Universitat Oberta de Catalunya

Estudios de Informática, Multimedia y Telecomunicaciones

Minería de datos: PEC3 - Clasificación con árboles de decisión

Eduardo Mora González

Noviembre 2021

- 1 Recursos básicos
- 2 Ejemplo ilustrativo
  - 2.1 Análisis inicial
  - 2.2 Preparación de los datos para el modelo
  - 2.3 Creación del modelo, calidad del modelo y extracción de reglas
  - 2.4 Validación del modelo con los datos reservados
  - 2.5 Prueba con una variación u otro enfoque algorítmico
- 3 Enunciado del ejercicio
  - 3.1 Análisis descriptivo y de correlaciones
  - 3.2 Primer árbol de decisión
  - 3.3 Explicación de las reglas obtenidas
  - 3.4 Análisis de la bondad de ajuste sobre el conjunto de test y matriz de confusión
  - 3.5 Modelos complementarios
  - 3.6 Conclusiones obtenidas
- 4 Rúbrica

1 Recursos básicos

Material didáctico de: Modelos Supervisados y Evaluación de Modelos.

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 (https://cran.r-project.org/web/packages/C50/index.html)
- Fichero de “German Credit”: credit.csv: https://www.kaggle.com/shravan3273/credit-approval (https://www.kaggle.com/shravan3273/credit-approval)

2 Ejemplo ilustrativo

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. Primero y a modo de ejemplo sencillo 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. Se mostrará un ejemplo sencillo de solución con estos datos pero los alumnos deberéis responder a las preguntas de la rúbrica para otro conjunto: German Credit. Para este conjunto, tomaréis como referencia la variable "default" que indica el impago de créditos.

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 sobrevivia 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?

A continuación, se plantean los puntos a realizar en la PEC 3 y, tomando como ejemplo el conjunto de datos de Titanic, se obtendrán, a modo de ejemplo, algunos resultados que pretender servir a modo de inspiración para los estudiantes. Los estudiantes deberán utilizar el conjunto de datos de "German Credit Data" que se pueden conseguir en este enlace: https://www.kaggle.com/shravan3273/credit-approval (https://www.kaggle.com/shravan3273/credit-approval)

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)
```

2.1 Análisis inicial

Empezaremos haciendo un breve análisis de los datos ya que nos interesa tener una idea general de los datos que disponemos.

2.1.1 Exploración de la base de datos

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
```

Como parte de la preparación de los datos, miraremos si hay valores missing.

```
missing <- data[is.na(data),]
dim(missing)
```

```
## [1] 0 4
```

Observamos fácilmente que no hay valores missing y, por tanto, no deberemos preparar los datos en este sentido. En caso de haberlos, habría que tomar decisiones para tratar los datos adecuadamente.

Disponemos por tanto de un data frame formado por cuatro variables categóricas sin valores nulos.

2.1.2 Visualización

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

```
## Loading required package: ggpubr
```

```
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)
}
```

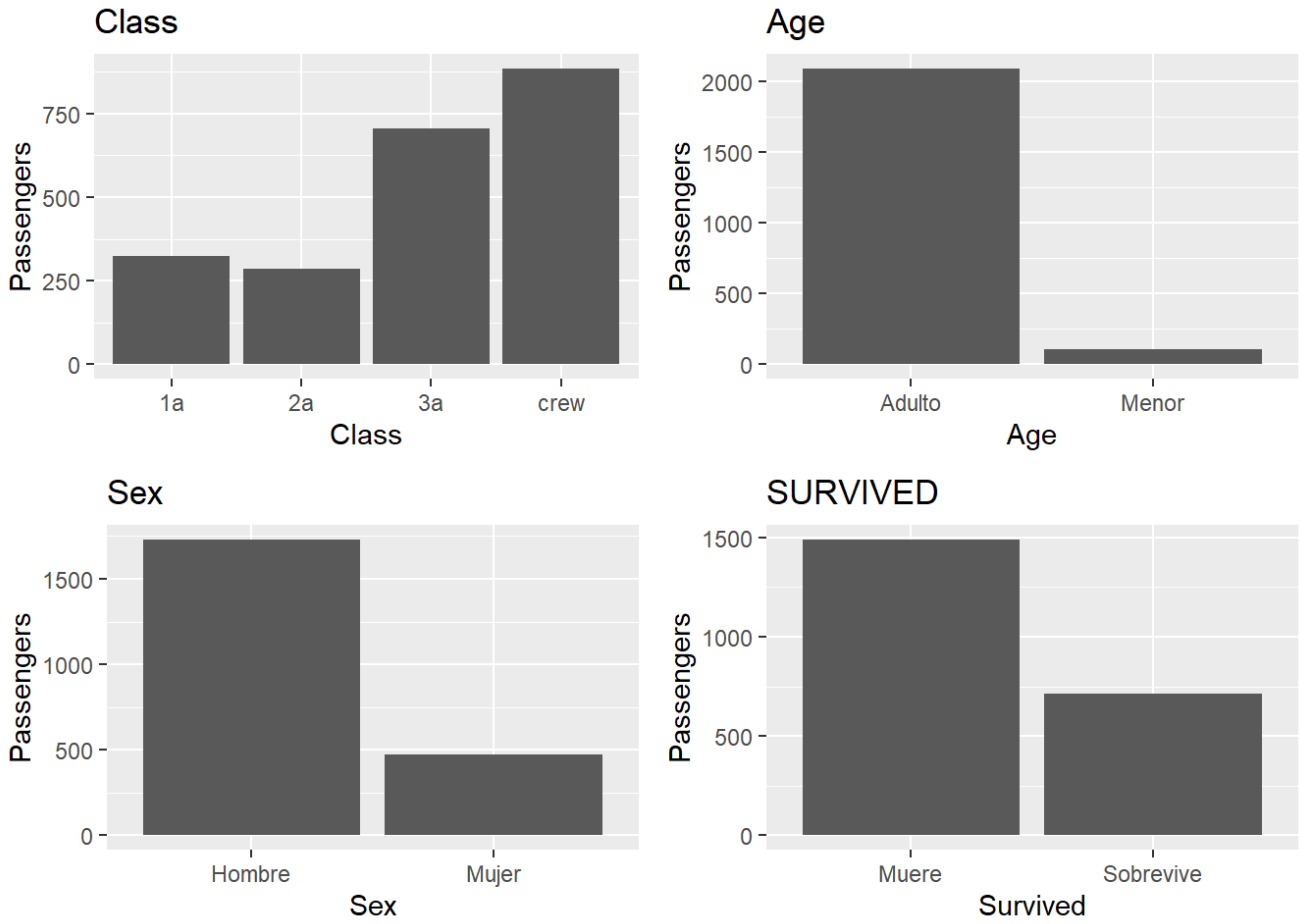
```
## Loading required package: gridExtra
```

```
if(!require(C50)){
    install.packages('C50', repos='http://cran.us.r-project.org')
    library(C50)
}
```

```
## Loading required package: C50
```

Siempre es importante analizar los datos que tenemos ya que las conclusiones dependerán de las características de la muestra.

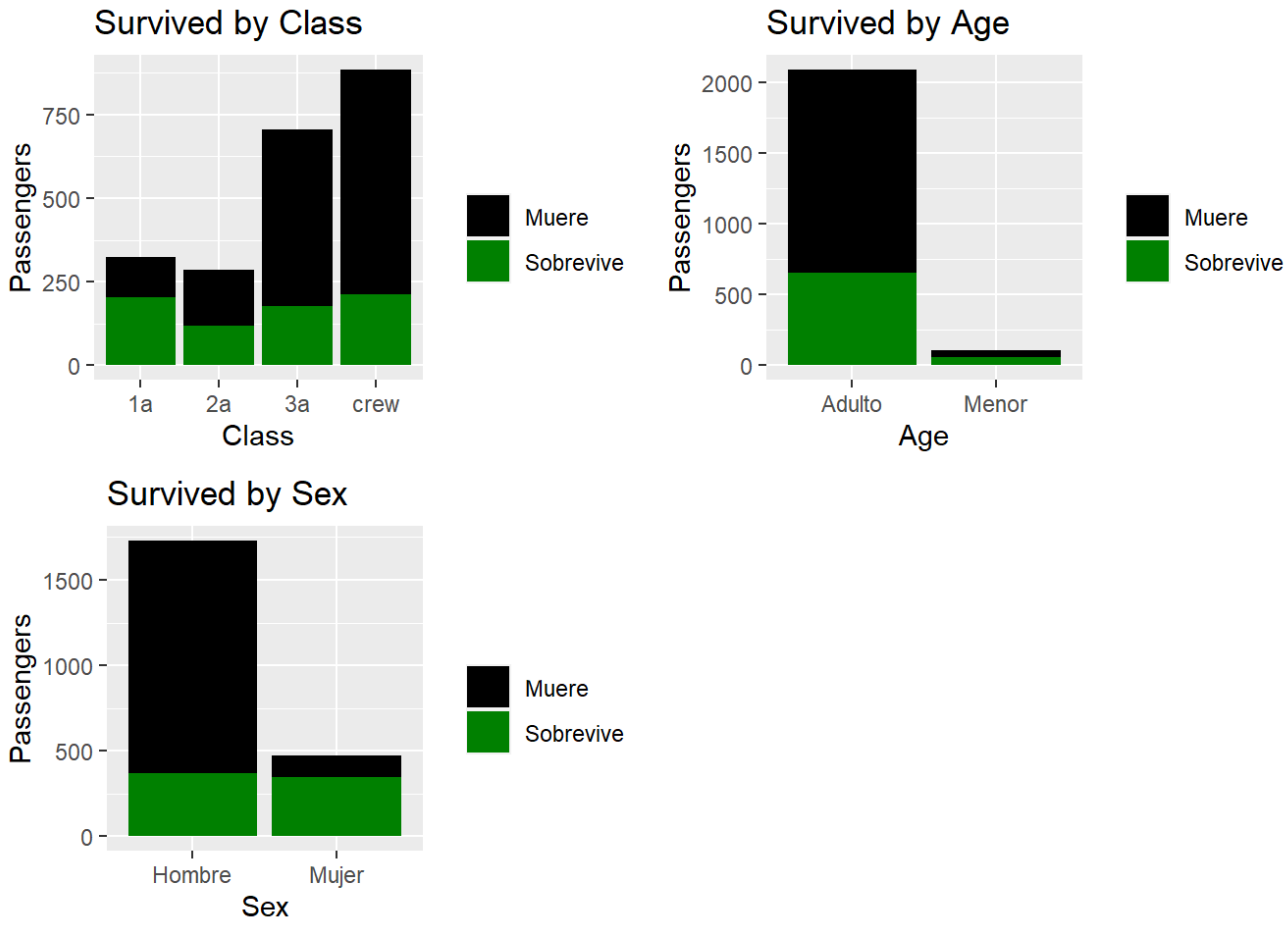
```
grid.newpage()
plotbyClass<-ggplot(data,aes(CLASS))+geom_bar() +labs(x="Class", y="Passengers")+ guides
(fill=guide_legend(title=""))+ scale_fill_manual(values=c("blue","#008000"))+ggtitle("Cl
ass")
plotbyAge<-ggplot(data,aes(AGE))+geom_bar() +labs(x="Age", y="Passengers")+ guides(fill=
guide_legend(title=""))+ scale_fill_manual(values=c("blue","#008000"))+ggtitle("Age")
plotbySex<-ggplot(data,aes(SEX))+geom_bar() +labs(x="Sex", y="Passengers")+ guides(fill=
guide_legend(title=""))+ scale_fill_manual(values=c("blue","#008000"))+ggtitle("Sex")
plotbySurvived<-ggplot(data,aes(SURVIVED))+geom_bar() +labs(x="Survived", y="Passengers"
)+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("blue","#008000"))+gg
title("SURVIVED")
grid.arrange(plotbyClass,plotbyAge,plotbySex,plotbySurvived,ncol=2)
```



Claramente vemos cómo es la muestra analizando la distribución de las variables disponibles. De cara a los informes, es mucho más interesante esta información que la obtenida en summary, que se puede usar para complementar.

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="Passen
gers")+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("black","#00800
0"))+ggtitle("Survived by Class")
plotbyAge<-ggplot(data,aes(AGE,fill=SURVIVED))+geom_bar() +labs(x="Age", y="Passengers")
+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("black","#008000"))+gg
title("Survived by Age")
plotbySex<-ggplot(data,aes(SEX,fill=SURVIVED))+geom_bar() +labs(x="Sex", y="Passengers")
+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("black","#008000"))+gg
title("Survived by Sex")
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
```

2.1.3 Test estadísticos de significancia

Los resultados anteriores muestran los datos de forma descriptiva, podemos añadir algún test estadístico para validar el grado de significancia de la relación. La librería “DescTools” nos permite instalarlo fácilmente.

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

```
## Loading required package: DescTools
```

```
Phi(tabla_SST)
```

```
## [1] 0.4556048
```

```
CramerV(tabla_SST)
```

```
## [1] 0.4556048
```

```
Phi(tabla_SAT)
```

```
## [1] 0.09757511
```

```
CramerV(tabla_SAT)
```

```
## [1] 0.09757511
```

```
Phi(tabla_SCT)
```

```
## [1] 0.2941201
```

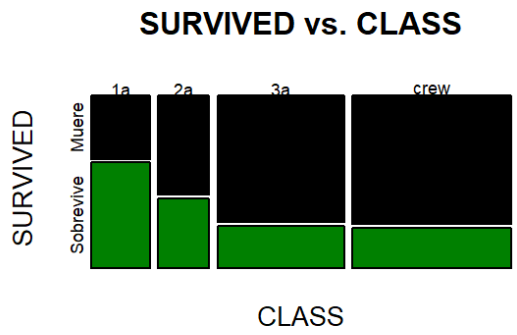
```
CramerV(tabla_SCT)
```

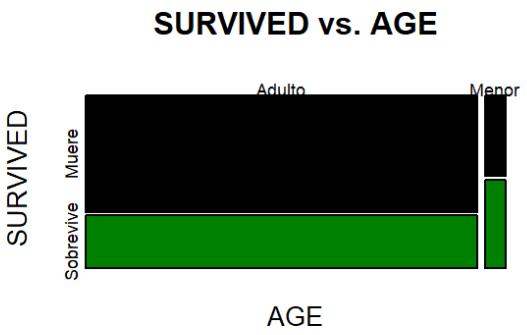
```
## [1] 0.2941201
```

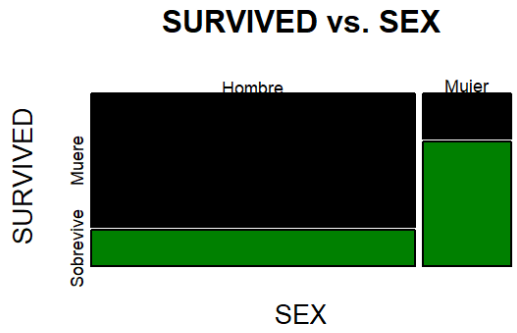
Valores de la V de Cramér (https://en.wikipedia.org/wiki/Cramér%27s_V (https://en.wikipedia.org/wiki/Cramér%27s_V)) y Phi (https://en.wikipedia.org/wiki/Phi_coefficient (https://en.wikipedia.org/wiki/Phi_coefficient)) entre 0.1 y 0.3 nos indican que la asociación estadística es baja, y entre 0.3 y 0.5 se puede considerar una asociación media. Finalmente, si los valores fueran superiores a 0.5 (no es el caso), la asociación estadística entre las variables sería alta. Como se puede apreciar, los valores de Phi y V coinciden. Esto ocurre en el contexto de analizar tablas de contingencia 2x2.

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 interesa “desordenarlos”. Guardaremos los datos con el nuevo nombre como “data_random”.

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

2.2 Preparación de los datos para el modelo

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. De esta forma, tendremos un conjunto de datos para el entrenamiento y uno para la validación

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

De forma dinámica podemos definir una forma de separar los datos en función de un parámetro, en este caso del “split_prop”. Definimos un parámetro que controla el split de forma dinámica en el test.

```
split_prop <- 3
max_split<-floor(nrow(X)/split_prop)
tr_limit <- nrow(X)-max_split
ts_limit <- nrow(X)-max_split+1

trainX <- X[1:tr_limit,]
trainy <- y[1:tr_limit]
testX <- X[ts_limit+1:nrow(X),]
testy <- y[ts_limit+1:nrow(X)]
```

En la segunda opción podemos crear directamente un rango utilizando el mismo parámetro anterior.

```
split_prop <- 3
indexes = sample(1:nrow(data), size=floor(((split_prop-1)/split_prop)*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. En este caso, verificaremos que la proporción del supervivientes es más o menos constante en los dos conjuntos:

```
summary(trainX);
```

```
##     CLASS               AGE                SEX
##  Length:1467        Length:1467        Length:1467
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
summary(trainy)
```

```
##    Length     Class      Mode
##      1467 character character
```

```
summary(testX)
```

```
##     CLASS               AGE                SEX
##  Length:734         Length:734         Length:734
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
summary(testy)
```

```
##    Length     Class      Mode
##       734 character character
```

Verificamos fácilmente que no hay diferencias graves que puedan sesgar las conclusiones.

2.3 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]      Sat Dec 11 10:50:00 2021
## -------------------------------
##
## 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 317 de los 1467 casos dados, una tasa de error del 21.6%.

