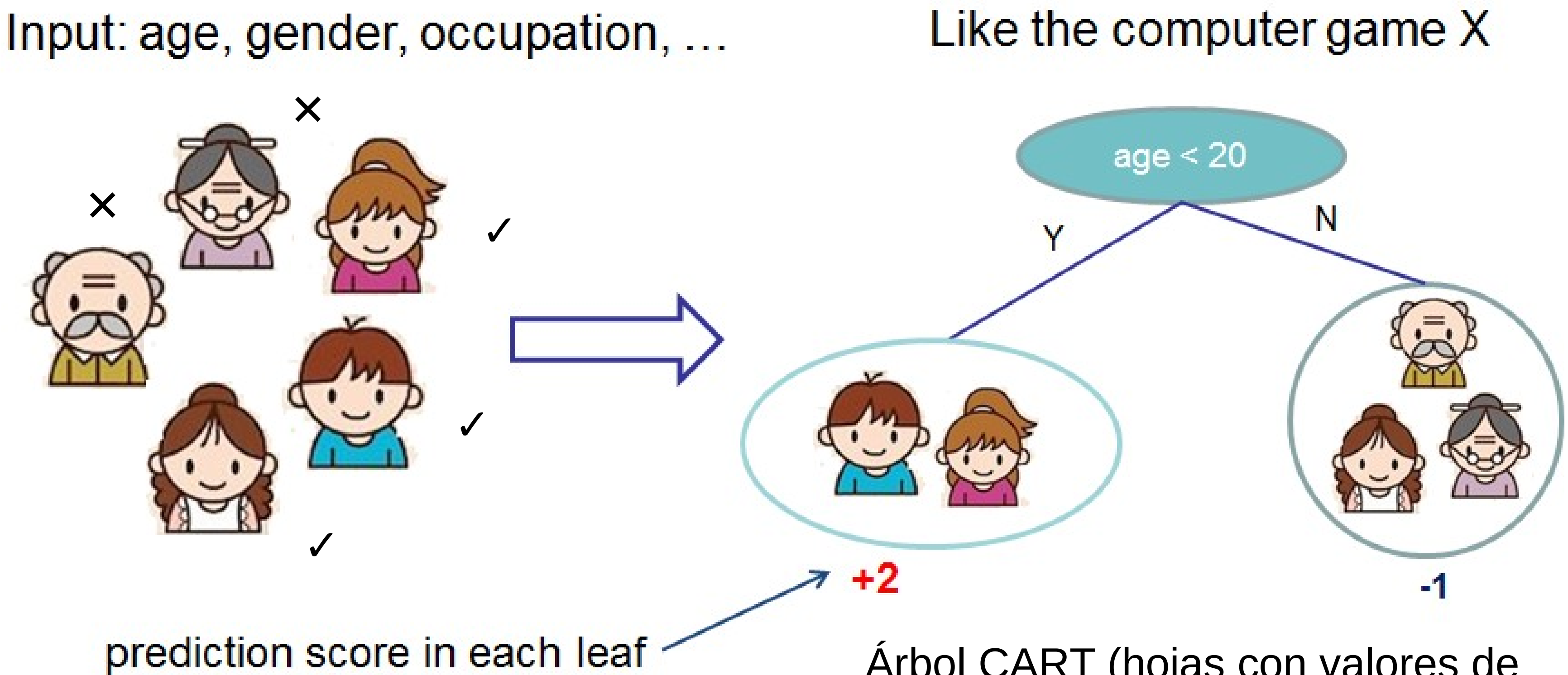
IIC 2433 Minería de Datos

https://github.com/marcelomendoza/IIC2433

Ensembles

XGBoost (eXtreme Gradient Boosting)

