Estructura de Directorios

Dentro de ~/cloud/cloud1

- `datasetsOri` un solo archivo con 36 meses
- `datasets`
  - `dias` fechas pasadas a dias
  - `ext` variables nuevas dentro del mismo mes
  - `hist` variables históricas
  - `exthist` variables históricas de las extendidas

- `R` código fuente
  - `kinder`
  - `elementary`
  - `rpart`
  - `ranger`
  - `xgboost`
  - `lightgbm`

  - `include`
  - `FeatureEngineering`

- `work` quedan los resultados de los programas

/elementary/Predicados_01.r

Plataforma de ejecución: PC local

Se tiene el primer acercamiento al dataset `201902.txt`, y se muestra el concepto de *patrón* (pattern).

ganancia = 19500 * 'BAJA+2' - 500 * ('BAJA+1' + 'CONTINUA')

Si se le hiciera la acción comercial a TODOS los clientes			
BAJA+1	**BAJA+2**	**CONTINUA**	**GANANCIA**
988	1,085	185,788	-72,230,500

Universo		
	cantidad	**porc**
POS	1,085	0.58
NEG	186,776	99.42
Total	**187,861**	**100.00**

Partición Perfecta (inalcanzable !)					
	BAJA+1	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
condicion_magica	0	1,085	0	173.14	21,157,500
NOT(condicion_magica)	988	0	185,788	0.00	-93,388,000
Total	**988**	**1,085**	**185,788**	**1.00**	-72,230,500

lift(pred) = (pred_pos/pred_cant) / (universo_pos / universo_cant)

Ejemplo de partición del Universo					
	BAJA+1	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
cliente_edad <= 33	167	212	29,146	1.24	-10,522,500
cliente_edad > 33	821	875	156,642	0.96	-61,708,000
Total	**988**	**1,085**	**185,788**	**1.00**	**-72,230,500**

Otro ejemplo más arbritario					
	BAJA+1	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
mcuentas_saldo <= -120000	96	87	110	51.41	1,593,500
mcuentas_saldo > -120000	892	998	185,678	0.92	-73,824,000
Total	**988**	**1,085**	**185,788**	**1.00**	**-72,230,500**

Nada mal, ya le hicimos ganar a la empresa 1.593.500 con el simplisimo predicado `(mcuentas_saldo <= -120000)`, el gran desafio es como encontrar predicados de ese tipo ...

¿Qué sucede si evalúo el corte en forma binaria POS vs NEG ?

	POS	**NEG**
mcuentas_saldo <= -120000	87	206
mcuentas_saldo > -120000	998	186,570
Total	**1,085**	**186,776**

Los NAs deben ser tenidos en cuenta !					
	BAJA+1	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
Visa_mconsumototal <= 20000	490	576	130,522	0.75	-54,274,000
Visa_mconsumototal > 20000	24	40	33,226	0.21	-15,845,000
is.na(Visa_mconsumototal)	474	469	22,040	3.53	-2,111,500
Total	**988**	**1,085**	**185,788**	**1.00**	**-72,230,500**

ttarjeta_visa	BAJA+1	BAJA+2	CONTINUA	lift	GANANCIA
0	510	446	6,501	10.35	5,191,500
1	472	628	176,097	0.61	-76,038,500
2	6	11	3,155	0.60	-1,366,000
3	0	0	35	0.00	-17,500

Visa_cuenta_estado	BAJA+1	BAJA+2	CONTINUA	lift	GANANCIA
10	236	639	179,260	0.61	-77,287,500
11	164	99	251	33.35	1,723,000
12	60	88	211	42.44	1,580,500
19	266	26	256	8.10	242,000
NA	262	233	5,802	6.41	1,511,500

