www.nataliejulian.com
Natalie Julian

Tópico: Árboles de decisión

i) Los algoritmos basados en árboles se consideran unos de los mejores y más utilizados métodos de aprendizaje supervisado; potencian los modelos predictivos con alta precisión, estabilidad y facilidad de interpretación. A diferencia de los modelos lineales, mapean bastante bien las relaciones no lineales. Son adaptables para resolver cualquier tipo de problema (clasificación o regresión). Ésta metodología es utilizada popularmente en todo tipo de problemas de ciencia de datos.

Perfil de clientes en pago de crédito

La base de datos `credit` contiene información de créditos otorgados por una institución financiera. Las variables se muestran a continuación:

- `months_loan_duration`: Cantidad de cuotas mensuales fijadas para el pago del crédito
- `purpose`: Finalidad del crédito
- `amount`: Cantidad de dinero solicitado en el crédito
- `employment_length`: Categoría de antiguedad laboral del cliente que solicitó el crédito
- `personal_status`: Indica sexo y estado marital del cliente que solicitó el crédito
- `residence_history`: Cantidad de cambios de domicilio del cliente que solicitó el crédito
- `age`: Edad del cliente que solicitó el crédito
- `housing`: Indica si el cliente paga arriendo, no paga arriendo o si la vivienda es suya
- `existing_credits`: Cantidad de otros créditos que posee al mismo tiempo de adquirir el crédito

La variable target corresponde a `credit_status`: Indica el estado de pago actual de la cuota del crédito:

- *critical* Si el cliente presenta varias cuotas sin pagar

- *delayed* Si el cliente se encuentra atrasado en el pago de la cuota actual
- *repaid* Si el cliente se encuentra al día con el pago de las cuotas
- *fully repaid* Si el cliente pagó todas las cuotas

Como usted podrá ver, un 29.3% de los créditos se encuentran en estado de pago *crítico*, lo cuál está lejos de ser el panorama ideal. La insitución financiera en cuestión le solicita a usted realizar un análisis para detectar cuáles son los perfiles de clientes que se asocian a los distintos comportamientos de pago, de manera de considerar dicha información al momento de evaluar a los clientes que soliciten un crédito.

Procedimiento

Revisar la estructura de la base de datos

```
str(credit) #Revisar formato de las variables
Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame':
1000 obs. of 10 variables:
 $ months_loan_duration: num 6 48 12 42 24 36 24 36 12 30 ...
 $ credit_status       : chr "critical" "repaid" "critical" "repaid" ...
 $ purpose             : chr "radio/tv" "radio/tv" "education" "furniture" ...
 $ amount              : num 1169 5951 2096 7882 4870 ...
 $ employment_length   : chr "> 7 yrs" "1 - 4 yrs" "4 - 7 yrs" "4 - 7 yrs" ...
 $ personal_status     : chr "single male" "female" "single male" "single male" ...
 $ residence_history   : num 4 2 3 4 4 4 4 2 4 2 ...
 $ age                 : num 67 22 49 45 53 35 53 35 61 28 ...
 $ housing             : chr "own" "own" "own" "for free" ...
 $ existing_credits    : num 2 1 1 1 2 1 1 1 1 2 ...

#variables en formato character:
# - credit_status
# - purpose
# - employment_length
# - personal_status
# - housing

#El resto se encuentran en formato numerical

summary(credit) #No existen observaciones faltantes
 months_loan_duration credit_status        purpose
 Min.   : 4.0         Length:1000        Length:1000
 1st Qu.:12.0         Class :character   Class :character
 Median :18.0         Mode  :character   Mode  :character
 Mean   :20.9
 3rd Qu.:24.0
 Max.   :72.0
     amount       employment_length  personal_status
 Min.   :  250    Length:1000        Length:1000
 1st Qu.: 1366    Class :character   Class :character
 Median : 2320    Mode  :character   Mode  :character
 Mean   : 3271
 3rd Qu.: 3972
 Max.   :18424
 residence_history      age           housing
 Min.   :1.000     Min.   :19.00   Length:1000
 1st Qu.:2.000     1st Qu.:27.00   Class :character
 Median :3.000     Median :33.00   Mode  :character
 Mean   :2.845     Mean   :35.55
 3rd Qu.:4.000     3rd Qu.:42.00
 Max.   :4.000     Max.   :75.00
 existing_credits
 Min.   :1.000
 1st Qu.:1.000
```

```
 Median :1.000
 Mean   :1.407
 3rd Qu.:2.000
 Max.   :4.000

#No existen observaciones faltantes

print(credit)
# A tibble: 1,000 x 10
   months_loan_dur  credit_status purpose amount
             <dbl> <chr>         <chr>    <dbl>
 1               6 critical      radio/      1169
 2              48 repaid        radio/      5951
 3              12 critical      educat      2096
 4              42 repaid        furnit      7882
 5              24 delayed       car ( n     4870
 6              36 repaid        educat      9055
 7              24 repaid        furnit      2835
 8              36 repaid        car ( u     6948
 9              12 repaid        radio/      3059
10              30 critical      car ( n     5234
#     with 990 more rows, and 6 more variables:
#   employment_length <chr>, personal_status <chr>,
#   residence_history <dbl>, age <dbl>, housing <chr>,
#   existing_credits <dbl>

names(credit)
 [1] "months_loan_duration" "credit_status"
 [3] "purpose"              "amount"
 [5] "employment_length"    "personal_status"
 [7] "residence_history"    "age"
 [9] "housing"              "existing_credits"

ncol(credit)   #10 variables
[1] 10

nrow(credit)   #1000 registros
[1] 1000
```

