Clasificación con árboles de decisión

Aucancela Brando

12/14/2021

Carga de archivo credit

```
data <- read.csv("credit.csv", header=TRUE)
# summary(data)
attach(data)
```

Exploracion de la base de datos

Primero obtenemos las dimensiones de la base de datos.

Exploración de la base de datos

Obtenemos que disponemos de 1000 registros o 1000 (filas) y 21 variables (columnas).

```
dim(data)
```

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

Verificamos la estructura del juego de datos principal.

```
str(data)
```

```
## '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 society 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 resident" "skilled em
```

Donde las columnas del datatest pertenecen a:

checking_balance - saldo que posee en la cuenta.

months_loan_duration - meses de duración del préstamo.

credit_history - historial crediticio

purpose - propósito del préstamo

amount - monto/valor del préstamo solicitado

savings_balance - saldo ahorrado

employment_length - duracion del empleo

installment_rate - tasa de pago / porcentaje de la cuota respecto del total del préstamos solicitado

personal_status - estado personal / estado civil

other_debtors - otros deudores además del prestatario del dinero

residence_history - historial de residencia

property - propiedad

age - edad

installment_plan - origen del préstamos, qué tipo de institución lo concede

housing - tipo de alojamiento

existing_credits - número de creditos que posee.

default - impago del credito donde 1 = pago correcto y 2 =impagos.

dependents - dependientes

telephone - posee o no teléfono de contacto

foreign_worker - trabajador extranjero.

job - empleo.

Como el enunciado del ejercicio plante que no es necesario discretizar el Dataset se omiten algunos procesos.

Cambiamos los valores de la columna default y dependents, donde el primer caso pase de 1 sea Pago OK y 2 sea Impago para que se lea como caracterires y no como datos enteros; en el segundo caso 1 será NO y 2 será SI

```
unique(data$default)
```

```
## [1] 1 2
```

```
data$default <- ifelse(data$default == 1, "Pago OK", "Impago")
data$dependents <- ifelse(data$dependents == 1, "NO", "YES")
```

Visualización

Instalacion de herramientas de visualización.

```
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)
}
```

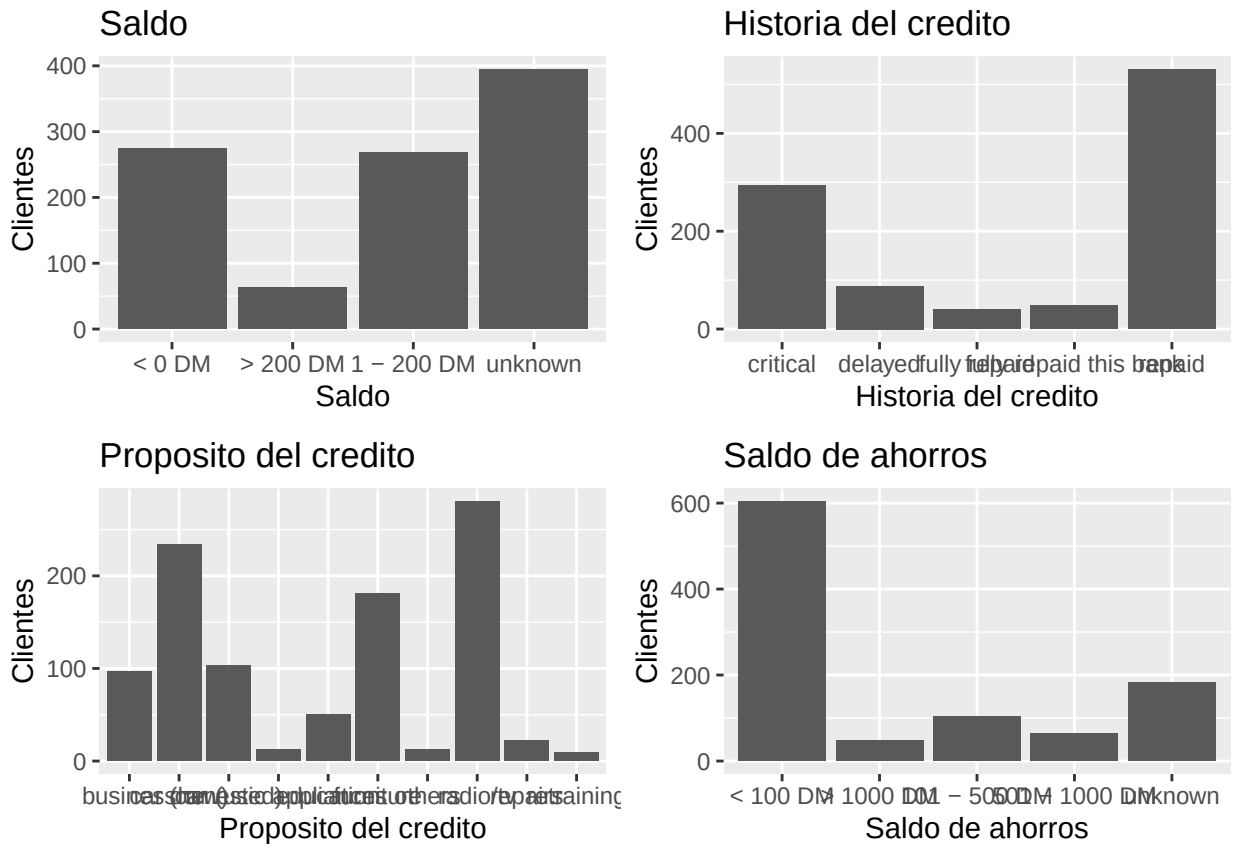
```
## Loading required package: gridExtra
```

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

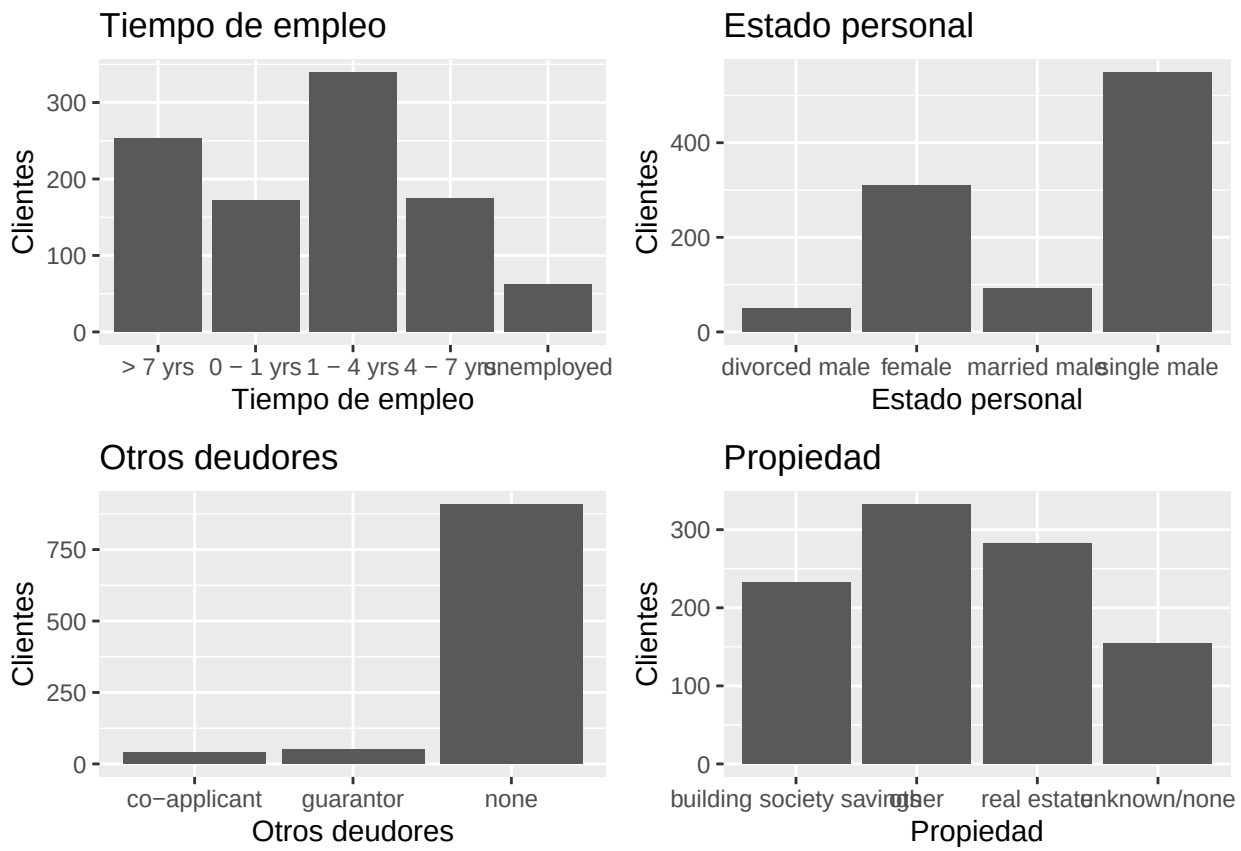
```
## Loading required package: C50
```

```
grid.newpage()
plotbyChecking<-ggplot(data,aes(checking_balance))+geom_bar() +labs(x="Saldo", y="Clientes")+ guides(fi

plotbyCreditHistory<-ggplot(data,aes(credit_history))+geom_bar() +labs(x="Historia del credito", y="Cli

plotbyPurpose<-ggplot(data,aes(purpose))+geom_bar() +labs(x="Proposito del credito", y="Clientes")+ gui

plotbySavingsBalance<-ggplot(data,aes(savings_balance))+geom_bar() +labs(x="Saldo de ahorros", y="Clien

plotbyEmploymentLength<-ggplot(data,aes(employment_length))+geom_bar() +labs(x="Tiempo de empleo", y="C
```

