Classification Tree bbdd imputado_bsd

Grupo 5

ÍNDICE

1 Base de datos reducido_plus SIN BALANCEAR

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

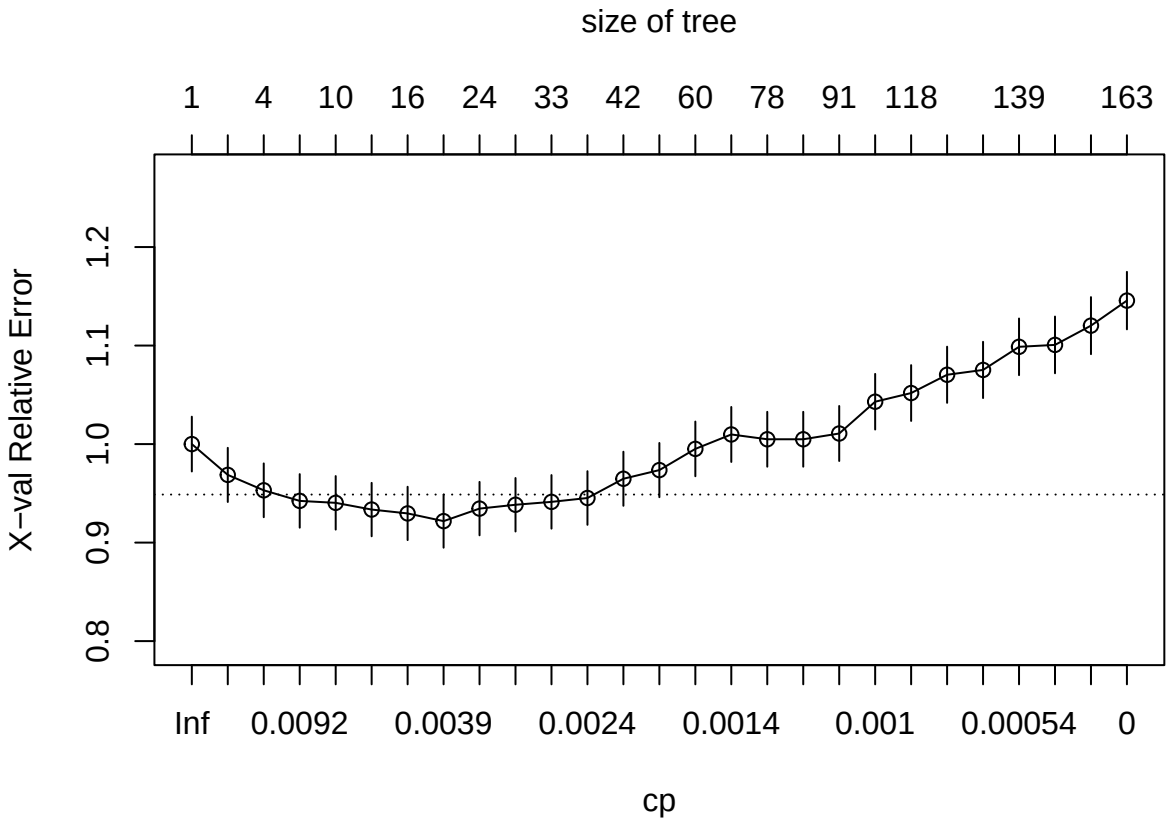
1.1 Método cp=0

```
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          SavingsAccountFlag     Tenure
## [19] TransactionFrequency
##
## Root node error: 1023/4900 = 0.20878
##
## n= 4900
##
##            CP nsplit rel error  xerror     xstd
## 1  0.02052786      0   1.00000 1.00000 0.027811
## 2  0.01857283      2   0.95894 0.96872 0.027485
## 3  0.01075269      3   0.94037 0.95308 0.027318
## 4  0.00782014      7   0.89736 0.94233 0.027201
## 5  0.00586510      9   0.88172 0.94037 0.027180
## 6  0.00488759     12   0.86217 0.93353 0.027105
## 7  0.00439883     15   0.84751 0.92962 0.027062
## 8  0.00342131     21   0.81623 0.92180 0.026975
## 9  0.00293255     23   0.80938 0.93451 0.027116
## 10 0.00260671     29   0.79179 0.93842 0.027159
## 11 0.00244379     32   0.78397 0.94135 0.027191
## 12 0.00228087     34   0.77908 0.94526 0.027233
## 13 0.00195503     41   0.75953 0.96481 0.027444
## 14 0.00162920     55   0.73216 0.97361 0.027537
## 15 0.00146628     59   0.72434 0.99511 0.027761
## 16 0.00136852     65   0.71554 1.00978 0.027910
## 17 0.00130336     77   0.69306 1.00489 0.027861
## 18 0.00122190     83   0.68524 1.00489 0.027861
## 19 0.00111716     90   0.67644 1.01075 0.027920
## 20 0.00097752    108   0.65494 1.04301 0.028241
## 21 0.00073314    117   0.64614 1.05181 0.028326
## 22 0.00065168    122   0.64223 1.07038 0.028504
## 23 0.00058651    128   0.63832 1.07527 0.028551
## 24 0.00048876    138   0.63245 1.09873 0.028769
## 25 0.00039101    142   0.63050 1.10068 0.028787
## 26 0.00019550    157   0.62463 1.12023 0.028964
## 27 0.00000000    162   0.62366 1.14565 0.029190
```

```
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 0.9687195 0.9530792 0.9423265 0.9403715 0.9335288 0.9296188 0.9217986
##         9        10        11        12        13        14        15        16
## 0.9345064 0.9384164 0.9413490 0.9452590 0.9648094 0.9736070 0.9951124 1.0097752
##        17        18        19        20        21        22        23        24
## 1.0048876 1.0048876 1.0107527 1.0430108 1.0518084 1.0703812 1.0752688 1.0987292
##        25        26        27
## 1.1006843 1.1202346 1.1456500
```

