AdaBoost

La técnica detrás de AdaBoost consiste en entrenar un predictor, un clasificador base (por ejemplo un árbol de decisión), verificar los errores que comete y entrenar luego otro predictor que corrija estos errores (estas instancias mal clasificadas). AdaBoost repite este proceso hasta disminuir el error o encontrar un clasificador perfecto.

Utiliza para verificar el error el mismo conjunto de entrenamiento. Solo que, antes de entrenar el siguiente predictor, pondera las instancias mal clasificadas, aumentando su peso relativo. De esta manera, el siguiente predictor entrenado estará focalizado en corregir los errores del primero.

Contras

El entrenamiento de AdaBoost no puede hacerse en paralelo (como los árboles de Random Forest) y es por lo tanto poco escalable (es mas lento el entrenamiento y poco escalable cuando tenemos grandes volumenes de datos).

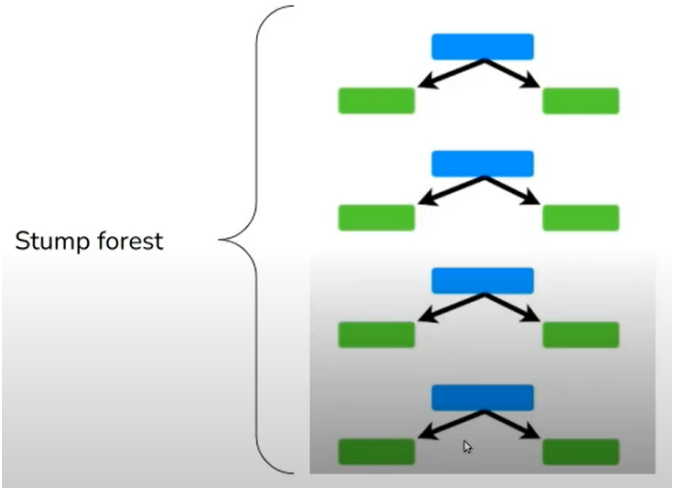
- La implementación más común es con árboles.

A diferencia de Random Forest, donde tenemos N árboles completos (de distinta profundidad pero completos), en AdaBoost tenemos un bosque de tocones (stumps)

Los **tocones** o **stump**, son árboles con un nodo raíz y dos hojas.

N stumps hacen un stump forest

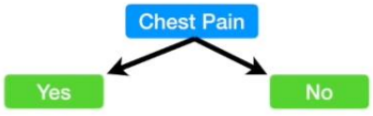
No son buenos clasificando



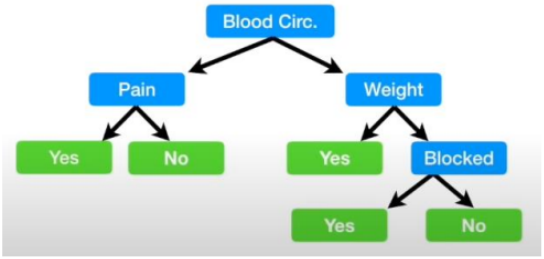
Ejemplo

Queremos saber si alguien puede tener una enfermedad del corazón, usando cuatro variables: Chest Pain, Good Blood Circulation, Blocked Arteries y Weight

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

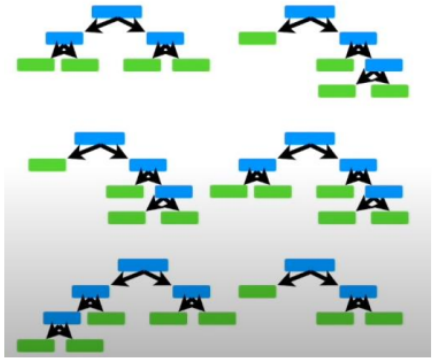


Pero un tocón o *stump* solo utiliza una variable.
Pero es así como funciona AdaBoost



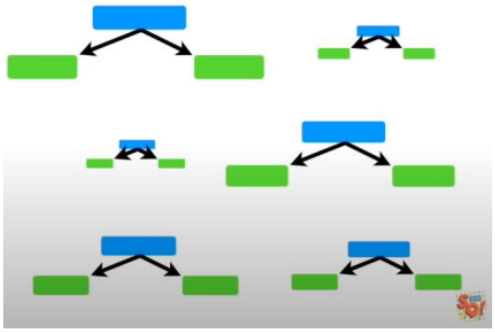
Así se vería un árbol completo. Tradicional.

