Classification Tree bbdd reducido_plus

Grupo 5

ÍNDICE

1 Base de datos reducido_plus SIN BALANCEAR

```
data4tree<-data_reducida_plus[0:7000,]
datatest<-data_reducida_plus[7001:10000,]
ind <- sample(1:nrow(data4tree), 0.7*nrow(data4tree))
train <- data4tree[ind,]
test <- data4tree[-ind,]
```

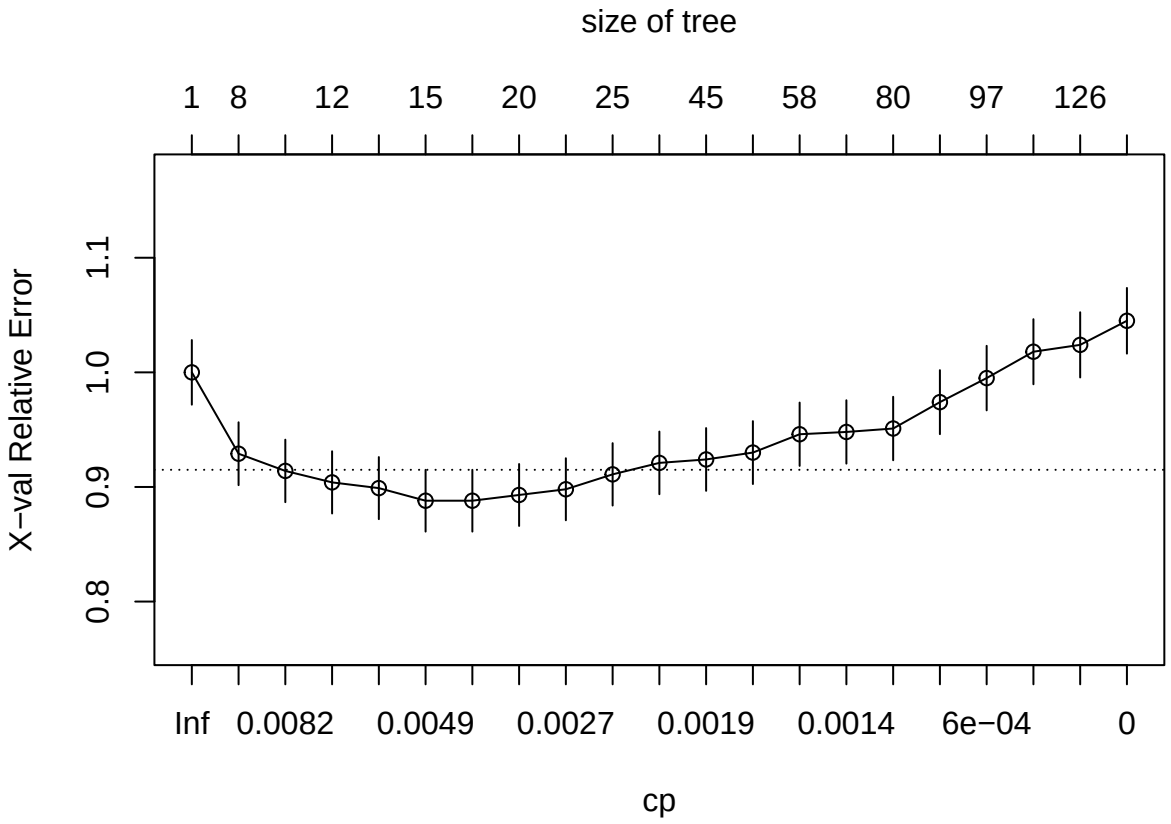
1.1 Método cp=0

```
tree <- rpart(Exited ~ ., data = train, cp = 0)
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = Exited ~ ., data = train, cp = 0)
##
## Variables actually used in tree construction:
## [1] Age                 Balance             CreditScore
## [4] EstimatedSalary     Gender              Geography
## [7] IsActiveMember      NumOfProducts_grupo
##
## Root node error: 1000/4900 = 0.20408
```

```
##
## n= 4900
##
##            CP nsplit rel error xerror     xstd
## 1  0.01500000      0     1.000  1.000 0.028212
## 2  0.00900000      7     0.881  0.929 0.027438
## 3  0.00750000      9     0.863  0.914 0.027267
## 4  0.00700000     11     0.848  0.904 0.027152
## 5  0.00600000     12     0.841  0.899 0.027094
## 6  0.00400000     14     0.829  0.888 0.026964
## 7  0.00350000     17     0.817  0.888 0.026964
## 8  0.00300000     19     0.810  0.893 0.027023
## 9  0.00250000     20     0.807  0.898 0.027082
## 10 0.00225000     24     0.797  0.911 0.027233
## 11 0.00200000     28     0.788  0.921 0.027348
## 12 0.00175000     44     0.753  0.924 0.027382
## 13 0.00166667     51     0.739  0.930 0.027450
## 14 0.00150000     57     0.729  0.946 0.027629
## 15 0.00133333     67     0.714  0.948 0.027651
## 16 0.00100000     79     0.691  0.951 0.027684
## 17 0.00071429     89     0.681  0.974 0.027936
## 18 0.00050000     96     0.676  0.995 0.028159
## 19 0.00033333    119     0.664  1.018 0.028399
## 20 0.00028571    125     0.662  1.024 0.028461
## 21 0.00000000    132     0.660  1.045 0.028673
```

```
plotcp(tree)
```