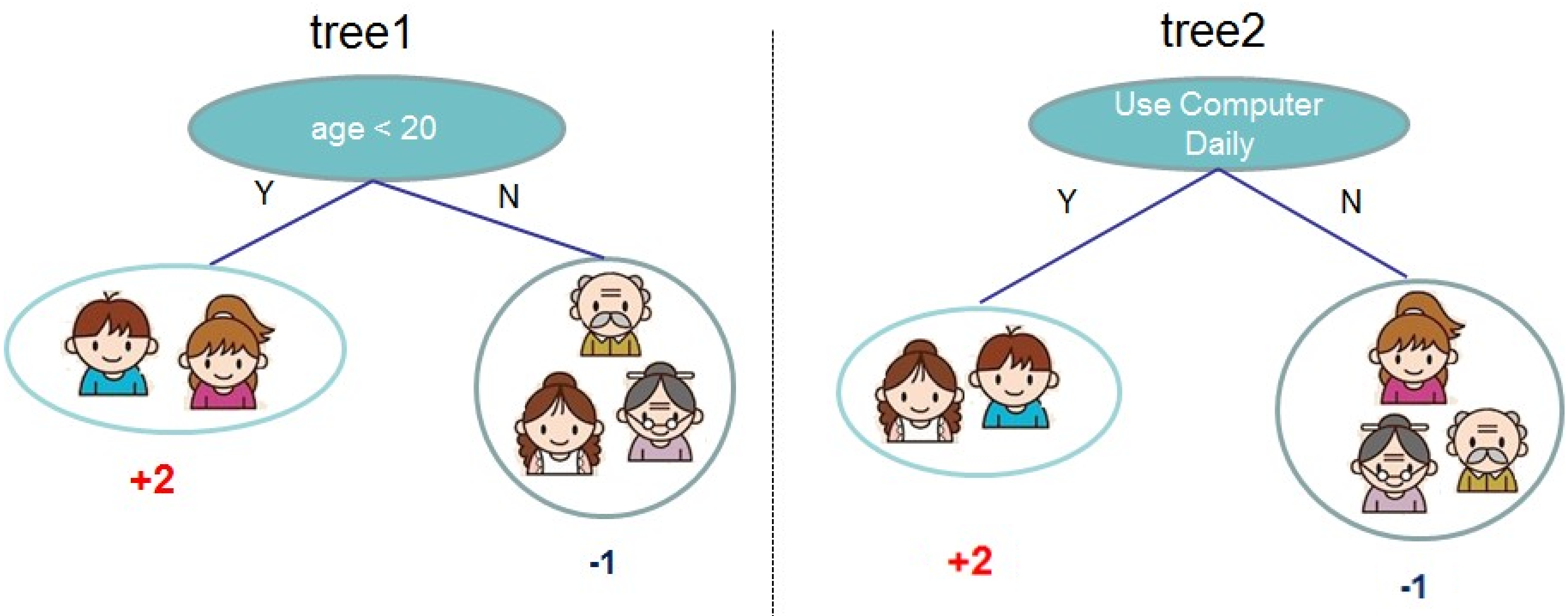
- Probablemente el algoritmo más usado de boosting.
- Su algoritmo base es el árbol de decisión.



Árbol CART (hojas con valores de decisión)

XGBoost (eXtreme Gradient Boosting)

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



f() = 2 + 2 = 4

f()= -1 + -1 = -2

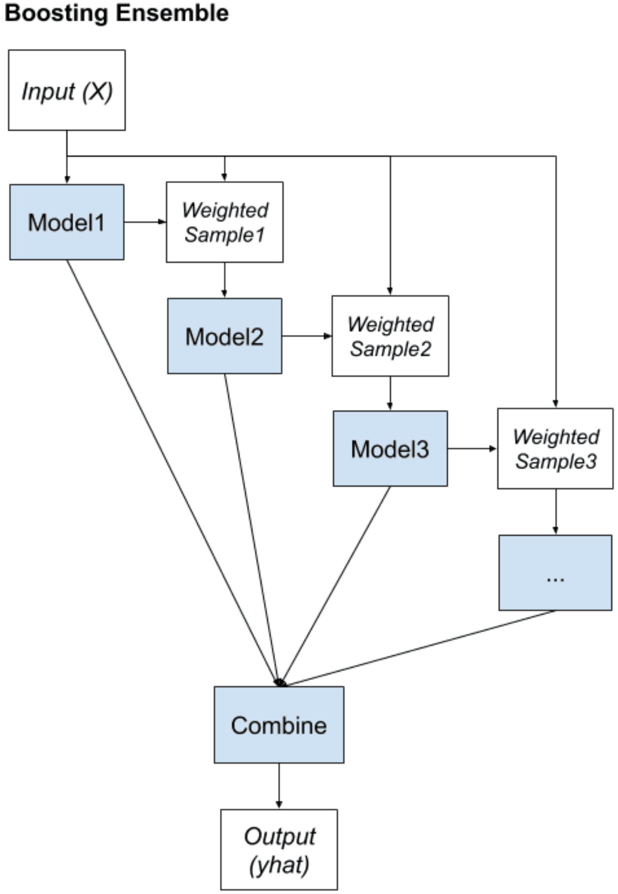
Se deben balancear los datos!

