# 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 |

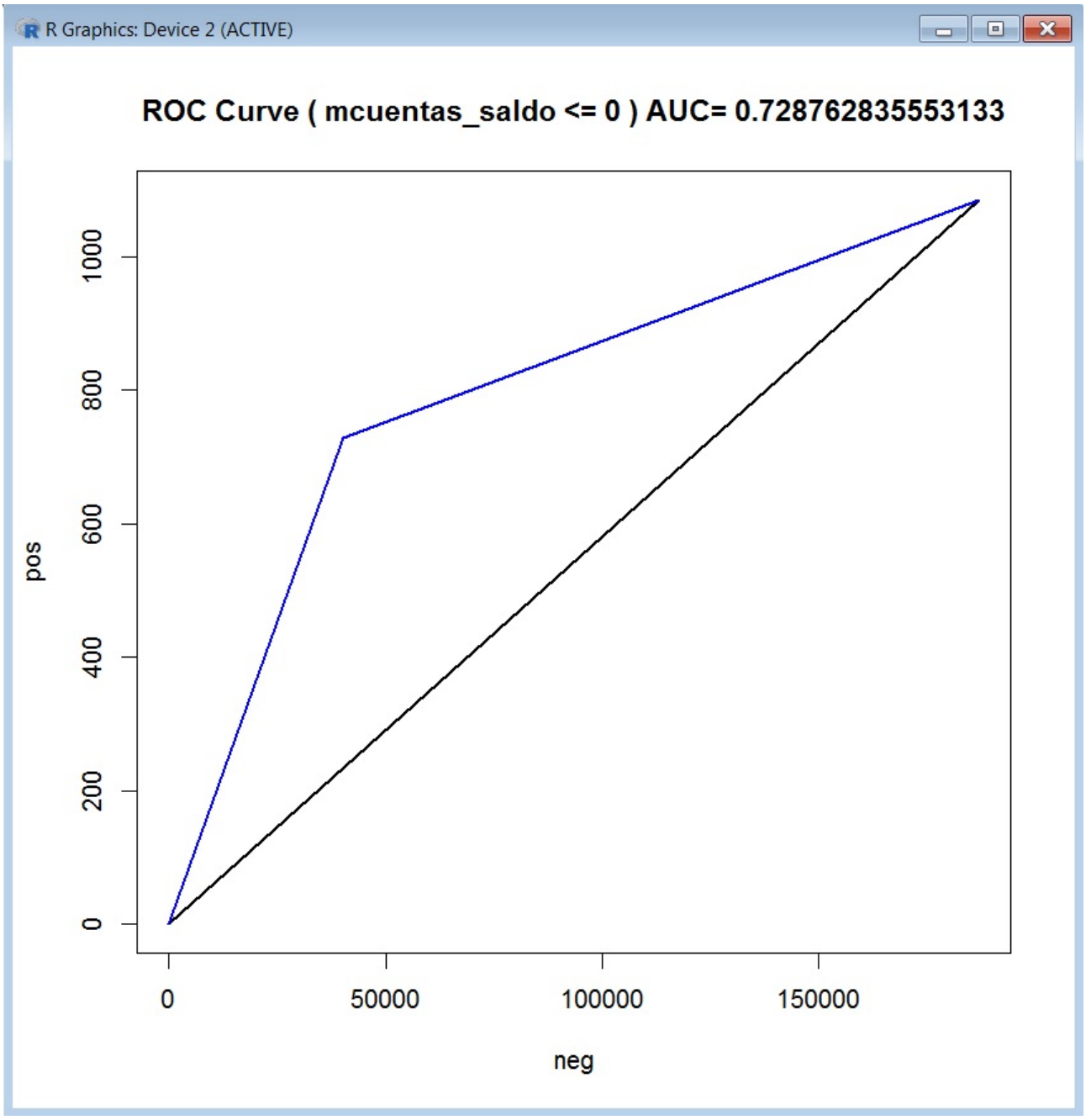
| Master_cuenta_estado | BAJA+1 | BAJA+2 | CONTINUA | lift | GANANCIA |
|---|---|---|---|---|---|
| 10 | 199 | 532 | 163,273 | 0.56 | -71,362,000 |
| 11 | 117 | 58 | 176 | 28.61 | 984,500 |
