Classification Tree bbdd imputado_bsd

Grupo 5

ÍNDICE

1 Base de datos transformada SIN BALANCEAR

```
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,]
```

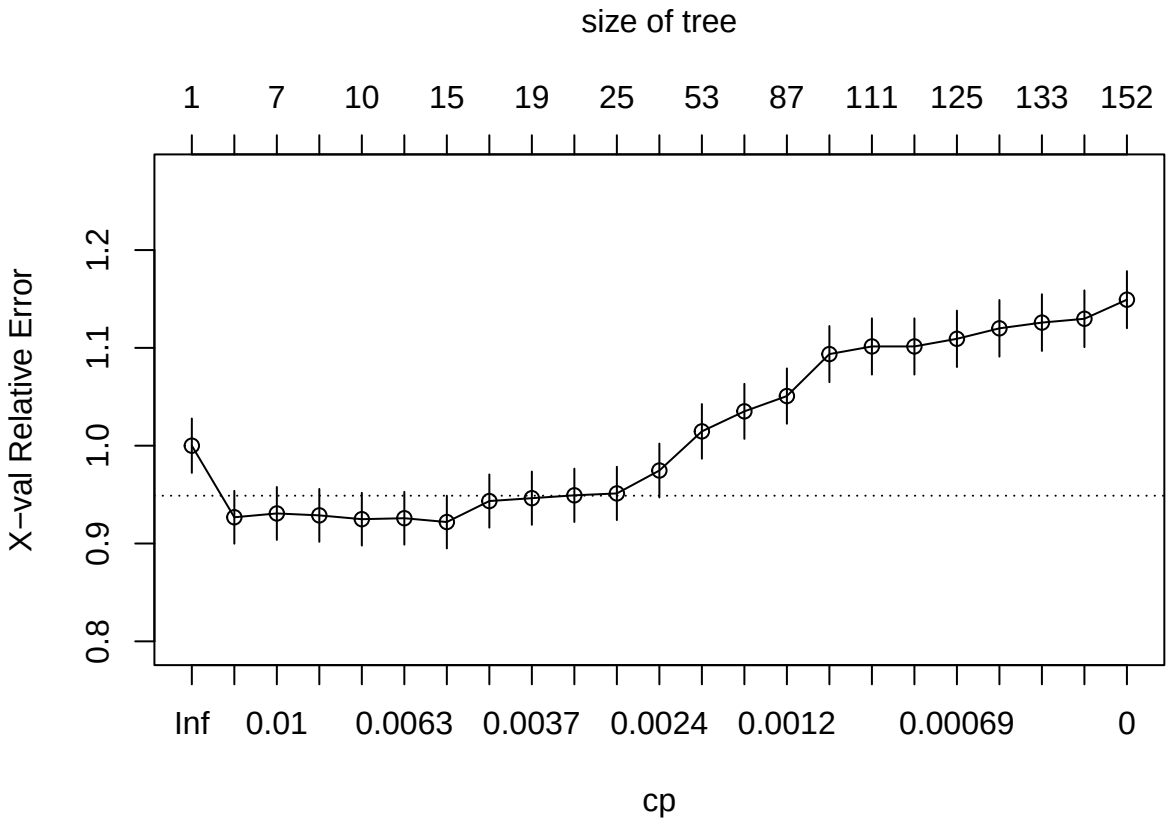
1.1 Método cp=0

```
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] ComplaintsCount_bin    CreditScore            CustomerSegment
##  [7] DigitalEngagementScore EducationLevel         EstimatedSalary
## [10] Gender                 Geography              HasCrCard
## [13] IsActiveMember         LoanStatus             MaritalStatus
```

```
## [16] NetPromoterScore       NumOfProducts_grupo    Tenure
## [19] TransactionFrequency
##
## Root node error: 1025/4900 = 0.20918
##
## n= 4900
##
##            CP nsplit rel error  xerror     xstd
## 1  0.01528455      0   1.00000 1.00000 0.027776
## 2  0.01170732      5   0.90341 0.92683 0.026998
## 3  0.00878049      6   0.89171 0.93073 0.027042
## 4  0.00731707      7   0.88293 0.92878 0.027020
## 5  0.00682927      9   0.86829 0.92488 0.026977
## 6  0.00585366     10   0.86146 0.92585 0.026988
## 7  0.00439024     14   0.83805 0.92195 0.026944
## 8  0.00390244     16   0.82927 0.94341 0.027180
## 9  0.00341463     18   0.82146 0.94634 0.027212
## 10 0.00317073     20   0.81463 0.94927 0.027244
## 11 0.00292683     24   0.80195 0.95122 0.027265
## 12 0.00195122     38   0.75512 0.97463 0.027514
## 13 0.00146341     52   0.72488 1.01463 0.027925
## 14 0.00130081     83   0.66341 1.03512 0.028128
## 15 0.00105691     86   0.65951 1.05073 0.028281
## 16 0.00097561     99   0.64488 1.09366 0.028686
## 17 0.00083624    110   0.63317 1.10146 0.028758
## 18 0.00073171    120   0.62244 1.10146 0.028758
## 19 0.00065041    124   0.61951 1.10927 0.028829
## 20 0.00058537    127   0.61756 1.12000 0.028925
## 21 0.00048780    132   0.61463 1.12585 0.028978
## 22 0.00032520    148   0.60390 1.12976 0.029012
## 23 0.00000000    151   0.60293 1.14927 0.029184
```

```
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 0.9268293 0.9307317 0.9287805 0.9248780 0.9258537 0.9219512 0.9434146
##         9        10        11        12        13        14        15        16
## 0.9463415 0.9492683 0.9512195 0.9746341 1.0146341 1.0351220 1.0507317 1.0936585
##        17        18        19        20        21        22        23
## 1.1014634 1.1014634 1.1092683 1.1200000 1.1258537 1.1297561 1.1492683
```