Random Forest



- Todos los árboles votan y su voto vale 1
- Cada árbol está construido de forma independiente de los otros
  - No importa el orden de creación

AdaBoost



- Los árboles votan de forma ponderada
- La creación de uno depende de los anteriores.
  - Importa el orden de creación

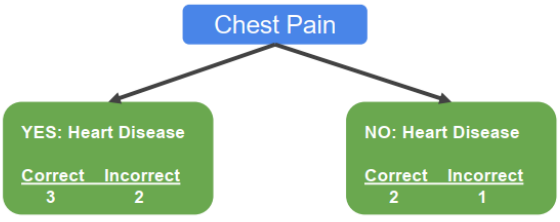
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

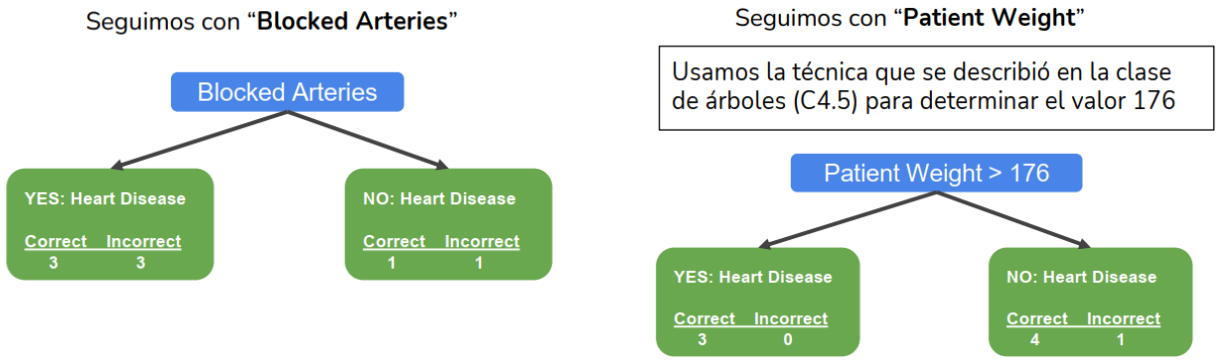
Al principio todos los ejemplos, tienen el mismo peso.

1 / #ejemplos

Hay que encontrar el atributo que mejor clasifica los ejemplos

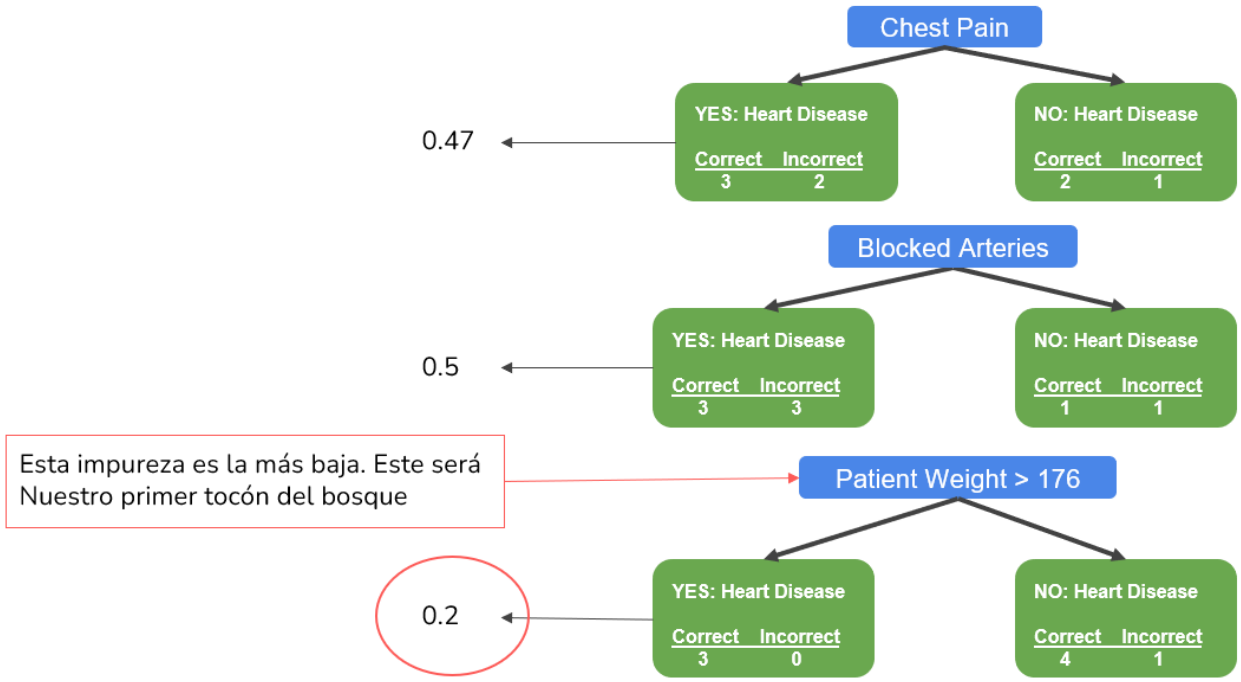
Empezamos con "**Chest Pain**"





