Classification Tree bbdd imputado_bsd

Grupo 5

ÍNDICE

1 Base de datos transformada SIN BALANCEAR

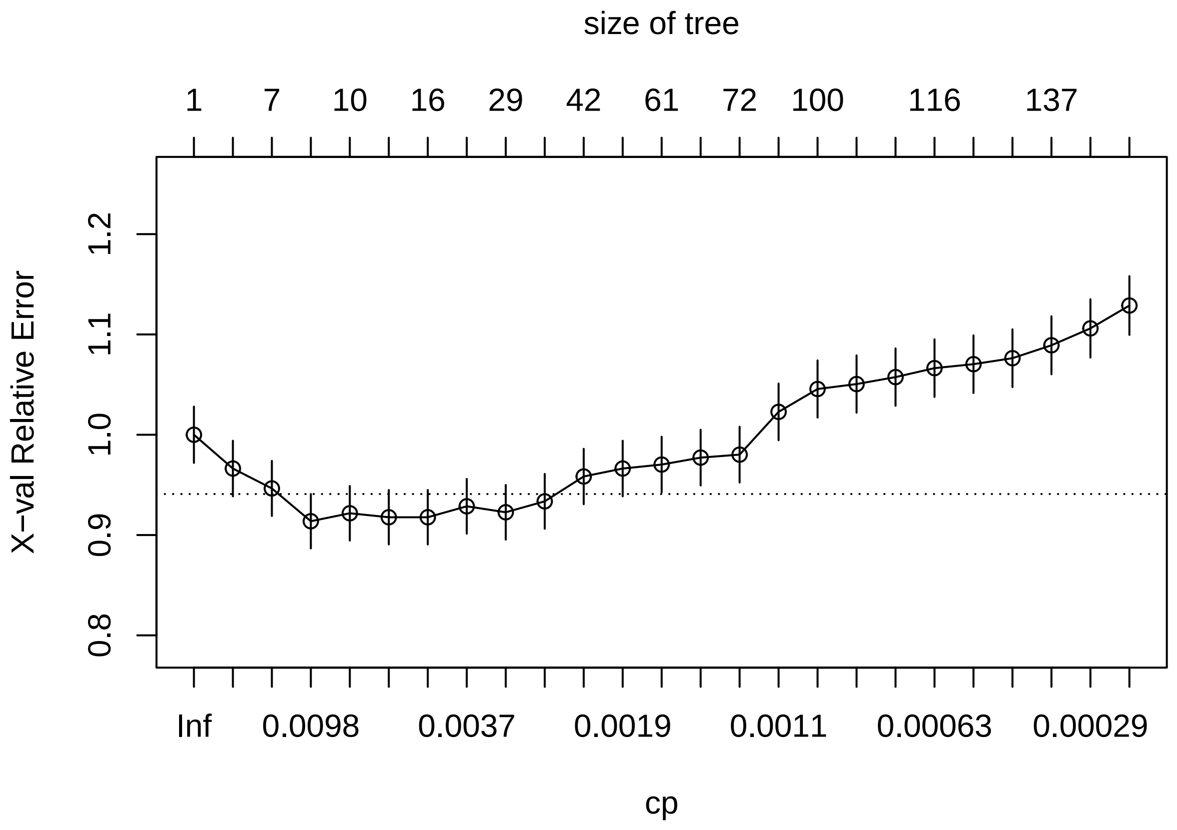
```
data_transformada <- data_transformada %>%
  mutate(across(where(is.numeric), ~ as.numeric(scale(.x))))
data4tree<-data_transformada[0:7000,]
datatest<-data_transformada[7001:10000,]
ind <- sample(1:nrow(data4tree), 0.7*nrow(data4tree))
train <- data4tree[ind,]
test <- data4tree[-ind,]
```

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                    AvgTransactionAmount   Balance
##  [4] CreditScore            CustomerSegment        DigitalEngagementScore
##  [7] EducationLevel         EstimatedSalary        Gender
## [10] Geography              HasCrCard              IsActiveMember
## [13] LoanStatus             MaritalStatus          NetPromoterScore
## [16] NumOfProducts_grupo    SavingsAccountFlag     Tenure
## [19] TransactionFrequency
##
## Root node error: 1009/4900 = 0.20592
##
## n= 4900
##
##            CP nsplit rel error  xerror     xstd
## 1  0.01651800      0   1.00000 1.00000 0.028053
## 2  0.01585728      5   0.90387 0.96630 0.027697
## 3  0.01288404      6   0.88801 0.94648 0.027481
## 4  0.00743310      7   0.87512 0.91378 0.027115
## 5  0.00528576      9   0.86026 0.92170 0.027205
## 6  0.00462504     12   0.84440 0.91774 0.027160
## 7  0.00396432     15   0.83053 0.91774 0.027160
## 8  0.00346878     19   0.81467 0.92864 0.027283
## 9  0.00297324     28   0.78295 0.92270 0.027216
## 10 0.00240691     33   0.76809 0.93360 0.027338
## 11 0.00198216     41   0.74529 0.95837 0.027611
## 12 0.00173439     56   0.71160 0.96630 0.027697
## 13 0.00165180     60   0.70466 0.97027 0.027740
## 14 0.00148662     63   0.69970 0.97721 0.027814
## 15 0.00132144     71   0.68682 0.98018 0.027845
## 16 0.00099108     74   0.68285 1.02279 0.028287
## 17 0.00088096     99   0.65312 1.04559 0.028516
## 18 0.00074331    108   0.64519 1.05055 0.028565
## 19 0.00066072    112   0.64222 1.05748 0.028633
## 20 0.00059465    115   0.64024 1.06640 0.028719
## 21 0.00056633    120   0.63726 1.07037 0.028758
## 22 0.00042475    129   0.63132 1.07631 0.028815
## 23 0.00033036    136   0.62834 1.08920 0.028937
## 24 0.00024777    139   0.62735 1.10605 0.029095
## 25 0.00000000    143   0.62636 1.12884 0.029304
```

