

# Ensembles de Clasificadores

## Leo Breiman - author of Random Forest



*«Random Forest is an example of a tool that is useful in doing analyses of scientific data. But the cleverest algorithms are no substitute for human intelligence and knowledge of the data in the problem. Take the output of random forests not as absolute truth, but as smart computer generated guesses that may be helpful in leading to a deeper understanding of the problem.»*

(27.01.1928 - 07.07.2005)





# Agenda

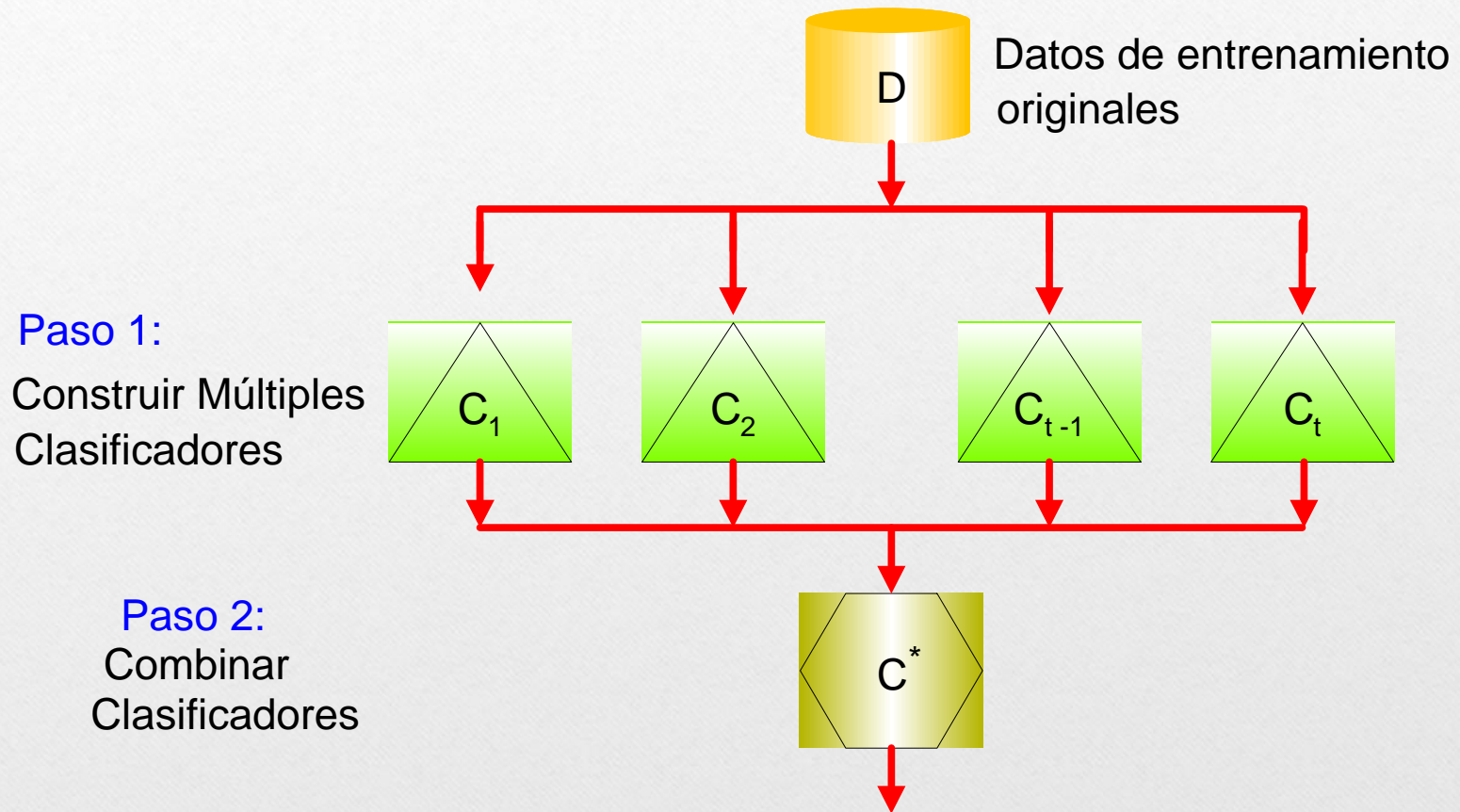
- Métodos para construir ensambladores independientes
  - Bagging and Random Forest: regresión y clasificación
- Métodos para construcción coordinada de ensambladores
  - Boosting: regresión y clasificación

# Ensamblas de Clasificadores

- **La idea básica** es aprender un conjunto de clasificadores (expertos) y permitirles votar.
- **Ventaja:** mejora en la precisión predictiva.
- **Desventaja:** es difícil entender un conjunto de clasificadores.
- <https://www.youtube.com/watch?v=BUA1bxJgCN8>



# Majority Vote



Stacking: <https://www.youtube.com/watch?v=Un9zObFjBH0>

# Porqué Majority Voting funciona?

Suponga que hay un grupo de 25 consultores, a los que se les va a pedir una predicción: A ó B (una de las 2 es correcta, la otra es errónea).

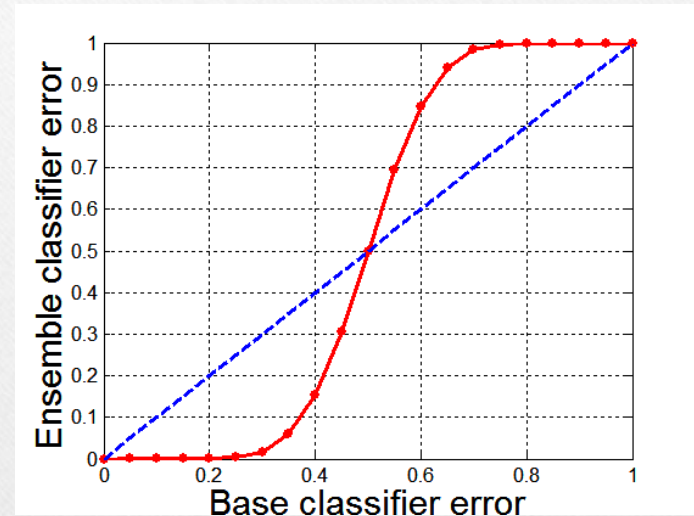
- Cada consultor da su predicción la cual puede ser correcta o errónea. Al final se hará una votación para dar la predicción final. Se tomará la predicción con mayor votación.
- Cada consultor tiene una tasa de error,  $\varepsilon = 0.35$ , es decir, da una predicción errónea con esa probabilidad.
- Asuma que los errores cometidos por cada consultor, no están correlacionados.
- Cuál es la probabilidad de que el grupo – en conjunto – haga una predicción errónea?



# Porqué Majority Voting funciona?

- Suponga que hay 25 clasificadores básicos
  - Cada clasificador tiene una tasa de error,  $\varepsilon = 0.35$
  - Asuma que los errores cometidos, no están correlacionados
  - La probabilidad de que un ensamble de clasificadores haga una predicción errónea:

$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1 - \varepsilon)^{25-i} = 0.06$$



# Bagging

- Emplea la forma más sencilla de combinar predicciones que pertenecen al mismo tipo.
- La combinación se puede realizar mediante votación o promediando
- Cada modelo recibe igual peso
- Versión “Idealizada” de Bagging:
  - Muestree varios conjuntos de entrenamiento de tamaño  $n$  (en lugar de tener solo un conjunto de entrenamiento de tamaño  $n$ )
  - Construya un clasificador para cada conjunto de entrenamiento
  - Combine las predicciones del clasificador
- Esto mejora el rendimiento en casi todos los casos si el esquema de aprendizaje es inestable (es decir, árboles de decisión)
- <https://www.youtube.com/watch?v=2Mg8QD0F1dQ>



# Bagging

## Generación de Clasificadores

Sea  $n$  ser el tamaño del conjunto de entrenamiento.

Para cada una de las  $t$  iteraciones:

- Muestree  $n$  instancias con reemplazo, del conjunto de entrenamiento.
- Aplique el algoritmo de aprendizaje a la muestra.
- Almacene el clasificador resultante.

