

Repensando el overfitting



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 & Underfitting
5. On comparing Classifiers: Pitfalls to Avoid and a Recommended

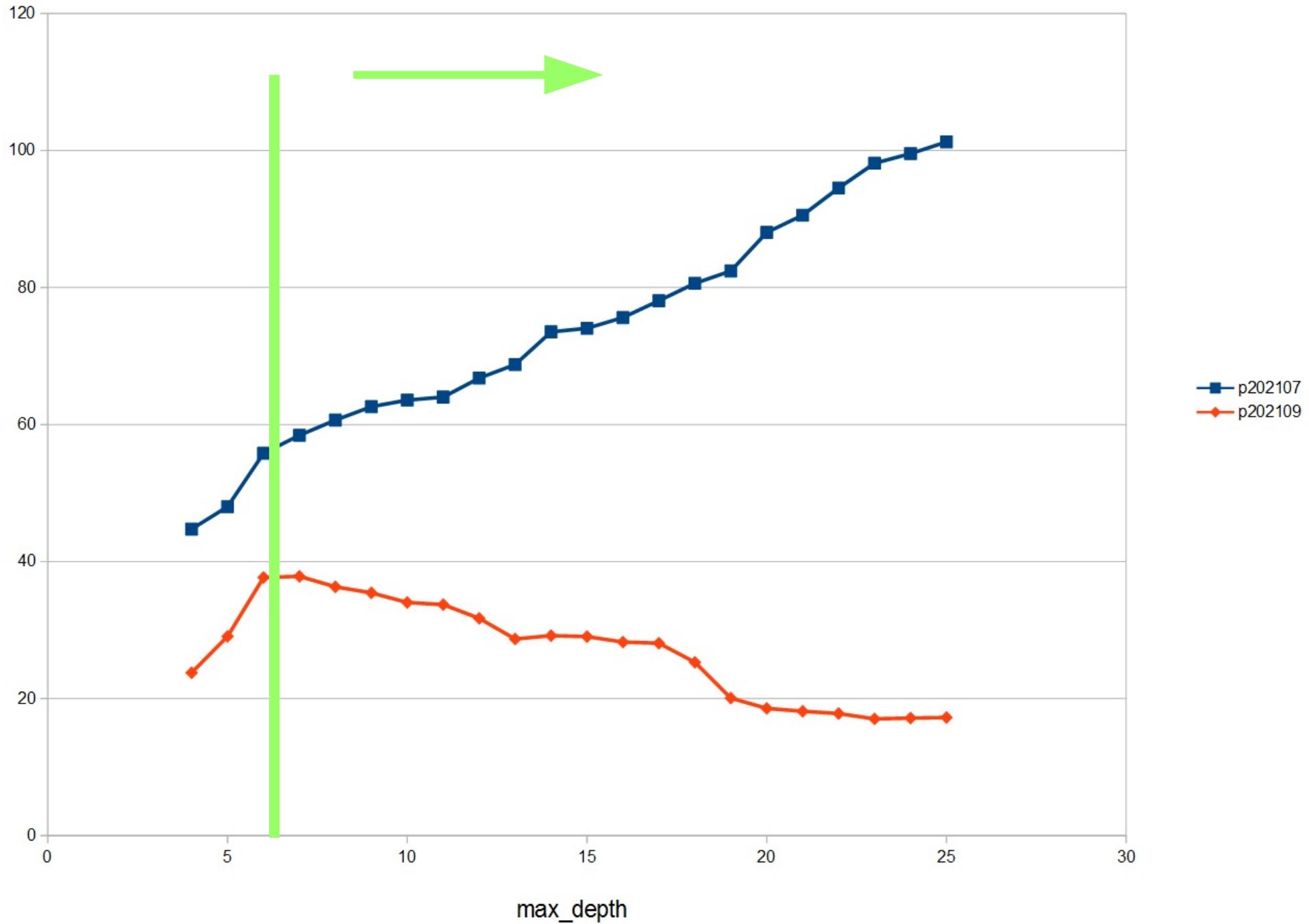
Creamos diversos árboles
entrenando en todo 202107
y los aplicamos a los
datos del futuro de
202109 (Public Leaderboard)

medimos ganancia en
202107 y 202109 (Public Leaderboard)

z480_rpart_alturas.r

```
for( vmaxdepth in 4:25 )
{
  #genero el modelo
  modelo <- rpart(formula= "clase_binaria ~ .",
                    data= dataset_entrenar,
                    model= TRUE,
                    xval= 0,
                    cp= 0,
                    minsplit= 5,
                    maxdepth= vmaxdepth
  )
```