```r
plotcp(tree)
```



Mirando el gráfico, el mínimo se encuentra aproximadamente en la región donde Lambda es alrededor de 0.0032 - 0.0019, Número de variables = 34 - 64, Error relativo = 0.95

El punto más bajo parece estar alrededor de lambda = 0.0032 con aproximadamente 34-51 variables en el modelo.

1.1.1 Elección cp óptimo

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

```
##         1         2         3         4         5         6         7         8 
## 1.0000000 0.9663033 0.9464817 0.9137760 0.9217047 0.9177403 0.9177403 0.9286422 
##         9        10        11        12        13        14        15        16 
## 0.9226957 0.9335976 0.9583746 0.9663033 0.9702676 0.9772052 0.9801784 1.0227948 
##        17        18        19        20        21        22        23        24 
## 1.0455897 1.0505451 1.0574827 1.0664024 1.0703667 1.0763132 1.0891972 1.1060456 
##        25 
## 1.1288404
```

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

```
## 4 
## 4
```

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

```
##          CP      nsplit   rel error      xerror        xstd 
## 0.007433102 7.000000000 0.875123885 0.913776016 0.027114940
```

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

```
##        4 
## 0.940891
```

Los valores son bastante similados para el caso de la base de datos *balanceado plus* El mínimo error es 0.923 en la posición 6, que corresponde a un árbol con 14 divisiones y un CP = 0.005429418

```


tree2 <- prune(tree, cp = 0.005429418)
importance <- tree2$variable.importance
importance <- round(100*importance/sum(importance), 1)
importance
```

1.1.1.1 Cp mínimo

```
##                    Age    NumOfProducts_grupo                Balance 
##                   49.6                   34.5                    5.1 
##         IsActiveMember            CreditScore   TransactionFrequency 
##                    3.5                    3.4                    1.7 
##   AvgTransactionAmount        EstimatedSalary DigitalEngagementScore 
##                    0.7                    0.5                    0.3 
##         EducationLevel              Geography                 Tenure 
##                    0.3                    0.2                    0.2
```

Los resultados muestran que NumOfProducts (39.7%) y Age (39.4%) son las variables dominantes, explicando conjuntamente el 79.1% de la capacidad predictiva del modelo para identificar clientes que abandonarán el banco. Esto indica que el número de productos contratados y la edad del cliente son los factores más determinantes en la decisión de abandono. Las siguientes variables en importancia, IsActiveMember (4%) y Balance 11.7%), tienen un impacto considerablemente menor, mientras que factores como Tenure (0.4%) resultan prácticamente irrelevantes para el modelo.

Matriz de confusión para train

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 3196  505
##          1  695  504
##                                           
##                Accuracy : 0.7551          
##                  95% CI : (0.7428, 0.7671)
##     No Information Rate : 0.7941
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3
##
##  Mcnemar's Test P-Value : 4.871e-08
##
##             Sensitivity : 0.4995
##             Specificity : 0.8214
##          Pos Pred Value : 0.4204
##          Neg Pred Value : 0.8636
##              Prevalence : 0.2059
##          Detection Rate : 0.1029
##    Detection Prevalence : 0.2447
##       Balanced Accuracy : 0.6604
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1355  248
##          1  304  193
##
##                Accuracy : 0.7371
##                  95% CI : (0.7178, 0.7559)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.2431
##
##  Mcnemar's Test P-Value : 0.01923
##
##             Sensitivity : 0.4376
##             Specificity : 0.8168
##          Pos Pred Value : 0.3883
##          Neg Pred Value : 0.8453
##              Prevalence : 0.2100
##          Detection Rate : 0.0919
##    Detection Prevalence : 0.2367
##       Balanced Accuracy : 0.6272
##
##        'Positive' Class : 1
##
```

F1 score

```
## f1 datos train 0.4565217
```

```
## f1 datos test 0.4115139
```

Los resultados no son satisfactorios: - Valores pobres para F1score y recall - Hay indicios de ligero overfitting: el F1 en train es claramente mayor que en test por minima diferencia, aunque la diferencia no es grande.

```r
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         IsActiveMember
##                  52.1                   36.4                    3.7
##               Balance            CreditScore   AvgTransactionAmount
##                   3.4                    3.2                    0.4
##       EstimatedSalary DigitalEngagementScore   TransactionFrequency
##                   0.4                    0.3                    0.0
```

El peso de las variables ha augmentado en algunos casos.

Matriz de confusión para train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3196  505
##          1  695  504
##
##                Accuracy : 0.7551
##                  95% CI : (0.7428, 0.7671)
##     No Information Rate : 0.7941
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3
##
##  Mcnemar's Test P-Value : 4.871e-08
##
##             Sensitivity : 0.4995
##             Specificity : 0.8214
##          Pos Pred Value : 0.4204
##          Neg Pred Value : 0.8636
##              Prevalence : 0.2059
##          Detection Rate : 0.1029
##    Detection Prevalence : 0.2447
##       Balanced Accuracy : 0.6604
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1355  248
##          1  304  193
##
##                Accuracy : 0.7371
##                  95% CI : (0.7178, 0.7559)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.2431
##
##  Mcnemar's Test P-Value : 0.01923
##
##             Sensitivity : 0.4376
##             Specificity : 0.8168
##          Pos Pred Value : 0.3883
##          Neg Pred Value : 0.8453
##              Prevalence : 0.2100
##          Detection Rate : 0.0919
##    Detection Prevalence : 0.2367
##       Balanced Accuracy : 0.6272
##
##        'Positive' Class : 1
##
```

F1 score