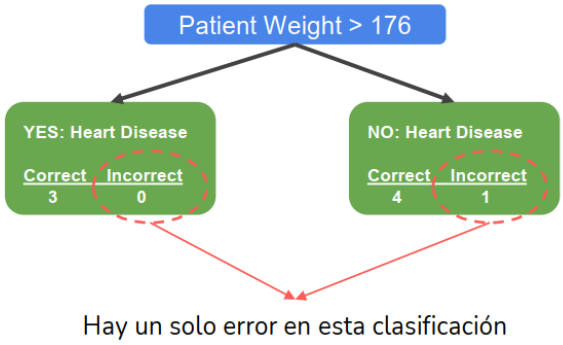
Calculamos el indice de Gini, para los 3 stumps o tocones

0: el arbol es puro y clasifica correctamente



Tenemos que calcular el peso relativo del tocón en el clasificador final.
Esta ponderación es llamada: **Amount of say** y para calcularla necesitamos el Error Total del tocón

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



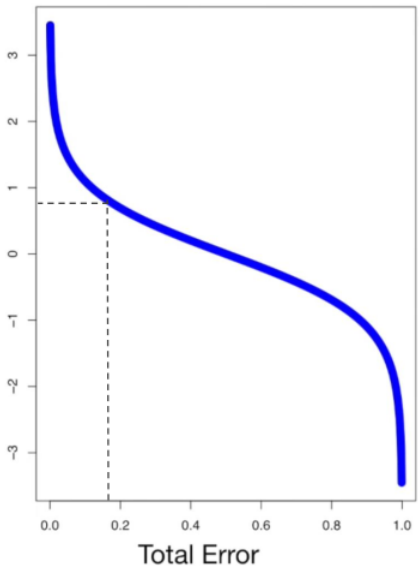
Hay un solo error en esta clasificación

El **Error Total** cometido es 1/8

Error total = $\frac{1}{8} * 1$

Amount of Say = $\frac{1}{2}\ ln(\frac{1-TotalError}{TotalError})$

Amount of Say = $\frac{1}{2}\ ln(\frac{\frac{7}{8}}{\frac{1}{8}}) = \frac{1}{2}\ ln(7) = 0.97$



Ahora tenemos que ponderar las instancias mal clasificadas

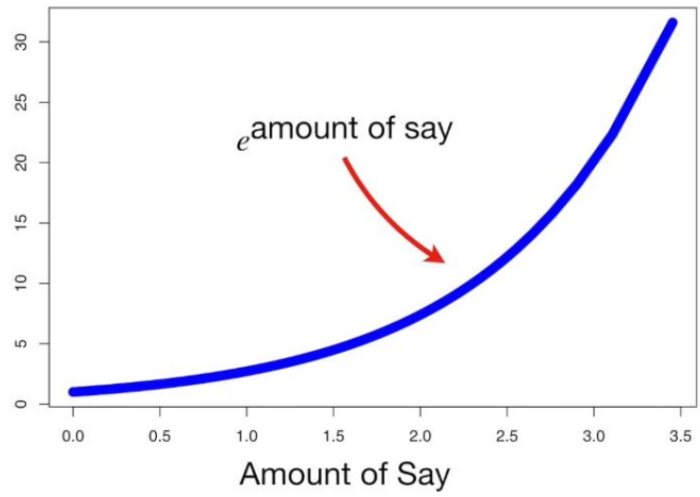
Solo tenemos que ver cómo modificamos los pesos, para que el próximo tocón que entrenemos tenga en cuenta los errores cometidos por el primero

Aumentando el peso relativo de la instancia mal clasificada, y disminuyendo el de los demás
(no puede ser de otro modo, ya que todos los pesos suman 1).

