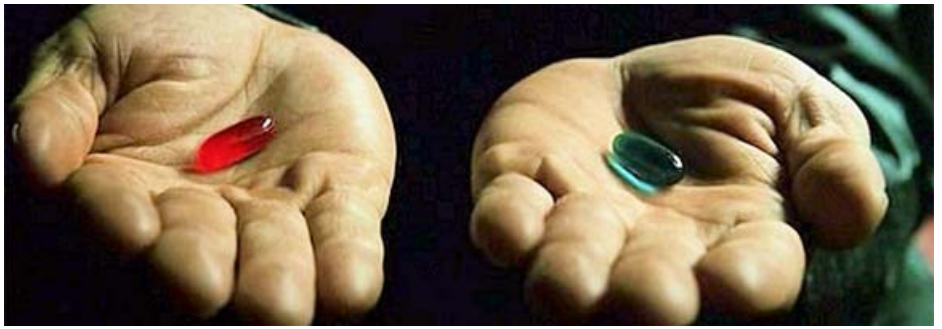
Repensando el overfitting

The problem is not people being uneducated.

The problem is that people are educated just enough to believe what they have been taught, and not educated enough to question anything from what they have been taught.

Richard Feynman



https://www.youtube.com/watch?v=Mces9bHCBoE

Bibliografía

La bibliografía se brinda en

Cinco niveles de Complejidad

inspirado en

https://www.youtube.com/watch?v=QcUey-DVYjk

Bibliografía tradicional

1. Overfitting
2. Bias and Variance
3. Overfitting vs. Underfitting: A Complete Example
4. Holy Grail for Bias-Variance tradeoff, Overfitting & Underf
5. On comparing Classifiers: Pitfalls to Avoid and a Recomme

Creamos diversos árboles entrenando en todo 202011 y los aplicamos a los datos del futuro de 202101 (Public Leaderboard)

medimos ganancia en 202011 y 202101 (Public Leaderboard)

901_rpart_alturas.r

```
for( vmaxdepth  in 4:30 )
{

 #genero el modelo
 modelo <-
    rpart(formula= "clase_binaria ~ .",
          data= dataset_entrenar,
          model= TRUE,
          xval= 0,
          cp= 0,
          minsplit= 5,
          maxdepth=  vmaxdepth
         )
```


Overfitting
30000000
25000000
20000000
15000000
10000000
5000000
0
0
5
10
15
20
25
30
Noviembre
Private

MUY Importante

El overfitting NO es la diferencia entre las curvas de training y testing.
Incluso, en Gradient Boosting vamos a ver curvas de training perfectas, muchísima diferencia con testing, y esos van a ser los mejores modelos !

- Sin la menor duda hay un punto óptimo de complejidad del modelo el cual maximiza la ganancia en datos nuevos, pero eso *no se detecta* en training

- ¿Cómo evito el overfitting?

- ¿Cómo encuentro los hiperparámetros óptimos del árbol que me generen la mayor ganancia en datos nuevos (que jamás fueron vistos al entrenar)?

Solución

Entrenar el modelo en una porción de los datos pero medir la ganancia del modelo en otra porción de los datos, que no debe haber sido vista durante el entrenamiento.

Una vez establecido el mecanismo anterior, el problema pasa a ser de optimización de la ganancia en testing, pudiendo utilizar una técnica burda como Grid Search o eficiente como Bayesian Optimization

Entrenar y Testear

- 70% training , 30% testing (inestable!)

- Montecarlo Estimation (repetir n veces train/test)
- 5-fold cross validation
- 5 repeated 10-fold cross validation

- Leave One Out (método teórico)

ok hago cross-validación,
me da tal o cual resultado

¿y ahora qué?

¿Cómo sigue la construcción del modelo del árbol?

¿Cómo comparo los errores o performance de diferentes modelos basado en esto?