## Clasificación

Para cada uno de los  $t$  clasificadores:

- Pronostique la clase de instancia usando el clasificador.

Retorne la clase que se pronosticó con mayor frecuencia

# Random Forests

## Generación de Clasificadores

Sea  $n$  el tamaño del conjunto de entrenamiento.

Para cada una de las  $t$  iteraciones:

- (1) Muestree  $n$  instancias con reemplazo, del conjunto de entrenamiento.
- (2) Entrene un árbol de decisión s.t. la variable para cualquier Nuevo nodo es la mejor variable entre  $m$  variables aleatoriamente seleccionadas.
- (3) Almacene el árbol de decisión resultante.

## Clasificación

Para cada uno de los  $t$  árboles de decisión:

Pronostique la clase de la instancia.

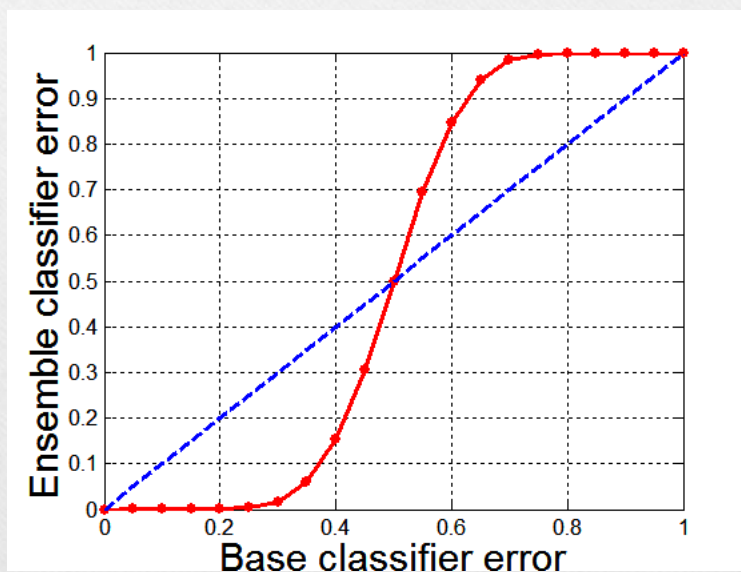
Retorne la clase que se pronosticó con mayor frecuencia.

[https://www.youtube.com/watch?v=J4Wdy0Wc\\_xQ](https://www.youtube.com/watch?v=J4Wdy0Wc_xQ)

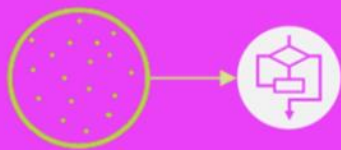


# Bagging y Random Forests

- Bagging usualmente mejora los árboles de decisión.
- Random Forests usualmente supera a bagging debido al hecho de que los errores de los árboles de decisión en el bosque, están menos correlacionados.

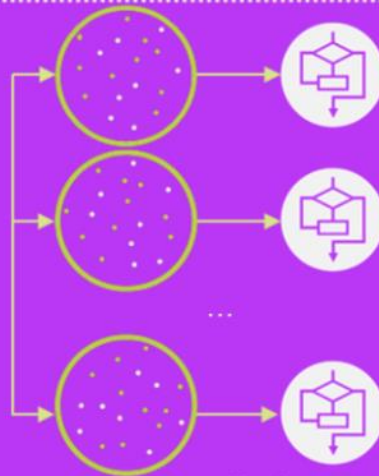


single



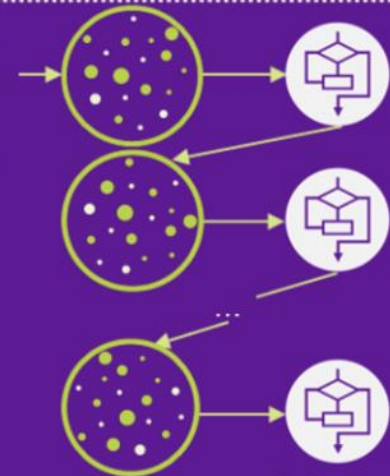
1 iteration

bagging



parallel

boosting



sequential



# Boosting

- También usa votación/promedios, pero los modelos son ponderados de acuerdo con su desempeño.
- Procedimiento iterativo: los nuevos modelos son influenciados por el desempeño de los modelos previamente desarrollados
  - El Nuevo modelo es motivado a convertirse en experto para las instancias incorrectamente clasificadas por los modelos anteriores.
  - Justificación intuitiva: los modelos deberían ser expertos que se complementan entre ellos.
- Existen diversas variantes para este algoritmo
- <https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/44629#250927>

# Boosting – para regresión. Ejemplo

PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

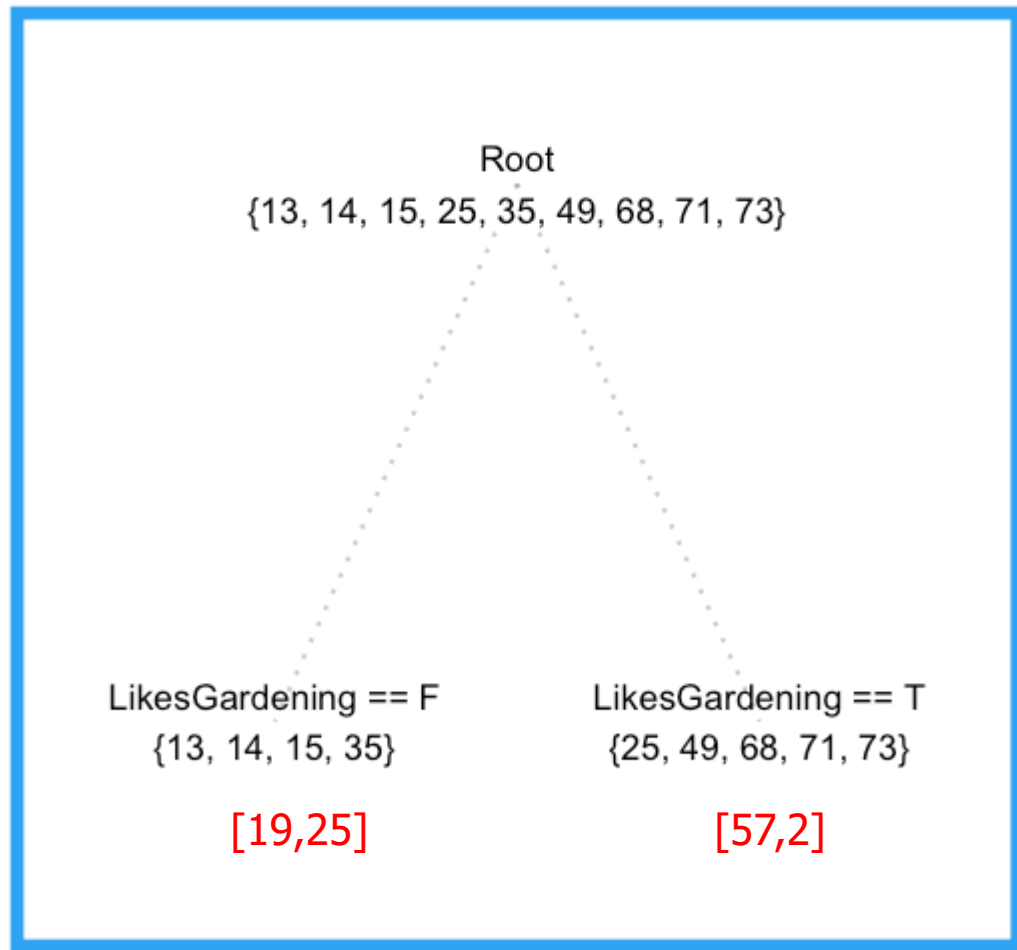
Feature	FALSE	TRUE
LikesGardening	{13, 14, 15, 35}	{25, 49, 68, 71, 73}
PlaysVideoGames	{49, 71, 73}	{13, 14, 15, 25, 35, 68}
LikesHats	{14, 15, 49, 71}	{13, 25, 35, 68, 73}



