TP1

Catalina Dolhare, Camila Cauzzo, Renata Squillari

2024-08-30

# 1. Introducción al problema

El siguiente conjunto de datos presenta información acerca de una encuesta de satisfacción a pasageros de una cierta aerolinea. Obtuvimos este dataset de Kaggle[[1]](#footnote-20), el cual cuenta con un conjunto de train con 104000 datos, representando el 80% del data set, y un conjunto de test con 26000 datos, representando el 20% del data set. Este data set cuenta con 23 variables, como “Type of travel”, “Class”, “Flight distance”, “Departure Delay in Minutes”, “Arrival Delay in Minutes”. Principalmente, cuenta con “Satisfaction”, variable categorica que puede tomar dos opciones: “Satisfaction” o “neutral or dissatisfaction”.Con toda esta información, intentaremos predecir la satisfacción de un pasajero dado las respuesta a las variables.

Utilizamos este conjunto de datos dado que contiene diversas variables predictoras y no tiene un gran numero de datos faltantes (solo se encuentran datos faltas en una de las variables), por lo que podremos predecir un buen modelo.

Para el siguiente trabajo se necesitan importar las siguientes librerias:

#install.packages("ggplot2")  
#install.packages("corrplot")  
#install.packages("MLmetrics")  
  
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.2.3

library(corrplot)

## corrplot 0.94 loaded

library(rpart)

## Warning: package 'rpart' was built under R version 4.2.3

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.2.3

library(MLmetrics)

## Warning: package 'MLmetrics' was built under R version 4.2.3

##   
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':  
##   
## Recall

library(ROCR)

## Warning: package 'ROCR' was built under R version 4.2.3

# 2. Preparación de los datos

Como nuestro data set venia con dos conjuntos de datos que sumaban un total de aproximadamente 130.000 observaciones, decidimos juntar todos los datos y elegir aleatoriamente 50.000 datos. Para hacer esto y que sea repicable los datos que se eligan, utilizamos una semilla.

datos\_train <- read.csv("train.csv")  
datos\_test <- read.csv("test.csv")

datos<-rbind(datos\_train, datos\_test)

set.seed(123)  
sample\_datos <- datos[sample(nrow(datos), 50000),]  
nrow(sample\_datos)

## [1] 50000

Para la variable satisfaction, que es lo que queremos predecir, originalmente era categorica, pero decidimos transformarla en numerica donde satisfied = 1 y neutral o disatisfied = 0. Esto lo hicimos para poder analizar graficamente las variables predictoras contra satisfaction, dado que no podiamos realizarlo si esta era categorica.

sample\_datos$satisfaction <- as.numeric(as.factor(sample\_datos$satisfaction)) - 1

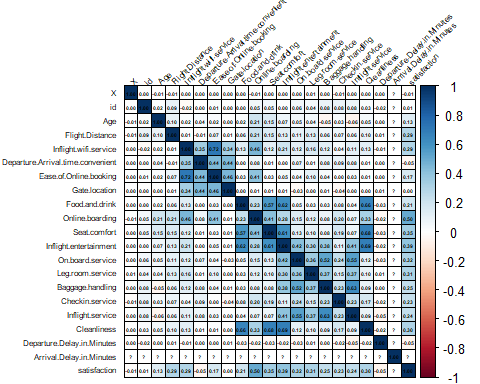
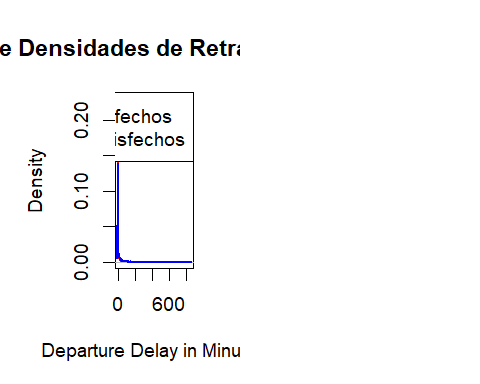
Para poder ver las estadisticas descriptivas de las varibales, realizamos un summary de los datos. Este nos provee el minimo, los quartiles, la media, la mediana y el maximo de las variables numericas.

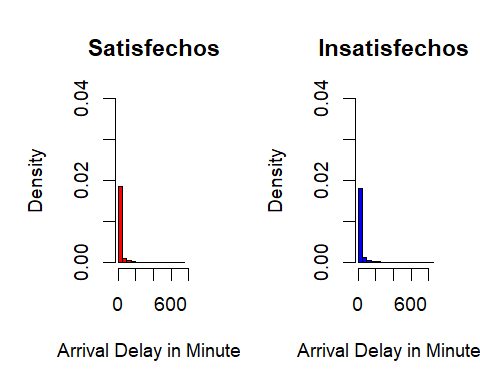
summary(sample\_datos)