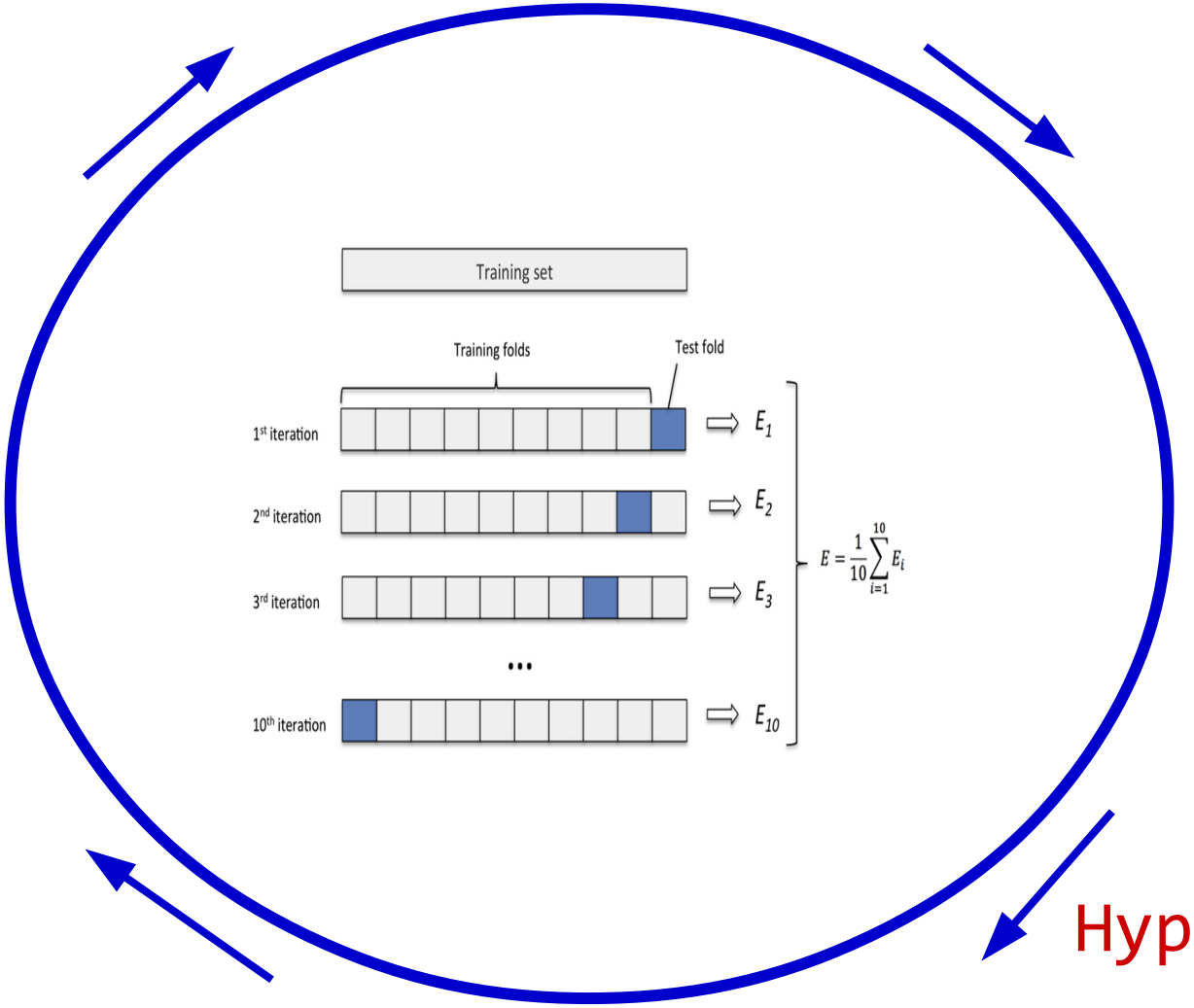
ok hago cross-validación,
me da tal o cual resultado

¿y ahora qué?

¿Cómo sigue la construcción del modelo del árbol?

¿Cómo comparo los errores o performance de diferentes modelos basado en esto?

First slow Step Hyperparameter Optimization



N cycles of Bayesian Optimization

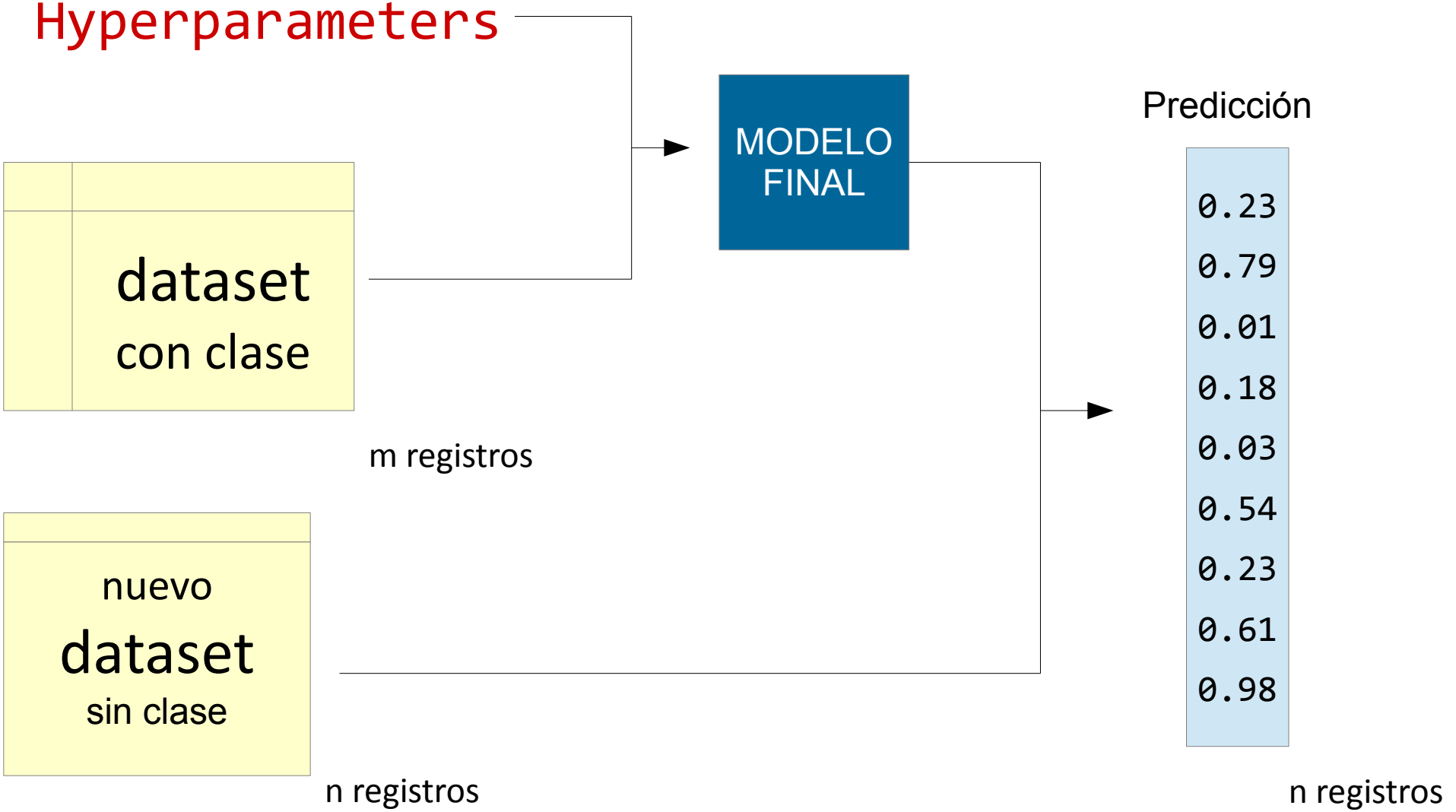
Optimal Hyperparameters

```
cp=-0.44, minsplit=1064,
maxdepth=5, minbucket=524
```

