Departamento de Informática
Universidad Técnica Federico Santa María

Universidad Técnica Federico Santa María
Campus Santiago San Joaquín
INF293 - Investigación de Operaciones
Primer Semestre 2019

Informe Árboles de Clasificación

Juan Pablo Jorquera 201573533-6 juan.jorqueraz@sansano.usm.cl

En el informe que se presenta a continuación, se desarrolló utilizando el concepto de árboles de clasificación con una muestra de datos buscando ser capaces de predecir qué estudiantes podrían aprobar o no una asignatura, en base a distintas características asociadas a ellos.

1. Conjunto de Datos

Para resolver los problemas que se presentan a continuación, se utilizó un conjunto de datos de 870 estudiantes distintos, para los cuales se utilizaron los siguientes atributos:

- **Sexo**: Predictor, variables categóricas: Masculino o Femenino.
- **Horas de Estudio Semanales**: Predictor, variables discretas: <2 horas, [2, 5] horas, [5, 10] horas y >10 horas.
- **VTR** (Veces que ha tomado el ramo): Predictor, variables discretas: 0, 1 ó 2 veces.
- **Tiempo Libre**: Predictor, variables discretas: Nada, Poco, Normal, Mucho o Demasiado.
- **Carrete**: Predictor, variables discretas: Nada, Poco, Normal, Mucho o Demasiado.
- **Salud**: Predictor, variables discretas: Muy Mala, Suficiente, Normal, Buena o Muy Buena.
- **Inasistencias**: Predictor, variables discretas entre 0 a 10 inasistencias.
- **Nota Final**: Clasificador (objetivo), variables discretas: con nota <55 ó >=55.

2. Árbol resultante

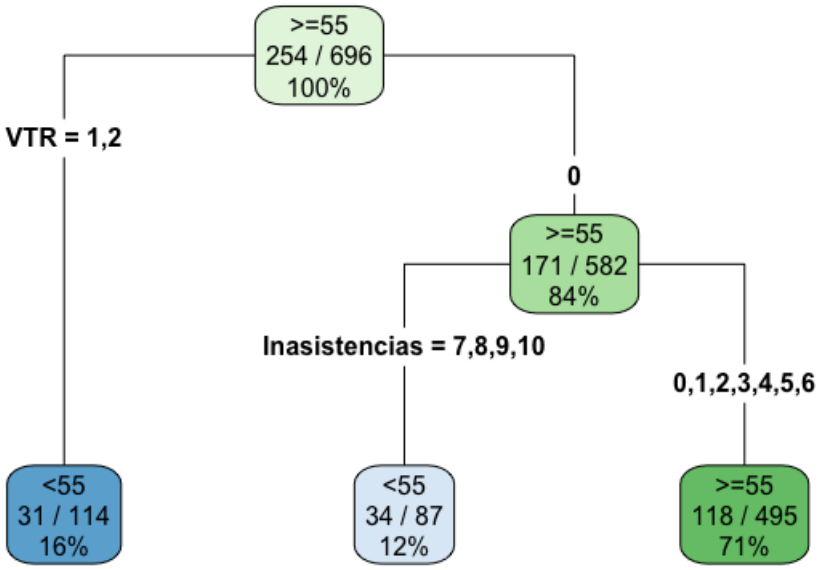

Figura 1: Árbol de clasificación, incluyendo fracción de error de cada nodo y porcentaje sobre el total

El árbol se construyó utilizando un 80 % de los datos para training y el otro 20 % para testing. Haciendo un análisis cualitativo, se encuentra que el árbol está dentro de lo esperado, usando como principal predictor el VTR y luego las Inasistencias, lo cual resulta razonable, en donde aquellos que vuelven a rendir el ramo y los que faltan constantemente a clases son más probables a reprobar el ramo. Por otro lado, el árbol presenta un error de clasificación cercano al 36.5 %, lo cual resulta muy elevado y viendo la poca cantidad de nodos en el árbol se puede asumir un posible problema de underfitting, pero comparándolo con el error de clasificación de la muestra de test de un 27.0 %, se ve que la predicción presenta un rendimiento adecuado.

Comandos utilizados: read.csv, rpart, rpart.plot, factor, as.factor, sample_frac, setdiff, summary, predict, confusionMatrix, set.seed.