1.1.1 Elección cp óptimo

```
xerror <- tree$cptable[,"xerror"]
xerror
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 1.000 0.929 0.914 0.904 0.899 0.888 0.888 0.893 0.898 0.911 0.921 0.924 0.930
##    14    15    16    17    18    19    20    21
## 0.946 0.948 0.951 0.974 0.995 1.018 1.024 1.045
```

```
imin.xerror <- which.min(xerror)
imin.xerror
```

```
## 6
## 6
```

```
tree$cptable[imin.xerror, ]
```

```
##          CP      nsplit   rel error      xerror        xstd
##  0.00400000 14.00000000  0.82900000  0.88800000  0.02696428
```

```
upper.xerror <- xerror[imin.xerror] + tree$cptable[imin.xerror, "xstd"]
upper.xerror
```

```
##         6
## 0.9149643
```

```
tree2 <- prune(tree, cp = 0.002967359)
importance <- tree2$variable.importance
importance <- round(100*importance/sum(importance), 1)
importance
```

1.1.1.1 Cp mínimo

```
##                 Age NumOfProducts_grupo             Balance           Geography
##                41.0                32.4                10.7                 4.2
##         CreditScore              Gender     EstimatedSalary      IsActiveMember
##                 3.8                 3.2                 2.8                 1.9
```

Matriz de confusión para train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3767  674
##          1  133  326
##
##                Accuracy : 0.8353
##                  95% CI : (0.8246, 0.8456)
##     No Information Rate : 0.7959
##     P-Value [Acc > NIR] : 1.286e-12
##
##                   Kappa : 0.3654
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.32600
##             Specificity : 0.96590
##          Pos Pred Value : 0.71024
##          Neg Pred Value : 0.84823
##              Prevalence : 0.20408
##          Detection Rate : 0.06653
##    Detection Prevalence : 0.09367
##       Balanced Accuracy : 0.64595
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1563  330
##          1   87  120
##
##                Accuracy : 0.8014
##                  95% CI : (0.7837, 0.8183)
##     No Information Rate : 0.7857
##     P-Value [Acc > NIR] : 0.04103
##
##                   Kappa : 0.2662
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.26667
##             Specificity : 0.94727
##          Pos Pred Value : 0.57971
##          Neg Pred Value : 0.82567
##              Prevalence : 0.21429
##          Detection Rate : 0.05714
##    Detection Prevalence : 0.09857
##       Balanced Accuracy : 0.60697
##
##        'Positive' Class : 1
##
```

F1 score

```
## f1 datos train 0.4468814
```

```
## f1 datos test 0.3652968
```

Los resultados no son satisfactorios: - Valores pobres para F1score y recall - Indicios de overfitting (0.4702809»0.3987915)

```
icp <- which(tree$cptable[, "xerror"] <= upper.xerror)[1]
cp_optimo_1se <- tree$cptable[icp, "CP"]
tree3 <- prune(tree, cp = cp_optimo_1se)
importance <- tree3$variable.importance
importance <- round(100*importance/sum(importance), 1)
importance
```

1.1.1.2 Cp mínimo+ error estándar

```
##                 Age NumOfProducts_grupo             Balance           Geography
##                47.0                38.3                 7.7                 4.0
##      IsActiveMember         CreditScore     EstimatedSalary              Gender
##                 2.2                 0.4                 0.4                 0.0
```

Vemos como han cambiado los valores de importancia. Ahora se comprovará si mejoran los KPI. Matriz de confusión para train

Matriz de confusión para test

F1 score