Last fast Step Apply Best Model

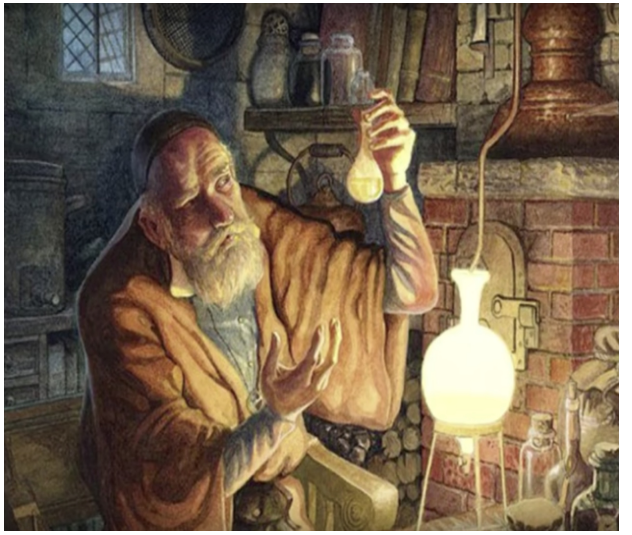
```
cp=-0.44, minsplit=1064,
maxdepth=5, minbucket=524
```

Optimal Hyperparameters



El ciclo de la optimización de modelos predictivos

Hyperparameter tuning
Bayesian Optimization
(varias horas)



Data Drifting
+
Feature Engineering
+
ciencias ocultas

¿Cuánto lleva hacer una optimizacion bayesiana de 100 iteraciones de rpart con sus 4 hiperparámetros utilizando 5-fold cross validation?

- La Bayesian Optimization hace 4*4 + 100 = 116 iteraciones
- En cada iteracion se hace 5-fold cross validation, se entrenan 5 arboles sonbre el 80% de los datos y se aplica cada uno al 80% restante
- Al final, con los hiperparámetros de la mejor iteración, se entrena un arbol final sobre TODO el dataset y se aplica a los datos del futuro.

Se generan 116*5 + 1

581 arboles

Por más que es rentable económicamente,
¿realmente hace falta generar 581 árboles y evaluarlos
para encontrar el mejor modelo?

¿ Hace falta tanto esfuerzo para encontrar
`<cp=-0.44, minsplit=1064,`
`maxdepth=5, minbucket=524>` ?

despues de todo, el final es construir un solo modelo
usando esos parámetros
sobre todos los datos de 202011

Lo explicado hasta aquí es la teoría clásica del modelado predictivo, y la solución tradicional al overfitting.

Estamos usando el estado del arte con la Bayesian Optimization.

Las soluciones expuestas son las usadas en la academia y en la industria.

Repensemos el Overfitting

Bibliografía

1. https://www.youtube.com/watch?v=6ZxIzVjV1DE
2. https://statweb.stanford.edu/~candes/talks/slides/Wald1.pdf
3. https://link.springer.com/article/10.1023/A:1007631014630
4. https://link.springer.com/article/10.1007/s10654-019-00517-2
5. https://arxiv.org/pdf/1605.03391.pdf

¿Por qué se produce el overfitting?

¿Es culpa de los datos,
o nuevamente el culpable es el algoritmo?

¿Por qué debo controlar el crecimiento del arbol con `cp, minsplit, maxdepth, minbucket` ?

¿Por qué el árbol no crece solo hasta cuando debe, sin importar la profundidad?

¿Por qué el parámetro `cp` no alcanza por si solo para controlar el crecimiento alocado del arbol?

Entiendo que training/testing o Cross Validation logran una buena estimación de como va a funcionar el modelo en datos nuevos, pero
¿Es realmente la única forma?

¿realmente hacen falta training y testing?

¿realmente hacen faltan hiperparámetros para controlar el crecimiento?

hemos aceptado ciertas ideas
como verdaderas
sin haberlas meditado
minuciosamente

Los canaritos se han utilizado en las minas de carbon para detectar monóxido de carbono y otros gases tóxicos antes que afecten al ser humano.

Los canaritos actuan como una *especie sentinela* :
un animal más sensible a los inoloros e incoloros gases venenosos.
Si el canarito se desvanecía o moría, eso alertaba a los mineros a evacuar la mina inmediatamente.

Variable “canarito”

Agregar al dataset
variables numéricas reales random
con distribución uniforme
en el intervalo continuo [0.0, 1.0]

