Informe: árboles de clasificación

Contexto:

El Departamento de Informática necesita crear un sistema que ayude a determinar qué estudiantes podrían aprobar una asignatura, considerando distintas características sobre ellos. El objetivo es clasificar a los estudiantes entre aquellos que podrían obtener un promedio final sobre o bajo 55. Además, los estudiantes fueron encuestados y entregaron información sobre su estado actual.

1. Describir el conjunto de datos: Cantidad de datos, Atributo Predictor/Clasificador, Tipo de Dato por atributo, Valores posibles.

```
#lectura de archivo
data <- read.csv(file="DatosInforme19.csv",head=TRUE,sep=";")
#Muestra la data
data
# Resumen estadistico
summary(data)
```

Figura 1: Código utilizado para abrir el dataframe y luego mostrar un resumen estadístico.

Como se muestra en la figura 1, lo primero es abrir el archivo "DatosInforme.csv". guardando en la variable **data** el dataframe con toda la información de los estudiantes.

Sexo	Horas.Estudio.Semanal	VTR	Tiempo.Libre	Carrete	Salud	Inasistencias	Nota.Final
<fct>	<fct>	<int>	<fct>	<fct>	<fct>	<int>	<fct>
F	2-5 hr	0	Normal	Mucho	Normal	6	<55
F	2-5 hr	0	Normal	Normal	Normal	4	<55
F	5-10 hr	0	Poco	Poco	Muy Buena	2	>=55
F	2-5 hr	0	Normal	Poco	Muy Buena	4	<55
M	2-5 hr	0	Mucho	Poco	Muy Buena	2	>=55
M	2-5 hr	0	Mucho	Mucho	Normal	4	>=55

Figura 2: Muestra las primeras 6 filas contenidas en data.

Data tiene la forma que se muestra en la figura 2 donde se obtiene que sus dimensiones es de $870X8$, es decir que tiene 870 registros (filas). En el encabezado de la tabla se encuentra el nombre de la columna y el tipo de dato que tiene.

A continuación se muestra una tabla resumen de las variables en el encabezado:

Nombre: César Quiroz Mansilla
Email: cesar.quirozm@sansano.usm.cl
Rol: 201573578-6

Nombre Columna	Tipo de Dato	Descripción
Sexo	fctr	El sexo del estudiante encuestado
Horas.Estudio.Semanal	fctr	Cantidad de horas de estudio dedicadas a la semana
VTR	int	Cantidad de veces que ha tomado el ramo [0-2]
Tiempo.Libre	fctr	Cantidad de tiempo libre separado en categorías
Carrete	fctr	Cantidad de carrete separado en categorías
Salud	fctr	Estado de salud separado en categorías
Inasistencia	int	Cantidad de tiempo libre [0-10]
Nota.Final	fctr	Nota final en dos categorías <55,>=55

Data contiene dos tipos de datos:
fcrt: significa factores, que R usa para representar variables categóricas con valores posibles fijos.
int: Números enteros.

Se resumen las categorías de las columnas en la siguiente tabla:

Nombre Columna	Posibles datos
Sexo	F,M
Horas.Estudio.Semanal	<2 hr, [2-10] hr, [5-10] hr,>10 hr
VTR	0,1,2
Tiempo.Libre	Nada,Poco, Normal, Mucho o Demasiado
Carrete	Nada, Poco, Normal, Mucho o Demasiado
Salud	Muy Mala, Suficiente, Normal, Buena, Muy Buena
Inasistencia	1,2,3,4,5,6,7,8,9,10
Nota.Final	<55 , >= 55

Todas las variables son del tipo predictor menos Nota.Final que es del tipo clasificador.
Luego se realiza un resumen estadístico del dataframe (contenido en data) usando el comando summary, obteniendo la figura 3.

```
Sexo    Horas.Estudio.Semanal      VTR           Tiempo.Libre      Carrete
F:501   <2 hr  :266           Min.   :0.0000   Demasiado: 89   Demasiado:137
M:369   >10 hr : 59           1st Qu.:0.0000   Mucho     :237   Mucho     :188
        2-5 hr :409           Median :0.0000   Nada      : 61   Nada      : 62
        5-10 hr:136           Mean   :0.2057   Normal    :330   Normal    :272
                              3rd Qu.:0.0000   Poco      :153   Poco      :211
                              Max.   :2.0000
       Salud     Inasistencias    Nota.Final
Buena     :148   Min.   : 0.000   <55 :325
Muy Buena :306   1st Qu.: 2.000   >=55:545
Muy Mala  :121   Median : 3.000
Normal    :190   Mean   : 3.863
Suficiente:105   3rd Qu.: 6.000
                 Max.   :10.000
```

Figura 3: Resultados obtenidos al aplicar summary(data).