## X id Gender Customer.Type   
## Min. : 0 Min. : 3 Length:50000 Length:50000   
## 1st Qu.: 16126 1st Qu.: 32529 Class :character Class :character   
## Median : 38758 Median : 64899 Mode :character Mode :character   
## Mean : 44065 Mean : 64899   
## 3rd Qu.: 71427 3rd Qu.: 97113   
## Max. :103902 Max. :129880   
##   
## Age Type.of.Travel Class Flight.Distance  
## Min. : 7.00 Length:50000 Length:50000 Min. : 31   
## 1st Qu.:27.00 Class :character Class :character 1st Qu.: 414   
## Median :40.00 Mode :character Mode :character Median : 848   
## Mean :39.44 Mean :1190   
## 3rd Qu.:51.00 3rd Qu.:1744   
## Max. :85.00 Max. :4983   
##   
## Inflight.wifi.service Departure.Arrival.time.convenient Ease.of.Online.booking  
## Min. :0.000 Min. :0.000 Min. :0.000   
## 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000   
## Median :3.000 Median :3.000 Median :3.000   
## Mean :2.738 Mean :3.058 Mean :2.762   
## 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:4.000   
## Max. :5.000 Max. :5.000 Max. :5.000   
##   
## Gate.location Food.and.drink Online.boarding Seat.comfort   
## Min. :1.00 Min. :0.000 Min. :0.000 Min. :1.000   
## 1st Qu.:2.00 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000   
## Median :3.00 Median :3.000 Median :3.000 Median :4.000   
## Mean :2.98 Mean :3.213 Mean :3.254 Mean :3.447   
## 3rd Qu.:4.00 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:5.000   
## Max. :5.00 Max. :5.000 Max. :5.000 Max. :5.000   
##   
## Inflight.entertainment On.board.service Leg.room.service Baggage.handling  
## Min. :0.000 Min. :0.000 Min. :0.000 Min. :1.000   
## 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:3.000   
## Median :4.000 Median :4.000 Median :4.000 Median :4.000   
## Mean :3.365 Mean :3.377 Mean :3.347 Mean :3.622   
## 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:5.000   
## Max. :5.000 Max. :5.000 Max. :5.000 Max. :5.000   
##   
## Checkin.service Inflight.service Cleanliness Departure.Delay.in.Minutes  
## Min. :1.000 Min. :0.000 Min. :0.000 Min. : 0.00   
## 1st Qu.:2.000 1st Qu.:3.000 1st Qu.:2.000 1st Qu.: 0.00   
## Median :3.000 Median :4.000 Median :3.000 Median : 0.00   
## Mean :3.303 Mean :3.631 Mean :3.293 Mean : 14.67   
## 3rd Qu.:4.000 3rd Qu.:5.000 3rd Qu.:4.000 3rd Qu.: 12.00   
## Max. :5.000 Max. :5.000 Max. :5.000 Max. :853.00   
##   
## Arrival.Delay.in.Minutes satisfaction   
## Min. : 0.00 Min. :0.0000   
## 1st Qu.: 0.00 1st Qu.:0.0000   
## Median : 0.00 Median :0.0000   
## Mean : 15.08 Mean :0.4338   
## 3rd Qu.: 13.00 3rd Qu.:1.0000   
## Max. :823.00 Max. :1.0000   
## NA's :149

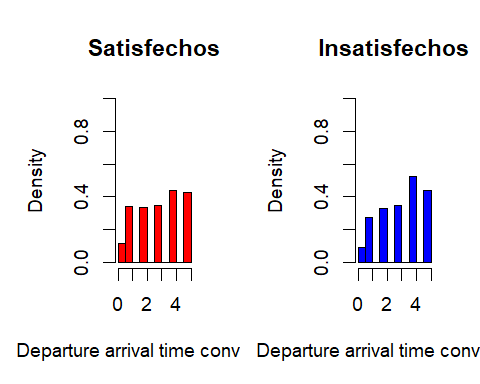
A simple vista, vemos en la variable de satisfaction una mediana de 0 y una media 0.4338. Esto nos indica que hay mas pasajeros insatisfechos (0) que satisfechos (1).

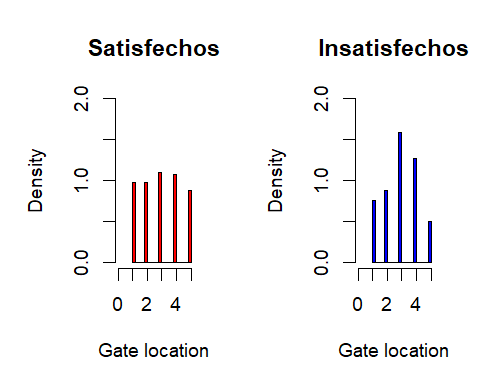
Por otro lado, notamos que las variables que tiene una escala de 0 a 5 tiene mediana y media en aproximadamente 3, esto significa que en su mayoria las personas se encuentran indiferente dentro de cada categoría.

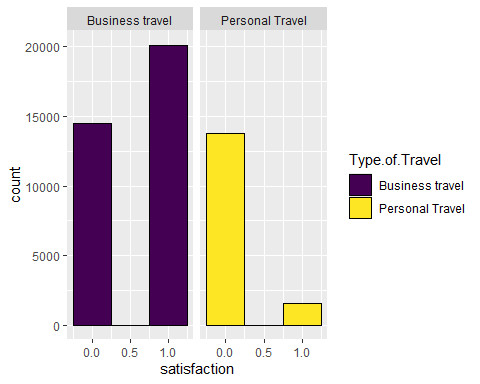
Para ver si las otras variables predictoras influyen en satisfacion hicimos una matriz de correlacion. Sabemos que correlacion no implica causalidad, no obstante, es bueno para visualizar y descartar variables que no influyen directamente en la satisfación de los pasajeros.  Se puede ver que hay algunas variables que no tienen correlación con la satisfación, y son “departure arrival time convenient”, “gate of location”, “departure delay in minutes”, “departure arrival in minutes”. Como no podemos descartar que no haya causalidad, analizamos estas variables para ver si inciden sobre la satisfación. 

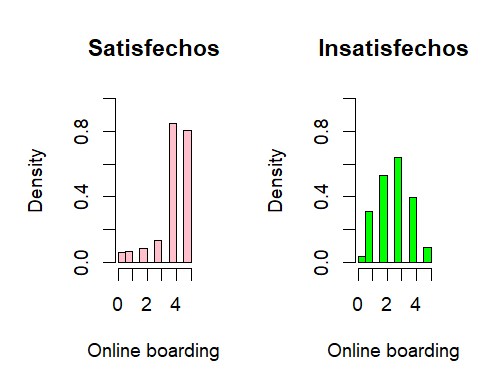


En cuanto a ambos delays, notamos que no afectan a la satisfacción de una persona, dado que la distribución es muy similar entre los satisfechos y no satisfechos.



