CART reducido_plus

2025-11-20

```r
load("~/GitHub/Mineria/DATA/dataaaaaaaaaaaaaa.RData")
library(rpart)
library(caret)
```

```
## Cargando paquete requerido: ggplot2

## Cargando paquete requerido: lattice
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.5.2
```

```r
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,]
```

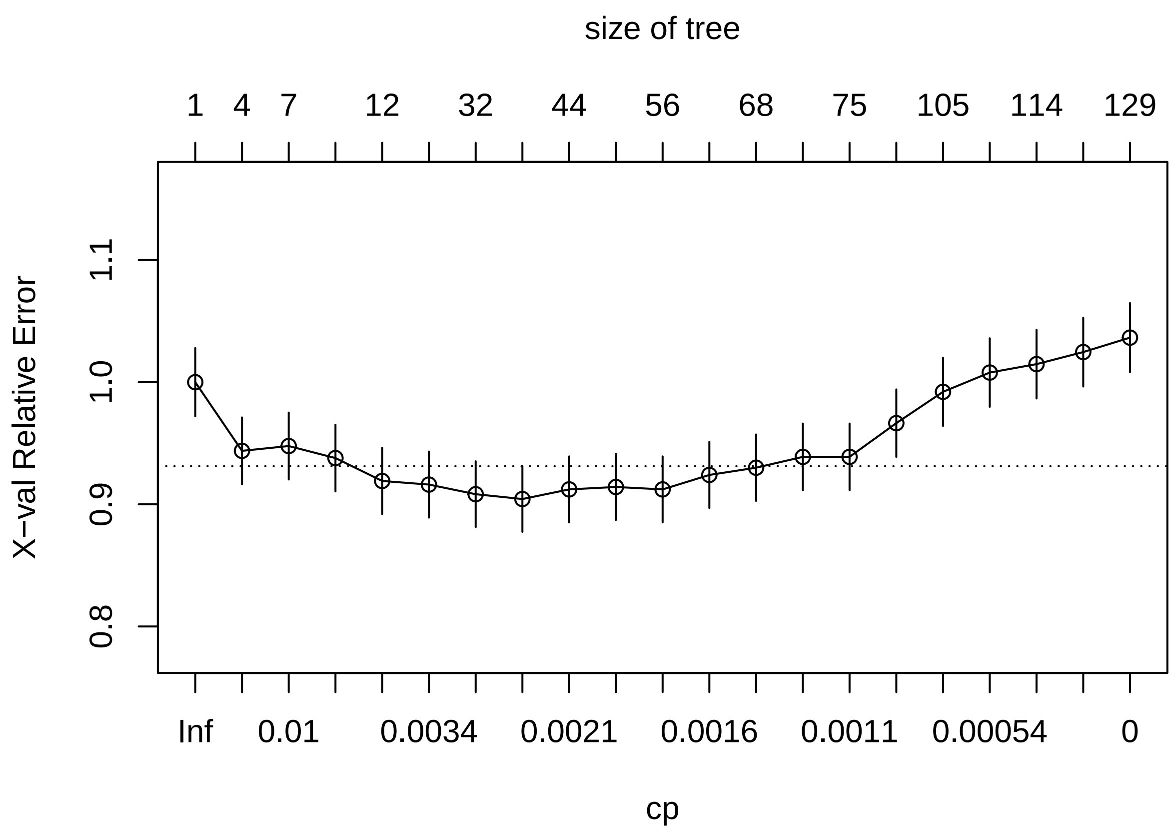
Método cp=0

```r
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: 1014/4900 = 0.20694
##
## n= 4900
##
##              CP nsplit rel error  xerror     xstd
## 1   0.02366864      0   1.00000 1.00000 0.027966
## 2   0.01084813      3   0.92899 0.94379 0.027367
```

```
## 3  0.00986193      6   0.89645 0.94773 0.027411
## 4  0.00788955      9   0.86686 0.93787 0.027302
## 5  0.00394477     11   0.85108 0.91913 0.027093
## 6  0.00295858     17   0.82742 0.91617 0.027060
## 7  0.00246548     31   0.78205 0.90828 0.026970
## 8  0.00230112     39   0.75937 0.90434 0.026925
## 9  0.00197239     43   0.74951 0.91223 0.027015
## 10 0.00177515     50   0.73570 0.91420 0.027037
## 11 0.00164366     55   0.72682 0.91223 0.027015
## 12 0.00147929     62   0.71400 0.92406 0.027148
## 13 0.00131492     67   0.70611 0.92998 0.027215
## 14 0.00123274     70   0.70217 0.93886 0.027313
## 15 0.00098619     74   0.69724 0.93886 0.027313
## 16 0.00071723     86   0.68343 0.96647 0.027613
## 17 0.00059172    104   0.66765 0.99211 0.027884
## 18 0.00049310    109   0.66469 1.00789 0.028047
## 19 0.00032873    113   0.66272 1.01479 0.028118
## 20 0.00021915    119   0.66075 1.02465 0.028218
## 21 0.00000000    128   0.65878 1.03649 0.028336
```

```
plotcp(tree)
```