| 12 | 44 | 74 | 157 | 46.59 | 1,342,500 |
| 19 | 225 | 23 | 234 | 8.26 | 219,000 |
| NA | 403 | 398 | 21,948 |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

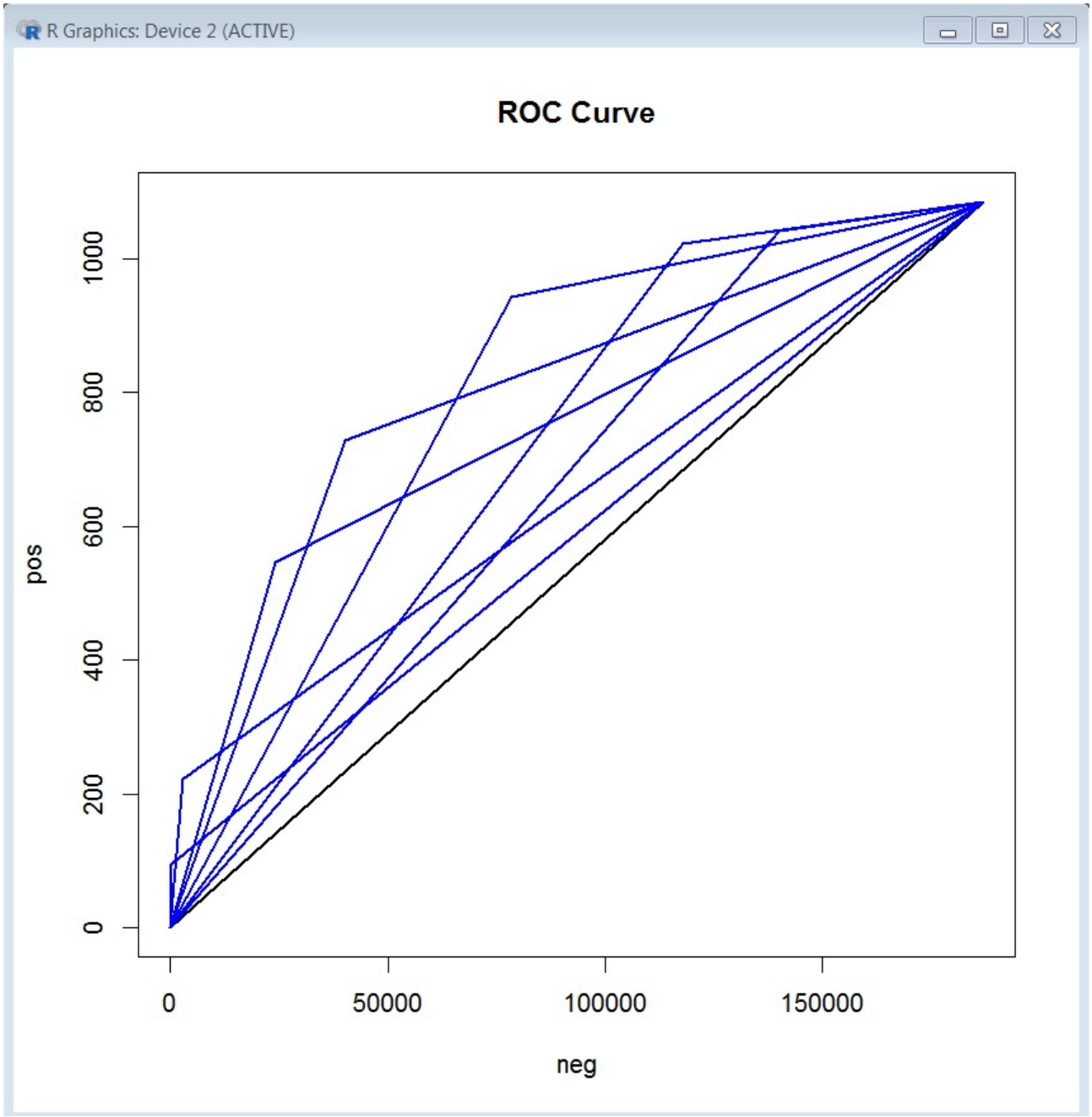
Se muestran distintas curvas ROC  de una variable bajo distintos cortes de la misma.