Código utilizado:

- Inicialmente para el trabajo fueron necesarias las siguientes librerias, además de sus dependencias:

```
library(rpart)
library(rpart.plot)
library(tidyverse)
library(caret)
```

```
library(e1071)
library(tree)
```

- Luego para la generación del árbol principal:

```
# Semilla de randomizacion
set.seed(581)
# Lectura archivo
Data <- read.csv(file="./DatosInforme19.csv", header=TRUE, sep=";")
# Discretizacion (en orden si corresponde)
Data$VTR <- factor(Data$VTR, levels=c(0:2), ordered=TRUE)
Data$TiempoLibre <- factor(Data$TiempoLibre, levels=c("Nada", "Poco",
"Normal", "Mucho", "Demasiado"), ordered=TRUE)
Data$Salud <- factor(Data$Salud, levels=c("Muy Mala", "Suficiente",
"Normal", "Buena", "Muy Buena"), ordered=TRUE)
Data$Carrete <- factor(Data$Carrete, levels=c("Nada", "Poco",
"Normal", "Mucho", "Demasiado"), ordered=TRUE)
Data$Inasistencias <- factor(Data$Inasistencias, levels=c(0:10),
ordered=TRUE)
Data$HorasEstudioSemanal <- factor(Data$HorasEstudioSemanal,
levels=c("<2 hr", "2-5 hr", "5-10 hr", ">10 hr"), ordered=TRUE)
Data$NotaFinal <- factor(Data$NotaFinal, levels=c("<55", ">=55"),
ordered=TRUE)
Data$Sexo <- as.factor(Data$Sexo)
# Sets de entrenamiento y testing
Data_Sample <- sample_frac(Data, .8)
Data_Test <- setdiff(Data, Data_Sample)
# Generaci n  rbol
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal +
VTR + TiempoLibre + Carrete + Salud + Inasistencias,
data = Data_Sample, method = "class")
# Resultado, resumen y grafico del arbol
arbol
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
# Evaluaci n resultados
prediccion_1 <- predict(arbol, newdata = Data_Test, type = "class")
confusionMatrix(prediccion_1, Data_Test[["NotaFinal"]])
```

3. Preguntas

3.a.

En el árbol resultante, las variables relevantes en orden son VTR y las Inasistencias para la clasificación de los estudiantes (además de Nota Final como objetivo). Esto se debe al nivel de pureza de cada uno de los atributos, es decir, en base a ellos se puede realizar una mejor división, la cual clasifique de manera más heterogénea a los estudiantes.

Comandos relevantes: summary. ***Código utilizado:***

```
summary(arbol)
```

3.b.

Nada, ya que si se tomaran las variables 0, 1 y 2 como categóricas, la agrupación resultante para clasificar sería exactamente la misma que al ser discreta, debido a que es la mejor división encontrada. Luego, si se toma VTR como variable continua, la clasificación sigue siendo la misma:

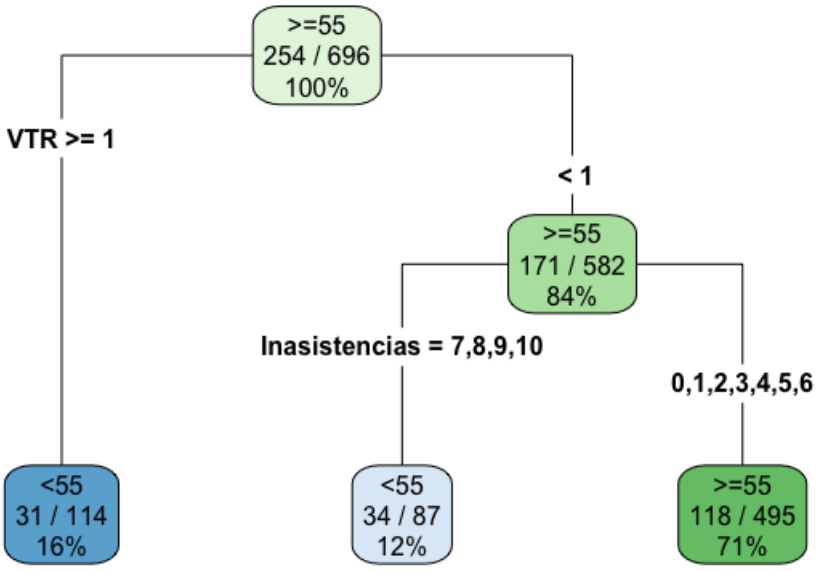

Figura 2: Árbol con VTR como variable continua

Sin embargo, la clasificación de VTR >= 1 y <1 está aproximada, entonces, analizando la clasificación realizada por debajo:

```
1) root 696 254 >=55 (0.3649425 0.6350575)
  2) VTR>=0.5 114  31 <55 (0.7280702 0.2719298) *
  3) VTR< 0.5 582 171 >=55 (0.2938144 0.7061856)
    6) Inasistencias=7,8,9,10 87  34 <55 (0.6091954 0.3908046) *
    7) Inasistencias=0,1,2,3,4,5,6 495 118 >=55 (0.2383838 0.7616162) *
```

Figura 3: Árbol real con VTR como variable continua

Se puede ver que la clasificación está hecha dividiendo VTR en 0.5, lo cual es conceptualmente incorrecto (no es posible tomar un ramo 0.5 veces) y, pese a generar el mismo árbol en esta situación, podría provocar diferencias en otros casos.

Comandos relevantes: rpart, rpart.plot, summary, as.numeric.

Código utilizado: Se genera igual al árbol principal, pero tomando VTR como flotante.

```
Data$VTR <- as.numeric(Data$VTR)
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal + VTR +
TiempoLibre + Carrete + Salud + Inasistencias, data = Data_Sample,
method = "class")
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
```

3.c.

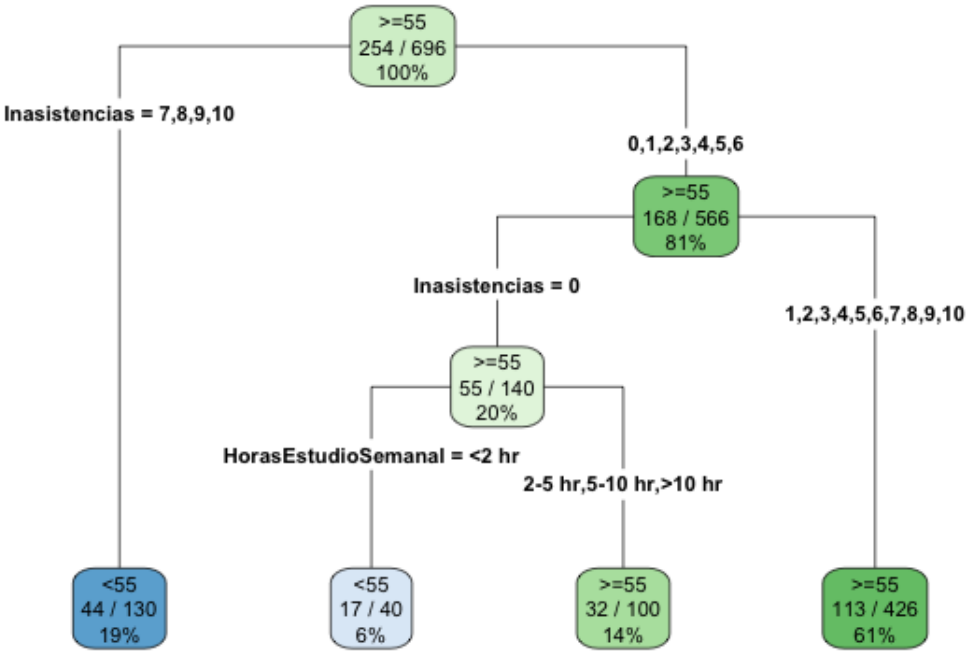

Figura 4: Árbol sin atributo VTR

En este caso, las Inasistencias serían el atributo más importante para definir la clasificación, que se esperaba al ser el segundo más importante en el árbol original, luego, el atributo de Horas de Estudio Semanal resulta como el segundo más relevante. El árbol se convierte en uno más complejo con lo que se dificulta la clasificación de los estudiantes. En este árbol se presenta un error de clasificación de 36.5 % para training y 27.6 % para test, por lo que el rendimiento es prácticamente el mismo al original y se considera más valioso ese árbol al tener una clasificación más simple y con divisiones más claras.

Comandos relevantes: rpart, rpart.plot, summary, predict, confusionMatrix.

