



# IIC 2433 Minería de Datos

<https://github.com/marcelomendoza/IIC2433>

# Ensembles

# Ensembles

En ensembles asumimos que al usar varios modelos podemos obtener mejores resultados que usando un modelo.

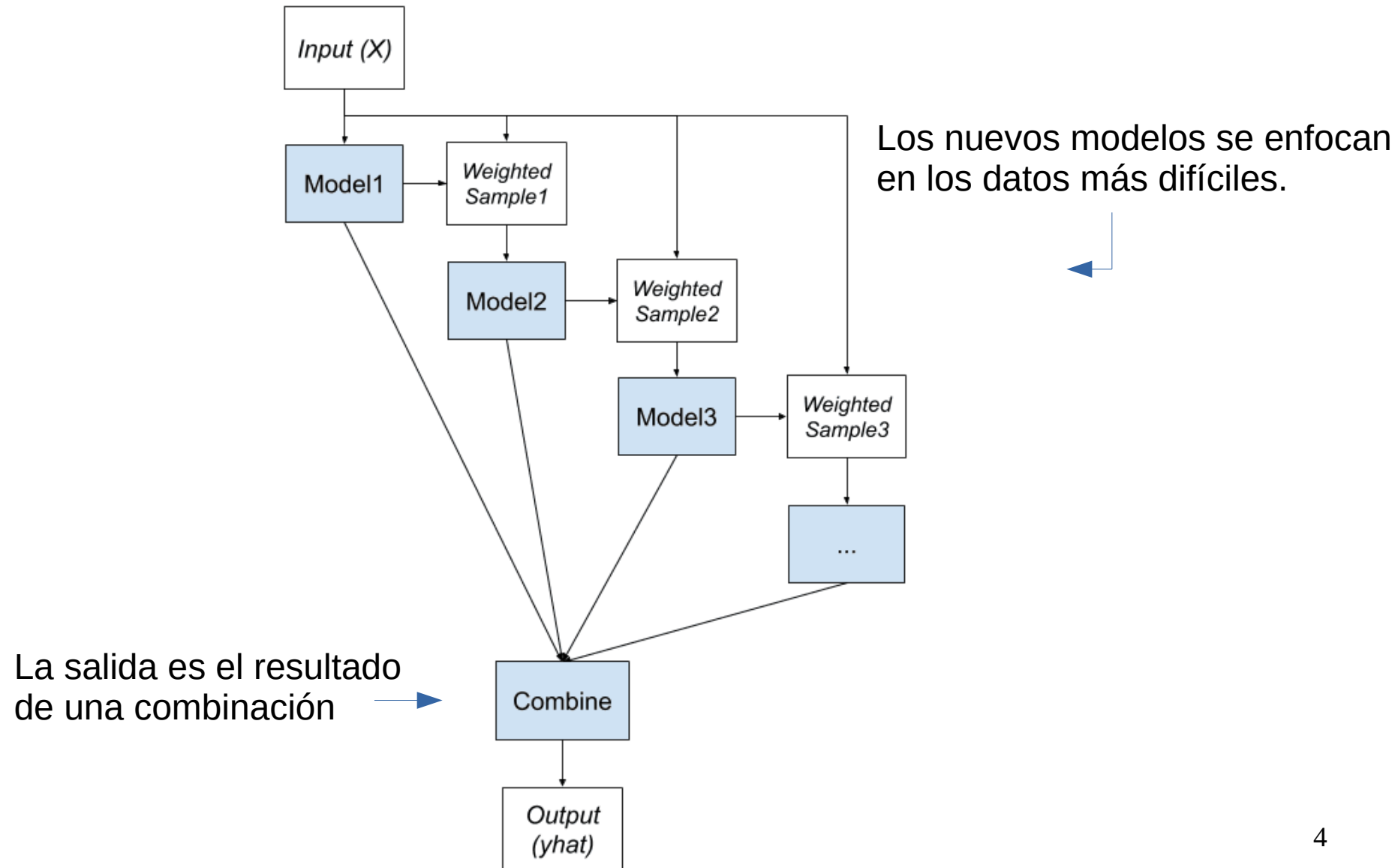
Los ensembles denominan a los modelos base *weak learners*.

Existen tres estrategias para combinar modelos:

- Boosting (**AdaBoost**, **XGBoost**, ...)
- Bagging (**Random Forest**, extra trees, ...)
- Stacking (canonical stacking, super ensemble, ...)

# Boosting

## Boosting Ensemble



# Boosting

---

**Input:** Sample distribution  $\mathcal{D}$ ;  
Base learning algorithm  $\mathcal{L}$ ;  
Number of learning rounds  $T$ .