 Al graficar estas 4 variables, podemos ver que la ditribución entre pasajeros satisfechos e insatisfechos es muy similar por lo que concluimos que no impactan en lo que queremos predecir.

Por la misma razon de que correlación on implica causalidad, decidimos probar en algunas variables si existia este tipo de relación. El siguiente grafico muestra la relación entre la variable “Type of Travel” y “satisfaction”:  En este, podemos ver que existe una clara división entre el tipo de viaje. Podemos concluir que las personas que viajan por placer o temas personales suelen estar más insatisfechos que los que viajan por negocios. Viendo la distribución podemos ver que es una variable que realmente influye en lo que queremos predecir.

En el siguiente histograma, queremos demostrar la relación entre la variable “Online boarding” y “satisfaction”.  Vemos que las distribuciones entre ambos son diferentes por lo que entendemos a “online boarding” como una variable influyente en la satisfacción del pasajero. Tiene sentido el grafico ya que los pasajeros satisfechos puntuan con una nota alta el online boarding mientras que los pasajeros insatisfechos puntuan con una nota más baja.

# 3. Construcción de un árbol de decisión básico

Para construir el arbol, dividimos nuestro conjunto de datos en entrenamiento, con 70% de los datos, validación, con 15% de los datos, y testo, con 15% de los datos. Como nuestro sample de datos ya fue elegido al azar de un conjunto más grande, simplemente podemos dividir entre los respectivos porcentajes y nos seguiremos garantizando que sea aleatorio.

entrenamiento <- sample\_datos[1:35000,]  
validacion <- sample\_datos[35001:42500,]  
testeo <- sample\_datos[42501:50000,]

Con la librería rpart generamos un arbol de decision a partir de los datos de entrenamiento, utilizando los hiperparamentros por defecto.

tree <- rpart(formula = satisfaction ~ Gender + Customer.Type + Age + Type.of.Travel + Class + Flight.Distance + Inflight.wifi.service + Departure.Arrival.time.convenient + Ease.of.Online.booking + Gate.location + Food.and.drink + Online.boarding + Seat.comfort + Inflight.entertainment + On.board.service + Leg.room.service + Baggage.handling + Checkin.service + Inflight.service + Cleanliness + Departure.Delay.in.Minutes + Arrival.Delay.in.Minutes,   
 data = entrenamiento,   
 method = "class")  
tree$control

## $minsplit  
## [1] 20  
##   
## $minbucket  
## [1] 7  
##   
## $cp  
## [1] 0.01  
##   
## $maxcompete  
## [1] 4  
##   
## $maxsurrogate  
## [1] 5  
##   
## $usesurrogate  
## [1] 2  
##   
## $surrogatestyle  
## [1] 0  
##   
## $maxdepth  
## [1] 30  
##   
## $xval  
## [1] 10

Dentro de los hiperparametros que toma el árbol por defecto nos encontramos que minsplit toma por defecto el valor de 20. Esto quiere decir que el numero minimo de observaciones que se necesita para dividir un nodo intermedio es de 20.

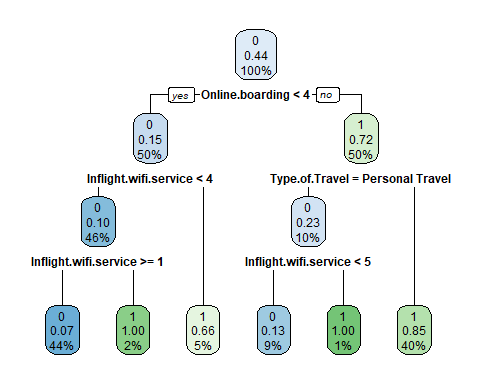
En cuanto a minbucket, este toma un valor de 7. Lo que refiere a que se necesita un minimo de 7 observaciones en cada hoja de un arbol. Esto evita que las hojas del arbol tenga muy pocas observaciones y logra que no tengamos un modelo overfitted.

El hiperparametro cp, es el parametro de complejidad para la poda del arbol. Es de 0.01 por lo que significa que una división en el arbol debe reducir el error en al menos 1% para ser considerada util.

Maxdepth es la maxima produnfidad permitida para la construcción del árbol y en este caso, puede tener hasta 30 niveles. Si bien cuanto más grande el árbol mejor se ajusta a los datos, si el arbol es muy profundo, se puede generar overfitting.

Por ultimo, xval toma un valor de 10, por lo que se utilizaran 10 grupos para validación cruzada. Un mayor valor de xval te asegura una estimación más precisa pero mayor tiempo de computo.

rpart.plot(tree)



En general, cada nodo cuenta con el porcentaje de datos que caen en ese nodo del total, la proporción de datos que corresponden con un dato satisfecho (satisfied = 1) y el color del nodo indica la pureza del mismo.

En el nodo raiz encontramos el 100% de los datos, de los cuales el 44% se corresponde con pasajeros satisfechos. El color del nodo no es tan intenso, dado que en este se encuentran todos los datos, los satisfechos como los insatisfecho. Como la clase predominante es insatisfecho, el nodo contiene un 0.

El factor mas importante para determinar la satisfacción del pasajero es el online boarding. Este divide a los datos exactamente a la mitad dependiendo de si toman un valor <4. Si esto es asi, toman la rama de la izquierda, y si no, la de la derecha.

Por la rama de la izquierda, se encuentra un nodo donde predomina la insatisfacción, al tener 0.15 de los datos satisfechos. En este podemos ver un color más puro en comparación al nodo raiz. Desde aca, se vuelve a dividir segun la variable predictoria Inflight Wifi Service. Si este es menor a 4, toma la rama de la izquierda, y si no toma la rama de la derecha.

