

Classification Tree bdd imputado_bsd

Grupo 5

ÍNDICE

1	Base de datos transformada SIN BALANCEAR	1
1.1	Método cp=0.....	1
1.1.1	Elección cp óptimo	3
1.2	Método Caret	7
1.3	Conclusiones para data imputado sin balancear	11
2	Base de datos transformada BALANCEO	12
2.1	Conclusiones balanceo	13

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,]
```

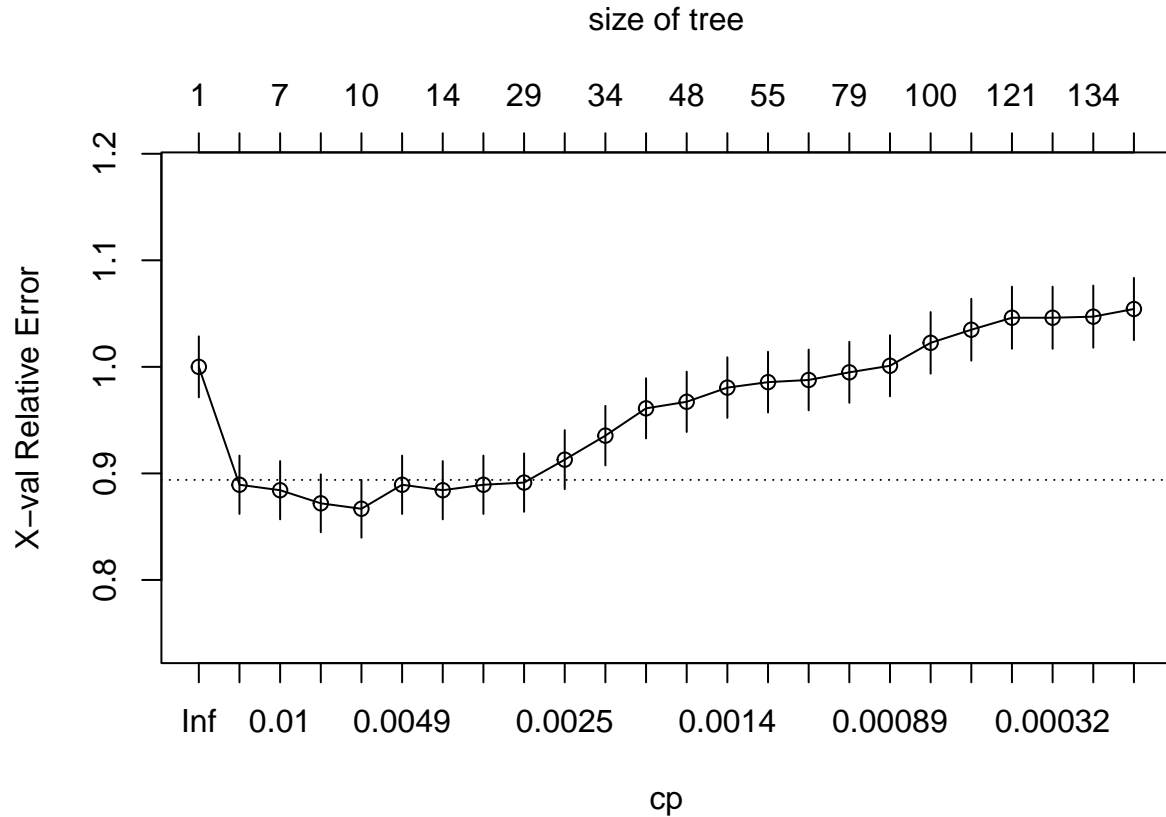
1.1 Método cp=0