# Boosting – para regresión. Ejemplo

Requeriremos  
que los nodos  
terminales  
tengan por lo  
menos 3  
muestras

Tree 1



# Boosting – para regresión. Ejemplo

Medimos los errores de entrenamiento con el Árbol 1:

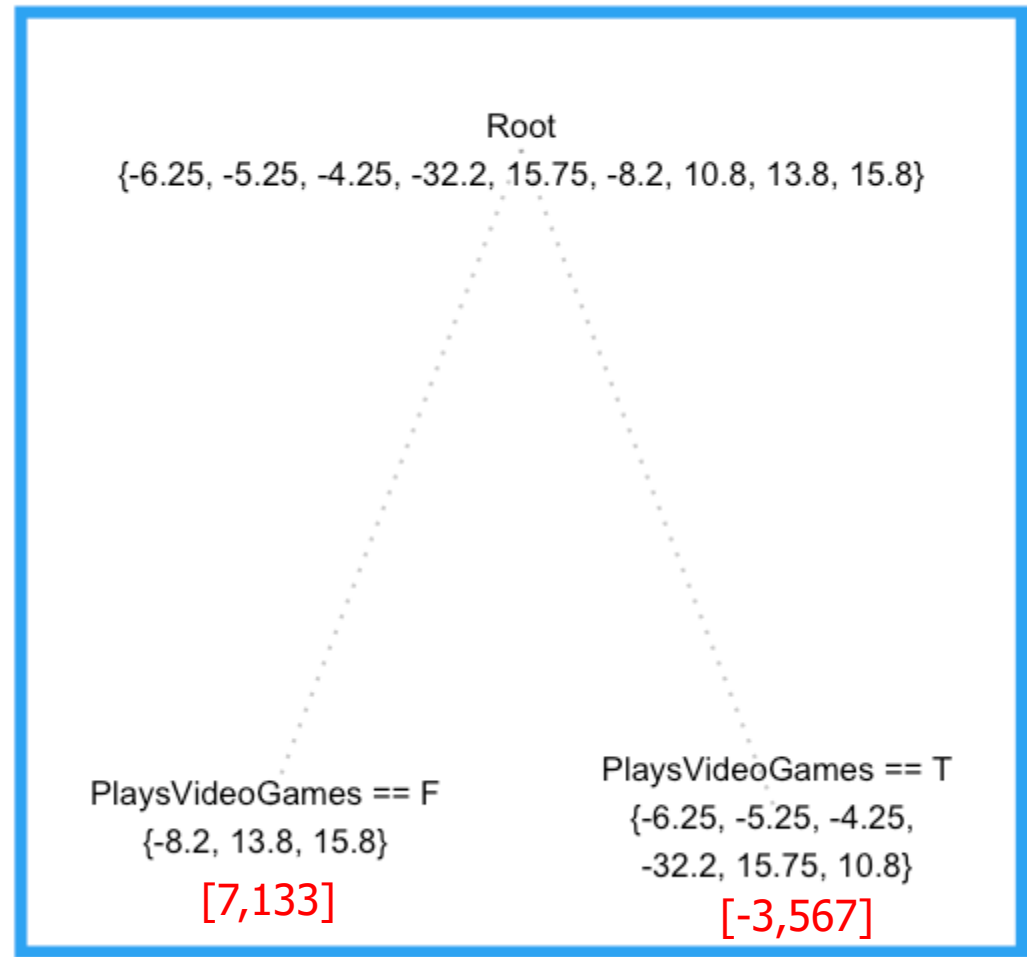
PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8



# Boosting – para regresión. Ejemplo

Ajustamos un  
segundo árbol  
con los  
residuos del  
primer árbol

Tree2



# Boosting – para regresión. Ejemplo

Mejoramos las predicciones del primer árbol, agregando Las predicciones de “corrección del error”

PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.567	15.68	2.683
2	14	19.25	-5.25	-3.567	15.68	1.683
3	15	19.25	-4.25	-3.567	15.68	0.6833
4	25	57.2	-32.2	-3.567	53.63	28.63
5	35	19.25	15.75	-3.567	15.68	-19.32
6	49	57.2	-8.2	7.133	64.33	15.33
7	68	57.2	10.8	-3.567	53.63	-14.37
8	71	57.2	13.8	7.133	64.33	-6.667
9	73	57.2	15.8	7.133	64.33	-8.667
Tree1 SSE			Combined SSE			
1994			1765			