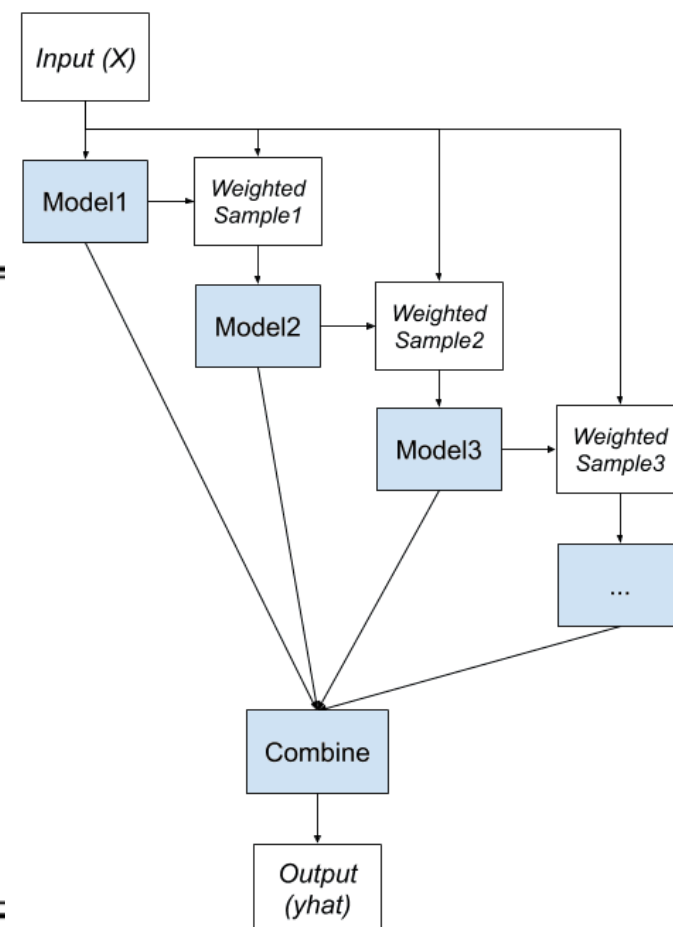
**Process:**

1.  $\mathcal{D}_1 = \mathcal{D}$ .     % Initialize distribution
2. **for**  $t = 1, \dots, T$ :
3.      $h_t = \mathcal{L}(\mathcal{D}_t)$ ;   % Train a weak learner from  $\mathcal{D}_t$
4.      $\epsilon_t = P_{\mathbf{x} \sim \mathcal{D}_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$ ;   % Evaluate the error
5.      $\mathcal{D}_{t+1} = \text{Adjust\_Distribution}(\mathcal{D}_t, \epsilon_t)$
6. **end**

**Output:**  $H(\mathbf{x}) = \text{Combine\_Outputs}(\{h_1(\mathbf{x}), \dots, h_t(\mathbf{x})\})$

---

Boosting Ensemble



# AdaBoost

AdaBoost es un algoritmo basado en Boosting muy usado.

AdaBoost tiene por objetivo minimizar la pérdida según:

$$\ell_{\text{exp}}(h \mid \mathcal{D}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})h(\mathbf{x})}] \quad \text{Exponential loss}$$

└─ Clases en  $\{-1, +1\}$

# AdaBoost

AdaBoost es un algoritmo basado en Boosting muy usado.

AdaBoost tiene por objetivo minimizar la pérdida según:

$$\ell_{\text{exp}}(h \mid \mathcal{D}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})h(\mathbf{x})}] \quad \text{Exponential loss}$$

└─ Clases en  $\{-1, +1\}$   
└─ Error  $\rightarrow e$

Los *weak learners* se combinan de forma lineal:

$$H(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) .$$

└─ Hay que ajustarlos

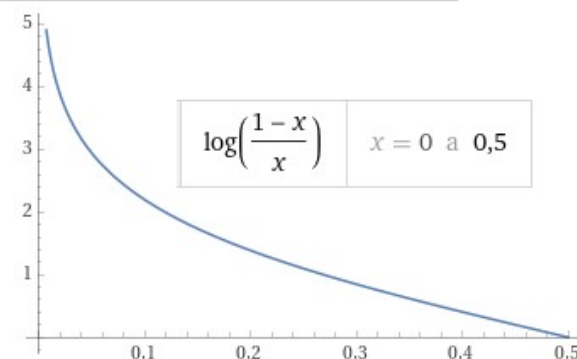
# AdaBoost

Unamos las piezas para obtener el algoritmo.

**Input:** Data set  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ;  
Base learning algorithm  $\mathcal{L}$ ;  
Number of learning rounds  $T$ .