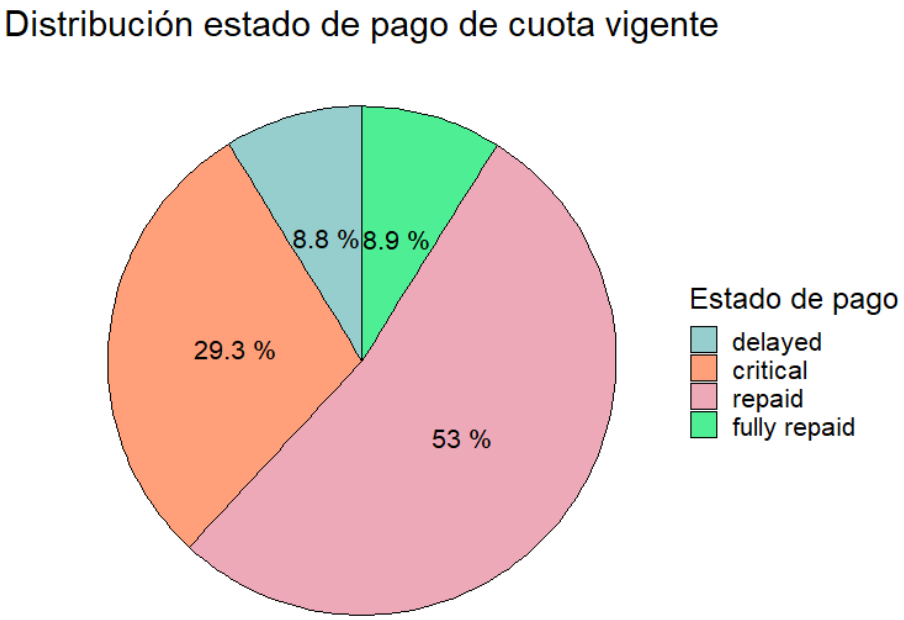
Aspectos importantes:

- Cargar la base de datos y verificar que su importación ha sido la correcta
- Analizar en caso de presentar ID en la base de datos si dicho ID es único.
- Determinar si existen observaciones faltantes o algún dato inconsistente (por ejemplo, en este caso, la variable edad recorre un rango de 19 a 75 años, lo cual resulta factible en términos de solicitar un crédito)
- Observar la cantidad de variables y de registros que se poseen en la base de datos

Observar la variable target

```
 (table(credit$credit_status)/nrow(credit))*100

    critical      delayed fully repaid       repaid
        29.3          8.8          8.9         53.0

df<-data.frame((table(credit$credit_status)/nrow(credit))*100)
df[,3]<-df[,2]*10

colnames(df)<-c("class","prop","n")

#install.packages("dplyr")

library(dplyr)

df$class<-factor(df$class,levels=c("delayed","critical","repaid","fully repaid"))
```

```
df <- df %>%
  arrange(desc(class)) %>%
  mutate(lab.ypos = cumsum(prop) - 0.5*prop)

 df
         class prop   n
1     critical 29.3 293
2      delayed  8.8  88
3 fully repaid  8.9  89
4       repaid 53.0 530


#Grafico de torta

ggplot(df, aes(x = "", y = prop, fill = class)) +
  geom_bar(width = 2, stat = "identity", color = "black") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = lab.ypos, label = paste(prop,"%")),size=5.5, color = "black")+
  scale_fill_manual(values = c("paleturquoise3","lightsalmon","pink2","seagreen2")) +
  theme_void()+labs(fill="Estado de pago",
  title = "Distribucion estado de pago de cuota vigente")+
  theme(plot.title = element_text(hjust=0.5,size=22),
  legend.title = element_text(size=18),legend.text = element_text(size=16))
```