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: 78,1%

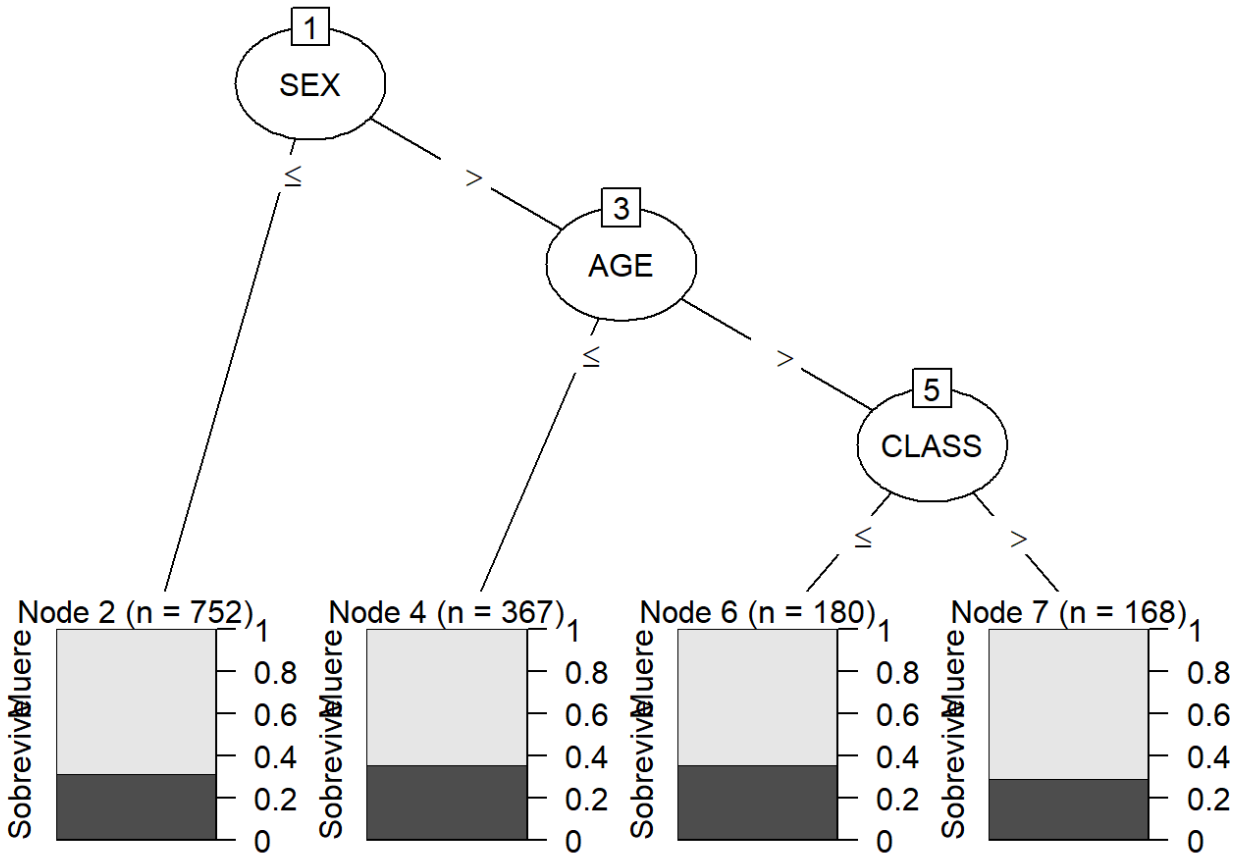
CLASS “1ª”, “2ª” y AGE = “Menor” → Sobrevive. Validez: 95,5%

SEX = “Mujer” → Sobrevive. Validez: 74,7%

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)
```



2.4 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) / l
ength(predicted_model)))
```

```
## [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_cor
rect))
```

```
## [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
```

```
## Registered S3 method overwritten by 'gdata':
##   method         from
##   reorder.factor DescTools
```

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

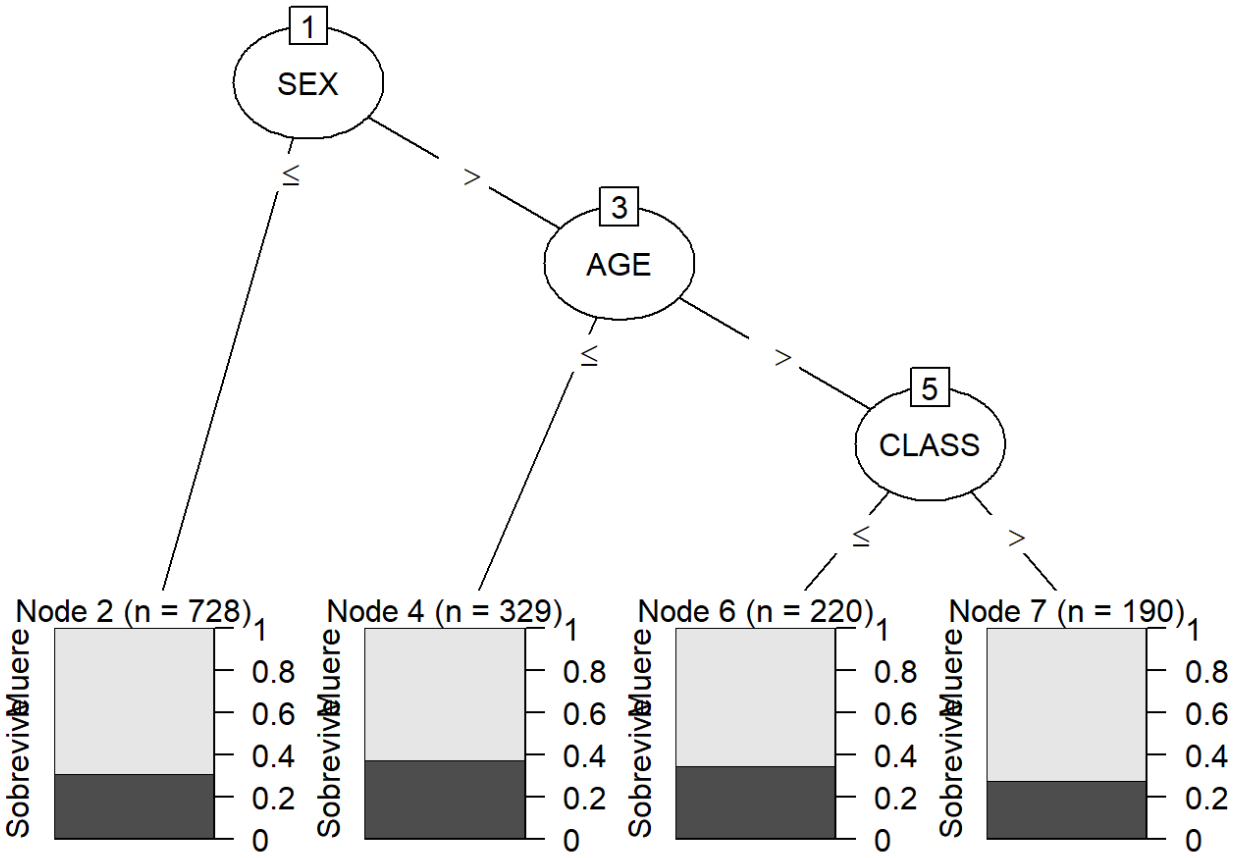
```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  734
##
##
##              | Prediction
##      Reality |     Muere | Sobrevive | Row Total |
## -------------|-----------|-----------|-----------|
##        Muere |       450 |        51 |       501 |
##              |     0.613 |     0.069 |           |
## -------------|-----------|-----------|-----------|
##    Sobrevive |       109 |       124 |       233 |
##              |     0.149 |     0.169 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       559 |       175 |       734 |
## -------------|-----------|-----------|-----------|
##
##
```

2.5 Prueba con una variación u otro enfoque algorítmico

En este apartado buscaremos probar con las variaciones que nos ofrece el paquete C5.0 para analizar cómo afectan a la creación de los árboles generados. Existen muchas posibles variaciones con otras funciones que podéis investigar. La idea es seguir con el enfoque de árboles de decisión explorando posibles opciones. Una vez tengamos un método alternativo, debemos analizar cómo se modifica el árbol y cómo afecta a la capacidad predictiva en el conjunto de test.

A continuación, utilizamos otro enfoque para comparar los resultados: incorpora como novedad “adaptative boosting”, basado en el trabajo Rob Schapire and Yoav Freund (1999). La idea de esta técnica es generar varios clasificadores, con sus correspondientes arboles de decisión y su ser de reglas. Cuando un nuevo caso va a ser clasificado, cada clasificador vota cual es la clase predicha. Los votos son sumados y determina la clase final.

```
modelo2 <- C50::C5.0(trainX, trainy, trials = 10)
plot(modelo2)
```



En este caso, dada la simplicidad del conjunto de ejemplo, no se aprecian diferencias, pero aparecerán en datos de mayor complejidad y modificando el parámetro “trials” se puede intentar mejorar los resultados.

Vemos a continuación cómo son las predicciones del nuevo árbol:

```
predicted_model2 <- predict( modelo2, testX, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model2 == testy) /
 length(predicted_model2)))
```

```
## [1] "La precisión del árbol es: 79.5640 %"
```

Observamos como se modifica levemente la precisión del modelo a mejor.

```
mat_conf<-table(testy,Predicted=predicted_model2)
mat_conf
```

```
##            Predicted
## testy       Muere Sobrevive
##   Muere       491        10
##   Sobrevive   140        93
```

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_cor
rect))
```

```
## [1] "El % de registros correctamente clasificados es: 79.5640 %"
```

El algoritmo C5.0 incorpora algunas opciones para ver la importancia de las variables (ver documentación para los detalles entre los dos métodos):

```
importancia_usage <- C50::C5imp(modelo2, metric = "usage")
importancia_splits <- C50::C5imp(modelo2, metric = "splits")
importancia_usage
```

```
##       Overall
## CLASS  100.00
## SEX    100.00
## AGE     91.55
```

```
importancia_splits
```

```
##        Overall
## CLASS 36.84211
## SEX   36.84211
## AGE   26.31579
```

Curiosamente y aunque el conjunto de datos es muy sencillo, se aprecian diferencias en los métodos de importancia de las variables. Se recomienda en vuestro ejercicio mejorar la visualización de los resultados con la función ggplo2 o similar.

3 Enunciado del ejercicio

Para el conjunto de datos German Credit, los alumnos deben completar aquí la solución a la PEC3 que consiste de los siguientes apartados. Notad que se detalla el contenido necesario para cada apartado en la Sección 4 (Rúbrica).

Carga de los datos:

```
datos<-read.csv(file = "./credit.csv",header=T,sep=",")
```

Cargar librerias:

```
install.packages("fBasics")
```

```
## package 'fBasics' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\eduar\AppData\Local\Temp\RtmpmI0AcG\downloaded_packages
```

```
library(fBasics) #summary statistics
library(plotly) #data visulization
library(dplyr) #count function
library(C50) #C50 Decision Tree algorithm
library(gmodels) #CrossTable()
library(caret) #Confusion Matrix
library(rmarkdown)
```

3.1 Análisis descriptivo y de correlaciones

Lo primero que debemos hacer es analizar las variables que componen el fichero, para ello obtendremos su dimensión, el número de filas, su estructura y sus estadísticas básicas.

```
#Dimensión
dim(datos)
```

```
## [1] 1000   21
```

```
#Filas
filas=dim(datos)[1]
```

Podemos ver que tenemos 1000 entradas distintas y tenemos 21 características que definen el conjunto de datos

```
#Estructura
str(datos)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ checking_balance    : chr  "< 0 DM" "1 - 200 DM" "unknown" "< 0 DM" ...
##  $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credit_history      : chr  "critical" "repaid" "critical" "repaid" ...
##  $ purpose             : chr  "radio/tv" "radio/tv" "education" "furniture" ...
##  $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ savings_balance     : chr  "unknown" "< 100 DM" "< 100 DM" "< 100 DM" ...
##  $ employment_length   : chr  "> 7 yrs" "1 - 4 yrs" "4 - 7 yrs" "4 - 7 yrs" ...
##  $ installment_rate    : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ personal_status     : chr  "single male" "female" "single male" "single male" ...
##  $ other_debtors       : chr  "none" "none" "none" "guarantor" ...
##  $ residence_history   : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ property            : chr  "real estate" "real estate" "real estate" "building soc
iety savings" ...
##  $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ installment_plan    : chr  "none" "none" "none" "none" ...
##  $ housing             : chr  "own" "own" "own" "for free" ...
##  $ existing_credits    : int  2 1 1 1 2 1 1 1 1 2 ...
##  $ default             : int  1 2 1 1 2 1 1 1 1 2 ...
##  $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ telephone           : chr  "yes" "none" "none" "none" ...
##  $ foreign_worker      : chr  "yes" "yes" "yes" "yes" ...
##  $ job                 : chr  "skilled employee" "skilled employee" "unskilled reside
nt" "skilled employee" ...
```

Viendo la estructura vemos que tenemos 7 variables de tipo numérico y el resto son de tipo categórico.

```
#Estadísticas básicas
summary(datos)
```

```
##  checking_balance   months_loan_duration credit_history       purpose             am
ount      savings_balance    employment_length  installment_rate personal_status
##  Length:1000        Min.   : 4.0         Length:1000        Length:1000        Min.
:  250   Length:1000        Length:1000        Min.   :1.000    Length:1000
##  Class :character   1st Qu.:12.0         Class :character   Class :character   1st Q
u.: 1366   Class :character   Class :character   1st Qu.:2.000    Class :character
##  Mode  :character   Median :18.0         Mode  :character   Mode  :character   Median
: 2320   Mode  :character   Mode  :character   Median :3.000    Mode  :character
##                     Mean   :20.9                                                 Mean
: 3271                                       Mean   :2.973
##                     3rd Qu.:24.0                                                3rd Q
u.: 3972                                       3rd Qu.:4.000
##                     Max.   :72.0                                                Max.
:18424                                       Max.   :4.000
##  other_debtors      residence_history   property              age        installment_
plan     housing          existing_credits    default      dependents
##  Length:1000        Min.   :1.000     Length:1000        Min.   :19.00   Length:1000
Length:1000        Min.   :1.000    Min.   :1.0   Min.   :1.000
##  Class :character   1st Qu.:2.000     Class :character   1st Qu.:27.00   Class :chara
cter   Class :character   1st Qu.:1.000    1st Qu.:1.0   1st Qu.:1.000
##  Mode  :character   Median :3.000     Mode  :character   Median :33.00   Mode  :chara
cter   Mode  :character   Median :1.000    Median :1.0   Median :1.000
##                     Mean   :2.845                         Mean   :35.55
Mean   :1.407    Mean   :1.3   Mean   :1.155
##                     3rd Qu.:4.000                         3rd Qu.:42.00
3rd Qu.:2.000    3rd Qu.:2.0   3rd Qu.:1.000
##                     Max.   :4.000                         Max.   :75.00
Max.   :4.000    Max.   :2.0   Max.   :2.000
##   telephone         foreign_worker         job
##  Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

Y gracias a las estadísticas básicas, podemos ver (sobre todo en las variables de tipo numérica) entre que rangos nos encontramos:

–> *checking_balance:* Estado de la cuenta corriente existente. Los valores que esta variable puede tomar son: <0 DM, 1 - 200 DM, unknown o > 200 DM.

–> *months_loan_duration:* Duración del crédito en meses.

–> *credit_history:* Historial de crédito. Los valores que esta variable puede tomar son: repaid, fully repaid this bank, fully repaid, delayed y critical.

–> *purpose:* Propósito. Los valores que esta variable puede tomar son: automóvil (nuevo), automóvil (usado), mobiliario / equipo, radio / televisión, electrodomésticos, reparaciones, educación, reciclaje, empresa u otros.

–> *amount:* Importe del crédito.

–> *savings_balance:* Cantidad en la cuenta de ahorro. Los valores que esta variable puede tomar son: < 100 DM, 101 - 500 DM, 501 - 1000 DM o unknown.

–> *employment_length:* Duración del empleo actual: > 7 yrs, 0 - 1 yrs, 1 - 4 yrs, 4 - 7 yrs y unemployed.

–> *installment_rate:* Tasa de cuotas en porcentaje de la renta disponible.

–> *personal_status:* Estado personal y sexo. Los valores que esta variable puede tomar son: hombre: divorciado / separado, mujer: divorciado / separado / casado, hombre: soltero, hombre: casado / viudo y mujer: soltero.

–> *other_debtors:* Otros deudores. Los valores que esta variable puede tomar son: none, co-applicant y guarantor.

–> *residence_history:* Residencia actual desde.

–> *property:* Propiedades o seguros. Los valores que esta variable puede tomar son: building society savings, other, real estate y unknown/none.

–> *age:* Edad en años.

–> *installment_plan:* Otros planes de pago. Los valores que esta variable puede tomar son: bank, none y stores.

–> *housing:* Tipo de vivienda. Los valores que esta variable puede tomar son: for free, own y rent.

–> *existing_credits:* Número de créditos existentes en este banco.

–> *default:* ¿Ha pagado el crédito?. Los valores que esta variable puede tomar son: Yes o No.

–> *dependents:* Número de personas que están obligadas a proporcionar mantenimiento para el crédito.

–> *telephone:* Teléfono. Los valores que esta variable puede tomar son: Yes o No.

–> *foreign_worker:* trabajador en el extranjero. Los valores que esta variable puede tomar son: Yes o No.

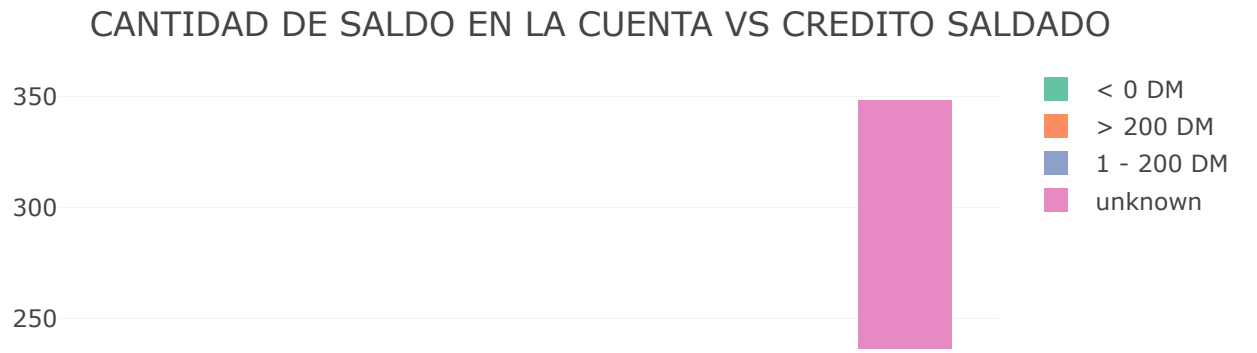
–> *job:* Tipo de trabajo mangement. Los valores que esta variable puede tomar son: self-employed, skilled employee, unemployed non-resident y unskilled resident.