```
## f1 datos train 0.4565217
```

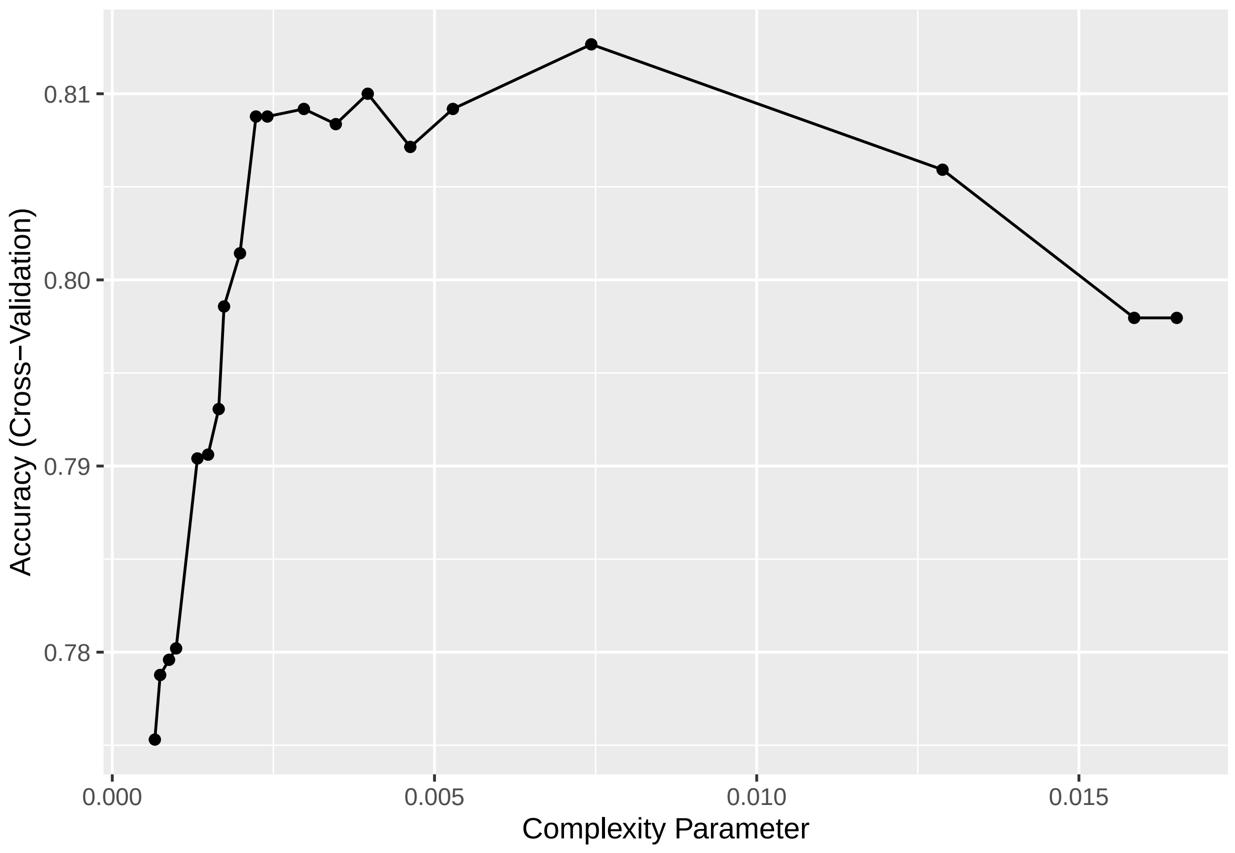
```
## f1 datos test 0.4115139
```

No hay overfitting, el f1 ha disminuido para este caso, no obstante los dos valores se podrian considerar parecidos

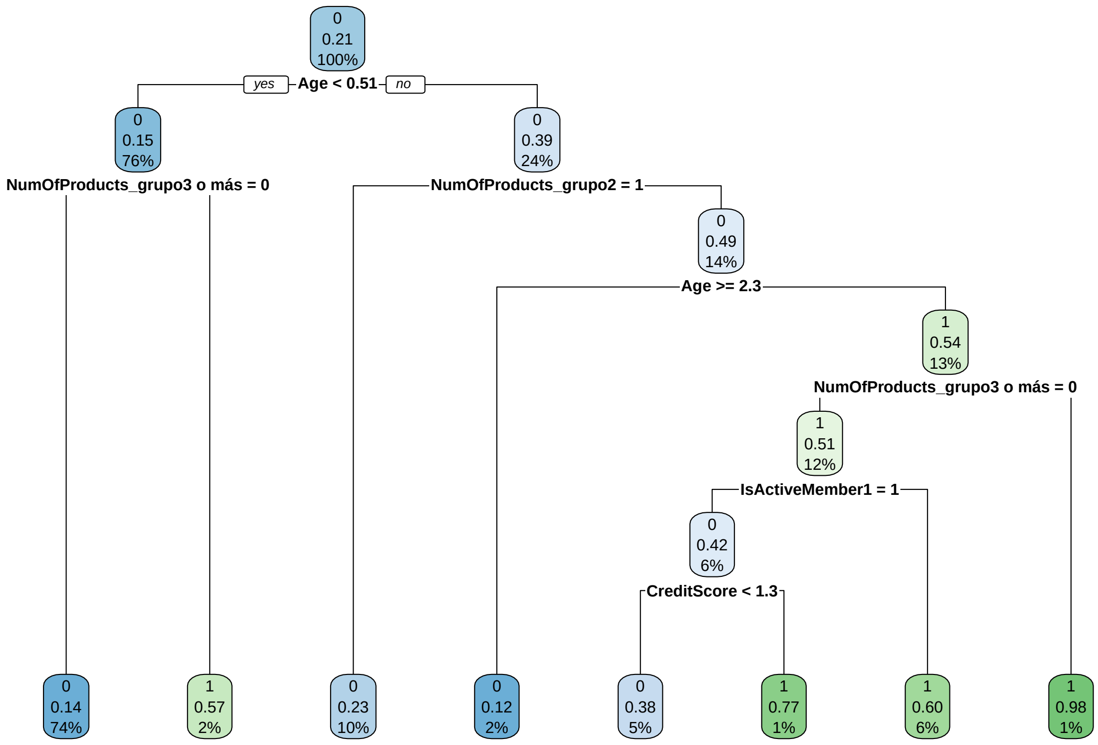
1.2 Método Caret

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

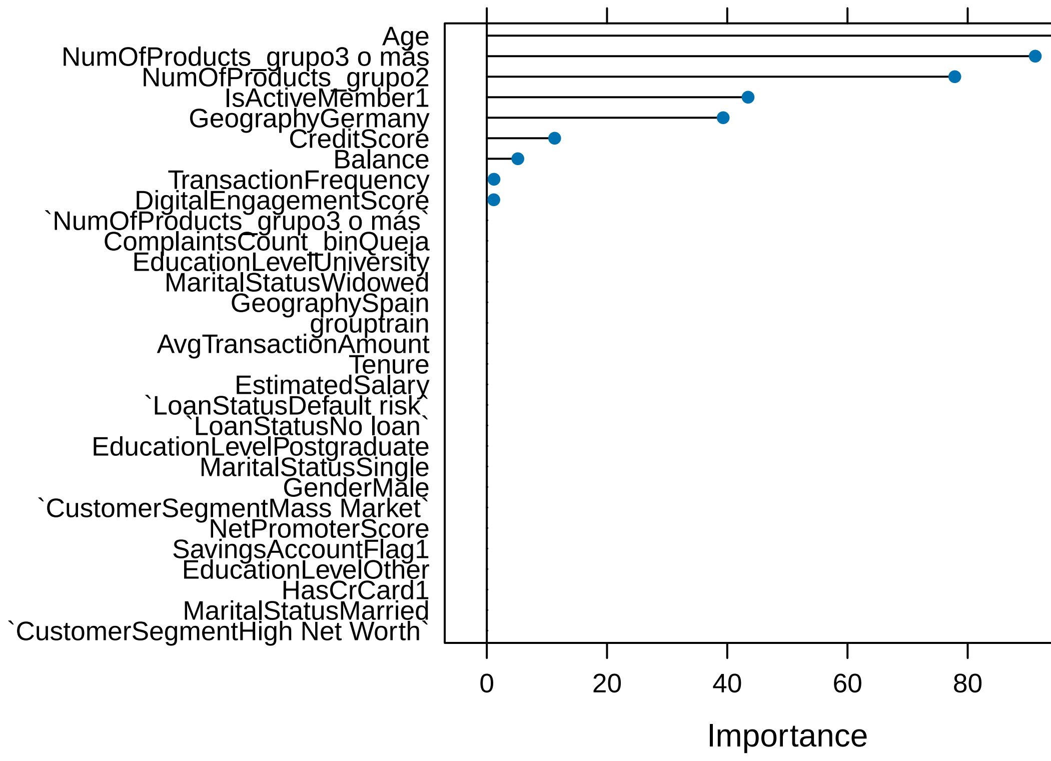
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the caret package.
##   Please report the issue at <https://github.com/topepo/caret/issues>.
## 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)
```