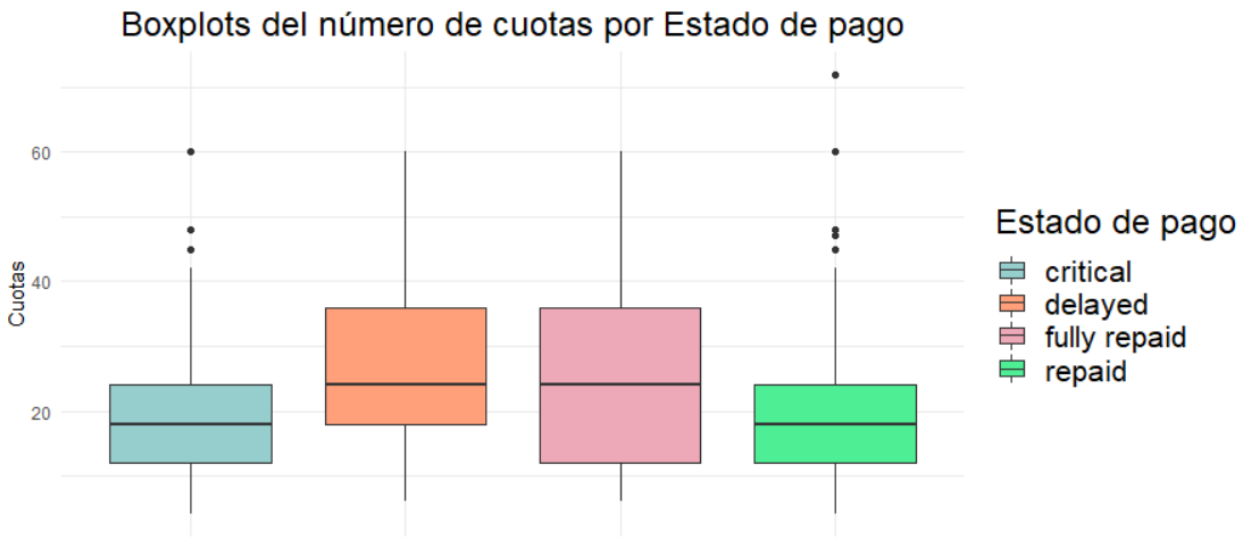
Importante comentar sobre la distribución de la variable target:

- El estado de pago crítico corresponde a un 29.3%, es alto
- El estado de pago atrasado corresponde a un 8.8%
- El estado de pago fully repaid corresponde a un 8.9%
- El estado de pago repaid corresponde a un 53%, la categoría más alta

En este caso, utilizaremos un árbol de clasificación pues la variable que nos interesa particionar en término de las demás variables, es de tipo categórica.

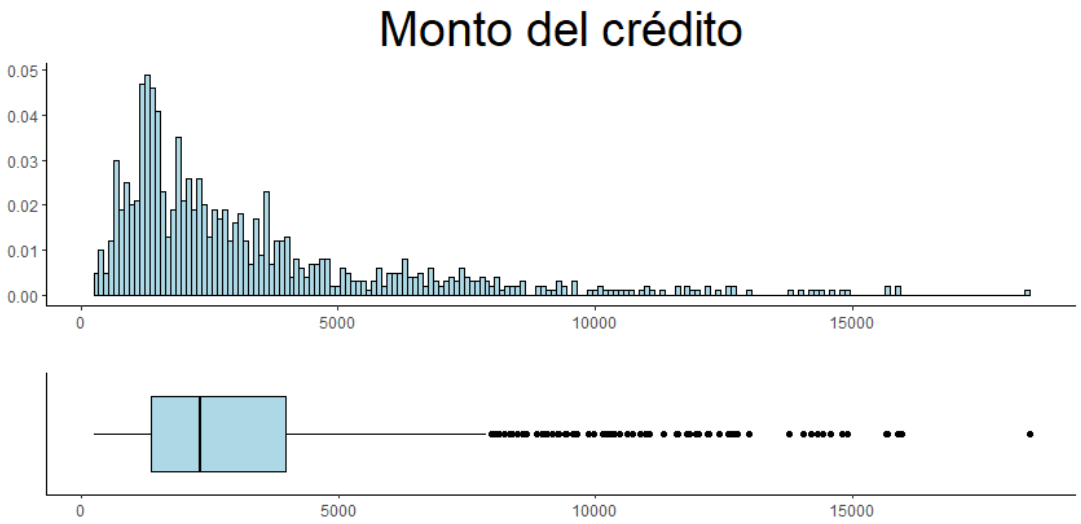
Observar las demás variables

```
#Variable personal status

table(credit$personal_status)
divorced male      female   married male    single male
          50          310            92          548

#Extranamente, no aparece married female o single female,
# la mayor a de las categorias son referentes a male

#Creacion de variable sexo:

sex<-credit$personal_status

sex<-ifelse(sex=="divorced male"|sex=="married male"|sex=="single male","male","female")

table(sex)
female   male
   310    690

credit$sex<-sex

#Variables cuantitativas

#months_loan duration

table(credit$months_loan_duration)

df<-data.frame(credit)

ggplot(aes(y = months_loan_duration, x = credit_status,fill=credit_status), data = df) +
  geom_boxplot()+theme_minimal()+
  scale_fill_manual(values = c("paleturquoise3","lightsalmon","pink2","seagreen2"))+
  theme(axis.text.x = element_blank())+xlab("")+ylab("Cuotas")+
  ggtitle("Boxplots del n mero de cuotas por Estado de pago")+labs(fill="Estado de pago")+
  theme(plot.title = element_text(hjust=0.5,size=18),legend.title = element_text(size=18),
  legend.text = element_text(size=16))
```