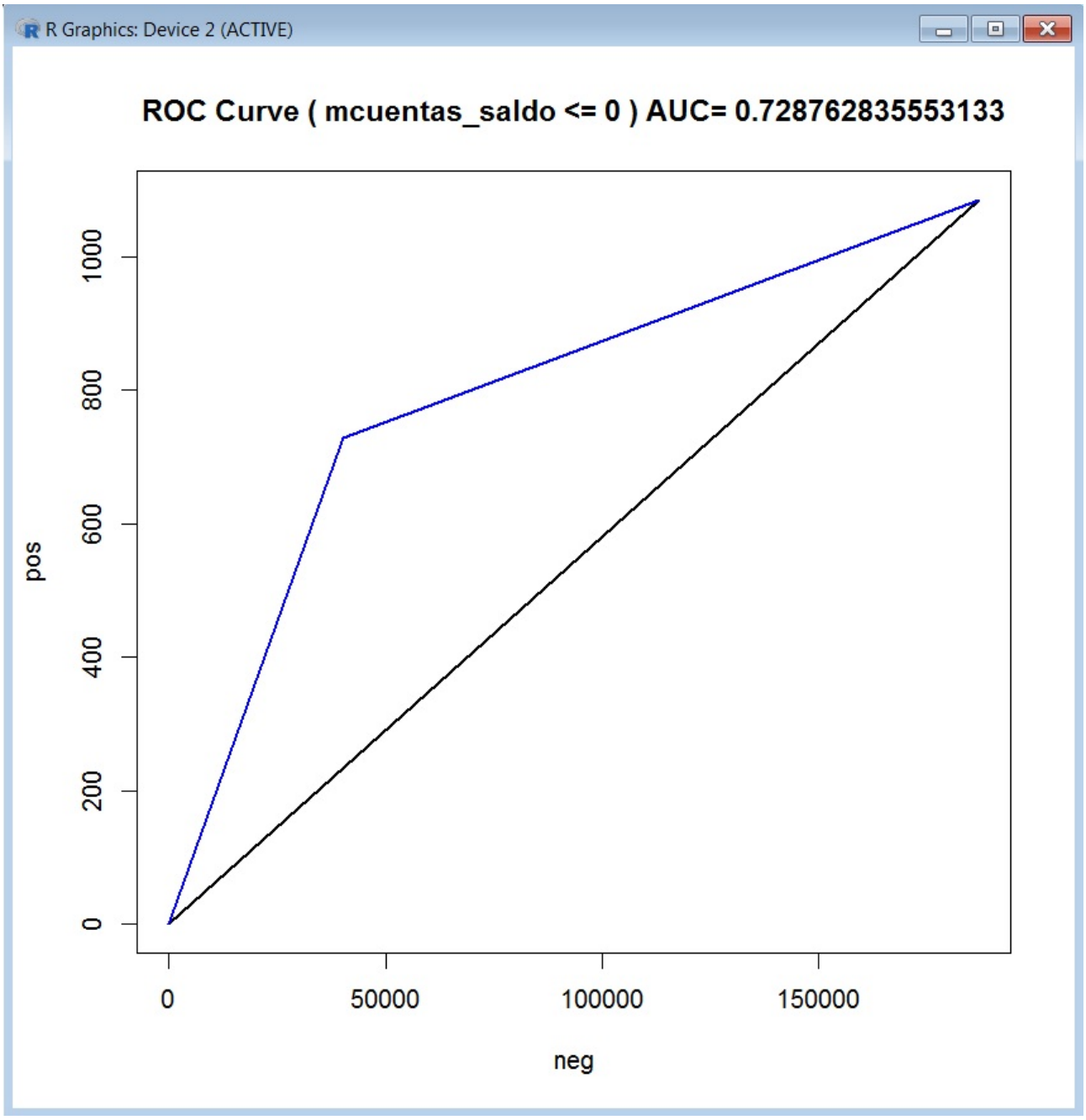
Master_cuenta_estado	BAJA+1	BAJA+2	CONTINUA	lift	GANANCIA
10	199	532	163,273	0.56	-71,362,000
11	117	58	176	28.61	984,500
12	44	74	157	46.59	1,342,500
19	225	23	234	8.26	219,000
NA	403	398	21,948	3.03	-3,414,500

/elementary/ROC_01.r

Plataforma de ejecución: PC local

En clase se verá en profundidad la curva ROC.
Se dibuja la Curva ROC de un predicado del tipo `( variable <=  valor )`
Se define una función en R por primera vez, se muestra la sintaxis.