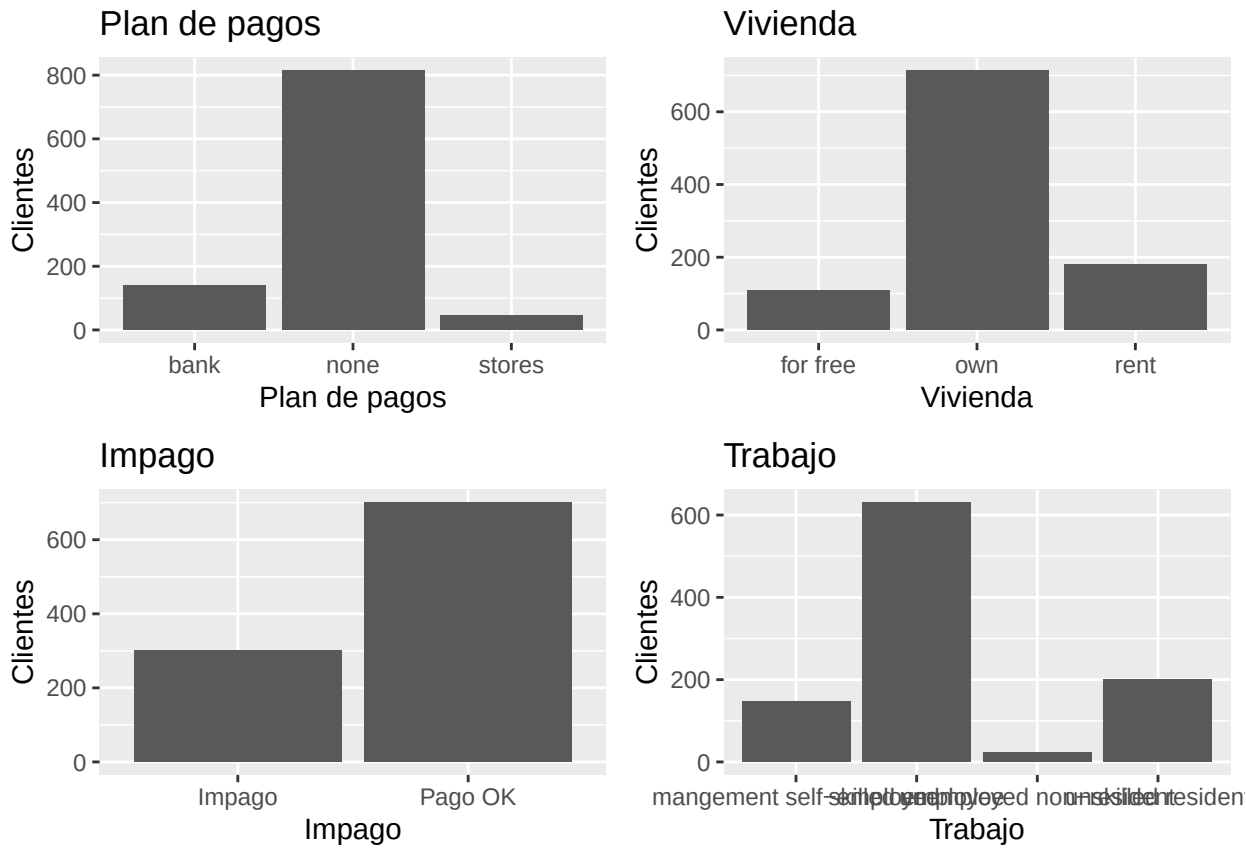
```
plotbyPersonalStatus<-ggplot(data,aes(personal_status))+geom_bar() +labs(x="Estado personal", y="Client
plotbyOtherDebtors<-ggplot(data,aes(other_debtors))+geom_bar() +labs(x="Otros deudores", y="Clientes")+
plotbyProperty<-ggplot(data,aes(property))+geom_bar() +labs(x="Propiedad", y="Clientes")+ guides(fill=gu
plotbyInstallmentPlan<-ggplot(data,aes(installment_plan))+geom_bar() +labs(x="Plan de pagos", y="Client
plotbyHousing<-ggplot(data,aes(housing))+geom_bar() +labs(x="Vivienda", y="Clientes")+ guides(fill=guid
plotbyDefault<-ggplot(data,aes(default))+geom_bar() +labs(x="Impago", y="Clientes")+ guides(fill=guide_l
plotbyJob<-ggplot(data,aes(job))+geom_bar() +labs(x="Trabajo", y="Clientes")+ guides(fill=guide_legend(
grid.arrange(plotbyChecking,plotbyCreditHistory,plotbyPurpose,plotbySavingsBalance, ncol=2, nrow=2)
```




```
grid.arrange(plotbyEmploymentLength,plotbyPersonalStatus,plotbyOtherDebtors,plotbyProperty, ncol=2, nrou
```



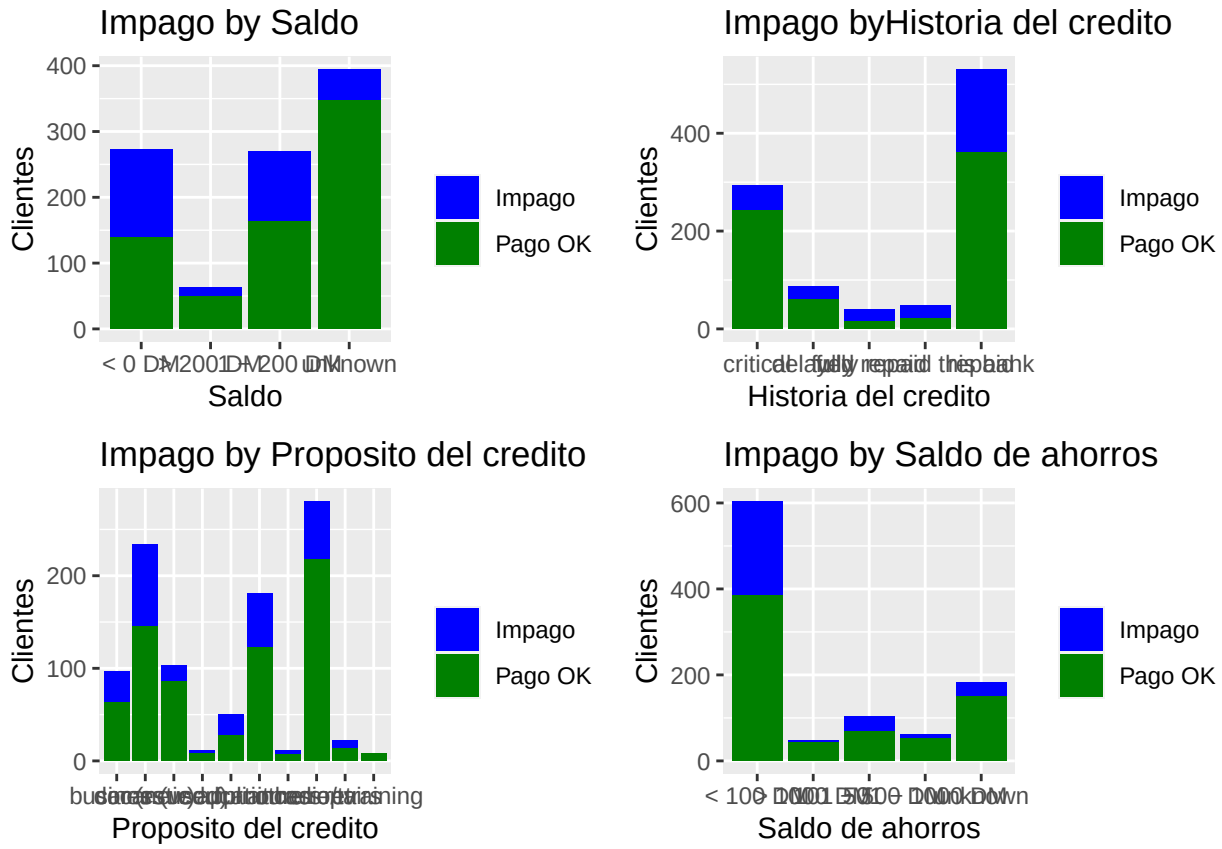

```
grid.arrange(plotbyInstallmentPlan,plotbyHousing,plotbyDefault,plotbyJob, ncol=2, nrow=2)
```



Como se puede observar existen cada grafico plantea una marcada diferencia entre toda la información disponible, pero lo que en realidad nos interesa es la relación de las variables respecto al impago del credito "default" es así que se debe realizar sus respectivas graficas de barras.

```
grid.newpage()
plotbyChecking<-ggplot(data,aes(checking_balance,fill=default))+geom_bar() +labs(x="Saldo", y="Clientes

plotbyCreditHistory<-ggplot(data,aes(credit_history,fill=default))+geom_bar() +labs(x="Historia del cre

plotbyPurpose<-ggplot(data,aes(purpose,fill=default))+geom_bar() +labs(x="Proposito del credito", y="Cl

plotbySavingsBalance<-ggplot(data,aes(savings_balance,fill=default))+geom_bar() +labs(x="Saldo de ahorr

plotbyEmploymentLength<-ggplot(data,aes(employment_length,fill=default))+geom_bar() +labs(x="Tiempo de

plotbyPersonalStatus<-ggplot(data,aes(personal_status,fill=default))+geom_bar() +labs(x="Estado persona

plotbyOtherDebtors<-ggplot(data,aes(other_debtors,fill=default))+geom_bar() +labs(x="Otros deudores", y

plotbyProperty<-ggplot(data,aes(property,fill=default))+geom_bar() +labs(x="Propiedad", y="Clientes")+

plotbyInstallmentPlan<-ggplot(data,aes(installment_plan,fill=default))+geom_bar() +labs(x="Plan de pago

plotbyHousing<-ggplot(data,aes(housing,fill=default))+geom_bar() +labs(x="Vivienda", y="Clientes")+ gui

plotbyForeignWorker<-ggplot(data,aes(foreign_worker,fill=default))+geom_bar() +labs(x="Trabajador extra
```

```
plotbyJob<-ggplot(data,aes(job,fill=default))+geom_bar() +labs(x="Trabajo", y="Clientes")+ guides(fill=g

grid.arrange(plotbyChecking,plotbyCreditHistory,plotbyPurpose,plotbySavingsBalance, ncol=2, nrow=2)
```



Listado de tablas de contingencia

```
tabla_DChT<-table(checking_balance, default)
tabla_DChT
```

```
##                 default
## checking_balance   1   2
##        < 0 DM     139 135
##        > 200 DM    49  14
##        1 - 200 DM 164 105
##        unknown    348  46
```

```
prop.table(tabla_DChT,margin =1)
```

```
##                 default
## checking_balance         1         2
##        < 0 DM     0.5072993 0.4927007
##        > 200 DM   0.7777778 0.2222222
##        1 - 200 DM 0.6096654 0.3903346
##        unknown    0.8832487 0.1167513
```

```
tabla_DCrT<-table(credit_history, default)
tabla_DCrT
```

```
##                          default
## credit_history             1   2
##   critical               243  50
##   delayed                 60  28
##   fully repaid            15  25
##   fully repaid this bank  21  28
##   repaid                 361 169
```

```
prop.table(tabla_DCrT,margin =1)
```

```
##                          default
## credit_history                  1         2
##   critical               0.8293515 0.1706485
##   delayed                0.6818182 0.3181818
##   fully repaid           0.3750000 0.6250000
##   fully repaid this bank 0.4285714 0.5714286
##   repaid                 0.6811321 0.3188679
```

```
tabla_DPT<-table(purpose, default)
tabla_DPT
```

```
##                      default
## purpose                1   2
##   business            63  34
##   car (new)          145  89
##   car (used)          86  17
##   domestic appliances  8   4
##   education           28  22
##   furniture          123  58
##   others               7   5
##   radio/tv           218  62
##   repairs             14   8
##   retraining           8   1
```

```
prop.table(tabla_DPT,margin =1)
```

```
##                      default
## purpose                      1         2
##   business           0.6494845 0.3505155
##   car (new)          0.6196581 0.3803419
##   car (used)         0.8349515 0.1650485
##   domestic appliances 0.6666667 0.3333333
##   education          0.5600000 0.4400000
##   furniture          0.6795580 0.3204420
##   others             0.5833333 0.4166667
##   radio/tv           0.7785714 0.2214286
##   repairs            0.6363636 0.3636364
##   retraining         0.8888889 0.1111111
```

```
tabla_DSaT<-table(savings_balance, default)
tabla_DSaT
```