Entrenamiento vs Futuro



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)

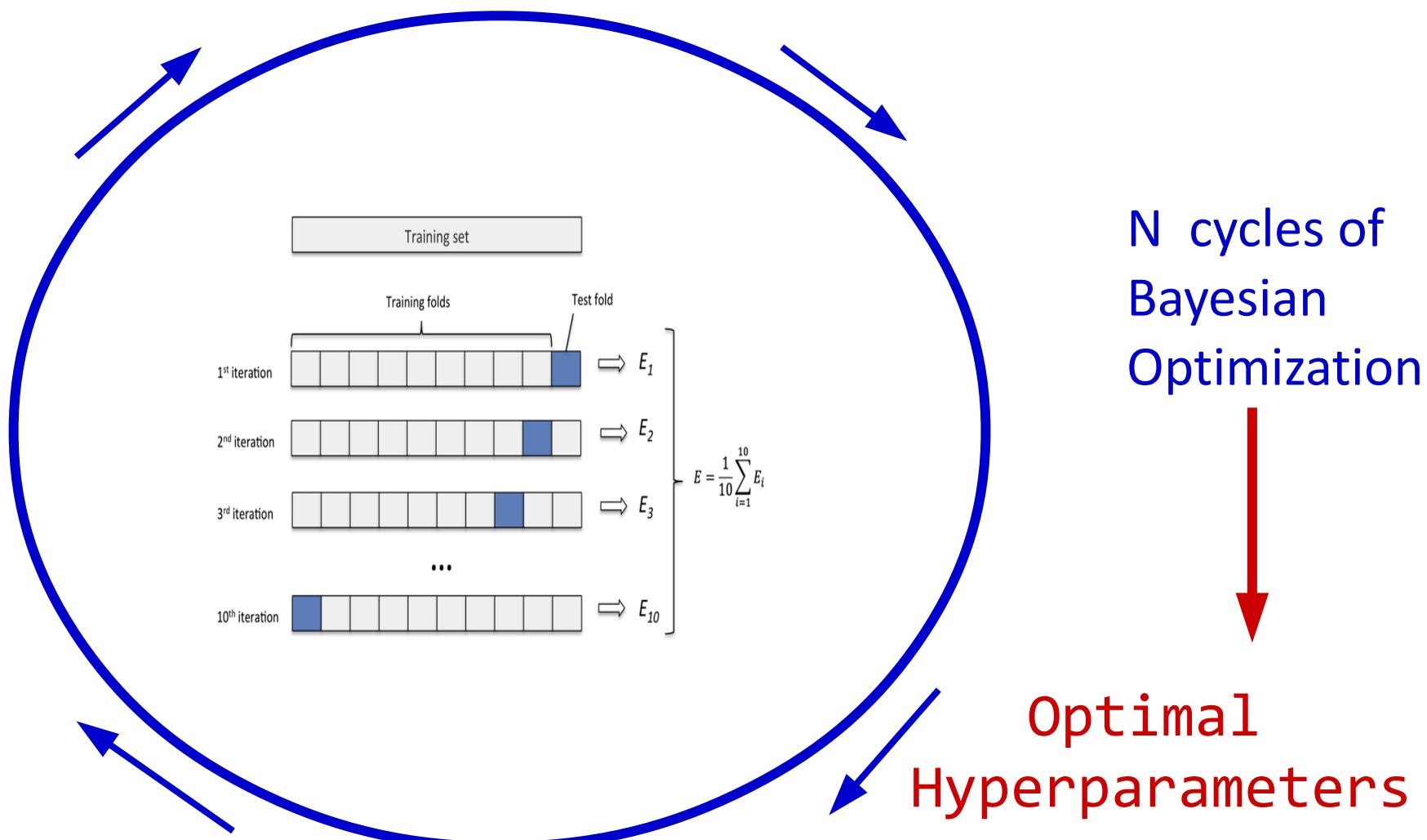
ok hago cross-validation,
me da tal o cual resultado