```
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           IsActiveMember      LoanStatus
## [13] MaritalStatus        NetPromoterScore   NumOfProducts_grupo
## [16] SavingsAccountFlag   Tenure              TransactionFrequency
##
## Root node error: 976/4900 = 0.19918
##
## n= 4900
##
##          CP nsplit rel error  xerror    xstd
## 1  0.01775956      0   1.00000 1.00000 0.028645
## 2  0.01229508      5   0.89139 0.88934 0.027382
## 3  0.00870902      6   0.87910 0.88422 0.027320
## 4  0.00717213      8   0.86168 0.87193 0.027170
## 5  0.00512295      9   0.85451 0.86680 0.027107
## 6  0.00461066     10   0.84939 0.88934 0.027382
## 7  0.00409836     13   0.83504 0.88422 0.027320
## 8  0.00358607     24   0.78279 0.88934 0.027382
## 9  0.00307377     28   0.76742 0.89139 0.027407
## 10 0.00204918     29   0.76434 0.91291 0.027664
## 11 0.00179303     33   0.75615 0.93545 0.027926
## 12 0.00170765     38   0.74590 0.96107 0.028217
## 13 0.00153689     47   0.73053 0.96721 0.028286
## 14 0.00136612     51   0.72439 0.98053 0.028433
## 15 0.00128074     54   0.72029 0.98566 0.028489
## 16 0.00122951     73   0.68443 0.98770 0.028511
## 17 0.00102459     78   0.67828 0.99488 0.028589
## 18 0.00076844     95   0.65676 1.00102 0.028656
## 19 0.00055887     99   0.65369 1.02254 0.028884
## 20 0.00051230    110   0.64754 1.03484 0.029013
## 21 0.00034153    120   0.64242 1.04611 0.029129
## 22 0.00029274    126   0.64037 1.04611 0.029129
## 23 0.00025615    133   0.63832 1.04713 0.029139
## 24 0.00000000    137   0.63730 1.05430 0.029213
```

```
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 0.8893443 0.8842213 0.8719262 0.8668033 0.8893443 0.8842213 0.8893443
##          9         10         11         12         13         14         15         16
## 0.8913934 0.9129098 0.9354508 0.9610656 0.9672131 0.9805328 0.9856557 0.9877049
##         17         18         19         20         21         22         23         24
## 0.9948770 1.0010246 1.0225410 1.0348361 1.0461066 1.0461066 1.0471311 1.0543033
```

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

```
## 5
## 5
```

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

```
##          CP      nsplit  rel error      xerror      xstd
## 0.005122951 9.000000000 0.854508197 0.866803279 0.027106859
```

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

```
##          5
## 0.8939101
```

Los valores son bastante similares 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

```
##      NumOfProducts_grupo      Age      Balance
##              40.2          39.7          7.6
##      IsActiveMember  TransactionFrequency  AvgTransactionAmount
##              5.4              2.3              0.9
##              Tenure  DigitalEngagementScore      Geography
##              0.9              0.8              0.8
##      EstimatedSalary      EducationLevel      CreditScore
##              0.7              0.3              0.2
##              Gender
##              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 3380  495
##      1  544  481
##
##      Accuracy : 0.788
```

```

##          95% CI : (0.7762, 0.7993)
##      No Information Rate : 0.8008
##      P-Value [Acc > NIR] : 0.9880
##
##          Kappa : 0.3476
##
##      McNemar's Test P-Value : 0.1365
##
##          Sensitivity : 0.49283
##          Specificity : 0.86137
##          Pos Pred Value : 0.46927
##          Neg Pred Value : 0.87226
##          Prevalence : 0.19918
##          Detection Rate : 0.09816
##          Detection Prevalence : 0.20918
##          Balanced Accuracy : 0.67710
##
##          'Positive' Class : 1
##

```

Matriz de confusión para test

```

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##          0 1409  280
##          1  217  194
##
##          Accuracy : 0.7633
##          95% CI : (0.7446, 0.7814)
##      No Information Rate : 0.7743
##      P-Value [Acc > NIR] : 0.889592
##
##          Kappa : 0.2895
##
##      McNemar's Test P-Value : 0.005418
##
##          Sensitivity : 0.40928
##          Specificity : 0.86654
##          Pos Pred Value : 0.47202
##          Neg Pred Value : 0.83422
##          Prevalence : 0.22571
##          Detection Rate : 0.09238
##          Detection Prevalence : 0.19571
##          Balanced Accuracy : 0.63791
##
##          'Positive' Class : 1
##

```

F1 score

```
## f1 datos train 0.4807596
```

```
## f1 datos test 0.4384181
```

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      Balance
##           46.1      38.8      6.8
##      IsActiveMember DigitalEngagementScore AvgTransactionAmount
##           6.7      0.6      0.4
##      EstimatedSalary      Gender TransactionFrequency
##           0.3      0.2      0.0
##           CreditScore
##           0.0
```

El peso de las variables ha aumentado en algunos casos.