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

```
## 8
## 8
```

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

```
##          CP      nsplit   rel error      xerror        xstd
##  0.00342131 21.00000000  0.81622678  0.92179863  0.02697523
```

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

```
##         8
## 0.9487739
```

Los valores son bastante similados para el caso de la base de datos *balanceado plus* El mínimo error es 0.9341 en la posición 6, que corresponde a un árbol con 14 divisiones y un CP = 0.004491018.

```
tree2 <- prune(tree, cp = 0.003940887)
importance <- tree2$variable.importance
importance <- round(100*importance/sum(importance), 1)
importance
```

1.1.1.1 Cp mínimo

```
##                   Age          NumOfProducts                Balance
##                  40.2                   32.4                    7.5
##        EstimatedSalary                 Gender         IsActiveMember
##                   3.4                    2.7                    2.6
##              Geography            CreditScore   AvgTransactionAmount
##                   2.1                    1.8                    1.5
##             LoanStatus   TransactionFrequency          MaritalStatus
##                   1.0                    1.0                    0.7
## DigitalEngagementScore       NetPromoterScore        CustomerSegment
##                   0.6                    0.6                    0.5
##         EducationLevel        ComplaintsCount     SavingsAccountFlag
##                   0.5                    0.4                    0.3
##                Tenure
##                   0.2
```

Los resultados muestran que NumOfProducts (33.8%) y Age (38.9%) son las variables dominantes, explicando conjuntamente el 72.7% 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 (4.9 %) y Balance (8%), tienen un impacto considerablemente menor, mientras que factores como Geography (0.4%) resultan prácticamente irrelevantes para el modelo.

Matriz de confusión para train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
```

```
##          0 3751  709
##          1  126  314
##
##                Accuracy : 0.8296
##                  95% CI : (0.8188, 0.84)
##     No Information Rate : 0.7912
##     P-Value [Acc > NIR] : 7.408e-12
##
##                   Kappa : 0.3473
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.30694
##             Specificity : 0.96750
##          Pos Pred Value : 0.71364
##          Neg Pred Value : 0.84103
##              Prevalence : 0.20878
##          Detection Rate : 0.06408
##    Detection Prevalence : 0.08980
##       Balanced Accuracy : 0.63722
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1594  285
##          1   79  142
##
##                Accuracy : 0.8267
##                  95% CI : (0.8098, 0.8426)
##     No Information Rate : 0.7967
##     P-Value [Acc > NIR] : 0.000281
##
##                   Kappa : 0.3478
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.33255
##             Specificity : 0.95278
##          Pos Pred Value : 0.64253
##          Neg Pred Value : 0.84832
##              Prevalence : 0.20333
##          Detection Rate : 0.06762
##    Detection Prevalence : 0.10524
##       Balanced Accuracy : 0.64267
##
##        'Positive' Class : 1
##
```

F1 score

