Repensando el Overfitting

¿Pero qué es el overfitting?

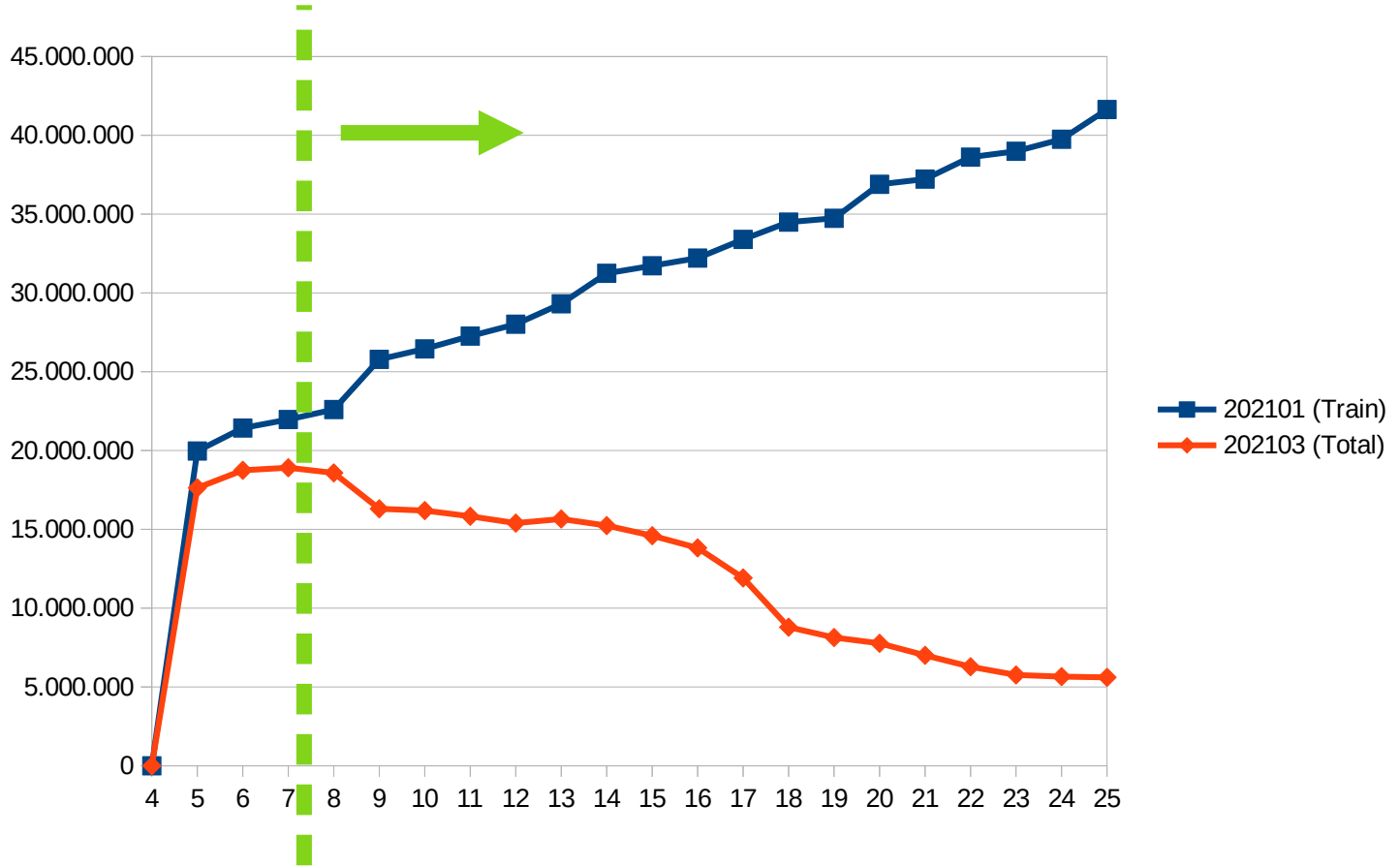
Experimento (Con datos de otros años):

- Entrenamiento en todo 202101
- Aplicamos en 202103 (Leaderboard)

Medimos ganancia en ambos casos y graficamos de acuerdo a un solo parámetro (maxdepht)

Script: z410

¿Pero que es el overfitting?