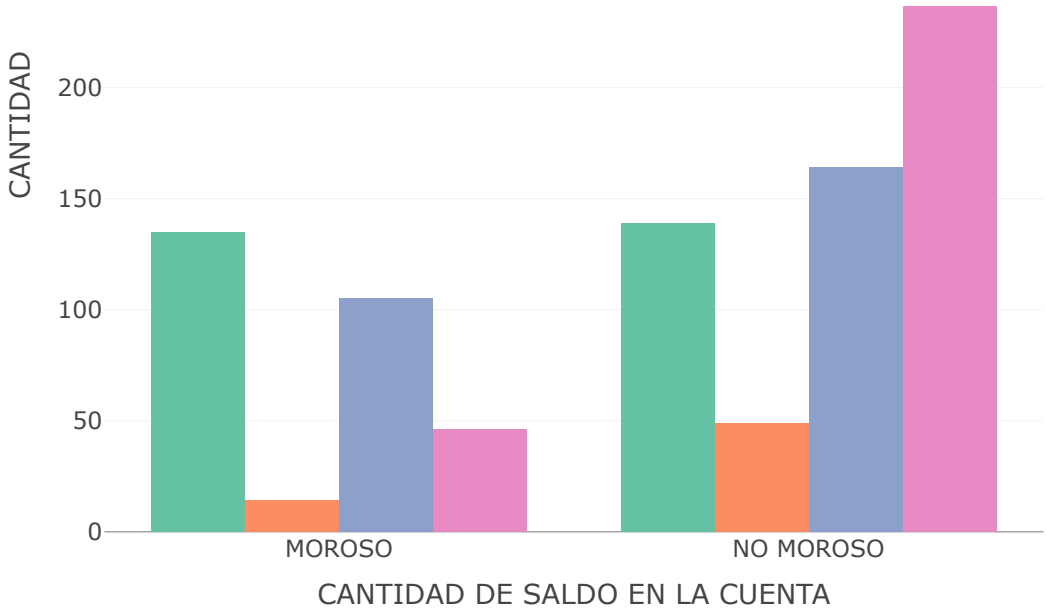
Lo primero que vamos a hacer es normalizar el campo default:

```
datos$default[datos$default == 1] <- "NO MOROSO"
datos$default[datos$default == 2] <- "MOROSO"
datos$default <- as.factor(datos$default)
```

A continuación, se crearán algunas tablas y gráficas útiles que ofrecen estadísticas resumidas de las características del préstamo, como cuenta corriente, ahorros, duración y monto en función del incumplimiento (checking, savings, duration, amount, and default).

```
#Gráfica
datos %>% count(default, checking_balance) %>% plot_ly(x = ~default, y = ~n, color = ~ch
ecking_balance, type = "bar") %>%  layout(title = "CANTIDAD DE SALDO EN LA CUENTA VS CRE
DITO SALDADO", xaxis = list(title = "CANTIDAD DE SALDO EN LA CUENTA"), yaxis = list(titl
e = "CANTIDAD"))
```



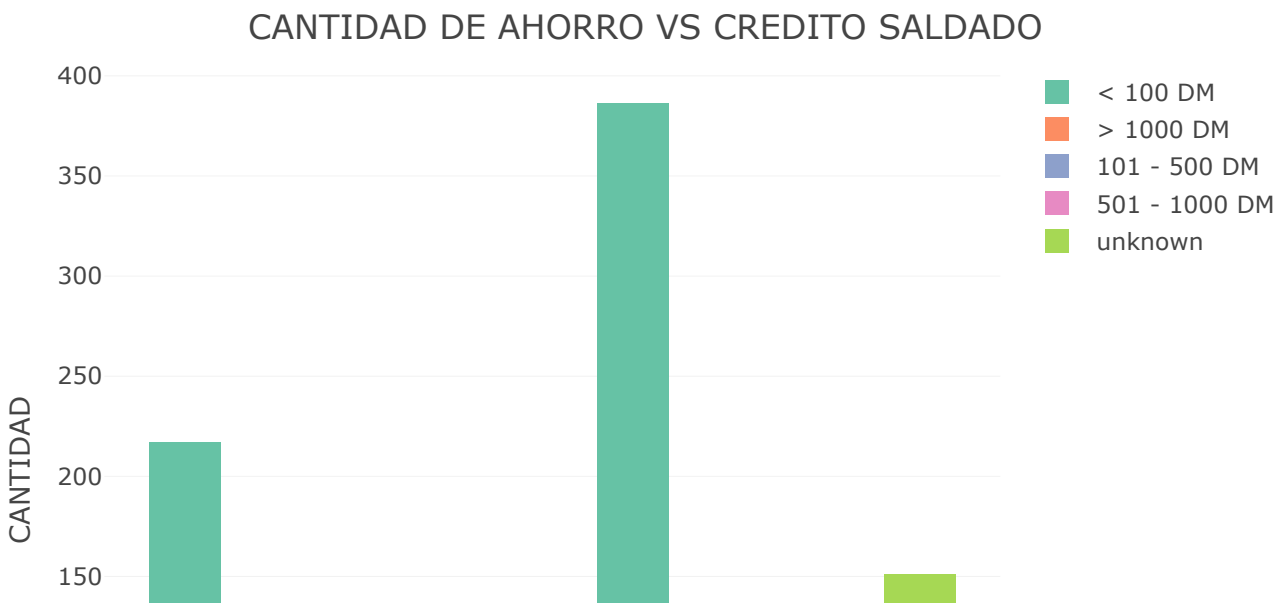


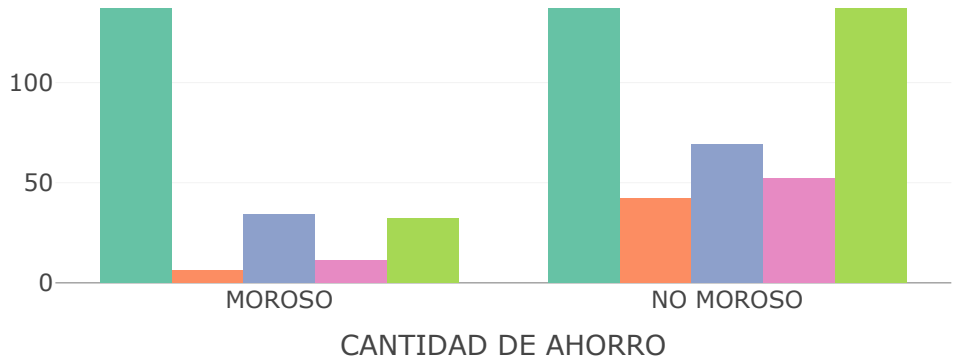

```
#Estadisticas
knitr::kable(data.frame(table(datos$checking_balance)), Caption = "Checking Balance", co
l.names = c("Cuenta corriente", "Frecuencia"))
```

Cuenta corriente	Frecuencia
< 0 DM	274
> 200 DM	63
1 - 200 DM	269
unknown	394

Como se puede observar, las cuentas desconocidas son la inmensa mayoría, pero las cuentas de entre 1 - 200 DM son las que suelen pagar más los créditos.

```
#Gráfica
datos %>% count(default, savings_balance) %>% plot_ly(x = ~default, y = ~n, color = ~sav
ings_balance, type = "bar") %>%  layout(title = "CANTIDAD DE AHORRO VS CREDITO SALDADO",
xaxis = list(title = "CANTIDAD DE AHORRO"), yaxis = list(title = "CANTIDAD"))
```





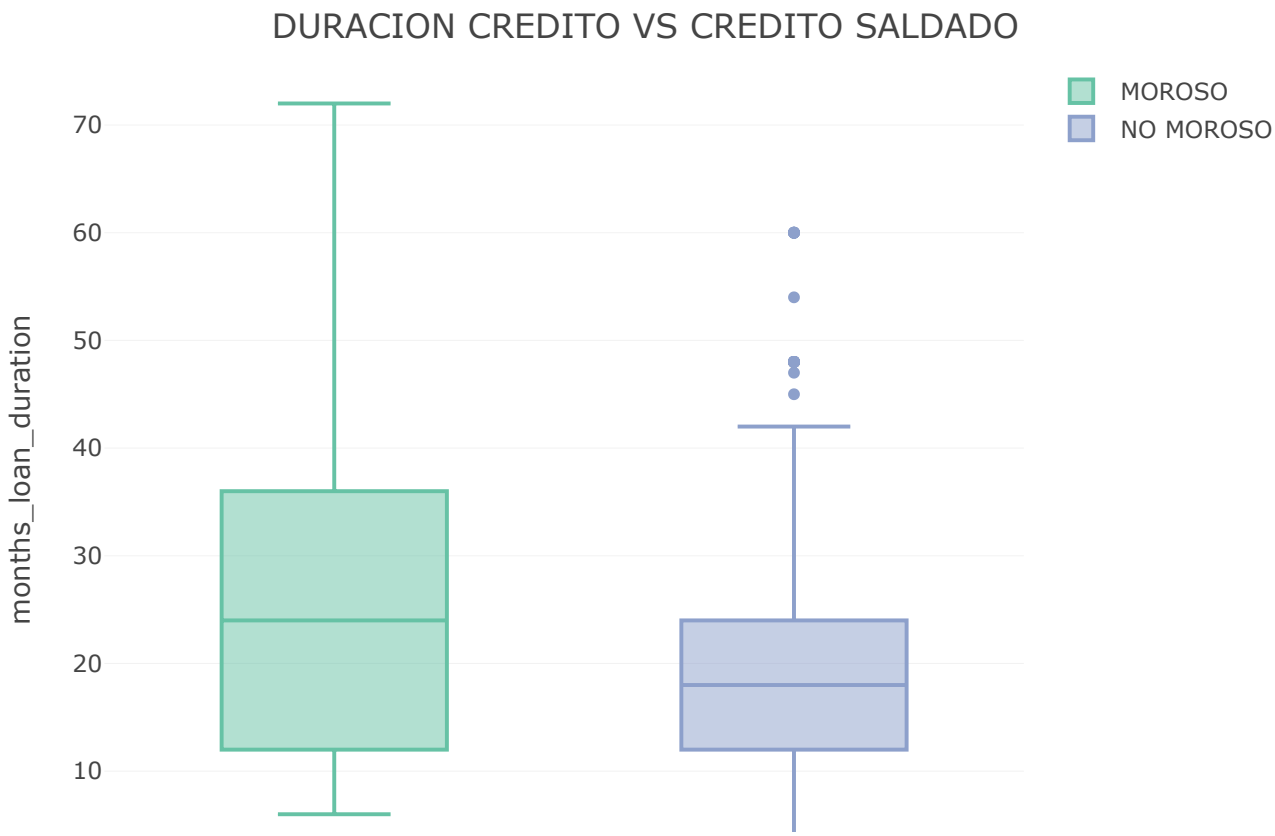
CANTIDAD DE AHORRO

```
#Estadisticas
knitr::kable(data.frame(table(datos$savings_balance)), Caption = "savings_balance", col.
names = c("Saldo ahorrado", "Frecuencia"))
```

Saldo ahorrado	Frecuencia
< 100 DM	603
> 1000 DM	48
101 - 500 DM	103
501 - 1000 DM	63
unknown	183

Como se puede observar, las personas suelen tener un saldo ahorrado < 100 DM, y tienen un índice menor de morosidad.

```
#Gráfica
plot_ly(datos, y = ~months_loan_duration, color = ~default, type = "box") %>%  layout(ti
tle = "DURACIÓN CRÉDITO VS CREDITO SALDADO")
```

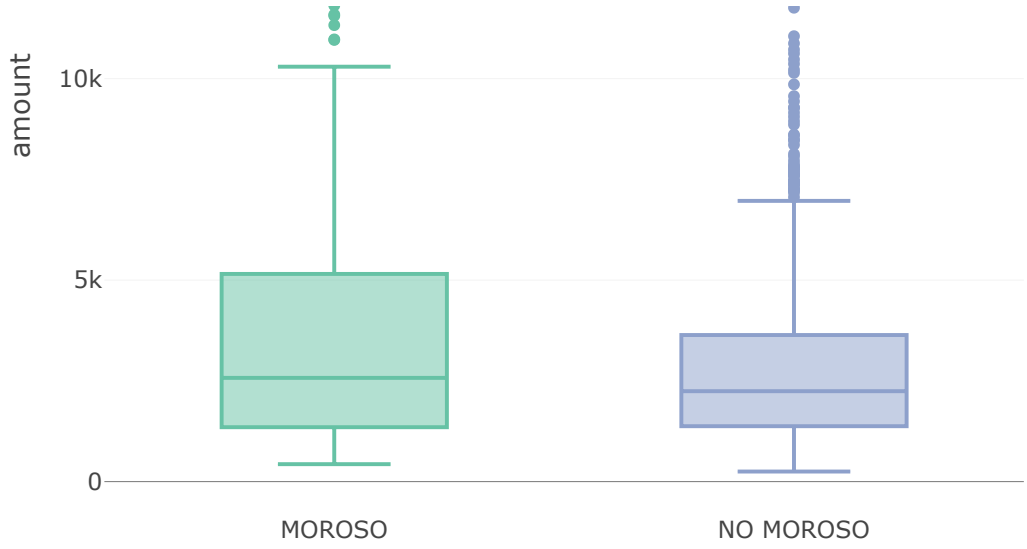



```
#Estadisticas
knitr::kable(basicStats(datos$months_loan_duration), Caption = "Duración", digits = 2, c
ol.names = c("Valores"))
```

	Valores
nobs	1000.00
NAs	0.00
Minimum	4.00
Maximum	72.00
1. Quartile	12.00
3. Quartile	24.00
Mean	20.90
Median	18.00
Sum	20903.00
SE Mean	0.38
LCL Mean	20.15
UCL Mean	21.65
Variance	145.42
Stdev	12.06
Skewness	1.09
Kurtosis	0.90

Como se puede observar, los créditos más cortos se suelen pagar más que los largos Además, la tabla de estadísticas da una información bastante completa sobre los cuartiles, la media, varianza...

```
#Gráfica
plot_ly(datos, y = ~amount, color = ~default, type = "box")  %>%  layout(title = "CANTID
AD CRÉDITO VS CREDITO SALDADO")
```




```
#Estadisticas
knitr::kable(basicStats(datos$amount), Caption = "Importe del crédito", digits = 2, col.
names = c("Valores"))
```

	Valores
nobs	1000.00
NAs	0.00
Minimum	250.00
Maximum	18424.00
1. Quartile	1365.50
3. Quartile	3972.25
Mean	3271.26
Median	2319.50
Sum	3271258.00
SE Mean	89.26
LCL Mean	3096.09
UCL Mean	3446.42
Variance	7967843.47
Stdev	2822.74
Skewness	1.94
Kurtosis	4.25

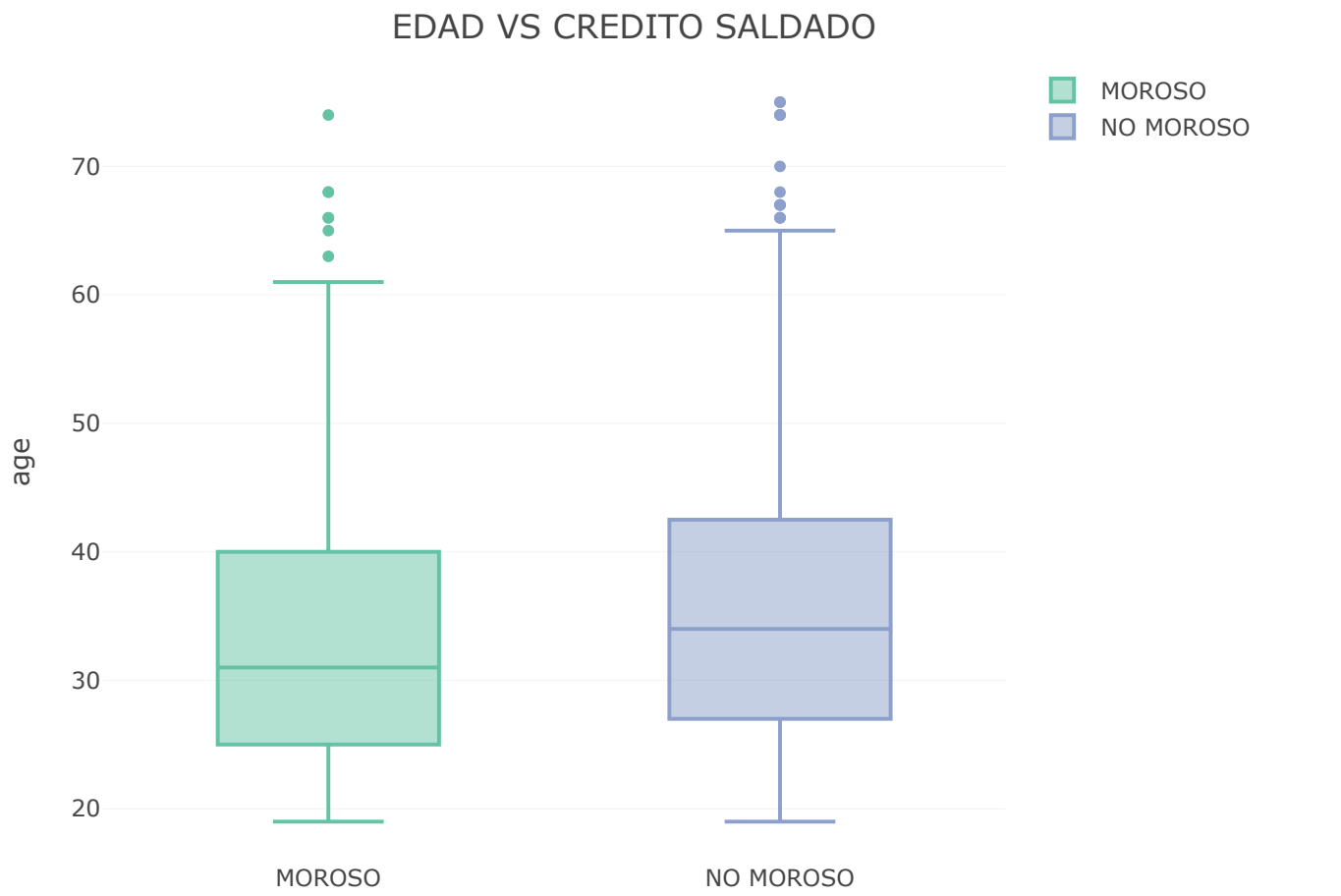
Como se puede observar, los créditos más pequeños se suelen pagar más que los más grandes Además, la tabla de estadísticas da una información bastante completa sobre los cuartiles, la media, varianza...

A primera vista podemos observar que los créditos mas grandes y la cantidad de tiempo de pago mas larga son los que menos se suelen pagar. Además, las personas con una cantidad de dinero ahorrado o en la cuenta menor son los más morosos.