Matriz de confusión para train

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 3394  533
##           1  530  443
##
##           Accuracy : 0.7831
##           95% CI : (0.7713, 0.7945)
##           No Information Rate : 0.8008
##           P-Value [Acc > NIR] : 0.9990
##
##           Kappa : 0.3192
##
## Mcnemar's Test P-Value : 0.9511
##
##           Sensitivity : 0.45389
##           Specificity : 0.86493
##           Pos Pred Value : 0.45529
##           Neg Pred Value : 0.86427
##           Prevalence : 0.19918
##           Detection Rate : 0.09041
##           Detection Prevalence : 0.19857
##           Balanced Accuracy : 0.65941
```

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

Matriz de confusión para test

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction    0    1
##      0 1422  299
##      1  204  175
##
##      Accuracy : 0.7605
##      95% CI : (0.7416, 0.7786)
##      No Information Rate : 0.7743
##      P-Value [Acc > NIR] : 0.9374
##
##      Kappa : 0.2624
##
##      McNemar's Test P-Value : 2.774e-05
##
##      Sensitivity : 0.36920
##      Specificity : 0.87454
##      Pos Pred Value : 0.46174
##      Neg Pred Value : 0.82626
##      Prevalence : 0.22571
##      Detection Rate : 0.08333
##      Detection Prevalence : 0.18048
##      Balanced Accuracy : 0.62187
##
##      'Positive' Class : 1
##
```

F1 score

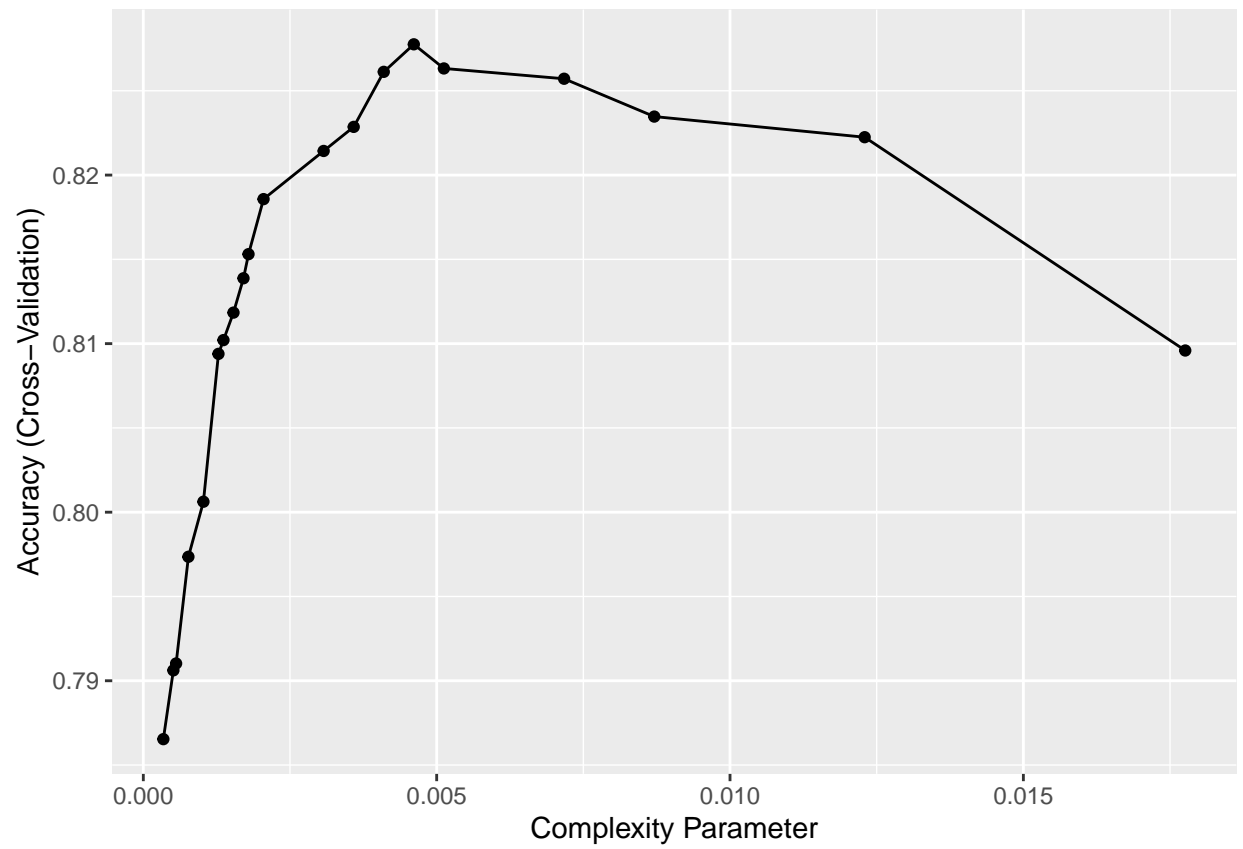
```
## f1 datos train 0.4545921
```

```
## f1 datos test 0.4103165
```