Este resumen estadístico en las variables discretas indica la cantidad de registros, además nos informa

de manera inmediata que variables no son discretas (algo a tener en cuenta más adelante).

2. Construir e incluir un árbol de clasificación con los datos. Evalúe el árbol, interprete los resultados. Entregue el error de clasificación

Para construir el árbol de clasificación con las 8 clases, primero se necesita discretizar las clases VTR y Inasistencias.

En la figura 4, donde las clases VTR y Inasistencias pasan a ser de categóricas, de esta forma se puede construir en árbol de clasificación.

```
print("Forma conpacta de la data antes de discretizar")
str(data)
#Discretizar variables
data$Inasistencias = as.factor(data$Inasistencias)
data$VTR = as.factor(data$VTR)
print("Forma conpacta de la data despues de discretizar")
str(data)
```

```
[1] "Forma conpacta de la data antes de discretizar"
'data.frame':   870 obs. of  8 variables:
 $ Sexo                 : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 1 2 2 1 ...
 $ Horas.Estudio.Semanal: Factor w/ 4 levels "<2 hr",">10 hr",..: 3 3 4 3 3 3 3 3 3 3 ...
 $ VTR                  : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ Tiempo.Libre         : Factor w/ 5 levels "Demasiado","Mucho",..: 4 4 5 4 2 2 3 5 1 4 ...
 $ Carrete              : Factor w/ 5 levels "Demasiado","Mucho",..: 2 4 5 5 5 2 2 5 3 4 ...
 $ Salud                : Factor w/ 5 levels "Buena","Muy Buena",..: 4 4 2 2 2 4 3 3 2 5 ...
 $ Inasistencias        : Factor w/ 11 levels "0","1","2","3",..: 7 5 3 5 3 5 7 2 4 1 ...
 $ Nota.Final           : Factor w/ 2 levels "<55",">=55": 1 1 2 1 2 2 1 2 2 1 ...
[1] "Forma conpacta de la data despues de discretizar"
'data.frame':   870 obs. of  8 variables:
 $ Sexo                 : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 1 2 2 1 ...
 $ Horas.Estudio.Semanal: Factor w/ 4 levels "<2 hr",">10 hr",..: 3 3 4 3 3 3 3 3 3 3 ...
 $ VTR                  : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ Tiempo.Libre         : Factor w/ 5 levels "Demasiado","Mucho",..: 4 4 5 4 2 2 3 5 1 4
 $ Carrete              : Factor w/ 5 levels "Demasiado","Mucho",..: 2 4 5 5 5 2 2 5 3 4
 $ Salud                : Factor w/ 5 levels "Buena","Muy Buena",..: 4 4 2 2 2 4 3 3 2 5
 $ Inasistencias        : Factor w/ 11 levels "0","1","2","3",..: 7 5 3 5 3 5 7 2 4 1 ...
 $ Nota.Final           : Factor w/ 2 levels "<55",">=55": 1 1 2 1 2 2 1 2 2 1 ...
```

Figura 4: Código de discretizar variables y sus resultados.

Una vez las variables son discretas se pasa a construir dos conjuntos de datos con el dataframe original, uno sera de entrenamiento y otro de prueba, esto se realiza ya que una vez se construye un árbol se debe comprobar si realmente logra resolver el problema (logra generalizar el problema) o sufre de overfiting.

```
# Creacion Conjuntos para usar en el árbol de clasificacion
#Librerias
library(tidyverse)
# Creacion data entrenamiento
set.seed(7)
data_train <- sample_frac(data, .8) #Al 80%
# Creacion data test
data_test <- setdiff(data, data_train)
#Muestra la cantidad de registros en cada conjunto de datos
print("Dimensiones data completa")
dim(data)
print("Dimensiones data train")
dim(data_train)
print("Dimensiones data test")
dim(data_test)
```

```
[1] "Dimensiones data completa"
870  8
[1] "Dimensiones data train"
696  8
[1] "Dimensiones data test"
174  8
```

Figura 5: Código creación conjunto de entrenamiento y de prueba.

La figura 5 muestra la creación de ambos conjuntos, donde se define que el 80 % de la data completa pasa a ser del conjunto de entrenamiento (data_train)y el 20 % restante se van a conjunto de pruebas (data_test).

Recordando que la data completa tiene 870 registros (filas), implica que el conjunto de entrenamiento y de prueba tendrán 696 y 174 respectivamente.