¿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

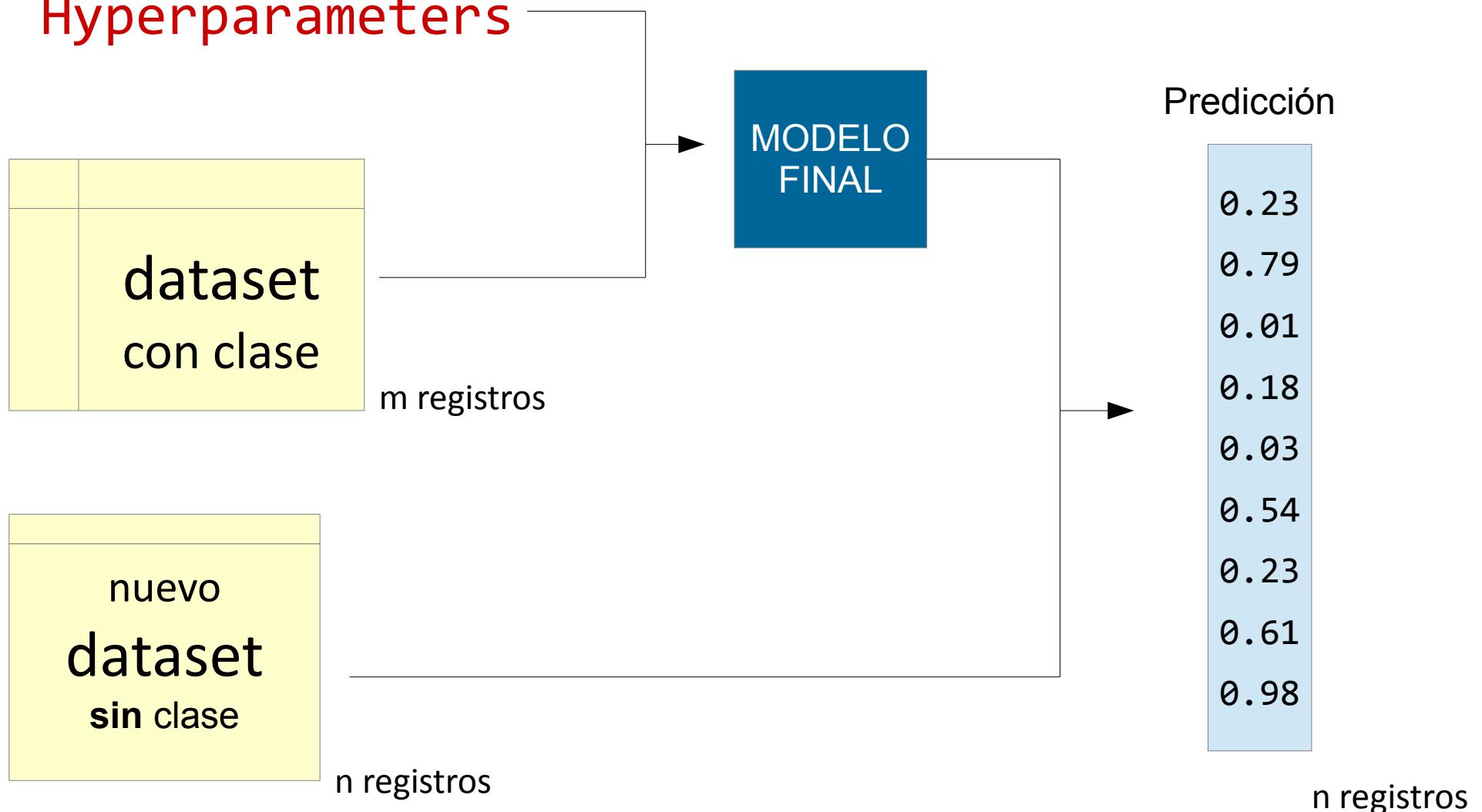


cp= -0.82, minsplit=769,
minbucket=8, maxdepth=6

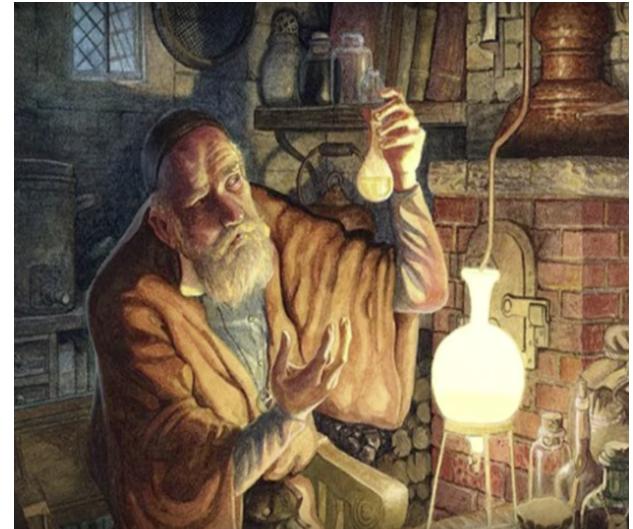
Last fast Step Apply Best Model

cp=-0.82, minsplit=769,
minbucket=8, maxdepth=6

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 optimización 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 iteración se hace 5-fold cross validation, se entrena 5 árboles sobre 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 árbol final sobre TODO el dataset y se aplica a los datos del futuro.