## Process:

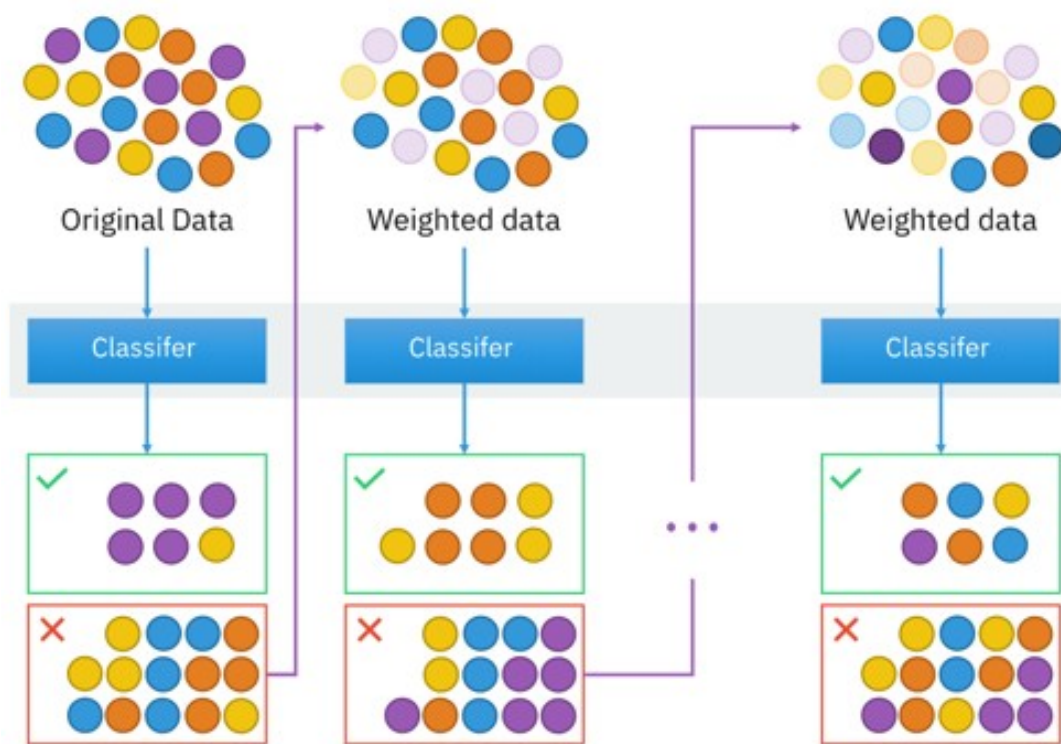
1.  $\mathcal{D}_1(\mathbf{x}) = 1/m$ .    % Initialize the weight distribution
  2. **for**  $t = 1, \dots, T$ :
  3.     $h_t = \mathcal{L}(D, \mathcal{D}_t)$ ; % Train a classifier  $h_t$  from  $D$  under distribution  $\mathcal{D}_t$
  4.     $\epsilon_t = P_{\mathbf{x} \sim \mathcal{D}_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$ ; % Evaluate the error of  $h_t$
  5.    **if**  $\epsilon_t > 0.5$  **then break**
  6.     $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ ; % Determine the weight of  $h_t$
  7.     $\mathcal{D}_{t+1}(\mathbf{x}) = \mathcal{D}_t(\mathbf{x}) \cdot e^{-f(\mathbf{x})\alpha_t h_t(\mathbf{x})} \frac{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_t(\mathbf{x})}]}$  > 1 si  $H_t$  es mejor que  $H_{t-1}$
  8. **end**
- Output:**  $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$  Se deben cumplir dos condiciones para aumentar la probabilidad de muestreo de  $\mathbf{x}$  ¿Cuáles son?





# AdaBoost

## BOOSTING LEARNING PROCEDURE



Strong Learner

Weak Learners

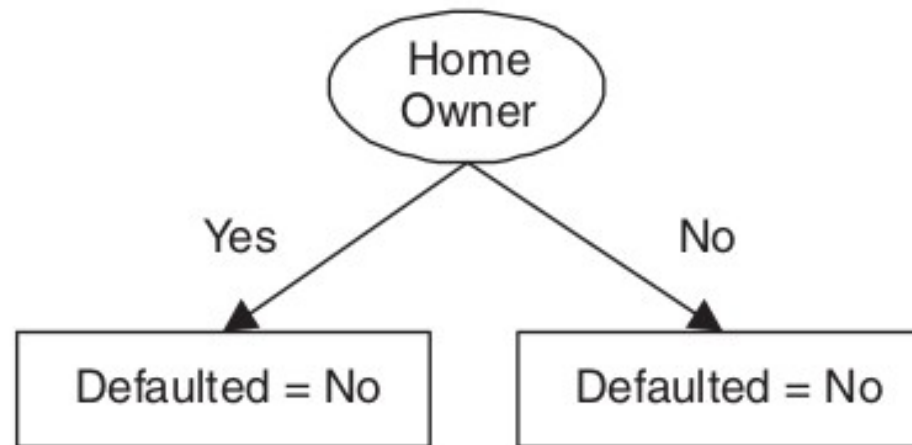
$$f(x) = \sum_t \alpha_t h_t(x)$$

Weight calculated by considering the last iteration's error

# XGBoost

# Weak learner

## Árbol de decisión



Objetivo: el *split* produce nodos puros

Minimizar Gini index  $= 1 - \sum_{i=0}^{c-1} p_i(t)^2$



Fracción de ejemplos de una clase en el nodo

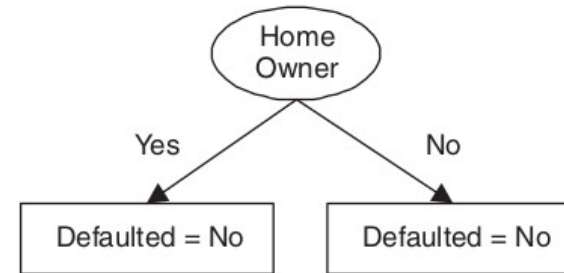
Si todos los ejemplos de una clase están en un nodo, Gini = 0