Ahora con los conjuntos de datos creados se pasa a la construcción de árbol de clasificación para esto se encontraron dos formas de crearlos, usando la librería**Tree** o usando **Rpart**.

La creación y muestra de el árbol usando la librería Tree se ve en el código mostrado en la figura 6.

```
#import libreria
library(tree)
#Arbol creado usando libreria tree
mytree <- tree(
  Nota.Final ~ Sexo + Horas.Estudio.Semanal + VTR + Tiempo.Libre + Carrete + Salud + Inasistencias,
  data = data_train,
  method = "class",
)
#Mostrar plot del arbol creado
plot(mytree)
text(mytree, pretty = 1)
```

Figura 6: Código creación y muestra de árbol creado usando librería Tree.

Obteniendo el siguiente árbol como resultado:

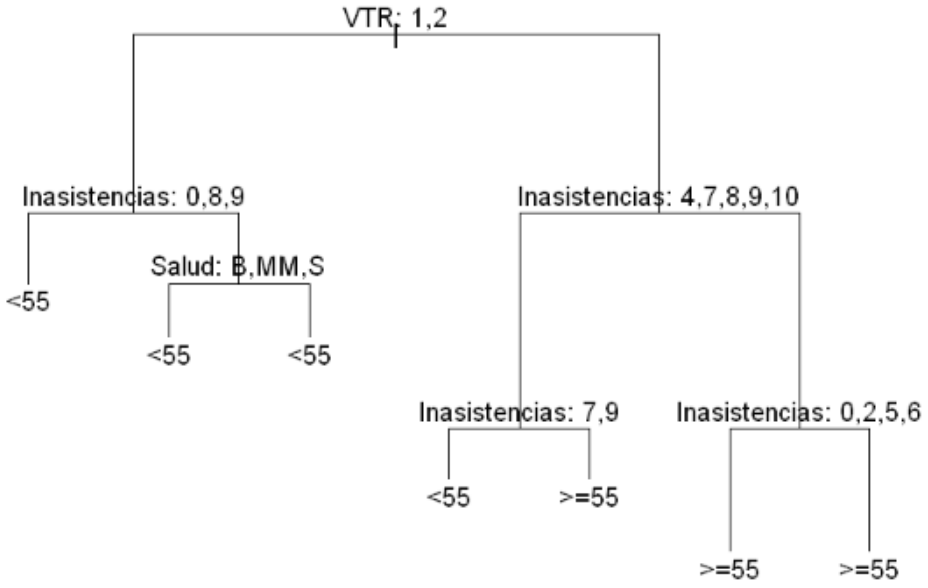


Figura 7: árbol de clasificación resultante usando libreria Tree.

Por otro lado, la creación y muestra de el árbol usando la librería Rpart se ve en el código mostrado en la figura 8.

```r
#Import libreria
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
#Arbol creado usando libreria rpart
mytree2 <- rpart(
  Nota.Final ~.,
  data = data_train,
  method = "class",
)
# plot mytree2
fancyRpartPlot(mytree2, caption = NULL)
```

Figura 8: Código creación y muestra de árbol creado usando librería Rpart.

Obteniendo el siguiente árbol como resultado:

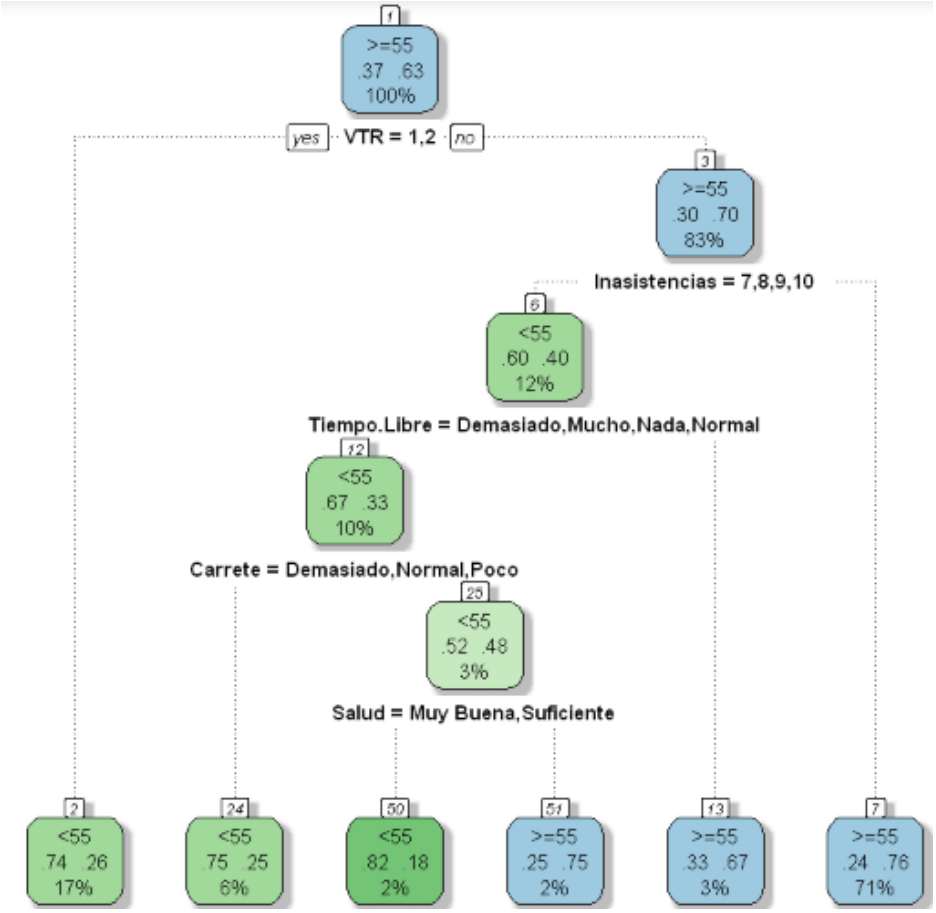


Figura 9: árbol de clasificación resultante usando libreria Rpart.

Ambos arboles entregan resultados similares, y ambos coinciden que el sexo no es un predictor que condicione si se aprueba o no un ramo.

Al comparar el accuracy de ambos arboles se obtiene que el árbol obtenido con Rpart es ligeramente mejor, tal como muestra la figura 10.

```
#Calculo precision de modelos.
#Arbol con Rpart
library(caret)
prediccion_1 <- predict(mytree2, newdata = data_test, type = "class")
confusionMatrix(prediccion_1, data_test[["Nota.Final"]])
#Arbol con Tree
prediccion_2 <- predict(mytree, newdata = data_test, type = "class")
confusionMatrix(prediccion_2, data_test[["Nota.Final"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction <55 >=55
      <55   34   13
      >=55  34   93

               Accuracy : 0.7299
                 95% CI : (0.6575, 0.7943)
    No Information Rate : 0.6092
    P-Value [Acc > NIR] : 0.0005658
```

Resultados Rpart

```
Confusion Matrix and Statistics

          Reference
Prediction <55 >=55
      <55   26    9
      >=55  42   97

               Accuracy : 0.7069
                 95% CI : (0.6333, 0.7733)
    No Information Rate : 0.6092
    P-Value [Acc > NIR] : 0.004606
```

Resultados Tree

Figura 10: Comparación de acurracy de ambos arboles obtenidos.

Dado a lo mostrado en la figura anterior, se utilizara Rpart para el resto de análisis en el informe.

Al árbol obtenido con Rpart, ya que tiene una accuracy del 73 % aprox, implica que tiene un error del 0,27 % aprox.

También resulta raro que el algoritmo no tome como predictor importante las horas de estudio semanales (el árbol usando Tree tampoco cuenta esta variable) y le de mucho valor a la inasistencia (segunda más importante).

Por otra parte que el VTR tenga tanto valor tiene sentido ya que si uno reprueba un ramo, lo más común es hacer todo lo posible para pasar la segunda vez (en el caso del tercer intento esta obligado a pasar practicamente), sin embargo como se vio en el resumen estadístico hay muy pocas personas que tienen VTR mayor que 0 esto indica que es necesario tener mas predictores importante para así poder tener mejores perfiles de estudiantes.

3. Contestar las siguientes preguntas:

a) ¿Cuáles fueron las variables relevantes en la construcción del árbol? Explique por qué una variable es más relevante que otra.

Las variables relevantes para la construcción del árbol son: VTR, Inasistencias, Tiempo libre, Carrete, Salud. En ese mismo orden de importancia. Son las mas relevantes por la pureza de sus atributos, ya que, gracias a estos se pueden dividir de manera más óptima.