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

```
## 7
## 7
```

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

```
##           CP       nsplit    rel error       xerror         xstd
##  0.004390244 14.000000000  0.838048780  0.921951220  0.026944321
```

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

```
##         7
## 0.9488955
```

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
##                  38.2                   37.7                   10.5
##         IsActiveMember DigitalEngagementScore   AvgTransactionAmount
##                   5.6                    2.5                    2.0
##              Geography        EstimatedSalary            CreditScore
##                   1.7                    0.6                    0.4
##                 Tenure             LoanStatus          MaritalStatus
##                   0.3                    0.1                    0.1
##       NetPromoterScore   TransactionFrequency
##                   0.1                    0.1
```

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 3358  533
##          1  517  492
##
##                Accuracy : 0.7857
```

```
##                  95% CI : (0.774, 0.7971)
##     No Information Rate : 0.7908
##     P-Value [Acc > NIR] : 0.8150
##
##                   Kappa : 0.3486
##
##  Mcnemar's Test P-Value : 0.6434
##
##             Sensitivity : 0.4800
##             Specificity : 0.8666
##          Pos Pred Value : 0.4876
##          Neg Pred Value : 0.8630
##              Prevalence : 0.2092
##          Detection Rate : 0.1004
##    Detection Prevalence : 0.2059
##       Balanced Accuracy : 0.6733
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1465  236
##          1  210  189
##
##                Accuracy : 0.7876
##                  95% CI : (0.7695, 0.8049)
##     No Information Rate : 0.7976
##     P-Value [Acc > NIR] : 0.8782
##
##                   Kappa : 0.3268
##
##  Mcnemar's Test P-Value : 0.2365
##
##             Sensitivity : 0.4447
##             Specificity : 0.8746
##          Pos Pred Value : 0.4737
##          Neg Pred Value : 0.8613
##              Prevalence : 0.2024
##          Detection Rate : 0.0900
##    Detection Prevalence : 0.1900
##       Balanced Accuracy : 0.6597
##
##        'Positive' Class : 1
##
```

F1 score

```
## f1 datos train 0.4837758
```

```
## f1 datos test 0.4587379
```

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.

```
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
##                    46.8                   36.7                    7.7
##                 Balance DigitalEngagementScore            CreditScore
##                     6.6                    0.8                    0.5
##    AvgTransactionAmount        EstimatedSalary   TransactionFrequency
##                     0.4                    0.4                    0.1
##                  Tenure
##                     0.1
```

El peso de las variables ha augmentado en algunos casos.