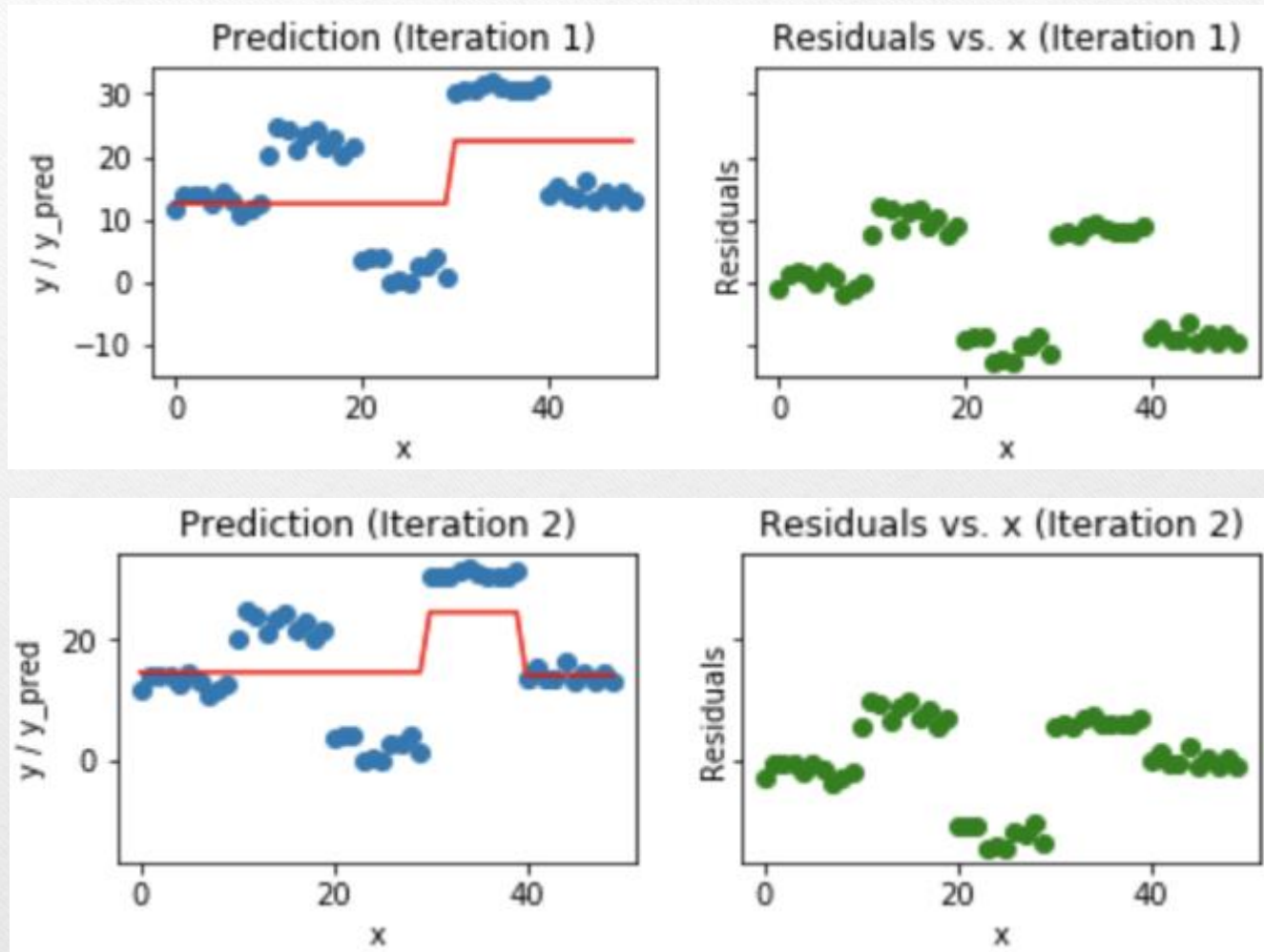


# Boosting – para regresión. Ejemplo

x: entrada, y: salida

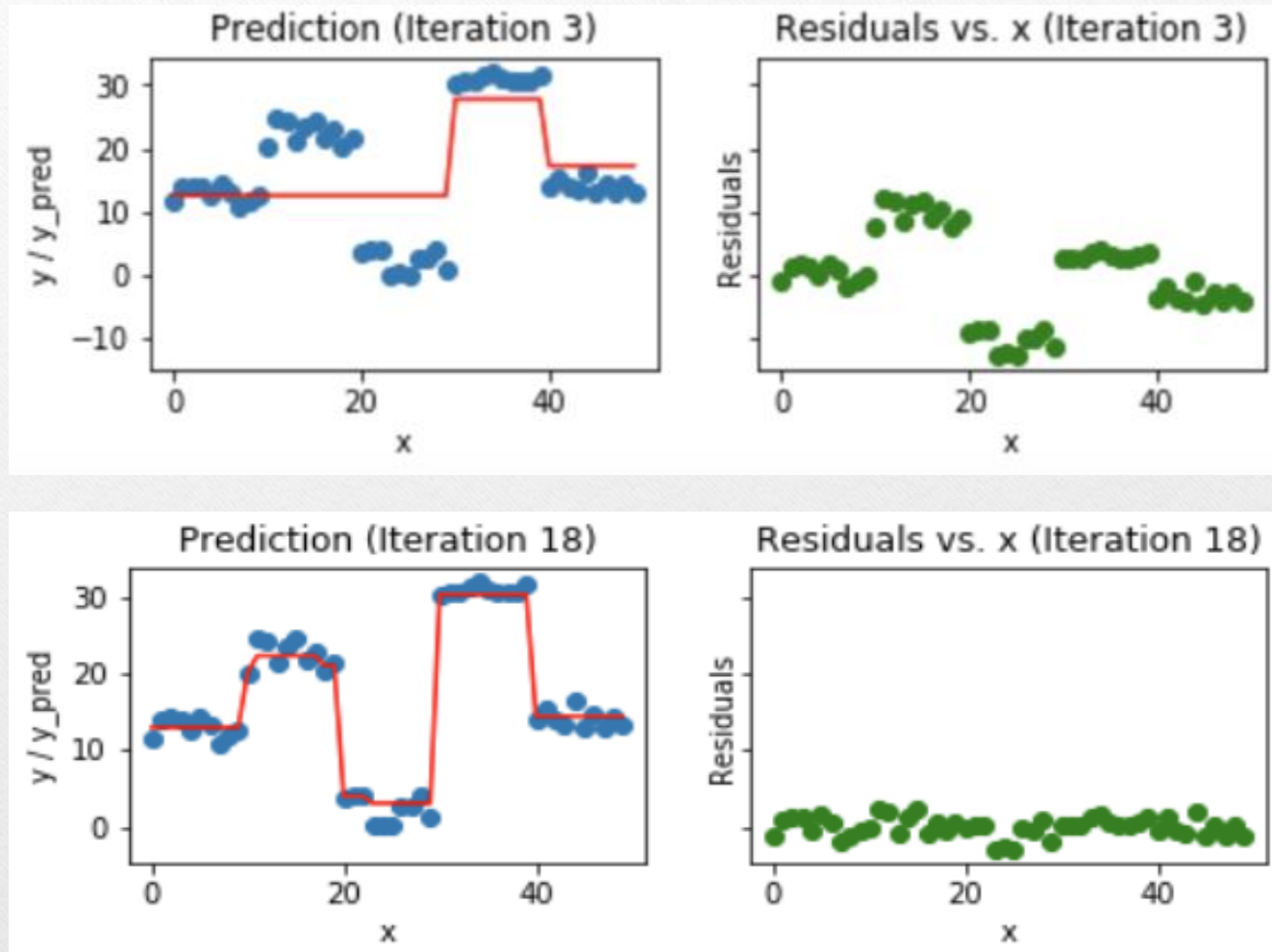


# Boosting – para regresión. Ejemplo

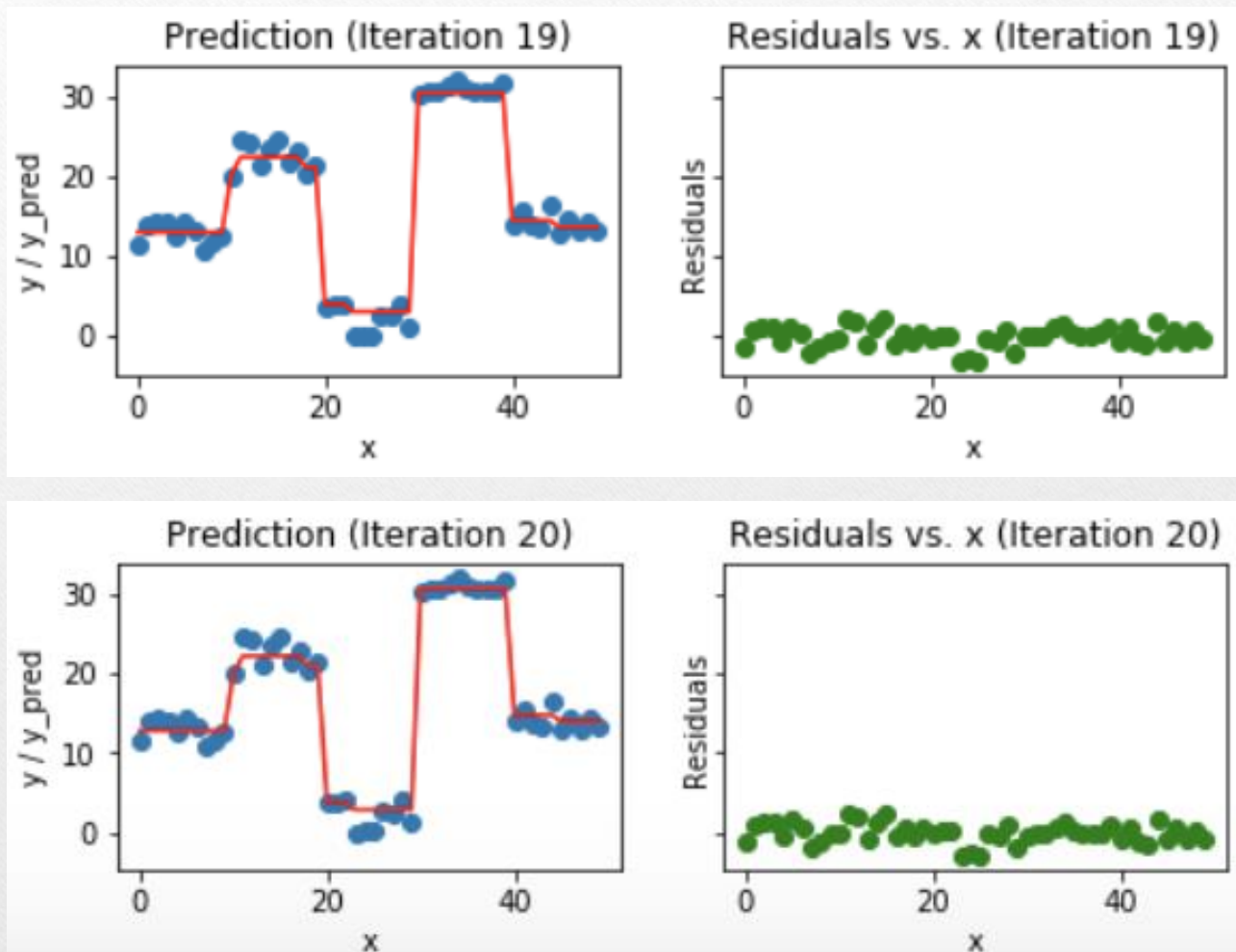




# Boosting – para regresión. Ejemplo



# Boosting – para regresión. Ejemplo



<https://www.youtube.com/watch?v=3CC4N4z3GJc>



# Boosting – parámetros de sintonización

- La cantidad de árboles  $B$ . A diferencia de los RF y Bagging, Boosting puede sobreajustar si  $B$  es demasiado grande, aunque este sobreajuste tiende a ocurrir lentamente si es que lo hace. Usamos validación cruzada para seleccionar  $B$ .
- El parámetro de contracción  $\lambda$ , es un número positivo pequeño. Controla la velocidad a la que Boosting aprende. Los valores típicos son 0.01 o 0.001, y la elección correcta puede depender del problema. Un  $\lambda$  muy pequeño puede requerir usar un valor muy grande de  $B$  para lograr un buen rendimiento.
- El número  $d$  de splits en cada árbol, el cual controla la complejidad del ensamble “boosted”. Usualmente  $d = 1$  (una sola división) funciona bien.