XGBoost (eXtreme Gradient Boosting)

Es decir, podemos combinar las salidas CART de cada árbol para obtener la salida del ensemble:

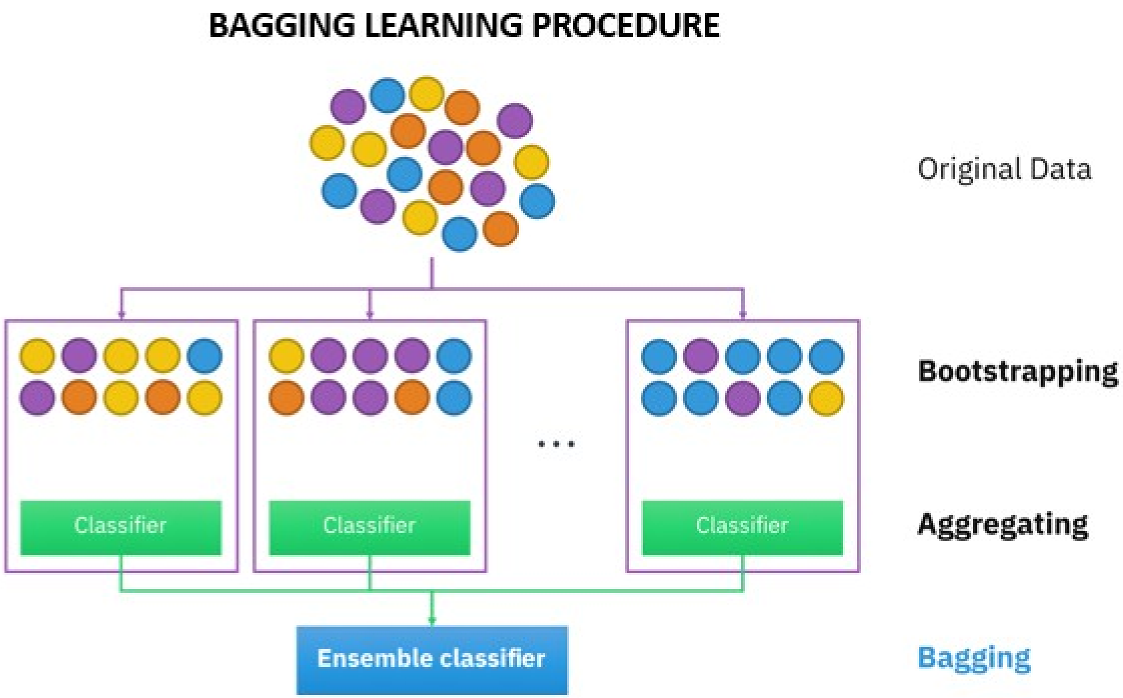
$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x}).$$

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

- Método de ensembles paralelo.