Código utilizado: Se genera igual al árbol principal, pero excluyendo el atributo VTR en la generación del árbol en rpart.

```
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal + TiempoLibre +
Carrete + Salud + Inasistencias, data = Data_Sample, method = "class")
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
prediccion_1 <- predict(arbol, newdata = Data_Test, type = "class")
confusionMatrix(prediccion_1, Data_Test[["NotaFinal"]])
```

3.d.

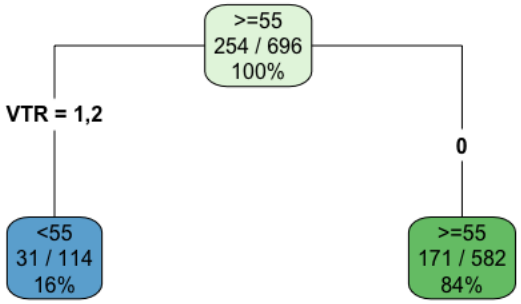

Figura 5: Árbol sin atributo Inasistencias

En este caso resultaría VTR como único atributo relevante, lo que puede generar el problema de underfitting, ya que al ser muy poco específico se cuestiona si sirve adecuadamente para evaluar a los estudiantes. Evaluando los errores de clasificación se ve que hay un 36.5 % en el training y un 30.5 % de testing, presentando una diferencia en el conjunto de test de un 3.5 % sobre el árbol original, lo cual resulta considerable. Es por esto que este árbol sirve más para recalcar la importancia de tener VTR como principal predictor.

Analizando el árbol en la práctica no tiene mucho sentido, ya que se deduce que sólo los alumnos que han reprobado el ramo lo van a reprobar, lo que implicaría que nadie hubiese reprobado el ramo en primer lugar.

Comandos relevantes: rpart, rpart.plot, summary, predict, confusionMatrix.

Código utilizado: Al igual que antes, se genera igual al árbol principal, pero excluyendo el atributo Inasistencias en la generación del árbol en rpart.

```
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal + VTR +
TiempoLibre + Carrete + Salud, data = Data_Sample, method = "class")
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
prediccion_1 <- predict(arbol, newdata = Data_Test, type = "class")
confusionMatrix(prediccion_1, Data_Test[["NotaFinal"]])
```

3.e.