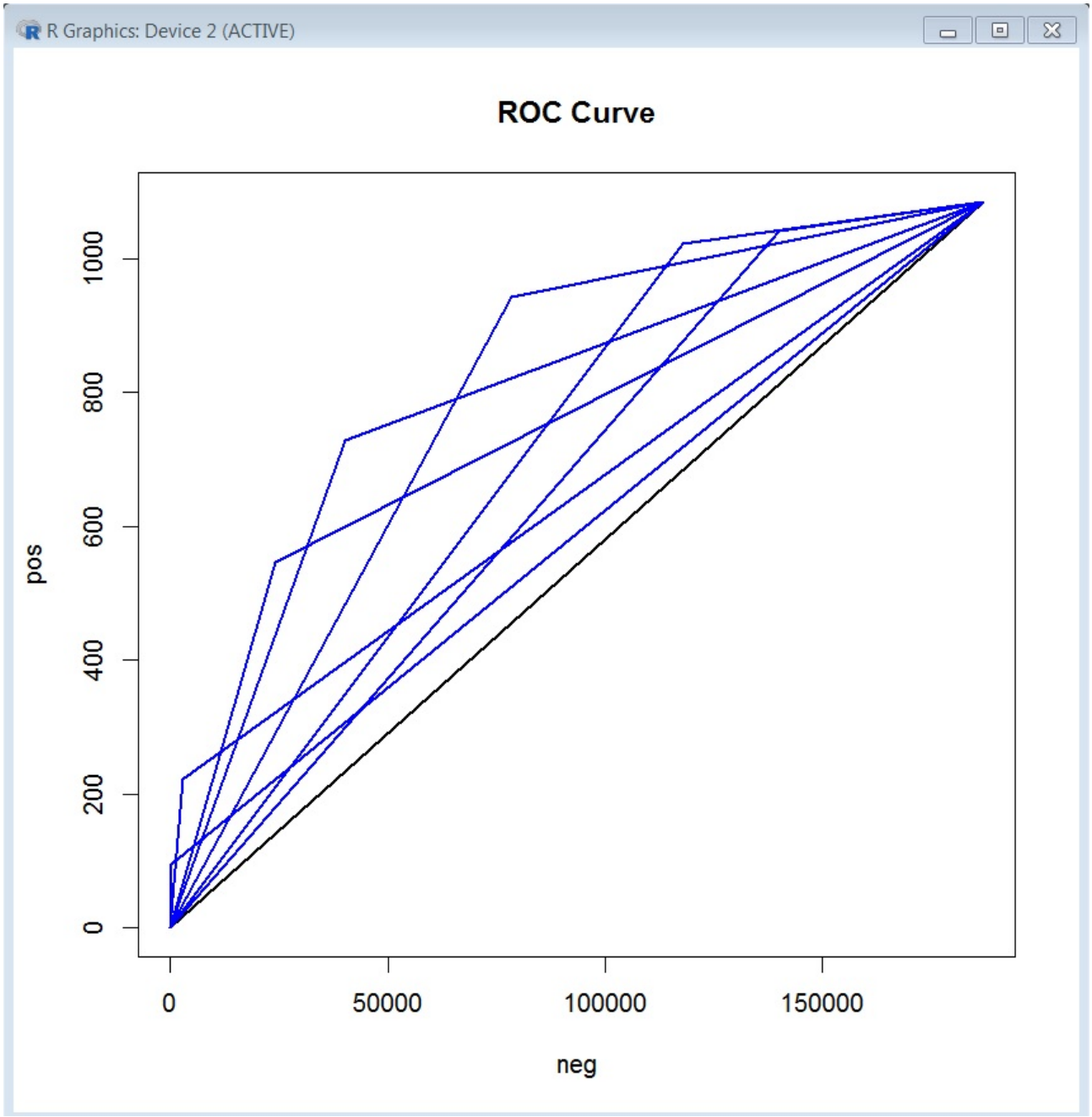
Se muestran distintas curvas ROC de una variable bajo distintos cortes de la misma.