```
## f1 datos train 0.4052378
```

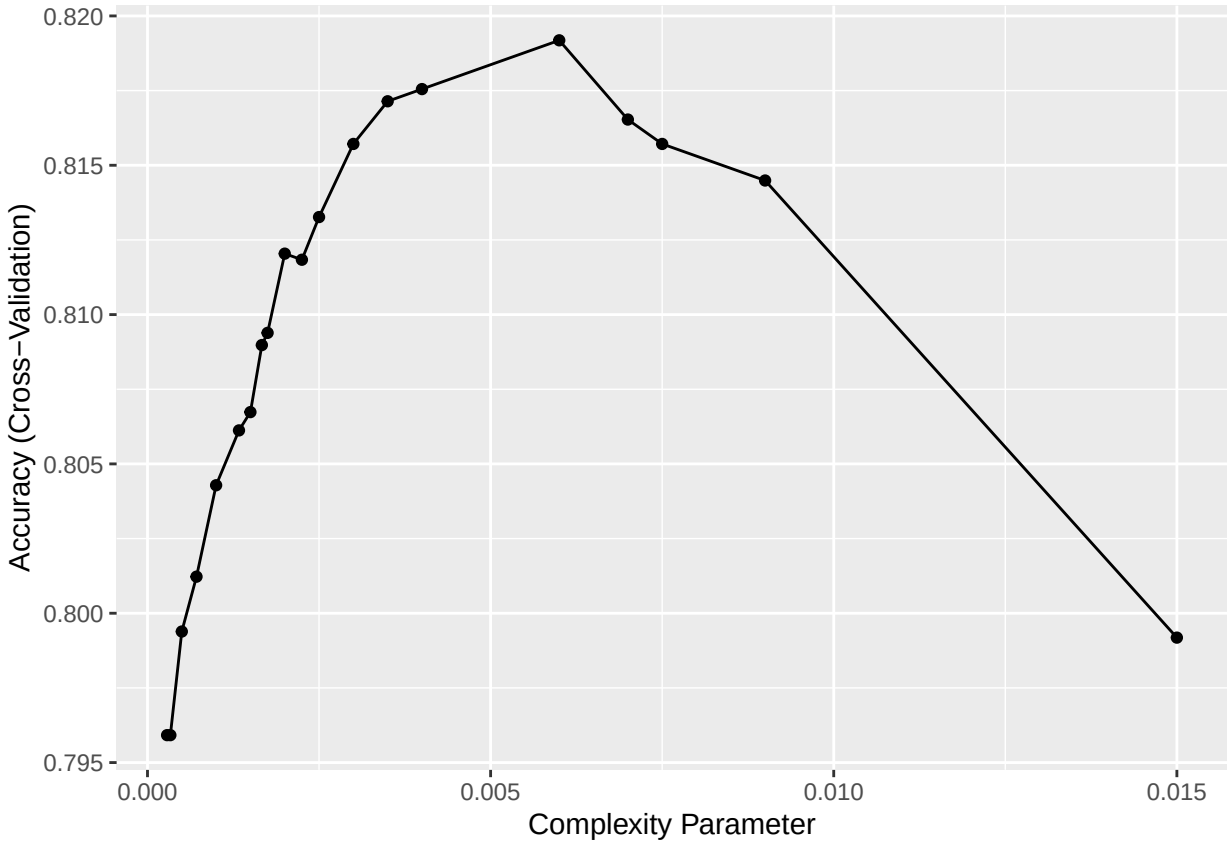
```
## f1 datos test 0.3627907
```

No hay overfitting, pero el f1 score es notablemente peor.

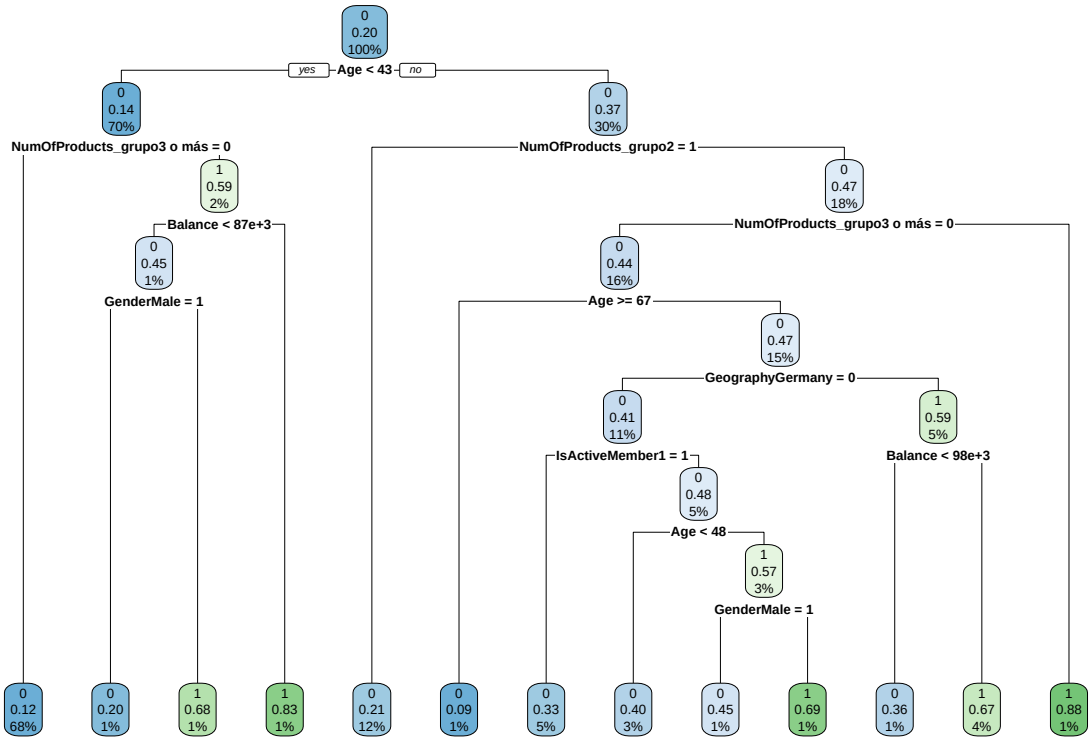
1.2 Método Caret

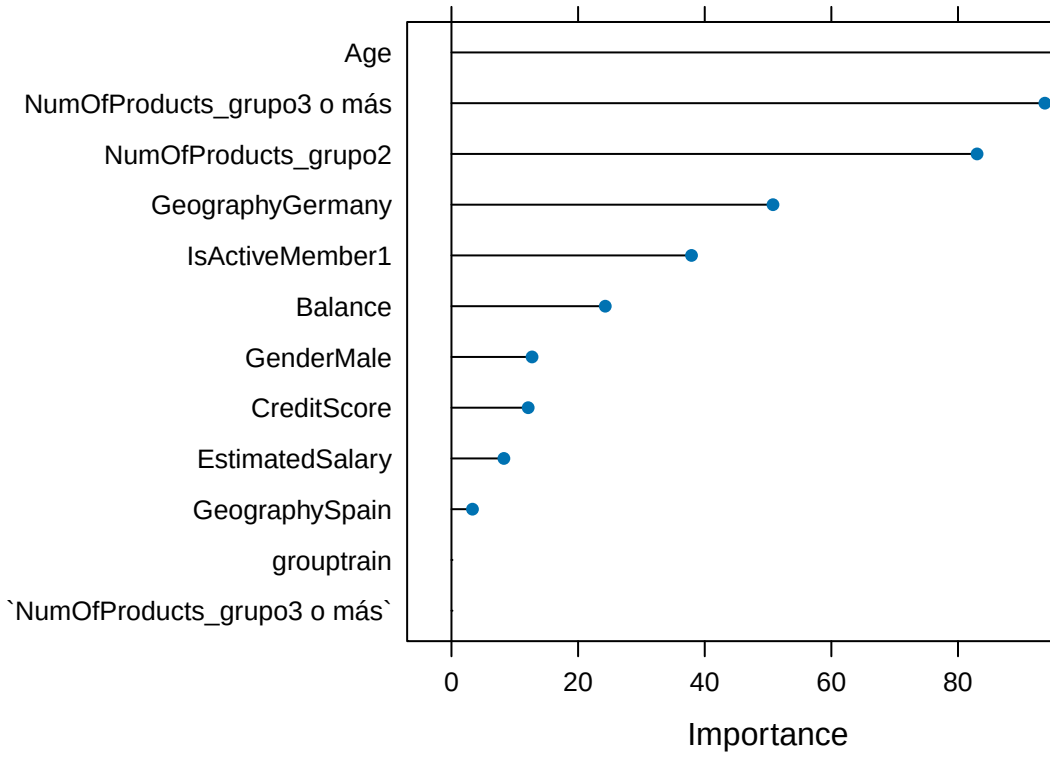
```
caret.rpart <- train(Exited ~ ., method = "rpart", data = train,
 tuneLength = 20,
 trControl = trainControl(method = "cv", number = 10))
ggplot(caret.rpart)
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the caret package.
##   Please report the issue at <https://github.com/topepo/caret/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
rpart.plot(caret.rpart$finalModel)
```





Importancia de las variables:

Las predicciones:

Matriz de confusión datos train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3801  742
##          1   99  258
##
##                Accuracy : 0.8284
##                  95% CI : (0.8175, 0.8388)
##     No Information Rate : 0.7959
##     P-Value [Acc > NIR] : 4.902e-09
##
##                   Kappa : 0.3057
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.25800
##             Specificity : 0.97462
##          Pos Pred Value : 0.72269
##          Neg Pred Value : 0.83667
##              Prevalence : 0.20408
##          Detection Rate : 0.05265
##    Detection Prevalence : 0.07286
##       Balanced Accuracy : 0.61631
##
##        'Positive' Class : 1
##
```