El árbol muestra que la edad y el número de productos son los factores clave: los clientes más jóvenes (<42) casi siempre permanecen, mientras que los pocos casos con muchos productos y alta actividad o mayores con >=3 productos tienen mayor probabilidad de marcharse.



Importancia de las variables:
Las predicciones:

Matriz de confusión datos train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1355  248
##          1  304  193
##
##                Accuracy : 0.7371
##                  95% CI : (0.7178, 0.7559)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.2431
##
##  Mcnemar's Test P-Value : 0.01923
##
##             Sensitivity : 0.4376
##             Specificity : 0.8168
##          Pos Pred Value : 0.3883
##          Neg Pred Value : 0.8453
##              Prevalence : 0.2100
##          Detection Rate : 0.0919
##    Detection Prevalence : 0.2367
##       Balanced Accuracy : 0.6272
##
##        'Positive' Class : 1
##
```

Matriz de confusión datos test:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1571  316
##          1   88  125
##
##                Accuracy : 0.8076
##                  95% CI : (0.7901, 0.8243)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 0.02437
##
##                   Kappa : 0.2844
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.28345
##             Specificity : 0.94696
##          Pos Pred Value : 0.58685
##          Neg Pred Value : 0.83254
##              Prevalence : 0.21000
##          Detection Rate : 0.05952
##    Detection Prevalence : 0.10143
##       Balanced Accuracy : 0.61520
##
##        'Positive' Class : 1
##
```

Los F1score:

```
## f1 datos train 0.382263
```

```
## f1 datos test 0.4115139
```