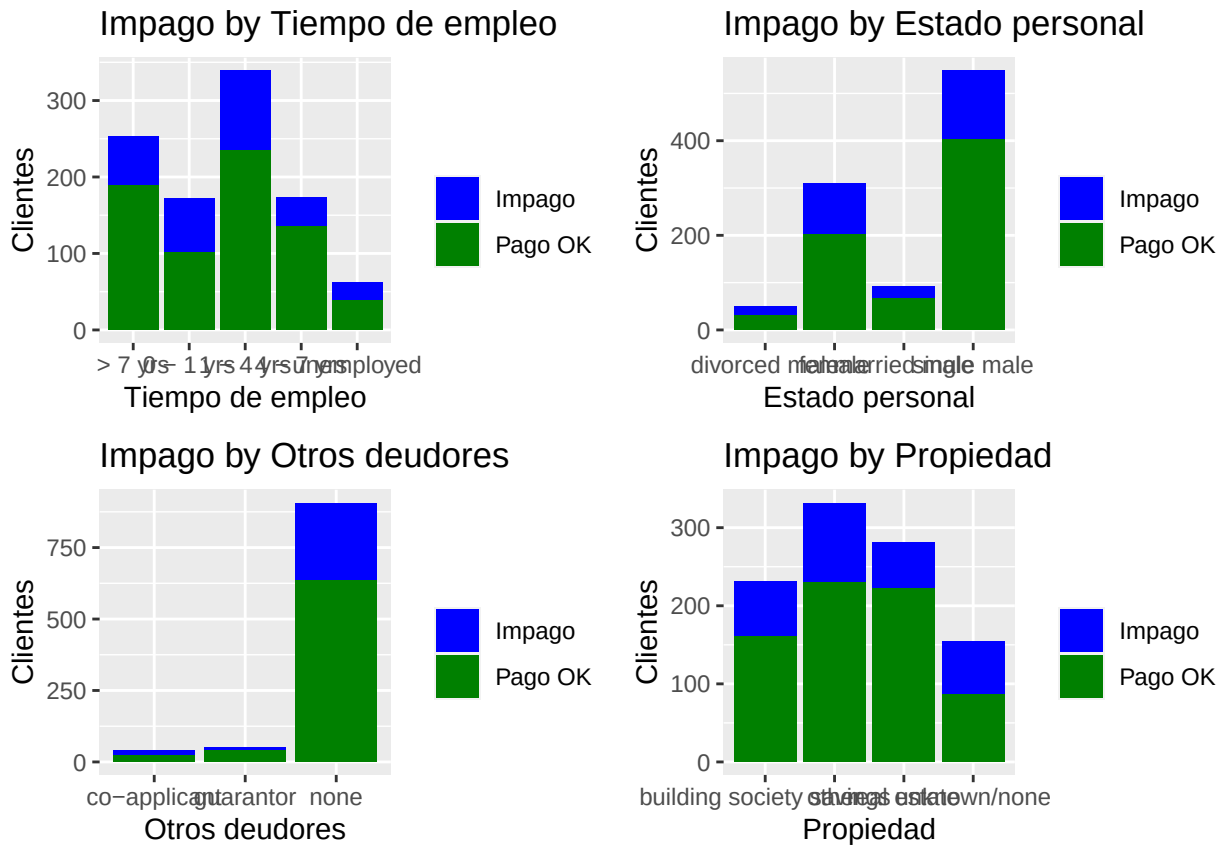
```
##                default
## savings_balance   1   2
##   < 100 DM      386 217
##   > 1000 DM      42   6
##   101 - 500 DM   69  34
##   501 - 1000 DM  52  11
##   unknown       151  32
```

```
prop.table(tabla_DSaT,margin =1)
```

```
##                default
## savings_balance         1         2
##   < 100 DM      0.6401327 0.3598673
##   > 1000 DM     0.8750000 0.1250000
##   101 - 500 DM  0.6699029 0.3300971
##   501 - 1000 DM 0.8253968 0.1746032
##   unknown       0.8251366 0.1748634
```

Del primer conjunto de graficas y sus respectivas tablas de contingtencia se tiene que en todos los casos de la cantidad de Pago OK (1) es mayor en todos los casos, es decir respecto a las variables Saldo, Historial crediticio, etc. En el caso del Saldo podemos observar que existen más clientes con su saldo desconocido mismos que presentan 348 (88.3%) clientes que han pagado el credito y tan solo 46 (11.7%) que no han pagado. En cuanto al historial crediticio se tiene un caso similar al de saldo donde los que tienen un historial crítico (critical) y pagado (repaid) son los que presentan más clientes 293 en el primer caso y 530 en el segundo. Para el proposito del credito se tiene que existen clientes con mayor predominancia siendo estos por motivo de radio/tv, inmoviliario (furniture) y car y los de menor predominancia son retraining, doméstico y otros. Esta variable cumple con que los Pagos_OK son mayores a los Impagos. Por último el saldo ahorrado se tiene que los que posee un saldo <100 DM tienen mayor predominacia donde 386 tiene un pago_OK (64 %) y 217 (36 %) Impago.

```
grid.arrange(plotbyEmploymentLength,plotbyPersonalStatus,plotbyOtherDebtors,plotbyProperty, ncol=2, nrow
```



Listado de tablas de contingencia

```
tabla_DEmT<-table(employment_length, default)
tabla_DEmT
```

```
##                   default
## employment_length   1   2
##        > 7 yrs    189  64
##        0 - 1 yrs  102  70
##        1 - 4 yrs  235 104
##        4 - 7 yrs  135  39
##        unemployed  39  23
```

```
prop.table(tabla_DEmT,margin =1)
```

```
##                   default
## employment_length         1         2
##        > 7 yrs    0.7470356 0.2529644
##        0 - 1 yrs  0.5930233 0.4069767
##        1 - 4 yrs  0.6932153 0.3067847
##        4 - 7 yrs  0.7758621 0.2241379
##        unemployed 0.6290323 0.3709677
```

```
tabla_DPeT<-table(personal_status, default)
tabla_DPeT
```

```
##                default
## personal_status   1   2
##   divorced male  30  20
##   female        201 109
##   married male   67  25
##   single male   402 146
```

```
prop.table(tabla_DPeT,margin =1)
```

```
##                default
## personal_status         1         2
##   divorced male 0.6000000 0.4000000
##   female        0.6483871 0.3516129
##   married male  0.7282609 0.2717391
##   single male   0.7335766 0.2664234
```

```
tabla_DOT<-table(other_debtors, default)
tabla_DOT
```

```
##               default
## other_debtors    1   2
##   co-applicant  23  18
##   guarantor     42  10
##   none         635 272
```

```
prop.table(tabla_DOT,margin =1)
```

```
##               default
## other_debtors          1         2
##   co-applicant 0.5609756 0.4390244
##   guarantor    0.8076923 0.1923077
##   none         0.7001103 0.2998897
```

```
tabla_DPrT<-table(property, default)
tabla_DPrT
```

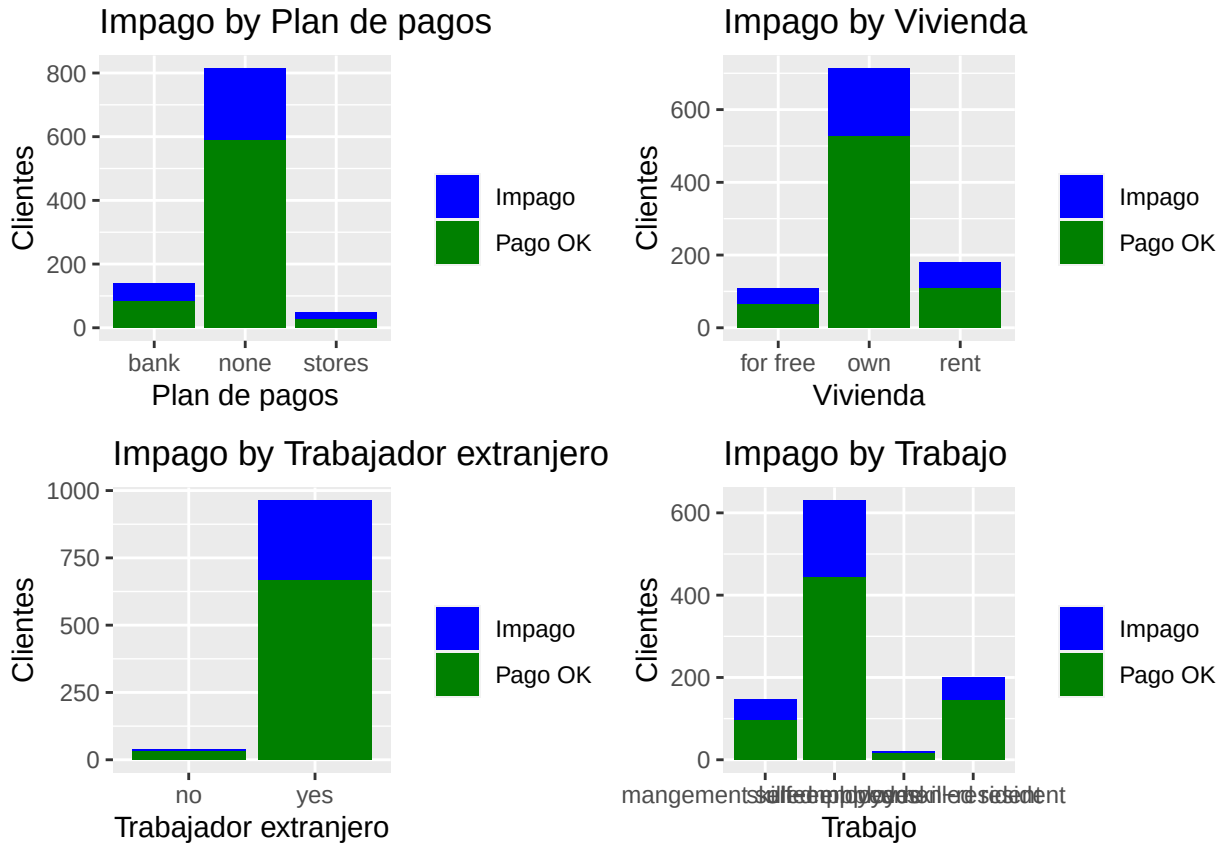
```
##                           default
## property                    1   2
##   building society savings 161  71
##   other                    230 102
##   real estate              222  60
##   unknown/none              87  67
```

```
prop.table(tabla_DPrT,margin =1)
```

```
##                           default
## property                          1         2
##   building society savings 0.6939655 0.3060345
##   other                    0.6927711 0.3072289
##   real estate              0.7872340 0.2127660
##   unknown/none             0.5649351 0.4350649
```