# Algoritmo de Boosting para Regresión

1. Fije  $\hat{f}(x) = 0$ ,  $r_i = y_i$  para todo  $i$  en el dataset de entrenamiento
2. Para  $b = 1, 2, \dots, B$ , repetir:
  - a) Ajuste un árbol  $\hat{f}$  con  $d$  splits ( $d + 1$  nodos terminales) a los datos de entrenamiento  $(X, r)$
  - b) Actualice los residuos

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x)$$

3. Generar el modelo de boosting:

$$\hat{f}(x) = \hat{f}^1(x) + \sum_{b=2}^B \lambda \hat{f}^b(x)$$



# Algoritmo de Boosting para Clasificación

## Durante entrenamiento

- ❑ Se asigna un peso a cada registro del conjunto de entrenamiento
- ❑ Cada iteración aprende un modelo que busca minimizar la suma de los pesos de los ejemplos mal clasificados
- ❑ Los errores se usan para actualizar los pesos:
  - Se aumenta el peso de los registros mal clasificados
  - Se reduce el peso de los bien clasificados

## Durante pruebas

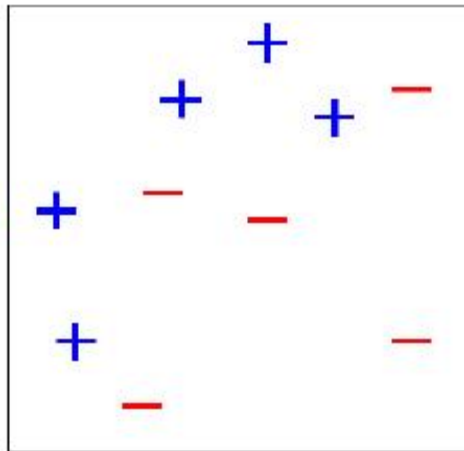
- ❑ Da más relevancia a los modelos que tienen mejor comportamiento

Dos de los métodos de Boosting para clasificación son: XGBOOST y ADABOOST ([fhernanb.github.io/libro\\_mod\\_pred/adaboost.html](https://fhernanb.github.io/libro_mod_pred/adaboost.html))

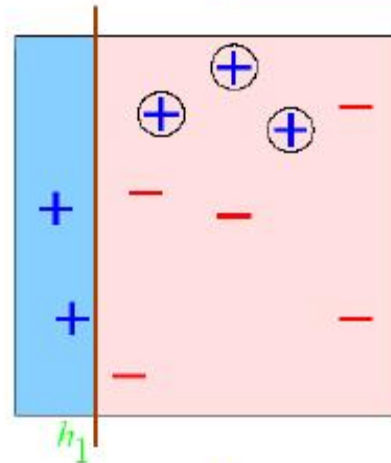
<https://www.youtube.com/watch?v=LsK-xG1cLYA&t=807s>

# Ejemplo

$D_1$



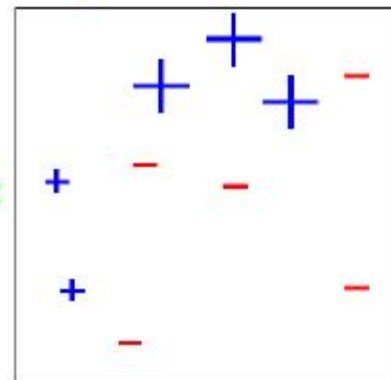
Round 1



$$\epsilon_1 = 0.30$$

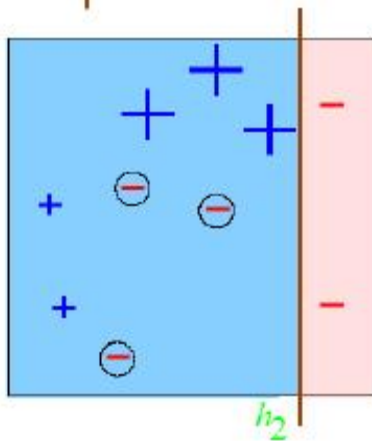
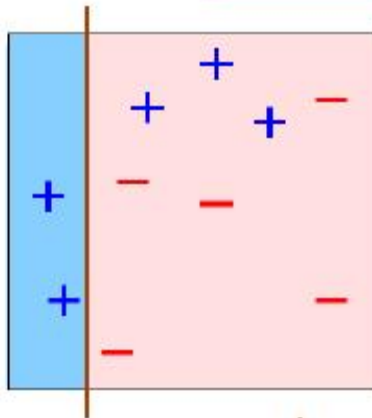
$$\alpha_1 = 0.42$$

$D_2$



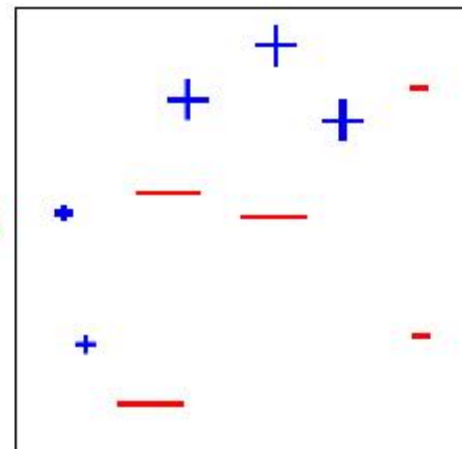
# Ejemplo

## Round 2



$$\begin{aligned}\epsilon_2 &= 0.21 \\ \alpha_2 &= 0.65\end{aligned}$$

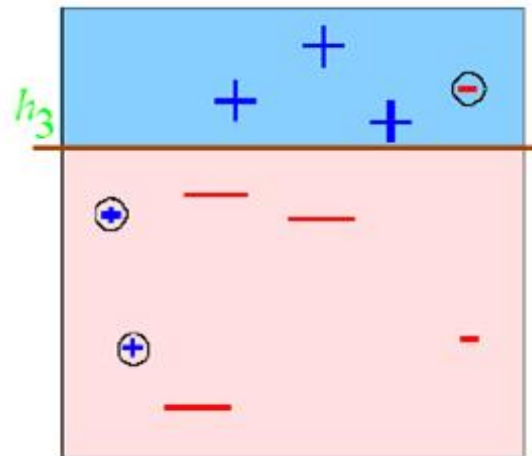
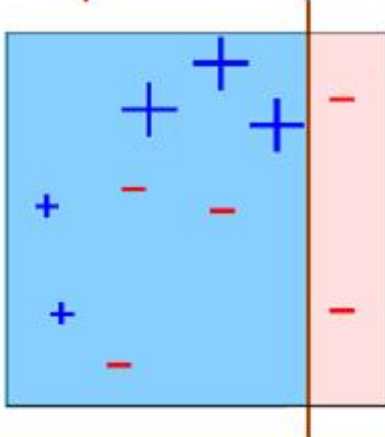
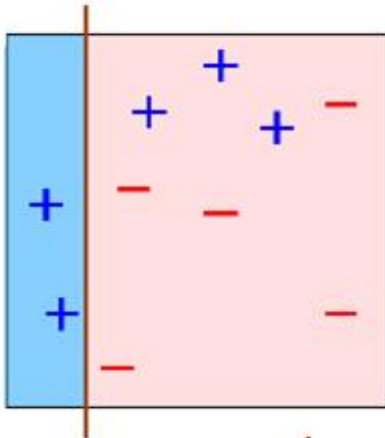
$D_3$





