Árboles de Decisión

Siguiendo con los modelos discriminantes, en este apartado se analizará el algoritmo de los Árboles de Decisión, CART en adelante, con el mismo propósito específico: la clasificación de clientes en categorías de riesgo crediticio. En particular, nos enfocaremos en discendir entre aquellos clientes que puedan tener dificultades de pago y aquellos que son financieramente solventes.

El algoritmo de Árboles de Decisión se revela como una herramienta particularmente poderosa en este contexto, ya que su capacidad para modelar relaciones complejas entre variables puede proporcionar insights para la toma de decisiones financieras. Exploraremos cómo el algoritmo selecciona de manera inteligente las variables más influyentes para segmentar eficientemente el conjunto de datos, permitiendo la identificación de patrones que podrían indicar riesgos financieros.

Algoritmo En este contexto, la estructura de un Árbol de Decisión se modela de forma análoga a un proceso de decisiones estratégicas:

- Cada nodo interno del árbol representa una evaluación crítica sobre un atributo financiero específico. Estas evaluaciones sirven como puntos clave para discernir las distintas condiciones financieras de los clientes.
- Las ramas que se desprenden de cada nodo interno representan las diferentes trayectorias que un cliente puede seguir según el resultado de la evaluación realizada en ese nodo.
- Las hojas del árbol en el contexto financiero contienen la información crucial: la etiqueta o el valor predicho relacionado con la capacidad del cliente para afrontar compromisos financieros. Esto puede manifestarse como una clasificación de riesgo, como "solvente" o "en riesgo", proporcionando una guía clara para las decisiones crediticias.

Así pues, a continuación se procede a realizar dicho análisis discriminante.

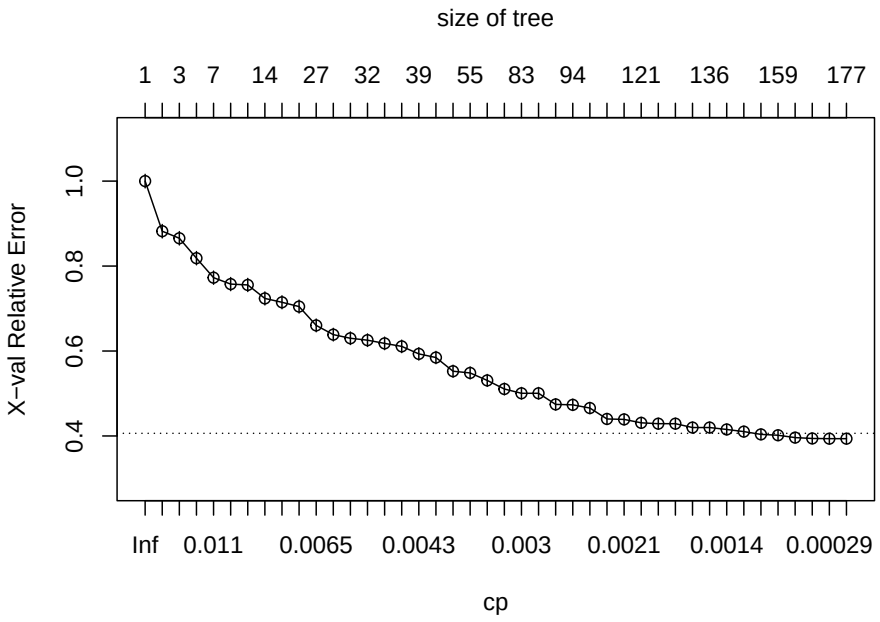
Desarrollo del CART

Para inciar el desarrollo del modelo, el primer paso es encontrar el valor óptimo del complexity parameter, o parámetro de complejidad, que controla la cantidad de ramificaciones y nodos terminales en el árbol. Este parámetro juega un papel importante en la regularización del árbol, evitando que éste se vuelva demasiado complejo y se adapte demasiado a los datos de entrenamiento, lo que podría resultar en un sobreajuste del modelo.