Matriz de confusión para train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3351  579
##          1  524  446
##
##                Accuracy : 0.7749
##                  95% CI : (0.7629, 0.7865)
##     No Information Rate : 0.7908
##     P-Value [Acc > NIR] : 0.9969
##
##                   Kappa : 0.3059
##
##  Mcnemar's Test P-Value : 0.1040
##
##             Sensitivity : 0.43512
##             Specificity : 0.86477
##          Pos Pred Value : 0.45979
##          Neg Pred Value : 0.85267
##              Prevalence : 0.20918
##          Detection Rate : 0.09102
##    Detection Prevalence : 0.19796
##       Balanced Accuracy : 0.64995
```

```
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1465  253
##          1  210  172
##
##                Accuracy : 0.7795
##                  95% CI : (0.7612, 0.7971)
##     No Information Rate : 0.7976
##     P-Value [Acc > NIR] : 0.98095
##
##                   Kappa : 0.2903
##
##  Mcnemar's Test P-Value : 0.05095
##
##             Sensitivity : 0.4047
##             Specificity : 0.8746
##          Pos Pred Value : 0.4503
##          Neg Pred Value : 0.8527
##              Prevalence : 0.2024
##          Detection Rate : 0.0819
##    Detection Prevalence : 0.1819
##       Balanced Accuracy : 0.6397
##
##        'Positive' Class : 1
##
```

F1 score

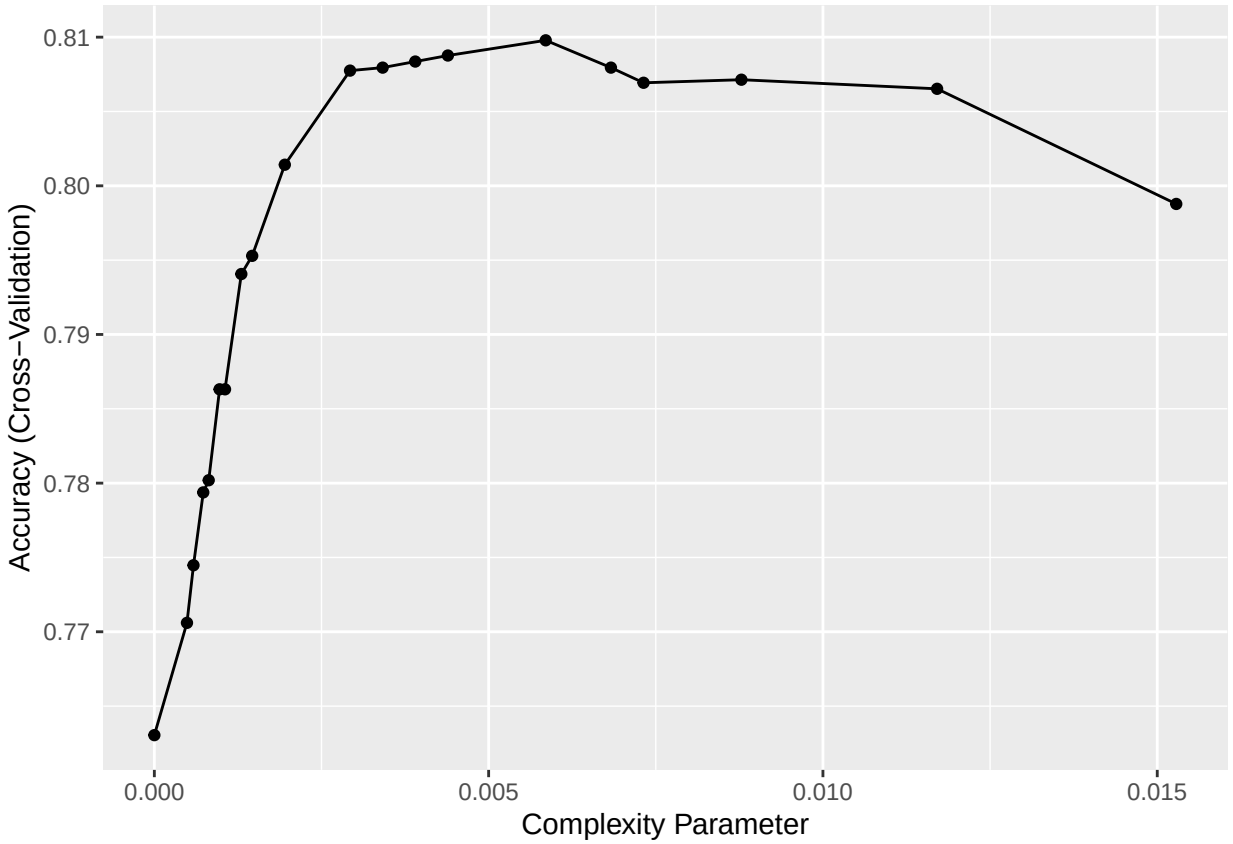
```
## f1 datos train 0.4471178
```