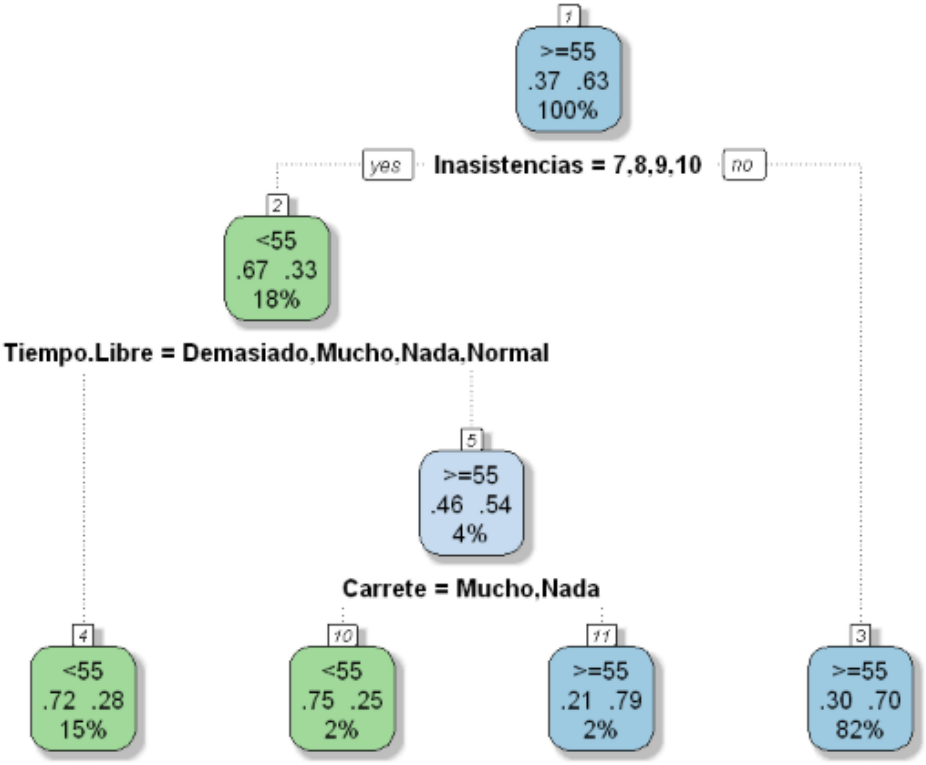
Para desarrollar y verificar esto, se ocupó el comando summary.

b) ¿Qué sucede si se modifica el tipo de dato de VTR?

El principio no cambia nada, ya que, al cambiar VTR a una variable continua se obtiene el mismo árbol (se cometio el error al comienzo de no discretizar), pero debido a que no es posible tomar un ramo 0,5 o 0,8 veces, puede provocar inconsistencia en el árbol al tener más predictores, categóricas o hojas en el árbol, provocando llegar a resultados que es imposible que se den en la vida real.

c) Si la variable VTR no es incluida en la construcción del árbol, ¿Qué sucede con el árbol? Evalúe los cambios

El árbol obtenido quitando la categoría VTR es el siguiente:



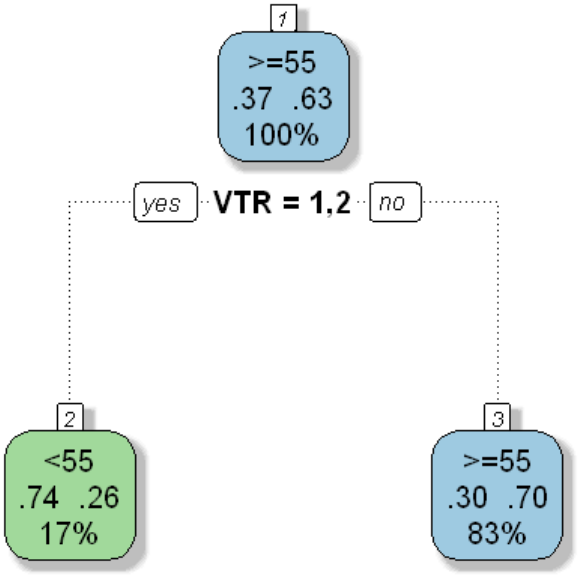
Lo que ocurre en el árbol, es que la categoría que estaba en la segunda posición de importancia (inasistencia) pasa a ser la más importante.

Al calcular el error este prácticamente se mantiene a lo obtenido en el punto 2. Además a diferencia del árbol anterior se pierde la variable salud del árbol.

Además es un árbol mucho más sencillo al del punto 2, que si permitiría encontrar el perfil de estudiantes que estén dando por primera vez el ramo de mucho mejor manera.

d) Si la variable Inasistencias no es incluida en la construcción del árbol, ¿Qué sucede con el árbol? Evalúe los cambios

El árbol obtenido quitando la categoría Inasistencias es el siguiente:



En este caso el árbol se reduce a un solo al predictor VTR, es decir que el VTR solamente basta pasar saber si un estudiante aprobara o no.