New sample weight = (sample weight) * $e^{\text{amount of say}}$

= (⅛) $e^{\text{amount of say}}$

= (⅛) $e^{0.97}$ = ⅛ * 2.64 = 0.33



Cómo se ve en esta gráfica: $\mathbf{e}^{\text{"amount of say"}}$ crece exponencialmente, pero si "amount of say" es relativamente bajo, crecerá menos que si es relativamente alto.

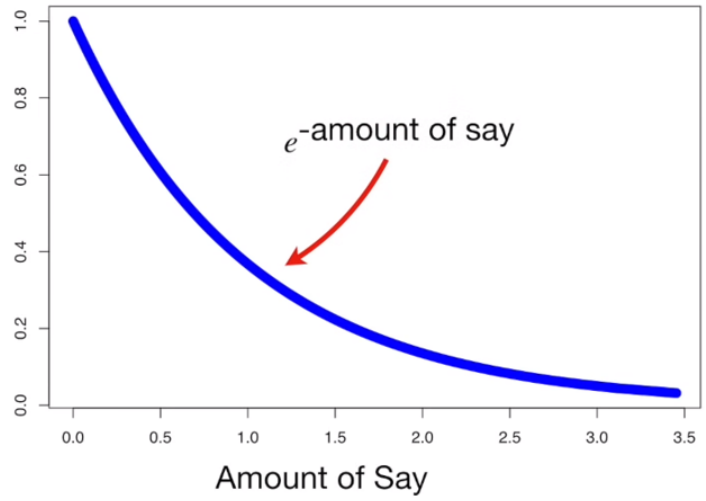
Además al estar multiplicado por el peso anterior (que es un número entre 0 y 1), el nuevo peso será un número intermedio.

Ahora hay que reducir el peso relativo de los ejemplos bien clasificados

New sample weight = (sample weight) * $e^{-\text{amount of say}}$

= (⅛) $e^{-\text{amount of say}}$

= (⅛) $e^{-0.97}$ = ⅛ * 0.38 = 0.05



Cómo se ve en esta gráfica: $\mathbf{e}^{\text{-amount of say}}$ decrece exponencialmente, si ***amount of say*** es relativamente bajo estará cerca de 1 (sin jamás llegar a 1).
Y si es relativamente alto, estará cerca de 0.

En otras palabras, si el ***amount of say*** es muy bajo (el nodo anterior no tiene mucho que decir), los pesos serán multiplicados por 1, o un número cercano a 1 y quedarán casi igual. Pero si el nodo anterior tiene mucho que decir los pesos serán disminuidos proporcionalmente.

Agregamos la nueva columna
con pesos y normalizamos

Dividimos cada peso, por 0.68 (la suma de ellos), para normalizar los valores

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New S. Weight	Norm. Weight
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

Reemplazamos la vieja columna de pesos con la nueva.

Con estos nuevos pesos, creamos el siguiente tocón (*stump*) del bosque

2 posibilidades

Utilizar la **impureza de Gini ponderada** para saber qué tocón clasifica mejor el ejemplo con más peso.

Generar un nuevo *dataset* teniendo en cuenta los pesos de los ejemplos

Generamos un nuevo dataset teniendo en cuenta los pesos.

Creamos un conjunto de datos vacío del mismo tamaño que el anterior

Armamos una distribución de valores, sumando los pesos.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	Dist.
Yes	Yes	205	Yes	0.07	0.07
No	Yes	180	Yes	0.07	0.14
Yes	No	210	Yes	0.07	0.21
Yes	Yes	167	Yes	0.49	0.70
No	Yes	156	No	0.07	0.77
No	Yes	125	No	0.07	0,84
Yes	No	168	No	0.07	0,91
Yes	Yes	172	No	0.07	1

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

Elegimos al azar un número entre 0 y 1. Verificamos en que fila cae.