Pero que NO es el overfitting?

NO ES la diferencia entre las curvas de train/test.

El mismo fenómeno se ve en LGBM o XGBoost

Se ve que existe un punto óptimo de ganancia en test, pero es indetectable en train.

Entonces, ¿Cómo lo evito?

¿Que enfoque debería tener para encontrar parámetros optimos que me den la mayor ganancia en datos no vistos (aka train, aka datos del futuro)?

Solución vista hasta ahora

Entreno en una porción de los datos y mido en otra porción no vista en entrenamiento.

El problema pasa a ser una optimización de una función de hiperparámetros a ganancia.
(Random Search, Grid Search, Optimización Bayesiana)

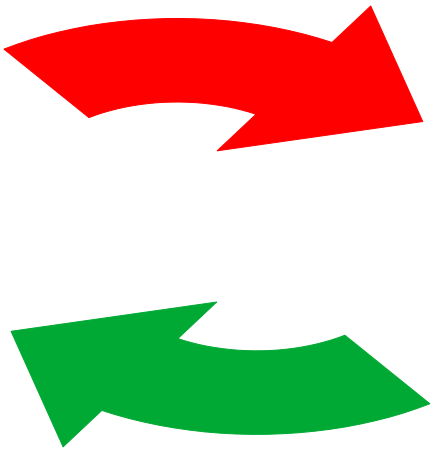
Solución train/test

Distintas implementaciones con la misma idea:

- Solo train/test (inestable, depende del azar)
- Montecarlo (muchas veces lo anterior)
- K-fold CV
- N-repeticiones de K-fold CV
- Leave-One-Out

Entreno muchas veces!

El ciclo de optimización



Optimización de hiperparámetros

(Horas, incluso días)

Data drifting
+
Feature Engineering

(Tiempo de pensar)

El paso más tedioso

¿Cuanto lleva hacer una optimización Grid Search de hiperparámetros?

Rpart, 100 iteraciones, 5-fold CV, 4 hiperparámetros:

- 6 x 9 x 6 x 2 = 684 iteraciones
- En cada iteració se hacen 5 modelos (5 CV)

684 * 5 = 3420 modelos !!!!

Y de yapa se entrena el modelo final para aplicar a los datos nuevos

El paso más tedioso

¿Cuanto lleva hacer una optimización bayesiana de hiperparámetros?

Rpart, 100 iteraciones, 5-fold CV, 4 hiperparámetros:

- BO, 4*4 + 100 = 116 iteraciones
- En cada iteració se hacen 5 modelos (5 CV)

116 * 5 = 580 modelos

Y de yapa se entrena el modelo final para aplicar a los datos nuevos

El paso más tedioso

El resultado final son 4 valores:
<cp = -0.78, minsplit = 122, maxdepth = 8, minbucket = 28>

Y el fin último es armar un modelo con esos parámetros para aplicar a los datos del futuro.

¿Podemos evitar construir 580 modelos extra?

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

Repensando el Overfitting

¿Son los datos?

¿Es el algoritmo?

¿Por qué el árbol crece?

¿Por qué le tengo que decir de antemano hasta donde crecer?

¿Por qué existen parámetros crípticos como el “cp”?

En el fondo somos mineros, volvamos a las fuentes.

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

Podemos usar una idea similar para encontrar donde el árbol debe parar.

Variable “canarito”

Nuestro dataset está “envenenado” y nuestro árbol no se da cuenta.

Agregamos variables “sentinela” que sean marcadores de que algo anda mal.

Variables con distribución uniforme continuas en el intervalo [0.0, 1.0]

Variable “canarito”

Una variable “canarito” por diseño es aleatoria, no esta relacionada con nada de nuestro dataset ni con otro canarito.