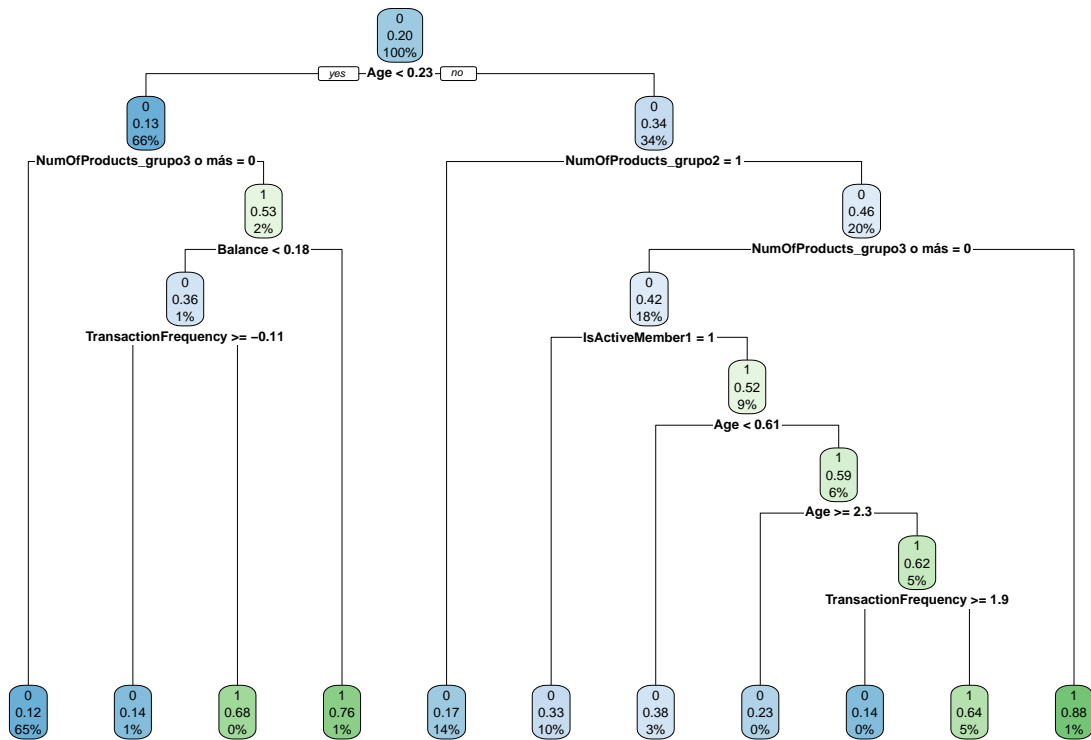
No hay overfitting, el f1 ha disminuido para este caso, no obstante los dos valores se podrían considerar parecidos

