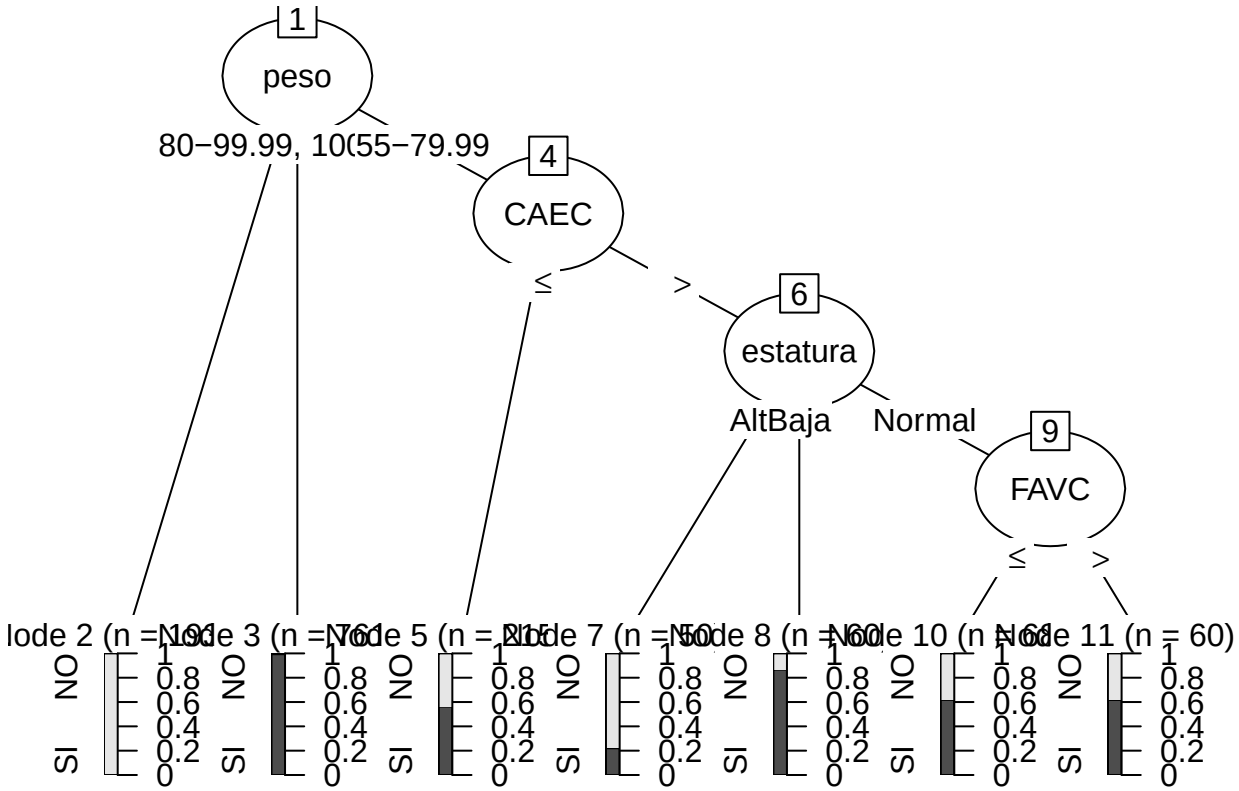
```
##      No      Errors
##
##    0      4  142(10.1%)
##    1      4  189(13.4%)
##    2      5  107( 7.6%)
## boost            107( 7.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     309    77    (a): class NO
##      30   991    (b): class SI
##
##
##  Attribute usage:
##
##  100.00% peso
##  100.00% CAEC
##   72.42% estatura
##
##
## Time: 0.0 secs
```

A partir del árbol de decisión que hemos modelado, se pueden extraer las siguientes reglas de decisión en el Trial 2:

- Si peso está en el rango entre 0-54.99 kg -> NO tiene sobrepeso. **Validez = 99.2%**
- Si come algo entre comidas frecuentemente o siempre (CAEC in {Frequently, Always}) y, estatura está en {Baja, Normal} -> NO tiene sobrepreso. **Validez = 88.3%**
- Si peso está entre 55-79.99 kg y, estatura = Alta -> NO tiene sobrepreso. **Validez = 77.1%**
- Si peso está entre 80-99.99 kg o 100-173 kg -> SI tiene sobrepeso [0.998]
- Si Nunca come algo entre comidas o lo hace a veces (CAEC in {Sometimes, no}) y, la estatura está en {Baja, Normal} -> SI tiene sobrepeso. **Validez = 76.7%**

Graficamos el árbol en su intento 3 (trials = 3)

```
# Graficamos el árbol
tree_boost <- C50::C5.0(x = trainX[, variables], y = trainy, trials = 3)
plot(tree_boost, trial = 2)
```



Predecimos el árbol en su intento 3 (trials = 3) para comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
# Predecimos
model_boost_pred3 <- predict(tree_boost, testX, type="class")

print(sprintf("La precisión del árbol es: %.4f %%",100*sum(model_boost_pred3 == testy) / length(model_b
```

```
## [1] "La precisión del árbol es: 92.8977 %"
```

10% Se presenta unas conclusiones donde se expone el conocimiento adquirido tras el trabajo realizado.

Se concluye que para los 2 casos de obtuvo una excelente precisión de los árboles. El primer árbol de decisión obtenido tiene una mejor (95.31 %) precisión que el segundo (92.89%). A pesar que el *boosting* es una técnica para mejorar el clasificador del árbol, en este caso no sucedió así. Posiblemente si se aumenta el número de trials mejore la precisión.

También notamos que las variables *peso* y *CAEC* (Frecuencia de ingerir algo entre comidas) tiene un alto uso en la definición de las reglas.

Rúbrica

- 15% Se explica de forma clara la base de datos seleccionada y la razón de su elección.
- 10% Hay un estudio sobre los datos de los que se parte y los datos son preparados correctamente.
- 20% Se aplica un árbol de decisión de forma correcta y se obtiene una estimación del error.
- 5% Se muestra de forma gráfica el árbol obtenido.
- 10% Se explican las reglas que se obtienen.
- 10% Se usa el modelo para predecir con muestras no usadas en el entrenamiento y se obtiene una estimación del error.
- 15% Se prueba otro modelo de árbol o variantes diferentes del C50 obteniendo mejores resultados.

- 5% Se presenta el código y es fácilmente reproducible.
- 10% Se presenta unas conclusiones donde se expone el conocimiento adquirido tras el trabajo realizado.

Bibliografía

---

[1] Estimation of obesity levels based on eating habits and physical condition Data Set, alojado en http://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+.

[2] Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico, alojado en https://doi.org/10.1016/j.dib.2019.104344

[3] Obesity Level Estimation Software based on Decision Trees, alojado en http://thescipub.com/pdf/jcssp.2019.67.77.pdf

[4] C5.0 Classification Models, https://cran.r-project.org/web/packages/C50/vignettes/C5.0.html

---