# Weak learner

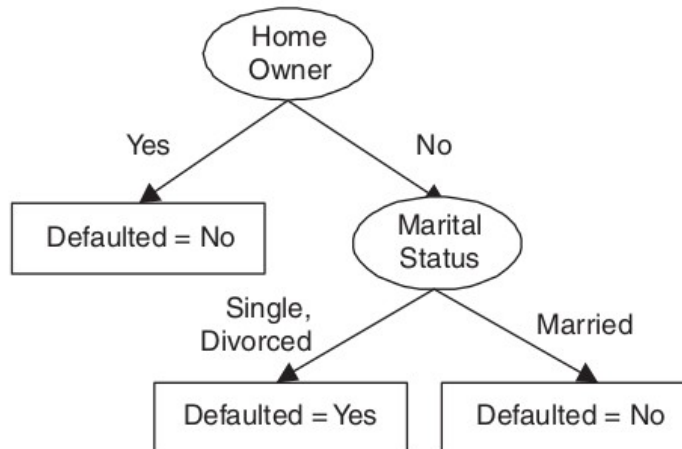
## Árbol de decisión



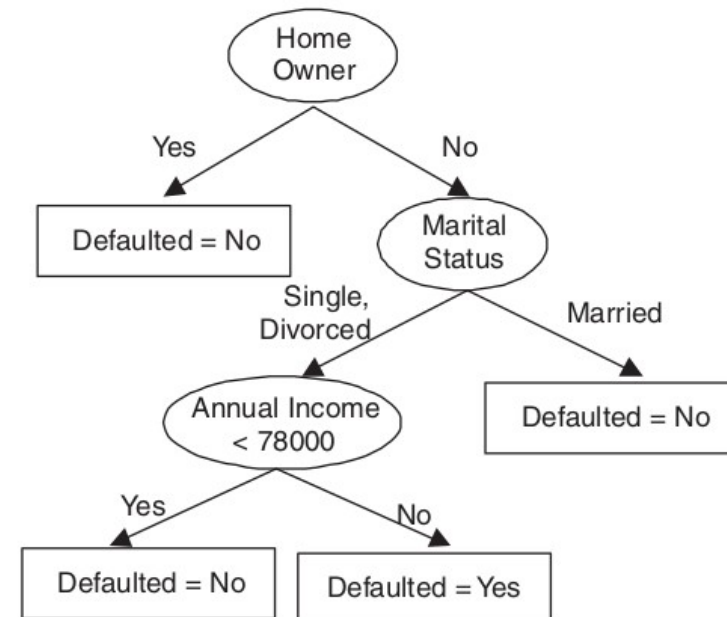
(a)



(b)



(c)



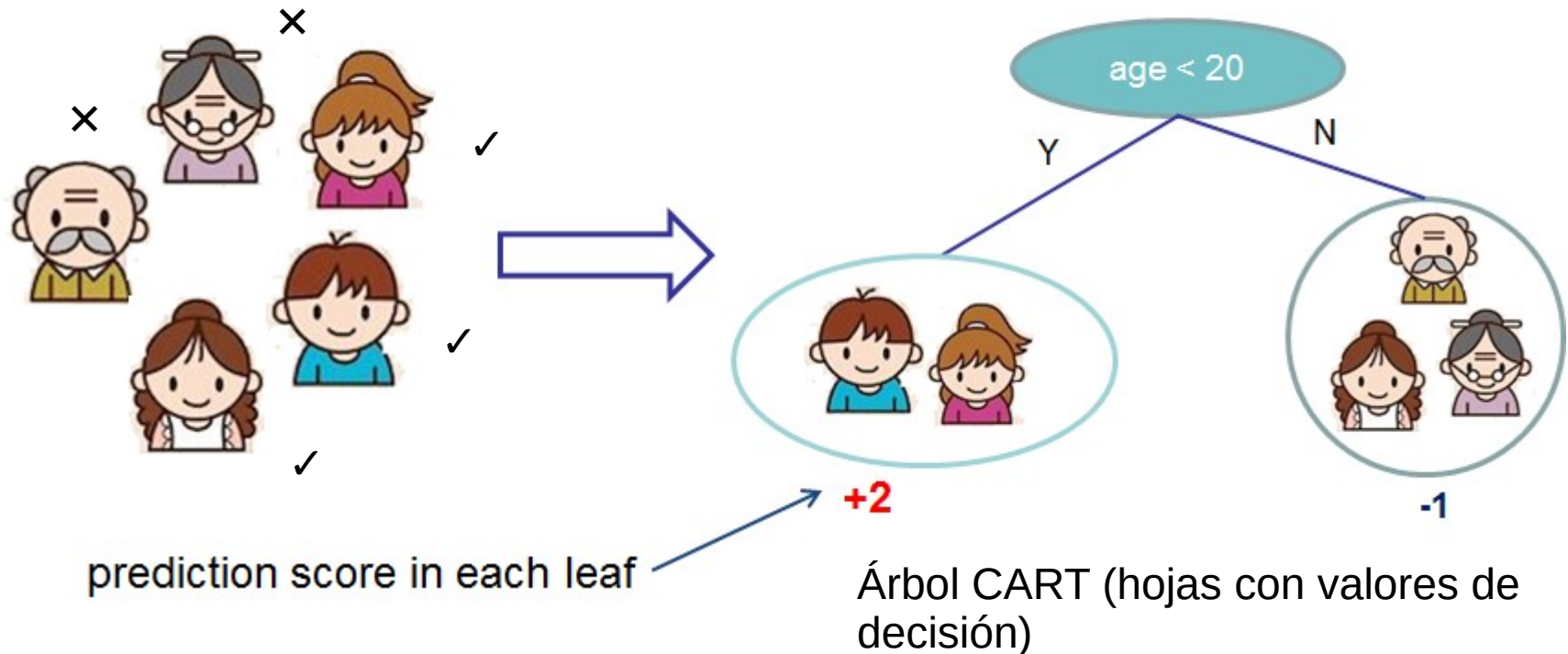
(d)

# XGBoost (eXtreme Gradient Boosting)

- Probablemente el algoritmo más usado de boosting.
- Su algoritmo base es el árbol de decisión.