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

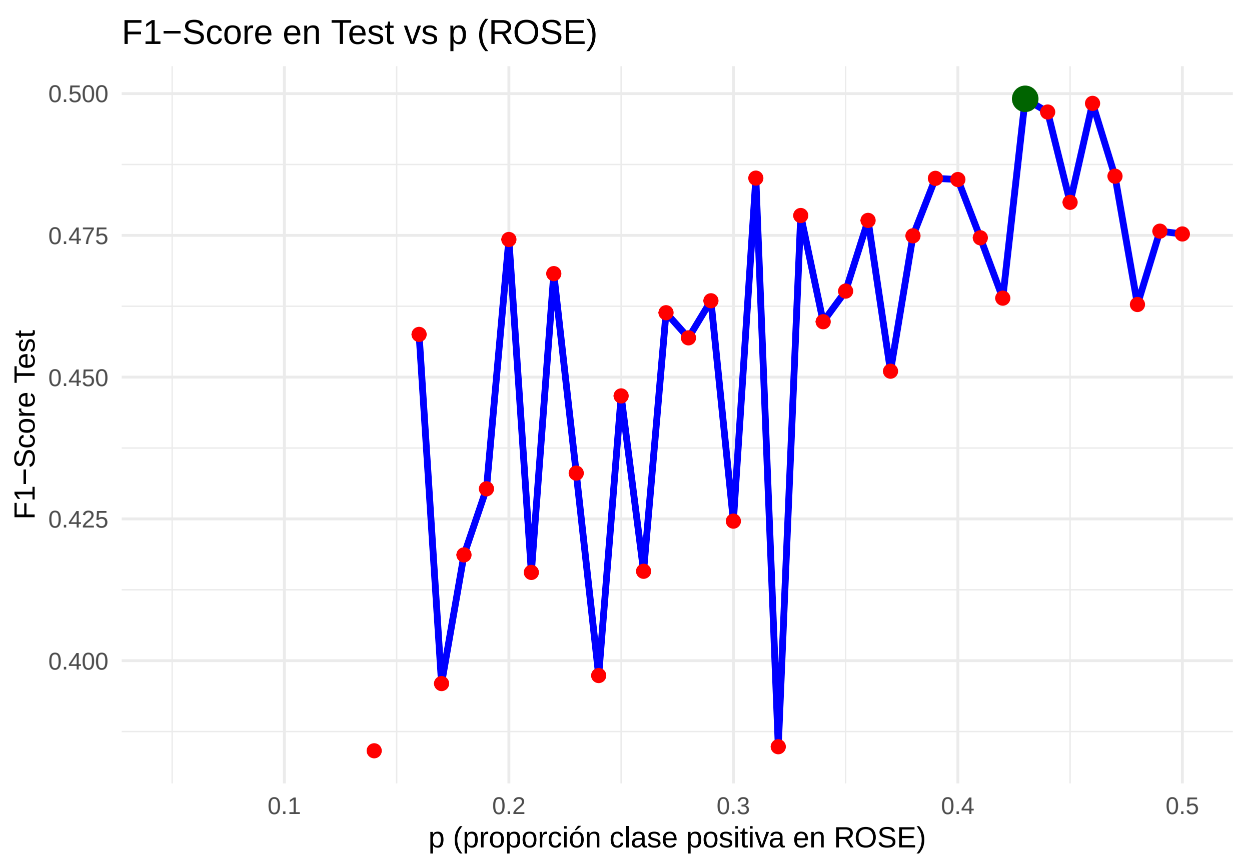
1.3 Conclusiones para data imputado sin balancear

Con los datos imputados, los resultados siguen siendo claramente mejorables: ni el tuning con caret ni ajustar el Cp óptimo aportan mejoras sustanciales. Esto indica que el problema no está en el modelo sino en la estructura del conjunto imputado, por lo que balancear las clases es el siguiente paso adecuado para intentar mejorar el rendimiento.