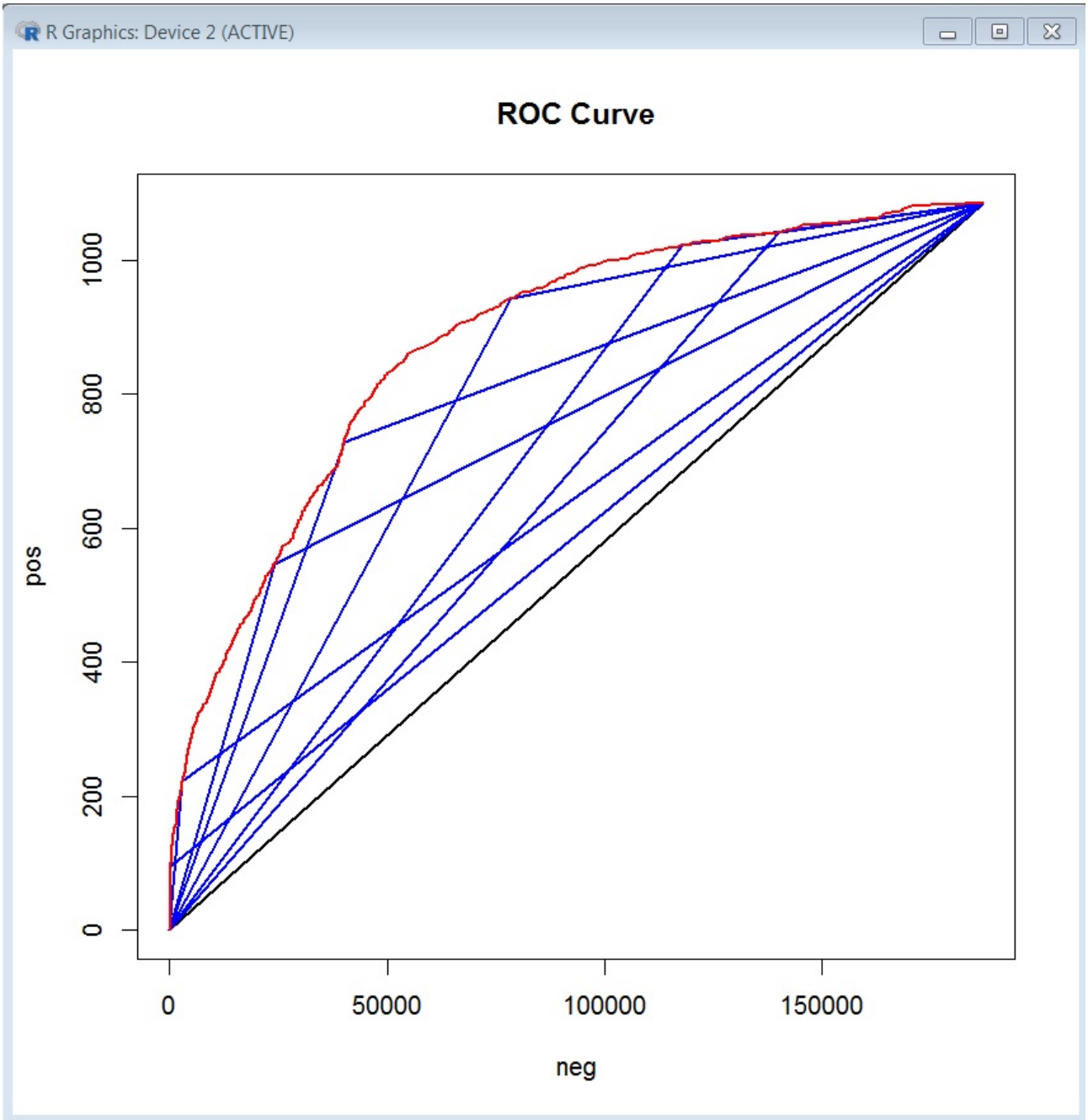


# /elementary/ROC_02.r

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.





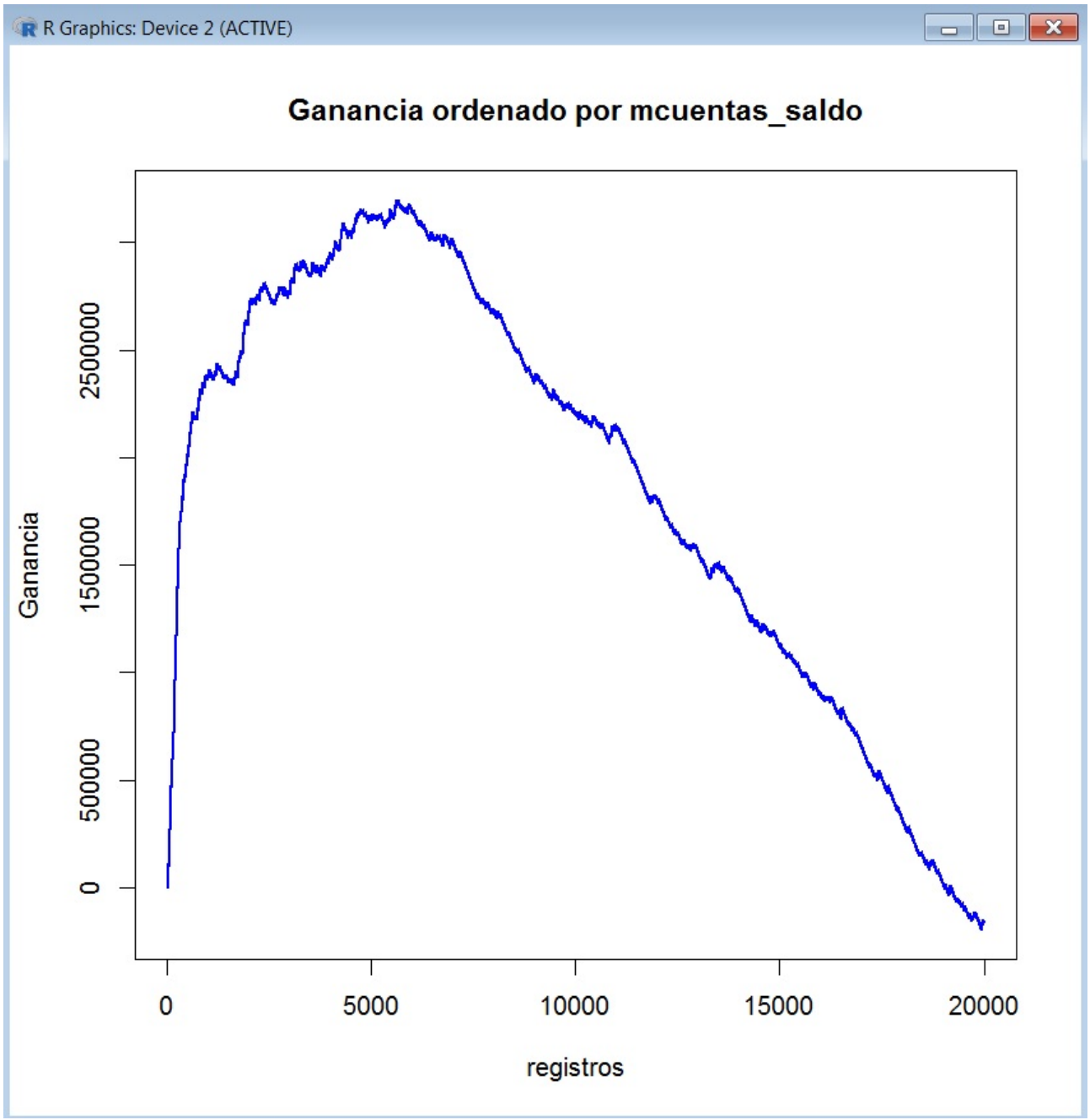

Recordando la tabla de contingencia del inicio

| 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 ?