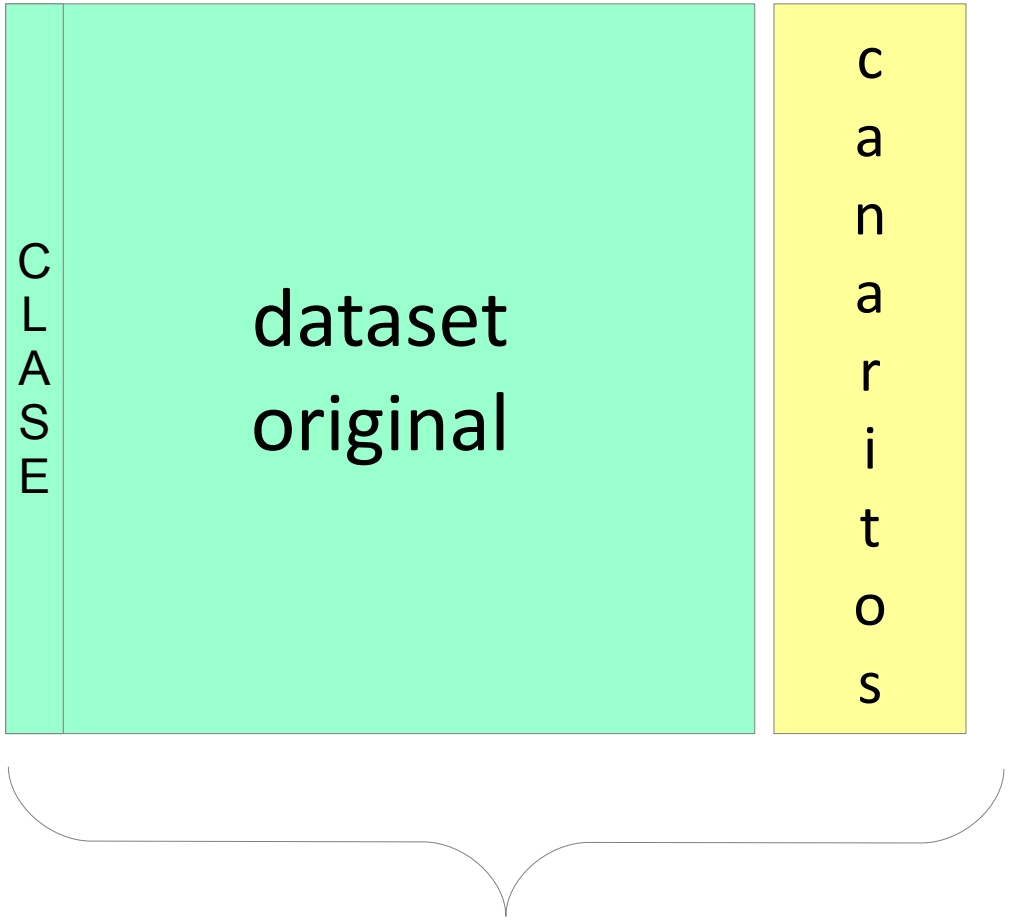
En lenguaje R se hace con `runif()`

```
for( i in 1:100 )  dataset[ , paste0("canarito", i ) :=
                                runif( nrow(dataset))
                             ]
```

Propiedades de un “canarito”

Una variable “canarito” al ser random
no puede estar correlacionada con la clase
ni tampoco con ninguna variable del dataset
tampoco con ningún otro canarito