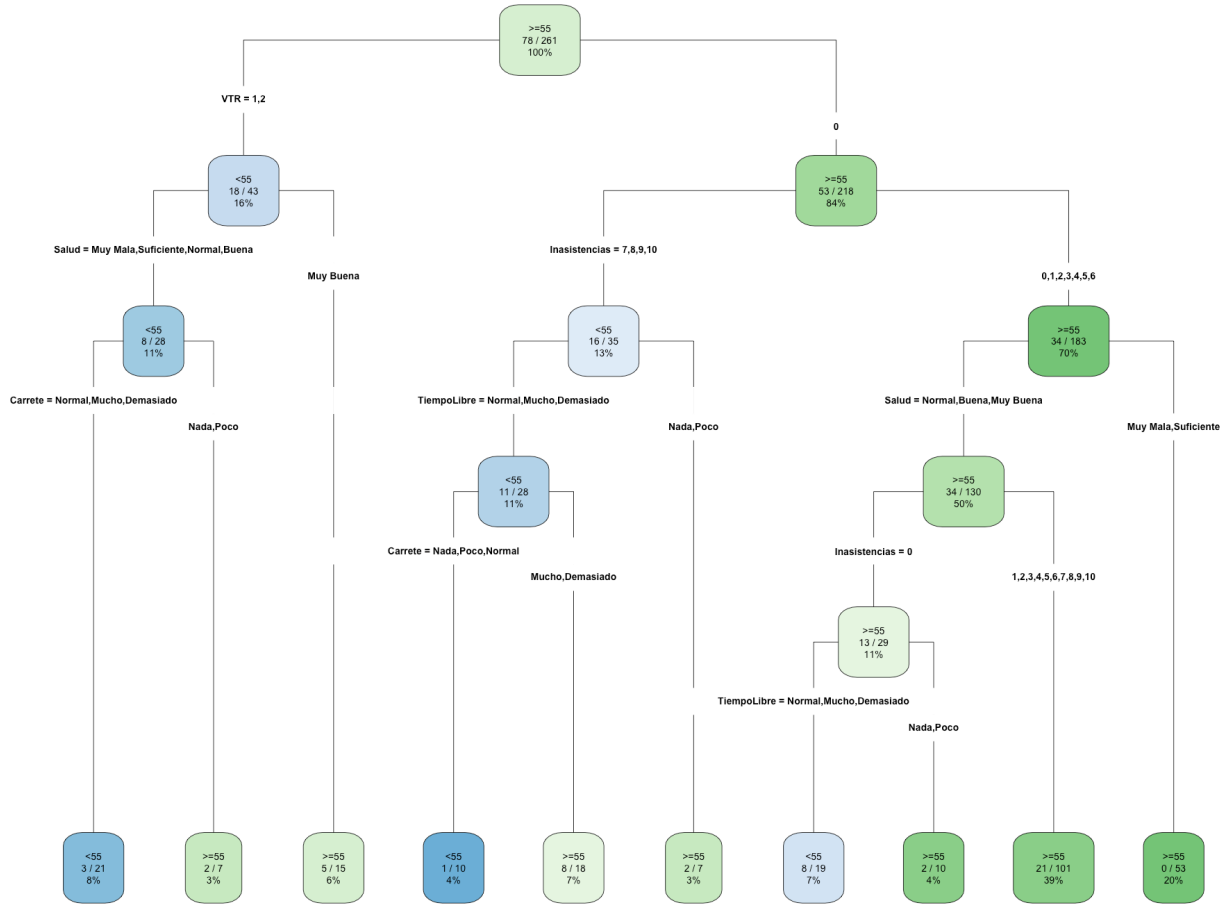

Figura 6: Árbol con 30 % de los datos

Código utilizado: Se genera igual al árbol principal, pero usando el 30 % de los datos.

```
# Seleccion de datos
Data_Sample <- sample_frac(Data, .3)
Data_Test <- setdiff(Data, Data_Sample)
# Generacion de arbol
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal + VTR + TiempoLibre +
Carrete + Salud + Inasistencias, data = Data_Sample, method = "class")
# Resultados para analisis
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
prediccion_1 <- predict(arbol, newdata = Data_Test, type = "class")
confusionMatrix(prediccion_1, Data_Test[["NotaFinal"]])
```