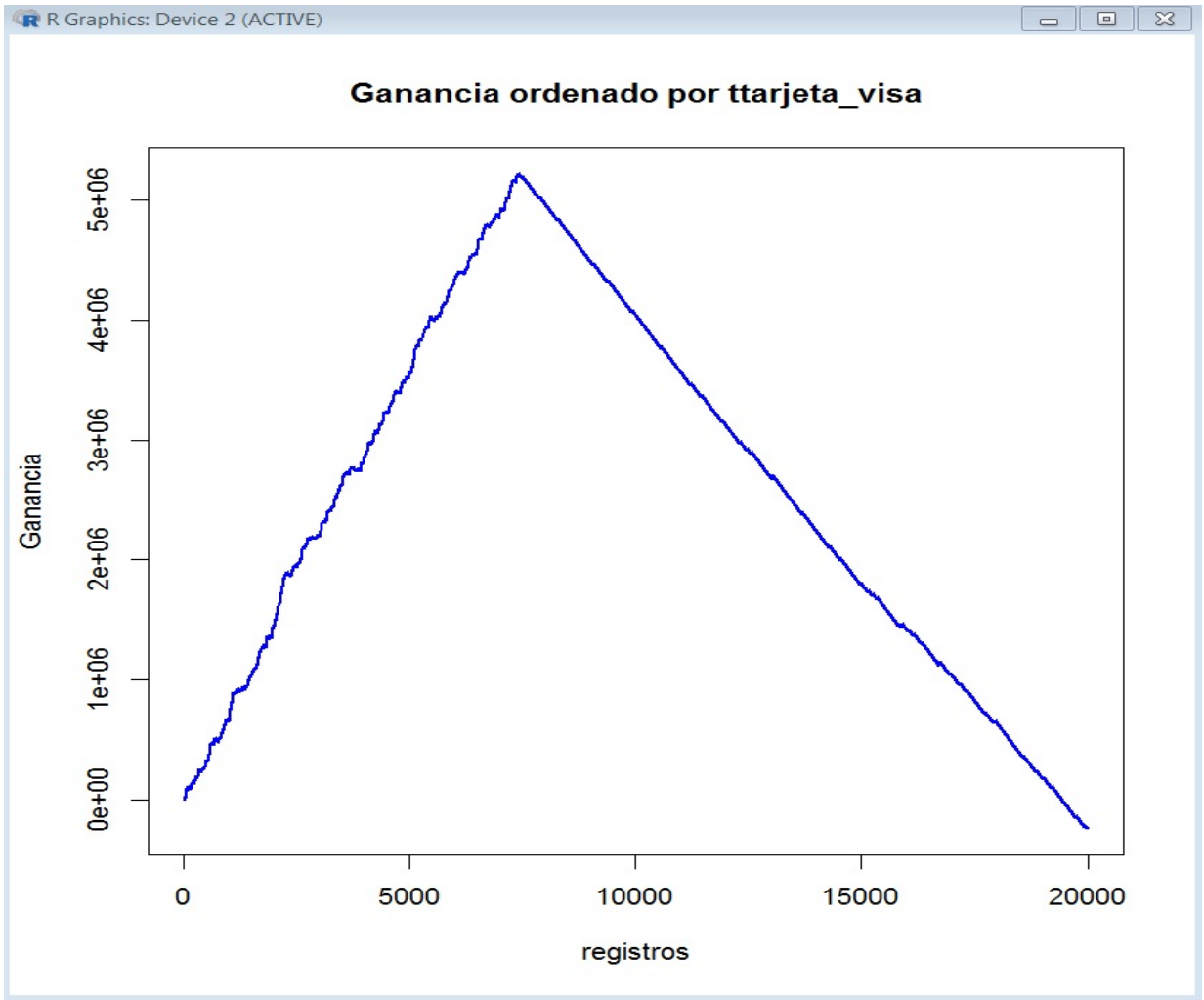
```
> columna_graficar_ganancia_n( dataset, "ttarjeta_visa", 20000  )

$`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

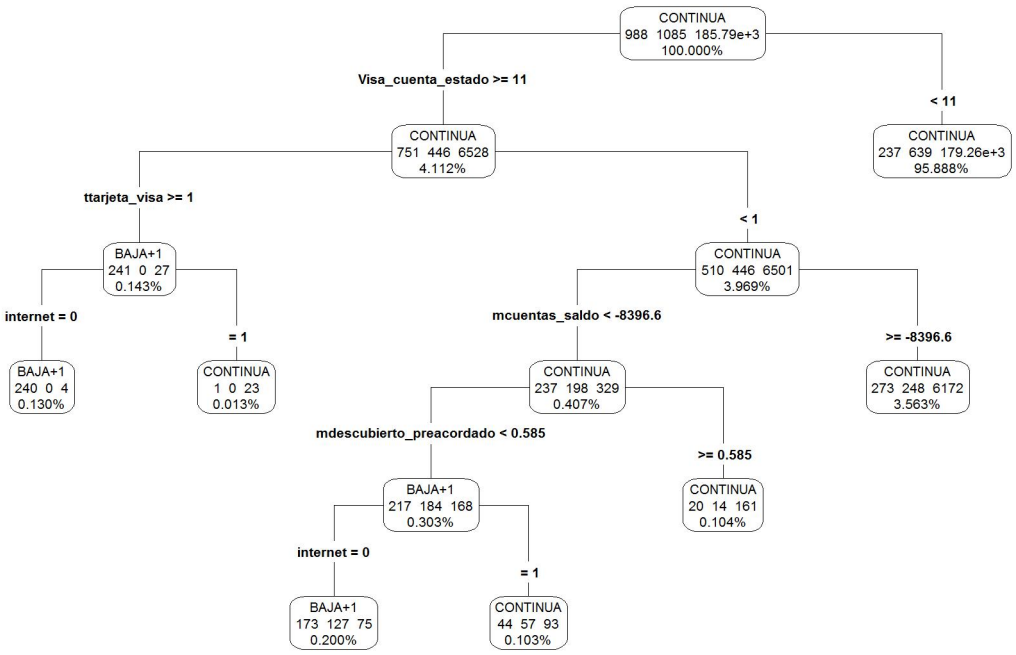
# /elementary/MiPrimerArbol_01.r

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:

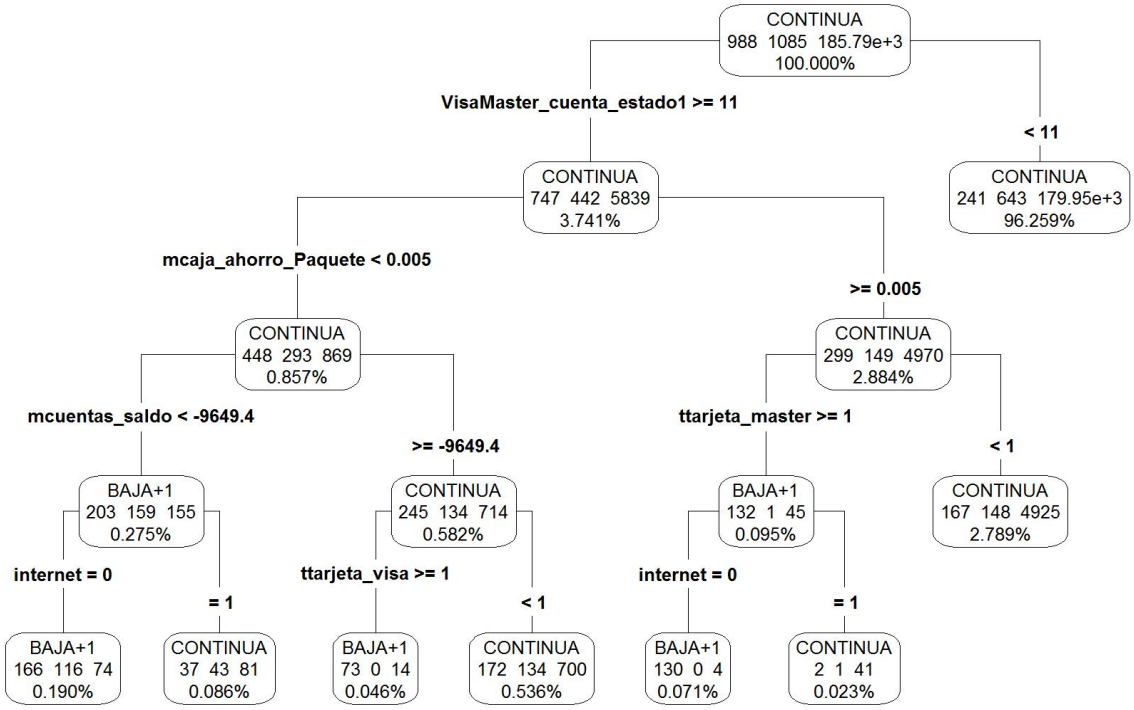
Volver a correr `MiPrimerArbol_01.r` cambiando en la llamada a rpart el parámetro cp=0.01 por cp=0.0 , teniendo el cuidado que la representación gráfica del arbol quede en un nuevo archivo llamado `arbol_01_bis.jpg` , comparar visualmente ambos árboles y calcular la ganancia de ese nuevo arbol .