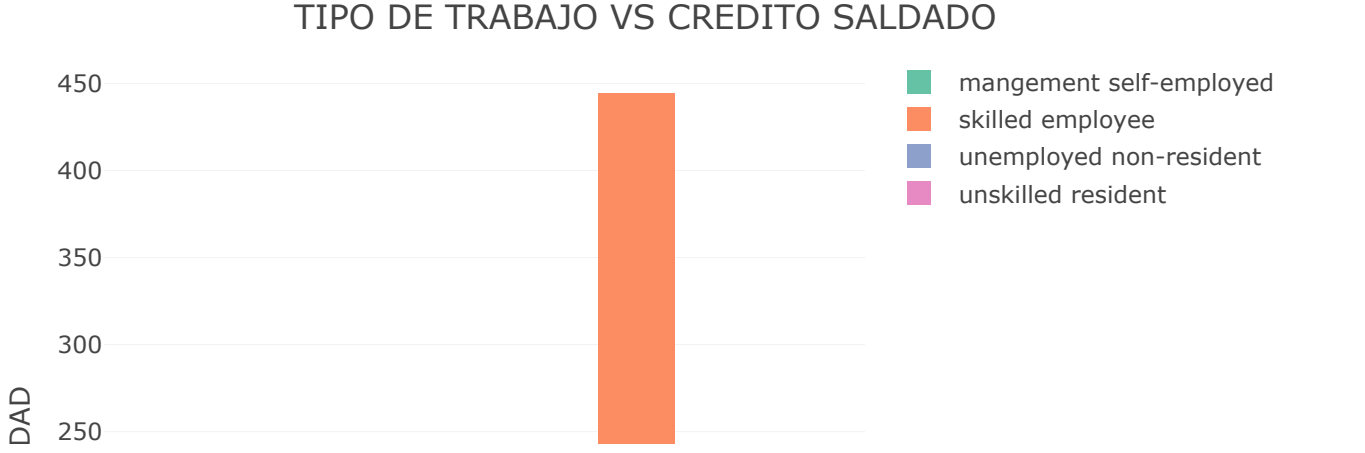
Ahora seguiremos explorando el resto de los campos, visualizándolos y comparándolos con el campo default.

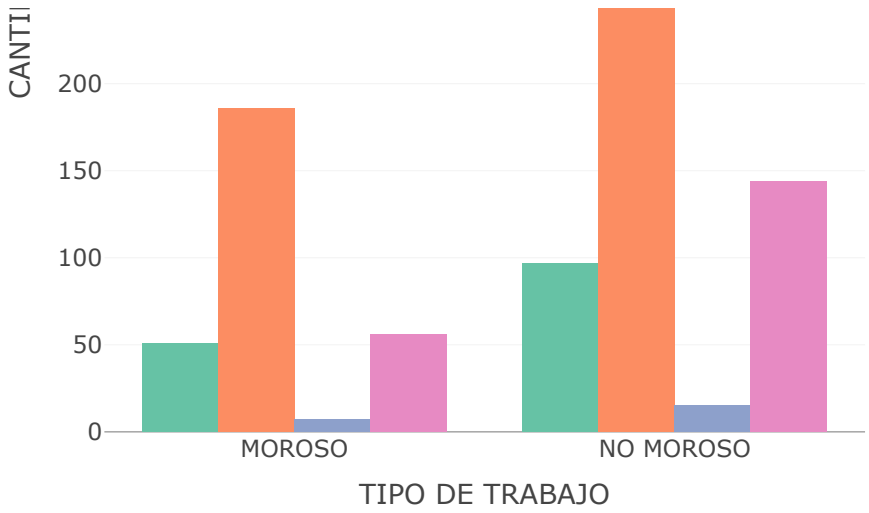
Los siguientes campos para comparar son los relativos a la personas y su trabajo:

```
#Gráfica EDAD
plot_ly(datos, y = ~age, color = ~default, type = "box")   %>%  layout(title = "EDAD VS
 CREDITO SALDADO")
```

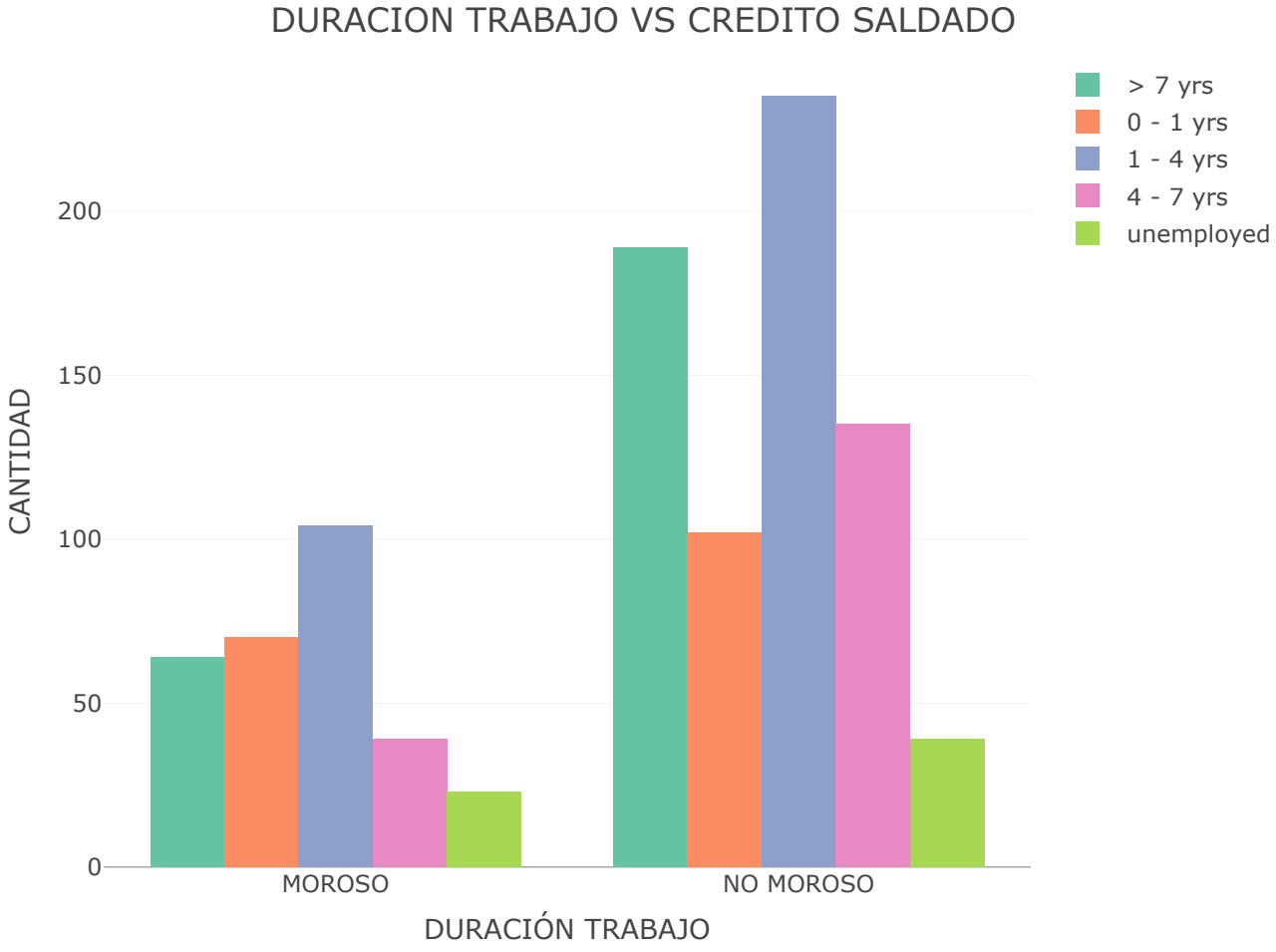



```
#Gráfica tipo Trabajo
datos %>% count(default, job) %>% plot_ly(x = ~default, y = ~n, color = ~job, type = "ba
r")  %>%  layout(title = "TIPO DE TRABAJO VS CREDITO SALDADO", xaxis = list(title = "TIP
O DE TRABAJO"), yaxis = list(title = "CANTIDAD"))
```

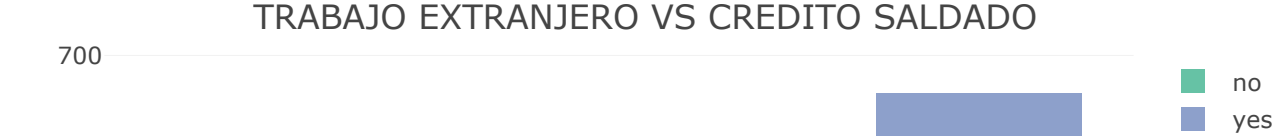


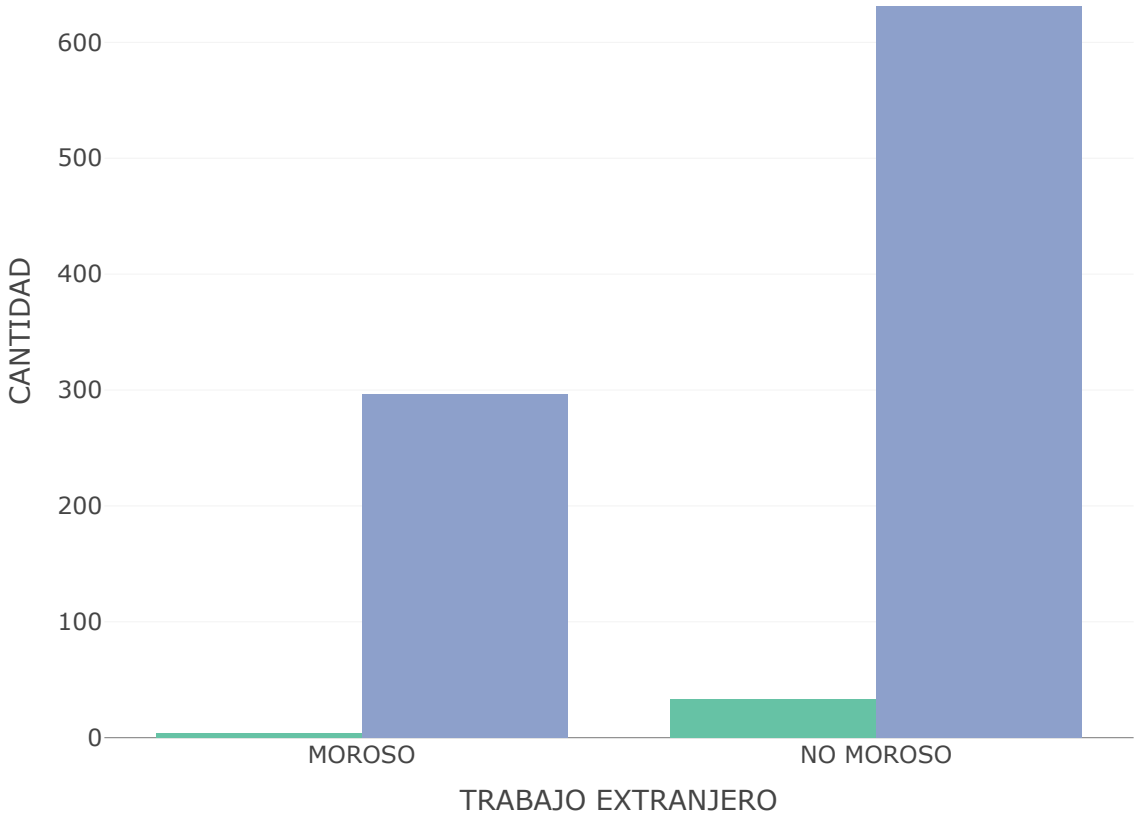



```
#Gráfica duración del Trabajo
datos %>% count(default, employment_length) %>% plot_ly(x = ~default, y = ~n, color = ~e
mployment_length, type = "bar") %>%  layout(title = "DURACIÓN TRABAJO VS CREDITO SALDAD
O", xaxis = list(title = "DURACIÓN TRABAJO"), yaxis = list(title = "CANTIDAD"))
```

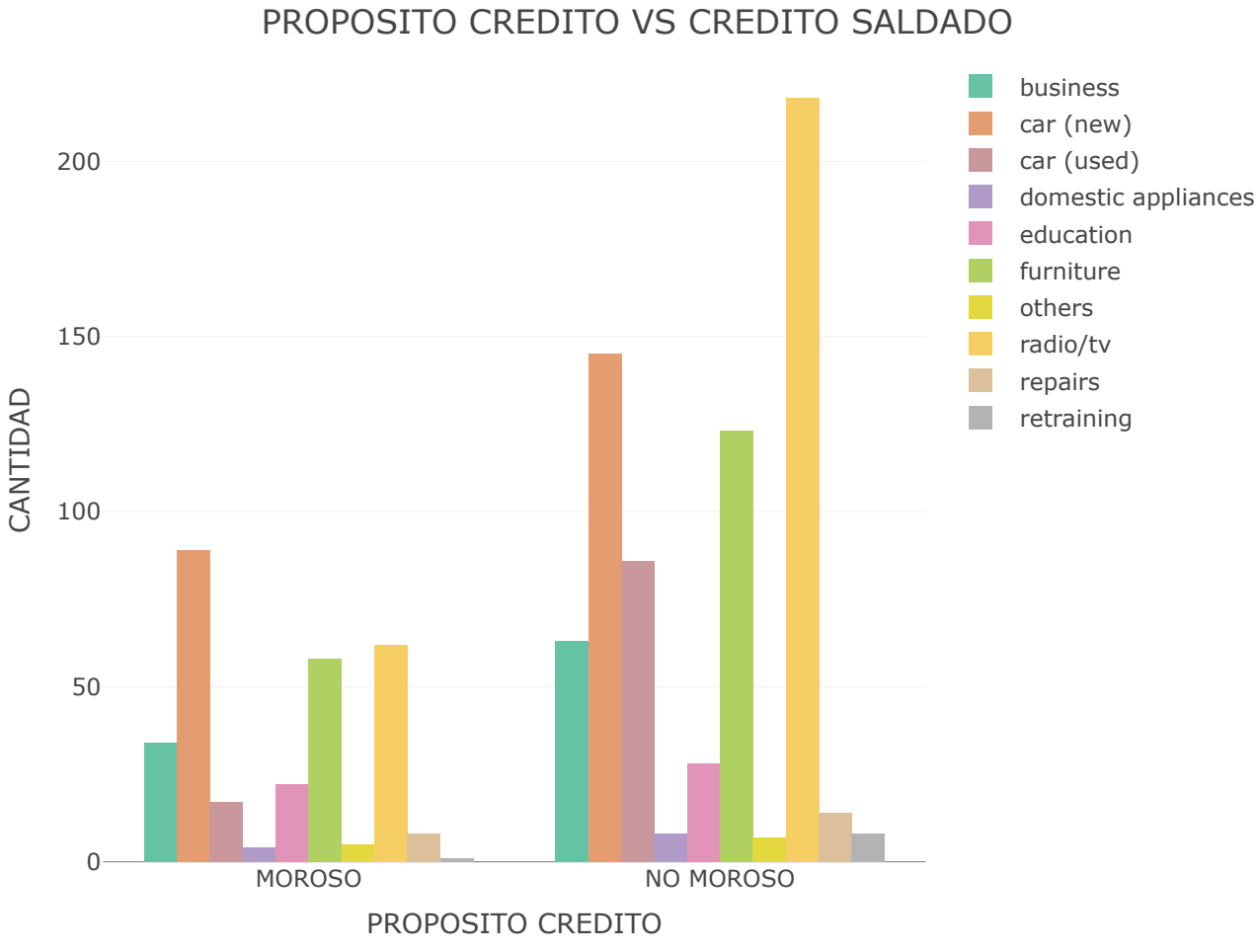



```
#Gráfica Trabajador extranjero
datos %>% count(default, foreign_worker) %>% plot_ly(x = ~default, y = ~n, color = ~fore
ign_worker, type = "bar") %>%  layout(title = "TRABAJO EXTRANJERO VS CREDITO SALDADO", x
axis = list(title = "TRABAJO EXTRANJERO"), yaxis = list(title = "CANTIDAD"))
```






```
#Gráfica Proposito
datos %>% count(default, purpose) %>% plot_ly(x = ~default, y = ~n, color = ~purpose, ty
pe = "bar") %>%  layout(title = "PROPOSITO CREDITO VS CREDITO SALDADO", xaxis = list(tit
le = "PROPOSITO CREDITO"), yaxis = list(title = "CANTIDAD"))
```



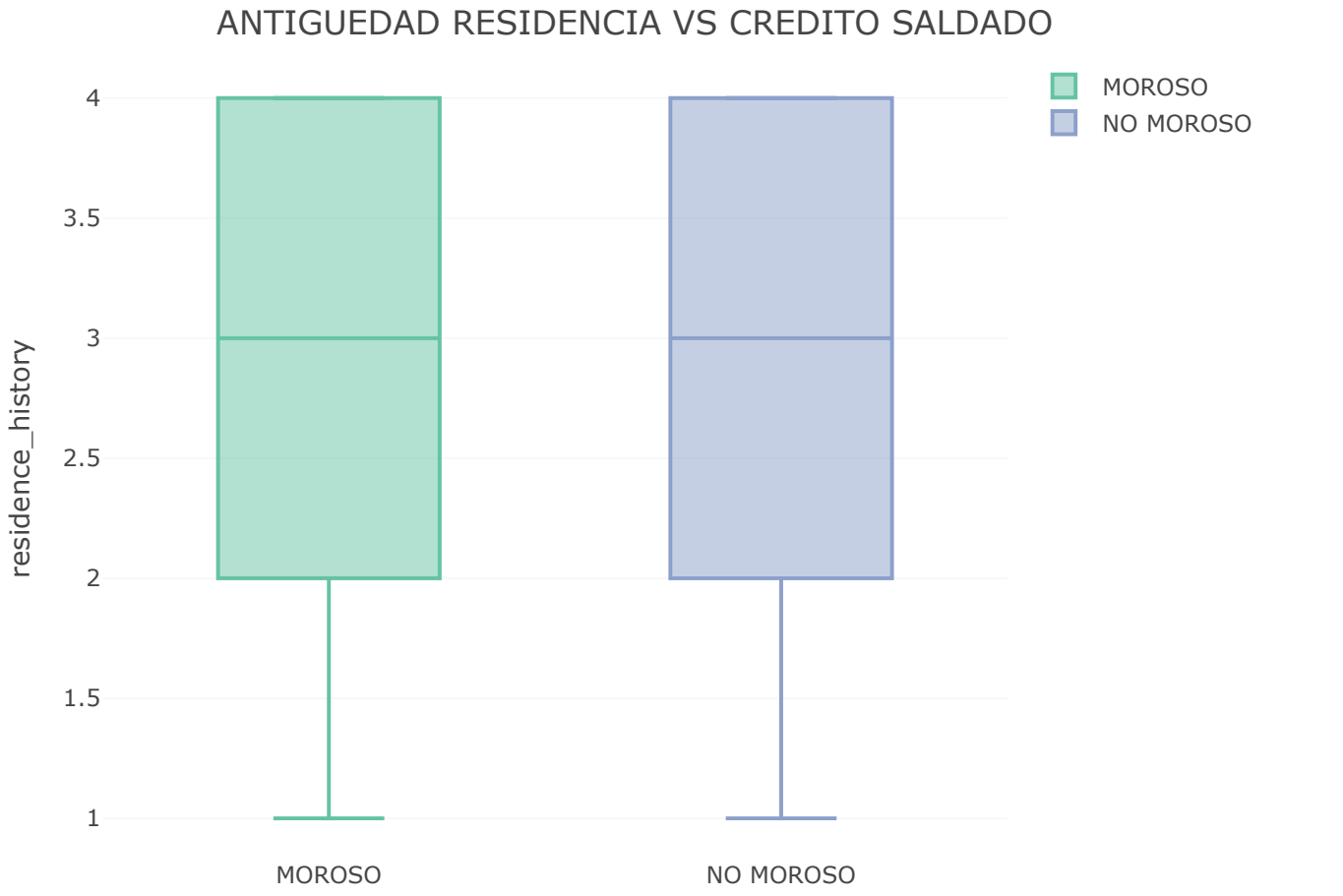
Se puede observar que la variable edad, al tipo de trabajo de la persona y si trabaja en el extranjero no son influyentes en si se paga o no el crédito, ya que no hay mucha diferencia.