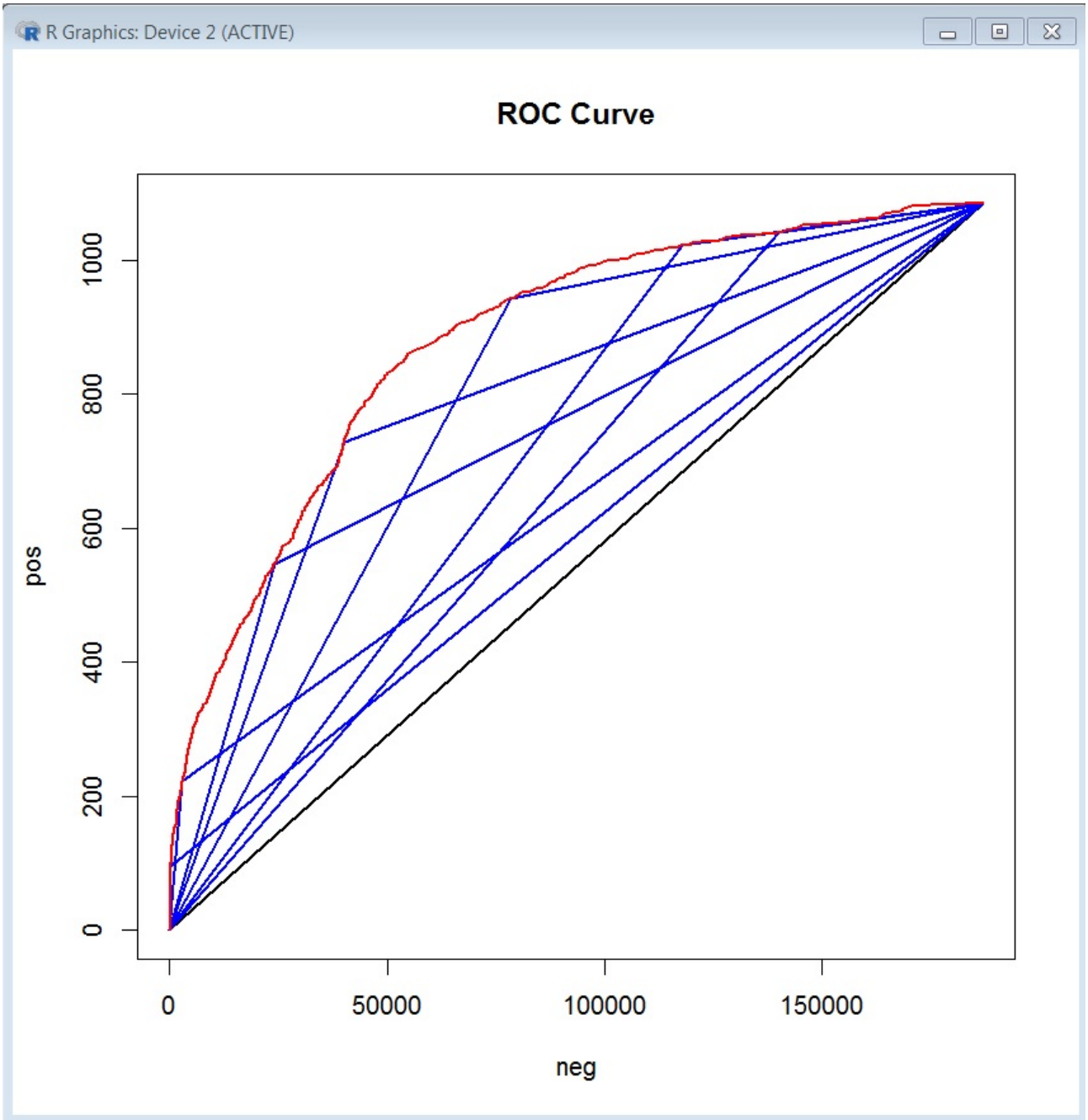


/elementary/ROC_02.r

Plataforma de ejecución: PC local

Se dibuja la Curva ROC de una variable, se introduce la elección del punto de corte óptimo de una variable.
En clase se explica en gran detalle como se ordena el dataset por una variable, y se van acumulando los positivos y negativos, el modelo que es devolver una lista ordenada de los registros, consiste simplemente en ordenar por esa variable, quizas en forma ascendente o descentente, y teniendo cuidado de donde ubicar los nulos, si al comienzo o al final del ordenamiento.





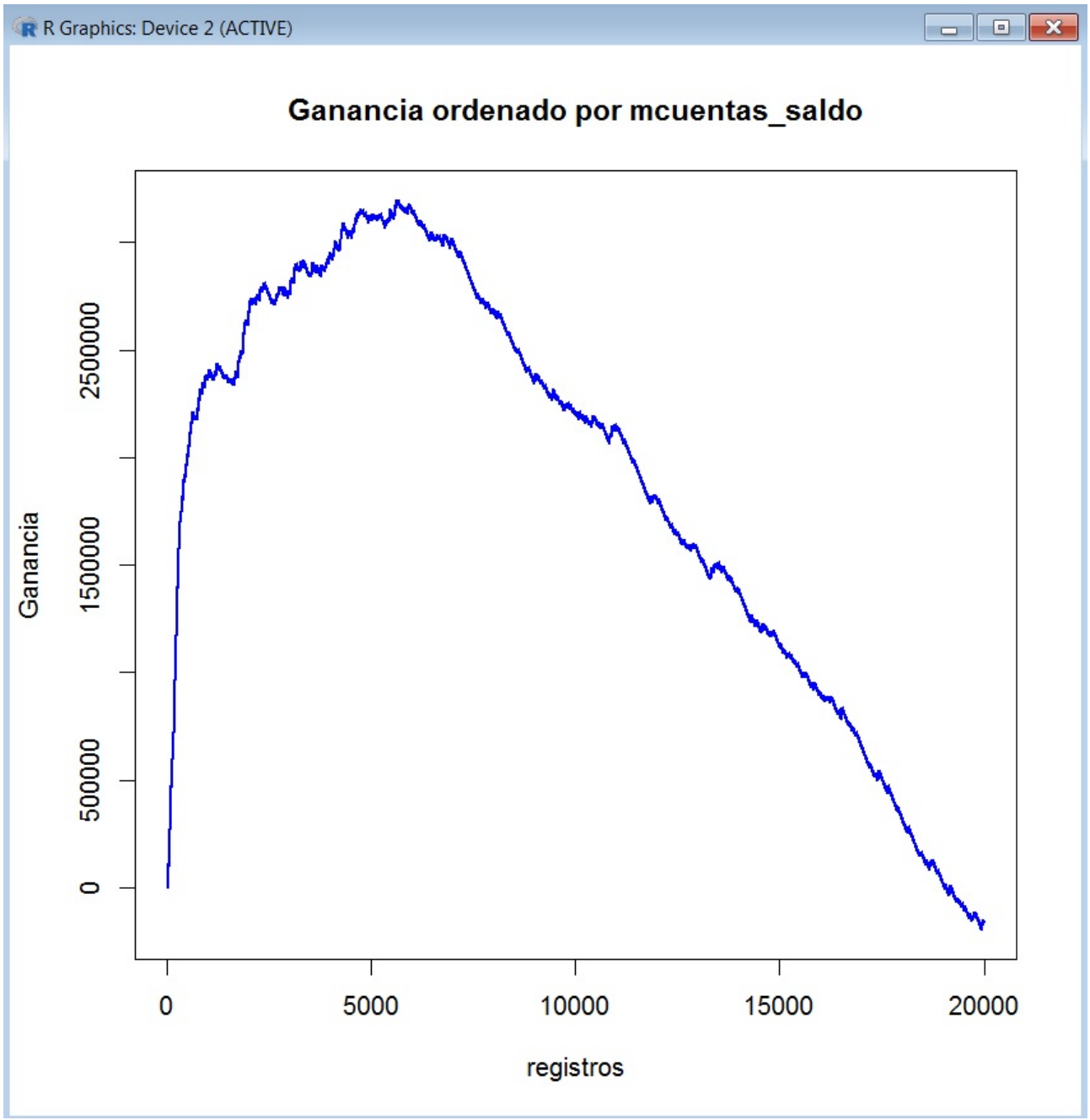
Recordando la tabla de contingencia del inicio