2 Base de datos transformada BALANCEO

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

```
##                     p          cp  F1_Train   F1_Test Sensitivity_Train
## Pos Pred Value   0.05 0.007246377       NaN       NaN         0.0000000
## Pos Pred Value1  0.06 0.004744958       NaN       NaN         0.0000000
## Pos Pred Value2  0.07 0.007507508       NaN       NaN         0.0000000
## Pos Pred Value3  0.08 0.005064045       NaN       NaN         0.0000000
## Pos Pred Value4  0.09 0.004245283       NaN       NaN         0.0000000
## Pos Pred Value5  0.10 0.004550321       NaN       NaN         0.0000000
## Pos Pred Value6  0.11 0.007858546       NaN       NaN         0.0000000
## Pos Pred Value7  0.12 0.006773619       NaN       NaN         0.0000000
## Pos Pred Value8  0.13 0.006700168       NaN       NaN         0.0000000
## Pos Pred Value9  0.14 0.005335366 0.4266292 0.3841060         0.4509415
## Pos Pred Value10 0.15 0.008932769       NaN       NaN         0.0000000
## Pos Pred Value11 0.16 0.006570302 0.4324853 0.4575311         0.4380575
## Pos Pred Value12 0.17 0.007407407 0.4222798 0.3959596         0.4846383
## Pos Pred Value13 0.18 0.002923977 0.4504657 0.4186508         0.5272547
## Pos Pred Value14 0.19 0.005446623 0.4455128 0.4303087         0.5510406
## Pos Pred Value15 0.20 0.004629630 0.4739382 0.4742729         0.4866204
## Pos Pred Value16 0.21 0.006882989 0.4284559 0.4155598         0.5252725
## Pos Pred Value17 0.22 0.003289474 0.4666667 0.4682540         0.5411298
## Pos Pred Value18 0.23 0.004046763 0.4611111 0.4330709         0.4112983
## Pos Pred Value19 0.24 0.003301477 0.4363636 0.3973758         0.5351833
## Pos Pred Value20 0.25 0.003344482 0.4456468 0.4466891         0.4489594
## Pos Pred Value21 0.26 0.004796163 0.4436918 0.4157635         0.5193261
## Pos Pred Value22 0.27 0.004583652 0.4645355 0.4613610         0.4608523
## Pos Pred Value23 0.28 0.003681885 0.4564984 0.4569288         0.4212091
## Pos Pred Value24 0.29 0.003556188 0.4721689 0.4634703         0.4876115
## Pos Pred Value25 0.30 0.006168609 0.4555886 0.4246032         0.5312190
## Pos Pred Value26 0.31 0.004497002 0.4759806 0.4850976         0.5351833
## Pos Pred Value27 0.32 0.003238342 0.4425287 0.3848238         0.3815659
## Pos Pred Value28 0.33 0.006558401 0.4644052 0.4785006         0.4816650
## Pos Pred Value29 0.34 0.003022975 0.4850025 0.4597701         0.4727453
## Pos Pred Value30 0.35 0.003907776 0.4825581 0.4651685         0.4935580
## Pos Pred Value31 0.36 0.004859920 0.4873311 0.4776423         0.5718533
## Pos Pred Value32 0.37 0.003070910 0.4807692 0.4510451         0.5203171
## Pos Pred Value33 0.38 0.004687218 0.4883619 0.4749232         0.5510406
## Pos Pred Value34 0.39 0.007615546 0.4784736 0.4850575         0.4846383
## Pos Pred Value35 0.40 0.011282051 0.4772215 0.4848485         0.5242815
## Pos Pred Value36 0.41 0.004506760 0.4824078 0.4745763         0.5639247
## Pos Pred Value37 0.42 0.003642545 0.4757812 0.4639269         0.6035679
## Pos Pred Value38 0.43 0.004039924 0.4870878 0.4990826         0.6075322
## Pos Pred Value39 0.44 0.003810624 0.4761905 0.4967563         0.5946482
## Pos Pred Value40 0.45 0.003389065 0.4960876 0.4808232         0.6283449
## Pos Pred Value41 0.46 0.003756076 0.4841240 0.4982759         0.6422200
## Pos Pred Value42 0.47 0.003907946 0.4791509 0.4854369         0.6263627
## Pos Pred Value43 0.48 0.005314626 0.4716649 0.4628099         0.6640238
## Pos Pred Value44 0.49 0.005016722 0.4610238 0.4757433         0.6828543
## Pos Pred Value45 0.50 0.004526749 0.4797272 0.4752475         0.6273538
##                  Sensitivity_Test Specificity_Train Specificity_Test
## Pos Pred Value          0.0000000         1.0000000        1.0000000
## Pos Pred Value1         0.0000000         1.0000000        1.0000000
## Pos Pred Value2         0.0000000         1.0000000         1.0000000
## Pos Pred Value3         0.0000000         1.0000000         1.0000000
## Pos Pred Value4         0.0000000         1.0000000         1.0000000
## Pos Pred Value5         0.0000000         1.0000000         1.0000000
## Pos Pred Value6         0.0000000         1.0000000         1.0000000
## Pos Pred Value7         0.0000000         1.0000000         1.0000000
## Pos Pred Value8         0.0000000         1.0000000         1.0000000
## Pos Pred Value9         0.3945578         0.8280648         0.8245931
## Pos Pred Value10        0.0000000         1.0000000         1.0000000
## Pos Pred Value11        0.4580499         0.8475970         0.8553345
## Pos Pred Value12        0.4444444         0.7897713         0.7872212
## Pos Pred Value13        0.4784580         0.7890003         0.7854129
## Pos Pred Value14        0.5215420         0.7607299         0.7600964
## Pos Pred Value15        0.4807256         0.8529941         0.8547318
## Pos Pred Value16        0.4965986         0.7597019         0.7625075
## Pos Pred Value17        0.5351474         0.7982524         0.8004822
## Pos Pred Value18        0.3741497         0.9033667         0.9059675
## Pos Pred Value19        0.4807256         0.7620149         0.7504521
## Pos Pred Value20        0.4512472         0.8532511         0.8487040
## Pos Pred Value21        0.4784580         0.7869442         0.7811935
## Pos Pred Value22        0.4535147         0.8643022         0.8637734
## Pos Pred Value23        0.4149660         0.8900026         0.8933092
## Pos Pred Value24        0.4603175         0.8501671         0.8601567
## Pos Pred Value25        0.4852608         0.7923413         0.7872212
## Pos Pred Value26        0.5351474         0.8149576         0.8215793
## Pos Pred Value27        0.3219955         0.9110768         0.9065702
## Pos Pred Value28        0.4920635         0.8463120         0.8499096
## Pos Pred Value29        0.4535147         0.8763814         0.8619650
## Pos Pred Value30        0.4693878         0.8568491         0.8541290
## Pos Pred Value31        0.5328798         0.7990234         0.8143460
## Pos Pred Value32        0.4648526         0.8329478         0.8414708
## Pos Pred Value33        0.5260771         0.8170136         0.8167571
## Pos Pred Value34        0.4784580         0.8596762         0.8685955
## Pos Pred Value35        0.5260771         0.8254947         0.8288125
## Pos Pred Value36        0.5396825         0.7992804         0.8047016
## Pos Pred Value37        0.5759637         0.7579029         0.7588909
## Pos Pred Value38        0.6167800         0.7699820         0.7727547
## Pos Pred Value39        0.6077098         0.7658700         0.7769741
## Pos Pred Value40        0.5827664         0.7653559         0.7763713
## Pos Pred Value41        0.6553288         0.7378566         0.7408077
## Pos Pred Value42        0.6235828         0.7437677         0.7486438
## Pos Pred Value43        0.6349206         0.7013621         0.7052441
## Pos Pred Value44        0.6893424         0.6682087         0.6787221
## Pos Pred Value45        0.5986395         0.7437677         0.7552743
##                   Accuracy_Train Accuracy_Test
## Pos Pred Value         0.7940816     0.7900000
## Pos Pred Value1        0.7940816     0.7900000
## Pos Pred Value2        0.7940816     0.7900000
## Pos Pred Value3        0.7940816     0.7900000
## Pos Pred Value4        0.7940816     0.7900000
## Pos Pred Value5        0.7940816     0.7900000
## Pos Pred Value6        0.7940816     0.7900000
## Pos Pred Value7        0.7940816     0.7900000
## Pos Pred Value8        0.7940816     0.7900000
## Pos Pred Value9       0.7504082     0.7342857
## Pos Pred Value10      0.7940816     0.7900000
## Pos Pred Value11      0.7632653     0.7719048
## Pos Pred Value12      0.7269388     0.7152381
## Pos Pred Value13      0.7351020     0.7209524
## Pos Pred Value14      0.7175510     0.7100000
## Pos Pred Value15      0.7775510     0.7761905
## Pos Pred Value16      0.7114286     0.7066667
## Pos Pred Value17      0.7453061     0.7447619
## Pos Pred Value18      0.8020408     0.7942857
## Pos Pred Value19      0.7153061     0.6938095
## Pos Pred Value20      0.7700000     0.7652381
## Pos Pred Value21      0.7318367     0.7176190
## Pos Pred Value22      0.7812245     0.7776190
## Pos Pred Value23      0.7934694     0.7928571
## Pos Pred Value24      0.7755102     0.7761905
## Pos Pred Value25      0.7385714     0.7238095
## Pos Pred Value26      0.7573469     0.7614286
## Pos Pred Value27      0.8020408     0.7838095
## Pos Pred Value28      0.7712245     0.7747619
## Pos Pred Value29      0.7932653     0.7761905
## Pos Pred Value30      0.7820408     0.7733333
## Pos Pred Value31      0.7522449     0.7552381
## Pos Pred Value32      0.7685714     0.7623810
## Pos Pred Value33      0.7622449     0.7557143
## Pos Pred Value34      0.7824490     0.7866667
## Pos Pred Value35      0.7634694     0.7652381
## Pos Pred Value36      0.7508163     0.7490476
## Pos Pred Value37      0.7261224     0.7204762
## Pos Pred Value38      0.7365306     0.7400000
## Pos Pred Value39      0.7306122     0.7414286
## Pos Pred Value40      0.7371429     0.7357143
## Pos Pred Value41      0.7181633     0.7228571
## Pos Pred Value42      0.7195918     0.7223810
## Pos Pred Value43      0.6936735     0.6904762
## Pos Pred Value44      0.6712245     0.6809524
## Pos Pred Value45      0.7197959     0.7223810
```



```
##                       p          cp  F1_Train   F1_Test Sensitivity_Train
## Pos Pred Value38 0.43 0.004039924 0.4870878 0.4990826         0.6075322
##                  Sensitivity_Test Specificity_Train Specificity_Test
## Pos Pred Value38          0.61678          0.769982        0.7727547
##                  Accuracy_Train Accuracy_Test
## Pos Pred Value38      0.7365306          0.74
```