Respecto a la duración del trabajo de las personas, los datos suelen estar proporcionados, a excepción de los que tienen un trabajo de entre 1 - 4 años, que suelen ser menos morosos.

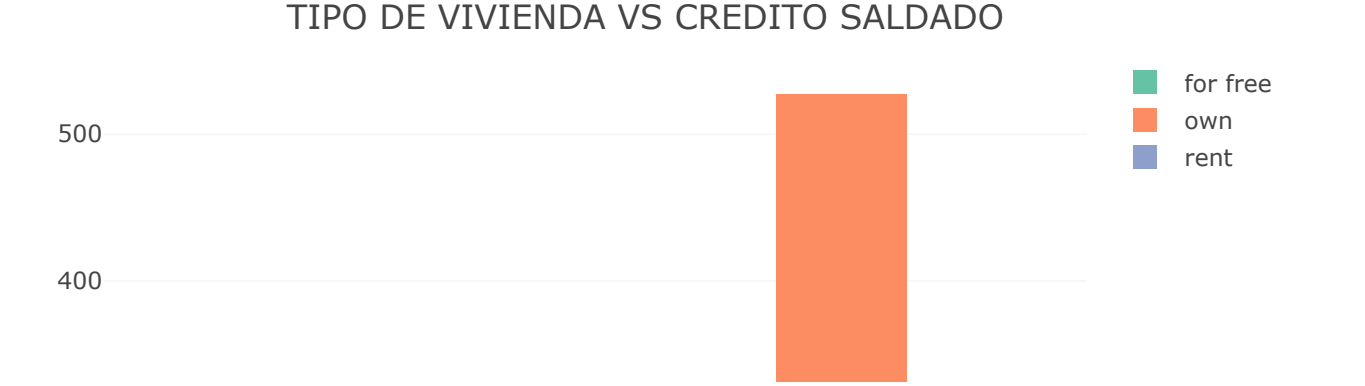
Finalmente, observamos que la gente suele pedir crédito para comprar un coche nuevo, muebles y radio/tv.

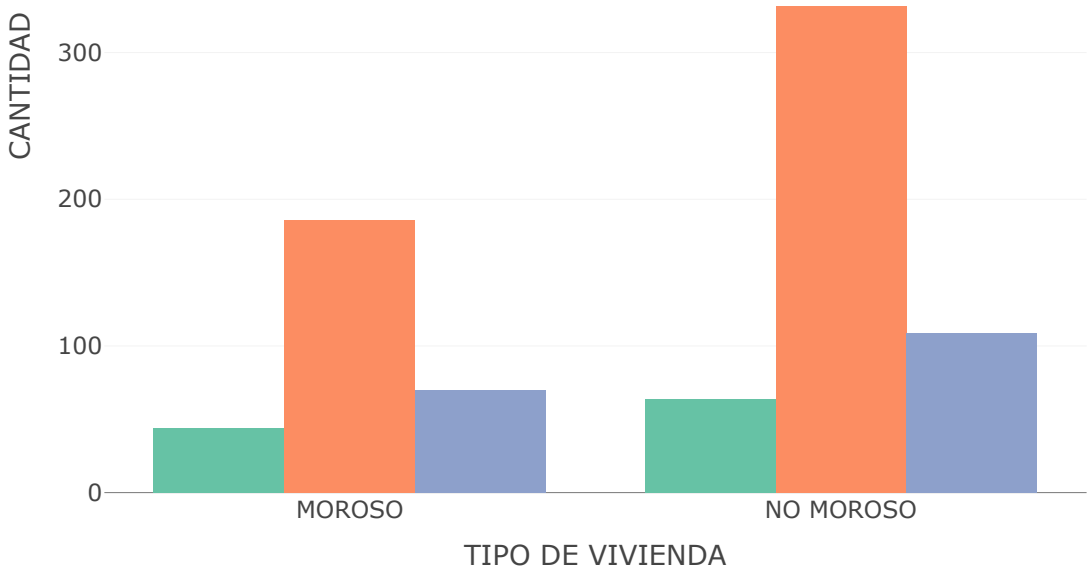
Ahora se va a comparar campos referentes al tipo de propiedades/inversiones

```
#Gráfica Antiguedad Residencia
plot_ly(datos, y = ~residence_history, color = ~default, type = "box")  %>%  layout(titl
e = "ANTIGUEDAD RESIDENCIA VS CREDITO SALDADO")
```

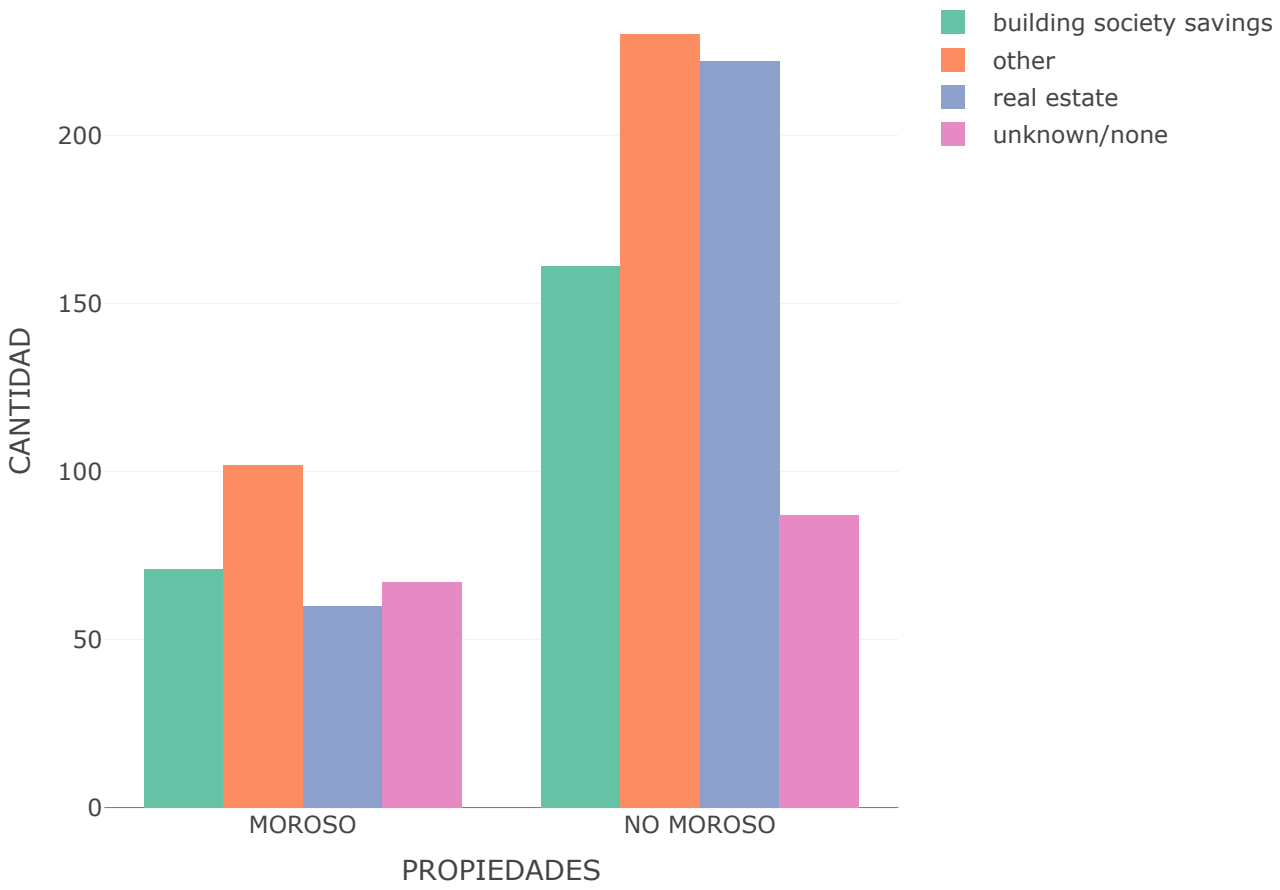



```
#Gráfica Tipo de Vivienda
datos %>% count(default, housing) %>% plot_ly(x = ~default, y = ~n, color = ~housing, ty
pe = "bar") %>%  layout(title = "TIPO DE VIVIENDA VS CREDITO SALDADO", xaxis = list(titl
e = "TIPO DE VIVIENDA"), yaxis = list(title = "CANTIDAD"))
```

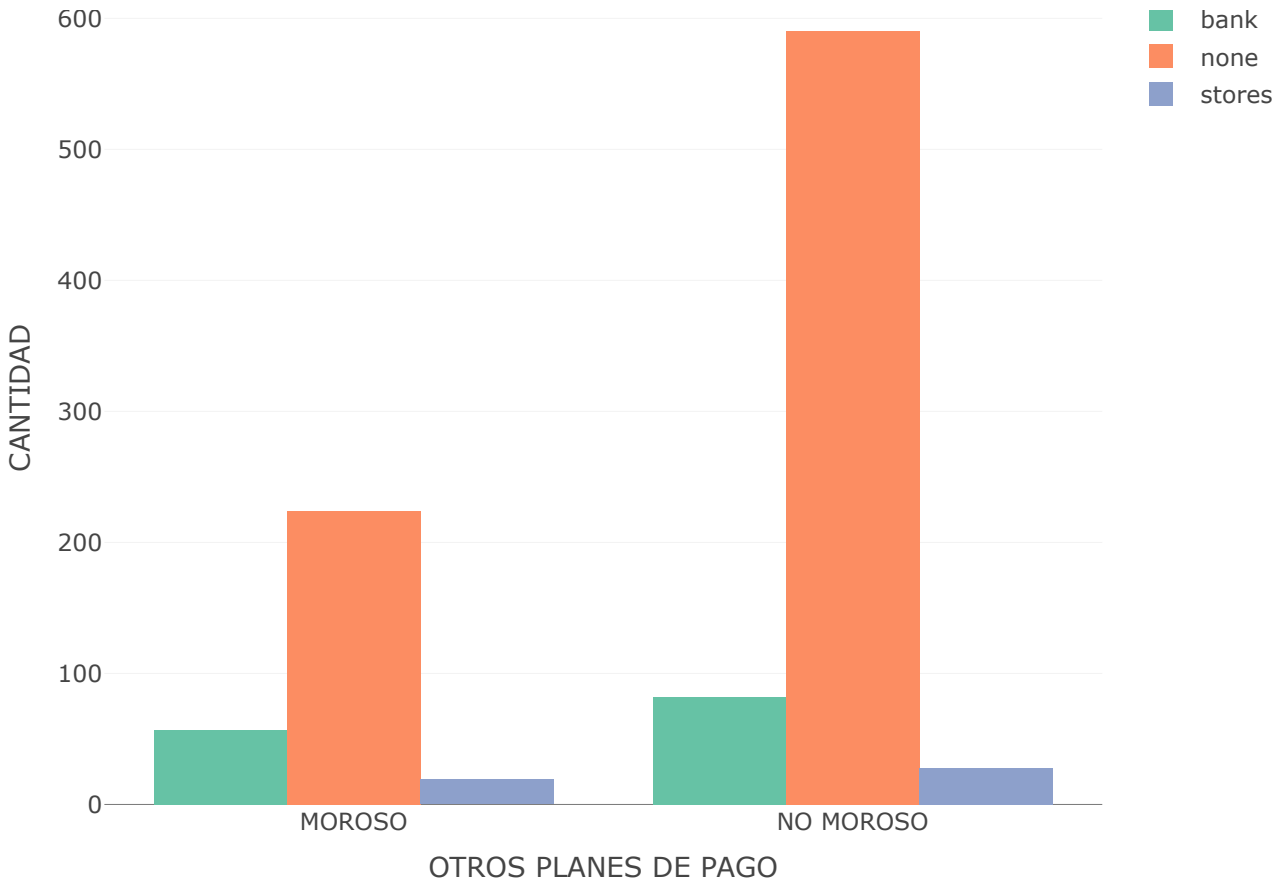





```
#Gráfica Propiedades o seguros
datos %>% count(default, property) %>% plot_ly(x = ~default, y = ~n, color = ~property,
 type = "bar") %>%  layout(title = "PROPIEDADES VS CREDITO SALDADO", xaxis = list(title
 = "PROPIEDADES"), yaxis = list(title = "CANTIDAD"))
```




```
#Gráfica Otros planes de pago
datos %>% count(default, installment_plan) %>% plot_ly(x = ~default, y = ~n, color = ~in
stallment_plan, type = "bar") %>%  layout(title = "OTROS PLANES DE PAGO VS CREDITO SALDA
DO", xaxis = list(title = "OTROS PLANES DE PAGO"), yaxis = list(title = "CANTIDAD"))
```



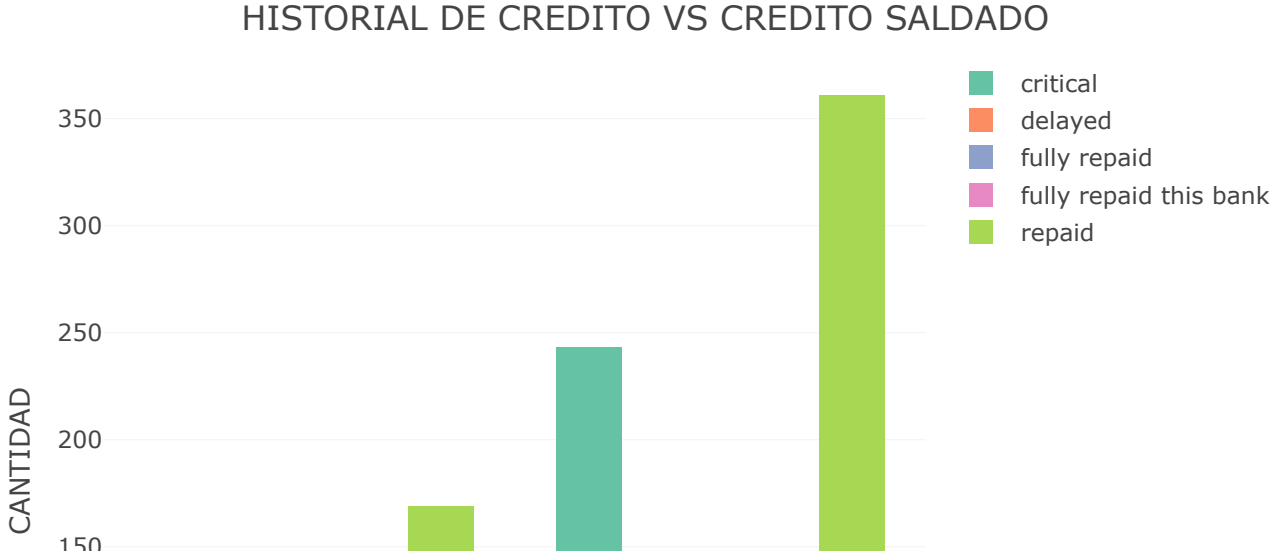
Vemos que la antigüedad de la residencia y que tengan otras formas de pago no son unos factores que influyentes en la morosidad, moviéndose el grueso de estos valores entre 2,4 en la antigüedad de la residencia y que no dispongan de otro planes de pago.

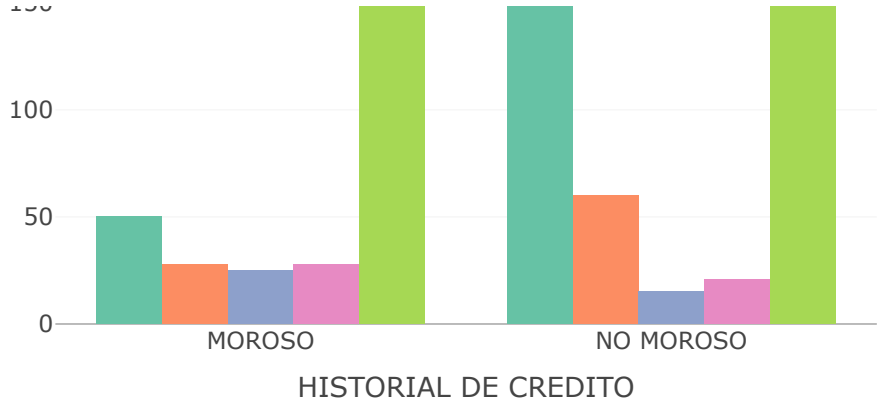
Por otro lado, el tipo de propiedad que tienen son mayoritariamente en propiedad, no existiendo (a simple vista) ninguna variación rara respecto a la morosidad.

Finalmente, las personas que pagan el crédito suelen tener en mayor o menor medida alguna otra propiedad, mientras los que o tienen otro tipo de propiedad o del tipo “real estate” suelen ser los menos morosos.