El error de este árbol es del 31 % aprox.

Además como se comento en el punto 2 si el VTR es el unico predictor, el árbol no ayudaria a encortrar verdaderamente los perfiles de estudiantes que aprobaran o no el ramo, ya que como se dijo antes, cualquier estudiante al reprobar se sabe que hará todo lo posible para no volver a reprobar.

e) ¿Qué sucede si utiliza un 30, 50 y 70 % de los datos entregados como Training? Evaluar e imprimir los arboles obtenidos. Explique.

La tabla resumen de los errores obtenidos es:

	30 %	50 %	70 %
Data_test	0,3186	0,2920	0,2797
Data_train	0,2184	0,2414	0,2348

Los mejores resultados se encuentran con un conjunto de entrenamiento del 70 %, además se comprueba que el árbol obtenido no presenta overfiting, ya que, entrenamiento y prueba presentan resultados muy similares (especialmente en el 70 %).

sin embargo estos arboles aumentan mucho su dificultad respecto al presentado en el punto 2, pero a diferencia del árbol obtenido antes estos arboles si presentan las horas de estudio como un predictor importante, por lo tanto es tentador asegurar que el mejor árbol es el obtenido con un conjunto de

entrenamiento del 70 % ya que es el que presenta el error menor y contiene una variable que por sentido si debería tener peso para pasar un ramo, las horas de estudio invertidas.

Los arboles obtenidos con 30, 50 y 70 % son los siguientes:

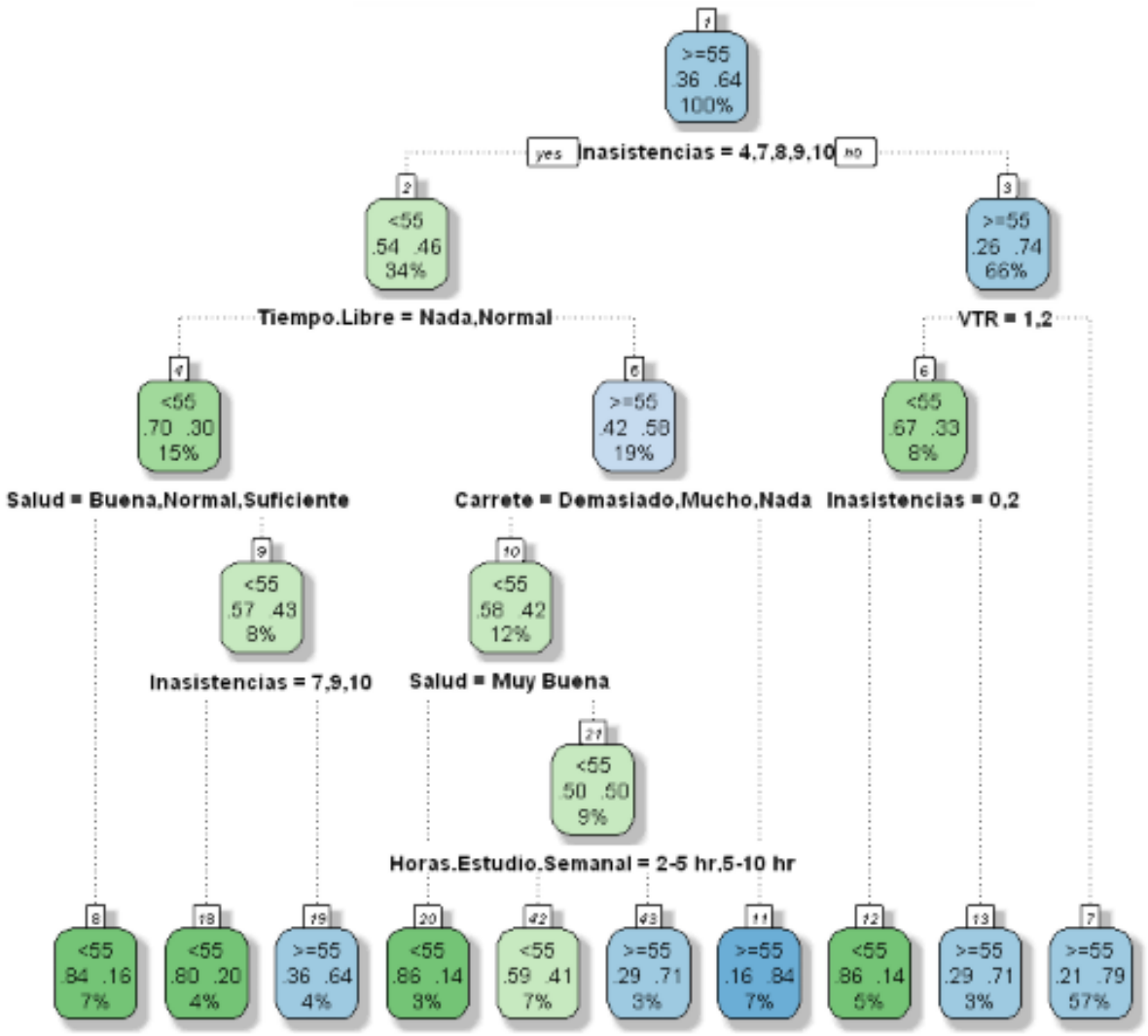


Figura 11: árbol obtenido con 30 % de data en entrenamiento.

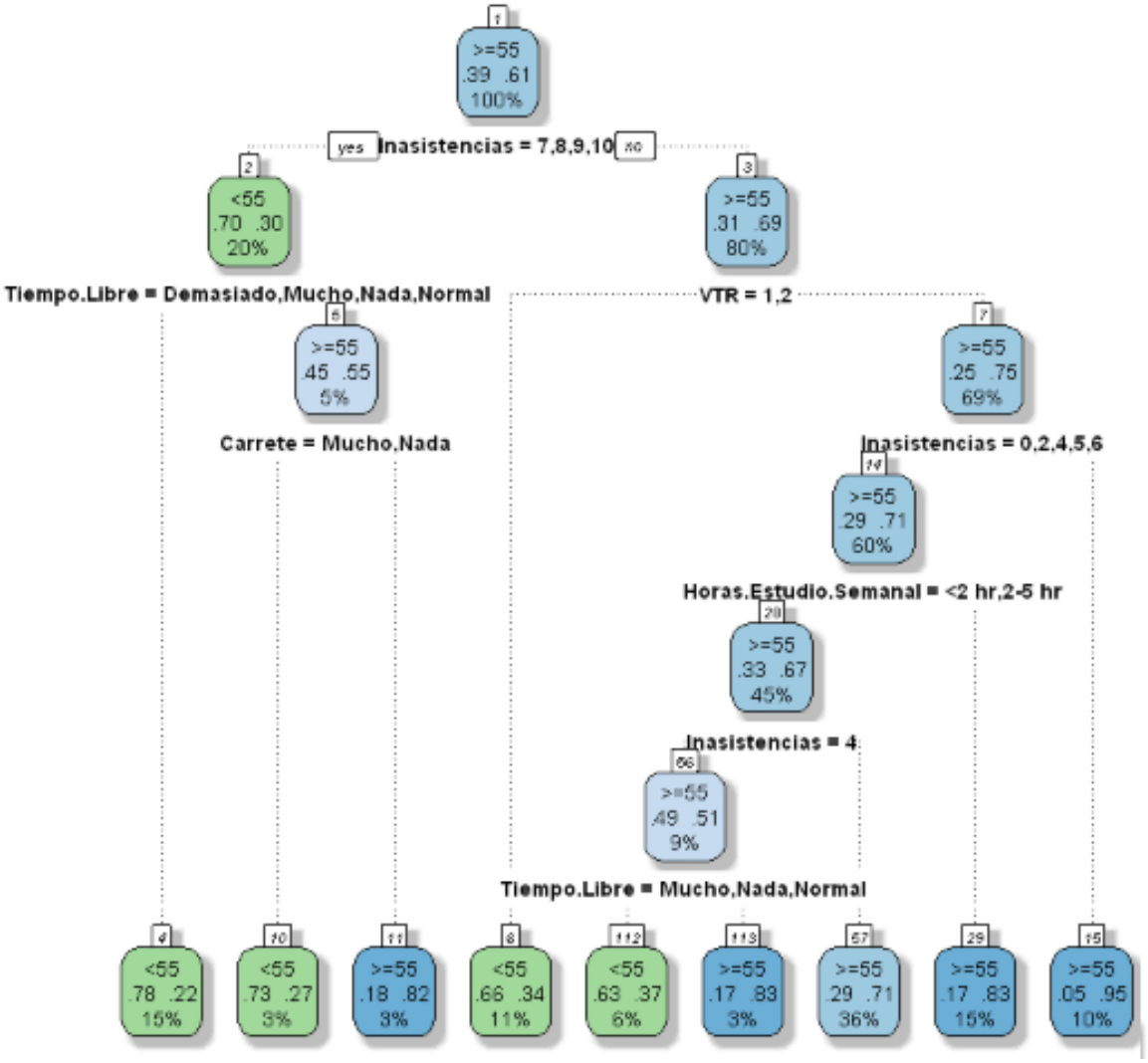


Figura 12: árbol obtenido con 50 % de data en entrenamiento.