B.O. : se generan $116^*5 + 1$
581 árboles

Grid Search “exhaustivo”:
se generan $2500^*5 + 1$
12501 árboles

21.5 x speedup

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=-0.82, minsplit= 769,`
`minbucket= 8, maxdepth=6>` ?

después de todo, el proceso final
es construir **un solo modelo**
(usando esos parámetros)
sobre **todos** los datos de 202107

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`, `minbucket`, `maxdepth` ?

¿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 descontrolado* del árbol?

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 carbón para detectar monóxido de carbono y otros gases tóxicos *antes* que afecten al ser humano.

Los canaritos actúan como una *especie sentinela* : un animal más sensible a los inodoros 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]

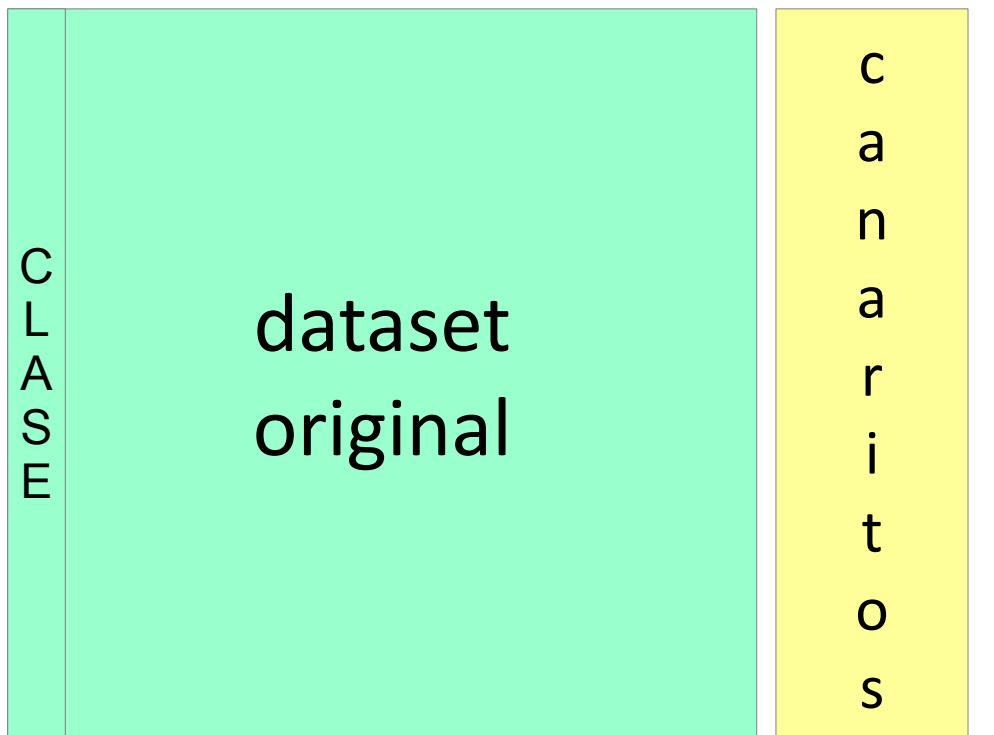
En lenguaje R se hace con `runif()`

```
for( i in 1:30 ) 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

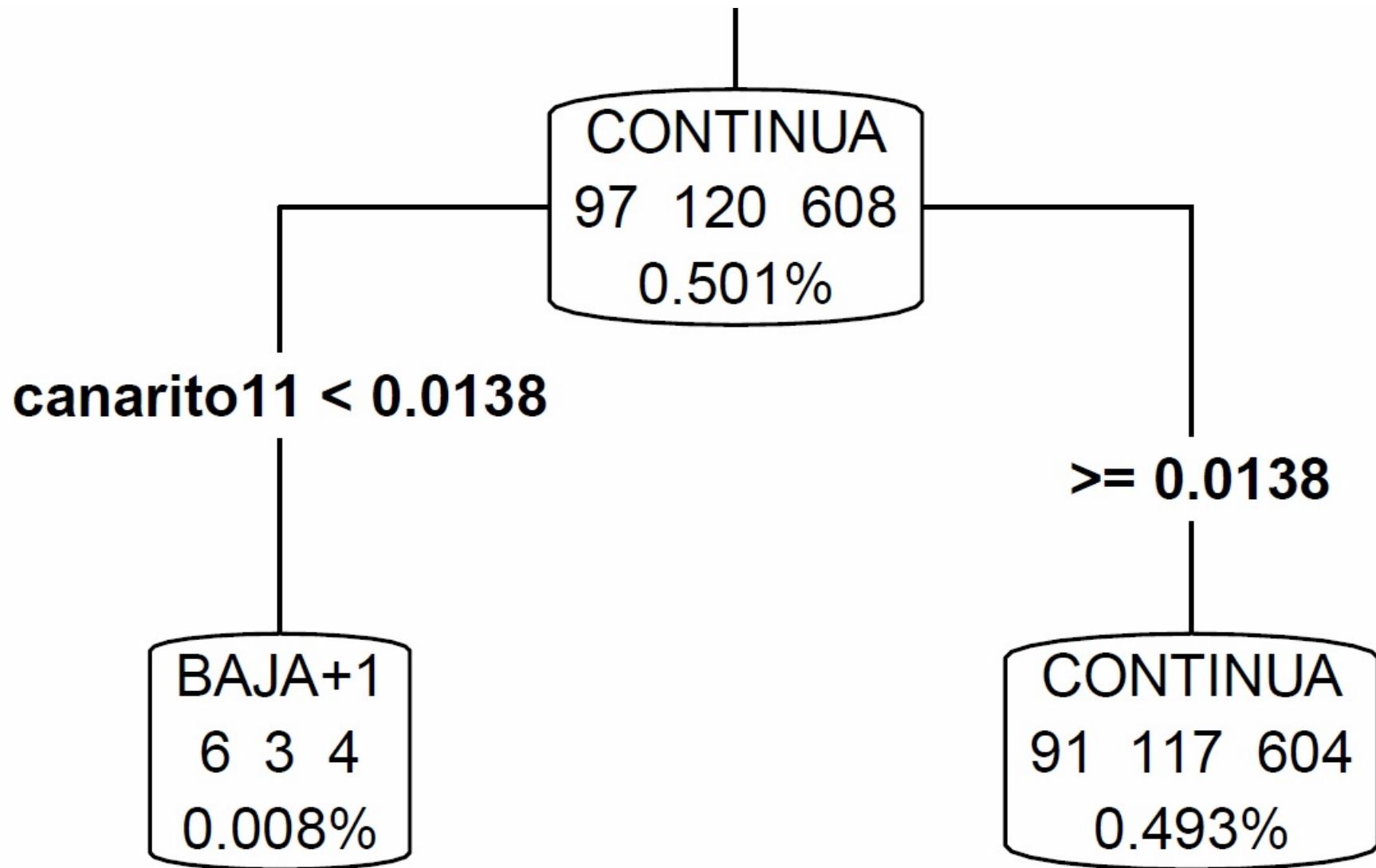


Nuevo Dataset

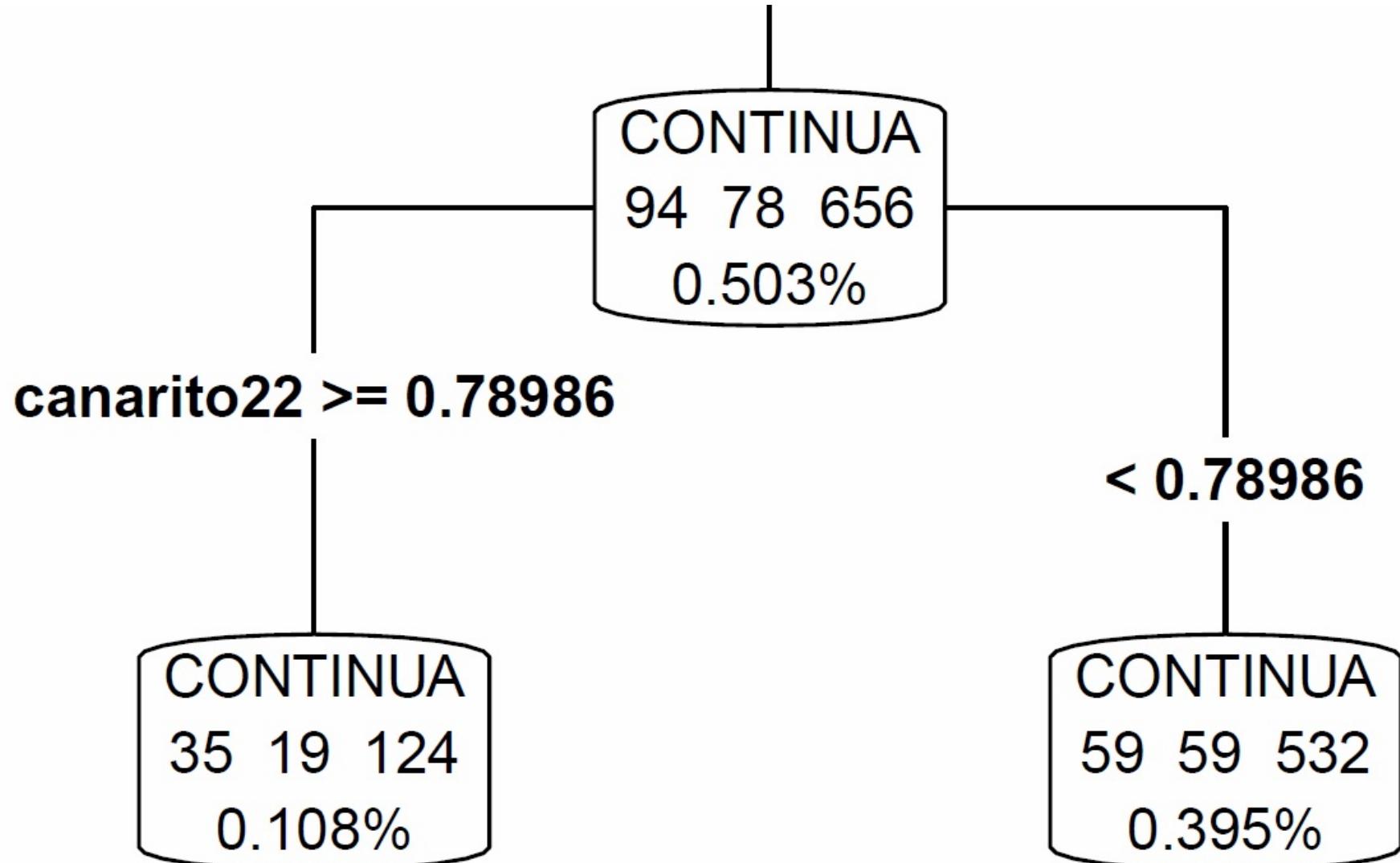
Primer experimento

- Script `z481_rpart_canaritos_prp.r`
- Agrego 30 nuevas variables canarito al dataset
 - `cp= -0.82`
 - `minsplit= 769`
 - `Minbucket= 8`
 - `maxdepth= 6`
- Analizo el dibujo del árbol buscando canaritos