Una variable “canarito” jamás debería aparecer en un modelo predictivo


CLASE
dataset original
canaritos
Nuevo Dataset

Primer experimento

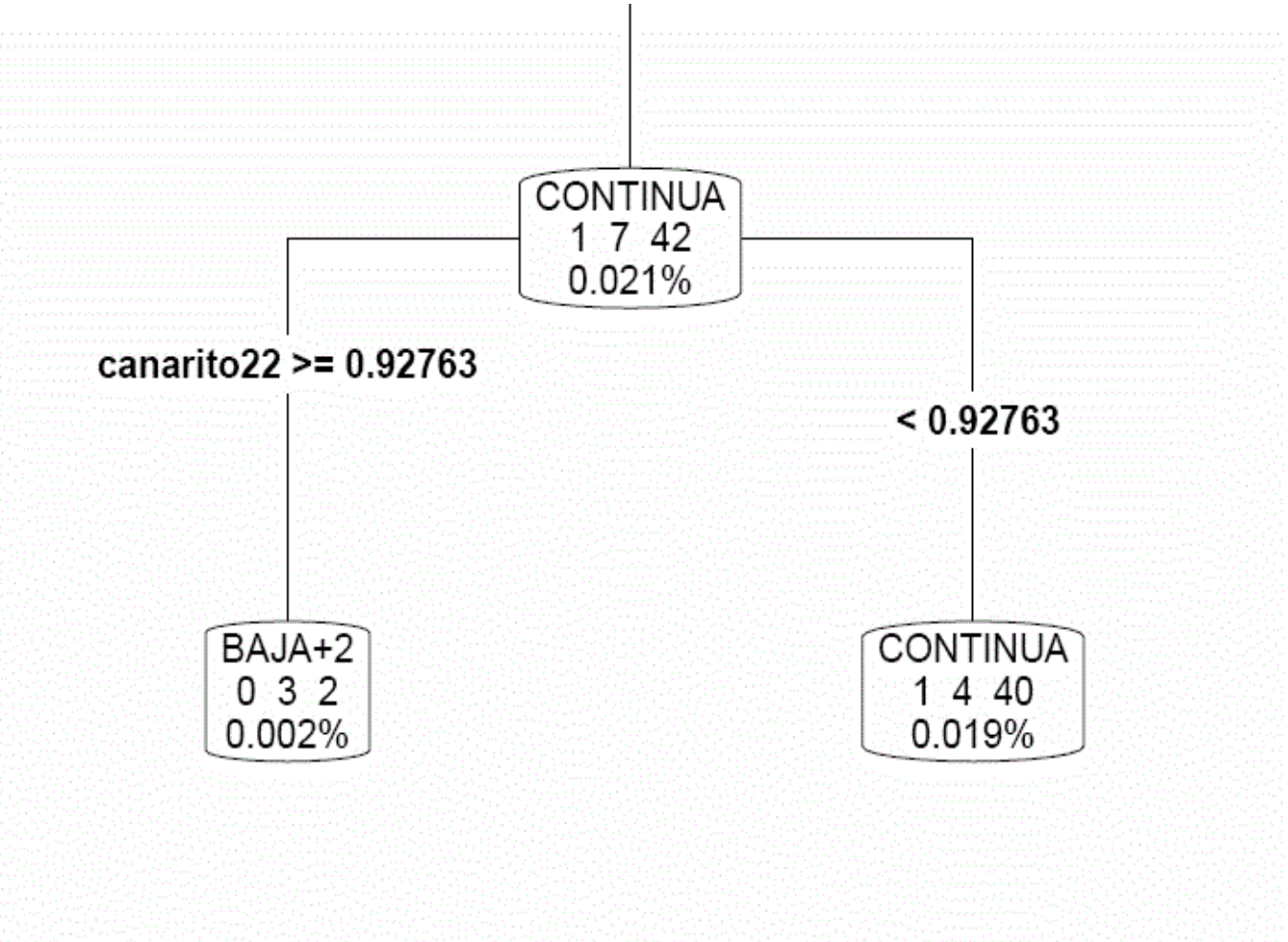
- Script `910_arbol_canaritos_prp.r`

- Agrego 100 nuevas variables canarito al dataset
  - `cp= 0.0`
  - `minsplit= 10`
  - `maxdepth= 10`

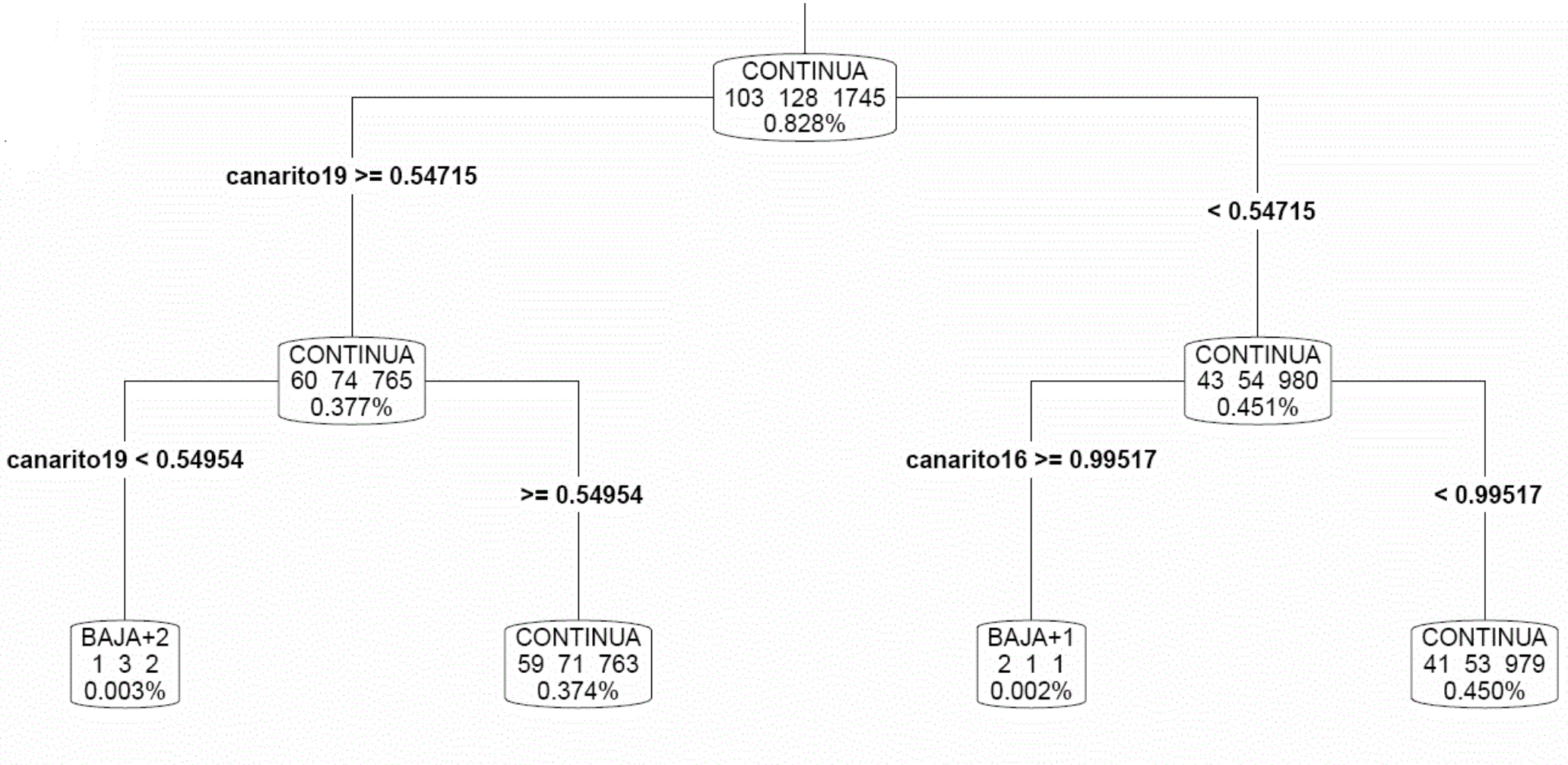
- Analizo el dibujo del arbol buscando canaritos