Otro ejemplo más arbritario					
	BAJA+1	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
mcuentas_saldo <= -120000	96	87	110	51.41	1,593,500
mcuentas_saldo > -120000	892	998	185,678	0.92	-73,824,000
Total	**988**	**1,085**	**185,788**	**1.00**	**-72,230,500**

y ahora la nueva tabla

Otro ejemplo más arbritario					
	BAJA+1	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
mcuentas_saldo <= 1275.59	781	858	53,688		-10,503,500
mcuentas_saldo > 1275.59	207	227	132,100		-61,727,000
Total	**988**	**1,085**	**185,788**	**1.00**	**-72,230,500**

Entonces de que sirve cortar maximizando el AUC si ahora a ambas partes tengo ganancia negativa ?



/elementary/ROC_03.r

Plataforma de ejecución: PC local

La idea es repetir lo anterior para toda las variables.
Se obtiene esta tabla, que se la muestra ordenada primero por AUC_max decreciente, y la segunda tabla por gan_max decreciente .

columna	AUC_max	gan_max
clase01	1.0000	21,157,500
clase_ternaria	0.9974	20,663,500
mcaja_ahorro_Paquete	0.7788	4,162,500
mtarjeta_visa_consumo	0.7710	606,500
ctarjeta_visa_transacciones	0.7681	606,500
tmovimientos_ultimos90dias	0.7672	3,987,500
Visa_mpagospesos	0.7638	59,000
mpasivos_margen	0.7512	1,470,000
mcuentas_saldo	0.7496	3,198,500
Visa_tconsumos	0.7390	59,000

columna	AUC_max	gan_max
clase01	1.0000	21,157,500
clase_ternaria	0.9974	20,663,500
ttarjeta_visa	0.6872	5,219,500
Visa_cuenta_estado	0.6865	5,067,000
mdescubierto_preacordado	0.6605	4,418,500
Visa_mfinanciacion_limite	0.6740	4,381,000
mcaja_ahorro_Paquete	0.7788	4,162,500
tmovimientos_ultimos90dias	0.7672	3,987,500
mcuenta_corriente_Paquete	0.6916	3,919,000
Visa_marca_atraso	0.6605	3,819,000

En que valor tengo que cortar a la variable `ttarjeta_visa` para obtener una ganancia de 5,219,500 ?

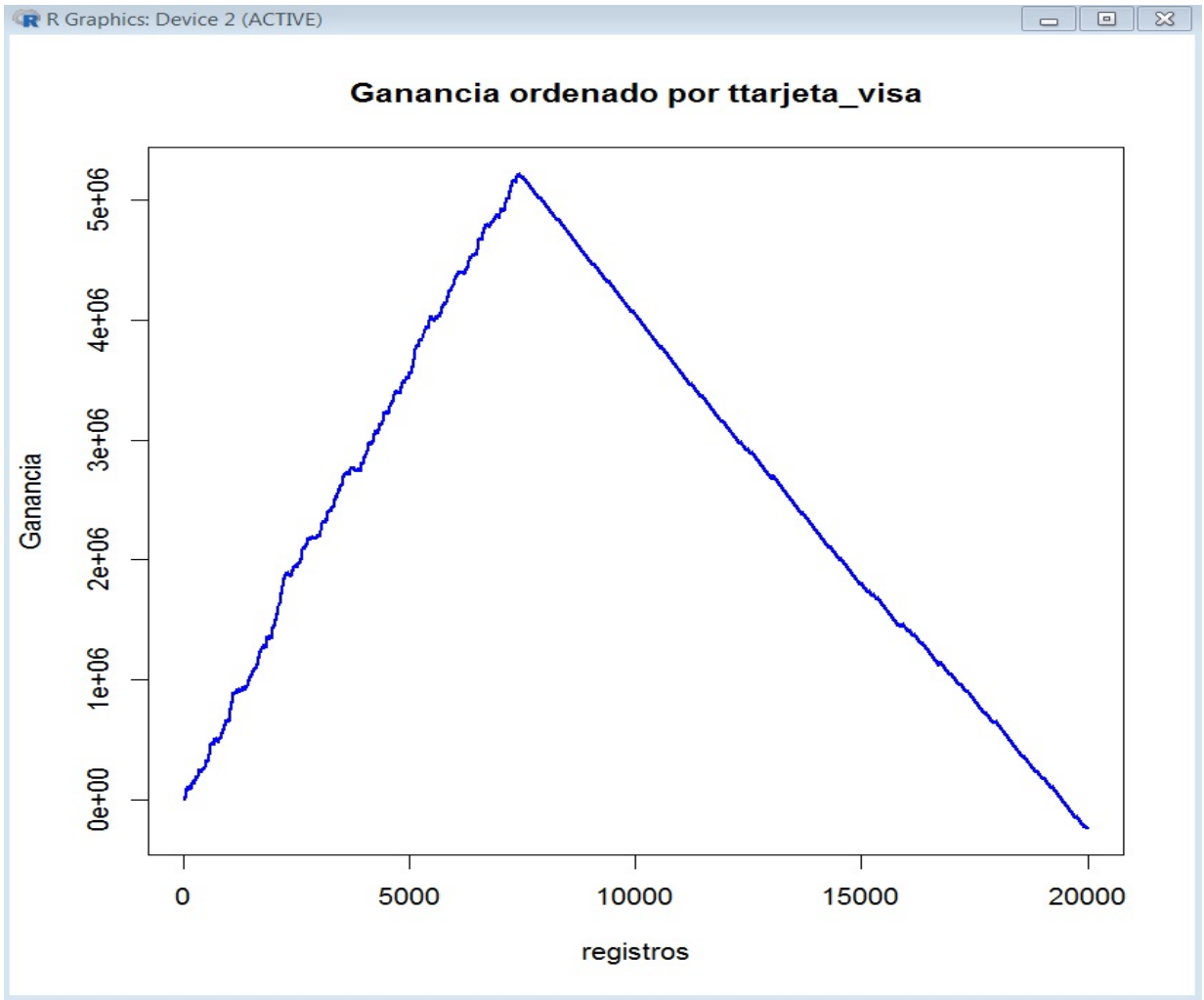
```
> columna_graficar_ganancia_n( dataset, "ttarjeta_visa", 20000  )

$`variable`
[1] "ttarjeta_visa"

$valor
[1] 0

$gan_max
[1] 5219500

$regis
[1] 7401
```