Elección cp óptimo

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

```
##         1         2         3         4         5         6         7         8
## 1.0000000 0.9437870 0.9477318 0.9378698 0.9191321 0.9161736 0.9082840 0.9043393
##         9        10        11        12        13        14        15        16
## 0.9122288 0.9142012 0.9122288 0.9240631 0.9299803 0.9388560 0.9388560 0.9664694
##        17        18        19        20        21
## 0.9921105 1.0078895 1.0147929 1.0246548 1.0364892
```

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

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

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

```
##          CP      nsplit   rel error      xerror        xstd
##  0.002301118 39.000000000  0.759368836  0.904339250  0.026924883
```

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

```
##         8
## 0.9312641
```

Cp mínimo

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

```
##                 Age NumOfProducts_grupo             Balance      IsActiveMember
##                43.0                30.5                10.3                 6.7
##           Geography     EstimatedSalary         CreditScore              Gender
##                 3.4                 3.2                 2.6                 0.3
```

Matriz de confusión para train

```r
p <- predict(tree2, train, type = 'class')
(conf_train<-confusionMatrix(p, train$Exited, positive="1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3711  664
##          1  175  350
##
##                Accuracy : 0.8288
##                  95% CI : (0.8179, 0.8392)
##     No Information Rate : 0.7931
##     P-Value [Acc > NIR] : 1.564e-10
##
##                   Kappa : 0.3652
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.34517
##             Specificity : 0.95497
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.84823
##              Prevalence : 0.20694
##          Detection Rate : 0.07143
##    Detection Prevalence : 0.10714
##       Balanced Accuracy : 0.65007
##
##        'Positive' Class : 1
##
```

Matriz de confusión para test

```r
p2 <- predict(tree2, test, type = 'class')
(conf_test<-confusionMatrix(p2, test$Exited, positive="1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1560  314
##          1  104  122
##
##                Accuracy : 0.801
##                  95% CI : (0.7832, 0.8178)
##     No Information Rate : 0.7924
##     P-Value [Acc > NIR] : 0.1734
##
##                   Kappa : 0.2643
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.2798
##             Specificity : 0.9375
##          Pos Pred Value : 0.5398
##          Neg Pred Value : 0.8324
```

```
##              Prevalence : 0.2076
##          Detection Rate : 0.0581
##    Detection Prevalence : 0.1076
##       Balanced Accuracy : 0.6087
##
##        'Positive' Class : 1
##
```

F1 score

```
f1_score <- function(cm){
precision <- cm$byClass["Precision"]
recall <- cm$byClass["Sensitivity"]
f1 <- 2 * (precision * recall) / (precision + recall)
return(as.numeric(f1))
}
print(f1_train <- f1_score(conf_train))
```

```
## [1] 0.4548408
```

```
print(f1_test <- f1_score(conf_test))
```

```
## [1] 0.3685801
```