Pudiera ser de interés realizar algunos gráficos de determinadas variables interesantes, esta etapa no es estrictamente necesaria, pero siempre es bueno observar gráficas y desarrollar algunas intuiciones.

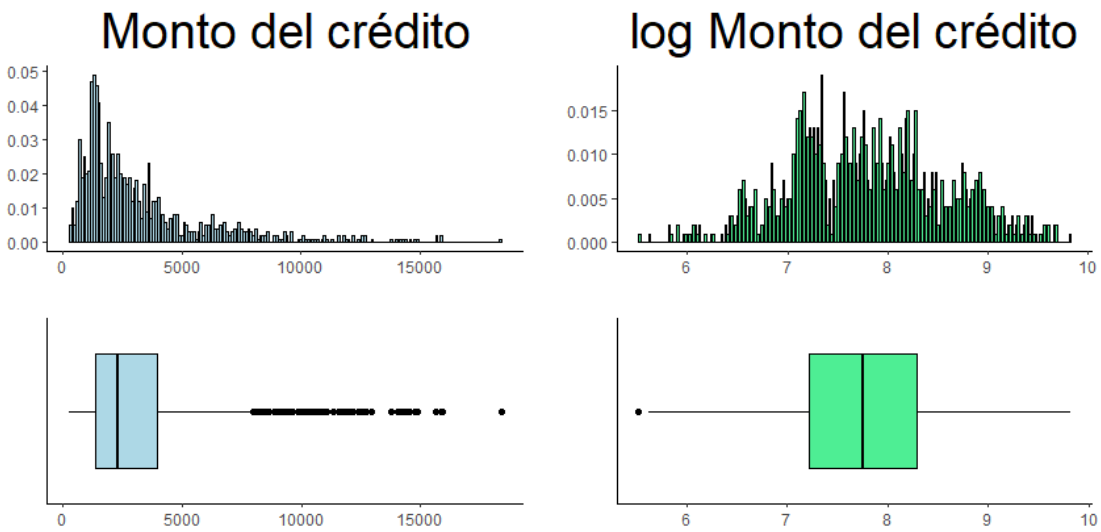
```
plt1 <- credit %>% select(amount) %>%
  ggplot(aes(x="", y = amount)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  coord_flip() +
  theme_classic() + xlab("")+
  ylab("") +
  theme(axis.text.y=element_blank(),
        axis.ticks.y=element_blank())

plt2 <- credit %>% select(amount) %>%
  ggplot() +
  geom_histogram(aes(x = amount, y = (..count..)/sum(..count..)),
                 position = "identity", binwidth = 100,
                 fill = "lightblue", color = "black") +
  ylab("") + xlab("")+ ggtitle("Monto del cr dito")+
  theme_classic()+ theme(plot.title = element_text(hjust=0.5,size=28),
  legend.title = element_text(size=18),legend.text = element_text(size=16))

#install.packages("patchwork)

library(patchwork)
plt2 + plt1 + plot_layout(nrow = 2, heights = c(2, 1))
```



Cuando se tiene una variable con información de montos de dinero, por lo general, pudiera verse cierta asimetría en la distribución de dicha variable. Un tratamiento posible es utilizar el logaritmo de dicha variable.