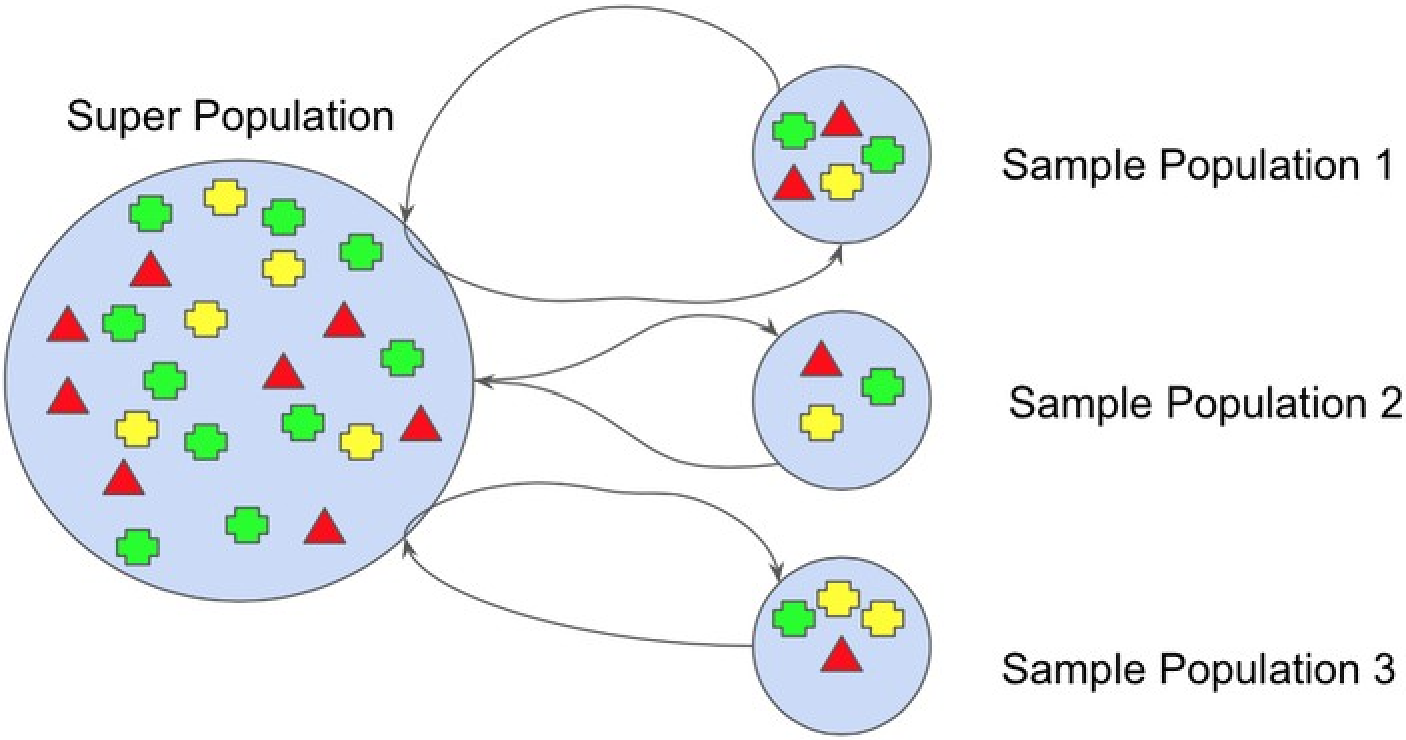
$$f(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$$

Weak Learners

Generates Bootstrapping Sets

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 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of base learners T.

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

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

Bagging

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

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

Output: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$ ← 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 (random tree)

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
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;
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

Random sobre características

Estrategias de combinación

Promedio simple (sobre salida):

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

El error esperado es:

$$err(H) = \frac{1}{T}\overline{err}(h)$$

Estrategias de combinación

Promedio simple (sobre salida):

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

El error esperado es:

$$err(H) = \frac{1}{T}\overline{err}(h)$$

Promedio ponderado (sobre salida):

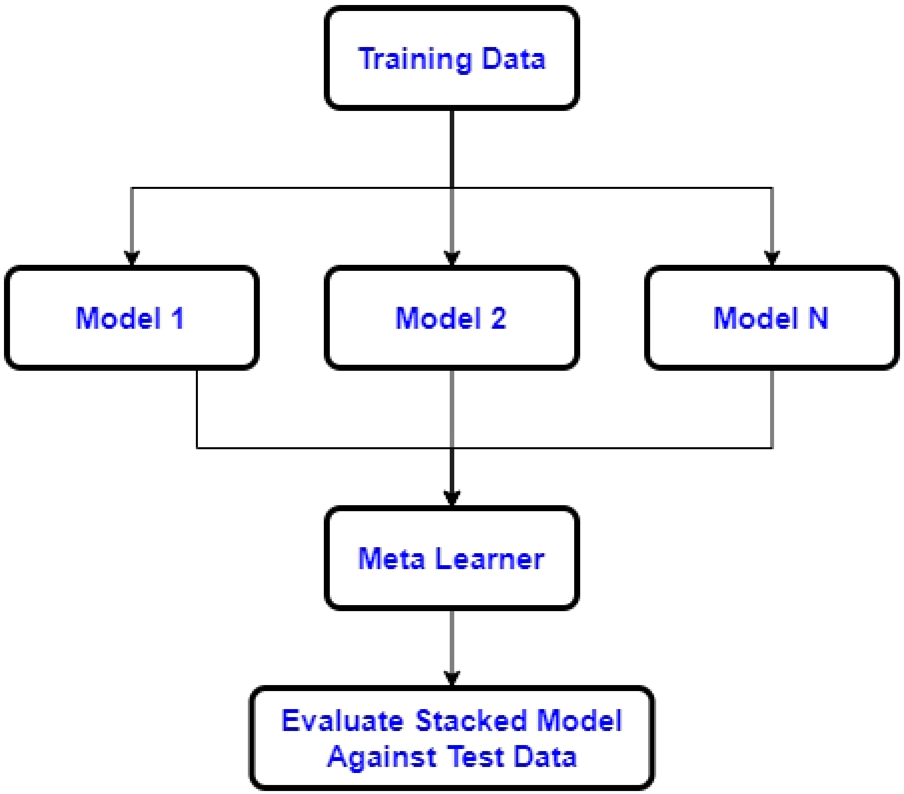
(usado en boosting, por ejemplo, AdaBoost)