- ¿ 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.

## /elementary/Modelo_aplicar_01.r

Plataforma de ejecución: PC local

Es la primer corrida donde se aplica un modelo usando los hiperparámetros ganadores.
El modelo se entrena sobre el periodo completo 201902, y se aplica al periodo 201904 .
Es muy importante notar que para la aplicación del modelo no se hace training/testing, los hiperparámetros se utilizan para generar el modelo sobre todos los datos de 201902.

Aqui aparece por primera vez el problema de las variables referidas al tiempo, fechas en el caso de este dataset.

Conceptualmente se explica el tema de eventos relativos y absolutos, independientemente de como este representada esa variable en el dataset.

Se hace imperiosa la realidad de modifcar el dataset pasando las fechas de absolutas a relativas.

| pcorte=0.025 rpart vxval=0 vmaxdepth=16 vminbucket=6 vminsplit=20 vcp=0 | | | |
|---|---|---|---|
| Generación Modelo | Aplicación Modelo | Ganancia | AUC |
| 201902.txt | 201904.txt | 5,659,000 | 0.8227 |

## /cloud/cloud_optim.r

Plataforma de ejecución: Cloud, 24 GB RAM, 2 vcpu , HDD estandar

Dado que a partir de este momento la mayoria de los programas se comenzará a correr en la nube, se muestra una incompatibilidad entre la funcion `fread` de la librería `data.table` y la implementación de Google Cloud de los Buckets de datos los que no son un file system común; se brinda una solución a esa incompatibilidad.

Se miden los tiempos de lectura con la funcion `fread` del package `data.table` en estas tres versiones

- `fread( karchivo_entrada,`
- `fread( cmd=paste( "cat", karchivo_entrada), ...`
- `fread( cmd=paste( "gunzip -cq", karchivo_entrada_zip), ...`

| **Tiempos de lectura de paquete_premium (en segundos)** | | | |
|---|---|---|---|
| **origen** | **.txt** | **cat .txt** | **.zip** |
| local | 62 | 88 | 76 |
| bucket | 3,557 | 93 | 69 |

Hay una incompatibilidad severa para la lectura de archivos desde el bucket mediante un fread simple.
Como conclusión o se lo lee de un .zip o con el shortcut de `fread( cmd=paste( "cat", karchivo_entrada), … )` .

## /FeatureEngineering/fechas_relativas.r

Plataforma de ejecución: Cloud, con suficiente espacio en disco y RAM (descubrir)

Es la corrida de Modelo_aplicar_01.r surge que si se entrena un modelo en un dataset con fechas absolutas, entonces hay problemas al aplicarlo.
Por eso, se relativizan las fechas a la fecha de la foto del dataset .

foto_fecha=201904 es en realidad foto_tomada_en_instante=20190501-00:00:00

Todas las variables de tipo fecha, se pasan relativas a la cantidad de dias que faltan para la foto_fecha , con lo cual quedan los campos quedan expresados en dias que pueden tomar un valor positivo o negativo.

## /elementary/Modelo_aplicar_02.r

Plataforma de ejecución: Cloud, 8 GB RAM, 1 vcpu

Es lo mismo que Modelo_aplicar_01.r pero se utilizan los archivos que tienen fechas relativas 201902_dias.txt y 201904_dias.txt

| pcorte=0.025 rpart vxval=0 vmaxdepth=16 vminbucket=6 vminsplit=20 vcp=0 | | | |
|---|---|---|---|
| Generación Modelo | Aplicación Modelo | Ganancia | AUC |
| 201902.txt | 201904.txt | 5,659,000 | 0.8227 |
| 201902_dias.txt | 201904_dias.txt | 5,900,500 | 0.8325 |

Se observa como aplicando correctamente el modelo, se obtiene una ganancia mayor 5,136,000 vs 5,659,000

A partir de ahora se trabajará con los archivos de fechas relativas `_dias.txt`

# Arbol
# de
# Probabilidad

librería : rpart

# Rpart

En el modulo incial se terminó con `/elementary/MiPrimerModelo_03.r` mostrando la necesidad de optimizar los parámetros del algoritmo `rpart` para este dataset, al menos por ahora para la foto 201902 .

Se introduce el concepto de Hyperarameter Optimization tambien llamado Hyperparameter Tuning.

## `/rpart/rpart_tune_gridsearch_01.r`

Plataforma de ejecución: PC local

Es la primer corrida de un modelo completo, donde se hace Montecarlo Estimation con un solo mes.
El Hyperparameter Optimization se implementa con la técnica de Grid Search con loops anidados.

La salida que se genera ya está en el formato que se utilizará para todo del curso, donde se compararán otros algoritmos, otros tipos de optimización de hiperparámetros, otros meses y conjuntos de meses tanto para training como para testing ademas de la clase con la cual se trabajará (ternaria, binaria1, binaria2, ... )

Se insta al alumno a que haga la corrida en su laptop/PC local y que vaya monitoreando la salida a lo largo de los varios dias que demora, ademas de monitorear el consumo de CPU y memoria RAM, mostrándoles que la libreria rpart es single threaded.