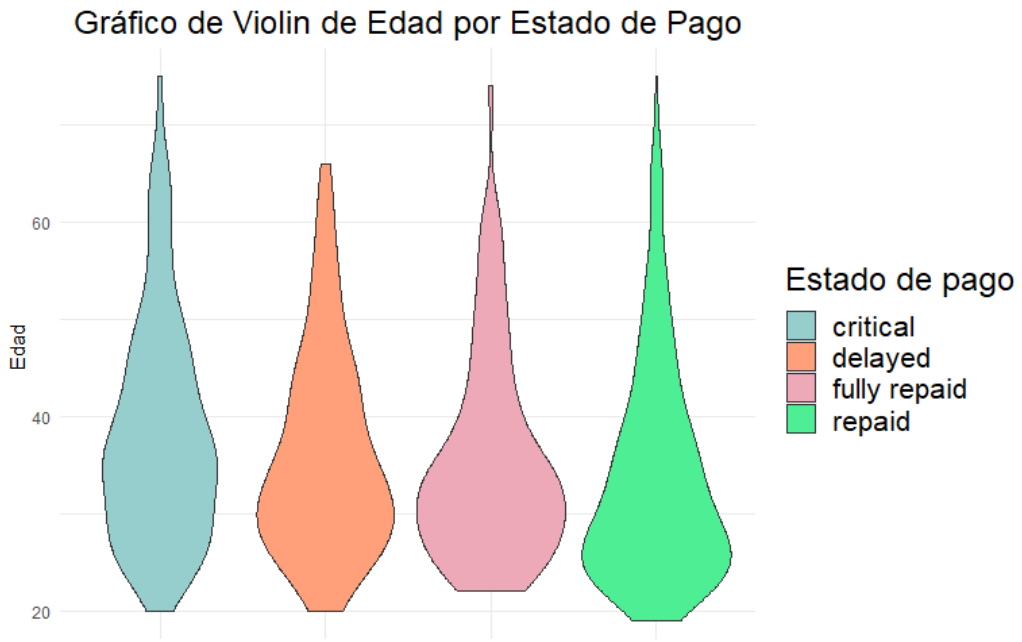
```
#logaritmo de amount

credit$logamount<-log(credit$amount)

plt3 <- credit %>% select(logamount) %>%
  ggplot(aes(x="", y = logamount)) +
  geom_boxplot(fill = "seagreen2", color = "black") +
  coord_flip() +
  theme_classic() + xlab("")+
  ylab("") +
  theme(axis.text.y=element_blank(),
        axis.ticks.y=element_blank())

plt4 <- credit %>% select(logamount) %>%
  ggplot() +
```

```
    geom_histogram(aes(x = logamount, y = (..count..)/sum(..count..)),
                   position = "identity", binwidth = 0.02,
                   fill = "seagreen2", color = "black") +
    ylab("") + xlab("")+ ggtitle("log Monto del cr dito")+
    theme_classic()+ theme(plot.title = element_text(hjust=0.5,size=28),
    legend.title = element_text(size=18),legend.text = element_text(size=16))

plt2 + plt4+plt1 +plt3+plot_layout(nrow = 2, heights = c(2, 2))
```



Para plantear un árbol de clasificación no es estrictamente necesario realizar esta transformación. Pero es mucho más fácil de interpretar el árbol y las conclusiones al llevar la escala de la variable *amount* a un intervalo más acotado.

```
# Age

ggplot(aes(y = age, x = credit_status,fill=credit_status), data = df) +
  geom_violin()+theme_minimal()+
  scale_fill_manual(values = c("paleturquoise3","lightsalmon","pink2","seagreen2"))+
  theme(axis.text.x = element_blank())+xlab("")+ylab("Edad")+
  ggtitle("Grafico de violin de Edad por Estado de Pago")+labs(fill="Estado de pago")+
  theme(plot.title = element_text(hjust=0.5,size=18),legend.title = element_text(size=18),
  legend.text = element_text(size=16))
```



La edad siempre es una variable muy interesante analizar en estos tópicos. Usualmente se puede realizar una categorización de ésta en intervalos. Para efectos de este ejercicio dicha categorización no se realizará, pero es bueno conocer otras alternativas.
El gráfico de la variable edad resulta super interesante. Es posible observar donde se acumulan los casos por estado de pago. Por ejemplo, para el grupo *repaid* se observa que las edades suelen concentrarse en torno al valor 26 aproximadamente. Para el grupo *fully repaid* se observa que, las edades suelen concentrarse en torno a los 30 años aproximadamente, sugiriendo que, quizás, los clientes más jóvenes no suelen pagar por adelantado todo el crédito solicitado.