En el caso de que tome la rama de la izquierda, vemos que los datos caen en un nodo donde predominan datos insatisfechos, con una proporcion de datos satisfechos del 0.10. Por esta razon vemos un 0 y un color muy puro. Por otro lado, vemos que la gran mayoría de los datos que caen en el nodo anterior, luego caen en este nodo al tener el 46% de los datos totales. Este nodo es divido por la misma variable predictora, con la condicion de que ahora si es mayor o igual a 1 toma la rama de la izquierda, y si no toma la rama de la derecha.

En la hoja terminal de la izquierda, vemos que si un dato cae alli es predicho como insatisfecho, dado que, en el entrenamiento, solo el 7% de los satisfechos caen alli. Podemos ver que la gran mayoría de los datos que caen en el nodo anterior estaran en este (44% del total). Por otro lado, si va hacia la hoja terminal de la derecha, este predecira como satisfecho, dado que el 100% de los datos que cayeron en este estan satisfechos. Por esta razon notamos el color más puro.

Volviendo al nodo que es divido por Inflight Wifi Service < 4, si se toma la rama de la derecha, llegamos a una hoja terminal que predecira como satisfecho. Sin embargo, este no es muy puro dado que en el entramiento solamente el 0.66 de los datos estaba satisfecho.

Volviendo al nodo raiz, si tomamos la rama de la derecha caemos en un nodo donde predomina la satisfacción, con una proporción del 0.72 de los datos estando satisfechos. Este es dividido por si la variable Typo of Travel es igual a Personal Travel o no. Si lo es, toma la rama de la izquierda. Si no lo es, es decir el type of travel es business travel, toma la rama de la derecha.

Si vamos a la rama de la izquierda, vemos que caemos en un nodo donde predominan los datos insatisfechos dado que contiene el 23% de datos satisfechos. Este es nuevamente dividio por la variable predictora Inflight Wifi Service. Si esta toma un valor menor a 5, va hacia la hoja terminal de la izquierda donde predice como insatisfechos. En cambio, si la variable es igual a 5, predice los datos como satisfechos.

Si vamos a la rama de la derecha, caemos en una hoja terminal que predice los datos como satisfechos.

# 4. Evaluación del árbol de decisión básico

predictions\_clase <- predict(tree, newdata = testeo, type = "class")  
predictions\_prob <- predict(tree, newdata = testeo, type = "prob")

ConfusionMatrix(y\_pred = predictions\_clase, y\_true = testeo$satisfaction)

## y\_pred  
## y\_true 0 1  
## 0 3729 576  
## 1 289 2906

Segun la matriz de confusion obtenida, podemos ver que a simple vista que el modelo predice correctamente la gran mayoría de las veces. Notamos que la categoria con mayor observaciones es True Negative, es decir los datos insatisfechos que se predijeron correctamente como insatisfechos. Este tiene 3729 datos, un 49.72%. La segunda categoría es True Positive, es decir los datos que son satisfechos que son predichos como tal, con el 38.746%. El porcentaje restante de los datos es predicho incorrectamente, con un 7.68% que fueron predichos como insatisfechos cuando estaban satisfechos, y un 3.853% fueron predichos como satisfechos cuando estaban insatisfechos.

Accuracy(y\_pred = predictions\_clase, y\_true = testeo$satisfaction)

## [1] 0.8846667

Como mencionamos anteriormente, el modelo predice correctamente el 88.46667% de las observaciones, la suma de los porcentajes de True-Positive y True-Negative.

Precision(y\_pred = predictions\_clase, y\_true = testeo$satisfaction)

## [1] 0.9280737

La precision indica el porcentaje de los datos que dije que son positivos, cuantos realmente lo son. En nuestro modelo la precision es del 92.80737%, por lo que el de todos los datos positivos le acerto al 0.92

Recall(y\_pred = predictions\_clase, y\_true = testeo$satisfaction)

## [1] 0.8662021

Recall mide de los datos que efectivamente son positivos, cuantos dice el modelo que lo son. En este caso, nuestro modelo identifico el 86.62021% de los datos positivos.

F1\_Score(y\_pred = predictions\_clase, y\_true = testeo$satisfaction)

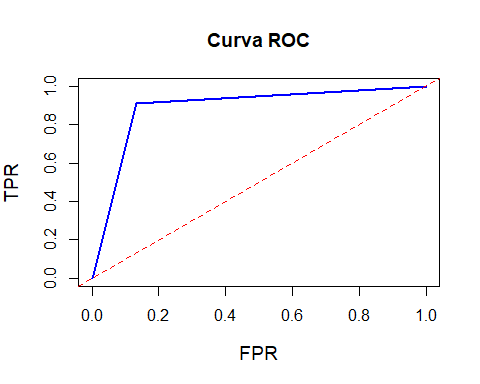
## [1] 0.8960711

Un F1-score de 0.8960711 indica que el modelo tiene un buen balance entre precision y recall.

AUC(y\_pred = predictions\_clase, y\_true = testeo$satisfaction)

## [1] 0.8878741

pred <- prediction(as.numeric(predictions\_clase), as.numeric(testeo$satisfaction))  
roc\_curve <- performance(pred, "tpr", "fpr")  
  
plot(roc\_curve,   
 main="Curva ROC",   
 col="blue",   
 lwd=2,   
 xlab="FPR",   
 ylab="TPR",   
 cex.main=1.2,   
 cex.lab=1.2)  
  
abline(a=0, b=1, lty=2, col="red")



AUC-ROC se refiere al area debajo de la curva ROC. Al tener un valor de AUC de 0.8878741 podemos decir que nuestro modelo distingue bien entre clases positivas y negativas, esto se asemeja con lo que se ve en el grafico.

Analizando todas las metricas de performance, podemos concluir que nuestro modelo es muy eficiente al predecir la satisfacción de una persona, dado que se vio que tiene unabuena performance.