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})\,,$$

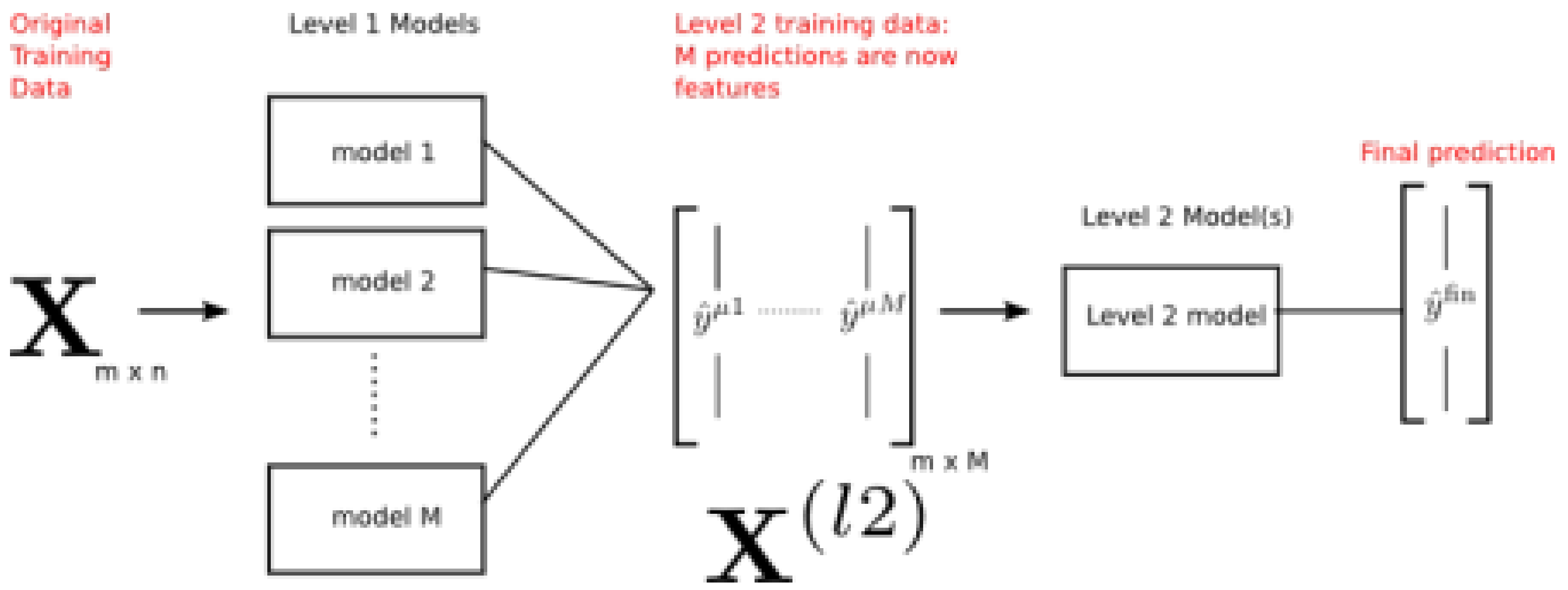
con: $w_i \geq 0$ y $\sum_{i=1}^{T} w_i = 1\,.$

Stacking

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



Stacking



Original
Training
Data
Level 1 Models
Level 2 training data:
M predictions are now
features
Final prediction
model 1
model 2
model M
Level 2 Model(s)
Level 2 model
$\mathbf{X}_{m \times n}$
$\hat{y}^{\mu 1}$ .... $\hat{y}^{\mu M}$
m x M
$\mathbf{X}^{(l2)}$
$\hat{y}^{\text{fin}}$