Por ejemplo:

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	Dist.
Yes	Yes	205	Yes	0.07	0.07
No	Yes	180	Yes	0.07	0.14
Yes	No	210	Yes	0.07	0.21
Yes	Yes	167	Yes	0.49	0.70
No	Yes	156	No	0.07	0.77
No	Yes	125	No	0.07	0,84
Yes	No	168	No	0.07	0,91
Yes	Yes	172	No	0.07	1

0.05 (está entre 0 y 0.07)

0.35 (entre 0.21 y 0.70)

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	Dist.
Yes	Yes	205	Yes	0.07	0.07
No	Yes	80	Yes	0.07	0.14
Yes	No	2	Yes	0.07	0.21
Yes	Yes	167		0.49	0.70
No	Yes			0.07	0.77
No	Y	125	No	0.07	0,84
Yes	No	168	No	0.07	0,91
Yes	Yes	172	No	0.07	1

Este ejemplo se agregó 4 veces.
Lo que refleja su peso mayor.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	156	No
Yes	Yes	167	Yes
No	Yes	125	No
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	172	No
Yes	Yes	205	Yes
Yes	Yes	167	Yes

Reemplazamos el viejo dataset con el nuevo.

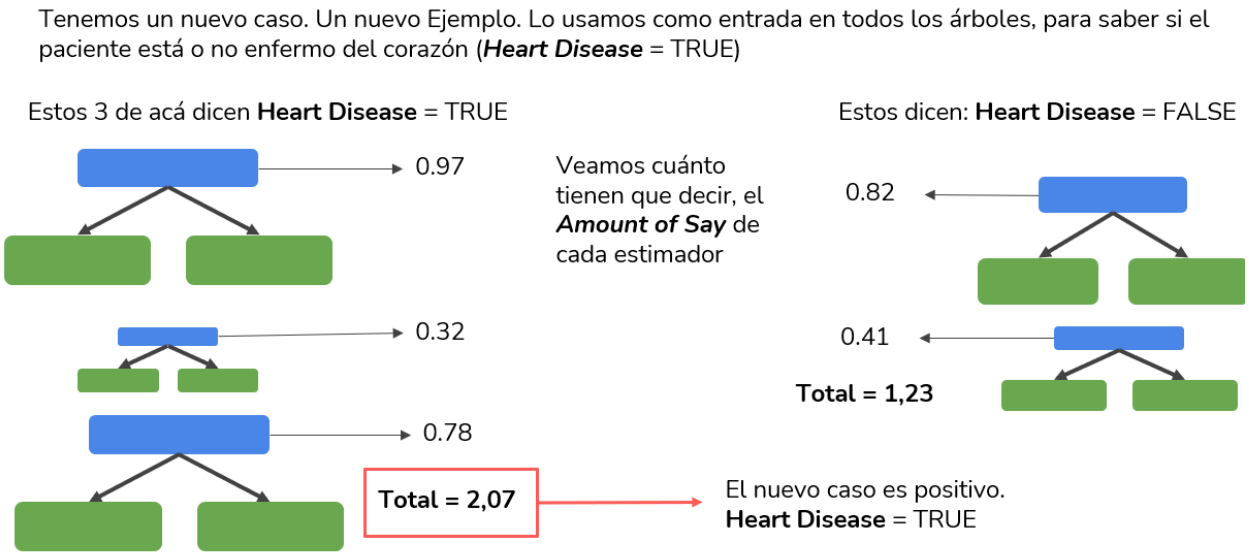
Agregamos nuevamente el peso, será igual para todos los ejemplos.

Sin embargo al haber un ejemplo repetido 4 veces, su verdadero peso es 1/2

Volvemos a calcular los tocones para cada atributo

De esta forma los errores cometidos por un árbol o tocón son utilizados para construir un segundo árbol. Y los errores del segundo árbol servirán a su vez para construir un tercero. Y asi seguimos…

¿Cómo clasifica AdaBoost una vez construido el bosque de tocones?



AdaBoost: Resumen

1. AdaBoost combina un monton de **weak learners** (estimadores pobres) para hacer clasificaciones. Estos weak learners, son generalmente stumps (tocones).

2. Algunos tocones tienen más peso que otros en la votación final, tienen más que decir (amount of say)

3. Cada uno de estos tocones está construido teniendo en cuenta los errores del tocón anterior.

    a. Lo podemos hacer usando una función de impureza de Gini ponderada, para cada ejemplo

    b. O simplemente regenerando los datos (como vimos en el ejemplo)