Para encontrar este valor óptimo de complejidad, se ejecutarán dos funciones diferentes. Por un lado, se utilizará la función `rpart` con cp=0 que construye el árbol utilizando la validación cruzada para podarlo. Por otro lado, se hará uso de la función `train` de la libreria CARET, que emplea bootstrap de las observaciones para seleccionar el valor óptimo del hiperparámetro cp, a través de emplear una valdiación cruzada.

Modelo de clasificación con `rpart` Obtenemos el árbol de clasifiación con todas las opciones por defecto (split="gini", etc), pero con un cp=0 para encontrar el valor óptimo del (hiper)parámetro de complejidad. Se construye un árbol de decisión completo y se emplea validación cruzada para podarlo.

Figura 122: Evaluación del error (reescalado) de validación cruzada en función del parámetro de complejidad.



El gráfico indica que la tasa de error para la 10-fold-cross-validation se estabiliza cuando el número de nodos hoja alcanza aproximadamente 128. Para determinar el tamaño ideal del árbol, la línea punteada representa el mínimo de la curva con un error estándar adicional de 1, que es una práctica común en la poda de árboles de decisión, donde se selecciona el árbol más pequeño dentro de 1 SE del mínimo.

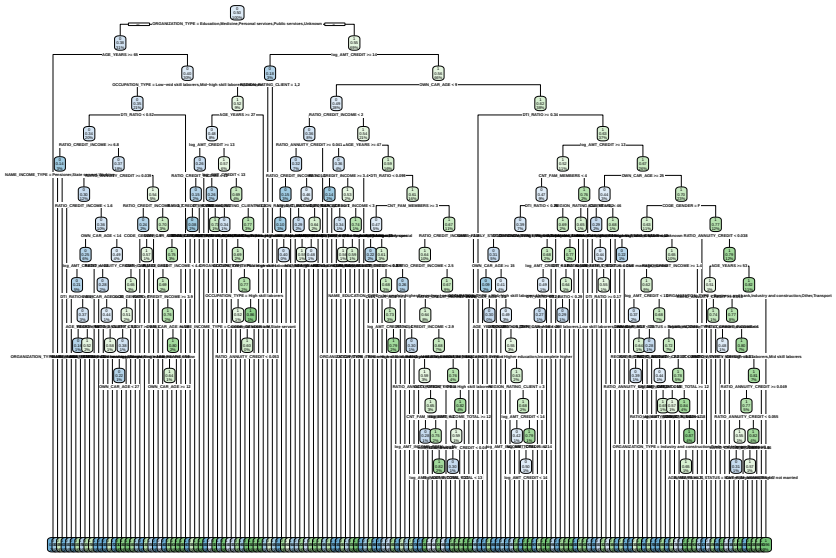
Para obtener una perspectiva más precisa sobre la selección del tamaño del árbol se imprimen los valores de CP.

```
printcp(model)
```

```
##
## Classification tree:
## rpart(formula = TARGET ~ ., data = train_balanceado, cp = 0)
##
## Variables actually used in tree construction:
##  [1] AGE_YEARS            CNT_FAM_MEMBERS      CODE_GENDER
##  [4] DTI_RATIO            log_AMT_CREDIT       log_AMT_INCOME_TOTAL
##  [7] NAME_EDUCATION_TYPE  NAME_FAMILY_STATUS   NAME_INCOME_TYPE
## [10] OCCUPATION_TYPE      ORGANIZATION_TYPE    OWN_CAR_AGE
## [13] RATIO_ANNUITY_CREDIT RATIO_CREDIT_INCOME  REGION_RATING_CLIENT
##
## Root node error: 1984/4001 = 0.49588
##
## n= 4001
##
##            CP nsplit rel error  xerror     xstd
## 1  0.13256048      0   1.00000 1.00000 0.015940
## 2  0.03326613      1   0.86744 0.88206 0.015815
```

```