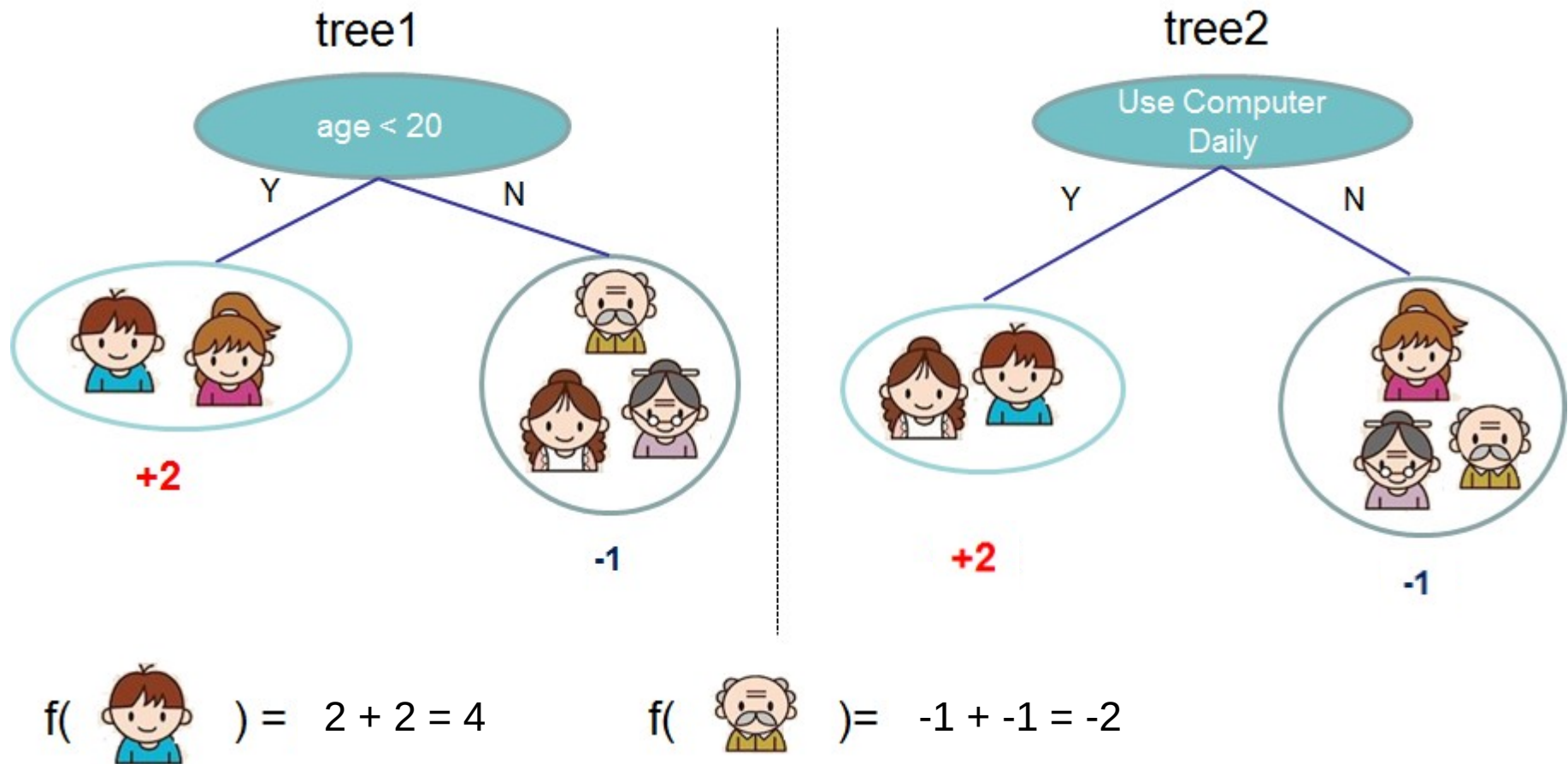
Input: age, gender, occupation, ...

Like the computer game X



# XGBoost (eXtreme Gradient Boosting)

Supongamos que clasificamos a estas personas en diferentes hojas usando dos árboles CART. Podemos combinar los scores de ámbos árboles.

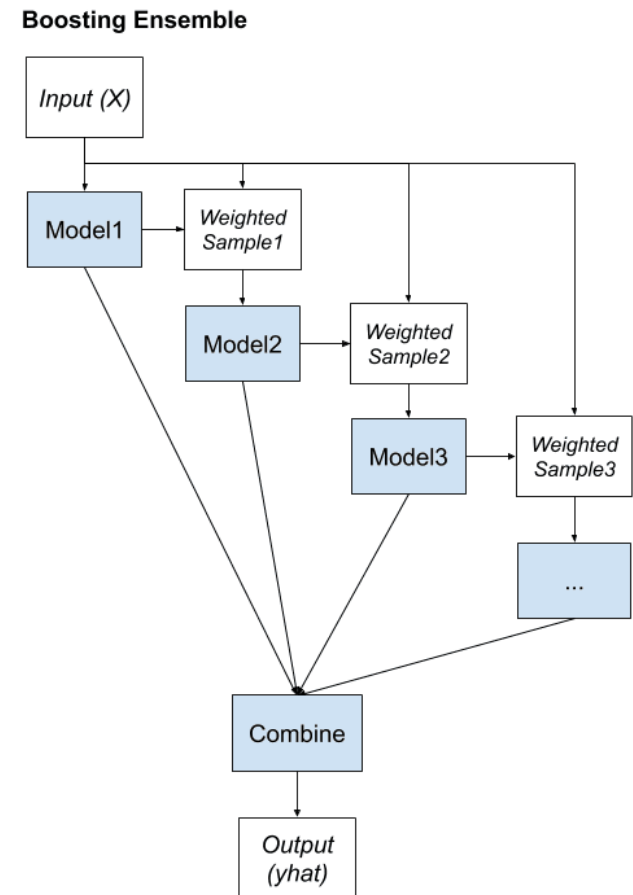


# XGBoost (eXtreme Gradient Boosting)

Podemos combinar las salidas CART de cada árbol para obtener la salida del ensemble:

$$H(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^T h_i(\mathbf{x}).$$

Recordar que para entrenar un ensemble XGBoost, estamos usando boosting, es decir, entrenamos un árbol a la vez en base a los  $t-1$  anteriores que están fijos.