En estem conjunto de graficas y tablas se tiene que al igual que el anterio analisis presentan un mayor porcentaje los clientes los cuales tienen un pago_OK sobre los que tienen Impago. Podemos destacar que las personas que cuentan con trabajo realizan más creditos mientras que las que estan desempleadas realizaron tan solo 62 (pago_OK y Impago). Mientras que para el estado personal se tiene que un hombre soltero realiza más creditos con 548, es decir más de la mitad de creditos ya que el Dataset presenta 1000. y las mujeres se las agrupa en un solo conjunto sin importar su estado con un total de 201 (65 %) de pago_OK y 109 (35 %) de Impagos. Por último tenemos que los que no poseen un co-solicitante o garante ya que posee 907 clientes repartidos en 635 pago_OK y 272 Impagos. En cuanto a la propiedad tenemos que la distribucion es más homgenea ya que no extiste mucha variación de un tipo de propiedad a otro, con diferencia de la propiedad desconocida/ninguna.

```
grid.arrange(plotbyInstallmentPlan,plotbyHousing,plotbyForeignWorker,plotbyJob, ncol=2, nrow=2)
```



Listado de tablas de contingencia

```
tabla_DDiT<-table(installment_plan, default)
tabla_DDiT
```

```
##                 default
## installment_plan   1   2
##           bank    82  57
##           none   590 224
##           stores  28  19
```

```r
prop.table(tabla_DDiT,margin =1)
```

```
##                 default
## installment_plan         1         2
##           bank   0.5899281 0.4100719
##           none   0.7248157 0.2751843
##           stores 0.5957447 0.4042553
```

```r
tabla_DHuT<-table(housing, default)
tabla_DHuT
```

```
##           default
## housing      1   2
##   for free  64  44
##   own      527 186
##   rent     109  70
```

```r
prop.table(tabla_DHuT,margin =1)
```

```
##           default
## housing            1         2
##   for free 0.5925926 0.4074074
##   own      0.7391304 0.2608696
##   rent     0.6089385 0.3910615
```

```r
tabla_DFoT<-table(foreign_worker, default)
tabla_DFoT
```

```
##               default
## foreign_worker   1   2
##            no   33   4
##            yes 667 296
```

```r
prop.table(tabla_DFoT,margin =1)
```

```
##               default
## foreign_worker         1         2
##            no  0.8918919 0.1081081
##            yes 0.6926272 0.3073728
```

```r
tabla_DJoT<-table(job, default)
tabla_DJoT
```

```
##                           default
## job                          1   2
##   mangement self-employed  97  51
##   skilled employee        444 186
##   unemployed non-resident  15   7
##   unskilled resident      144  56
```

```
prop.table(tabla_DJoT,margin =1)
```

```
##                           default
## job                              1         2
##   mangement self-employed 0.6554054 0.3445946
##   skilled employee        0.7047619 0.2952381
##   unemployed non-resident 0.6818182 0.3181818
##   unskilled resident      0.7200000 0.2800000
```

En el último comjunto de graficas y tablas tenemos que los clientes cuya vivienda propia son los que realizan más creditos y su porcentaje de pago_OK es mayor en comparación de los for free o rent. En cuanto si el cliente es un trabajador extranjero o no tenemos que los que realizan más creditos son los extranjeros con 963 clientes. Para finalizar tenemos el tipo de trabajo realizado por el cliente donde los empleados calificados realizan un mayor número de creditos y los desempleados no residentes son los que realizan menos creditos con 630 y 22 respectivamente. Tal como se mensionó previamente se tiene que el porcentaje de Pago_OK prodomina en comparación a los Impago en todo el análsis de variables.

Test estadísticos de significancia

Procedemos a determinar el grado de significancia de sla relación previamente planteada.

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

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

```
Phi(tabla_DChT)
```

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

```
CramerV(tabla_DChT)
```

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

```
Phi(tabla_DCrT)
```

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

```
CramerV(tabla_DCrT)
```

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

```
Phi(tabla_DPT)
```

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

```
CramerV(tabla_DPT)
```

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

```
Phi(tabla_DSaT)
```

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

```
CramerV(tabla_DSaT)
```

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

```
Phi(tabla_DEmT)
```

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

```
CramerV(tabla_DEmT)
```

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

```
Phi(tabla_DPeT)
```

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

```
CramerV(tabla_DPeT)
```

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

```
Phi(tabla_DOT)
```

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

```
CramerV(tabla_DOT)
```

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

```
Phi(tabla_DPrT)
```

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

```
CramerV(tabla_DPrT)
```

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

```
Phi(tabla_DDiT)
```

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

```
CramerV(tabla_DDiT)
```

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

```
Phi(tabla_DHuT)
```

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

```
CramerV(tabla_DHuT)
```

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

```
Phi(tabla_DFoT)
```

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

```
CramerV(tabla_DFoT)
```

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

```
Phi(tabla_DJoT)
```

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

```
CramerV(tabla_DJoT)
```

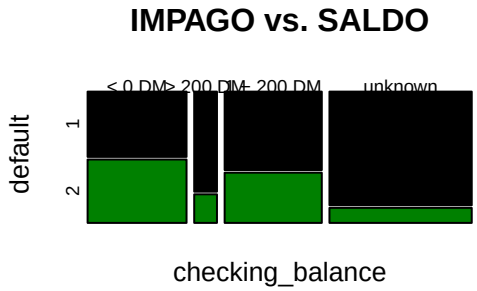
```
## [1] 0.04341838
```

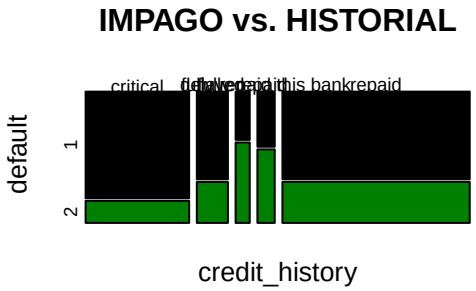
Segun la guia proporsionada tenemos que los valores de Cramér y Phi entre 0.1 y 0.3 nos indican que la asociación estadística es baja, y entre 0.3 y 0.5 es una asociación media. Finalmente los valores > 0.5 , la asociación estadística entre las variables sería alta.

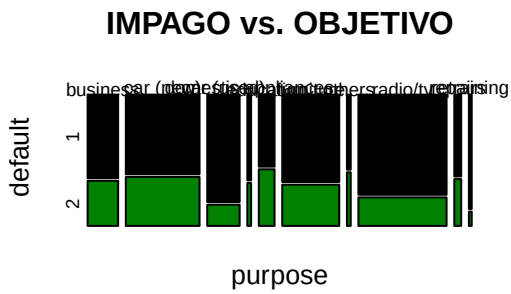
Como se puede ver, los valores de Phi y Crámer son identicos, debido a que se tratan de tablas 2x2.

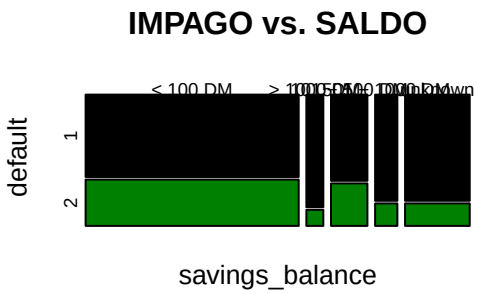
Plot de las tablas de contingencia para visualizar lor receptores de mejor manera.

```
par(mfrow=c(2,2))
plot(tabla_DChT, col = c("black","#008000"), main = "IMPAGO vs. SALDO")
plot(tabla_DCrT, col = c("black","#008000"), main = "IMPAGO vs. HISTORIAL")
plot(tabla_DPT, col = c("black","#008000"), main = "IMPAGO vs. OBJETIVO")
plot(tabla_DSaT, col = c("black","#008000"), main = "IMPAGO vs. SALDO")
```

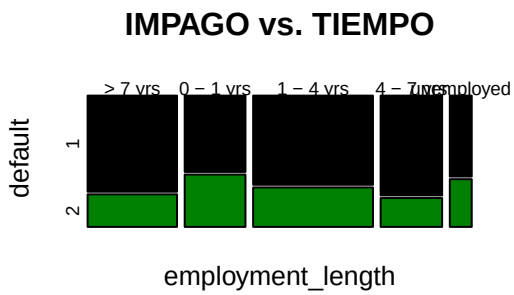


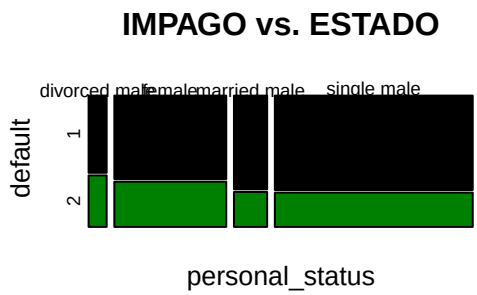


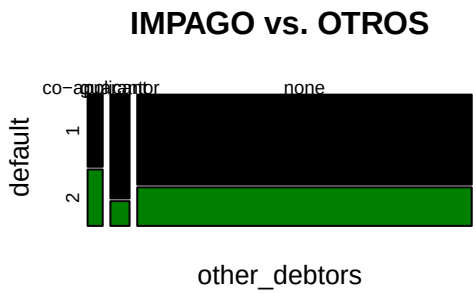


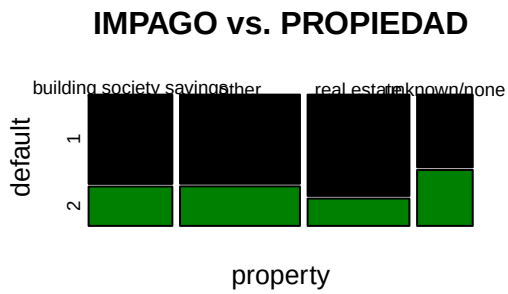



```
plot(tabla_DEmT, col = c("black","#008000"), main = "IMPAGO vs. TIEMPO")
plot(tabla_DPeT, col = c("black","#008000"), main = "IMPAGO vs. ESTADO")
plot(tabla_DOT, col = c("black","#008000"), main = "IMPAGO vs. OTROS")
plot(tabla_DPrT, col = c("black","#008000"), main = "IMPAGO vs. PROPIEDAD")
```

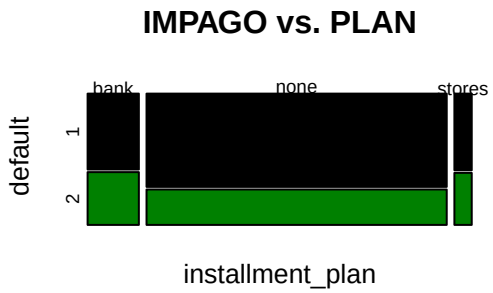


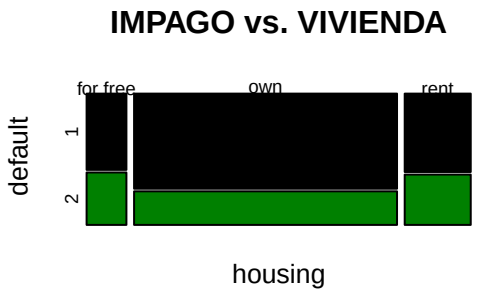


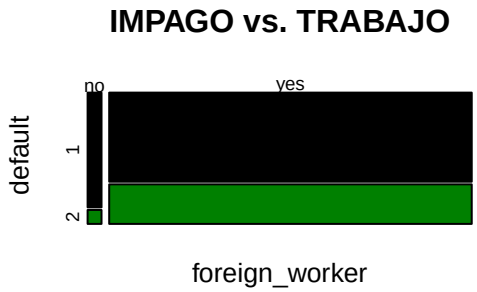





```
plot(tabla_DDiT, col = c("black","#008000"), main = "IMPAGO vs. PLAN")
plot(tabla_DHuT, col = c("black","#008000"), main = "IMPAGO vs. VIVIENDA")
plot(tabla_DFoT, col = c("black","#008000"), main = "IMPAGO vs. TRABAJO")
```







Se puede obtener las mismas asunciones que las graficas de barras anteriores ya que estan basadas en los mismos datos.