# Ejemplo

## Round 3



$$\epsilon_3 = 0,13$$

$$\alpha_3 = 0,96$$

# Ejemplo

## Final Hypothesis

$$H_{\text{final}} = \text{sign} \left( 0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{pink} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{pink} \\ \hline \end{array} + 0.96 \begin{array}{|c|c|} \hline \text{blue} & \text{pink} \\ \hline \end{array} \right)$$
  
$$= \begin{array}{|c|c|c|} \hline \text{blue} & \text{blue} & \text{pink} \\ \hline + & + & - \\ + & + & - \\ \hline + & - & - \\ + & - & - \\ \hline \end{array}$$

# AdaBoost.M1

1. Inicie con pesos  $w_b(i) = 1/n, i = 1, 2, \dots, n$
2. Repita para  $b = 1, 2, \dots, B$ 
  - a) Ajuste el clasificador  $C_b(\mathbf{x}_i) = \{1, 2, \dots, k\}$  usando los pesos  $w_b(i)$  sobre  $\mathbf{T}_b$
  - b) Calcule el error de clasificación  $e_b = \sum_{i=1}^n w_b(i) * I(C_b(\mathbf{x}_i) \neq y_i)$   
y  $\alpha_b = \frac{1}{2} \ln \left( \frac{1-e_b}{e_b} \right)$
  - c) Actualice los pesos  $w_{b+1}(i) = w_b(i) * \mathbf{EXP}(\alpha_b * I(C_b(\mathbf{x}_i) \neq y_i))$ ,  
 $w_{b+1}(i) = w_b(i) * \mathbf{EXP}(-\alpha_b * I(C_b(\mathbf{x}_i) = y_i))$   
y normalícelos.
3. El clasificador final:

$$c_f(\mathbf{x}_i) = \operatorname{argmax}_{j \in Y} \sum_{b=1}^B \alpha_b * I(C_b(\mathbf{x}_i) = j)$$



# ADABOOST – Ejemplo de aplicación

Tomado de: **StatQuest with Josh Starmer**,

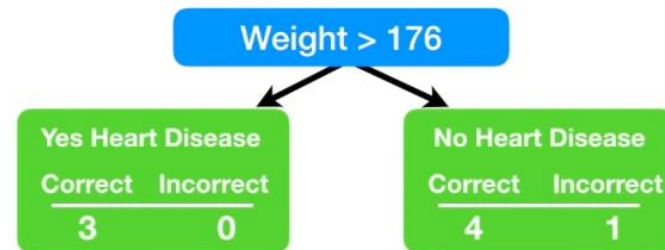
<https://www.youtube.com/watch?v=LsK-xG1cLYA&t=807s>

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



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

Este paciente, quien pesa menos de 176, tiene HEART DISEASE, pero el árbol dice que NO



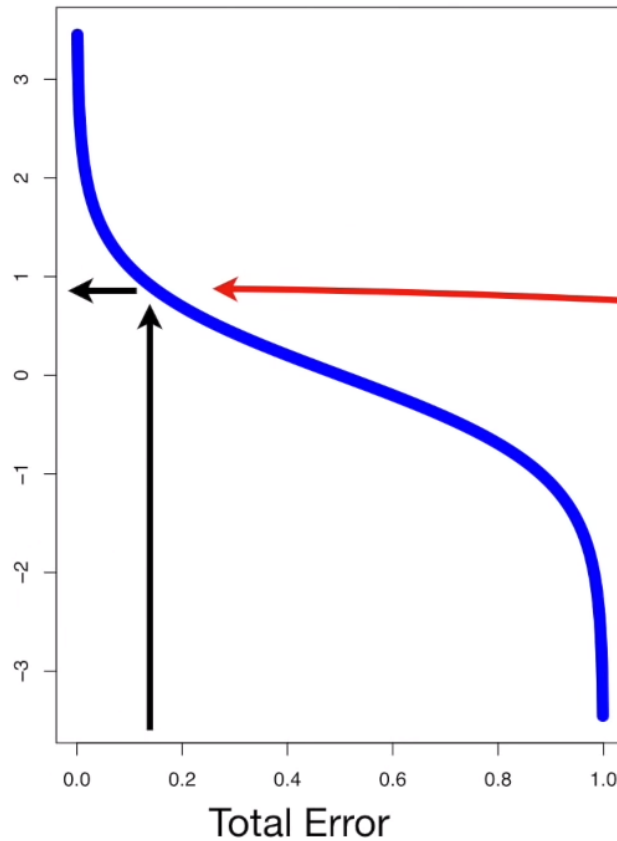


Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Usamos el **ERROR TOTAL** para determinar el **PESO** que este árbol tendrá en la clasificación final, con la siguiente fórmula:

$$\text{PESO DEL ÁRBOL} = \frac{1}{2} \log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$





Y el PESO que este árbol tendrá en la clasificación final será de 0.97:

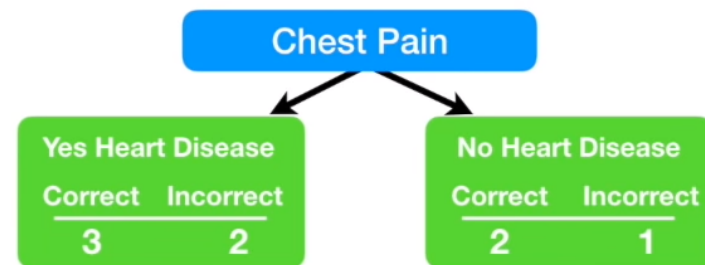
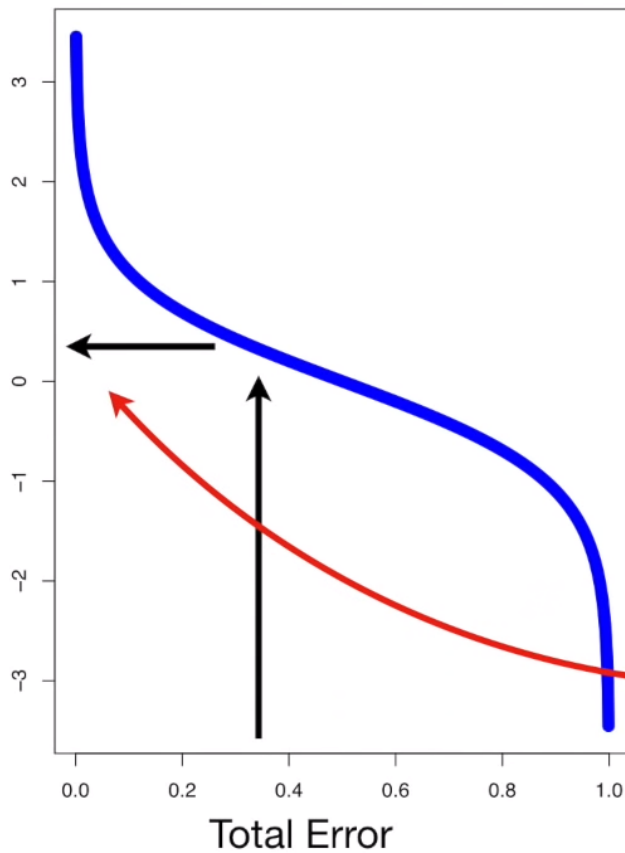
$$\text{PESO DEL ÁRBOL} = \frac{1}{2} \log(7) = 0.97$$



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



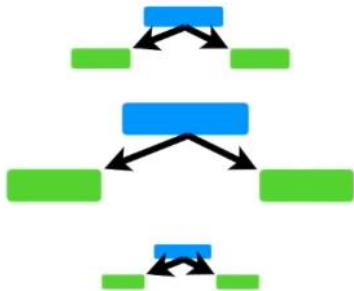




$$\text{Total Error} = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8}$$

Esperamos que el PESO  
esté entre 0 y 0.5. El  
resultado es: 0,2554

Ya sabemos cómo los pesos para las instancias incorrectamente clasificadas, son usados para determinar el PESO que tiene cada árbol



Ahora, necesitamos aprender cómo modificar los pesos para que el próximo árbol tenga en cuenta los errores que el actual árbol cometió



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

Enfatizaremos en la necesidad de que el próximo árbol lo clasifique correctamente incrementando el peso de la instancia