- Salida `arbol_canaritos.png`

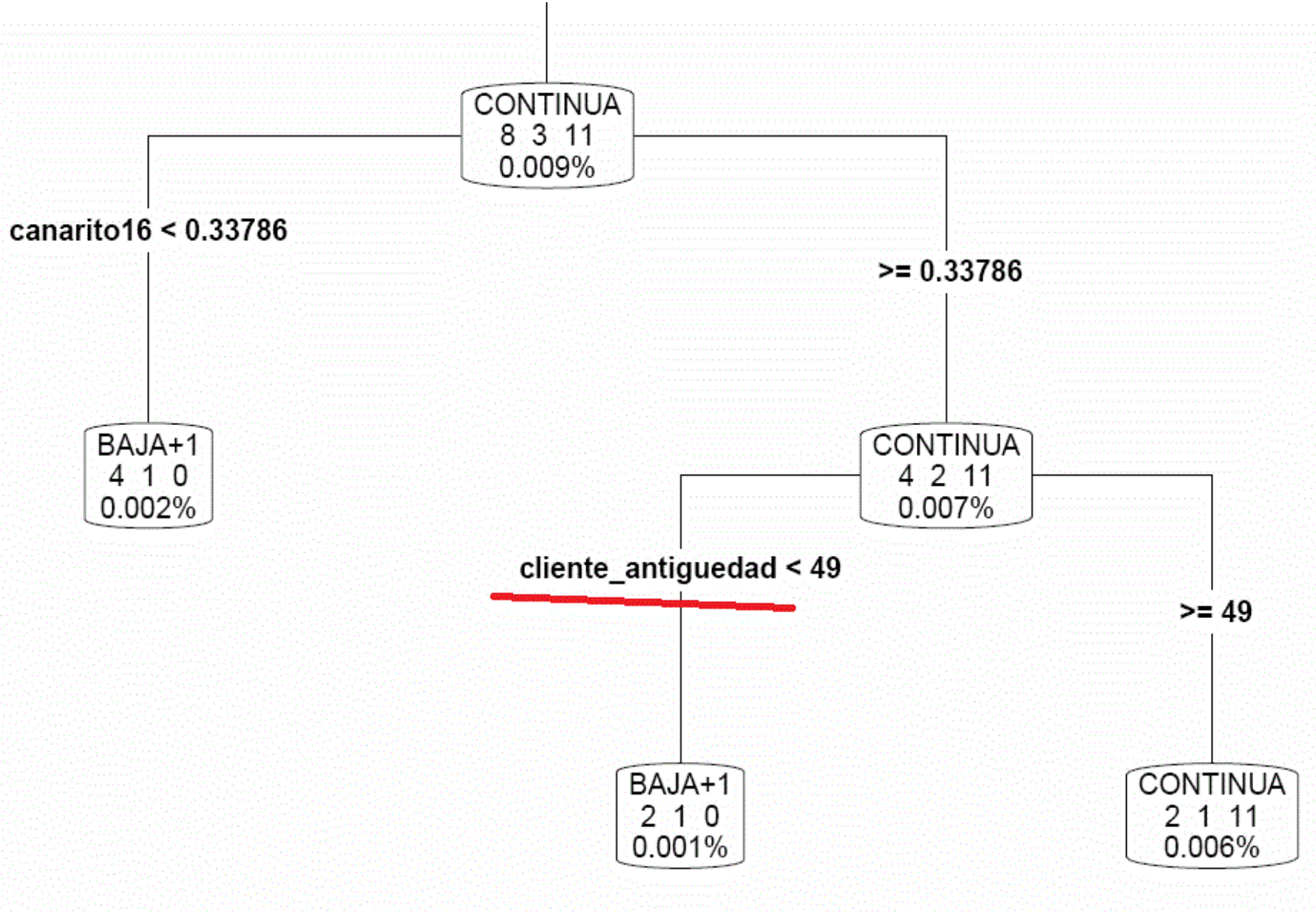
canarito al final



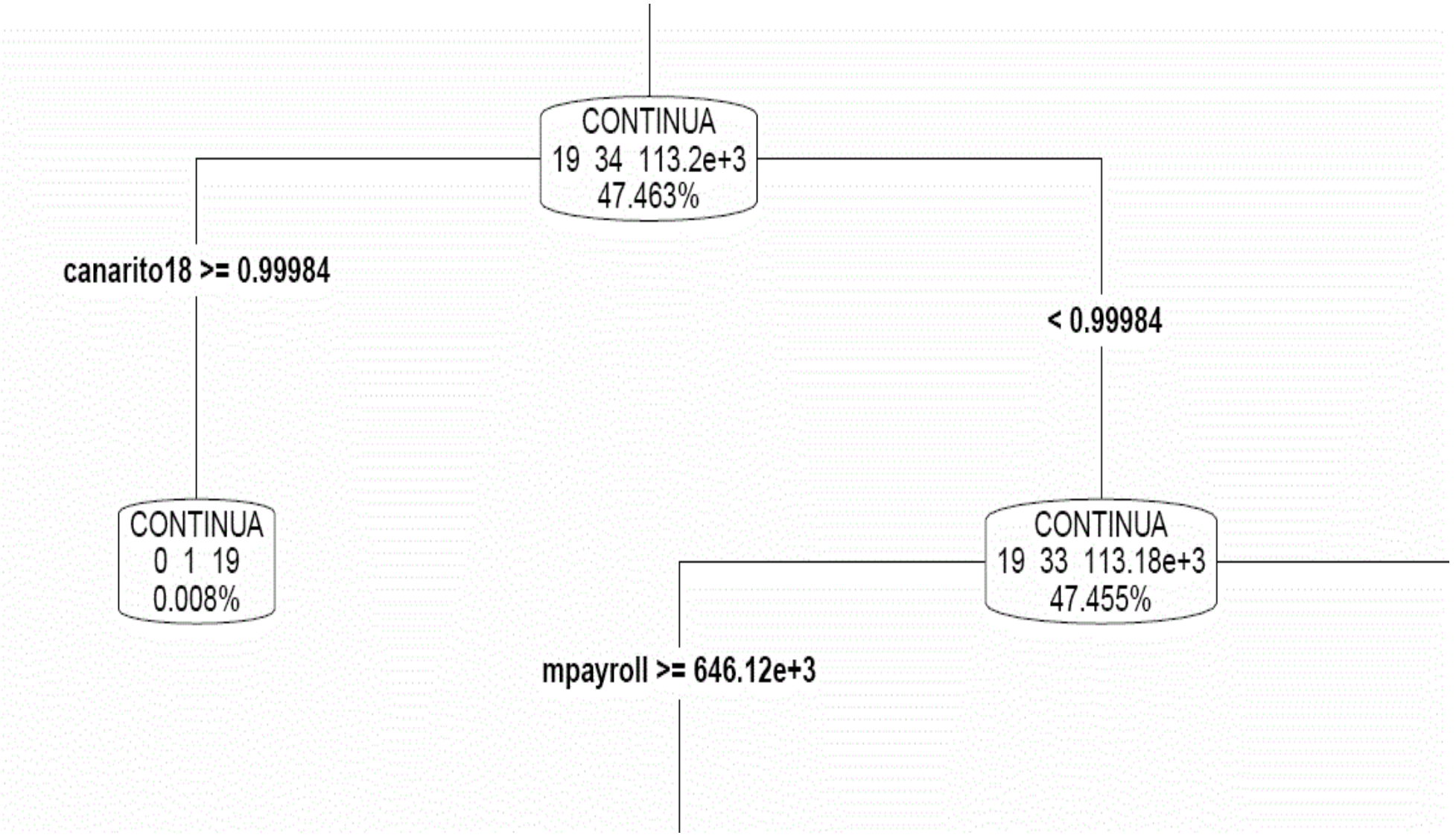
canaritos anidados



wtf ?


CONTINUA
8 3 11
0.009%
canarito16 < 0.33786
>= 0.33786
BAJA+1
4 1 0
0.002%
CONTINUA
4 2 11
0.007%