A pesar que el Dataset aparentemente está desordenado procedemos a desordenarlo de forma aleatoria al Dataset para ppsterior realización del Random forest.

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

MODELO 1

Preparación de los datos para el modelo

Procedemos a dividir el Dataset en conjunto de Entrenamiento o Train y Prueba o Test donde el primero tendrá 2/3 de todo los datos y 1/3 corresponderá para el conjunto de prueba para la evaluacion del modelo. Para las variables independientes o X podemos usar los valores de Cramér y Phi para determinar las variables que tienen m[as correlación con el target “default”

```
set.seed(666)
y<-data_random[,"default"]
X<-data_random[,c("checking_balance", "credit_history", "purpose", "savings_balance","employment_length
```

Separamos los datos en forma dinnámica en funcion de split_prop

```
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)]
```

Tambien se puede crear directamente un rango usando split_plot.

```
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]
```

Análisis de datos minimos para asegurarse de no obtener clasificadores sesgados.

```
summary(trainX)
```

```
##  checking_balance   credit_history       purpose          savings_balance
##  Length:666         Length:666         Length:666         Length:666
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  employment_length    property           housing
##  Length:666         Length:666         Length:666
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
summary(trainy)
```

```
##    Length     Class      Mode
##       666 character character
```

```
summary(testX)
```

```
##  checking_balance   credit_history       purpose          savings_balance
##  Length:334         Length:334         Length:334         Length:334
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  employment_length    property           housing
##  Length:334         Length:334         Length:334
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
summary(testy)
```

```
##    Length     Class      Mode
##       334 character character
```

Se verifica que no hay diferencias graves que puedan sesgar las conclusiones.

Creación del modelo, calidad del modelo y extracción de reglas

Se crea el árbol de decisión usando 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]      Mon May 23 13:00:04 2022
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 666 cases (8 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (6, lift 2.9)
##  checking_balance in {1 - 200 DM, < 0 DM}
##  purpose = radio/tv
##  employment_length = 1 - 4 yrs
##  property in {other, building society savings, unknown/none}
##  housing in {rent, for free}
##  ->  class Impago  [0.875]
##
## Rule 2: (17/5, lift 2.2)
##  checking_balance in {1 - 200 DM, < 0 DM}
##  purpose = radio/tv
##  employment_length in {unemployed, 0 - 1 yrs}
##  property in {other, building society savings, unknown/none}
##  ->  class Impago  [0.684]
##
## Rule 3: (49/16, lift 2.2)
##  checking_balance in {1 - 200 DM, < 0 DM}
##  credit_history = repaid
##  purpose in {car (new), others, repairs, domestic appliances, retraining}
##  property in {other, building society savings, unknown/none}
##  ->  class Impago  [0.667]
##
## Rule 4: (13/5, lift 2.0)
##  checking_balance = 1 - 200 DM
##  credit_history = repaid
##  purpose = furniture
##  property in {other, building society savings, unknown/none}
##  ->  class Impago  [0.600]
##
## Rule 5: (268/131, lift 1.7)
##  checking_balance in {1 - 200 DM, < 0 DM}
##  property in {other, building society savings, unknown/none}
```

```
##  ->  class Impago  [0.511]
##
## Rule 6: (298/39, lift 1.2)
##  checking_balance in {unknown, > 200 DM}
##  ->  class Pago OK  [0.867]
##
## Rule 7: (181/28, lift 1.2)
##  credit_history = critical
##  ->  class Pago OK  [0.842]
##
## Rule 8: (74/12, lift 1.2)
##  purpose = car (used)
##  ->  class Pago OK  [0.829]
##
## Rule 9: (195/41, lift 1.1)
##  property = real estate
##  ->  class Pago OK  [0.787]
##
## Rule 10: (373/122, lift 1.0)
##  credit_history = repaid
##  ->  class Pago OK  [0.672]
##
## Default class: Pago OK
##
##
## Evaluation on training data (666 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      10  156(23.4%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##      93   111    (a): class Impago
##      45   417    (b): class Pago OK
##
##
##  Attribute usage:
##
##   84.98% checking_balance
##   83.18% credit_history
##   69.52% property
##   23.87% purpose
##    3.45% employment_length
##    0.90% housing
##
##
## Time: 0.0 secs
```

Errors muestra el número y porcentaje de casos mal clasificados en el subconjunto de entrenamiento. El árbol obtenido clasifica erróneamente 156 de los 666 casos dados, una tasa de error del 23.4 %.

Mediante las reglas podemos asumir que: - checking_balance = 1 - 200 DM ó < 0 DM y purpose = radio/tv y employment_length = 1 - 4 yrs property = other, building society savings, unknown/none y housing = rent, for free → Impago. Validéz: 87.5%

- checking_balance = 1 - 200 DM ó < 0 DM y purpose = radio/tv y employment_length = unemployed ó 0 - 1 yrs y propiedad = property other, building society savings ó unknown/none → Impago. Validéz: 68.5%
- checking_balance = 1 - 200 DM ó < 0 DM y credit_history = repaid y purpose = car (new), others, repairs, domestic appliances ó retraining y property = other y building = society savings ó unknown/none → Impago. Validéz: 67.7%
- checking_balance = 1 - 200 DM y credit_history = repaid y purpose = furniture y property = other, building society savings ó unknown/none -> Impago. Validéz: 60%.
- checking_balance = 1 - 200 DM ó < 0 DM y property = other, building society savings ó unknown/none -> Impago. Validéz: 51.1%.
- checking_balance = unknown ó > 200 DM -> Pago OK. Validéz: 86.7%
- credit_history = critical -> Pago OK. Validéz: 84.2%
- purpose = car (used) -> Pago OK. Validéz = 82.9%
- property = real estate -> Pago OK. Validéz = 78.7%.
- credit_history = repaid -> Pago OK. Validéz: 67.2%.

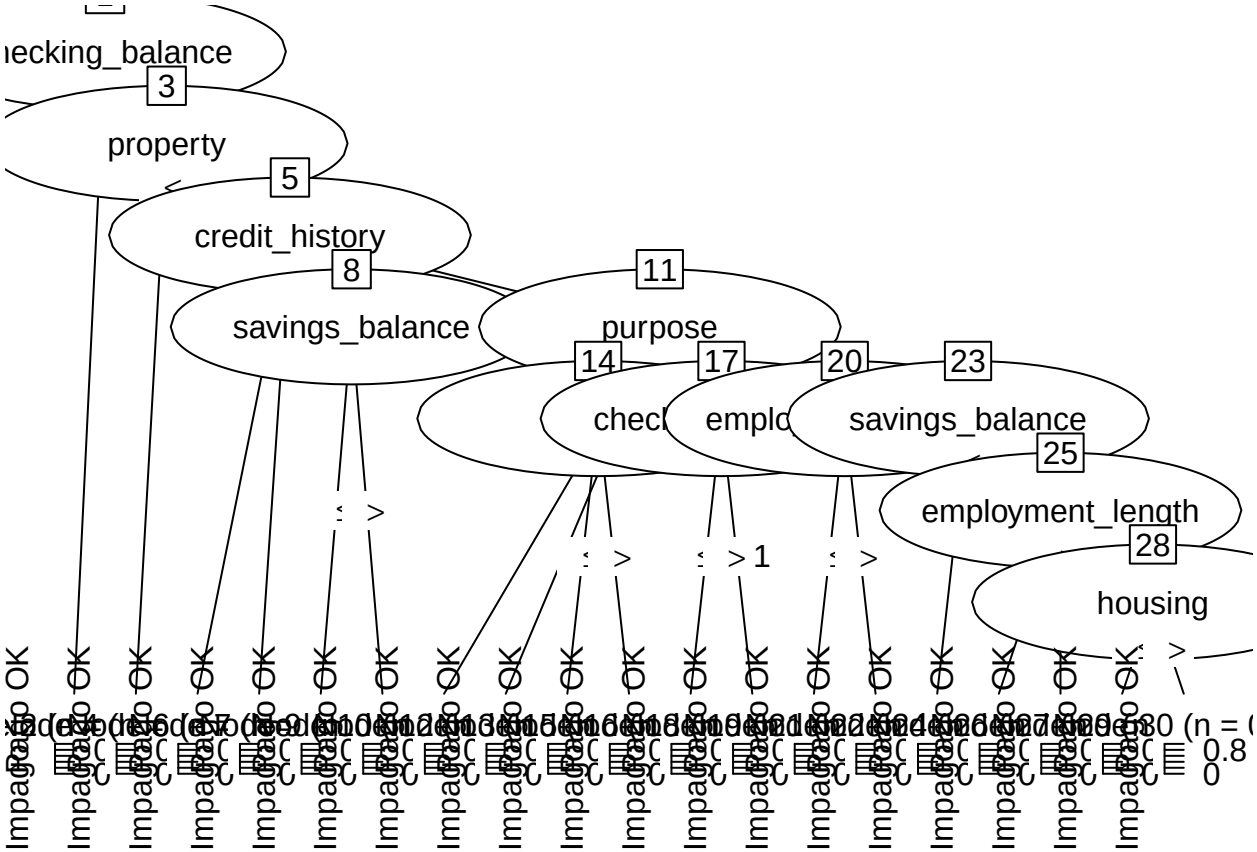
Por tanto, podemos concluir que el conocimiento extraído y cruzado con el análisis visual se resume en si se presentan un conjunto de caracteristicas desfavorables ese cliente tendrá un Impago tal es el caso de checking_balance < 200 DM o que esta desempleado o mientras menor sea el tiempo de empleo tendrá más probabilidades de no cumplir con el pago. Esto ratifica la logica que mientras mayor solvencia y estabilidad económica de una persona puede tener un pago correcto del credito, es decir si posee vivienda propia, si sus ahorros son elevados, si posee un empleo de muchos años, etc.

A continuación el árbol obtenido

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

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
```

```
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion


## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



Validación del modelo con los datos reservados

Comprobar la calidad del modelo propuesto a través de el conjunto de datos de prueba.

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