```
## f1 datos test 0.4262701
```

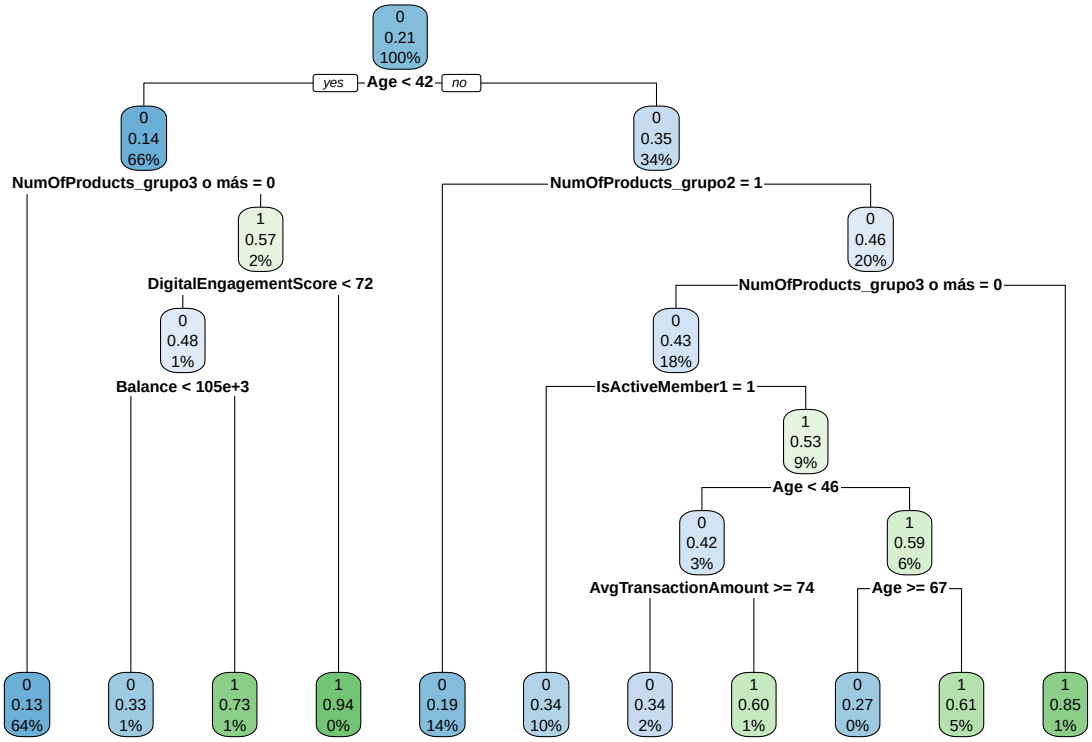
No hay overfitting, el f1 ha disminuido para este caso, no obstante los dos valores se podrian considerar parecidos

1.2 Método Caret

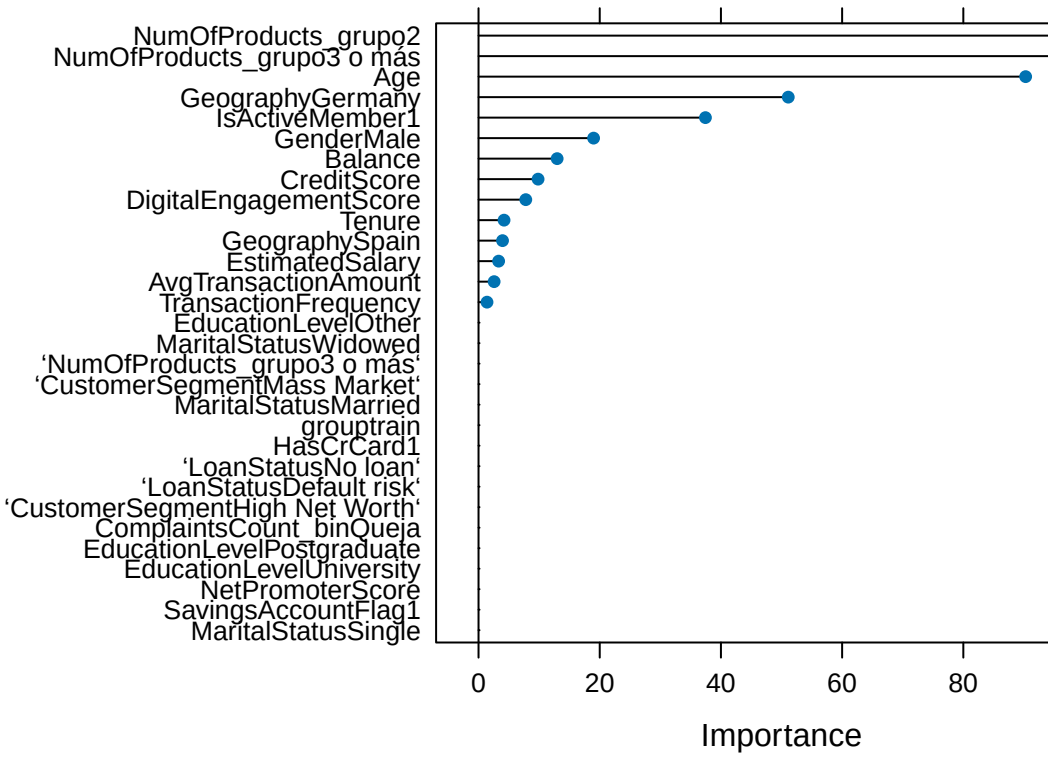
```
caret.rpart <- train(Exited ~ ., method = "rpart", data = train,
 tuneLength = 20,
 trControl = trainControl(method = "cv", number = 10))
ggplot(caret.rpart)
```