## 3  0.02192540      2   0.83417 0.86542 0.015780
## 4  0.01209677      5   0.76462 0.81855 0.015656
## 5  0.01008065      6   0.75252 0.77268 0.015500
## 6  0.00982863      7   0.74244 0.75756 0.015440
## 7  0.00957661      9   0.72278 0.75554 0.015432
## 8  0.00831653     13   0.68246 0.72379 0.015293
## 9  0.00756048     15   0.66583 0.71472 0.015250
## 10 0.00705645     24   0.58518 0.70464 0.015201
## 11 0.00604839     26   0.57107 0.66028 0.014961
## 12 0.00554435     27   0.56502 0.63861 0.014831
## 13 0.00529234     29   0.55393 0.63004 0.014777
## 14 0.00520833     31   0.54335 0.62550 0.014747
## 15 0.00504032     34   0.52772 0.61794 0.014698
## 16 0.00453629     37   0.51260 0.61089 0.014650
## 17 0.00403226     38   0.50806 0.59325 0.014528
## 18 0.00386425     40   0.50000 0.58468 0.014466
## 19 0.00378024     50   0.45514 0.55242 0.014218
## 20 0.00352823     54   0.44002 0.54839 0.014186
## 21 0.00327621     61   0.41532 0.53075 0.014040
## 22 0.00302419     72   0.37550 0.51058 0.013863
## 23 0.00289819     82   0.34274 0.50050 0.013772
## 24 0.00277218     87   0.32762 0.50050 0.013772
## 25 0.00264617     89   0.32208 0.47429 0.013522
## 26 0.00252016     93   0.31149 0.47329 0.013512
## 27 0.00226815    104   0.28377 0.46573 0.013436
## 28 0.00218414    112   0.26462 0.44002 0.013168
## 29 0.00201613    115   0.25806 0.43901 0.013157
## 30 0.00184812    120   0.24798 0.43095 0.013069
## 31 0.00176411    123   0.24244 0.42893 0.013047
## 32 0.00168011    125   0.23891 0.42893 0.013047
## 33 0.00163810    128   0.23387 0.41986 0.012945
## 34 0.00151210    135   0.22026 0.41986 0.012945
## 35 0.00126008    149   0.19909 0.41532 0.012893
## 36 0.00117608    153   0.19405 0.41028 0.012834
## 37 0.00100806    156   0.19052 0.40373 0.012757
## 38 0.00075605    158   0.18851 0.40171 0.012734
## 39 0.00050403    162   0.18548 0.39617 0.012667
## 40 0.00033602    169   0.18196 0.39415 0.012643
## 41 0.00025202    172   0.18095 0.39365 0.012637
## 42 0.00000000    176   0.17994 0.39365 0.012637
```

Para obtener el modelo final, seleccionamos el valor óptimo de complejidad siguiendo el criterio de un error estándar de Breiman et al. (1984) y podamos el árbol.