- Salida `exp/EA4810/arbol_canaritos.pdf`

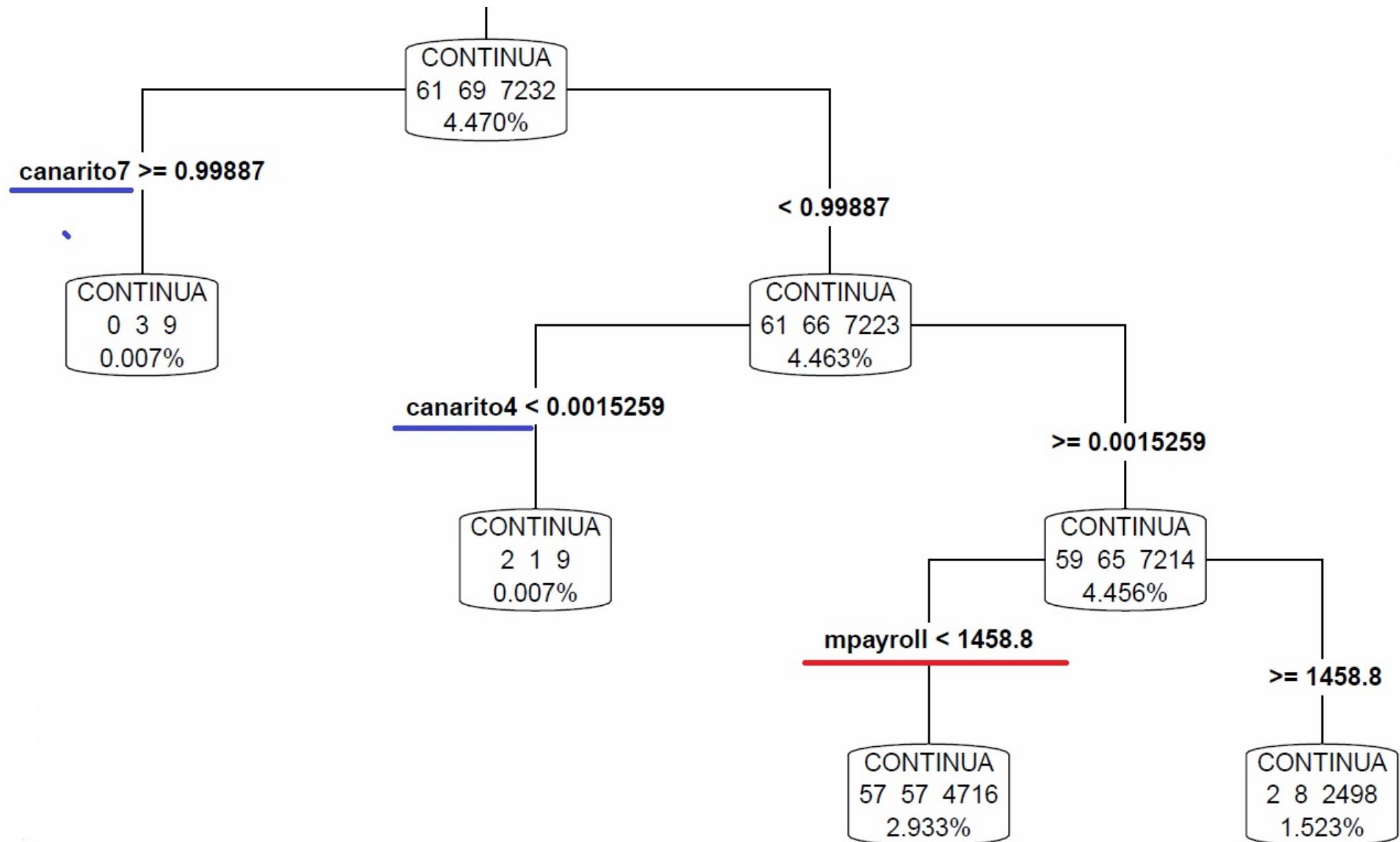
canarito al final



canaritos hojas grandes



wtf ?

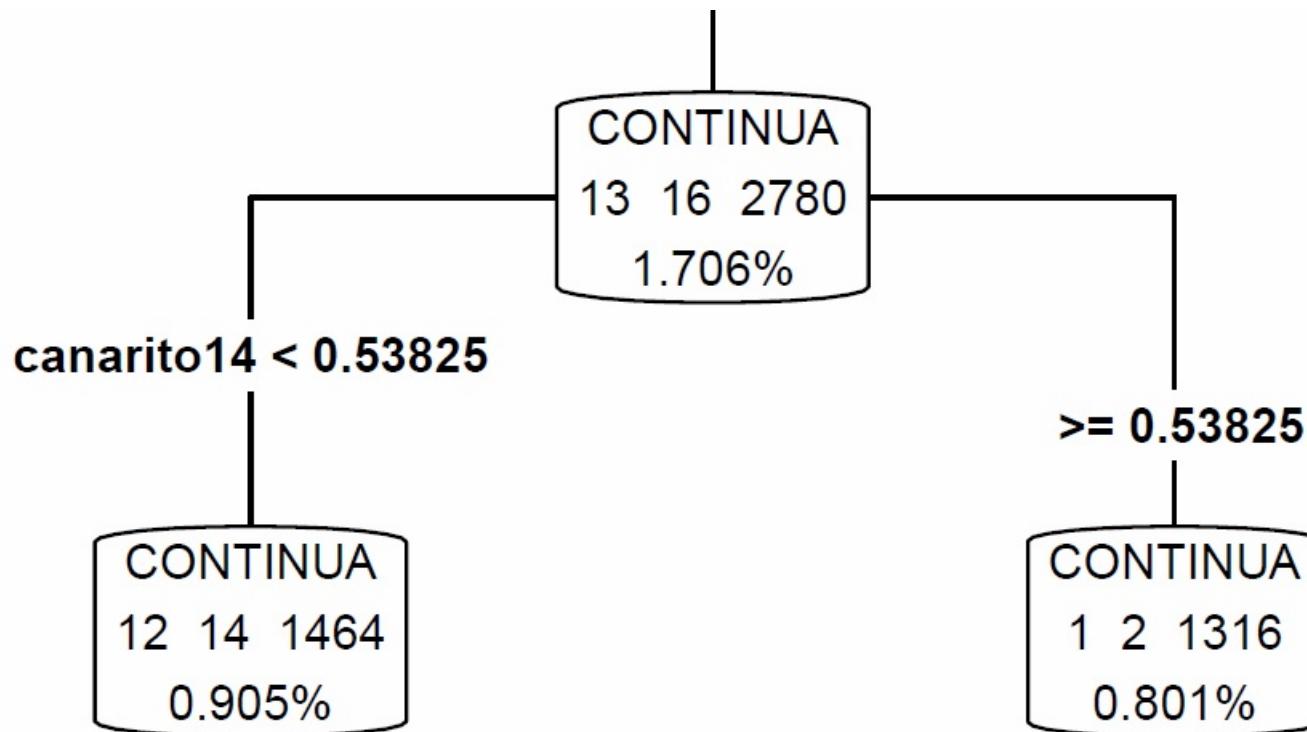


Alumn@ con formación en
la Estadística Clásica
razonablemente desconfiad@,
hace la siguiente pregunta:

¿Qué sucede si exijo hojas muy grandes,
por ejemplo de por lo menos 1000 registros?
¿siguen apareciendo canaritos?

[z482_canaritos_desconfiado.r](#)

`minbucket = 1000`



Observaciones 1

- ¿ Cómo puede ser que teniendo más de 150 variables para cortar elija una variable canarito, que por ser random, JAMÁS debería estar correlacionada con la clase ?
- Cualquier correlación de canarito en los datos de 202107 es **espúrea por diseño**, NO SE VA A CUMPLIR en 202109
- ¿Tiene algún 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 raíz
- 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 `z487_canaritos_pruning.r`
- Agrego nuevas variables canarito al dataset, tantas como atributos tiene el dataset
- Construyo el árbol sobre todo 202107 (con canaritos)
- Dejo crecer el árbol todo lo que puede
- Hago pruning (la poda del arbol) justo en los canaritos
- Calculo la ganancia del modelo pruning en los datos de 202109
- En ningún momento hago training/testing
- Se construye UN SOLO árbol

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