```
## [1] "La precisión del árbol es: 74.5509 %"
```

Análisis de la presición a través de la matriz de confusión que identifica los tipos de errores cometidos.

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

```
##          Predicted
## testy     Impago Pago OK
##   Impago      39      57
##   Pago OK     28     210
```

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_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 74.5509 %"
```

A través del 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',
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
```

```
## Total Observations in Table:  334
##
##
##              | Prediction
##      Reality |    Impago |   Pago OK | Row Total |
## -------------|-----------|-----------|-----------|
##       Impago |        39 |        57 |        96 |
##              |     0.117 |     0.171 |           |
## -------------|-----------|-----------|-----------|
##      Pago OK |        28 |       210 |       238 |
##              |     0.084 |     0.629 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        67 |       267 |       334 |
## -------------|-----------|-----------|-----------|
##
##
```

Prueba con una variación u otro enfoque algorítmico

Incorporación de "adaptative boosting" para 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)
```

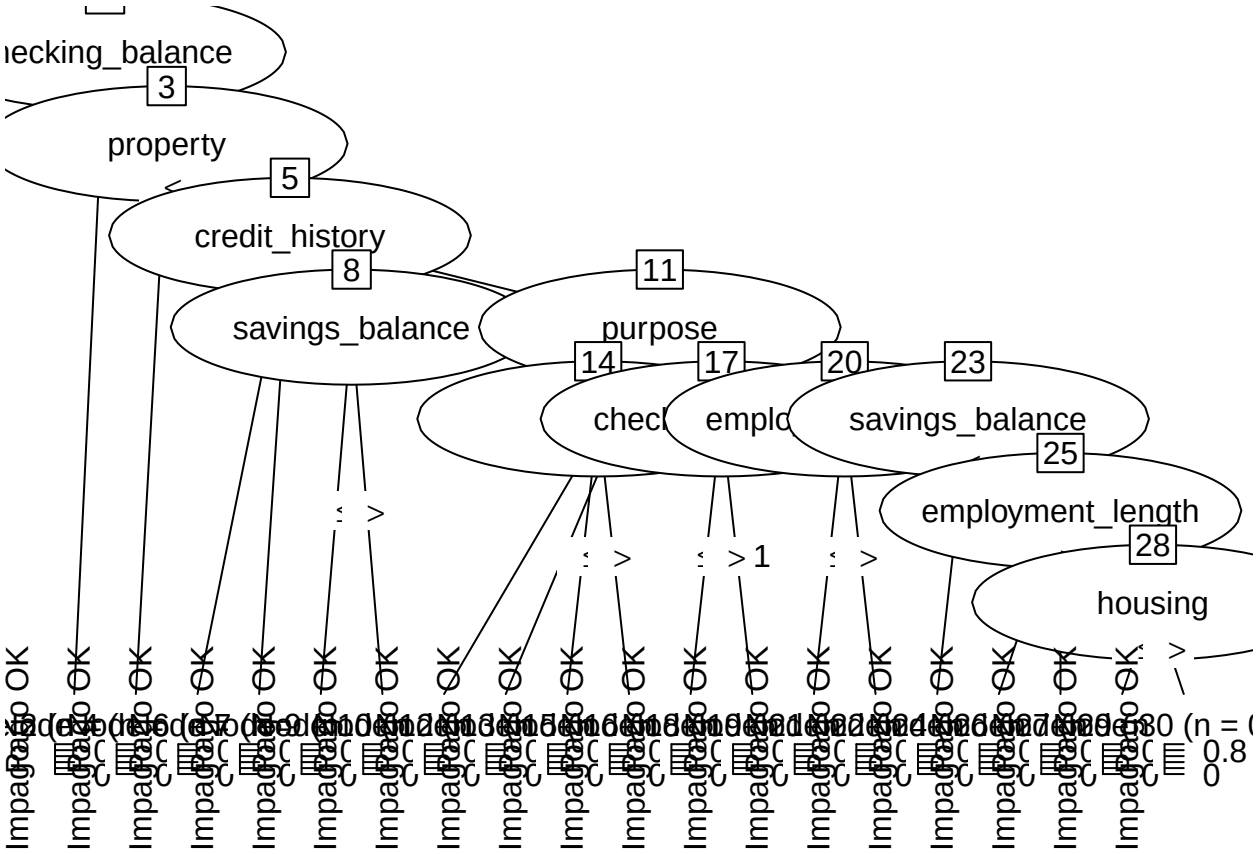
```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion
```

```
## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```

```
## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```

```
## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



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(predicte
```

```
## [1] "La precisión del árbol es: 76.0479 %"
```

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

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

```
##          Predicted
## testy     Impago Pago OK
##   Impago      38      58
##   Pago OK     22     216
```

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_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 76.0479 %"
```

El algoritmo C5.0 incorpora algunas opciones para ver la importancia de las variables.

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

```
##                   Overall
## checking_balance   100.00
## savings_balance    100.00
## purpose             99.40
## credit_history      88.44
## property            66.37
## employment_length   41.14
## housing             12.76
```

```
importancia_splits
```

```
##                     Overall
## checking_balance  32.727273
## savings_balance   20.000000
## credit_history    12.727273
## property          12.727273
## purpose            9.090909
## employment_length  7.272727
## housing            5.454545
```

Como se puede observar los resulatados tanto de importancia_usage yu de importancia_splits se da de forma decreciente señalando así que las primeras variables van a ser las que tengan mayor peso o importancia en nuestro modelo

MODELO 2

Preparación de los datos para el modelo

Procedemos a dividir el Dataset en conjunto de Entrenamiento o Train y Prueba o Test donde el primero tendrá 4/5 de todo los datos y 1/5 corresponderá para el conjunto de prueba para la evaluacion del modelo. Para las variables independientes o X podemos usar los valores de Cramér y Phi para determinar las variables que tienen m[as correlación con el target “default”

```
set.seed(800)
y<-data_random[,"default"]
X<-data_random[,c("checking_balance", "credit_history", "purpose", "savings_balance","employment_length
```

Tambien se puede crear directamente un rango usando split_plot.

```
split_prop<-5
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]
```

```
str(X)
```

```
## 'data.frame':    1000 obs. of  5 variables:
##  $ checking_balance : chr  "< 0 DM" "< 0 DM" "1 - 200 DM" "< 0 DM" ...
##  $ credit_history   : chr  "fully repaid" "repaid" "critical" "delayed" ...
##  $ purpose          : chr  "car (new)" "radio/tv" "car (used)" "car (new)" ...
##  $ savings_balance  : chr  "< 100 DM" "< 100 DM" "< 100 DM" "< 100 DM" ...
##  $ employment_length: chr  "1 - 4 yrs" "> 7 yrs" "unemployed" "1 - 4 yrs" ...
```

Análisis de datos minimos para asegurarse de no obtener clasificadores sesgados.

```
summary(trainX)
```

```
##  checking_balance   credit_history       purpose          savings_balance
##  Length:800         Length:800         Length:800         Length:800
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  employment_length
##  Length:800
##  Class :character
##  Mode  :character
```

```
summary(trainy)
```

```
##    Length     Class      Mode
##       800 character character
```

```
summary(testX)
```

```
##  checking_balance   credit_history       purpose          savings_balance
##  Length:200         Length:200         Length:200         Length:200
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  employment_length
##  Length:200
##  Class :character
##  Mode  :character
```

```
summary(testy)
```

```
##    Length     Class      Mode
##       200 character character
```

Se verifica que no hay diferencias graves que puedan sesgar las conclusiones.

Creación del modelo, calidad del modelo y extracción de reglas

Se crea el árbol de decisión usando 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]      Mon May 23 13:00:06 2022
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 800 cases (6 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (4, lift 2.7)
##  checking_balance = < 0 DM
##  credit_history = repaid
##  purpose = furniture
##  savings_balance = < 100 DM
##  employment_length = > 7 yrs
##  ->  class Impago  [0.833]
##
## Rule 2: (3, lift 2.6)
##  checking_balance = 1 - 200 DM
##  purpose = furniture
##  savings_balance = < 100 DM
##  employment_length = 1 - 4 yrs
##  ->  class Impago  [0.800]
##
## Rule 3: (17/3, lift 2.5)
##  checking_balance in {< 0 DM, 1 - 200 DM}
##  purpose in {radio/tv, furniture, domestic appliances, education}
##  savings_balance = 101 - 500 DM
##  ->  class Impago  [0.789]
##
## Rule 4: (55/15, lift 2.3)
```

```
##  checking_balance in {< 0 DM, 1 - 200 DM}
##  credit_history in {fully repaid this bank, fully repaid}
##  ->  class Impago  [0.719]
##
## Rule 5: (46/14, lift 2.2)
##  checking_balance in {< 0 DM, 1 - 200 DM}
##  credit_history in {repaid, delayed}
##  purpose = car (new)
##  savings_balance = < 100 DM
##  ->  class Impago  [0.688]
##
## Rule 6: (7/2, lift 2.2)
##  checking_balance = 1 - 200 DM
##  credit_history = critical
##  purpose = car (new)
##  savings_balance = < 100 DM
##  ->  class Impago  [0.667]
##
## Rule 7: (78/30, lift 2.0)
##  checking_balance in {< 0 DM, 1 - 200 DM}
##  purpose = car (new)
##  savings_balance = < 100 DM
##  ->  class Impago  [0.613]
##
## Rule 8: (358/46, lift 1.3)
##  checking_balance in {unknown, > 200 DM}
##  ->  class Pago OK  [0.869]
##
## Rule 9: (39/6, lift 1.2)
##  checking_balance = 1 - 200 DM
##  savings_balance = unknown
##  ->  class Pago OK  [0.829]
##
## Rule 10: (726/203, lift 1.0)
##  credit_history in {critical, repaid, delayed}
##  ->  class Pago OK  [0.720]
##
## Default class: Pago OK
##
##
## Evaluation on training data (800 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      10  181(22.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##      98   150    (a): class Impago
##      31   521    (b): class Pago OK
##
```

```
##
##  Attribute usage:
##
##   97.63% credit_history
##   68.25% checking_balance
##   17.63% savings_balance
##   12.75% purpose
##    0.88% employment_length
##
##
## Time: 0.0 secs
```

Errors muestra el número y porcentaje de casos mal clasificados en el subconjunto de entrenamiento. El árbol obtenido clasifica erróneamente 181 de los 800 casos dados, una tasa de error del 22.6 %.