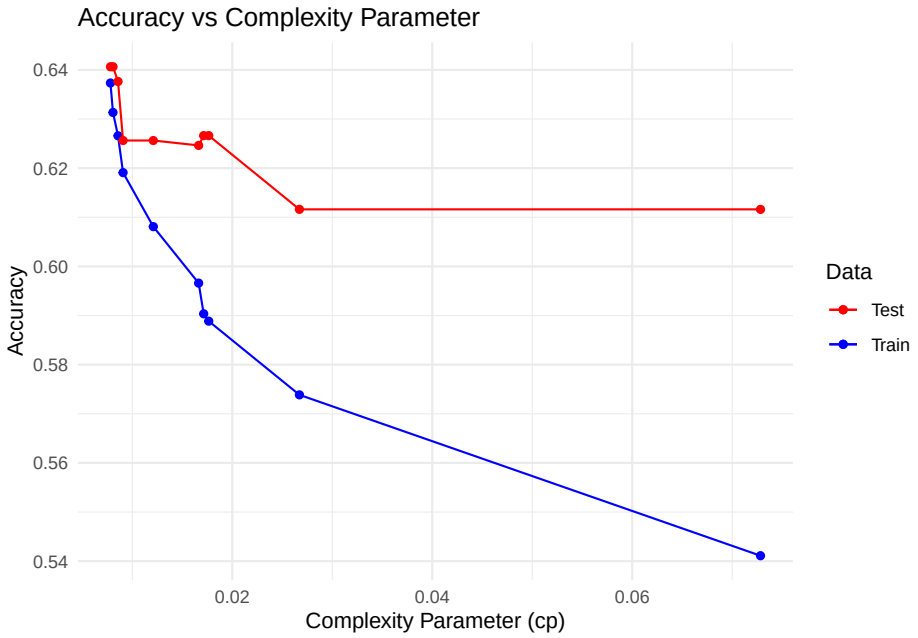
\begin{figure} \caption{Árbol de clasificación de `train_balanceado$TARGET` obtenido después de la poda} \end{figure}

Aunque esta estrategia ayuda a evitar el sobreajuste y a seleccionar un modelo que se encuentra en un punto óptimo entre sesgo y varianza, no nos es óptima ya que aunque el cp sea óptimo, el árbol resultante sigue siendo demasiado grande. Asi pues, consideramos ajustar la estrategia de selección de complejidad a partir de la función train como veremos a continuación.

Modelo de clasificación con interfaz de `caret` A través de la libreria `CARET` podemos ajustar un árbol CART seleccionando method = "rpart". Por defecto emplea bootstrap de las observaciones para seleccionar el valor óptimo del hiperparámetro cp (considerando únicamente tres posibles valores). Si queremos emplear validación cruzada, empleamos la función auxiliar trainControl() y para considerar un mayor rango de posibles valores, el argumento tuneLength.

Entonces, para encontrar este valor óptimo del parámetro de complejidad, se entrena el modelo con los datos balanceados de Train y se realiza un proceso de crosvalidación con 10 folds. Entonces calculamos el accuracy para cada 10 valores posibles del complexity parameter tanto para los datos train como test.

Figura 123: Evolución de la precisión (obtenida mediante validación cruzada) dependiendo del parámetro de complejidad



En el gráfico se observa como el primer valor del complexity parameter es el que reporta un mayor accuracy tanto para el conjunto de datos de entrenamiento como el de validación, por lo que 'r model2$bestTune' es el valor óptimo. Como el valor del accuracy es muy parecido para ambos conjuntos de datos, podemos concluir que no se produce un sobreajuste del modelo.

Validación del modelo

Una vez ejecutado el modelo CART se muestra en una tabla la matriz de confusión y se calcula la precisión con la que el algoritmo ha predicho la variable Target tanto en la población del Train como en la del Test, para observar si ha habido un sobreajuste o no.

Cuadro 46: Matriz de confusión del conjunto de validación

	Realidad	
	No moroso	Potencial moroso
Predicción		
No moroso	356	211
Potencial moroso	148	284

Cuadro 47: Medidas de Validación para el modelo CART

	Train	Test
Accuracy	0.6470882	0.6406406
Sensitivity	0.5680444	0.5737374
Specificity	0.7248389	0.7063492
Recall	0.5680444	0.5737374
F1	0.6148391	0.6127292
Precision	0.6700357	0.6574074

La anterior salida nos muestra la matriz de confusión junto con diversos estadísticos que tratan de explicar como de bien o mal ha predicho el algoritmo de CART. Así pues, como se observa que las medidas de validación son aproximadamente las mismas tanto para Train como para Test, afirmamos que no hay un sobreajuste de los datos.