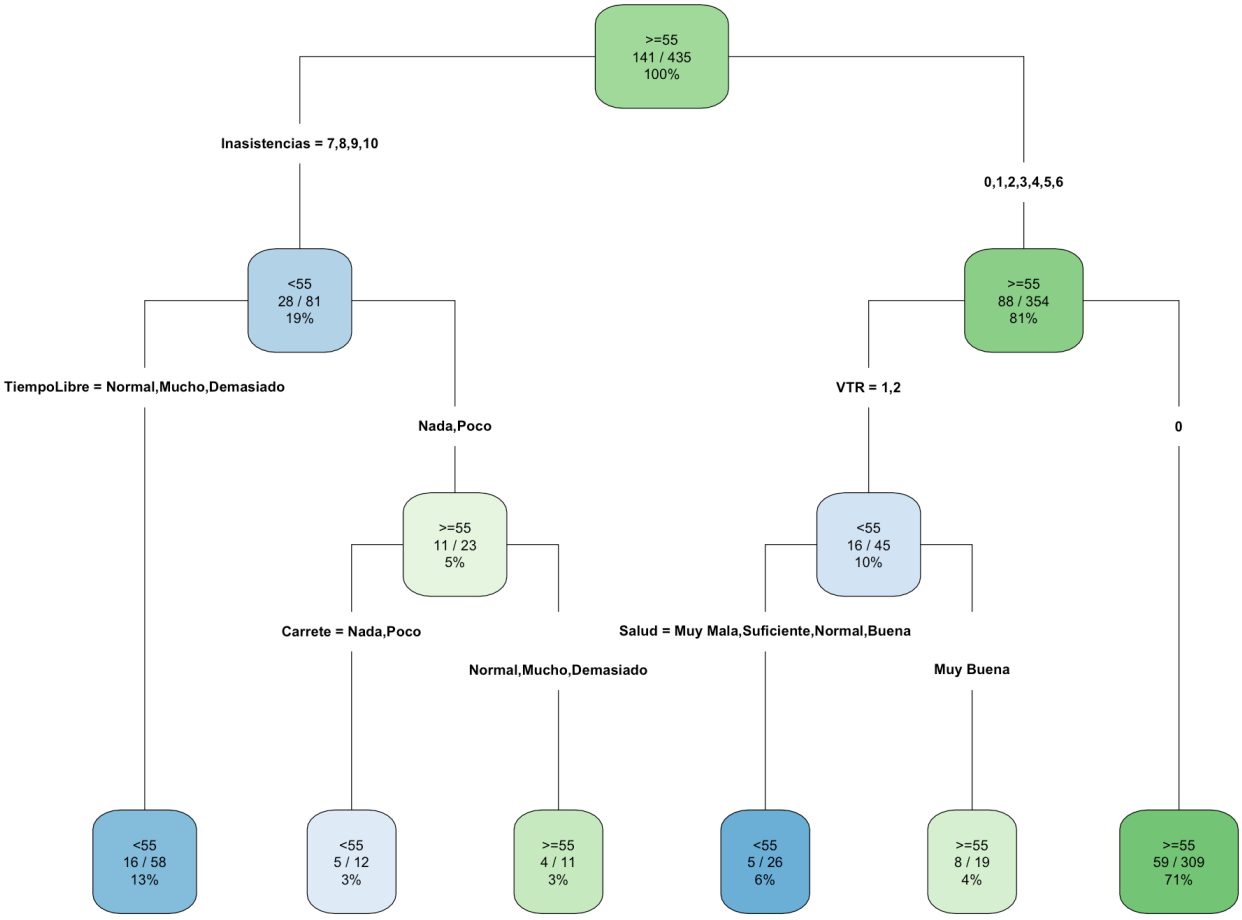

Figura 7: Árbol con 50 % de los datos

Código utilizado: Se genera igual al árbol principal, pero usando el 50 % de los datos.

```
# Seleccion de datos
Data_Sample <- sample_frac(Data, .5)
Data_Test <- setdiff(Data, Data_Sample)
# Generacion de arbol
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal + VTR + TiempoLibre +
Carrete + Salud + Inasistencias, data = Data_Sample, method = "class")
# Resultados para analisis
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
prediccion_1 <- predict(arbol, newdata = Data_Test, type = "class")
confusionMatrix(prediccion_1, Data_Test[["NotaFinal"]])
```