Matriz de confusión datos test:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1585  351
##          1   65   99
##
##                Accuracy : 0.8019
##                  95% CI : (0.7842, 0.8188)
##     No Information Rate : 0.7857
##     P-Value [Acc > NIR] : 0.03648
##
##                   Kappa : 0.2349
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.22000
##             Specificity : 0.96061
##          Pos Pred Value : 0.60366
##          Neg Pred Value : 0.81870
```

```
##              Prevalence : 0.21429
##          Detection Rate : 0.04714
##    Detection Prevalence : 0.07810
##       Balanced Accuracy : 0.59030
##
##        'Positive' Class : 1
##
```

Los F1score:

```
## f1 datos train 0.3224756
```

```
## f1 datos test 0.3802506
```

- Valores muy pobres de F1 y recall
- Indicios de overfitting

1.3 Conclusiones para data_reducido_plus sin balancear

Resultados bastante mejorables tanto con el paquete Caret como encontrando el Cp óptimo a partir del árbol más grande (Cp=0). Sumar la desviación típica tampoco mejora los resultados. Probaremos balanceando los datos.

2 Base de datos reducido_plus BALANCEO

A continuación se buscará para diferentes niveles de balanceo dónde se obtienen mejores kpi con caret

```
##
## Adjuntando el paquete: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
##          F1_Train    F1_Test Sensitivity_Train Sensitivity_Test
## p_0.05 0.09125475 0.07188161             0.048       0.03777778
## p_0.1  0.12407407 0.08281573             0.067       0.04444444
## p_0.15 0.17688266 0.16602317             0.101       0.09555556
## p_0.2  0.33883704 0.33279483             0.236       0.22888889
## p_0.25 0.38828338 0.39269406             0.285       0.28666667
## p_0.3  0.45710928 0.40425532             0.389       0.33777778
## p_0.35 0.46871795 0.44000000             0.457       0.41555556
## p_0.4  0.49836678 0.44000000             0.534       0.46444444
## p_0.45 0.49627012 0.45229682             0.632       0.56888889
## p_0.5  0.49171484 0.45077720             0.638       0.58000000
##        Specificity_Train Specificity_Test Accuracy_Train Accuracy_Test
```

```
## p_0.05       0.9989744       0.9963636       0.8048980       0.7909524
## p_0.1        0.9966667       0.9921212       0.8069388       0.7890476
## p_0.15       0.9894872       0.9848485       0.8081633       0.7942857
## p_0.2        0.9597436       0.9600000       0.8120408       0.8033333
## p_0.25       0.9530769       0.9527273       0.8167347       0.8100000
## p_0.3        0.9197436       0.9090909       0.8114286       0.7866667
## p_0.35       0.8735897       0.8709091       0.7885714       0.7733333
## p_0.4        0.8438462       0.8236364       0.7806122       0.7466667
## p_0.45       0.7653846       0.7418182       0.7381633       0.7047619
## p_0.5        0.7546154       0.7290909       0.7308163       0.6971429
```

2.1 Conclusiones balanceo

- Se obtienen mejores resultados que sin balancear
- con balanceo 50-50 se obtiene el mejor recall. Se logra detectar 6 de cada 10 clientes que abandonan el banco.
- En términos de F-score, los mejores resultados se obtienen con balanceo 0.45.