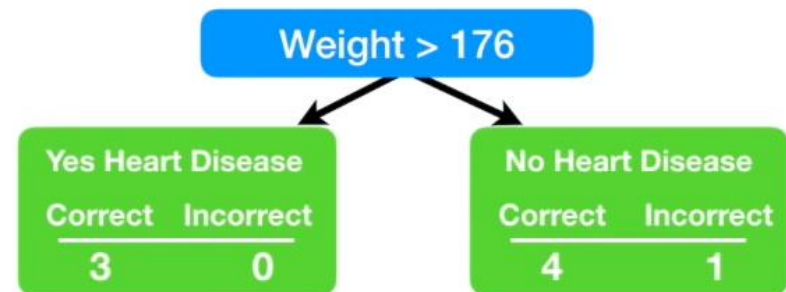




Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



.. Y disminuir el peso de las demás instancias



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

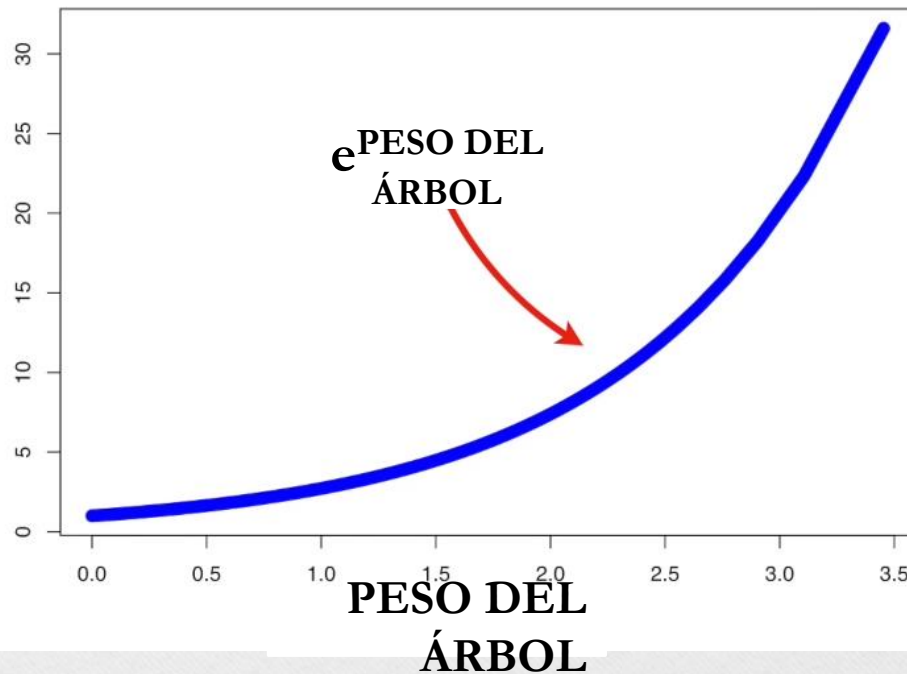
$$\text{Nuevo Peso} = \text{Peso actual} * e^{\text{peso del árbol}}$$



Esta es la fórmula que usaremos para incrementar el peso para la instancia que fue incorrectamente clasificada

$$\text{Nuevo Peso} = \text{Peso actual} * e^{\text{PESO DEL ÁRBOL}}$$

$$= \frac{1}{8} e^{\text{PESO DEL ÁRBOL}}$$



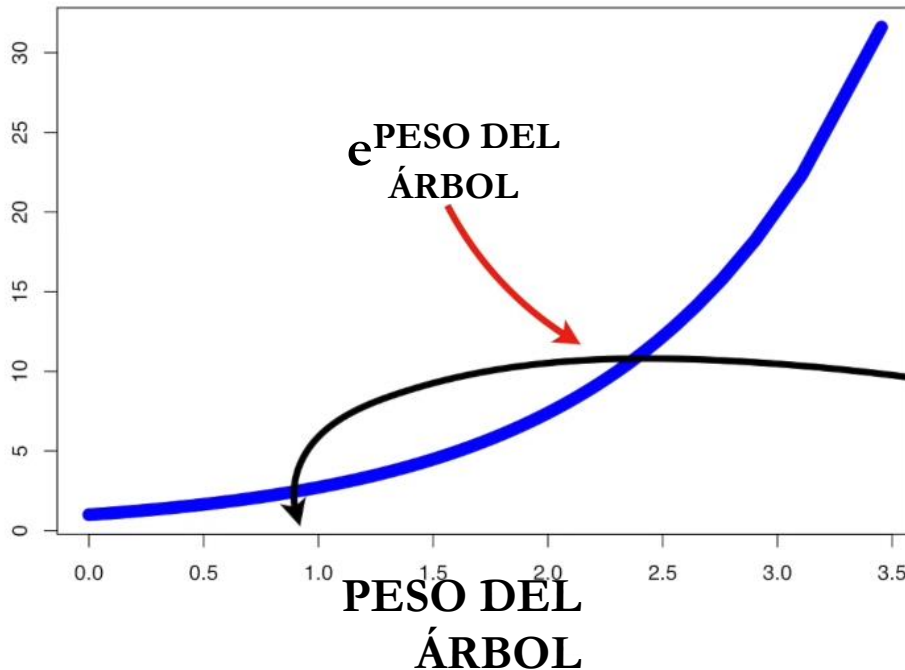


$$\text{Nuevo Peso} = \text{Peso actual} * e^{\text{PESO DEL ÁRBOL}}$$

$$= \frac{1}{8} e^{\text{PESO DEL ÁRBOL}}$$

$$= \frac{1}{8} e^{0.97}$$

$$= 0.33$$



En este ejemplo, el  
PESO DEL ÁRBOL  
fue 0.97

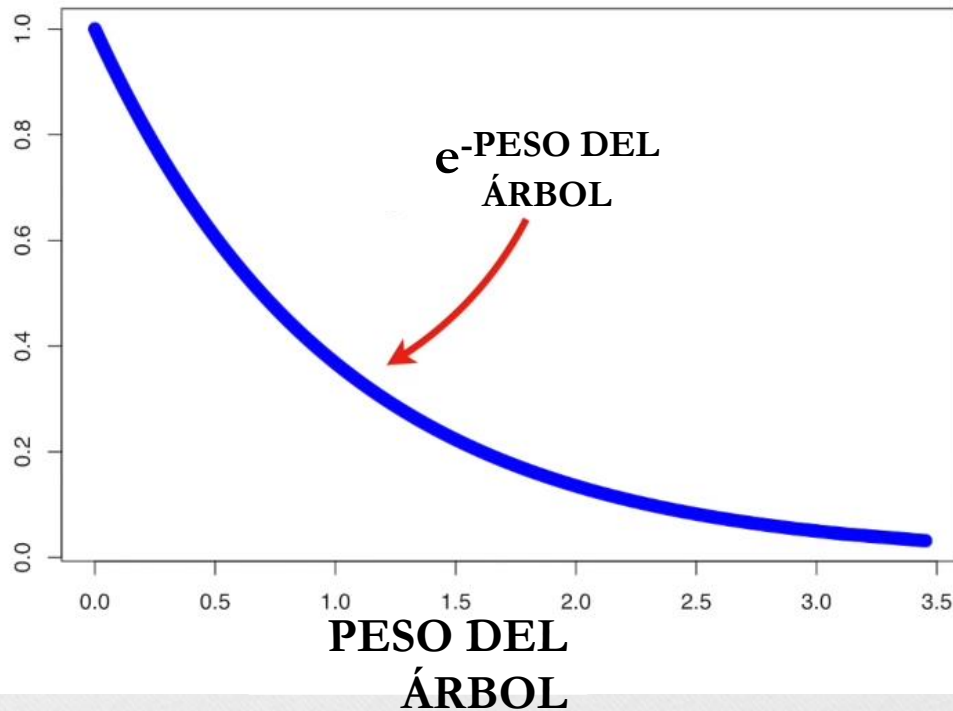
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

$$\text{Nuevo Peso} = \text{Peso actual} * e^{-\text{PESO DEL ÁRBOL}}$$



Esta es la fórmula que usaremos para disminuir los pesos de las instancias

$$\text{Nuevo Peso} = \text{Peso actual} * e^{-\text{PESO DEL ÁRBOL}}$$



$$\begin{aligned} &= \frac{1}{8} e^{-\text{PESO DEL ÁRBOL}} \\ &= \frac{1}{8} e^{-0.97} = \frac{1}{8} \times 0.38 = 0.05 \end{aligned}$$