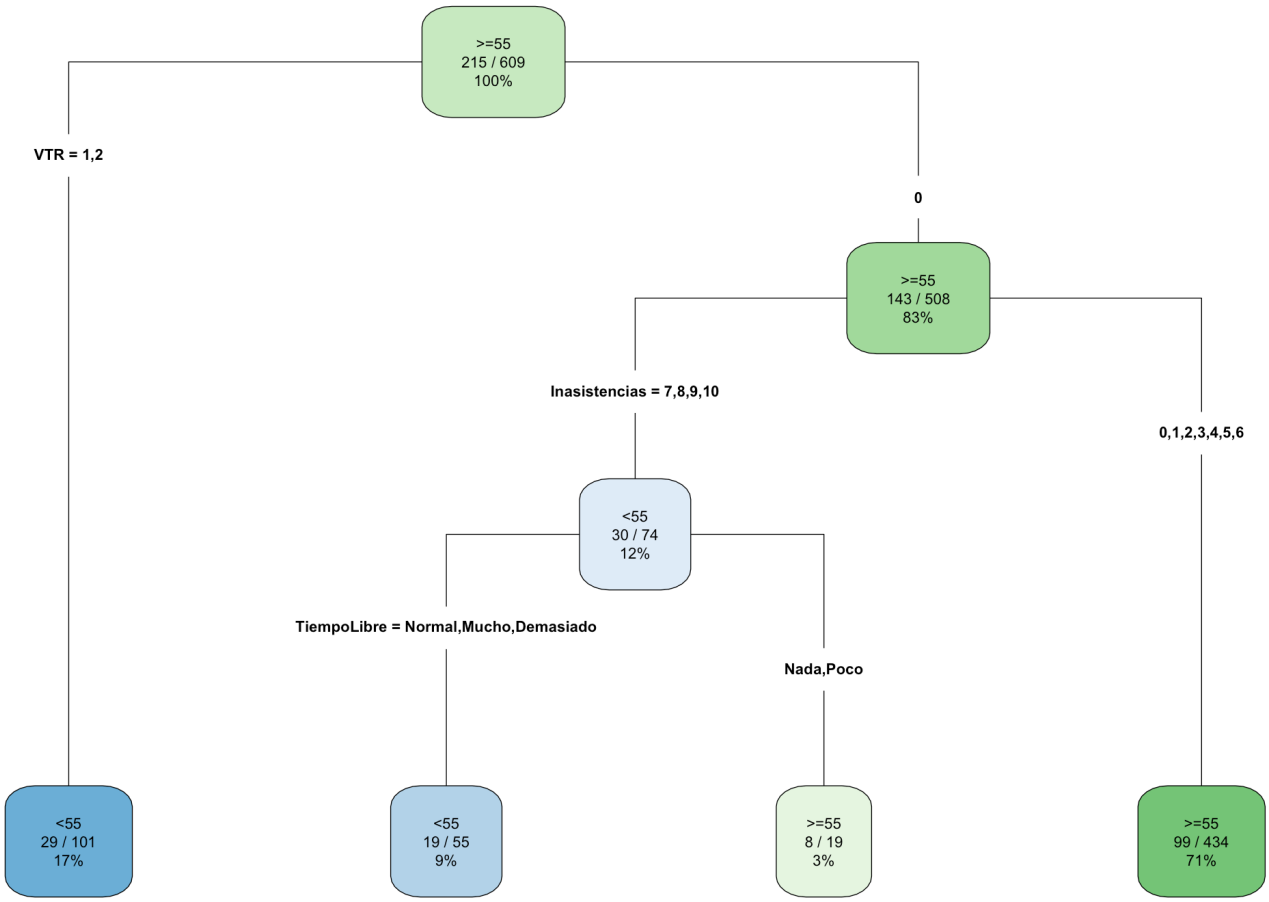

Figura 8: Árbol con 70 % de los datos

Código utilizado: Se genera igual al árbol principal, pero usando el 70 % de los datos.

```
# Seleccion de datos
Data_Sample <- sample_frac(Data, .7)
Data_Test <- setdiff(Data, Data_Sample)
# Generacion de arbol
arbol <- rpart(NotaFinal ~ Sexo + HorasEstudioSemanal + VTR + TiempoLibre +
Carrete + Salud + Inasistencias, data = Data_Sample, method = "class")
# Resultados para analisis
summary(arbol)
rpart.plot(arbol, type = 4, extra = 103)
prediccion_1 <- predict(arbol, newdata = Data_Test, type = "class")
confusionMatrix(prediccion_1, Data_Test[["NotaFinal"]])
```

Además, se incluye una tabla comparativa para poder evaluar los árboles obtenidos y los errores de

clasificación asociados.

Árboles	30 %	50 %	70 %	80 %
Muestra de training	29.9	32.4	35.3	36.5
Muestra de testing	39.6	32.9	27.6	27.0

Cuadro 1: Errores de clasificación (en %)

Se puede ver así que los resultados de clasificación de los árboles dependen de la cantidad de datos escogidos, además de cuales son escogidos.

Como se puede ver en la tabla, para el 30 % de los datos el error de testing es mucho mayor al de training, lo que, junto a la figura del árbol, se ve que existe un problema de overfitting, es decir el árbol está hecho muy específico para los datos que se tienen, por lo que tiene mal rendimiento en la muestra de testing. Esto se debe a la falta de una muestra representativa de datos, por lo que es necesario una muestra más grande. Esto se mejora considerablemente para los árboles de 50 % y de 70 %, donde en cada uno se van reconociendo mejor los atributos predictores para hacer más clara la clasificación, siendo el de 70 % el que presenta una mejor precisión. Por último, resulta importante comparar el de 70 % con el original de 80 %, donde pese a tener una mejor precisión el original, se puede cuestionar si dicha precisión es representativa debido a que al usar más datos para el training se disminuyen los que se usan para el testing. Es por esto que se considera que los mejores resultados están en el árbol de 70 %, pero que podría seguir evaluándose para training sets entre 50 % y 80 %.

Comandos relevantes: rpart, rpart.plot, summary, predict, confusionMatrix, sample_fac, setdiff.