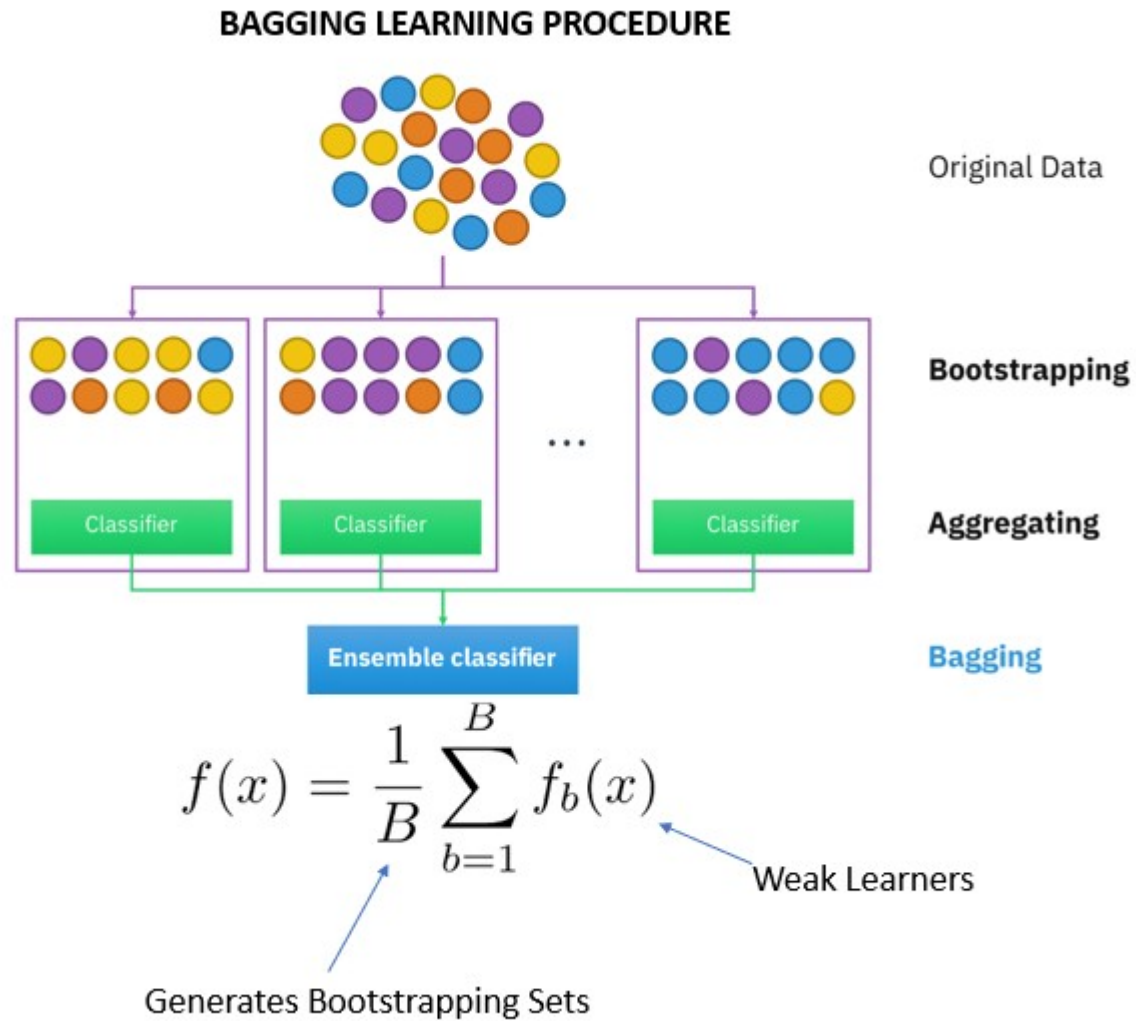


# Bagging y Random Forest



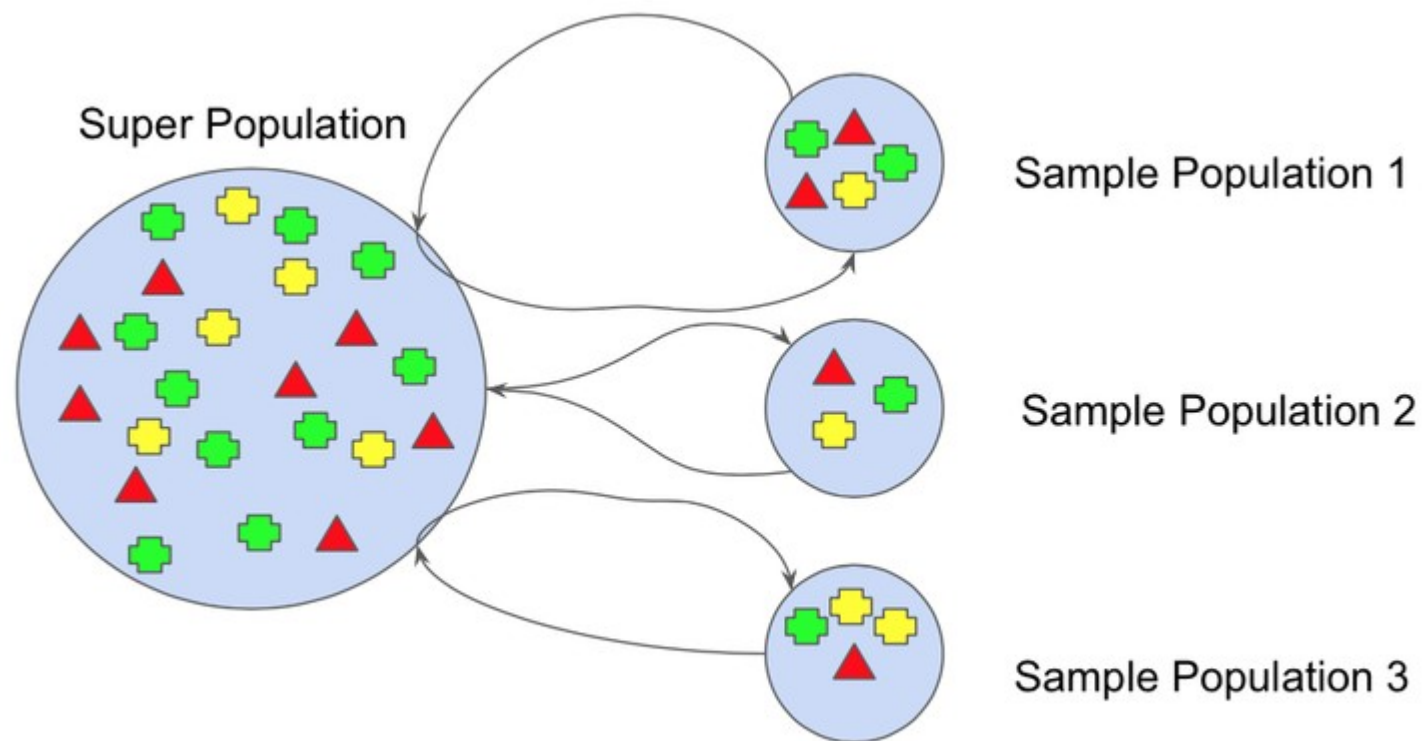
# Bagging

- Método de ensembles paralelo.



# Bagging

- Bootstrap: crear múltiples subsets del dataset usando muestreo con reemplazo, es decir, luego de muestrear, el dato vuelve al universo. Por lo tanto, una muestra puede aparecer en más de una partición.



# Bagging


---

---

**Input:** Data set  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ;  
Base learning algorithm  $\mathfrak{L}$ ;  
Number of base learners  $T$ .

**Process:**

1. **for**  $t = 1, \dots, T$ :
2.  $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$  %  $\mathcal{D}_{bs}$  is the bootstrap distribution
3. **end**

**Output:**  $H(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\mathbf{x}) = y)$   Votación!

---

---

# Bagging


---

---

**Input:** Data set  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ;  
Base learning algorithm  $\mathfrak{L}$ ;  
Number of base learners  $T$ .

**Process:**

1. **for**  $t = 1, \dots, T$ :
2.  $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$  %  $\mathcal{D}_{bs}$  is the bootstrap distribution
3. **end**