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

Cp mínimo+ error estándar

```
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
```

```
##                 Age NumOfProducts_grupo             Balance      IsActiveMember
##                43.8                33.3                 8.7                 7.4
##           Geography     EstimatedSalary         CreditScore              Gender
##                 3.3                 2.4                 0.9                 0.3
```

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

```
p <- predict(tree3, train, type = 'class')
conf_train<-confusionMatrix(p, train$Exited, positive="1")
```

Matriz de confusión para test

```r
p2 <- predict(tree3, test, type = 'class')
conf_test<-confusionMatrix(p2, test$Exited, positive="1")
```

F1 score

```r
print(f1_train <- f1_score(conf_train))
```

```
## [1] 0.4125255
```

```r
print(f1_test <- f1_score(conf_test))
```

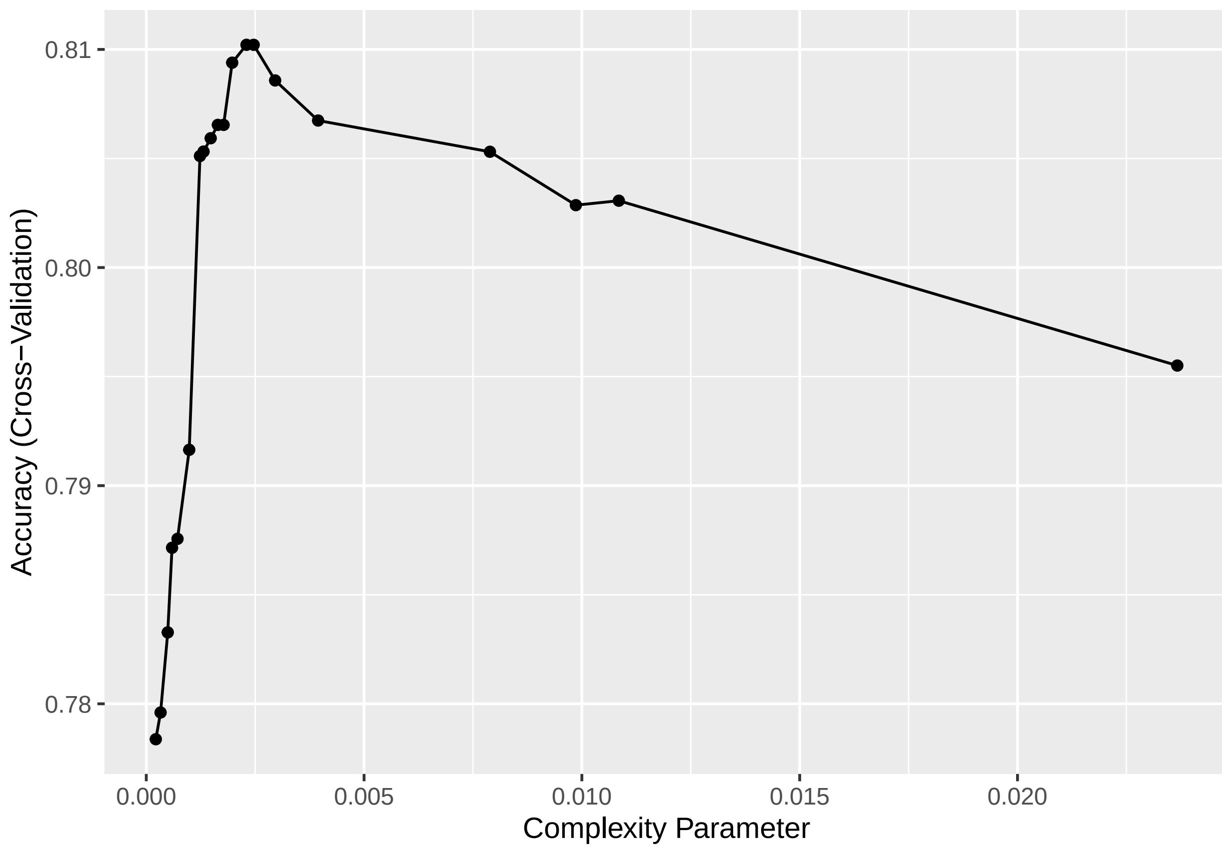
```
## [1] 0.3391442
```

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

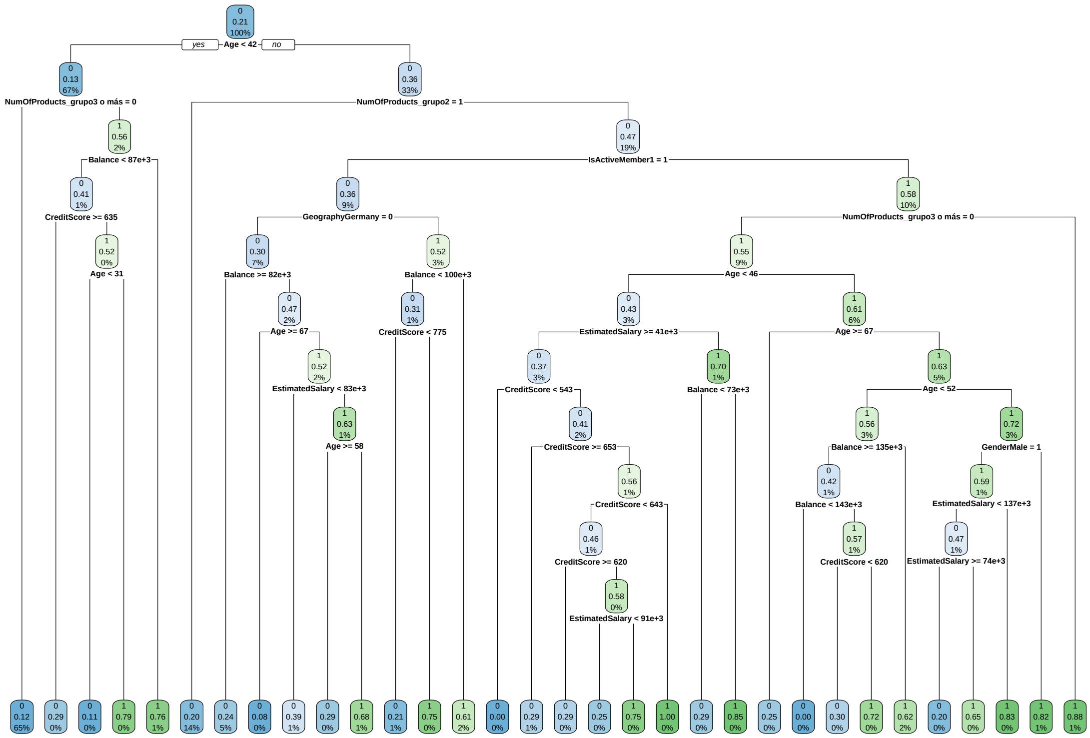
Método Caret

```r
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.
```