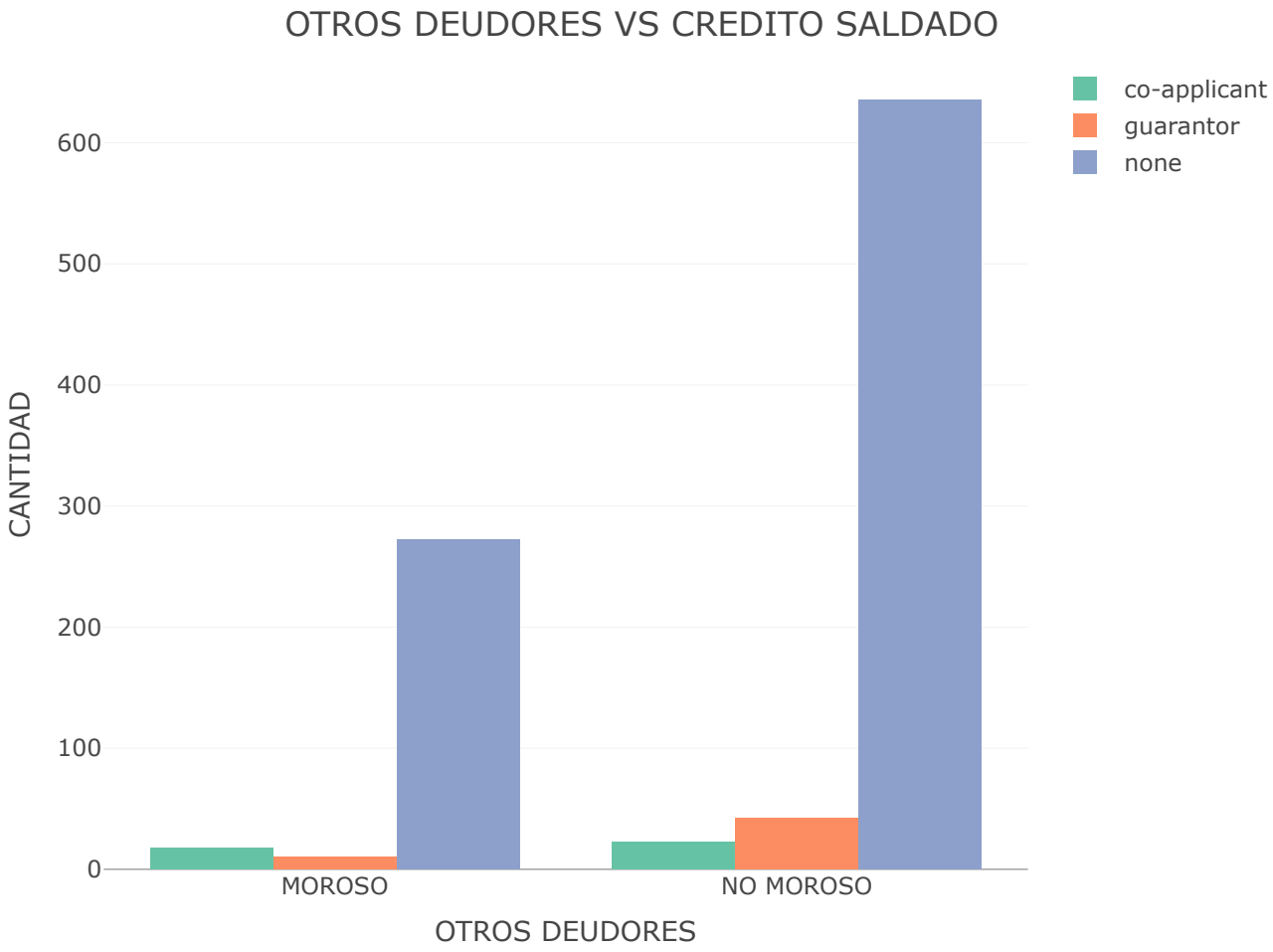
Ahora se va a comparar como funcionan el resto de las variables.

```
#Gráfica Historial de credito
datos %>% count(default, credit_history) %>% plot_ly(x = ~default, y = ~n, color = ~cred
it_history, type = "bar") %>%  layout(title = "HISTORIAL DE CREDITO VS CREDITO SALDADO",
xaxis = list(title = "HISTORIAL DE CREDITO"), yaxis = list(title = "CANTIDAD"))
```

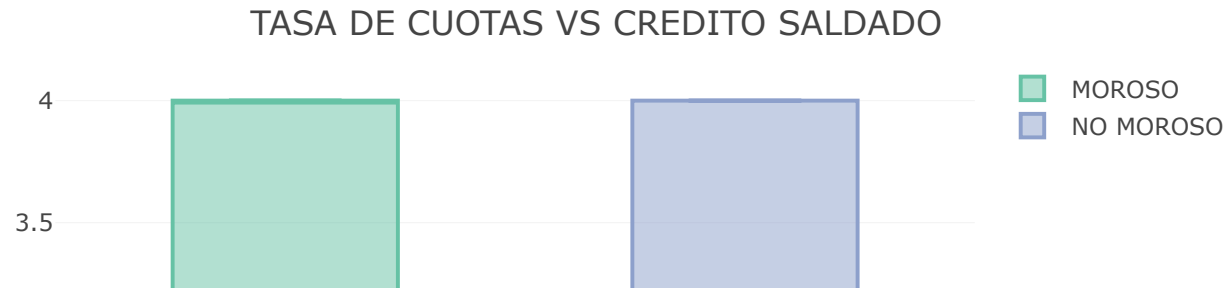


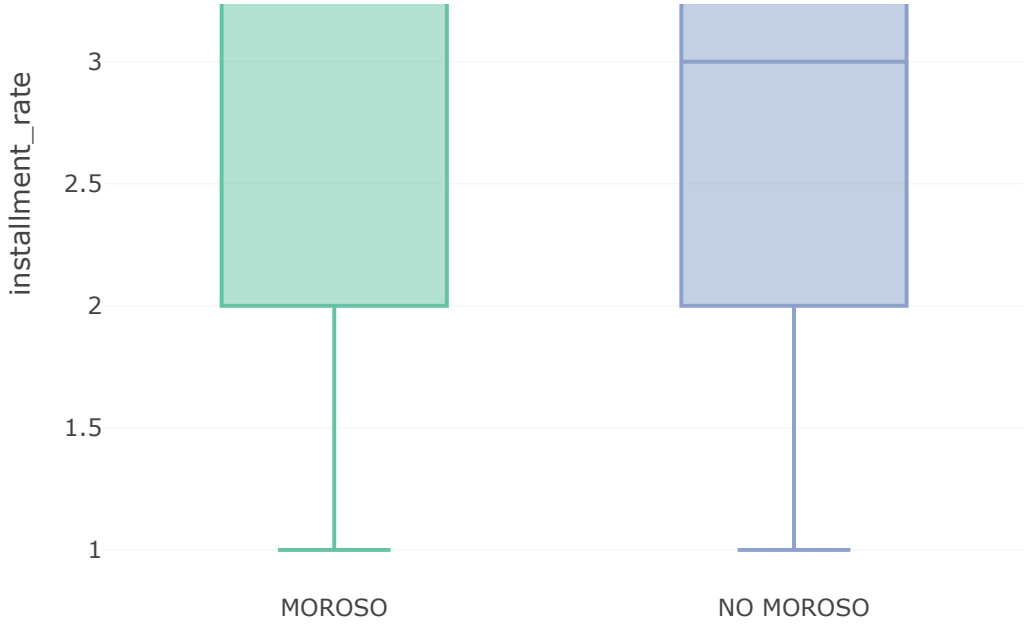



```
#Gráfica Otros deudores
datos %>% count(default, other_debtors) %>% plot_ly(x = ~default, y = ~n, color = ~other
_debtors, type = "bar") %>%  layout(title = "OTROS DEUDORES VS CREDITO SALDADO", xaxis =
list(title = "OTROS DEUDORES"), yaxis = list(title = "CANTIDAD"))
```

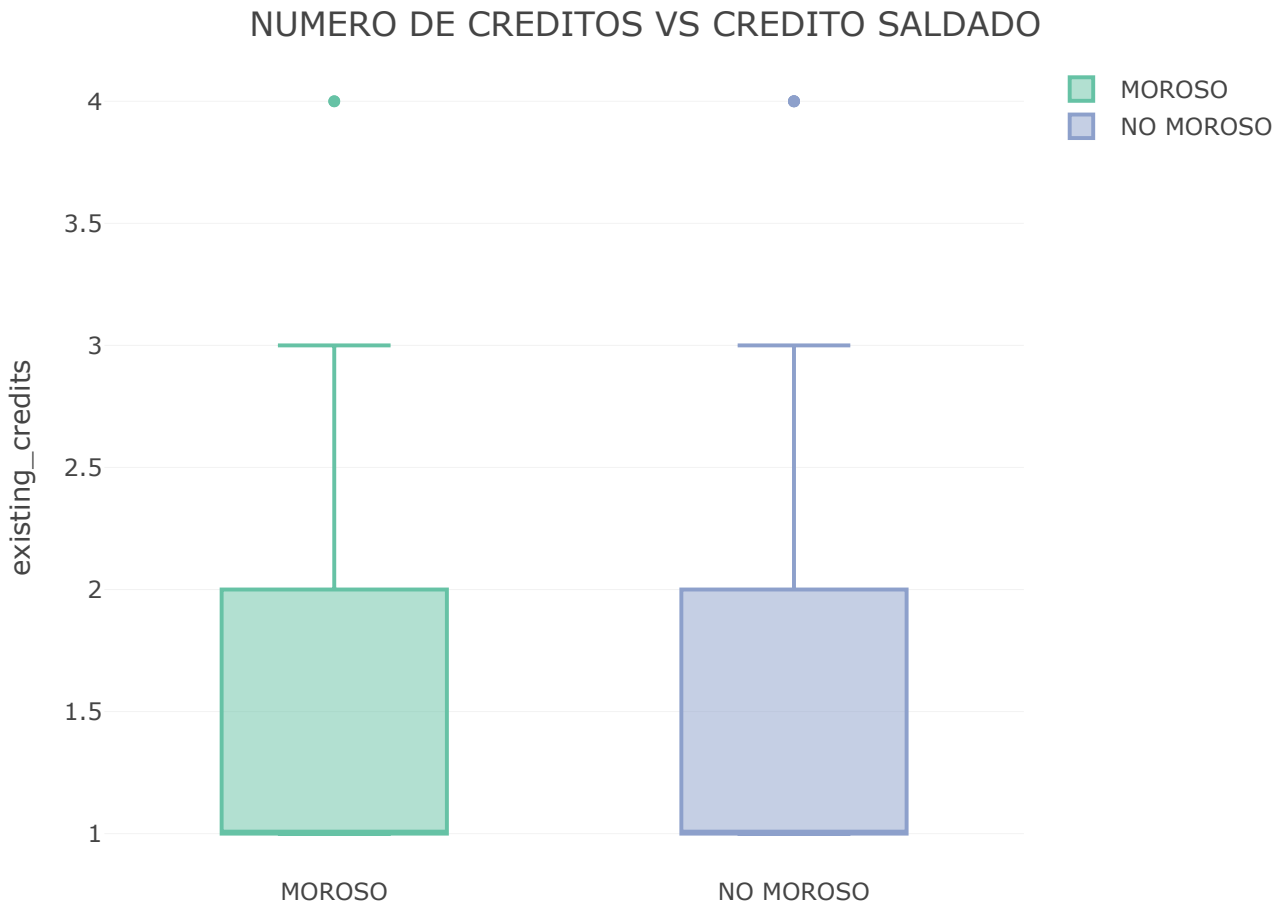



```
#Gráfica Tasa de cuotas
plot_ly(datos, y = ~installment_rate, color = ~default, type = "box")  %>%  layout(title
= "TASA DE CUOTAS VS CREDITO SALDADO")
```

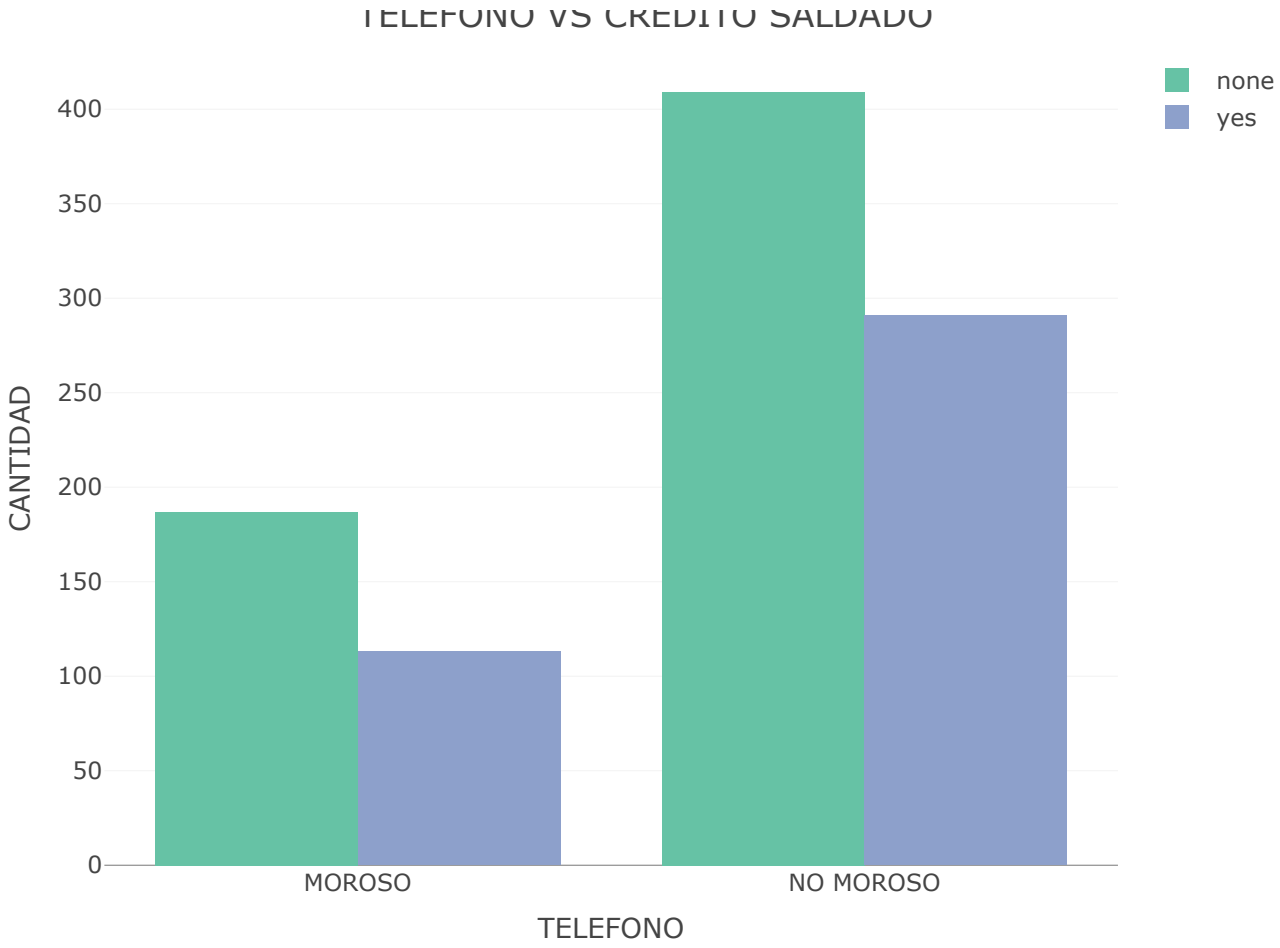





```
#Gráfica Número de créditos
plot_ly(datos, y = ~existing_credits, color = ~default, type = "box")  %>%  layout(title
= "NUMERO DE CREDITOS VS CREDITO SALDADO")
```




```
#Gráfica Telefono
datos %>% count(default, telephone) %>% plot_ly(x = ~default, y = ~n, color = ~telephon
e, type = "bar") %>%  layout(title = "TELEFONO VS CREDITO SALDADO", xaxis = list(title =
"TELEFONO"), yaxis = list(title = "CANTIDAD"))
```



Respecto a al campo historial de créditos, podemos ver que mayormente tienen algún otro crédito pagado anteriormente, sin embargo, los no morosos tienen algún otro crédito en estado crítico. En relación a este campo esta el numero de créditos, que aunque no es influyente, las personas suelen tener entre 1 y 2 créditos a la vez.

La tasa de cuotas de los créditos no algo influyente en la morosidad, pero suele ir mayormente de entre los 2 a 4 años, otro campo en las misma circunstancia es el poseer numero de teléfono, ya que hay mas casos en los que no se poseen.

Ahora, para comprobar los campos mas y menos influyente de una manera numérica, se hará unas pruebas estadísticas de significancia, para así determinar si se puede descartar algún campo. Para ellos se mirarán las proporciones, y luego se calculará los coeficientes V de Cramér y Phi.