Mediante las reglas podemos asumir que: - checking_balance = < 0 DM y credit_history = repaid y purpose = furniture y savings_balance = < 100 DM y employment_length = > 7 yrs -> Impago. Validéz: 83.3%

- checking_balance = 1 - 200 DM y purpose = furniture y savings_balance = < 100 DM y employment_length = 1 - 4 yrs -> Impago. Validéz: 80%
- checking_balance = < 0 DM ó 1 - 200 DM y purpose = radio/tv, furniture, domestic appliances, ó education y savings_balance = 101 - 500 DM y -> Impago. Validéz: 78.9%
- checking_balance in < 0 DM ó 1 - 200 DM y credit_history = fully repaid this bank, fully ó repaid -> Impago. Validéz: 71.9%
- checking_balance = < 0 DM ó 1 - 200 DM y credit_history = repaid ó delayed, purpose = car (new) y savings_balance = < 100 DM -> Impago. Validéz:68.8%.
- checking_balance = 1 - 200 DM, credit_history = critical y purpose = car (new) y savings_balance = < 100 DM -> Impago. Validéz: 66.7%
- checking_balance = < 0 DM ó 1 - 200 DM y purpose = car (new) y savings_balance = < 100 DM -> Impago. Validéz:61.3%
- checking_balance in {unknown ó > 200 DM} -> Pago OK. Validéz: 86.9%
- checking_balance = 1 - 200 DM y savings_balance = unknown -> Pago OK Validéz: 82.9%
- credit_history = critical, repaid ó delayed ->Pago OK. Validéz: 72%

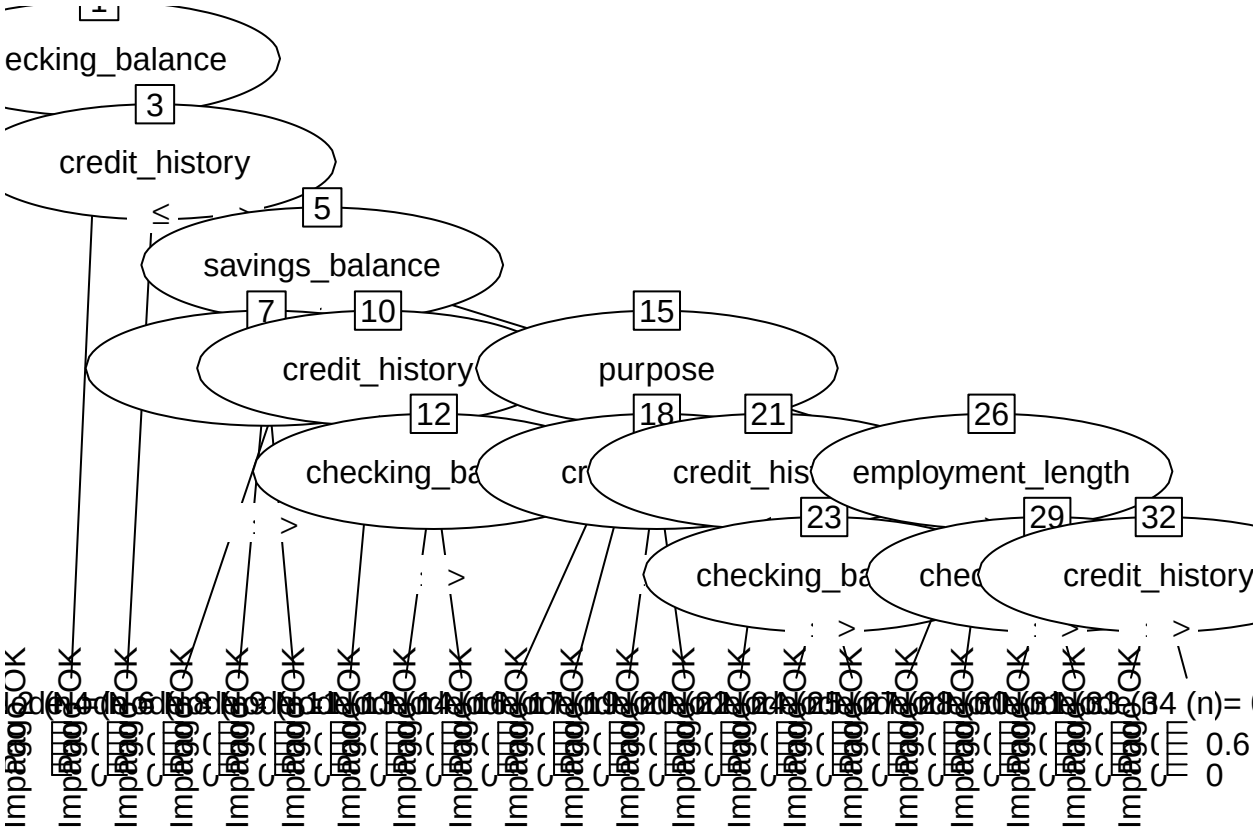
Se puede obtener la misma concluision que el modelo 1 planteado ya que las variables disminuyeron pero siguen siendo las mismas donde el conocimiento extraído y cruzado con el análisis visual se resume en si se presentan un conjunto de caracteristicas desfavorables ese cliente tendrá un Impago tal es el caso de checking_balance < 200 DM o que esta desempleado o mientras menor sea el tiempo de empleo tendrá más probabilidades de no cumplir con el pago.

A continuación el árbol obtenido

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

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



Validación del modelo con los datos reservados

Comprobar la calidad del modelo propuesto a través de el conjunto de datos de prueba.

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

```
## [1] "La precisión del árbol es: 72.5000 %"
```

Análisis de la presición a través de la matriz de confusión que identifica los tipos de errores cometidos.

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

```
##          Predicted
## testy     Impago Pago OK
##   Impago      19      33
##   Pago OK     22     126
```

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_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 72.5000 %"
```

A través del paquete gmodels para obtener información más completa:

```
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',
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  200
##
##
##              | Prediction
##      Reality |    Impago |   Pago OK | Row Total |
## -------------|-----------|-----------|-----------|
##       Impago |        19 |        33 |        52 |
##              |     0.095 |     0.165 |           |
## -------------|-----------|-----------|-----------|
##      Pago OK |        22 |       126 |       148 |
##              |     0.110 |     0.630 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        41 |       159 |       200 |
## -------------|-----------|-----------|-----------|
##
##
```

MODELO 3

Preparación de los datos para el modelo

Procedemos a dividir el Dataset en conjunto de Entrenamiento o Train y Prueba o Test donde el primero tendrá es decir 3/4 de todo los datos y 1/4 corresponderá para el conjunto de prueba para la evaluacion del modelo. Para las variables independientes o X podemos usar los valores de Cramér y Phi para determinar las variables que tienen más correlación con el target "default", a su vez al tratarse del una variante del modelo se usó las variables de "foreign_worker", "job" cuyos valores de Cramér y Phi son los más bajos

```
set.seed(750)
y<-data_random[,"default"]
X<-data_random[,c("checking_balance", "credit_history", "savings_balance","employment_length", "foreign
```

Creación de un rango usando split_plot.

```
split_prop<-4
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]
```

Creación del modelo, calidad del modelo y extracción de reglas

Se crea el árbol de decisión usando 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]      Mon May 23 13:00:07 2022
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 750 cases (7 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (8/1, lift 2.6)
##  checking_balance = < 0 DM
##  credit_history = delayed
##  savings_balance = < 100 DM
##  ->  class Impago  [0.800]
##
## Rule 2: (43/12, lift 2.3)
##  checking_balance in {< 0 DM, 1 - 200 DM, > 200 DM}
##  credit_history in {fully repaid this bank, fully repaid}
##  savings_balance in {< 100 DM, 101 - 500 DM}
##  ->  class Impago  [0.711]
##
## Rule 3: (59/20, lift 2.2)
##  checking_balance = < 0 DM
##  credit_history = repaid
##  savings_balance in {< 100 DM, 101 - 500 DM}
```

```
##  job in {skilled employee, unemployed non-resident}
##  ->  class Impago  [0.656]
##
## Rule 4: (29/11, lift 2.0)
##  checking_balance = 1 - 200 DM
##  savings_balance in {< 100 DM, 101 - 500 DM}
##  job = mangement self-employed
##  ->  class Impago  [0.613]
##
## Rule 5: (172/79, lift 1.8)
##  checking_balance = < 0 DM
##  savings_balance in {< 100 DM, 101 - 500 DM}
##  ->  class Impago  [0.540]
##
## Rule 6: (277/32, lift 1.3)
##  checking_balance = unknown
##  ->  class Pago OK  [0.882]
##
## Rule 7: (211/34, lift 1.2)
##  savings_balance in {unknown, > 1000 DM, 501 - 1000 DM}
##  ->  class Pago OK  [0.836]
##
## Rule 8: (220/39, lift 1.2)
##  credit_history = critical
##  ->  class Pago OK  [0.820]
##
## Rule 9: (581/157, lift 1.0)
##  credit_history in {repaid, critical, delayed}
##  job in {skilled employee, unskilled resident, unemployed non-resident}
##  ->  class Pago OK  [0.729]
##
## Default class: Pago OK
##
##
## Evaluation on training data (750 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##       9  180(24.0%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##      96   131    (a): class Impago
##      49   474    (b): class Pago OK
##
##
##  Attribute usage:
##
##   87.60% credit_history
##   81.33% job
##   66.40% checking_balance
```

```
##   57.60% savings_balance
##
##
## Time: 0.0 secs
```

Errors muestra el número y porcentaje de casos mal clasificados en el subconjunto de entrenamiento. El árbol obtenido clasifica erróneamente 180 de los 750 casos dados, una tasa de error del 24.0 %.