La mejor ganancia con un corte univariado					
ttarjeta_visa	**BAJA+1**	**BAJA+2**	**CONTINUA**	**lift**	**GANANCIA**
0	510	446	6,501	10.35	5,191,500
1	472	628	176,097	0.61	-76,038,500
2	6	11	3,155	0.60	-1,366,000
3	0	0	35	0.00	-17,500

Inicial

La idea es acercar a los alumnos a un arbol de probabilidad (decisión), mostrando código en lenguaje R de forma incremental. Se mostrará como cada hoja devuelve la probabilidad de ser positivo haciendo enfático caso omiso al label que se asigna basado en la clase mayoritaria, dado que eso parte de la suposición que las clases pesan lo mismo.
En el problema de la materia las clases pesan muy distinto al igual que en el 99% de los problemas que los alumnos se encontrarán en su vida profesional.
De hecho, por la definicion del problema, nos seran utiles todas las hojas con mas de un 2.5% de clase positivos.

Desde el inicio se muestra un fuerte interes por la optimización, de forma ir preparando el terreno para los procesamientos realmente pesados con modelos de Random Forest y XGBoost. Solo se utilizarán optimizaciones que no hagan perder claridad al código, basicamente basadas en el uso de librerias más rápidas y el uso adecuado de los parámetros de los algoritmos predictivos.

Se pide a los alumnos presten atención a como el código se va transformando, ya que se compejizará y profesionalizará rápidamente.

- Se deben instalar las librerias a utilizar
- Los alumnos deben utilizar sus propias cinco semillas aleatorias

Los evolucion dentro del este tramo del sendero que llamamos inicial se realizara de esta forma :

- Algoritmo : solo se utiliza la librería de arboles rpart
- Procesamiento : solo se corre en la PC local
- Estimación de la ganancia
  - Cálculo manual del dibujo del arbol sobre todo el dataset
  - Training / Testing y calculo automático
  - Multiple Training /Testing con calculo automático
  - Entrenar en un mes, testear con otro mes
- Optimización
  - Corrida con hiperparámetros default
  - Busqueda manual de hiperparámetros óptimos
- Código R
  - sin parametrización
  - parametrización por medio de constantes
  - variables listas, funciones
  - lapply