En este caso, la precisión ha sido del 0.6406 %, lo que indica que el algoritmo ha predicho correctamente el 64.0640641 % de los individuos de Test.

La "Sensitivity" mide la proporción de individuos de TARGET=0 que han sido clasificacdos correctamente, que en este caso ha sido de 57.3737374.

Por otro lado, la "Specificity" mide la proporción de individuos de TARGET=1 que han sido clasificados correctamente, que ha dado 70.6349206

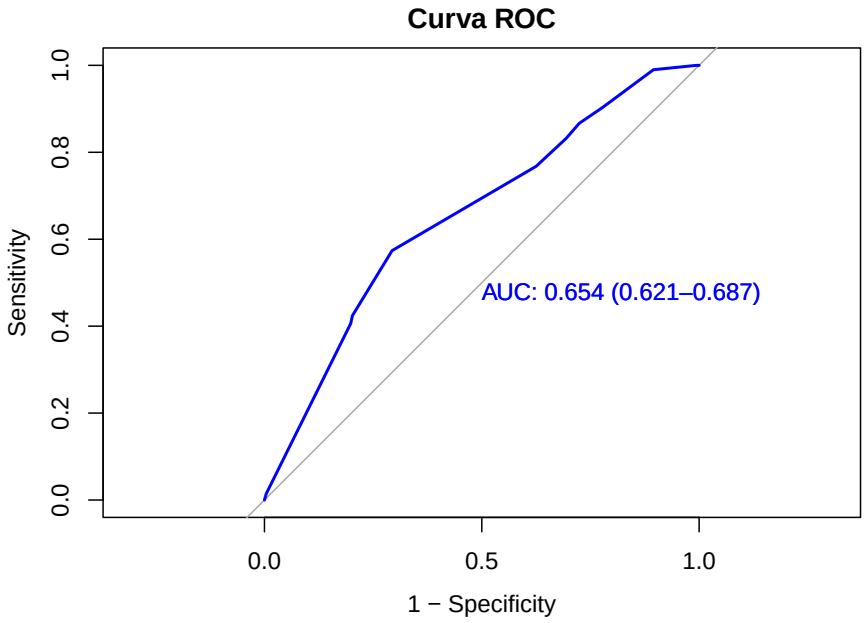
Si observamos otras métricas disponibles, apreciaremos una tasa de valores positivos predichos de 65.7407 % y una tasa de valores negativos predichos de 62.7866 %. Este hecho implica que al predecir una clase, la probabilidad de que ésta sea clasificada correctamente es de entorno al 64 %. Por último, podemos apreciar que el valor del F-score es de 0.6718922, métrica perjudicada por el bajo valor de la sensibilidad.

Curva ROC Para un análisis más profundo sobre la calidad de predicción del modelo, se representa la curva ROC y se interpreta su área bajo la curva (AUC).

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Figura 124: Curva ROC



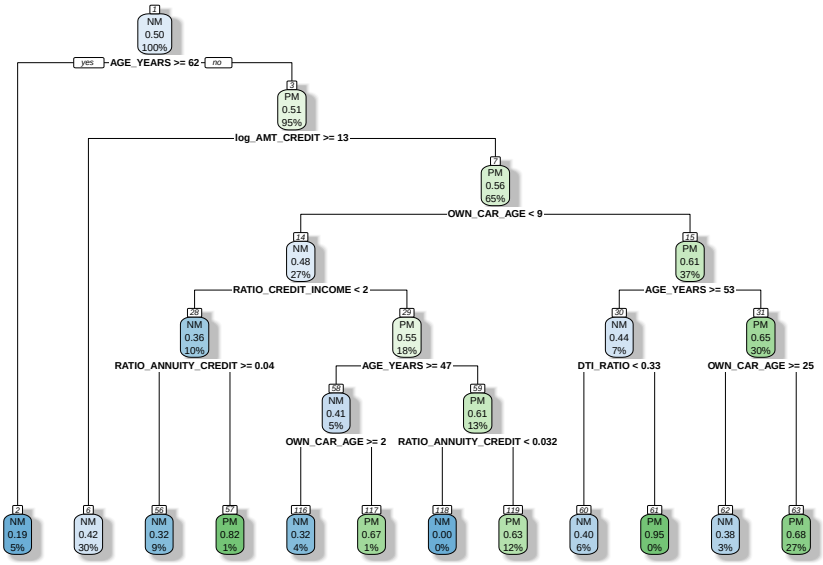
El AUC (Área Bajo la Curva) de 0.654 en la curva ROC sugiere que el modelo tiene un rendimiento moderado en la clasificación binaria entre morosos y no morosos. En otras palabras, el modelo es mejor que una clasificación aleatoria, pero hay margen para mejorar.