```
#hago el pruning de los canaritos
#haciendo un hackeo a la estructura modelo_original$frame
# -100 es un valor arbitrariamente 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 [202107]

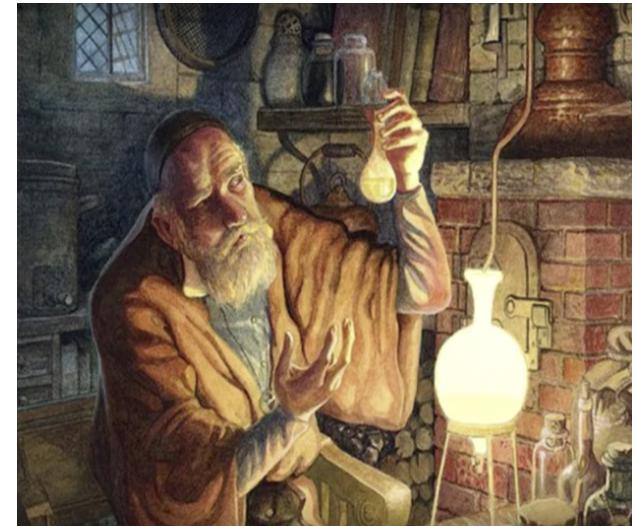
Método	Hiperparametros	Private Leaderboard	árboles construídos	tiempo (minutos)
Bayesian Optimization 5-fold xval $5*(16 + 100) + 1$	cp= -0.82 maxdepth= 6 minsplit= 769 minbucket= 8	38.6 M	581	78.1
Stopping at Canarito's	Ninguno	39.9 M	1	1.8

x 43 speedup

Se utilizó una sola vCPU
en ambas corridas

SIN hiperparámetros
SIN training/testing
ni cross validation

repito el ciclo muchas veces



Ahora ~2 minutos
(antes 78 minutos)

Feature
Engineering
+
ciencias
ocultas

Características *Stopping at Canarito's*

- El método de los canaritos NO mejora la ganancia, da la misma ganancia.
- El método canaritos corre ~43 veces mas rápido que Bayesian Optimization con 5-fold Cross Validation

Características *Stopping at Canarito's*

- No hacen falta los hiperparámetros que controlan el crecimiento del arbol `cp`, `maxdepth`, `minsplit`, `minbucket`
- No se usan `training`, `testing`, ni `montecarlo`, ni `5-fold cross validation`.
- SI hacen falta hiperparámetros para mejorar el arbol

GRANDES Limitaciones

Stopping at Canarito's

- Es un método aproximado, el correcto consiste en hacer randomización de la clase (ver bibliografía)
- 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 (aún) 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 ?