```
## f1 datos train 0.429255
```

```
## f1 datos test 0.4382716
```

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 (~5 puntos), 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                Balance
##                    48.8                   42.4                    4.0
##         IsActiveMember DigitalEngagementScore   AvgTransactionAmount
##                     3.5                    0.3                    0.3
##         EducationLevel          MaritalStatus            CreditScore
##                     0.2                    0.2                    0.1
##   TransactionFrequency        EstimatedSalary
##                     0.1                    0.1
```

El peso de las variables ha augmentado en algunos casos.

Matriz de confusión para train

Matriz de confusión para test

F1 score

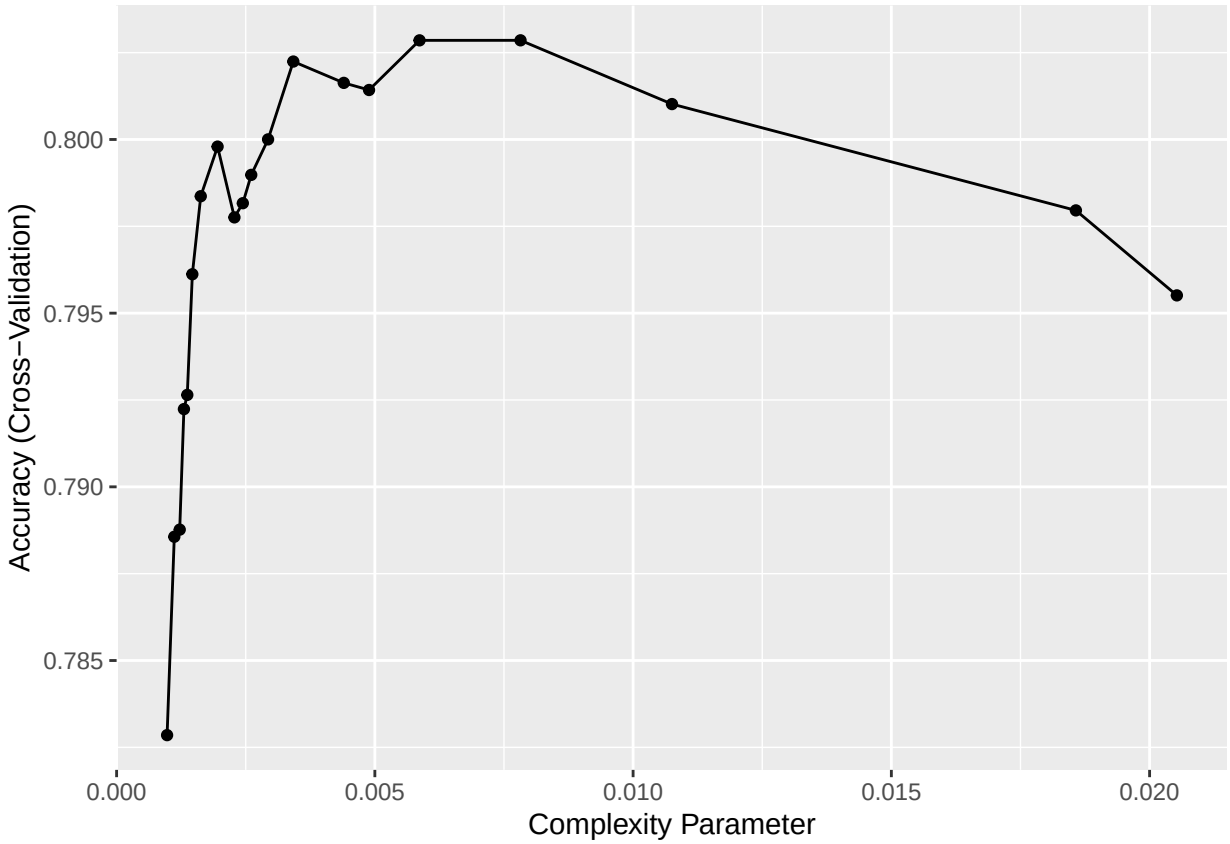
```
## f1 datos train 0.3633842
```

```
## f1 datos test 0.4149766
```