# 5. Optimización del modelo

valores\_maxdepth <- c(2, 5, 7, 10, 12, 15)  
valores\_minsplit <- c(15, 20, 25, 30, 35)  
valores\_minbucket <- c(10, 20, 30, 40, 50)

buscar\_hiperparametros <- function(entrenamiento, validacion, valores\_maxdepth, valores\_minsplit, valores\_minbucket){  
 valores\_auc <- c()  
 best\_auc <- 0  
 best\_params <- c()  
 for (i in 1:length(valores\_maxdepth)){  
 for (j in 1:length(valores\_minsplit)){  
 for (l in 1:length(valores\_minbucket)){  
 tree2 <- rpart(formula = satisfaction ~ Gender + Customer.Type + Age + Type.of.Travel + Class + Flight.Distance + Inflight.wifi.service + Departure.Arrival.time.convenient + Ease.of.Online.booking + Gate.location + Food.and.drink + Online.boarding + Seat.comfort + Inflight.entertainment + On.board.service + Leg.room.service + Baggage.handling + Checkin.service + Inflight.service + Cleanliness + Departure.Delay.in.Minutes + Arrival.Delay.in.Minutes,   
 data = entrenamiento,   
 method = "class",  
 control = rpart.control(maxdepth = valores\_maxdepth[i], minsplit = valores\_minsplit[j], minbucket = valores\_minbucket[l], cp = 0, xval = 0))  
   
 predicciones <- predict(tree2, newdata = validacion, type = "class")  
  
 auc <- AUC(y\_pred = predicciones, y\_true = validacion$satisfaction)  
 valores\_auc <- c(valores\_auc, auc)  
   
 if (auc > best\_auc){  
 best\_auc <- auc  
 best\_params <- c(valores\_maxdepth[i], valores\_minsplit[j], valores\_minbucket[l])  
 }  
 }  
 }  
 }  
 return (list(valores\_auc = valores\_auc, best\_auc = best\_auc, best\_params = best\_params))  
}

best <- buscar\_hiperparametros(entrenamiento = entrenamiento, validacion = validacion, valores\_maxdepth = valores\_maxdepth, valores\_minsplit = valores\_minsplit, valores\_minbucket = valores\_minbucket)  
best$valores\_auc

## [1] 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880  
## [8] 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880  
## [15] 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880 0.8542880  
## [22] 0.8542880 0.8542880 0.8542880 0.8542880 0.8986303 0.8986303 0.8986303  
## [29] 0.8986303 0.8972491 0.8986303 0.8986303 0.8986303 0.8986303 0.8972491  
## [36] 0.8986303 0.8986303 0.8986303 0.8986303 0.8972491 0.8986303 0.8986303  
## [43] 0.8986303 0.8986303 0.8972491 0.8986303 0.8986303 0.8986303 0.8986303  
## [50] 0.8972491 0.9346188 0.9345832 0.9338182 0.9322258 0.9286826 0.9346188  
## [57] 0.9345832 0.9338182 0.9322258 0.9286826 0.9348545 0.9345832 0.9338182  
## [64] 0.9322258 0.9286826 0.9344764 0.9345832 0.9338182 0.9322258 0.9286826  
## [71] 0.9344764 0.9345832 0.9338182 0.9322258 0.9286826 0.9422889 0.9411411  
## [78] 0.9379452 0.9377339 0.9326339 0.9422889 0.9411411 0.9379452 0.9377339  
## [85] 0.9326339 0.9426069 0.9411411 0.9379452 0.9377339 0.9326339 0.9443038  
## [92] 0.9411411 0.9379452 0.9377339 0.9326339 0.9440079 0.9411411 0.9379452  
## [99] 0.9377339 0.9326339 0.9460521 0.9465702 0.9418285 0.9400225 0.9352429  
## [106] 0.9460521 0.9465702 0.9418285 0.9400225 0.9352429 0.9468305 0.9465702  
## [113] 0.9418285 0.9400225 0.9352429 0.9496016 0.9465702 0.9418285 0.9400225  
## [120] 0.9352429 0.9496839 0.9465702 0.9418285 0.9400225 0.9352429 0.9483319  
## [127] 0.9463503 0.9433055 0.9421379 0.9363527 0.9483319 0.9463503 0.9433055  
## [134] 0.9421379 0.9363527 0.9492638 0.9463503 0.9433055 0.9421379 0.9363527  
## [141] 0.9525199 0.9463503 0.9433055 0.9421379 0.9363527 0.9527912 0.9463503  
## [148] 0.9433055 0.9421379 0.9363527

best$best\_auc

## [1] 0.9527912

best$best\_params

## [1] 15 35 10

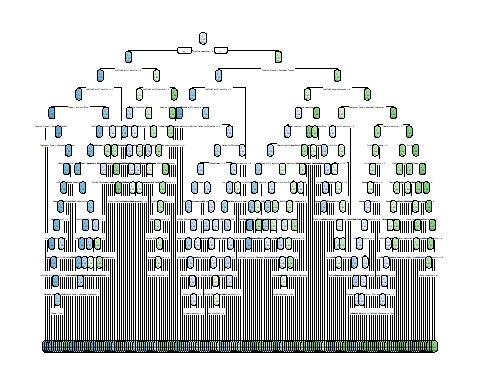
best\_tree <- rpart(formula = satisfaction ~ Gender + Customer.Type + Age + Type.of.Travel + Class + Flight.Distance + Inflight.wifi.service + Departure.Arrival.time.convenient + Ease.of.Online.booking + Gate.location + Food.and.drink + Online.boarding + Seat.comfort + Inflight.entertainment + On.board.service + Leg.room.service + Baggage.handling + Checkin.service + Inflight.service + Cleanliness + Departure.Delay.in.Minutes + Arrival.Delay.in.Minutes,   
 data = entrenamiento,   
 method = "class",  
 control = rpart.control(maxdepth = best$best\_params[1], minsplit = best$best\_params[2], minbucket = best$best\_params[3], cp = 0, xval = 0))  
best\_predicciones <- predict(best\_tree, newdata = testeo, type = "class")  
AUC(y\_pred = best\_predicciones, y\_true = testeo$satisfaction)