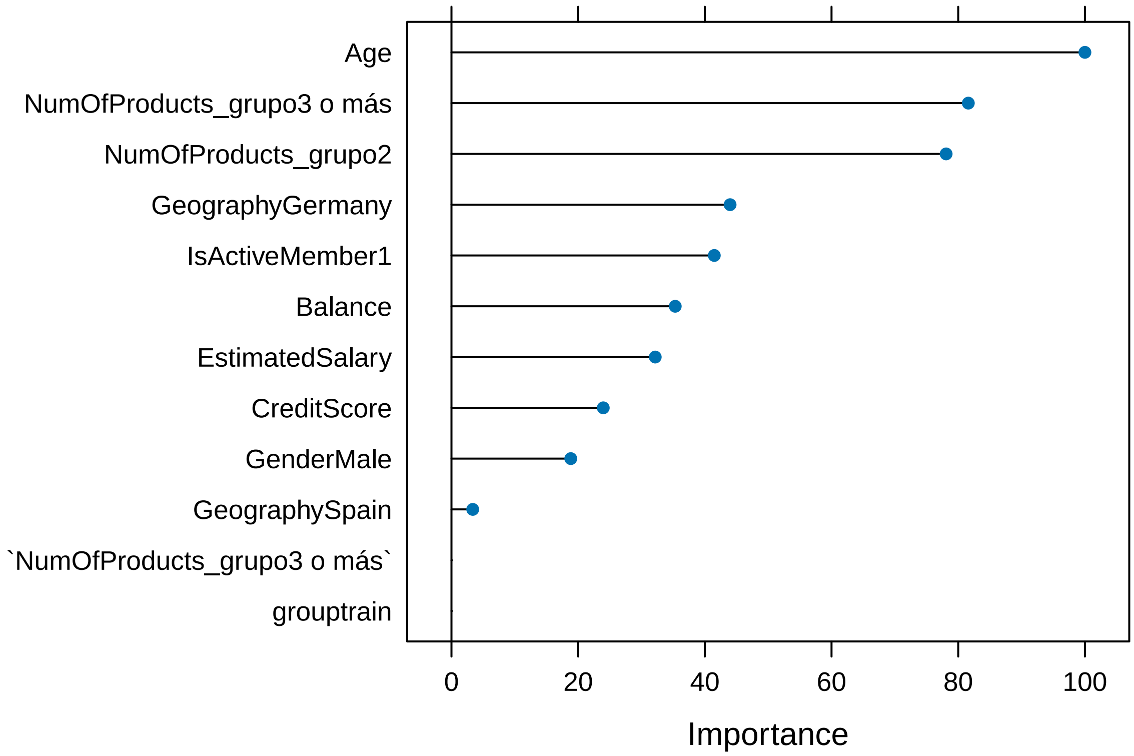



```r
rpart.plot(caret.rpart$finalModel)
```



Importancia de las variables:

```
var.imp <- varImp(caret.rpart)
plot(var.imp)
```



Las predicciones:

Matriz de confusión datos train

```
pred1 <- predict(caret.rpart, newdata = train)
(conf_test<-confusionMatrix(pred1, train$Exited, positive="1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3746  653
##          1  140  361
##
##                Accuracy : 0.8382
##                  95% CI : (0.8275, 0.8484)
##     No Information Rate : 0.7931
##     P-Value [Acc > NIR] : 5.965e-16
##
##                   Kappa : 0.3936
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.35602
##             Specificity : 0.96397
##          Pos Pred Value : 0.72056
##          Neg Pred Value : 0.85156
```

```
##               Prevalence : 0.20694
##           Detection Rate : 0.07367
##     Detection Prevalence : 0.10224
##        Balanced Accuracy : 0.65999
##
##         'Positive' Class : 1
##
```

Matriz de confusión datos test:

```
pred <- predict(caret.rpart, newdata = test)
(conf_train<-confusionMatrix(pred, test$Exited, positive="1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1559  319
##          1  105  117
##
##                Accuracy : 0.7981
##                  95% CI : (0.7803, 0.8151)
##     No Information Rate : 0.7924
##     P-Value [Acc > NIR] : 0.2691
##
##                   Kappa : 0.2506
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.26835
##             Specificity : 0.93690
##          Pos Pred Value : 0.52703
##          Neg Pred Value : 0.83014
##              Prevalence : 0.20762
##          Detection Rate : 0.05571
##    Detection Prevalence : 0.10571
##       Balanced Accuracy : 0.60262
##
##        'Positive' Class : 1
##
```

Los F1score:

```
print(f1_train <- f1_score(conf_train))
```

```
## [1] 0.3556231
```

```
print(f1_test <- f1_score(conf_test))
```

```
## [1] 0.4765677
```

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

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.