1.2 Método Caret

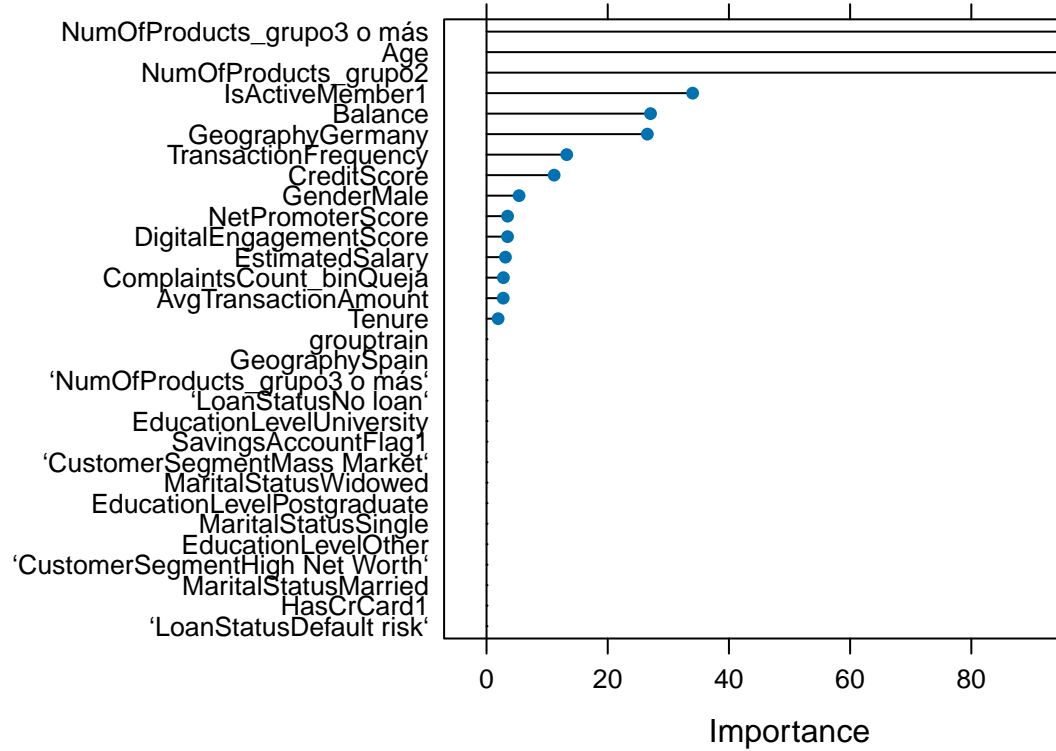
```
caret.rpart <- train(Exited ~ ., method = "rpart", data = train,
  tuneLength = 20,
  trControl = trainControl(method = "cv", number = 10))
ggplot(caret.rpart)
```



```
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 1422  299
##           1  204  175
##
##           Accuracy : 0.7605
##           95% CI : (0.7416, 0.7786)
##           No Information Rate : 0.7743
##           P-Value [Acc > NIR] : 0.9374
##
##           Kappa : 0.2624
##
##           McNemar's Test P-Value : 2.774e-05
##
##           Sensitivity : 0.36920
##           Specificity : 0.87454
##           Pos Pred Value : 0.46174
##           Neg Pred Value : 0.82626
##           Prevalence : 0.22571
##           Detection Rate : 0.08333
##           Detection Prevalence : 0.18048
```

```
##          Balanced Accuracy : 0.62187
##
##          'Positive' Class : 1
##
```

Matriz de confusión datos test:

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##          0 1564  385
##          1   62   89
##
##          Accuracy : 0.7871
##          95% CI : (0.769, 0.8045)
##    No Information Rate : 0.7743
##    P-Value [Acc > NIR] : 0.0826
##
##          Kappa : 0.1972
##
##    McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.18776
##          Specificity : 0.96187
##          Pos Pred Value : 0.58940
##          Neg Pred Value : 0.80246
##          Prevalence : 0.22571
##          Detection Rate : 0.04238
##    Detection Prevalence : 0.07190
##          Balanced Accuracy : 0.57482
##
##          'Positive' Class : 1
##
```

Los F1score:

```
## f1 datos train 0.2848
```

```
## f1 datos test 0.4103165
```

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

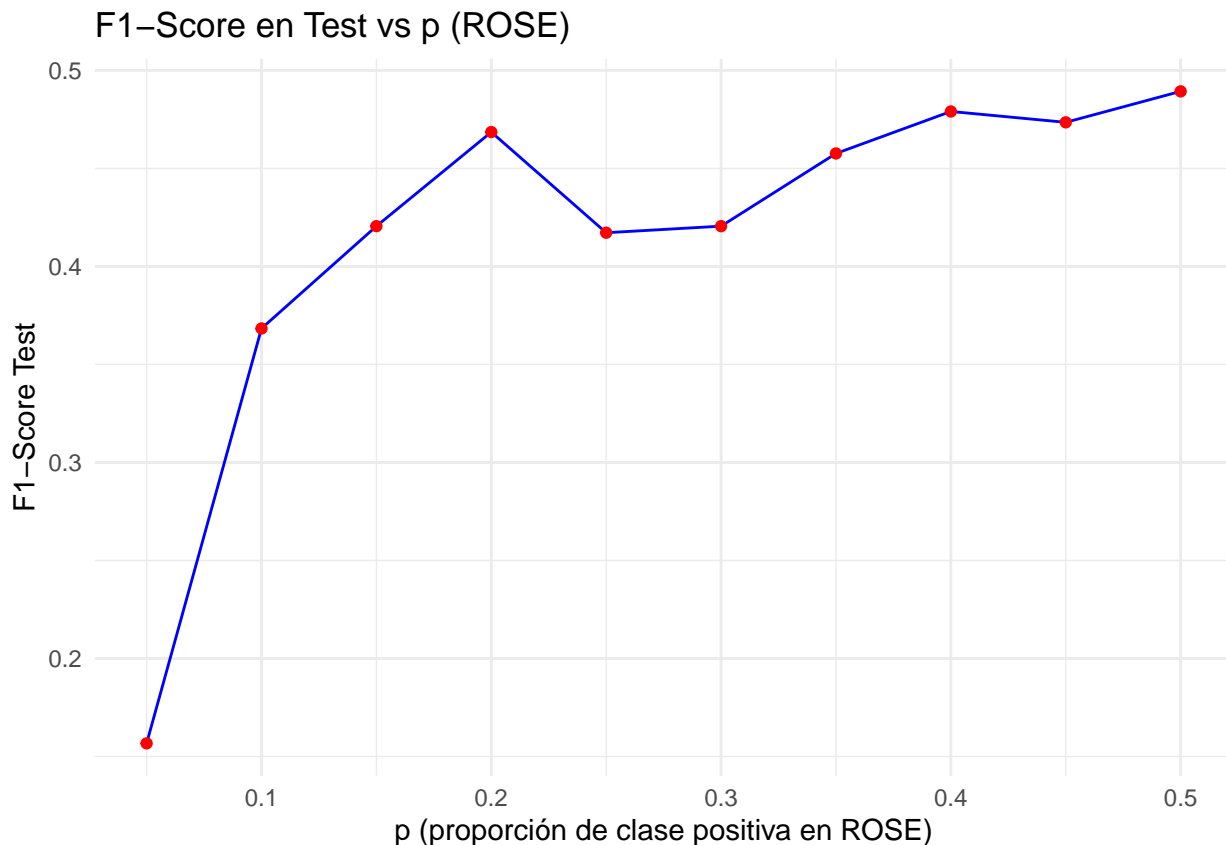
1.3 Conclusiones para data imputado sin balancear

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	F1_Train	F1_Test	Sensitivity_Train	Sensitivity_Test
##	Pos Pred Value	0.05	0.1814947	0.1567164	0.1045082	0.08860759
##	Pos Pred Value1	0.10	0.4325843	0.3683528	0.3944672	0.29957806
##	Pos Pred Value2	0.15	0.4647436	0.4205379	0.4456967	0.36286920
##	Pos Pred Value3	0.20	0.4681275	0.4685466	0.4815574	0.45569620
##	Pos Pred Value4	0.25	0.4517203	0.4171934	0.4170082	0.34810127
##	Pos Pred Value5	0.30	0.4647436	0.4205379	0.4456967	0.36286920
##	Pos Pred Value6	0.35	0.4693539	0.4576271	0.4354508	0.39873418
##	Pos Pred Value7	0.40	0.4867535	0.4790419	0.5553279	0.50632911
##	Pos Pred Value8	0.45	0.4838832	0.4735043	0.6536885	0.58438819
##	Pos Pred Value9	0.50	0.4801212	0.4893071	0.6495902	0.60337553
##			Specificity_Train	Specificity_Test	Accuracy_Train	Accuracy_Test
##	Pos Pred Value		0.9882773	0.9876999	0.8122449	0.7847619
##	Pos Pred Value1		0.8932212	0.9046740	0.7938776	0.7680952
##	Pos Pred Value2		0.8825178	0.8942189	0.7955102	0.7742857
##	Pos Pred Value3		0.8567788	0.8573186	0.7820408	0.7666667
##	Pos Pred Value4		0.8932212	0.9065191	0.7983673	0.7804762
##	Pos Pred Value5		0.8825178	0.8942189	0.7955102	0.7742857
##	Pos Pred Value6		0.8955148	0.8997540	0.8038776	0.7866667
##	Pos Pred Value7		0.8193170	0.8228782	0.7667347	0.7514286
##	Pos Pred Value8		0.7392966	0.7423124	0.7222449	0.7066667
##	Pos Pred Value9		0.7372579	0.7484625	0.7197959	0.7157143



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.
```