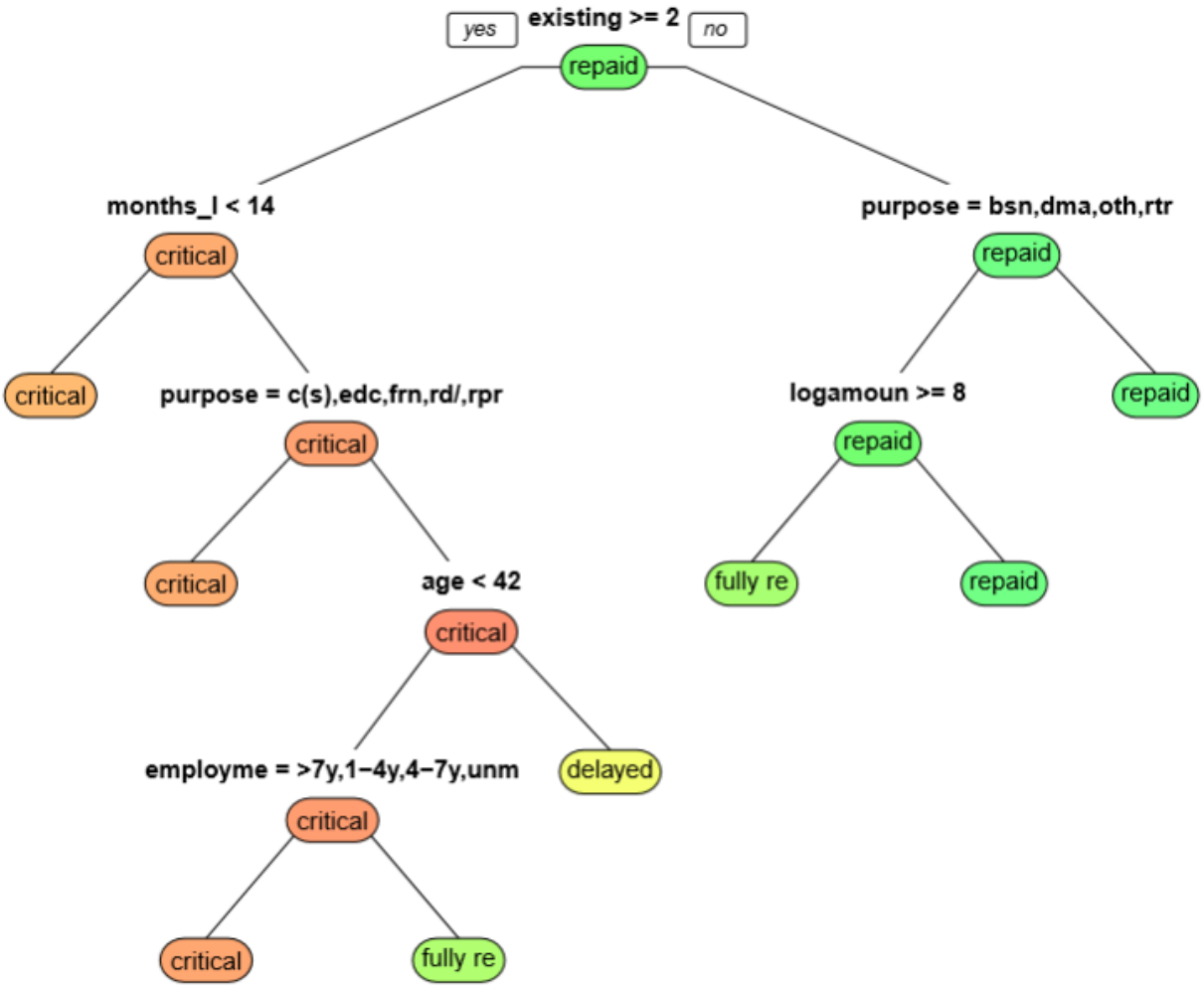
Set de entrenamiento y testeo

```
#Set de entrenamiento

#install.packages("caret")
library(caret)

set.seed(1) #Semilla de aleatoriedad para el split

#split de 60% entrenamiento

index <- createDataPartition(credit$credit_status, p = 0.6, list = FALSE)
Train <- credit[index,]
Test <- credit[-index,]


table(Train$credit_status)

    critical      delayed fully repaid       repaid
         176           53           54          318

table(Test$credit_status)

    critical      delayed fully repaid       repaid
         117           35           35          212
```

En este caso se utilizó una partición o *split* 60% entrenamiento y 40% de testeo. Se podría perfectamente haber utilizado otra división. También algunas veces se incluye un set de validación, usualmente más pequeño que el set de testeo. Usualmente este set se entrega externamente por el jefe/cliente para validar el modelo obtenido y analizado.