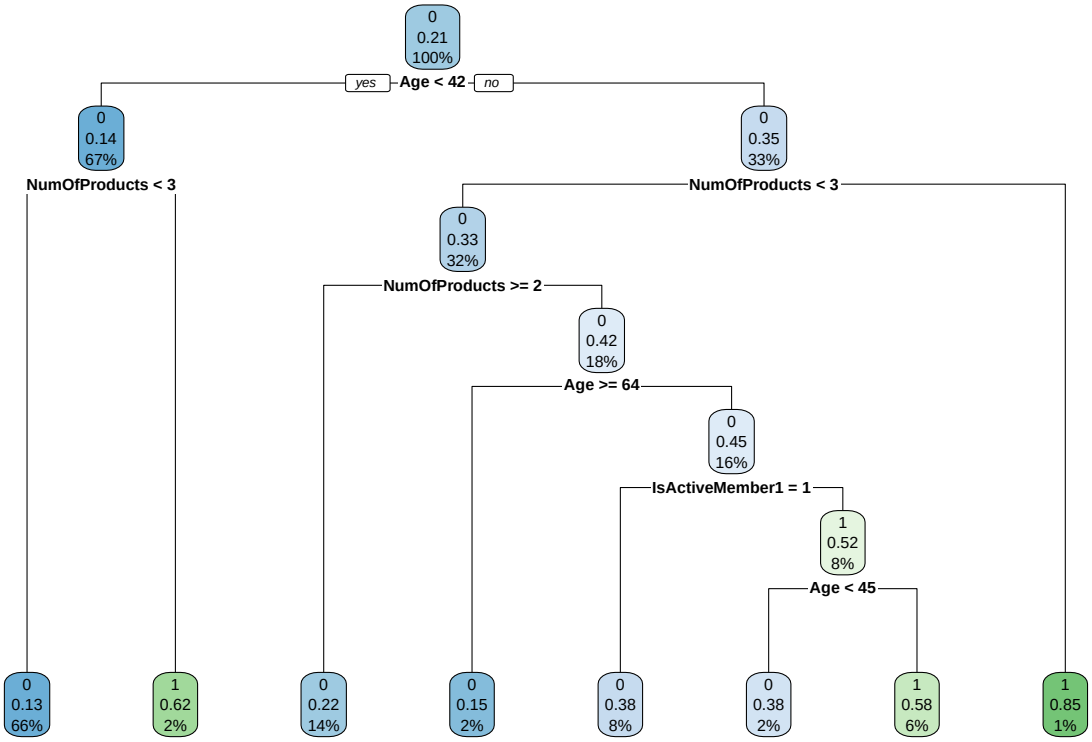
No hay overfitting, el f1 ha disminuido para este caso, no obstante los dos valores se podrian considerar parecidos