2.1 Características del mejor árbol

```
library(ROSE)
library(caret)
library(dplyr)


best_row <- resultados[which.max(resultados$F1_Test), ]
best_p  <- best_row$p
best_cp <- best_row$cp

data_best <- ROSE(Exited ~ ., data = train, p = best_p, seed = 123)$data

best_model <- train(
  Exited ~ .,
  data = data_best,
  method = "rpart",
```

```r
  tuneGrid = data.frame(cp = best_cp),
  trControl = trainControl(method = "none")
)

tree  <- best_model$finalModel
frame <- tree$frame

n_nodes        <- nrow(frame)
n_leaves       <- sum(frame$var == "<leaf>")
max_depth      <- max(frame$depth)
```

```
## Warning in max(frame$depth): ningun argumento finito para max; retornando -Inf
```

```r
min_leaf_size  <- min(frame$n[frame$var == "<leaf>"])
root <- frame[1, ]

p0_root <- root$yval2[4]
p1_root <- root$yval2[5]

gini_root <- 1 - (p0_root^2 + p1_root^2)
gini_root
```

```
## [1] 0.4900278
```

```r
gini_nodes <- sapply(1:nrow(frame), function(i) {
  p0 <- frame$yval2[i, 4]
  p1 <- frame$yval2[i, 5]
  1 - (p0^2 + p1^2)
})

gini_weighted <- sum(frame$n * gini_nodes) / sum(frame$n)
gini_weighted
```

```
## [1] 0.4602625
```

```r
gini_reduction <- tree$variable.importance


tabla_arbol <- data.frame(
  Medida = c(
    "Número total de nodos",
    "Número de hojas",
    "Profundidad máxima",
    "Tamaño mínimo de hoja",
    "Gini del nodo raíz",
    "Gini medio ponderado del árbol"
  ),
  Valor = c(
    n_nodes,
    n_leaves,
    max_depth,
    min_leaf_size,
    round(gini_root, 3),
    round(gini_weighted, 3)
  )
)

print(tabla_arbol)
```