**Output:**  $H(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\mathbf{x}) = y)$   Votación!

---

---

El método más conocido de Bagging se llama Random Forest y corresponde a un ensemble de árboles de decisión.

## Random Forest (bagging)

El método base o weak learner es el **random tree**.

---

---

**Input:** Data set  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ;  
Feature subset size  $K$ .

**Process:**

1.  $N \leftarrow$  create a tree node based on  $D$ ;
2. **if** *all instances in the same class* **then return**  $N$
3.  $\mathcal{F} \leftarrow$  the set of features that can be split further;
4. **if**  $\mathcal{F}$  *is empty* **then return**  $N$
5.  $\tilde{\mathcal{F}} \leftarrow$  select  $K$  features from  $\mathcal{F}$  randomly; Random sobre características
6.  $N.f \leftarrow$  the feature which has the best split point in  $\tilde{\mathcal{F}}$ ;
7.  $N.p \leftarrow$  the best split point on  $N.f$ ;
8.  $D_l \leftarrow$  subset of  $D$  with values on  $N.f$  smaller than  $N.p$ ;
9.  $D_r \leftarrow$  subset of  $D$  with values on  $N.f$  no smaller than  $N.p$ ;
10.  $N_l \leftarrow$  call the process with parameters  $(D_l, K)$ ;
11.  $N_r \leftarrow$  call the process with parameters  $(D_r, K)$ ;
12. **return**  $N$