1.2 Método Caret

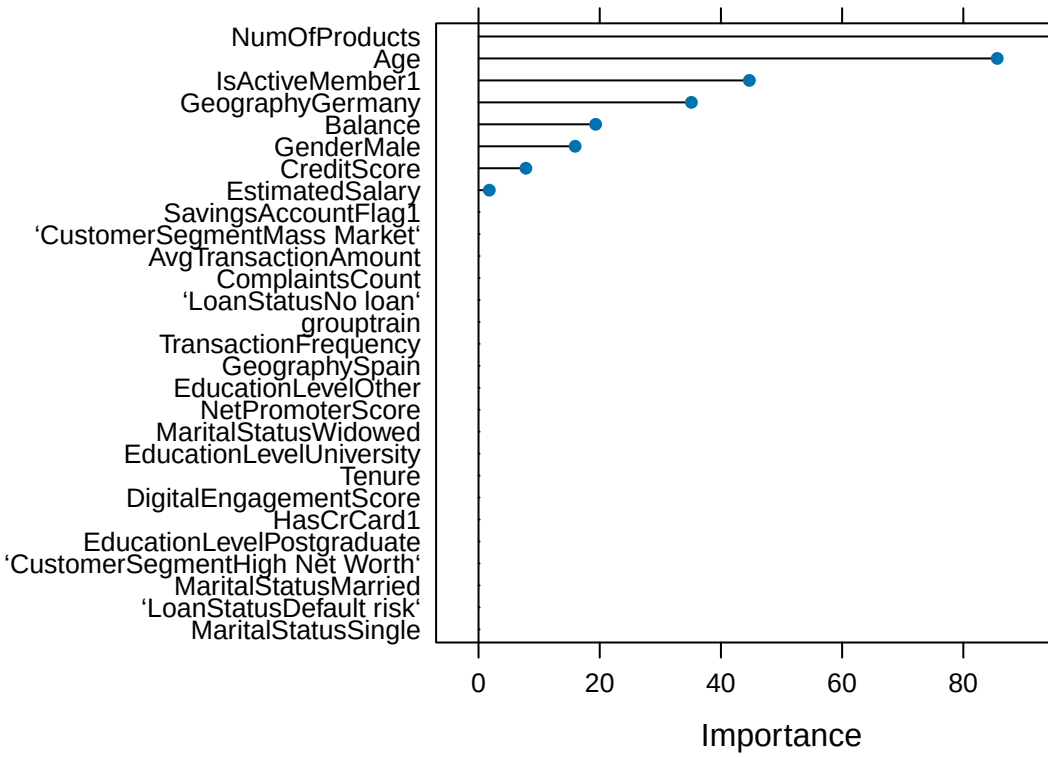
```
caret.rpart <- train(Exited ~ ., method = "rpart", data = train,
 tuneLength = 20,
 trControl = trainControl(method = "cv", number = 10))
ggplot(caret.rpart)
```




```
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 3720  761
##          1  157  262
##
##                Accuracy : 0.8127
##                  95% CI : (0.8014, 0.8235)
##     No Information Rate : 0.7912
##     P-Value [Acc > NIR] : 0.0001004
##
##                   Kappa : 0.2755
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.25611
##             Specificity : 0.95950
##          Pos Pred Value : 0.62530
##          Neg Pred Value : 0.83017
##              Prevalence : 0.20878
##          Detection Rate : 0.05347
##    Detection Prevalence : 0.08551
```

```
##       Balanced Accuracy : 0.60781
##
##        'Positive' Class : 1
##
```

Matriz de confusión datos test:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1592  294
##          1   81  133
##
##                Accuracy : 0.8214
##                  95% CI : (0.8044, 0.8376)
##     No Information Rate : 0.7967
##     P-Value [Acc > NIR] : 0.002319
##
##                   Kappa : 0.3231
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.31148
##             Specificity : 0.95158
##          Pos Pred Value : 0.62150
##          Neg Pred Value : 0.84411
##              Prevalence : 0.20333
##          Detection Rate : 0.06333
##    Detection Prevalence : 0.10190
##       Balanced Accuracy : 0.63153
##
##        'Positive' Class : 1
##
```

Los F1score:

```
## f1 datos train 0.4149766
```

```
## f1 datos test 0.3633842
```

- 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 reducido_plus BALANCEO

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

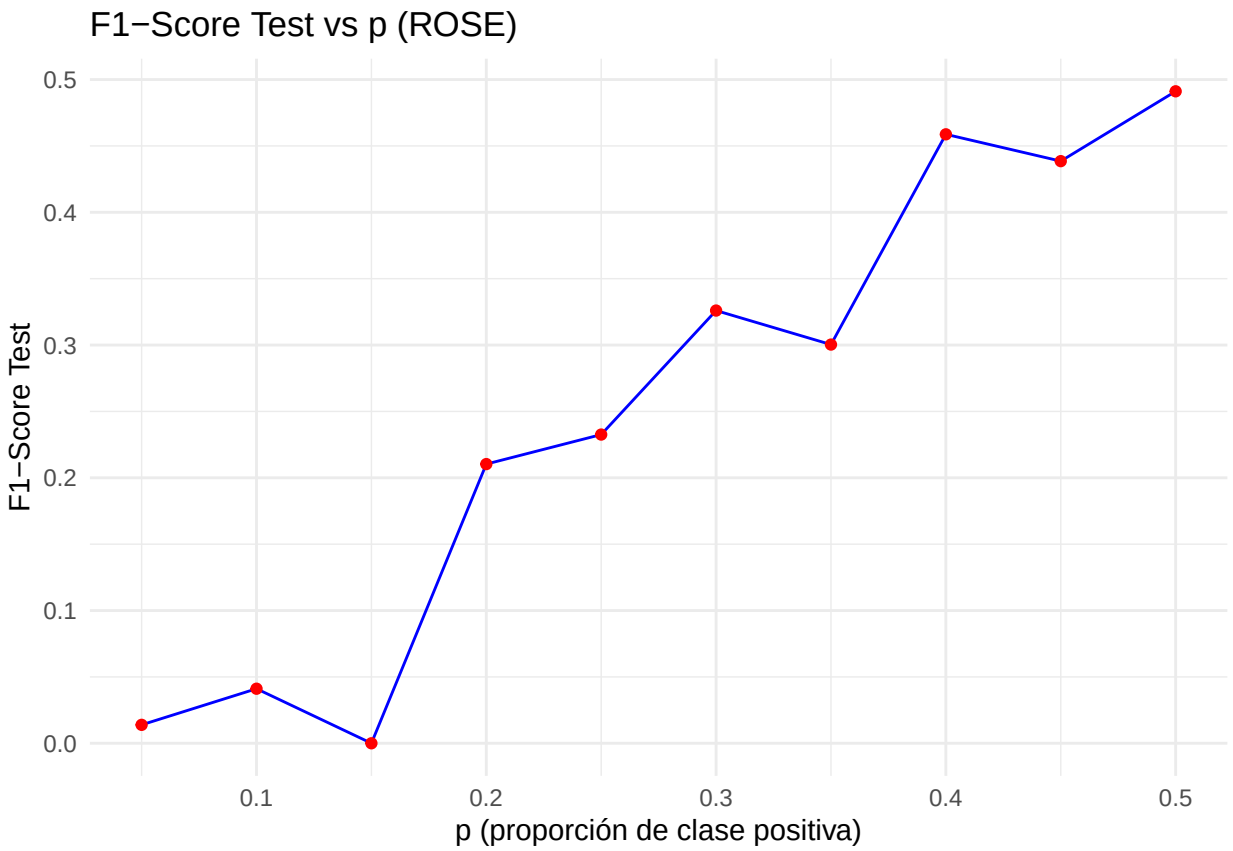
```
##
## Attaching package: 'dplyr'

## 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
Pos Pred Value	0.05	0.03262956	0.01388889	0.01661779	0.007025761
Pos Pred Value1	0.10	0.05502846	0.04109589	0.02834800	0.021077283
Sensitivity	0.15	0.00000000	0.00000000	0.00000000	0.000000000
Pos Pred Value2	0.20	0.19500000	0.21031746	0.11436950	0.124121780
Pos Pred Value3	0.25	0.20445177	0.23255814	0.12121212	0.140515222
Pos Pred Value4	0.30	0.31485588	0.32597623	0.20821114	0.224824356
Pos Pred Value5	0.35	0.27706283	0.30035336	0.17888563	0.199063232
Pos Pred Value6	0.40	0.40049140	0.45874126	0.31867058	0.384074941
Pos Pred Value7	0.45	0.46005775	0.43853821	0.46725318	0.463700234
Pos Pred Value8	0.50	0.48395426	0.49113924	0.64125122	0.681498829

	Specificity_Train	Specificity_Test	Accuracy_Train	Accuracy_Test
Pos Pred Value	0.9994841	0.9988045	0.7942857	0.7971429
Pos Pred Value1	0.9994841	0.9988045	0.7967347	0.8000000
Sensitivity	1.0000000	1.0000000	0.7912245	0.7966667
Pos Pred Value2	0.9845241	0.9856545	0.8028571	0.8104762
Pos Pred Value3	0.9829765	0.9826659	0.8030612	0.8114286
Pos Pred Value4	0.9698220	0.9605499	0.8108163	0.8109524
Pos Pred Value5	0.9703379	0.9677227	0.8051020	0.8114286
Pos Pred Value6	0.9280371	0.9258816	0.8008163	0.8157143
Pos Pred Value7	0.8511736	0.8338314	0.7710204	0.7585714
Pos Pred Value8	0.7338148	0.7208607	0.7144898	0.7128571



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.
```