Mediante las reglas podemos asumir que: - checking_balance = < 0 DM y credit_history = delayed y savings_balance = < 100 DM -> Impago. Validéz:80%

- checking_balance = < 0 DM, 1 - 200 DM, > 200 DM y credit_history = fully repaid this bank, fully repaid savings_balance = < 100 DM, 101 - 500 DM y -> Impago. Validéz: 71.1%
- checking_balance = < 0 DM y credit_history = repaid y savings_balance = < 100 DM, 101 - 500 DM y job = skilled employee, unemployed non-resident -> Impago. Validéz: 65.6%
- checking_balance = 1 - 200 DM y savings_balance = < 100 DM, 101 - 500 DM y job = mangementy self-employed -> Impago. Validéz:61.3%
- checking_balance = < 0 DM y savings_balance = < 100 DM, 101 - 500 DM -> Impago. Validéz: 54%
- checking_balance = unknown -> Pago OK. Validéz: 88.2%.
- savings_balance = unknown, > 1000 DM, 501 - 1000 DM -> Pago OK. Validéz: 83.6%
- credit_history = critical -> Pago OK. Validéz: 82%
- credit_history = repaid, critical, delayed y job = skilled employee, unskilled resident, unemployed non-resident -> Pago OK. Validéz: 72.9%

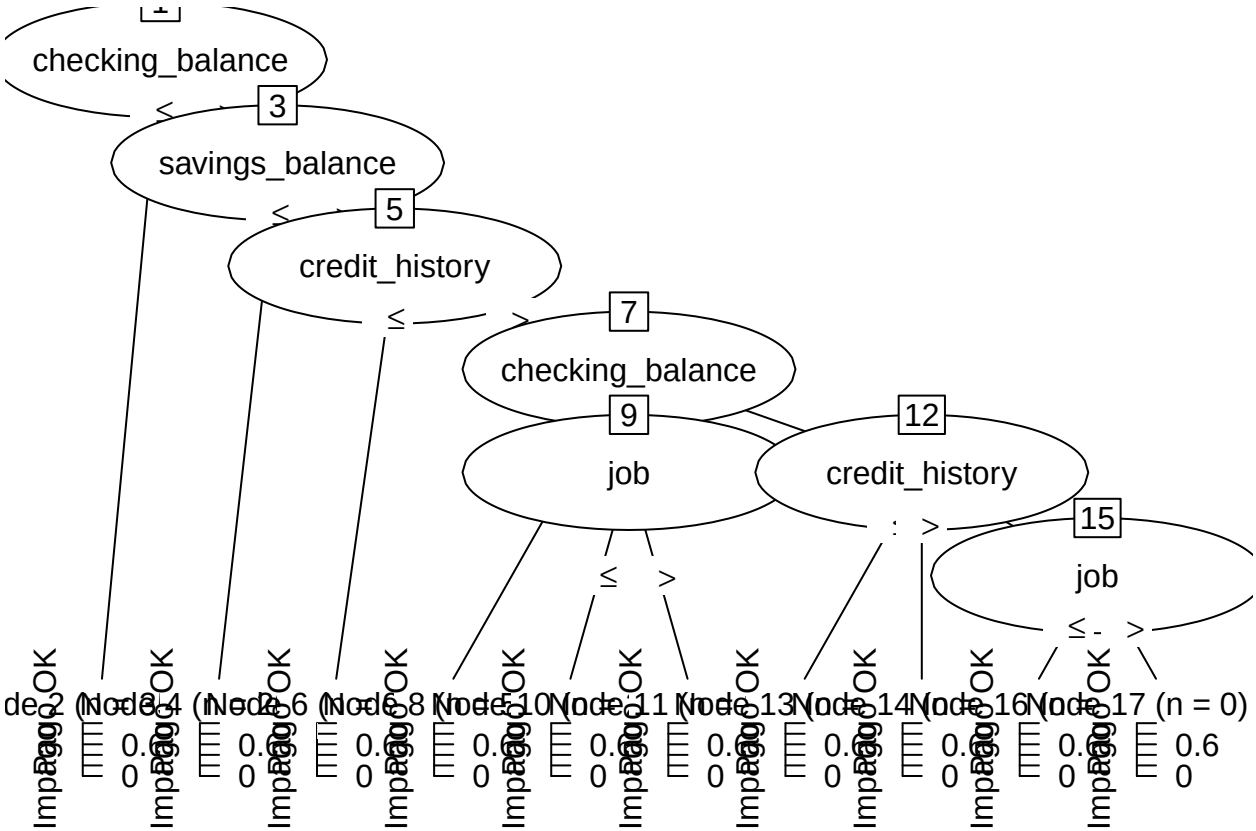
Se pude determinar similares coclusiones alos dos modelos anteriores con la adición que ahora tenemos la inclusion devariables cuyo valor de Cramér y Phi son los más bajos, es así que observamos que cuando se incluyen las variables de Job y foreign_worker en las reglas su Validéz tiende a disminuir, indicando así que no va a existir una eficiente prediccion de default.

A continuación el árbol obtenido

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

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



Validación del modelo con los datos reservados

Comprobar la calidad del modelo propuesto a través de el conjunto de datos de prueba.

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

```
## [1] "La precisión del árbol es: 75.6000 %"
```

Análisis de la presición a través de la matriz de confusión que identifica los tipos de errores cometidos.

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

```
##          Predicted
## testy     Impago Pago OK
##   Impago      30      43
##   Pago OK     18     159
```

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

```
##          Predicted
## testy     Impago Pago OK
##   Impago      30      43
##   Pago OK     18     159
```

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_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 75.6000 %"
```

A través del paquete gmodels para obtener información más completa:

```
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',
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  250
```

```
##
##
##              | Prediction
##      Reality |    Impago |   Pago OK | Row Total |
## -------------|-----------|-----------|-----------|
##       Impago |        30 |        43 |        73 |
##              |     0.120 |     0.172 |           |
## -------------|-----------|-----------|-----------|
##      Pago OK |        18 |       159 |       177 |
##              |     0.072 |     0.636 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        48 |       202 |       250 |
## -------------|-----------|-----------|-----------|
##
##
```

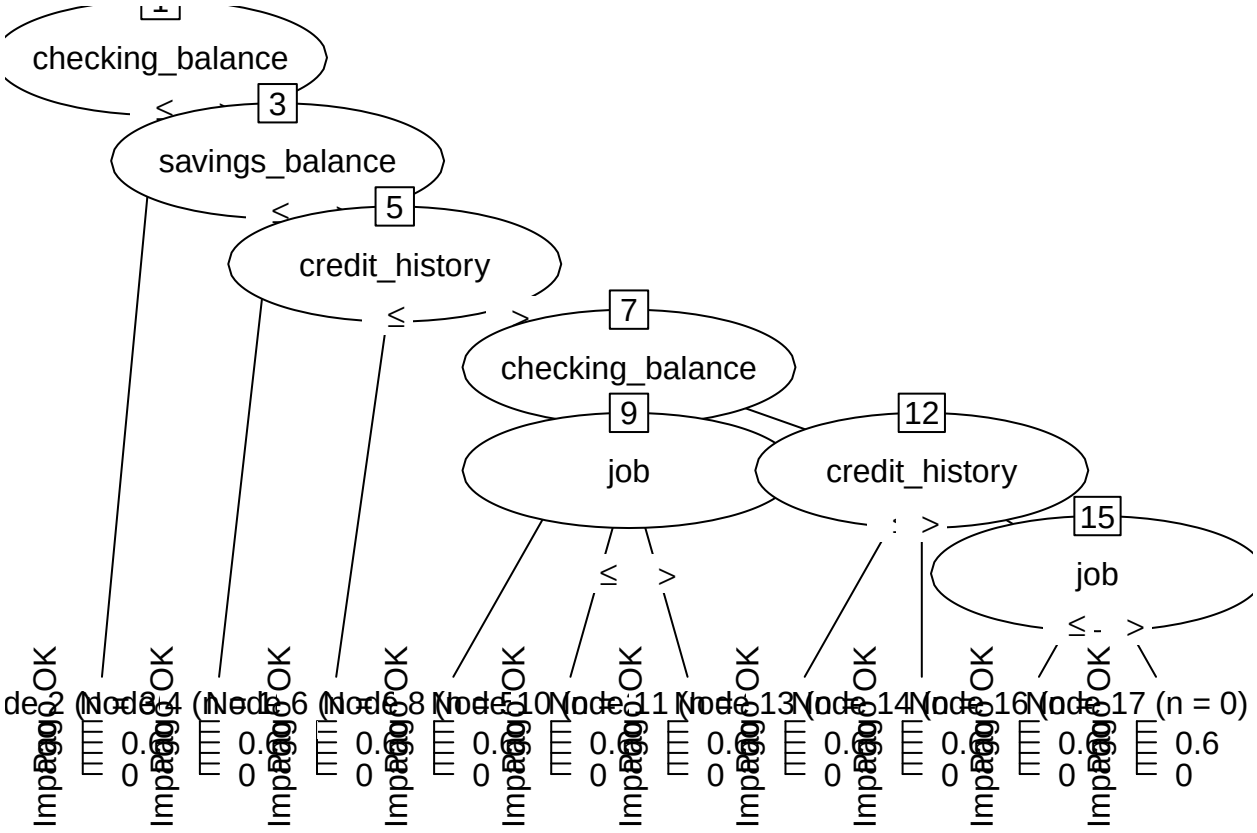
Prueba con una variación u otro enfoque algorítmico

Incorporación de "adaptative boosting" para 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.

```
modelo4<-C50::C5.0(trainX, trainy,trials =10)
plot(modelo4)
```

```
## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion

## Warning in partysplit(varid = as.integer(i), breaks = as.numeric(j[1]), : NAs
## introduced by coercion


## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```

```
## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion

## Warning in .bincode(as.numeric(x), breaks = unique(c(-Inf,
## breaks_split(split), : NAs introduced by coercion
```



Vemos a continuación cómo son las predicciones del nuevo árbol:

```
predicted_model4 <- predict( modelo4, testX, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model4 == testy) / length(predicte
```

```
## [1] "La precisión del árbol es: 73.6000 %"
```

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

```
mat_conf<-table(testy,Predicted=predicted_model4)
mat_conf
```

```
##          Predicted
## testy     Impago Pago OK
##   Impago      29      44
##   Pago OK     22     155
```

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_correct))
```

```
## [1] "El % de registros correctamente clasificados es: 73.6000 %"
```

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

```
##                   Overall
## checking_balance   100.00
## savings_balance    100.00
## purpose             99.40
## credit_history      88.44
## property            66.37
## employment_length   41.14
## housing             12.76
```

```
importancia_splits
```

```
##                     Overall
## checking_balance  32.727273
## savings_balance   20.000000
## credit_history    12.727273
## property          12.727273
## purpose            9.090909
## employment_length  7.272727
## housing            5.454545
```

Como se puede observar los resulatados tanto de importancia_usage yu de importancia_splits se da de forma decreciente señalando así que las primeras variables van a ser las que tengan mayor peso o importancia en nuestro modelo.

CONCLUSIONES

- En Clasificación con árboles de decisión se pudo observar que al aumentar las variables independientes (x) no significa que la presicion aumentará y el error disminuirá en todos los casos, es decir el modelo puede llegar a ser mejor con menos variables independientes.
- Mediante los valores de Valores de la V de Cramér y Phi se puedo obtener las variables cuya relacion con el target (default) es la más elevada, esta funcion fue empleada para la disminución de nuestros datos y por ende de los recursos computacionales al momento del entrenamiento y predicción.
- En el planteamiento de los tres modelos se tanto la presicion por el primer métod y el % de registros correctamente clasificados serán los mismos ya que nada mas fueron resueltos por métodos distintos como la matriz de confusión.
- El modelo C50 posibilita observar las reglas generadas para obtener el resutado requerido, es así que puee tratrase de una regla comuesta o no de diversas condiciones.
- El mejor modelo fue el MODELO 1 con el uso de adaptative boosting donde se obtuvo una pressicion de 76.048%, esto se debe a que se usó las variables que poseen mayor valor de la V de Cramér y Phi