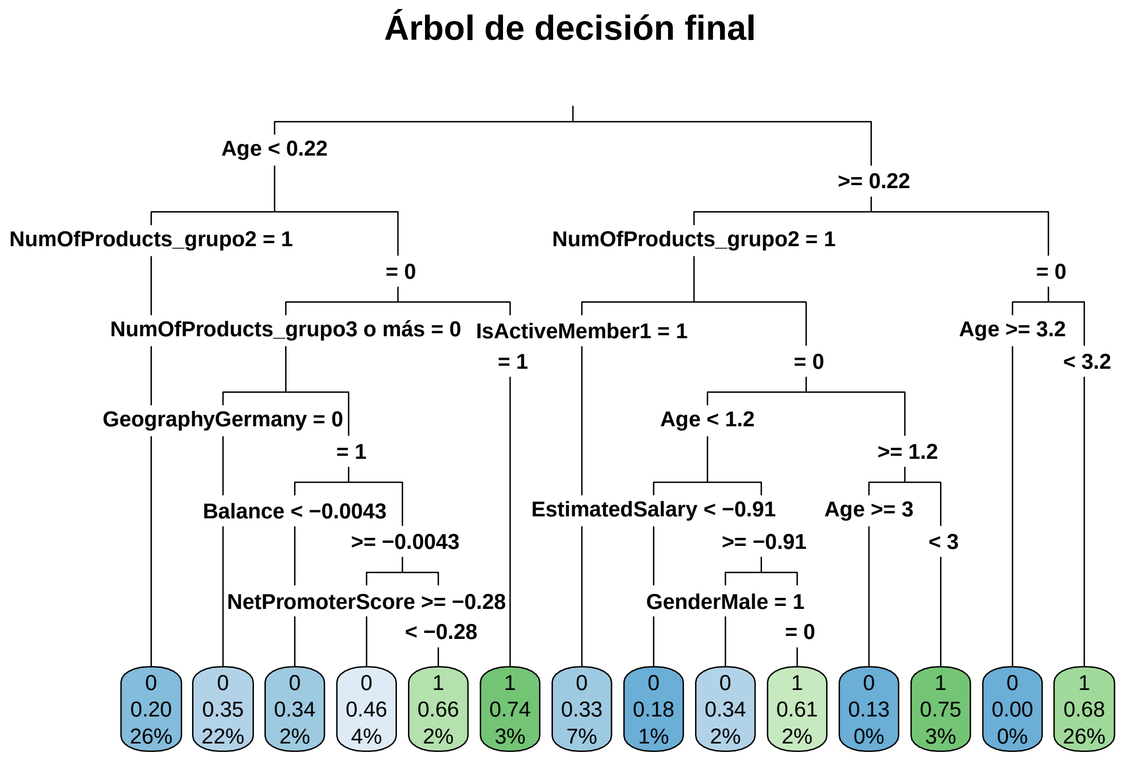
```
##                            Medida Valor
## 1          Número total de nodos 27.00
## 2                Número de hojas 14.00
## 3             Profundidad máxima  -Inf
## 4          Tamaño mínimo de hoja 15.00
## 5              Gini del nodo raíz  0.49
## 6 Gini medio ponderado del árbol  0.46
```

```
tabla_gini_vars <- data.frame(
  Variable = names(gini_reduction),
  Reduccion_Gini = as.numeric(gini_reduction)
) %>%
  arrange(desc(Reduccion_Gini))

print(tabla_gini_vars)
```

```
##                          Variable Reduccion_Gini
## 1                             Age    201.6608061
## 2              NumOfProducts_grupo2    119.3780791
## 3        NumOfProducts_grupo3 o más     28.1594983
## 4                         Balance     26.0400311
## 5                  IsActiveMember1     12.9732254
## 6                 GeographyGermany     11.1300850
## 7                  EstimatedSalary      8.6984762
## 8                 NetPromoterScore      7.8829268
## 9                       GenderMale      6.0067663
## 10                          Tenure      3.2625491
## 11                LoanStatusNo loan      2.8072871
## 12          DigitalEngagementScore      2.5663043
## 13         ComplaintsCount_binQueja      2.4926983
## 14                     CreditScore      2.4395913
## 15 CustomerSegmentHigh Net Worth      1.3422221
## 16    EducationLevelPostgraduate      1.1504761
## 17             AvgTransactionAmount      1.1452723
## 18            TransactionFrequency      0.7165213
## 19             SavingsAccountFlag1      0.6231746
```

2.2 Visualizamos el mejor árbol

```
library(rpart.plot)

rpart.plot(tree,
           type = 3,          # cajas completas con bordes
           extra = 106,       # muestra clase, probabilidades y nº de observaciones
           fallen.leaves = TRUE,
           cex = 0.7,         # tamaño del texto
           main = "Árbol de decisión final")
```



Árbol de decisión final
Age < 0.22
>= 0.22
NumOfProducts_grupo2 = 1
= 0
NumOfProducts_grupo2 = 1
= 0
NumOfProducts_grupo3 o más = 0
IsActiveMember1 = 1
= 1
= 0
Age >= 3.2
< 3.2
GeographyGermany = 0
= 1
Age < 1.2
>= 1.2
Balance < −0.0043
>= −0.0043
EstimatedSalary < −0.91
>= −0.91
Age >= 3
< 3
NetPromoterScore >= −0.28
< −0.28
GenderMale = 1
= 0
0 0.20 26%
0 0.35 22%
0 0.34 2%
0 0.46 4%
1 0.66 2%
1 0.74 3%
0 0.33 7%
0 0.18 1%
0 0.34 2%
1 0.61 2%
0 0.13 0%
1 0.75 3%
0 0.00 0%
1 0.68 26%