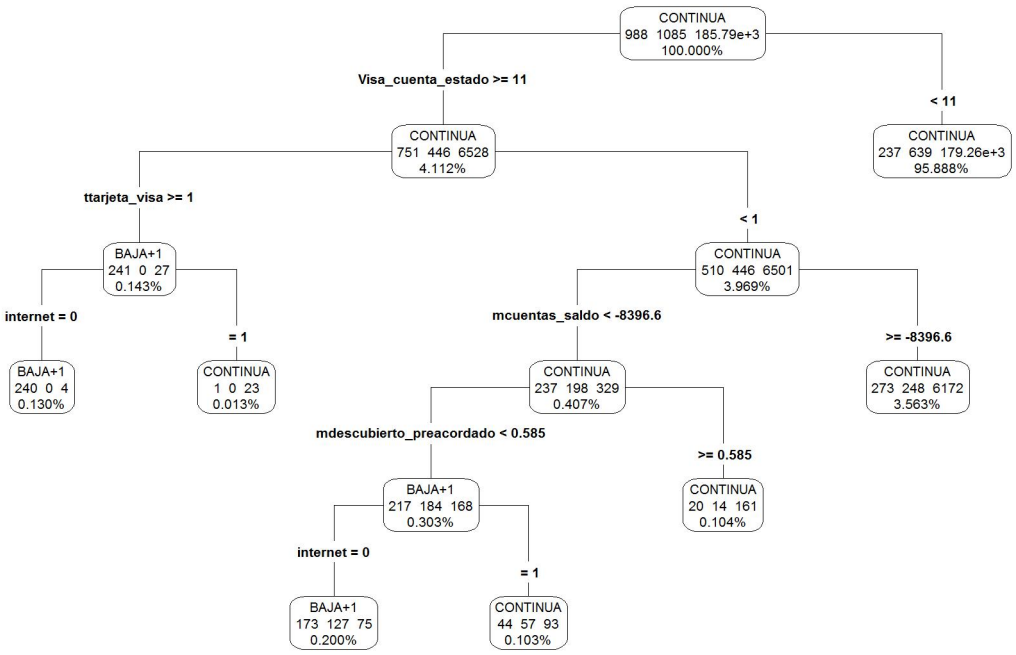
/elementary/MiPrimerArbol_01.r

Plataforma de ejecución: PC local

Es la primer corrida de un arbol de probabilidad (decisión) usando la librería `rpart`.
Se insta a los alumnos a entender la estructura del arbol y los valores que aparecen en el grafico.
Conceptualmente se muestra al arbol ademas es util para explorar los datos.

Se ordena el código fuente, con formato que se mantendrá a lo largo del curso.
Al inicio del código van las constantes, comienzan con la letra k, estas variables son globales a todo el código y solo se inicializan al comienzo y jamas se modifican.
Es importante para los no informáticos empezar a tener una disciplina de programación.

Para muchos alumnos esta forma de programar será la primera que verán, y les servirá como base para poder generar su propio estilo luego de algunos cientos de horas de programación.
Se solicita a los alumnos que ya tienen experiencia en programar , que por favor modifiquen los programas para adaptarlos a su estilo de programación.



Ganancia de un arbol						
Hoja	**BAJA+1**	**BAJA+2**	**CONTINUA**	**prob(POS)**	**GANANCIA**	**GAN POS**
1	240	0	4	0.0000	-122,000	
2	1	0	23	0.0000	-12,000	
3	173	127	75	0.3387	2,352,500	2,352,500
4	44	57	93	0.2938	1,043,000	1,043,000
5	20	14	161	0.0717	1,825,000	1,825,000
6	273	248	6,172	0.0370	1,613,500	1,613,500
7	237	639	179,260	0.0035	-77288000	
Total	**988**	**1,085**	**185,788**	**0.0058**	**-72,230,500**	**5,191,500**

¿Cómo puede ser que en la simple regla (`ttarjeta_visa = 0 `) se obtiene una ganancia de $ 5,191,500 exactamente igual a la de de este arbol de 7 hojas **?**

Notar lo siguiente :

- En ninguna de las hojas los BAJA+2 son mayoria
- Hay hojas con ganancia positiva en donde los BAJA+2 son la minoria
- Un árbol generado en 30 segundos ya genera una ganancia de **5,191,500**

Actividad:

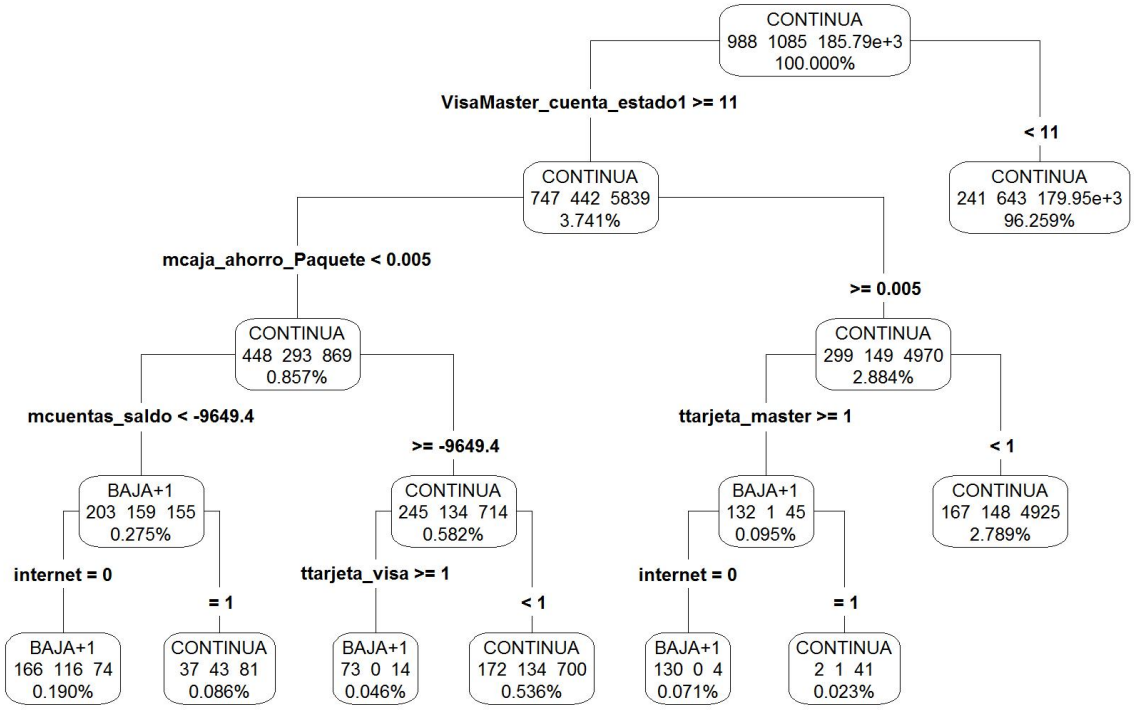
Volver a correr `MiPrimerArbol_01.r` cambiando en la llamada a rpart el parámetro cp=0.01 por cp=0.0 , teniendo el cuidado que la representación gráfica del arbol quede en un nuevo archivo llamado `arbol_01_bis.jpg` , comparar visualmente ambos árboles y calcular la ganancia de ese nuevo arbol .