cliente_antiguedad < 49
>= 49
BAJA+1
2 1 0
0.001%
CONTINUA
2 1 11
0.006%

nodo “grande” cortado x canarito

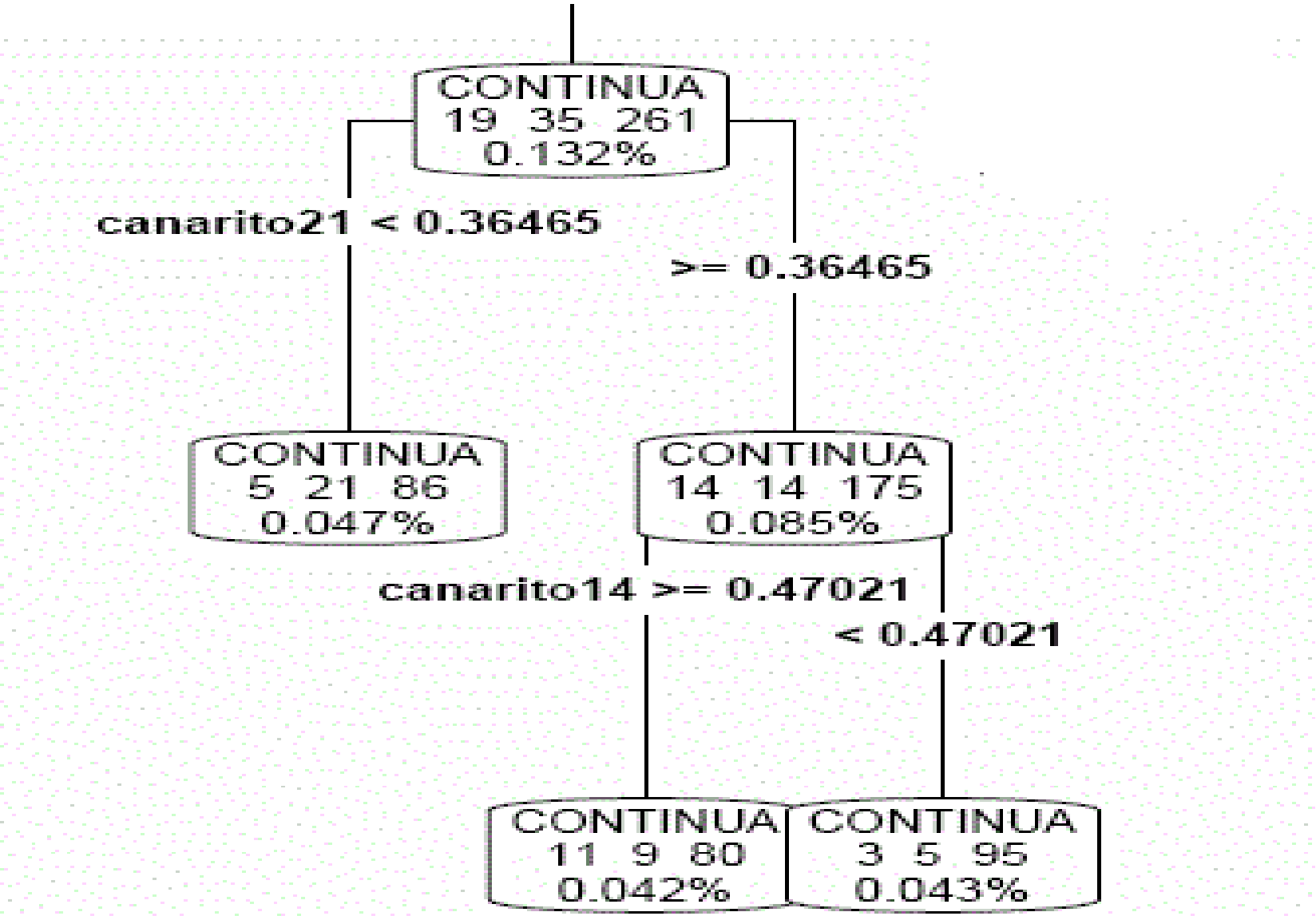


Pregunta

¿Qué sucede si exijo hojas grandes, por ejemplo de por lo menos 100 registros, siguen apareciendo canaritos?