## [1] 0.9444537

**6. Interpretación de resultados**

rpart.plot(best\_tree)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



importancia\_variables <- best\_tree$variable.importance  
  
importancia\_variables

## Online.boarding Inflight.wifi.service   
## 6401.41660 5674.63171   
## Seat.comfort Ease.of.Online.booking   
## 3167.37172 2972.66020   
## Inflight.entertainment Class   
## 2665.59937 2570.98078   
## Type.of.Travel Customer.Type   
## 2464.73747 774.08451   
## Age Cleanliness   
## 527.12625 512.05062   
## Inflight.service Departure.Arrival.time.convenient   
## 450.24023 445.41866   
## Gate.location Leg.room.service   
## 435.30226 378.71427   
## Baggage.handling On.board.service   
## 373.28475 358.21520   
## Checkin.service Food.and.drink   
## 319.55056 301.86181   
## Flight.Distance Arrival.Delay.in.Minutes   
## 260.95097 54.45496   
## Departure.Delay.in.Minutes Gender   
## 44.71892 40.39670

valores\_maxdepth2 <- c(10, 15, 20, 25, 30)  
valores\_minsplit2 <- c(10, 20, 30, 40, 50)  
valores\_minbucket2 <- c(100, 200, 300, 400, 500)

best2 <- buscar\_hiperparametros(entrenamiento = entrenamiento, validacion = validacion, valores\_maxdepth = valores\_maxdepth2, valores\_minsplit = valores\_minsplit2, valores\_minbucket = valores\_minbucket2)  
best2$valores\_auc

## [1] 0.9286129 0.9263117 0.9192997 0.9174051 0.9103306 0.9286129 0.9263117  
## [8] 0.9192997 0.9174051 0.9103306 0.9286129 0.9263117 0.9192997 0.9174051  
## [15] 0.9103306 0.9286129 0.9263117 0.9192997 0.9174051 0.9103306 0.9286129  
## [22] 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997  
## [29] 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306  
## [36] 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117  
## [43] 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051  
## [50] 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742  
## [57] 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997  
## [64] 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306  
## [71] 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117  
## [78] 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051  
## [85] 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742  
## [92] 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997  
## [99] 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306  
## [106] 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117  
## [113] 0.9192997 0.9174051 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051  
## [120] 0.9103306 0.9297742 0.9263117 0.9192997 0.9174051 0.9103306

best2$best\_auc

## [1] 0.9297742

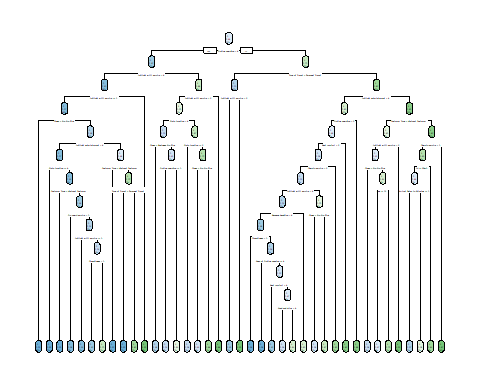
best2$best\_params

## [1] 15 10 100

best\_tree2 <- rpart(formula = satisfaction ~ .,   
 data = entrenamiento,   
 method = "class",  
 control = rpart.control(maxdepth = best2$best\_params[1], minsplit = best2$best\_params[2], minbucket = best2$best\_params[3], cp = 0, xval = 0))  
best\_predicciones <- predict(best\_tree2, newdata = testeo, type = "class")  
AUC(y\_pred = best\_predicciones, y\_true = testeo$satisfaction)

## [1] 0.9276515

rpart.plot(best\_tree2)



valores\_maxdepth3 <- c(30)  
valores\_minsplit3 <- c(1000)  
valores\_minbucket3 <- c(1000, 2000, 3000, 4000, 5000)

best3 <- buscar\_hiperparametros(entrenamiento = entrenamiento, validacion = validacion, valores\_maxdepth = valores\_maxdepth3, valores\_minsplit = valores\_minsplit3, valores\_minbucket = valores\_minbucket3)  
best3$valores\_auc

## [1] 0.8676974 0.8479167 0.8366128 0.7958549 0.7958549

best3$best\_auc

## [1] 0.8676974

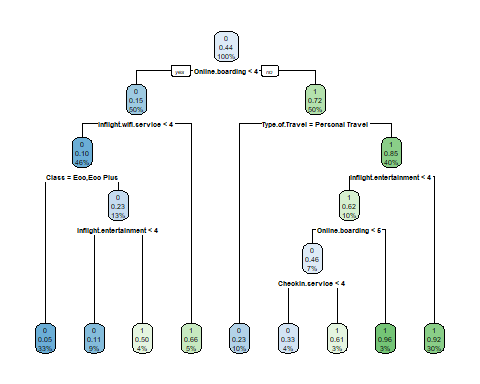
best3$best\_params

## [1] 30 1000 1000

best\_tree3 <- rpart(formula = satisfaction ~ .,   
 data = entrenamiento,   
 method = "class",  
 control = rpart.control(maxdepth = best3$best\_params[1], minsplit = best3$best\_params[2], minbucket = best3$best\_params[3], cp = 0, xval = 0))  
best\_predicciones <- predict(best\_tree3, newdata = testeo, type = "class")  
AUC(y\_pred = best\_predicciones, y\_true = testeo$satisfaction)

## [1] 0.8686509

rpart.plot(best\_tree3)

 **7. Análisis del impacto de los valores faltantes**

reemplazar\_na <- function(data, porcentaje\_na) {  
 total\_filas <- nrow(data)  
 total\_columnas <- ncol(data) # Número total de columnas  
  
 for (i in 1:(total\_columnas - 1)) { # Iterar hasta la anteúltima columna  
 col <- colnames(data)[i]  
   