```
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 1465  253
##          1  210  172
##
##                Accuracy : 0.7795
##                  95% CI : (0.7612, 0.7971)
##     No Information Rate : 0.7976
##     P-Value [Acc > NIR] : 0.98095
##
##                   Kappa : 0.2903
##
##  Mcnemar's Test P-Value : 0.05095
##
##             Sensitivity : 0.4047
##             Specificity : 0.8746
##          Pos Pred Value : 0.4503
##          Neg Pred Value : 0.8527
##              Prevalence : 0.2024
##          Detection Rate : 0.0819
##    Detection Prevalence : 0.1819
```

```
##       Balanced Accuracy : 0.6397
##
##        'Positive' Class : 1
##
```

Matriz de confusión datos test:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1610  327
##          1   65   98
##
##                Accuracy : 0.8133
##                  95% CI : (0.796, 0.8298)
##     No Information Rate : 0.7976
##     P-Value [Acc > NIR] : 0.03778
##
##                   Kappa : 0.2491
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.23059
##             Specificity : 0.96119
##          Pos Pred Value : 0.60123
##          Neg Pred Value : 0.83118
##              Prevalence : 0.20238
##          Detection Rate : 0.04667
##    Detection Prevalence : 0.07762
##       Balanced Accuracy : 0.59589
##
##        'Positive' Class : 1
##
```

Los F1score:

```
## f1 datos train 0.3333333
```

```
## f1 datos test 0.4262701
```