El nuevo peso de la instancia es 0.05,  
el cual es menor que el anterior  
( $1/8 = 0.125$ )



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New 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

Para desarrollar el nuevo árbol podríamos usar un **INDICE GINI PONDERADO**, o generar una nueva colección de instancias que contengan copias duplicadas de las instancias con los mayores pesos, utilizando un método de Montecarlo

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

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

Si el número está entre 0.14 y 0.21  
 ( $0.14 + 0.07 = 0.21$ ), pondríamos  
 esta instancia dentro del nuevo  
 dataset de entrenamiento

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07

Continuamos seleccionando números aleatorios y agregando instancias al nuevo dataset hasta que el número de instancias sea igual al original

Yes				
Yes				
No				
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

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

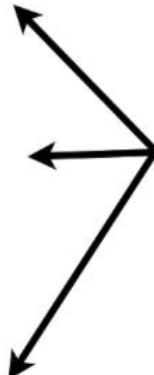


Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125		
Yes	No	168		
Yes	Yes	172		

Por ejemplo, esta instancia fue agregada al nuevo dataset, 4 veces, reflejando así su peso significativamente grande

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

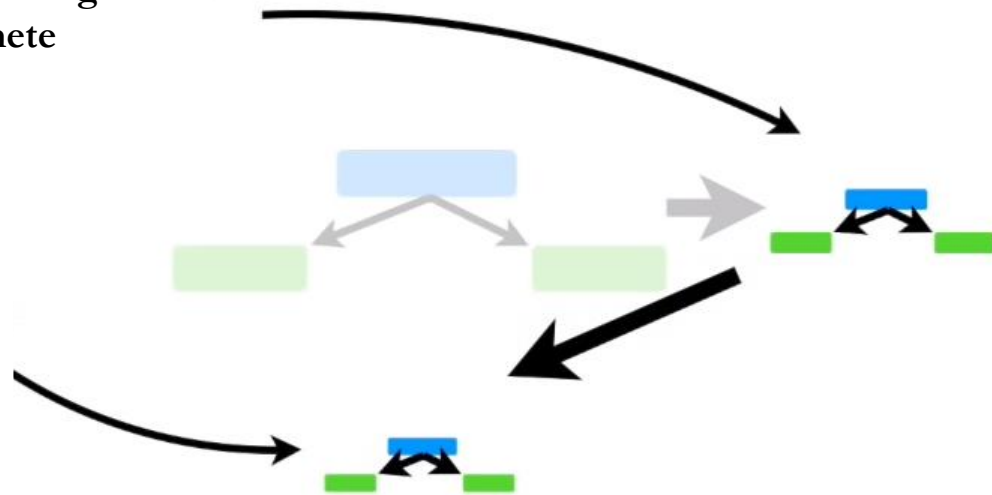
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8



Puesto que todas estas instancias son las mismas, serán tratadas como un bloque, creando una alta penalidad por haber sido mal clasificadas

Y los errores que el segundo  
árbol comete

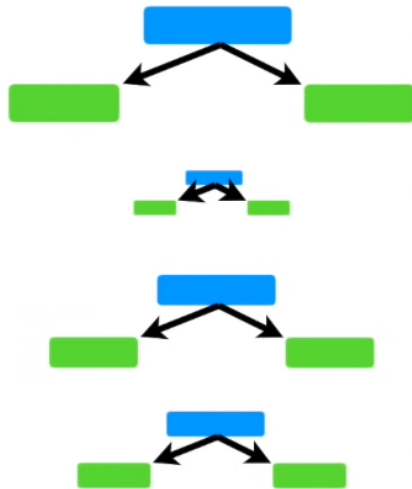
Influencian cómo el tercer  
árbol es desarrollado





En definitiva, el paciente es clasificado como que tiene HEART DISEASE debido a que corresponde a la suma más grande

TIENE HEART DISEASE



Total = 2.7

PESO DEL ÁRBOL → 0.97

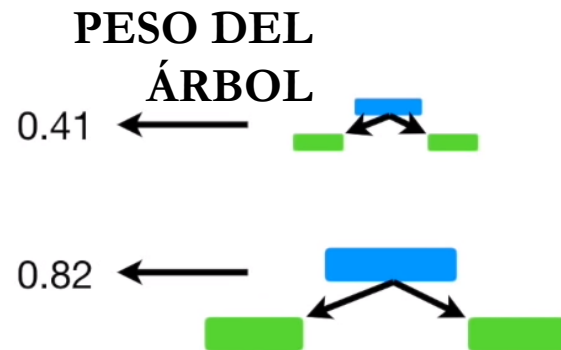
→ 0.32

→ 0.78

→ 0.63

Total = 1.23

NO TIENE HEART DISEASE



0.41

0.82

# TALLER 1

## Ensamblas para Regresión

### Boston Housing





# TALLER 2

## Ensamblas para Clasificación

### Adult Census Income

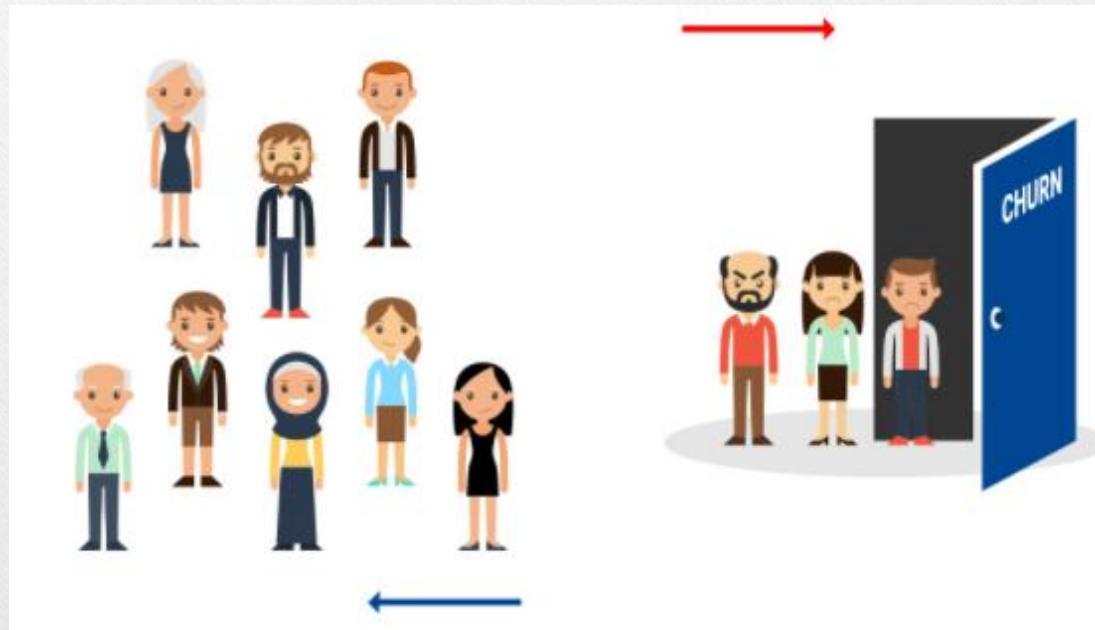




# TALLER 3

## Ensamblajes para Clasificación

### Caso de negocio: Retención de clientes en STC



## Test error of bagged model

- Each bagged tree makes use of approximately 63.2% of the observations. The remaining 36.8% of the observations are called the **out-of-bag** (OOB) observations.
- We predict the response for the  $i$ th observation using **each of the  $\approx B/3$  trees** in which that observation was OOB.
- We then average these predicted responses (regression) or take a majority vote (classification).
- An OOB prediction can be computed for each of the  $n$  observations, from which the overall OOB error can be computed → **test error of bagged model**.