`911_arbol_canarito_desconfiado.r`

minbucket = 100



Observaciones 1

- ¿ Cómo puede ser que teniendo más de 150 variables para cortar elija una variable canarito, que por ser random, JAMÁS puede estar correlacionada con la clase ?

- Cualquier correlación de canarito en los datos de 202011 es espúrea, NO SE VA A CUMPLIR en 202101

- ¿Tiene algun sentido estadístico el corte por un canarito?¿Depende de la cantidad de canaritos que se agregan al dataset?

Observaciones 2

- Los canaritos NO aparecen cerca de la raiz
- Los canaritos generalmente no aparecen en nodos grandes
- Algunos canaritos aparecen muy profundo, otros a profundidad intermedia.
- Si un canarito aparece en un nodo grande, ¿quiere eso decir que ninguna variable real es buena para cortar?

- ¿Que pasaría si podo, hago pruning, justo en los canaritos ?

Errors in adding components to a model,
usually called overfitting,
are probably the best known pathology
of induction algorithms.

Overfitting occurs when a
Multiple Comparison Procedure
is applied to model components

Afirmación

El modelo va a generalizar bien en datos nuevos si durante su construcción antes de agregar un nuevo elemento al modelo respondo esta pregunta :

¿Es esto realmente un patrón o es un simple producto de que por azar alguna de las variables tuvo suerte (dadas todas las variables que tengo) ?

Tercer Experimento

Tercer experimento

- Script `920_arbol_canaritos_pruning.r`

- Agrego 30 nuevas variables canarito al dataset
- Construyo el árbol sobre TODO [202011] (con canaritos)
- Dejo crecer el árbol todo lo que puede
- Hago pruning justo en los canaritos

- Calculo la ganancia del modelo pruning en los datos de 202101 (Kaggle)
- En ningún momento hago training/testing
- Se construye UN SOLO árbol

Como no puedo cambiar el código de rpart, utilizo la funcionalidad de prunning

```
#hago el pruning de los canaritos
#haciendo un hackeo a la estructura  modelo_original$frame
# -100 es un valor arbritrariamente negativo que jamas es
#generado por rpart

modelo_original$frame[ modelo_original$frame$var %like%
"canarito", "complexity"] <- -666

modelo_pruned  <- prune(  modelo_original, -666.0 )
```

Resumen entrenar rpart en [202011]

Método	Hiperparams	Leaderboard Privado	árboles construidos	tiempo (horas)
Bayesian Optimization 5-fold xval 5*(16 + 100) + 1	cp= -0.44 maxdepth= 5 minsplit= 1064 minbucket= 524	16.35	581	2.77
Stopping at Canarito's	Ninguno	16.40	1	0.02

x 150 speedup

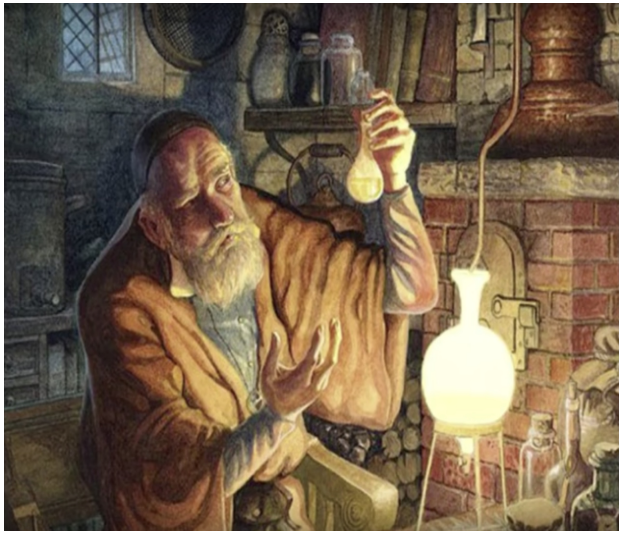
SIN hiperparámetros
SIN training/testing ni cross validation

Se utilizó una sola vCPU

repito el ciclo muchas veces

Ahora
menos de 2 minutos

(antes 2:47 horas)



Feature
Engineering
+
ciencias
ocultas

Características *Stopping at Canarito's*

- El método de los canaritos NO mejora la ganancia.
- Corre ~150 veces mas rápido que Bayesian Optimization con 5-fold Cross Validation

- No hacen falta los hiperparámetros que controlan el crecimiento del arbol `cp`, `maxdepth`, `minsplit`, `minbucket`
- SI hacen falta hiperparámetros para mejorar el arbol.

- No se usa training, testing ni nada parecido.

GRANDES Limitaciones

Stopping at Canarito's

- Es un método aproximado, el correcto consiste en hacer randomización de la clase (ver bibliografía)
- Es muy aproximado utilizar 30 canaritos, se deberían utilizar 100, y no cortar por la variable si las 5 mejores son todas canaritos
- Al no usar randomización de la clase, está siendo injusto con variables cuyo dominio tiene baja cardinalidad

- No se esta resolviendo el tema del BIAS en la estimación de la probabilidad en las hojas del árbol.

ATENCION

Usted deberá seguir utilizando la Optimización Bayesiana para los hiperparámetros del arbol que NO son `cp, maxdepth, minsplit, minbucket`

No está disponible en producción una versión de LightGBM que utilice canaritos, con lo cual ahí deberá utilizar Optimización Bayesiana PARA TODOS LOS HIPERPARAMETROS.

… y todo esto,
como se ve con
gradient boosting ?

https://www.youtube.com/watch?v=zmX7K8noikE