 # Contar el número actual de NA en la columna  
 na\_existentes <- sum(is.na(data[[col]]))  
   
 # Calcular cuántos NA totales debería haber para alcanzar el porcentaje deseado  
 num\_na\_deseado <- round(porcentaje\_na \* total\_filas)  
   
 nuevos\_na <- num\_na\_deseado - na\_existentes  
   
 # Solo agregar nuevos NA si el número necesario es positivo  
 if (nuevos\_na > 0) {  
 na\_indices <- sample(which(!is.na(data[[col]])), size = nuevos\_na)  
 data[na\_indices, col] <- NA  
 }  
 }  
 return(data)  
}

entrenamiento\_na\_20 <- reemplazar\_na(entrenamiento, 0.20)  
validacion\_na\_20 <- reemplazar\_na(validacion, 0.20)  
testeo\_na\_20 <- reemplazar\_na(testeo, 0.01)  
  
entrenamiento\_na\_50 <- reemplazar\_na(entrenamiento, 0.50)  
validacion\_na\_50 <- reemplazar\_na(validacion, 0.50)  
testeo\_na\_50 <- reemplazar\_na(testeo, 0.50)  
  
entrenamiento\_na\_75 <- reemplazar\_na(entrenamiento, 0.75)  
validacion\_na\_75 <- reemplazar\_na(validacion, 0.75)  
testeo\_na\_75 <- reemplazar\_na(testeo, 0.75)

best\_na\_20 <- buscar\_hiperparametros(entrenamiento = entrenamiento\_na\_20, validacion = validacion\_na\_20, valores\_maxdepth = valores\_maxdepth2, valores\_minsplit = valores\_minsplit2, valores\_minbucket = valores\_minbucket2)  
  
best\_na\_20$valores\_auc

## [1] 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099  
## [8] 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123  
## [15] 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594  
## [22] 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920  
## [29] 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489  
## [36] 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099  
## [43] 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123  
## [50] 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594  
## [57] 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920  
## [64] 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489  
## [71] 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099  
## [78] 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123  
## [85] 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594  
## [92] 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920  
## [99] 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489  
## [106] 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099  
## [113] 0.8736920 0.8682123 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123  
## [120] 0.8533489 0.8958594 0.8872099 0.8736920 0.8682123 0.8533489

best\_na\_20$best\_auc

## [1] 0.8958594

best\_na\_20$best\_params

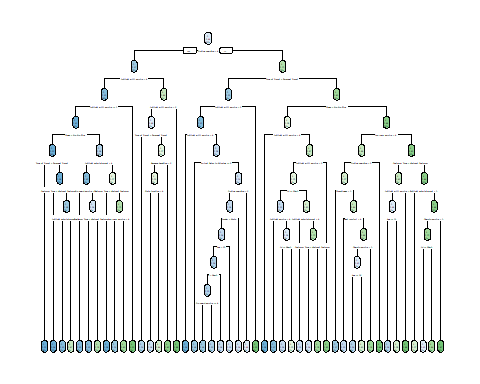
## [1] 10 10 100

best\_tree\_na\_20 <- rpart(formula = satisfaction ~ .,   
 data = entrenamiento\_na\_20,   
 method = "class",  
 control = rpart.control(maxdepth = best\_na\_20$best\_params[1], minsplit = best\_na\_20$best\_params[2], minbucket = best\_na\_20$best\_params[3], cp = 0, xval = 0))  
best\_predicciones <- predict(best\_tree\_na\_20, newdata = testeo\_na\_20, type = "class")  
AUC(y\_pred = best\_predicciones, y\_true = testeo\_na\_20$satisfaction)

## [1] 0.9229872

rpart.plot(best\_tree\_na\_20)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



best\_na\_50 <- buscar\_hiperparametros(entrenamiento = entrenamiento\_na\_50, validacion = validacion\_na\_50, valores\_maxdepth = valores\_maxdepth, valores\_minsplit = valores\_minsplit, valores\_minbucket = valores\_minbucket)  
  
  
best\_na\_50$valores\_auc

## [1] 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323  
## [8] 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323  
## [15] 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323 0.7170323  
## [22] 0.7170323 0.7170323 0.7170323 0.7170323 0.7876132 0.7877912 0.7867059  
## [29] 0.7867059 0.7867059 0.7876132 0.7877912 0.7867059 0.7867059 0.7867059  
## [36] 0.7876132 0.7877912 0.7867059 0.7867059 0.7867059 0.7876132 0.7877912  
## [43] 0.7867059 0.7867059 0.7867059 0.7876132 0.7877912 0.7867059 0.7867059  
## [50] 0.7867059 0.8072491 0.8076629 0.8051583 0.8053608 0.8061147 0.8072491  
## [57] 0.8076629 0.8051583 0.8053608 0.8061147 0.8072491 0.8076629 0.8051583  
## [64] 0.8053608 0.8061147 0.8072491 0.8076629 0.8051583 0.8053608 0.8061147  
## [71] 0.8072491 0.8076629 0.8051583 0.8053608 0.8061147 0.8270031 0.8275790  
## [78] 0.8267587 0.8255112 0.8252952 0.8270031 0.8275790 0.8267587 0.8255112  
## [85] 0.8252952 0.8273567 0.8275790 0.8267587 0.8255112 0.8252952 0.8268963  
## [92] 0.8275790 0.8267587 0.8255112 0.8252952 0.8270965 0.8275790 0.8267587  
## [99] 0.8255112 0.8252952 0.8328491 0.8307068 0.8283629 0.8271264 0.8256582  
## [106] 0.8328491 0.8307068 0.8283629 0.8271264 0.8256582 0.8332027 0.8307068  
## [113] 0.8283629 0.8271264 0.8256582 0.8325532 0.8307068 0.8283629 0.8271264  
## [120] 0.8256582 0.8327533 0.8307068 0.8283629 0.8271264 0.8256582 0.8375321  
## [127] 0.8363415 0.8325879 0.8308824 0.8289403 0.8375321 0.8363415 0.8325879  
## [134] 0.8308824 0.8289403 0.8378145 0.8363415 0.8325879 0.8308824 0.8289403  
## [141] 0.8369760 0.8363415 0.8325879 0.8308824 0.8289403 0.8371761 0.8363415  
## [148] 0.8325879 0.8308824 0.8289403