Como fue construida así, no puede estar nunca en nuestro modelo predictivo

Experimentando con canaritos

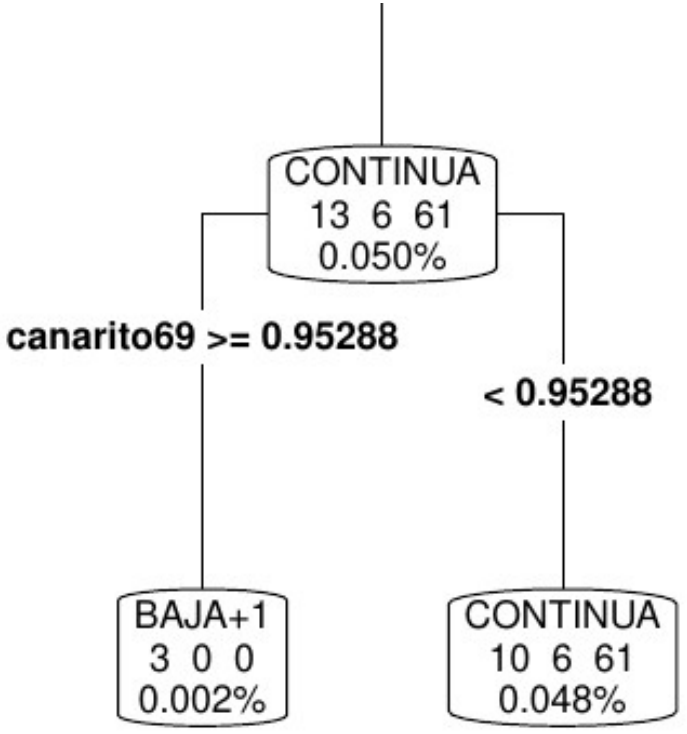
Agregamos 100 variables canarito a nuestro dataset.

Usamos hiperparámetros conocidos.

Analizamos el árbol que nos queda.