- ¿ Cuántas hojas tiene el nuevo árbol ?
- ¿ Se puede decir que el nuevo árbol es una extensión del anterior ?
- ¿ Cuál es la ganancia del nuevo árbol ?

/elementary/MiPrimerArbol_02.r

Plataforma de ejecución: PC local

Se calcula manualmente la gnancia que resulta al generar el arbol sobre el dataset al que se la ha ***agregado*** la variable `max( Visa_cuenta_estado, Master_cuenta_estado )`



Hoja	BAJA+1	BAJA+2	CONTINUA	prob(POS)	GANANCIA	GAN POS
1	166	116	74		2,142,000	2,142,000
2	37	43	81		779,500	779,500
3	73	0	14		-43,500	
4	172	134	700		2177000	2,177,000
5	130	0	4		-67000	
6	2	1	41		-2000	
7	167	148	4,925		340000	340000
8	241	643	179,949		-77556500	
Total	**988**	**1,085**	**185,788**	**0.0058**	**-72,230,500**	**5,438,500**

La nueva variable `max( Visa_cuenta_estado, Master_cuenta_estado )` es la nueva raíz del árbol de decision, y la nueva ganancia **5,438,500** que es algo superior a la del árbol incial de **5,191,500**

/elementary/MiPrimerModelo_01.r

Plataforma de ejecución: PC local

Se corre el primer modelo con la division de training y testing, se genera el modelo en training y se aplica a testing.
Se dedica gran cantidad de tiempo a explicar a los alumnos como es el proceso de aplicación de un modelo, la matriz de probabilidades que devuelve la funcion `predict` en `rpart`, y como se calcula la ganancia del problema.
Se muestra a los alumnos en gran detalle la función ganancia, y como suman los aciertos y restan los no aciertos. Este concepto resulta dificil para quienes se enfrentan a él por primera vez, y aburrido a quienes ya tienen experiencia.

Se relaciona el arbol que se genera con las probabilidades que se devuelven, mostrando con ejemplos como son las probabilidades de las clases en cada hoja lo que se está devolviendo.

Se explica el concepto de *ganancia normalizada* .
Se muestra el cálculo del AUC de la curva ROC, previamente se vio la teoria de la Curva ROC.

Se solicita a los alumnos a comparar las ganancias obtenidas por cada uno, donde utilizan su propia semilla aleatoria. Se busca provocar el impacto emocional de la gran varianza que tienen las ganancias, y a huir de concepto “buena semilla vs mala semilla”. Esto da pie a la necesidad de hacer varias divisiones del dataset original en training y testing y luego promedias las ganancias.

/elementary/MiPrimerModelo_02.r

Plataforma de ejecución: PC local

Se promedian semillas utilizando Repeated Random Sub Sampling Validation (Monte Carlo Cross Validation), aun NO se introduce el k-fold Cross Validation.

Se utiliza un loop `for( s in  1:cantidad_semillas )` para dar un punto de referencia a las personas que vienen de carreras de Sistemas, pero ya en el próximo programa se cambiará por la instrucción funcional `lapply`

El alumno avanzado prueba los efectos de trabajar en lugar de 5 semillas con 10 y hasta con 20, observando como cambia el valor del promedio, y la varianza.

/inicial/MiPrimerModelo_03.r

Plataforma de ejecución: PC local

Se reemplaza el loop que itera por las semillas con un `lapply` .

Se diseña la función a la que se le pasan como parametro los hiperparámetros de rpart y se la llama con algunos valores arbritarios, observando una gran variacion de las ganancias obtenidas, no ya producto de la division entre training y testing, sino por el efecto de los distintos hiperparametros.
Esto trae a la mesa la necesidad de buscar los mejores hiperparámetros para `rpart` en este dataset.