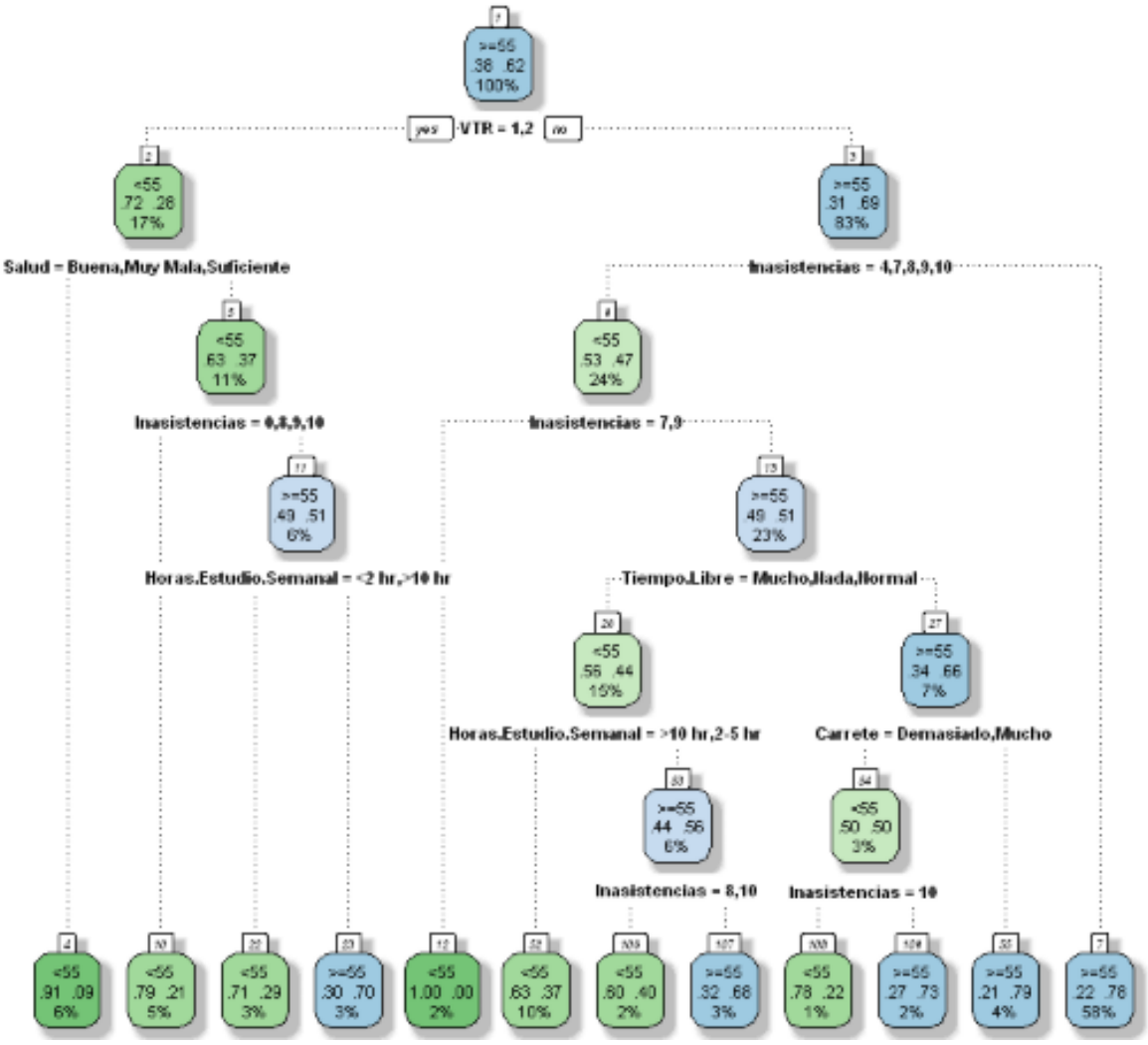


Figura 13: árbol obtenido con 70 % de data en entrenamiento.

Extra:

Todo el código utilizado para encontrar los resultados de este informe se encuentran en el siguiente link:

https://github.com/yuphyn/Investigaci-n-de-operaciones

Para ejecutarlo se debe instalar el kernel y librearías necesarias de R.