Árbol de decisión

A continuación, se presenta el árbol de decisión final con el parámetro de complejidad óptimo.

¿QUEREMOS DECIR ALGO DE LOS PORCENTAJES?

Figura 125: Árbol de clasificación de la variable TARGET, obtenido con la complejidad 'óptima'



El árbol de decisión generado se inicia evaluando la edad del solicitante. Si la edad es mayor o igual a 61.5 años, la clasificación resultante es "No Moroso" (NM). Este primer nivel de decisión sugiere que, en general, los solicitantes de mayor edad tienden a ser clasificados como no morosos directamente.

Por otro lado, si la edad es menor a 61.5 años, el árbol se ramifica, y la siguiente variable considerada es el importe de crédito del préstamo (log_AMT_CREDIT). Si el monto del crédito es mayor o igual a 13.42907, la clasificación es nuevamente "No Moroso" (NM). Esto indica que para los solicitantes más jóvenes con montos de crédito más altos, el modelo tiende a prever que no serán morosos, con una tasa de acierto del 30 %.

En caso de que el monto del crédito sea menor a 13.42907, el árbol examina la edad del automóvil propio (OWN_CAR_AGE), dividiéndose en dos caminos. Si la edad del automóvil es menor a 8.5 años, el modelo clasifica como "No Moroso" (NM). Sin embargo, si la edad del automóvil es mayor o igual a 8.5 años, se procede a una serie de condiciones adicionales basadas en otras variables como RATIO_CREDIT_INCOME, RATIO_ANNUITY_CREDIT, DTI_RATIO, y otras.

Este orden de variables en el árbol está determinado por la importancia relativa de cada variable en la tarea de clasificación. Las variables que ofrecen una mayor separación entre las clases son utilizadas en los niveles iniciales del árbol.

Así pues, como podemos observar, en orden de importancia la variable "Edad" es la más relevante a la hora de clasificar a los clientes en morosos y no morosos, donde en el 60 % de las veces que se cree un árbol de decisión, saldrá esta variable como la principal. SEGUIR INTERPRETACIÓN, NO PUEDO VER EL GRÁFICO.

Conclusiones

En resumen, el árbol de decisión proporciona un marco claro para entender cómo el modelo clasifica a los solicitantes en función de sus características, permitiendo una interpretación detallada de las reglas de

decisión utilizadas en la evaluación de la morosidad, siendo la variable EDAD la más importante en la tarea de clasificación de clientes morosos.

Como la sensibilidad obtenida ha sido más baja que la especificidad, concluimos que el modelo tiene más dificultades para identificar los casos positivos reales (morosos) en comparación con su habilidad para identificar correctamente los casos negativos reales (no morosos). Este resultado no nos es beneficioso en la clasificación, ya que en este contexto quizás sea mejor detectar adecuadamente casos posotivos (morosos), para así reducir el número de clientes morosos.

En resumen, observando los resultados obtenidos, se puede afirmar que los resultados obtenidos son un tanto pobres, siendo necesario utilizar algún otro tipo de modelo de predicción que aporte valores más óptimos.