Deben tener contacto emocional con la desesperante subutilización de CPU que hace el rpart , ya que en la mayoria de las laptops utilizará apenas el 25% de la CPU disponible.
Deben tener contacto emocional con la ineficiencia del Grid Search cuando recorre innecesariamente regiones del espacio de búsqueda en las que es obvio (por resultados anteriores) que solo hay valores suboptimos.

Debe quedar totalmente claro a los alumnos lo indispensable de la búsqueda de hiperparámetros, al hacerles notar la gran dispersión que hay en los valores de la métrica ( ganancia o AUC) según los parámetros pasados al algoritmo.

Los alumnos avanzados se animarán a crear varios programas a partir de este, que visiten distintas regiones del grid search y ponerlos a correr en paralelo en la misma maquina, y hasta en maquinas distintas.

Los alumnos muy avanzados encontrarán una librería que implemente `lapply` paralelo.

Data la inferioridad del método Grid Search de Hyperparameter Optimization, esta es la primera y última vez que se utiliza.

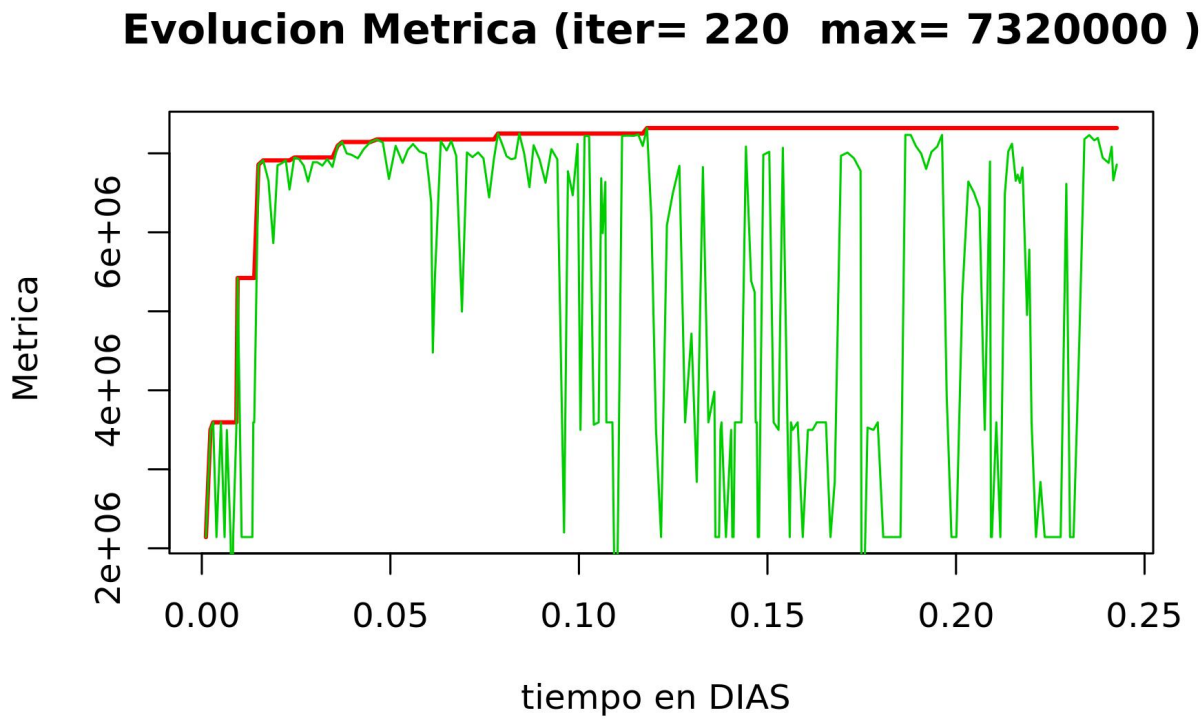
| Parametros | ganancia | |
|---|---|---|
| | 201902 | 201904 |
| xval=0, maxdepth=14, minbucket=6, minsplit=20, cp=0 | 7,289,667 | 5,913,500 |
| xval=0, maxdepth=12, minbucket=5, minsplit=20, cp=0 | 7,360,333 | 5,715,500 |
| xval=0, maxdepth=10, minbucket=16, minsplit=50, cp=5e-04 | 6,541,333 | 5,188,500 |

## /rpart/rpart_tune_MBO_01.r

Plataforma de ejecución: Cloud, 4 vcpu, 16 GB RAM

Es la primer corrida de Hyperparameter Optimization usando el método de Bayesian Optimization, liberia `mlrMBO`
Se muesta el complejo seteo de esta librería y la forma de especificar el tipo y rango de los hiperparametros.
Se muestra la decisión de ingenieria de como representar en `mlrMBO` a los parametros de `rpart` `minsplit` y `minbucket`, donde siempre `minsplit > minbucket`

El objetivo es que los alumnos comparen la diferencia de velocidad con el Grid Search y noten la superioridad absoluta de la optimización bayesiana .