- 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

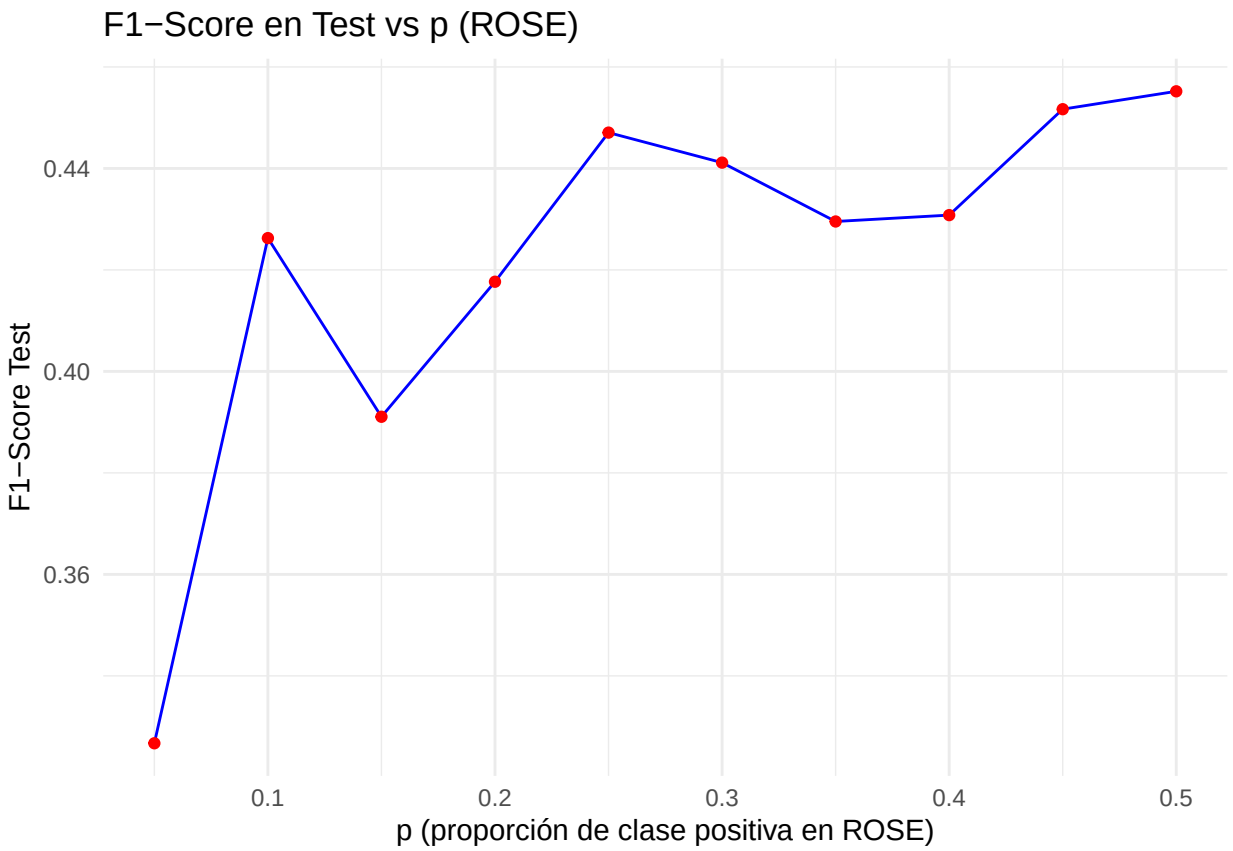
A continuación se buscará para diferentes niveles de balanceo dónde se obtienen mejores kpi con caret

```
##
## Attaching package: 'dplyr'

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

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

	p	F1_Train	F1_Test	Sensitivity_Train	Sensitivity_Test	Specificity_Train	Specificity_Test	Accuracy_Train	Accuracy_Test
Pos Pred Value	0.05	0.3560275	0.3267176	0.2780488	0.2517647	0.9249032	0.9265672	0.7895918	0.7900000
Pos Pred Value1	0.10	0.4471178	0.4262701	0.4351220	0.4047059	0.8647742	0.8746269	0.7748980	0.7795238
Pos Pred Value2	0.15	0.4344905	0.3910615	0.3785366	0.3294118	0.9037419	0.9098507	0.7938776	0.7923810
Pos Pred Value3	0.20	0.4507042	0.4176707	0.4058537	0.3670588	0.8954839	0.9008955	0.7930612	0.7928571
Pos Pred Value4	0.25	0.4621969	0.4470588	0.4741463	0.4470588	0.8472258	0.8597015	0.7691837	0.7761905
Pos Pred Value5	0.30	0.4676367	0.4411400	0.4546341	0.4188235	0.8704516	0.8782090	0.7834694	0.7852381
Pos Pred Value6	0.35	0.4905137	0.4295455	0.5170732	0.4447059	0.8436129	0.8411940	0.7753061	0.7609524
Pos Pred Value7	0.40	0.4906143	0.4308012	0.5609756	0.4870588	0.8080000	0.8035821	0.7563265	0.7395238
Pos Pred Value8	0.45	0.5060713	0.4516746	0.6302439	0.5552941	0.7723871	0.7707463	0.7426531	0.7271429
Pos Pred Value9	0.50	0.4913188	0.4551971	0.6487805	0.5976471	0.7375484	0.7391045	0.7189796	0.7104762



2.1 Conclusiones balanceo

Para valores bajos de p (< 0.2) el modelo casi no predice la clase positiva, dando F1 muy bajos.

Un balance cercano a p = 0.4 - 0.5 maximiza F1-Test y sensibilidad, siendo óptimo para detectar la clase minoritaria.

```
library(ggplot2)
library(reshape2)

# Seleccionar métricas para graficar
resultados_long <- melt(resultados, id.vars = "p",
                        measure.vars = c("F1_Test", "Sensitivity_Test", "Accuracy_Test"),
                        variable.name = "Metric", value.name = "Value")

# Gráfico
ggplot(resultados_long, aes(x = p, y = Value, color = Metric)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  labs(title = "Métricas Test vs p (ROSE)", x = "p (proporción clase positiva)", y = "Valor") +
  scale_color_manual(values = c("F1_Test" = "blue",
                                "Sensitivity_Test" = "red",
                                "Accuracy_Test" = "green")) +
  theme_minimal()
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
```

```
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```