Creando el árbol en R

```
# Creando el arbol

#install.packages("rpart")
library(rpart)

model <- rpart(credit_status~., data=Train[,-4],method="class",model=TRUE)


model$variable.importance
     existing_credits               purpose             logamount
           101.275094              9.997370              8.509348
months_loan_duration                   age               housing
             8.265394              6.565362              3.313374
   employment_length     residence_history       personal_status
             3.242636              1.488815              1.279428

#install.packages("rpart.plot")
library("rpart.plot")

prp(model,type=1,box.palette = "RdYlGn",legend.x=NA)
```



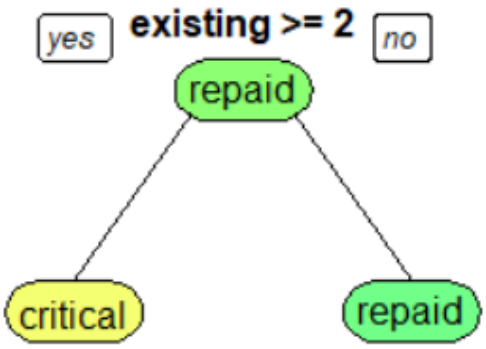
Es posible ver que el árbol de clasificación obtenido resulta ser amplio, utiliza 5 variables y presenta 7 reglas de decisión (nodos de decisión) y además 8 nodos terminales. En este caso no es tan grande el árbol pues sólo se tenían 10 variables en la base de datos, pero imagine tener 50 variables, ¿qué tan extenso podría ser el árbol obtenido? Recuerde que la parsimonia es importante además de que un árbol muy amplio podría caer en el abismo del *sobreajuste*.

¿Podar o no podar?

```
# Costo de complejidad

printcp(model)

Classification tree:
rpart(formula = credit_status ~ ., data = Train[, -4], method = "class",
    model = TRUE)
```

```
Variables actually used in tree construction:
[1] age                    employment_length      existing_credits
[4] logamount              months_loan_duration   purpose

Root node error: 283/601 = 0.47088

n= 601

        CP nsplit rel error  xerror     xstd
1 0.395760      0   1.00000 1.00000 0.043240
2 0.011779      1   0.60424 0.60424 0.039085
3 0.010601      5   0.55477 0.61837 0.039355
4 0.010000      7   0.53357 0.61484 0.039288


(bestcp <- model$cptable[which.min(model$cptable[,"xerror"]),"CP"])
[1] 0.01177856 #cantidad de splits que minimiza el xerror

model_dt.pruned <- prune(model, cp = bestcp) #arbol podado

prp(model_dt.pruned,type=1,box.palette = "RdYlGn",legend.x=NA)
```

El costo de complejidad es de alguna manera, una cantidad que frena el crecimiento excesivo y no beneficioso del árbol. Si el costo de añadir una variable adicional es superior al costo de complejidad establecido, se detiene el crecimiento del árbol y se obtiene el modelo final.
La elección del costo de complejidad va a depender de los criterios. Si se elije aquél costo de complejidad que minimiza el error en la validación cruzada (`xerror`) se obtiene el siguiente árbol:



Comparando desempeños predictivos en el set de testeo

```
pred <- unname(predict(model, Test[,-4], type = "class"))

table(pred==Test$credit_status)

FALSE  TRUE
  136   263

addmargins(table(pred,Test$credit_status),margin=1)

pred            critical delayed fully repaid repaid
  critical            73      18           11      20
```

```
  delayed              5       2            1       1
  fully repaid         3       3            4       7
  repaid              36      12           19     184
  Sum                117      35           35     212

#Tasa de clasificacion correcta:

sum(diag(addmargins(table(pred,Test$credit_status),margin=1)))/nrow(Test)

[1] 0.6591479
```

Al obtener las predicciones del árbol más grande (7 nodos) se observa una tasa de clasificación correcta (en el set de testeo) de un 65.9%, logra clasificar bien gran parte de los casos *repaid* (184 aciertos de 212 casos), mientras que en *critical* el resultado no es tan optimista (73 aciertos de 117), para las categorías *delayed* (2 aciertos de 35) y *fully repaid* (4 aciertos de 35) se tiene un rendimiento bastante deficiente deficiente.

```
pred2 <- unname(predict(model_dt.pruned, Test[,-4], type = "class"))
> table(pred2==Test$credit_status)

FALSE  TRUE
  128   271
> addmargins(table(pred2,Test$credit_status),margin=1)

pred2          critical delayed fully repaid repaid
  critical           80      21            13     21
  delayed             0       0             0      0
  fully repaid        0       0             0      0
  repaid             37      14            22    191
  Sum               117      35            35    212

#Tasa de clasificacion correcta:

sum(diag(addmargins(table(pred2,Test$credit_status),margin=1)))/nrow(Test)
[1] 0.679198
```