## /rpart/rpart_aplicar_01.r

Plataforma de ejecución:  Cloud, alcanza con  1 vcpu, 20 GB RAM

Aquí se produce realmente un gigantesco avance, es un alto en el camino para reflexionar lo hecho hasta ahora y replantearse ciertas suposiciones.

Hasta el momento se entrenaba con el mes 201902  con Montecarlo Estimation y se aplicaba el modelo al periodo 201904 en donde se evaluaba la ganancia.
Está claro que el último periodo con el campo clase completo es 201902 y que la generacion se debe hacer a lo sumo con el período 201904, pero porqué hay que limitarse a entrenar solamente con el periodo 201902  ?
Que tal si se entrena en el período 201903, o el 201902, etc
O que sucederia, si se entrena en la union de varios periodos, digamos el perido que surge de la union de estos cuatro meses { 201902, 201901, 201812, 201811 }

Se utilizan los parámetros  de rpart

```
  xval =0,  maxdepth =16,  minbucket =6,  minsplit=20,  cp=0
```

y se prueban combinaciones del tipo

```
mes_aplicacion= 201904_dias.txt
mes_generacion= 201902_dias.txt

mes_aplicacion= 201904_dias.txt
mes_generacion= 201901_dias.txt

mes_aplicacion= 201904_dias.txt
mes_generacion= 201812_dias.txt
```

así como también combinaciones mas complejas del tipo

```
mes_aplicacion= 201904_dias.txt
mes_generacion= {201902_dias.txt, 201901_dias.txt, 201812_dias.txt }
```

El alumno avanzado querrá agrandar  aun mas  el  dataset de generacion  { 201902_dias.txt, 201901_dias.txt, 201812_dias.txt, 201811_dias.txt, 201810_dias.txt, 201809_dias.txt, 201808_dias.txt, 201807_dias.txt, 201806_dias.txt, 201805_dias.txt, 201804_dias.txt, 201803_dias.txt}

Se debe atender a que esto no constituye feature engineering, ya que se está trabajando con las variables originales del dataset (salvo las fechas que se pusieron relativas a la fecha de la foto).

Como ejemplo de lo que se logra

| Rpart modelo constante | | |
|---|---|---|
| xval =0, maxdepth =16, minbucket =6, minsplit=20, cp=0 | | |
| Período Entrenar | Período Aplicar | Ganancia |
| 201902_dias.txt | 201904_dias.txt | 5,900,500 |
| 201901_dias.txt | 201904_dias.txt | 5,566,500 |
| 201812_dias.txt | 201904_dias.txt | 6,493,000 |
| 201902_dias.txt, 201902_dias.txt, 201812_dias.txt | 201904_dias.txt | 7,105,500 |
| 201902_dias.txt, 201902_dias.txt, 201812_dias.txt, 201811_dias.txt, 201810_dias.txt, 201809_dias.txt | 201904_dias.txt | 7,179,500 |
| 201902_dias.txt, 201902_dias.txt, 201812_dias.txt, 201811_dias.txt, 201810_dias.txt, 201809_dias.txt, 201808_dias.txt, 201807_dias.txt, 201806_dias.txt, 201805_dias.txt, 201804_dias.txt, 201803_dias.txt | 201904_dias.txt | 7,076,000 |

Notar que la ganancia es máxima cuando se genera el modelo sobre 6 meses

Notar que siempre estamos midiendo en 201904_dias.txt , pero que si se entrena en 6 meses de historia la ganancia pasa de $ 5,900,500 a $ 7,179,500

/rpart/rpart_tune_MBO_02.r

Plataforma de ejecución:  Cloud, alcanza con  1 vCPU, 20 GB RAM

Hasta ahora, al elegir un mes, la estimación de la ganancia y la AUC se hacia mediante Montecarlo Estimation ,  en nuestro caso dividimos el dataset 5 veces en 70% para training y 30% para testing, entrenamos en training, medimos ganancia y AUC en testing,  y al final promediamos las cinco ganancias y las cinco AUCs .

¿ Qué pasaría si entrenamos en 201902_dias.txt  y  testeamos en 201904_dias.txt  ?  ¿ Cómo comparan esos resultados con Montecarlo Estimation ?
Notar que se sigue cumpliendo que se mide la ganancia en datos que NO fueron utilizados para entrenar.

| **Eligiendo la mayor ganancia en 201902** (como corresponde!) | | | | | | |
|---|---|---|---|---|---|---|
| **Actual** | | **Futuro** | | | **Ganancia** | |
| **Train** | **Test** | **Train** | **Test** | **Montecarlo** | **201902** | **201904** |
| 201902_dias | 201902_dias | 201904_dias | 201904_dias | SI | 8,031,333 | 6,123,000 |
| **201812**_dias | 201902_dias | 201904_dias | 201904_dias | NO | 8,276,500 | 7,441,000 |

| **Haciendo TRAMPA,  eligiendo la mejor ganancia en 201904** | | |
|---|---|---|
| | **Ganancia** | |
| **Montecarlo** | **201902** | **201904** |
| SI | 7,434,333 | 8,310,000 |
| NO | 7,110,500 | 8,434,500 |