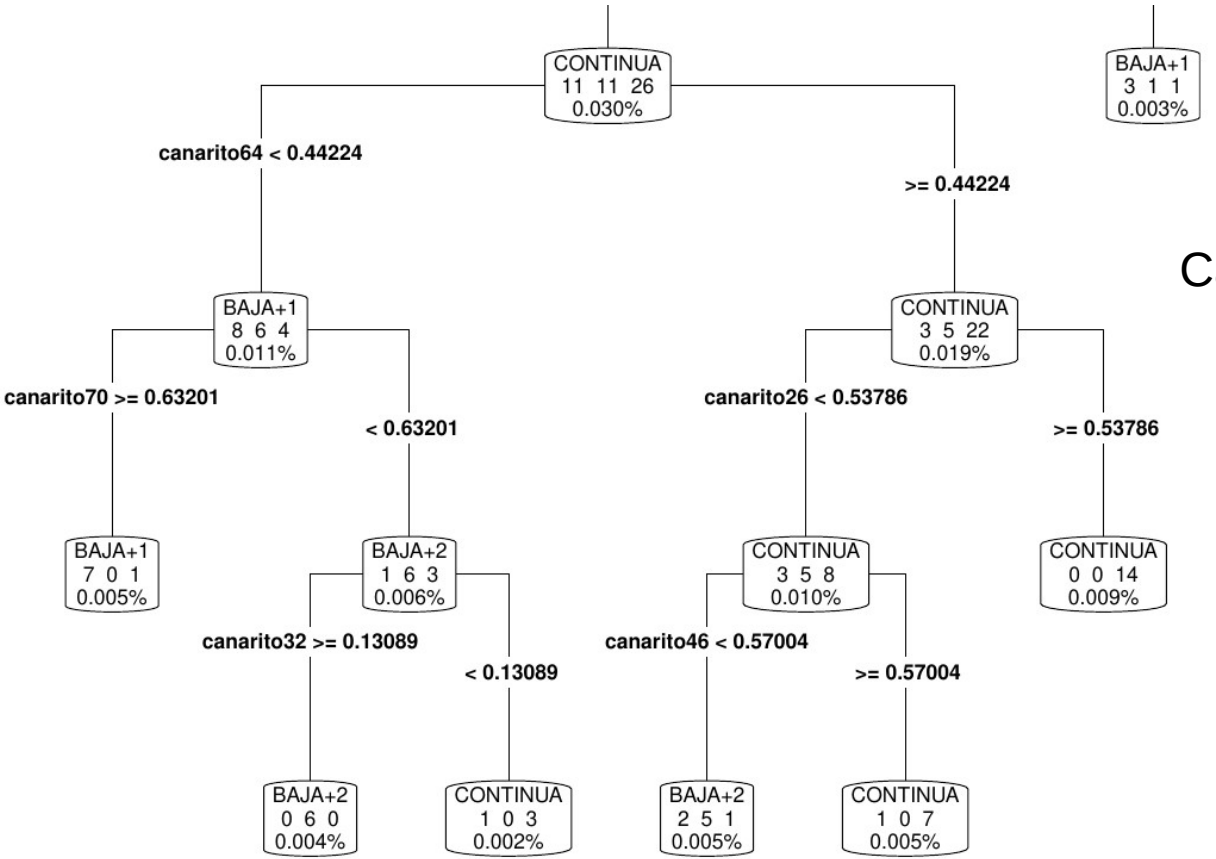
Script: z411_arbol_canaritos_prp.r

Experimentando con canaritos



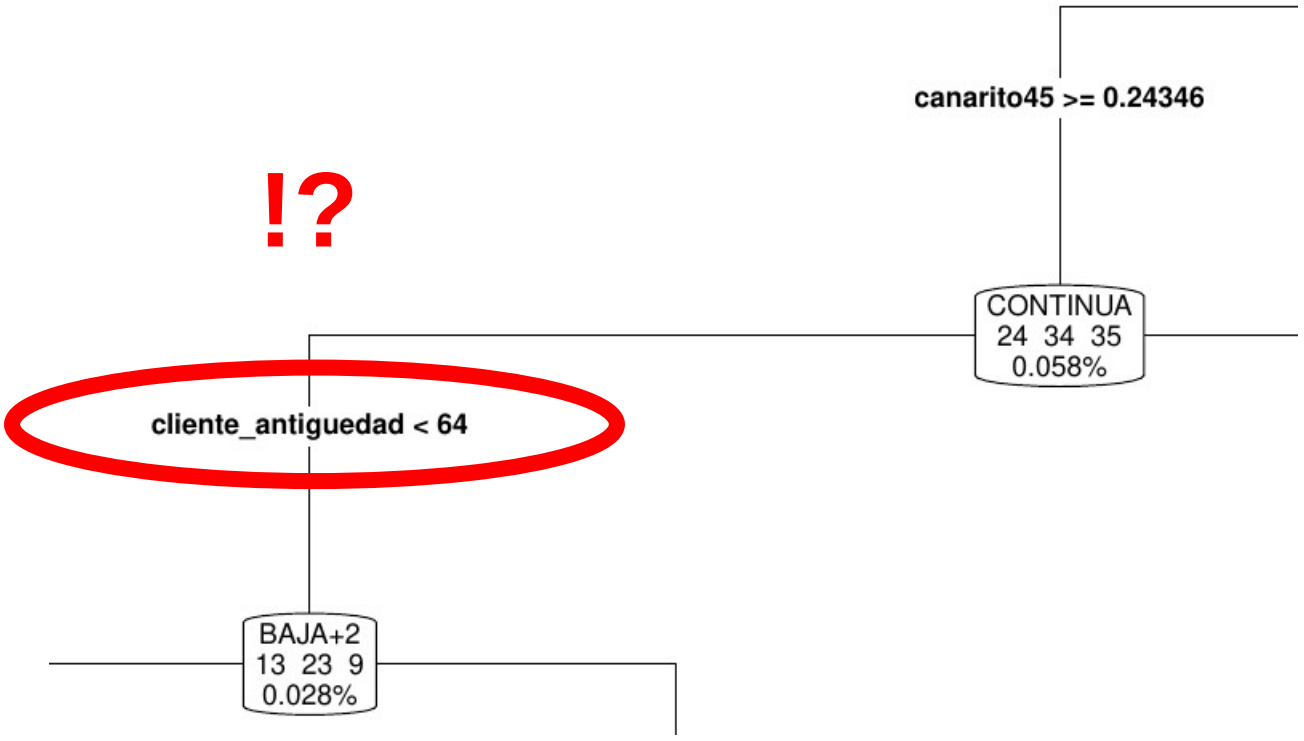
Canarito en nodo terminal

Experimentando con canaritos



Canaritos anidados

Experimentando con canaritos



Experimentando con canaritos

Observaciones:

¿Cómo el árbol no se da cuenta que las variables canarito son cualquier cosa?

Una correlación con variables canarito es espúrea. No se va a cumplir en datos del futuro.

Tiene sentido que nuestro árbol corte por un canarito? ¿Influye la cantidad de canaritos que metamos en el dataset?

Experimentando con canaritos

Observaciones:

- Curiosamente no aparecen cerca de la raíz
- Generalmente no aparecen en nodos grandes
- Aparecen en nodos intermedios o profundos
- Si un canarito aparece en un nodo grande, ¿quiere decir que no hay una buena variable para cortar?
- ¿Y si podo el árbol donde aparece un canarito?

Una loca idea

El modelo va a generalizar mejor si antes de cada decisión me pregunto: ¿Es una correlación real o la variable “tuvo suerte”?

Experimentando con canaritos

- Agregamos 30 canaritos al dataset
- Entrenamos en todo 202101 (canaritos incluidos)
- No limito el crecimiento del arbol
- Podo el arbol donde encuentro un canarito