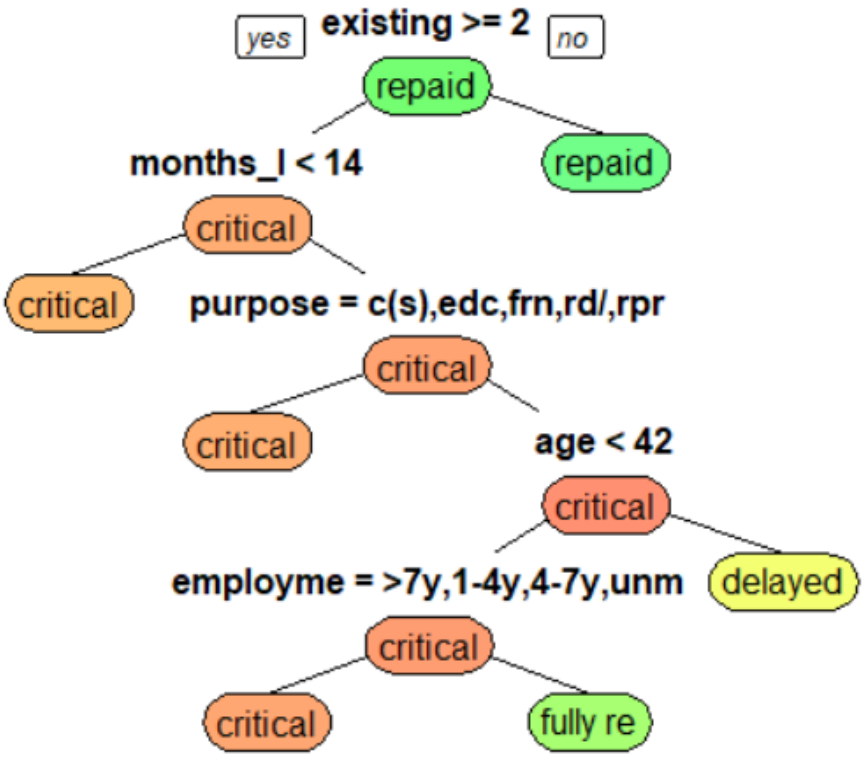
Si utilizaramos el árbol más pequeño (de sólo 1 nodo) se obtiene una tasa de clasificación correcta del 67.9%, superior a la tasa del árbol de 7 nodos. Sin embargo, note que este este modelo no logra predecir correctamente ningún caso *delayed* o *fully repaid*, porque de hecho el árbol sólo entrega dos valores como predicción: *critical* y *repaid*. Si sólo quisieramos predecir estas categorías quizás pudiera ser una buena opción.

Probando el árbol de 5 nodos

¿Qué pasa si no elegimos el árbol más pequeño pero tampoco el más grande? Es decir, ¿cómo se comportaría el árbol intermedio? (árbol de 5 nodos)

```
#5 nodos

printcp(model)
        CP nsplit rel error  xerror     xstd
1 0.395760      0   1.00000 1.00000 0.043240
2 0.011779      1   0.60424 0.60424 0.039085
3 0.010601      5   0.55477 0.61837 0.039355
4 0.010000      7   0.53357 0.61484 0.039288

model5nodos<-prune(model, cp = 0.010601)

model5nodos$variable.importance
    existing_credits                  age              purpose months_loan_duration
         101.2750941            6.5653625            5.4431725            4.3771993
  employment_length           logamount              housing     residence_history
```

```
        3.2426364          2.8537926          1.5460125          0.7818710
personal_status
        0.5724839

prp(model5nodos,type=1,box.palette = "RdYlGn",legend.x=NA)
```




```
#Desempeno arbol podado 5 nodos

pred3 <- unname(predict(model5nodos, Test[,-4], type = "class"))

table(pred3==Test$credit_status)

addmargins(table(pred3,Test$credit_status),margin=1)

pred3           critical delayed fully repaid repaid
  critical            73      18           11     20
  delayed              5       2            1      1
  fully repaid         2       1            1      0
  repaid              37      14           22    191
  Sum                117      35           35    212

#Tasa de clasificacion correcta:

sum(diag(addmargins(table(pred3,Test$credit_status),margin=1)))/nrow(Test)
[1] 0.6691729
```

La tasa de clasificación correcta de este modelo es de 66.9%, superior a la del modelo de 7 nodos. Aunque sí logra entregar predicciones para las categorías *delayed* y *fully repaid*, nuevamente, su desempeño es bastante deficiente. Aunque para detectar los casos *repaid* es bastante bueno. La selección de un árbol u otro dependerá de a qué tiene mayor valor agregado para el cliente. Clasificar mejor los casos *repaid*, *critical* u otro.

Utilizando un árbol de decisión

En base al árbol elegido, ¿cuál sería el comportamiento de pago que presentaría un cliente con la siguiente información?:

- `months_loan_duration=24`
- `purpose=education`
- `amount=10000`
- `employment_length=unemployed`
- `personal_status=single male`
- `residence_history=4`
- `age=53`
- `housing=for free`
- `existing_credits=3`

Sólo basta mirar el árbol y se esperaría que el comportamiento de pago del cliente sería *critical*. Cualquiera de los 3 árboles anteriores entrega el mismo resultado.