best\_na\_50$best\_auc

## [1] 0.8378145

best\_na\_50$best\_params

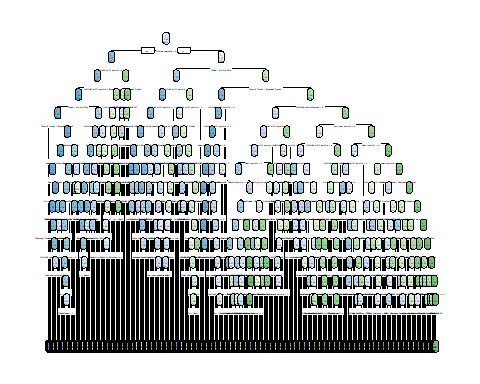
## [1] 15 25 10

best\_tree\_na\_50 <- rpart(formula = satisfaction ~ .,   
 data = entrenamiento\_na\_50,   
 method = "class",  
 control = rpart.control(maxdepth = best\_na\_50$best\_params[1], minsplit = best\_na\_50$best\_params[2], minbucket = best\_na\_50$best\_params[3], cp = 0, xval = 0))  
best\_predicciones <- predict(best\_tree\_na\_50, newdata = testeo\_na\_50, type = "class")  
AUC(y\_pred = best\_predicciones, y\_true = testeo\_na\_50$satisfaction)

## [1] 0.8285322

rpart.plot(best\_tree\_na\_50)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



best\_na\_75 <- buscar\_hiperparametros(entrenamiento = entrenamiento\_na\_75, validacion = validacion\_na\_75, valores\_maxdepth = valores\_maxdepth, valores\_minsplit = valores\_minsplit, valores\_minbucket = valores\_minbucket)  
  
best\_na\_75$valores\_auc

## [1] 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348  
## [8] 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348  
## [15] 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348 0.6228348  
## [22] 0.6228348 0.6228348 0.6228348 0.6228348 0.6989493 0.6987602 0.6976859  
## [29] 0.6983600 0.6983600 0.6989493 0.6987602 0.6976859 0.6983600 0.6983600  
## [36] 0.6989493 0.6987602 0.6976859 0.6983600 0.6983600 0.6989493 0.6987602  
## [43] 0.6976859 0.6983600 0.6983600 0.6989493 0.6987602 0.6976859 0.6983600  
## [50] 0.6983600 0.7258176 0.7257108 0.7259465 0.7270453 0.7263958 0.7258176  
## [57] 0.7257108 0.7259465 0.7270453 0.7263958 0.7258176 0.7257108 0.7259465  
## [64] 0.7270453 0.7263958 0.7258176 0.7257108 0.7259465 0.7270453 0.7263958  
## [71] 0.7258176 0.7257108 0.7259465 0.7270453 0.7263958 0.7344331 0.7340304  
## [78] 0.7331293 0.7348776 0.7331452 0.7344331 0.7340304 0.7331293 0.7348776  
## [85] 0.7331452 0.7344331 0.7340304 0.7331293 0.7348776 0.7331452 0.7346443  
## [92] 0.7340304 0.7331293 0.7348776 0.7331452 0.7345976 0.7340304 0.7331293  
## [99] 0.7348776 0.7331452 0.7415770 0.7411657 0.7405470 0.7421996 0.7405961  
## [106] 0.7415770 0.7411657 0.7405470 0.7421996 0.7405961 0.7415770 0.7411657  
## [113] 0.7405470 0.7421996 0.7405961 0.7417882 0.7411657 0.7405470 0.7421996  
## [120] 0.7405961 0.7422131 0.7411657 0.7405470 0.7421996 0.7405961 0.7472575  
## [127] 0.7454602 0.7443234 0.7450662 0.7447371 0.7472575 0.7454602 0.7443234  
## [134] 0.7450662 0.7447371 0.7472575 0.7454602 0.7443234 0.7450662 0.7447371  
## [141] 0.7474687 0.7454602 0.7443234 0.7450662 0.7447371 0.7475399 0.7454602  
## [148] 0.7443234 0.7450662 0.7447371

best\_na\_75$best\_auc

## [1] 0.7475399

best\_na\_75$best\_params

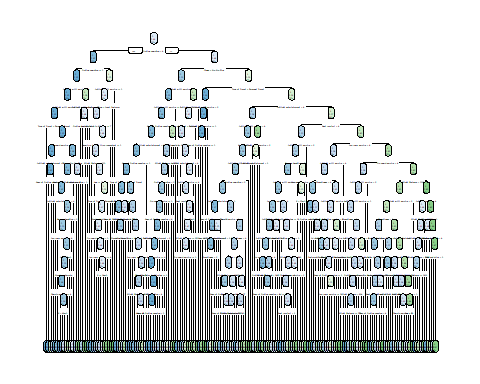
## [1] 15 35 10

best\_tree\_na\_75 <- rpart(formula = satisfaction ~ .,   
 data = entrenamiento\_na\_75,   
 method = "class",  
 control = rpart.control(maxdepth = best\_na\_75$best\_params[1], minsplit = best\_na\_75$best\_params[2], minbucket = best\_na\_75$best\_params[3], cp = 0, xval = 0))  
best\_predicciones <- predict(best\_tree\_na\_75, newdata = testeo\_na\_75, type = "class")  
AUC(y\_pred = best\_predicciones, y\_true = testeo\_na\_75$satisfaction)

## [1] 0.7443114

rpart.plot(best\_tree\_na\_75)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



1. <https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction> [↑](#footnote-ref-20)