**Output:** A random decision tree

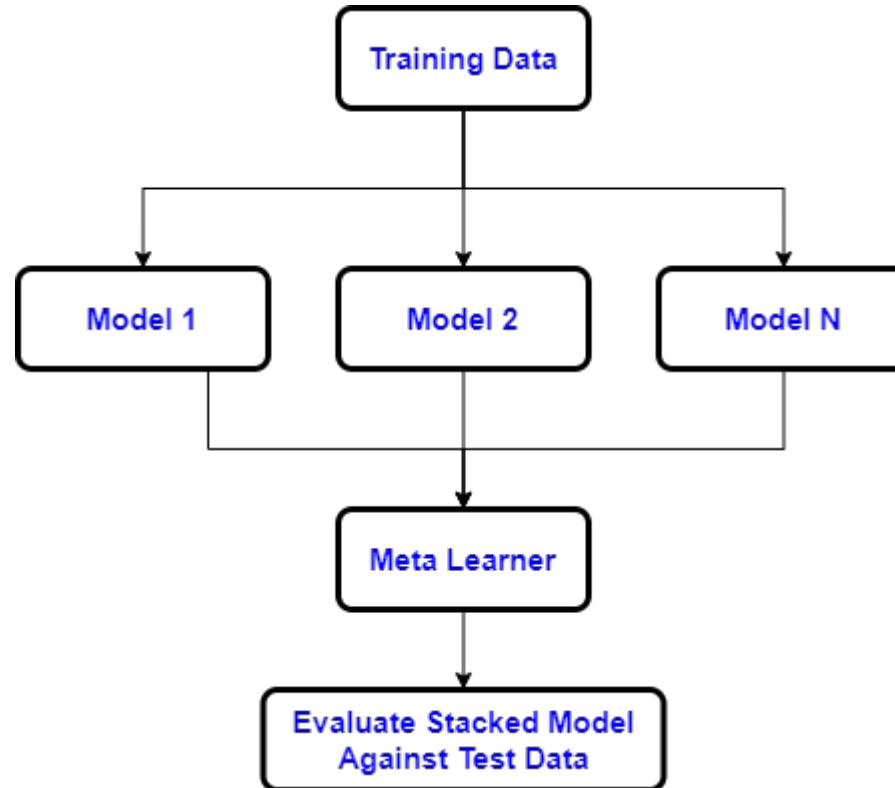
---

---

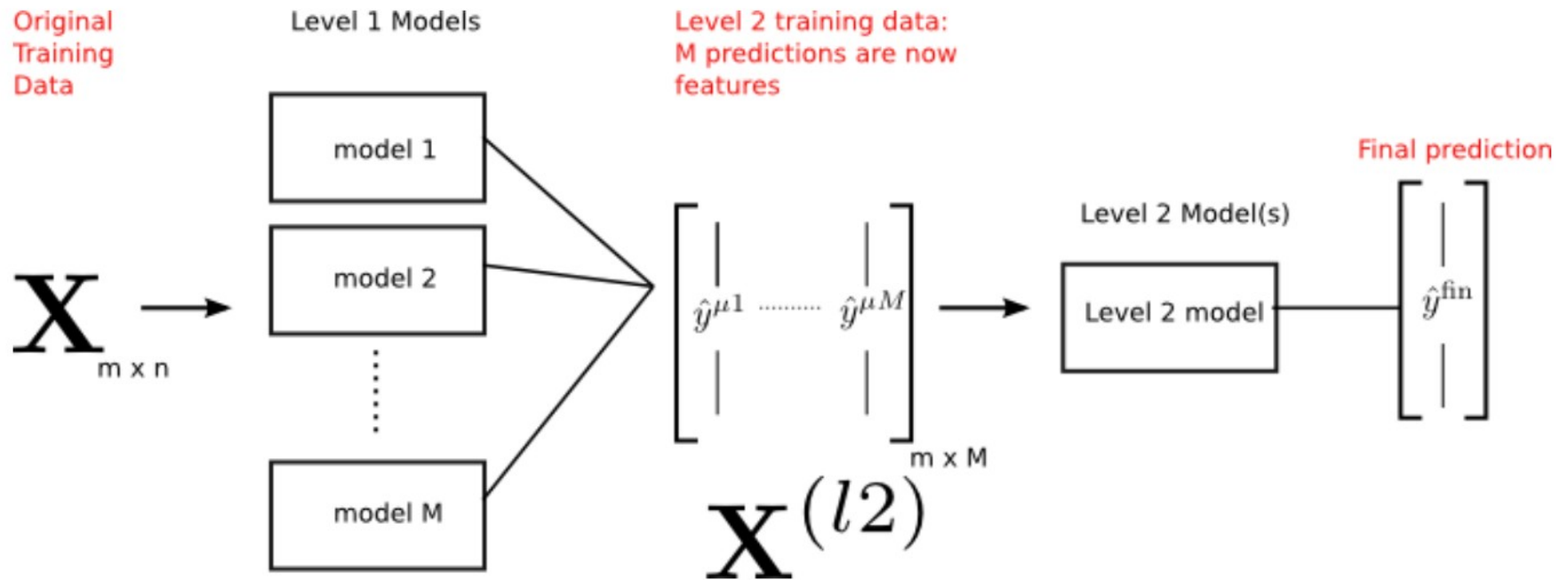
# Stacking

# Stacking

- Entrenar un **meta-learner** sobre la base de las salidas de learners base.



# Stacking



Las salidas de cada weak learner son las características usadas por el meta-learner



# Stacking

- Entrenar un meta-learner sobre la base de las salidas de learners base.

---

---

**Input:** Data set  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ;  
First-level learning algorithms  $\mathcal{L}_1, \dots, \mathcal{L}_T$ ;  
Second-level learning algorithm  $\mathcal{L}$ .

**Process:**

1. **for**  $t = 1, \dots, T$ :   % Train a first-level learner by applying the
2.      $h_t = \mathcal{L}_t(D)$ ;   % first-level learning algorithm  $\mathcal{L}_t$
3. **end**
4.    $D' = \emptyset$ ;                   % Generate a new data set
5. **for**  $i = 1, \dots, m$ :
6.     **for**  $t = 1, \dots, T$ :
7.          $z_{it} = h_t(\mathbf{x}_i)$ ;
8.     **end**
9.      $D' = D' \cup ((z_{i1}, \dots, z_{iT}), y_i)$ ;
10. **end**
11.  $h' = \mathcal{L}(D')$ ;               % Train the second-level learner  $h'$  by applying  
                                  % the second-level learning algorithm  $\mathcal{L}$  to the  
                                  % new data set  $D'$ .

**Output:**  $H(\mathbf{x}) = h'(h_1(\mathbf{x}), \dots, h_T(\mathbf{x}))$

---

---

# Stacking

Se suelen usar ensambles heterogéneos.