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

```
#Campo checking_balance
tabla_aux <- table(datos$checking_balance,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                  MOROSO NO MOROSO
##   < 0 DM     0.4927007 0.5072993
##   > 200 DM   0.2222222 0.7777778
##   1 - 200 DM 0.3903346 0.6096654
##   unknown    0.1167513 0.8832487
```

```
Phi(tabla_aux)
```

```
## [1] 0.3517399
```

```
CramerV(tabla_aux)
```

```
## [1] 0.3517399
```

El tipo de asociación es media, por lo que se deja el campo.

```
#Campo months_loan_duration
tabla_aux <- table(datos$months_loan_duration,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##         MOROSO NO MOROSO
##   4  0.0000000 1.0000000
##   5  0.0000000 1.0000000
##   6  0.1200000 0.8800000
##   7  0.0000000 1.0000000
##   8  0.1428571 0.8571429
##   9  0.2857143 0.7142857
##   10 0.1071429 0.8928571
##   11 0.0000000 1.0000000
##   12 0.2737430 0.7262570
##   13 0.0000000 1.0000000
##   14 0.2500000 0.7500000
##   15 0.1875000 0.8125000
##   16 0.5000000 0.5000000
##   18 0.3716814 0.6283186
##   20 0.1250000 0.8750000
##   21 0.3000000 0.7000000
##   22 0.0000000 1.0000000
##   24 0.3043478 0.6956522
##   26 0.0000000 1.0000000
##   27 0.3846154 0.6153846
##   28 0.3333333 0.6666667
##   30 0.3250000 0.6750000
##   33 0.3333333 0.6666667
##   36 0.4457831 0.5542169
##   39 0.2000000 0.8000000
##   40 1.0000000 0.0000000
##   42 0.2727273 0.7272727
##   45 0.8000000 0.2000000
##   47 0.0000000 1.0000000
##   48 0.5833333 0.4166667
##   54 0.5000000 0.5000000
##   60 0.4615385 0.5384615
##   72 1.0000000 0.0000000
```

```
Phi(tabla_aux)
```

```
## [1] 0.2808682
```

```
CramerV(tabla_aux)
```

```
## [1] 0.2808682
```

El tipo de asociación es media/baja, por lo que se deja el campo.

```
#Campo credit_history
tabla_aux <- table(datos$credit_history,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                            MOROSO NO MOROSO
##   critical               0.1706485 0.8293515
##   delayed                0.3181818 0.6818182
##   fully repaid           0.6250000 0.3750000
##   fully repaid this bank 0.5714286 0.4285714
##   repaid                 0.3188679 0.6811321
```

```
Phi(tabla_aux)
```

```
## [1] 0.2483775
```

```
CramerV(tabla_aux)
```

```
## [1] 0.2483775
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo purpose
tabla_aux <- table(datos$purpose,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                          MOROSO NO MOROSO
##   business            0.3505155 0.6494845
##   car (new)           0.3803419 0.6196581
##   car (used)          0.1650485 0.8349515
##   domestic appliances 0.3333333 0.6666667
##   education           0.4400000 0.5600000
##   furniture           0.3204420 0.6795580
##   others              0.4166667 0.5833333
##   radio/tv            0.2214286 0.7785714
##   repairs             0.3636364 0.6363636
##   retraining          0.1111111 0.8888889
```

```
Phi(tabla_aux)
```

```
## [1] 0.1826375
```

```
CramerV(tabla_aux)
```

```
## [1] 0.1826375
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo amount
tabla_aux <- table(datos$amount,datos$default )
Phi(tabla_aux)
```

```
## [1] 0.9652699
```

```
CramerV(tabla_aux)
```

```
## [1] 0.9652699
```

El tipo de asociación es muy alta, por lo que se deja el campo.

```
#Campo savings_balance
tabla_aux <- table(datos$savings_balance,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                     MOROSO NO MOROSO
##   < 100 DM      0.3598673 0.6401327
##   > 1000 DM     0.1250000 0.8750000
##   101 - 500 DM  0.3300971 0.6699029
##   501 - 1000 DM 0.1746032 0.8253968
##   unknown       0.1748634 0.8251366
```

```
Phi(tabla_aux)
```

```
## [1] 0.1899972
```

```
CramerV(tabla_aux)
```

```
## [1] 0.1899972
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo employment_length
tabla_aux <- table(datos$employment_length,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                  MOROSO NO MOROSO
##   > 7 yrs    0.2529644 0.7470356
##   0 - 1 yrs  0.4069767 0.5930233
##   1 - 4 yrs  0.3067847 0.6932153
##   4 - 7 yrs  0.2241379 0.7758621
##   unemployed 0.3709677 0.6290323
```

```
Phi(tabla_aux)
```

```
## [1] 0.1355296
```

```
CramerV(tabla_aux)
```

```
## [1] 0.1355296
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo installment_rate
tabla_aux <- table(datos$installment_rate,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##        MOROSO NO MOROSO
##   1 0.2500000 0.7500000
##   2 0.2683983 0.7316017
##   3 0.2866242 0.7133758
##   4 0.3340336 0.6659664
```

```
Phi(tabla_aux)
```

```
## [1] 0.07400535
```

```
CramerV(tabla_aux)
```

```
## [1] 0.07400535
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo personal_status
tabla_aux <- table(datos$personal_status,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                    MOROSO NO MOROSO
##   divorced male 0.4000000 0.6000000
##   female        0.3516129 0.6483871
##   married male  0.2717391 0.7282609
##   single male   0.2664234 0.7335766
```

```
Phi(tabla_aux)
```

```
## [1] 0.09800619
```

```
CramerV(tabla_aux)
```

```
## [1] 0.09800619
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo other_debtors
tabla_aux <- table(datos$other_debtors,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                    MOROSO NO MOROSO
##   co-applicant 0.4390244 0.5609756
##   guarantor    0.1923077 0.8076923
##   none         0.2998897 0.7001103
```

```
Phi(tabla_aux)
```

```
## [1] 0.08151912
```

```
CramerV(tabla_aux)
```

```
## [1] 0.08151912
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo residence_history
tabla_aux <- table(datos$residence_history,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##        MOROSO NO MOROSO
##   1 0.2769231 0.7230769
##   2 0.3149351 0.6850649
##   3 0.2885906 0.7114094
##   4 0.3002421 0.6997579
```

```
Phi(tabla_aux)
```

```
## [1] 0.02737328
```

```
CramerV(tabla_aux)
```

```
## [1] 0.02737328
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo property
tabla_aux <- table(datos$property,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                               MOROSO NO MOROSO
##   building society savings 0.3060345 0.6939655
##   other                    0.3072289 0.6927711
##   real estate              0.2127660 0.7872340
##   unknown/none             0.4350649 0.5649351
```

```
Phi(tabla_aux)
```

```
## [1] 0.1540115
```

```
CramerV(tabla_aux)
```

```
## [1] 0.1540115
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo age
tabla_aux <- table(datos$age,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##          MOROSO  NO MOROSO
##   19 0.50000000 0.50000000
##   20 0.35714286 0.64285714
##   21 0.35714286 0.64285714
##   22 0.40740741 0.59259259
##   23 0.41666667 0.58333333
##   24 0.43181818 0.56818182
##   25 0.46341463 0.53658537
##   26 0.28000000 0.72000000
##   27 0.25490196 0.74509804
##   28 0.34883721 0.65116279
##   29 0.40540541 0.59459459
##   30 0.27500000 0.72500000
##   31 0.28947368 0.71052632
##   32 0.26470588 0.73529412
##   33 0.39393939 0.60606061
##   34 0.34375000 0.65625000
##   35 0.15000000 0.85000000
##   36 0.15384615 0.84615385
##   37 0.27586207 0.72413793
##   38 0.16666667 0.83333333
##   39 0.28571429 0.71428571
##   40 0.24000000 0.76000000
##   41 0.23529412 0.76470588
##   42 0.36363636 0.63636364
##   43 0.29411765 0.70588235
##   44 0.29411765 0.70588235
##   45 0.20000000 0.80000000
##   46 0.22222222 0.77777778
##   47 0.29411765 0.70588235
##   48 0.25000000 0.75000000
##   49 0.07142857 0.92857143
##   50 0.25000000 0.75000000
##   51 0.12500000 0.87500000
##   52 0.11111111 0.88888889
##   53 0.71428571 0.28571429
##   54 0.20000000 0.80000000
##   55 0.37500000 0.62500000
##   56 0.00000000 1.00000000
##   57 0.33333333 0.66666667
##   58 0.40000000 0.60000000
##   59 0.33333333 0.66666667
##   60 0.50000000 0.50000000
##   61 0.42857143 0.57142857
##   62 0.00000000 1.00000000
##   63 0.12500000 0.87500000
##   64 0.00000000 1.00000000
##   65 0.20000000 0.80000000
##   66 0.40000000 0.60000000
##   67 0.00000000 1.00000000
##   68 0.66666667 0.33333333
```

```
##   70 0.00000000 1.00000000
##   74 0.25000000 0.75000000
##   75 0.00000000 1.00000000
```

```
Phi(tabla_aux)
```

```
## [1] 0.2397444
```

```
CramerV(tabla_aux)
```

```
## [1] 0.2397444
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo installment_plan
tabla_aux <- table(datos$installment_plan,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##             MOROSO NO MOROSO
##   bank   0.4100719 0.5899281
##   none   0.2751843 0.7248157
##   stores 0.4042553 0.5957447
```

```
Phi(tabla_aux)
```

```
## [1] 0.1133101
```

```
CramerV(tabla_aux)
```

```
## [1] 0.1133101
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo housing
tabla_aux <- table(datos$housing,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##               MOROSO NO MOROSO
##   for free 0.4074074 0.5925926
##   own      0.2608696 0.7391304
##   rent     0.3910615 0.6089385
```

```
Phi(tabla_aux)
```

```
## [1] 0.1349068
```

```
CramerV(tabla_aux)
```

```
## [1] 0.1349068
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo existing_credits
tabla_aux <- table(datos$existing_credits,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##        MOROSO NO MOROSO
##   1 0.3159558 0.6840442
##   2 0.2762763 0.7237237
##   3 0.2142857 0.7857143
##   4 0.3333333 0.6666667
```

```
Phi(tabla_aux)
```

```
## [1] 0.05168364
```

```
CramerV(tabla_aux)
```

```
## [1] 0.05168364
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo dependents
tabla_aux <- table(datos$dependents,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##        MOROSO NO MOROSO
##   1 0.3005917 0.6994083
##   2 0.2967742 0.7032258
```

```
Phi(tabla_aux)
```

```
## [1] 0.003014853
```

```
CramerV(tabla_aux)
```

```
## [1] 0.003014853
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo telephone
tabla_aux <- table(datos$telephone,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##           MOROSO NO MOROSO
##   none 0.3137584 0.6862416
##   yes  0.2797030 0.7202970
```

```
Phi(tabla_aux)
```

```
## [1] 0.03646619
```

```
CramerV(tabla_aux)
```

```
## [1] 0.03646619
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo foreign_worker
tabla_aux <- table(datos$foreign_worker,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##          MOROSO NO MOROSO
##   no  0.1081081 0.8918919
##   yes 0.3073728 0.6926272
```

```
Phi(tabla_aux)
```

```
## [1] 0.0820795
```

```
CramerV(tabla_aux)
```

```
## [1] 0.0820795
```

El tipo de asociación es baja, por lo que se descarta el campo.

```
#Campo job
tabla_aux <- table(datos$job,datos$default )
prop.table(tabla_aux, margin = 1)
```

```
##
##                               MOROSO NO MOROSO
##   mangement self-employed 0.3445946 0.6554054
##   skilled employee        0.2952381 0.7047619
##   unemployed non-resident 0.3181818 0.6818182
##   unskilled resident      0.2800000 0.7200000
```

```
Phi(tabla_aux)
```

```
## [1] 0.04341838
```

```
CramerV(tabla_aux)
```

```
## [1] 0.04341838
```

El tipo de asociación es baja, por lo que se descarta el campo.

Siguiendo los valores de V de Cramer y Phi, los valores entre 0.1 y 0.3 nos indican que la asociación estadística es baja, y entre 0.3 y 0.5 se puede considerar una asociación media. Finalmente, si los valores fueran superiores a 0.5, la asociación estadística entre las variables sería alta.

```
#Se hace una copia de los datos
datos_CRAMER_PHI_ALTO <- datos

#Se elimina los campos
datos_CRAMER_PHI_ALTO$dependents <- NULL
datos_CRAMER_PHI_ALTO$telephone <- NULL
datos_CRAMER_PHI_ALTO$job <- NULL
datos_CRAMER_PHI_ALTO$foreign_worker <- NULL
datos_CRAMER_PHI_ALTO$housing <- NULL
datos_CRAMER_PHI_ALTO$existing_credits <- NULL
datos_CRAMER_PHI_ALTO$installment_plan <- NULL
datos_CRAMER_PHI_ALTO$property <- NULL
datos_CRAMER_PHI_ALTO$purpose <- NULL
datos_CRAMER_PHI_ALTO$savings_balance <- NULL
datos_CRAMER_PHI_ALTO$employment_length <- NULL
datos_CRAMER_PHI_ALTO$installment_rate <- NULL
datos_CRAMER_PHI_ALTO$personal_status <- NULL
datos_CRAMER_PHI_ALTO$other_debtors <- NULL
datos_CRAMER_PHI_ALTO$residence_history <- NULL
datos_CRAMER_PHI_ALTO$age <- NULL
datos_CRAMER_PHI_ALTO$credit_history <- NULL
```

Ahora para proceder a preparar los datos, la primera cosa que debemos hacer es desordenar los datos.

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

3.2 Primer árbol de decisión

Como debemos dividir el conjunto de datos en dos grupos: entrenamiento y test, y al no existir un conjunto complementario ni proporción fijada, se hará 2/3 de los datos para el entrenamiento y 1/3 de los datos para el test.

La variable por la que clasificaremos es el campo de si la persona ha pagado el crédito o no, que está en la decimoséptima columna. De esta forma, tendremos un conjunto de datos para el entrenamiento y uno para la validación

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

De forma dinámica podemos definir una forma de separar los datos en función de un parámetro, en este caso del "split_prop". Definimos un parámetro que controla el split de forma dinámica en el test.

```
split_prop <- 3
max_split<-floor(nrow(X)/split_prop)
tr_limit <- nrow(X)-max_split
ts_limit <- nrow(X)-max_split+1

trainX <- X[1:tr_limit,]
trainy <- y[1:tr_limit]
testX <- X[(ts_limit+1):nrow(X),]
testy <- y[(ts_limit+1):nrow(X)]
```

En la segunda opción podemos crear directamente un rango utilizando el mismo parámetro anterior.

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

Al extraer aleatoriamente los datos, se hará un análisis mínimo de los datos para asegurarnos de no obtener clasificadores sesgados por los valores que contiene cada muestra.

En este caso, verificaremos que la proporción de morosos es más o menos constante en los dos conjuntos.

```
summary(trainX);
```

```
##  checking_balance   months_loan_duration     amount
##  Length:666         Min.   : 4.00        Min.   :  250
##  Class :character   1st Qu.:12.00        1st Qu.: 1389
##  Mode  :character   Median :18.00        Median : 2324
##                     Mean   :21.05        Mean   : 3280
##                     3rd Qu.:24.00        3rd Qu.: 3973
##                     Max.   :60.00        Max.   :18424
```

```
summary(trainy)
```

```
##    MOROSO NO MOROSO
##       204       462
```

```
summary(testX)
```

```
##  checking_balance   months_loan_duration     amount
##  Length:334         Min.   : 4.00        Min.   :  276
##  Class :character   1st Qu.:12.00        1st Qu.: 1303
##  Mode  :character   Median :18.00        Median : 2274
##                     Mean   :20.61        Mean   : 3254
##                     3rd Qu.:24.00        3rd Qu.: 3970
##                     Max.   :72.00        Max.   :15945
```

```
summary(testy)
```

```
##    MOROSO NO MOROSO
##        96       238
```

Se puede verificar, que hay aproximadamente la misma proporción en el conjunto de entrenamiento y de test. Siendo “NO” un poco mas del doble que “SI” en ambos casos.

Ya que tenemos los conjuntos preparados, se crea el árbol de decisión con los datos de entrenamiento.

```
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]      Sat Dec 11 10:50:06 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 666 cases (4 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (37/6, lift 2.7)
##  checking_balance = < 0 DM
##  months_loan_duration > 11
##  amount <= 1391
##  ->  class MOROSO  [0.821]
##
## Rule 2: (24/4, lift 2.6)
##  checking_balance in {1 - 200 DM, < 0 DM}
##  amount > 8588
##  ->  class MOROSO  [0.808]
##
## Rule 3: (34/10, lift 2.3)
##  checking_balance = < 0 DM
##  months_loan_duration > 30
##  ->  class MOROSO  [0.694]
##
## Rule 4: (160/72, lift 1.8)
##  checking_balance = < 0 DM
##  months_loan_duration > 11
##  ->  class MOROSO  [0.549]
##
## Rule 5: (298/39, lift 1.2)
##  checking_balance in {unknown, > 200 DM}
##  ->  class NO MOROSO  [0.867]
##
## Rule 6: (629/182, lift 1.0)
##  amount <= 8588
##  ->  class NO MOROSO  [0.710]
##
## Default class: NO MOROSO
##
##
## Evaluation on training data (666 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
```

```
##       6  152(22.8%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##      71   133    (a): class MOROSO
##      19   443    (b): class NO MOROSO
##
##
##  Attribute usage:
##
##   98.05% amount
##   71.62% checking_balance
##   24.02% months_loan_duration
##
##
## Time: 0.0 secs
```

3.3 Explicación de las reglas obtenidas

El árbol obtenido clasifica erróneamente 152 de los 666 casos dados, una tasa de error del 22.8%.

A partir del árbol de decisión, se pueden extraer las siguientes 6 reglas de:

checking_balance = < 0 DM && months_loan_duration > 11 && amount <= 1391 –> MOROSO. Validez: 82,1%

checking_balance entre los valores {1 - 200 DM, < 0 DM} && amount > 8588 –> MOROSO. Validez: 80,8%

checking_balance = < 0 DM && months_loan_duration > 30 –> MOROSO. Validez: 69,4%

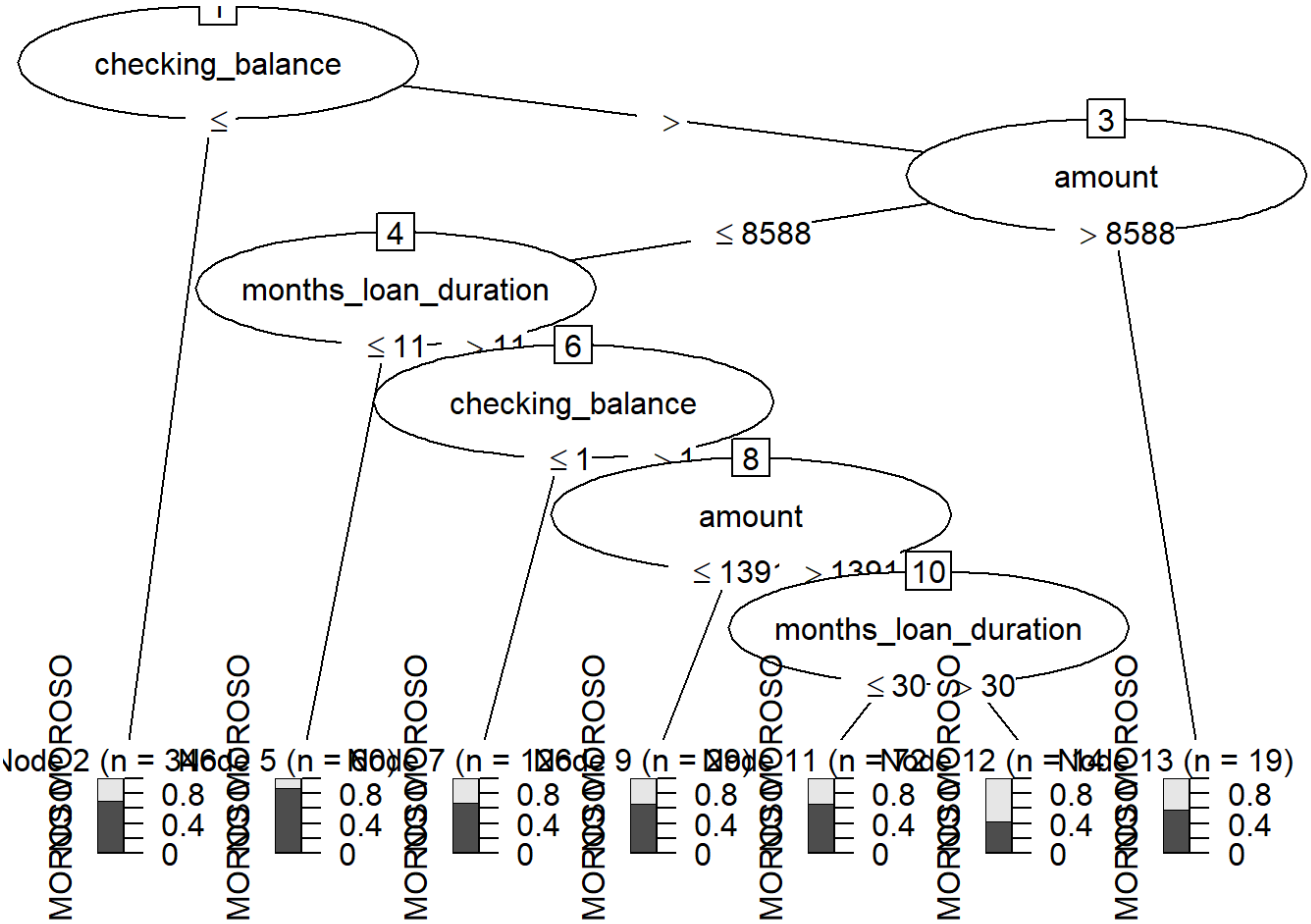
checking_balance = < 0 DM && months_loan_duration > 11 –> MOROSO. Validez: 54,9%

checking_balance entre {unknown, > 200 DM} –> NO MOROSO. Validez: 86,7%

amount <= 8588 –> NO MOROSO. Validez: 71%

A continuación, mostramos el árbol obtenido.