- Calculo la ganancia sobre kaggle

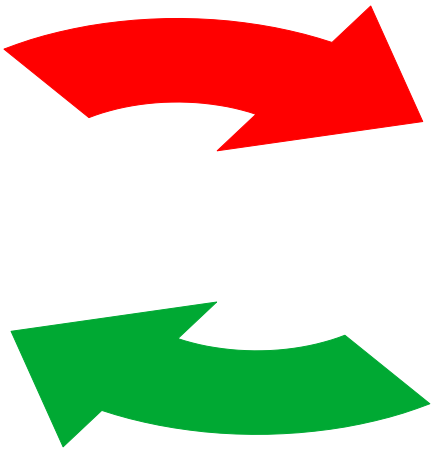
Construyo un único árbol, no hago train/test

Experimentando con canaritos

Método	Hiperparámetros	Score Privado	Árboles entrenados	Tiempo (horas)
BO + 5-fold CV	cp = -0.78, minsplit = 122 maxdepth = 8 minbucket = 28	17.40 M	581	2:22
Poda en canaritos	ninguno	17.97 M	1	0:01

En una sola CPU

El ciclo de optimización



Canaritos

(Antes horas, ahora segundos)

Data drifting
+
Feature Engineering

(Tiempo de pensar)

Experimentando con canaritos

- El método NO mejora la ganancia
- Corre x100 veces más rápido que una BO de 5-fold CV
- No hacen falta los hiperparámetros que controlan el crecimiento del árbol
- SI hacen falta los otros hiperparámetros del algoritmo
- No se usa train/test

Limitaciones del método

- Es un método aproximado, asume un filtro de “ruido blanco”. Es más corecto randomizar la clase (ver el z487).
- Es muy aproximado, se deberían utilizar más canaritos y no cortar si las primetas 5 candidatas son canaritos.
- Tiene problemas con variables de baja cardinalidad
- No resuelve el problema del BIAS en la estimación de probabilidad de las hojas del árbol.

ATENCIÓN: Los hiperparámetros que no controlan el crecimiento del árbol siguen necesitando una optimización!