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



3.4 Análisis de la bondad de ajuste sobre el conjunto de test y matriz de confusión

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) / l
ength(predicted_model)))
```

```
## [1] "La precisión del árbol es: 73.0539 %"
```

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       MOROSO NO MOROSO
##   MOROSO        25         71
##   NO MOROSO     19        219
```

Para tener información más completa se usará el paquete gmodels.

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

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

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  334
##
##
##              | Prediction
##      Reality |    MOROSO | NO MOROSO | Row Total |
## -------------|-----------|-----------|-----------|
##       MOROSO |        25 |        71 |        96 |
##              |     0.075 |     0.213 |           |
## -------------|-----------|-----------|-----------|
##    NO MOROSO |        19 |       219 |       238 |
##              |     0.057 |     0.656 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        44 |       290 |       334 |
## -------------|-----------|-----------|-----------|
##
##
```

Como se puede observar, el árbol tiene una precisión de un poco mas del 73%, lo que está bastante bien. Lo que voy a comprobar ahora es la precisión del árbol con todas las variables, ya que se ha descartado la inmensa mayoría.

```
#Asignamos los datos
set.seed(1)
data_random_completos <- datos[sample(nrow(datos)),]

#Separamos los valores
set.seed(666)
y_completo <- data_random_completos[,17]
X_completo <- data_random_completos
X_completo[,17] <- NULL

#Separamos los campos
split_prop <- 3
max_split<-floor(nrow(X_completo)/split_prop)
tr_limit <- nrow(X_completo)-max_split
ts_limit <- nrow(X_completo)-max_split+1

trainX <- X_completo[1:tr_limit,]
trainy <- y_completo[1:tr_limit]
testX <- X_completo[(ts_limit+1):nrow(X_completo),]
testy <- y_completo[(ts_limit+1):nrow(X_completo)]

split_prop <- 3
indexes = sample(1:nrow(datos), size=floor(((split_prop-1)/split_prop)*nrow(datos)))
trainX<-X_completo[indexes,]
trainy<-y_completo[indexes]
testX<-X_completo[-indexes,]
testy<-y_completo[-indexes]

#Se crea el arbol de decisión
trainy = as.factor(trainy)
model <- C50::C5.0(trainX, trainy,rules=TRUE )

#Se obtiene la precision del arbol
predicted_model <- predict( model, testX, type="class" )
print(sprintf("La precisión del árbol con todos los campos es: %.4f %%",100*sum(predicte
d_model == testy) / length(predicted_model)))
```

```
## [1] "La precisión del árbol con todos los campos es: 73.6527 %"
```

La variación de precisión es mínima con todos los campos respecto al modelo con campos simplificados.

Se puede concluir que la capacidad de predicción del árbol es bastante buena, y que como se ha comprobado un análisis inicial de los campos, pueden ayudar a simplificar mucho la creación del árbol.

3.5 Modelos complementarios

Ahora se va a implementar un modelo complementario con el paquete rpart de R. Lo primero es crear nuestro conjunto de entrenamiento y de prueba (Ejemplo de arbol basado en:
https://rpubs.com/jboscomendoza/arboles_decision_clasificacion
(https://rpubs.com/jboscomendoza/arboles_decision_clasificacion))

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

```
#Separamos los campos
set.seed(666)
y <- data_random[,4]
X <- data_random
X[,4] <- NULL

#Separamos los valores
split_prop <- 3
max_split<-floor(nrow(X)/split_prop)
tr_limit <- nrow(X)-max_split
ts_limit <- nrow(X)-max_split+1

trainX <- X[1:tr_limit,]
trainy <- y[1:tr_limit]
testX <- X[(ts_limit+1):nrow(X),]
testy <- y[(ts_limit+1):nrow(X)]

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

Usamos la función rpart de rpart para entrenar nuestro modelo. Esta función nos pide una fórmula para especificar la variable objetivo de la clasificación. La fórmula que usaremos es tipo ~ ., la cual expresa que intentaremos clasificar tipo usando a todas las demás variables como predictoras.

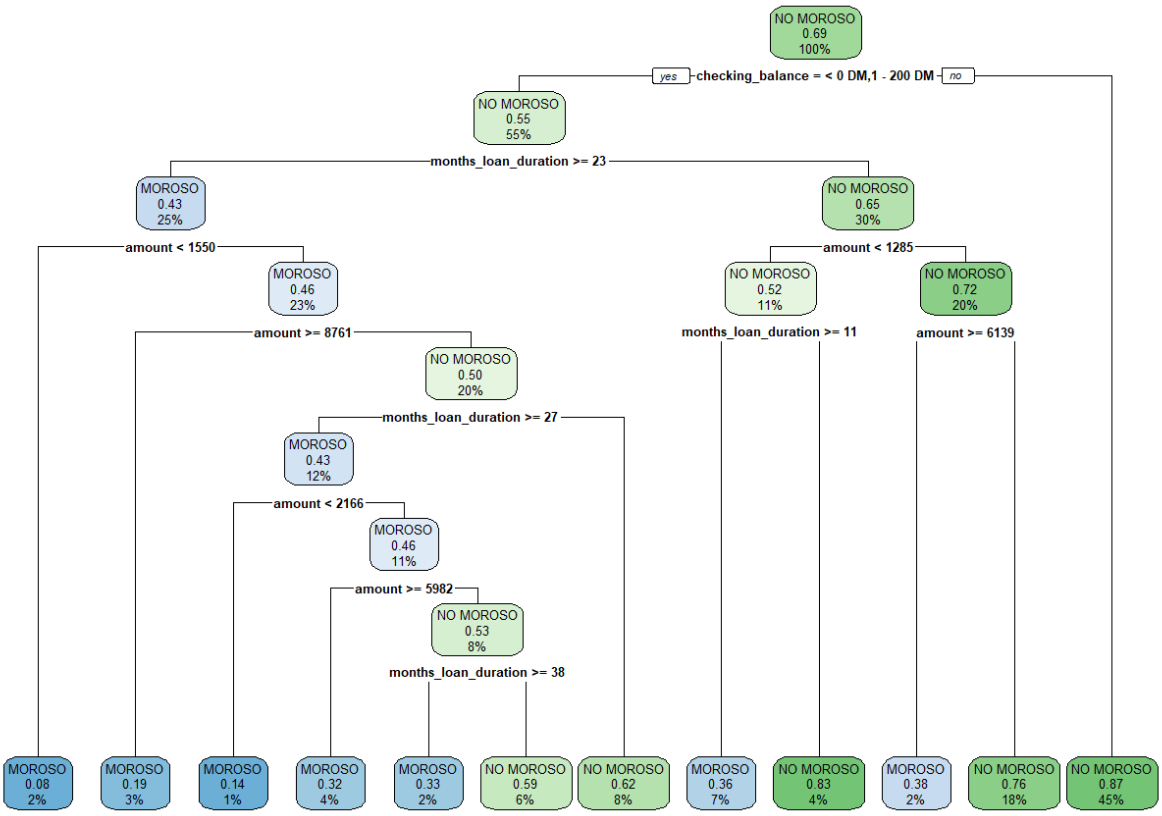
```
arbol <- rpart(formula = trainy ~ ., data = trainX)
arbol
```

```
## n= 666
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 666 204 NO MOROSO (0.30630631 0.69369369)
##     2) checking_balance=< 0 DM,1 - 200 DM 368 165 NO MOROSO (0.44836957 0.55163043)
##       4) months_loan_duration>=22.5 167  72 MOROSO (0.56886228 0.43113772)
##         8) amount< 1549.5 13   1 MOROSO (0.92307692 0.07692308) *
##         9) amount>=1549.5 154  71 MOROSO (0.53896104 0.46103896)
##          18) amount>=8760.5 21   4 MOROSO (0.80952381 0.19047619) *
##          19) amount< 8760.5 133  66 NO MOROSO (0.49624060 0.50375940)
##            38) months_loan_duration>=26.5 83  36 MOROSO (0.56626506 0.43373494)
##              76) amount< 2165.5 7   1 MOROSO (0.85714286 0.14285714) *
##              77) amount>=2165.5 76  35 MOROSO (0.53947368 0.46052632)
##               154) amount>=5981.5 25   8 MOROSO (0.68000000 0.32000000) *
##               155) amount< 5981.5 51  24 NO MOROSO (0.47058824 0.52941176)
##                 310) months_loan_duration>=37.5 12   4 MOROSO (0.66666667 0.33333333)
*
##                 311) months_loan_duration< 37.5 39  16 NO MOROSO (0.41025641 0.589743
59) *
##            39) months_loan_duration< 26.5 50  19 NO MOROSO (0.38000000 0.62000000) *
##       5) months_loan_duration< 22.5 201  70 NO MOROSO (0.34825871 0.65174129)
##        10) amount< 1285 71  34 NO MOROSO (0.47887324 0.52112676)
##          20) months_loan_duration>=11 47  17 MOROSO (0.63829787 0.36170213) *
##          21) months_loan_duration< 11 24   4 NO MOROSO (0.16666667 0.83333333) *
##        11) amount>=1285 130  36 NO MOROSO (0.27692308 0.72307692)
##          22) amount>=6138.5 13   5 MOROSO (0.61538462 0.38461538) *
##          23) amount< 6138.5 117  28 NO MOROSO (0.23931624 0.76068376) *
##     3) checking_balance=> 200 DM,unknown 298  39 NO MOROSO (0.13087248 0.86912752) *
```

Lo anterior muestra el esquema de nuestro árbol de clasificación. Cada inciso nos indica un nodo y la regla de clasificación que le corresponde. Siguiendo estos nodos, podemos llegar a las hojas del árbol, que corresponde a la clasificación de nuestros datos.

Todo lo anterior resulta mucho más claro si lo visualizamos.

```
rpart.plot(arbol)
```



Ahora generar un vector con los valores predichos por el modelo que hemos entrenado y mostramos la matriz de confusión.

```
prediccion <- predict(arbol, newdata = testX, type = "class")

confusionMatrix(prediccion, testy)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  MOROSO NO MOROSO
##   MOROSO        34        33
##   NO MOROSO     62       205
##
##                Accuracy : 0.7156
##                  95% CI : (0.6639, 0.7633)
##     No Information Rate : 0.7126
##     P-Value [Acc > NIR] : 0.479311
##
##                   Kappa : 0.2369
##
##  Mcnemar's Test P-Value : 0.004069
##
##             Sensitivity : 0.3542
##             Specificity : 0.8613
##          Pos Pred Value : 0.5075
##          Neg Pred Value : 0.7678
##              Prevalence : 0.2874
##          Detection Rate : 0.1018
##    Detection Prevalence : 0.2006
##       Balanced Accuracy : 0.6078
##
##        'Positive' Class : MOROSO
##
```

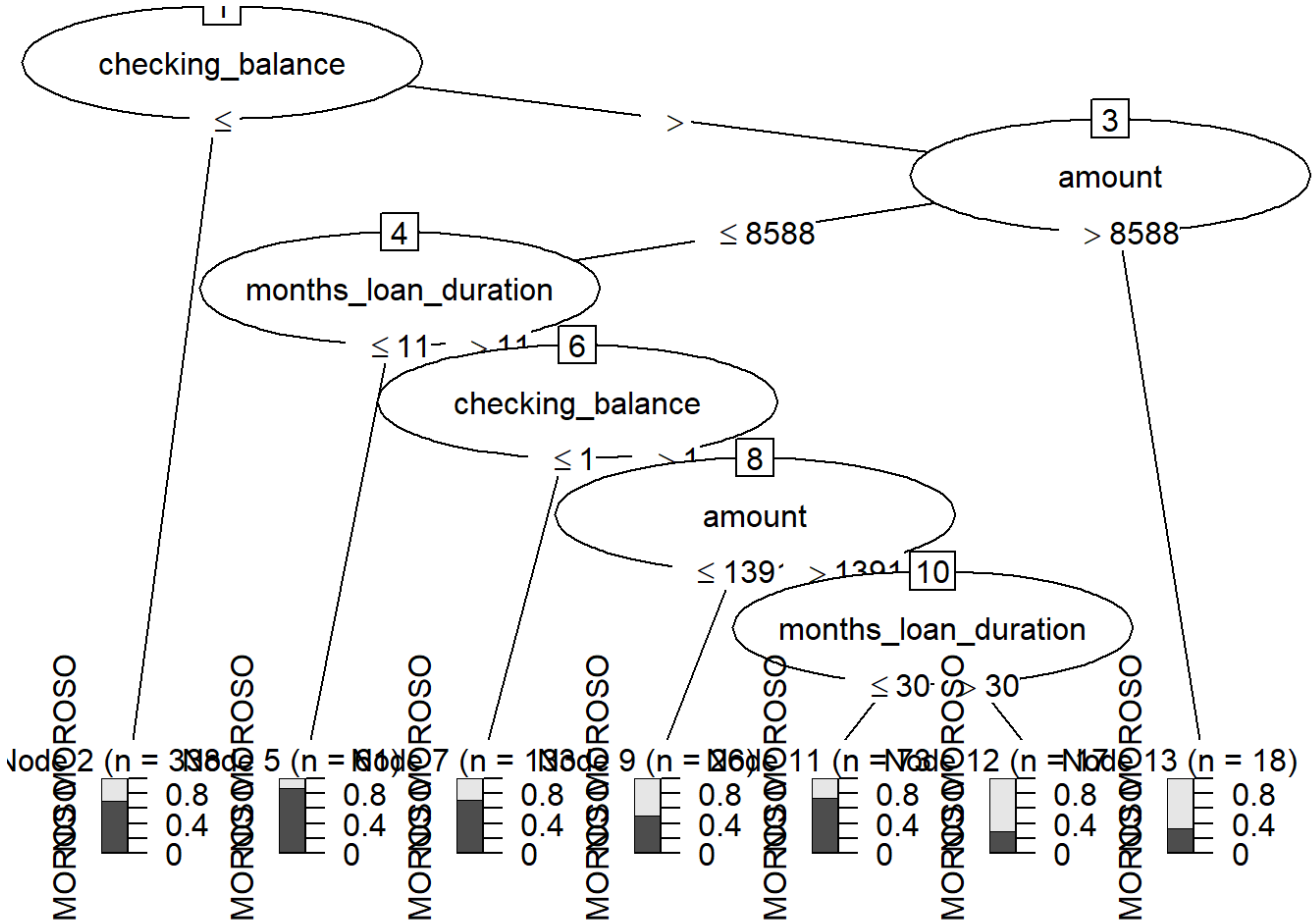
Y calculamos la precisión del árbol generado:

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

```
## [1] "La precisión del árbol es: 71.5569 %"
```

Se observa de manera clara, que las precisiones de los arboles generados no suele variar mucho. No obstante, se va a hacer otro ultimo árbol basado en el ejemplo del punto 2.5 del enunciado que incorpora como novedad “adaptative boosting”.

```
modelo2 <- C50::C5.0(trainX, trainy, trials = 10)
plot(modelo2)
```



Vemos a continuación cómo son las predicciones del nuevo árbol:

```
predicted_model2 <- predict( modelo2, testX, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model2 == testy) /
 length(predicted_model2)))
```

```
## [1] "La precisión del árbol es: 72.4551 %"
```

Observamos como se modifica levemente la precisión del modelo a peor.

3.6 Conclusiones obtenidas

Como conclusión de esta PEC3, me he dado cuenta la importancia de realizar un análisis previo de las variables y su relación con el propósito que queremos buscar, ya que no es lo mismo trabajar con 21 campos que con 5 (como ha ocurrido en este caso), y aunque el modelo de predicción con menos campos ha sido muy ligeramente inferior (menor que un 1%) la forma de comprender las reglas y visualizar el árbol ha sido mucho más fácil.

Respecto a los tres métodos de arboles (dos realmente ya que uno era con una variación), creo que pueden ser complementarios, ya que uno aporta un enfoque mas simples respecto a las reglas, y otro una visualización más clara.

---

4 Rúbrica

- (Obligatorio) Se debe realizar un breve informe (PDF, Html....) donde se respondan a las preguntas concretas, mostrando en primer lugar el código utilizado, luego los resultados y posteriormente los comentarios que se consideren pertinentes para cada apartado.
- 10% Hay un estudio sobre los datos de los que se parte, las variables que componen los datos. Los datos son preparados correctamente.
- 10% Se realiza un análisis descriptivo univariante (o análisis de relevancia) de algunas variables una vez se han tratado vs el target a nivel gráfico, comentando las que aparentemente son más interesantes. Análogamente se realiza un análisis de correlaciones.
- 20% Se aplica un árbol de decisión de forma correcta y se obtiene una estimación del error, mostrando gráficamente el árbol obtenido. La visualización debe ser comprensible y adecuada al problema a resolver.
- 15% Se explican las reglas que se obtienen en términos concretos del problema a resolver.
- 15% Se usa el modelo para predecir con muestras no usadas en el entrenamiento (holdout) y se obtiene una estimación del error. En base a la matriz de confusión, se comentan los tipos de errores y se valora de forma adecuada la capacidad predictiva del algoritmo.
- 15% Se prueba otro modelo de árbol o variantes diferentes del C50 y se comparan los resultados obtenidos, valorando si son mejores.
- 10% Con los resultados obtenidos anteriormente, se presentan unas conclusiones donde se expone un resumen de los diferentes modelos utilizados (al menos 3) así como el conocimiento adquirido tras el trabajo realizado y los descubrimientos más importantes realizados en el conjunto de datos.
- 5% Se presenta el código y es fácilmente reproducible.