Stacking

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

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

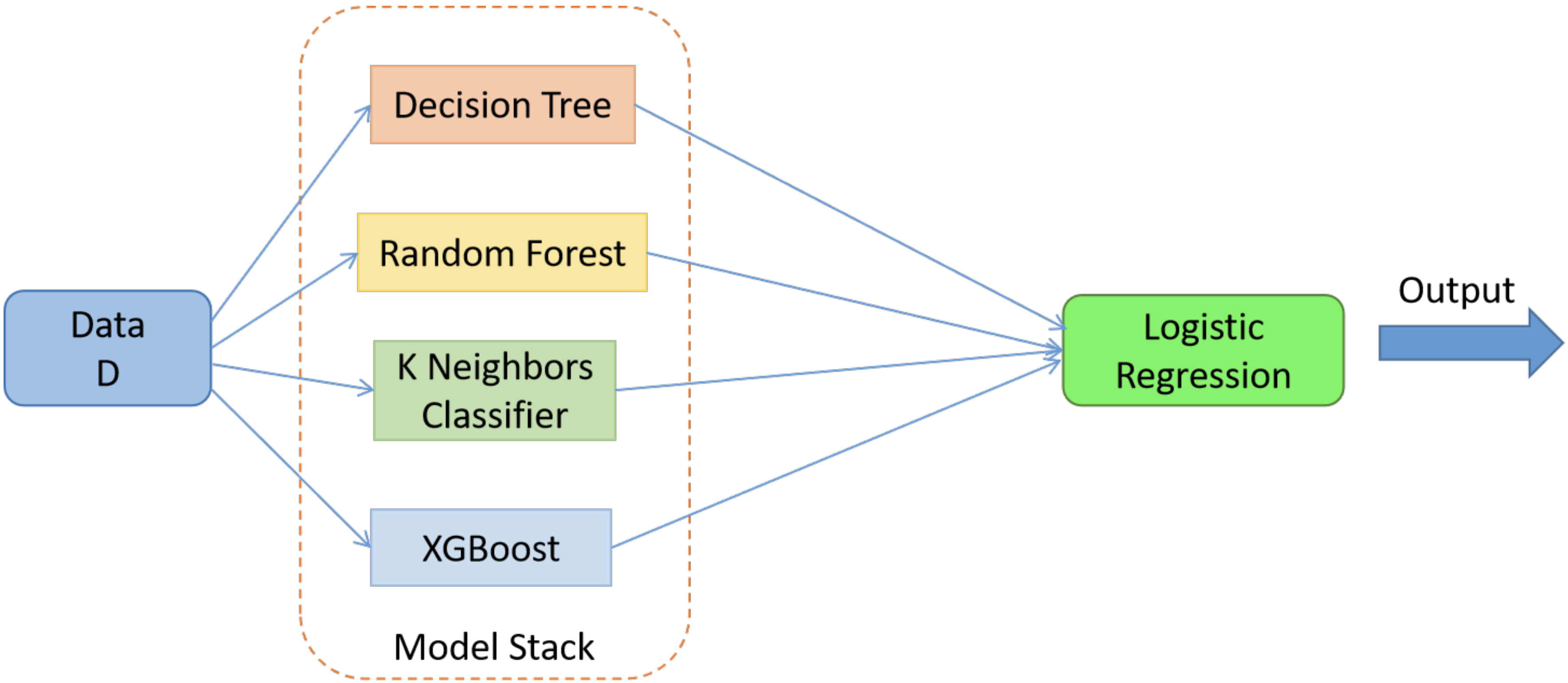
Process:

for $t = 1, \ldots, T$: % Train a first-level learner by applying the
  $h_t = \mathfrak{L}_t(D)$; % first-level learning algorithm $\mathfrak{L}_t$
end
$D' = \emptyset$; % Generate a new data set
for $i = 1, \ldots, m$:
  for $t = 1, \ldots, T$:
    $z_{it} = h_t(\boldsymbol{x}_i)$;
  end
  $D' = D' \cup ((z_{i1}, \ldots, z_{iT}), y_i)$;
end
$h' = \mathfrak{L}(D')$; % Train the second-level learner h' by applying
% the second-level learning algorithm $\mathfrak{L}$ to the
% new data set $\mathcal{D}'$.

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

Stacking

Se suelen usar ensambles heterogéneos.

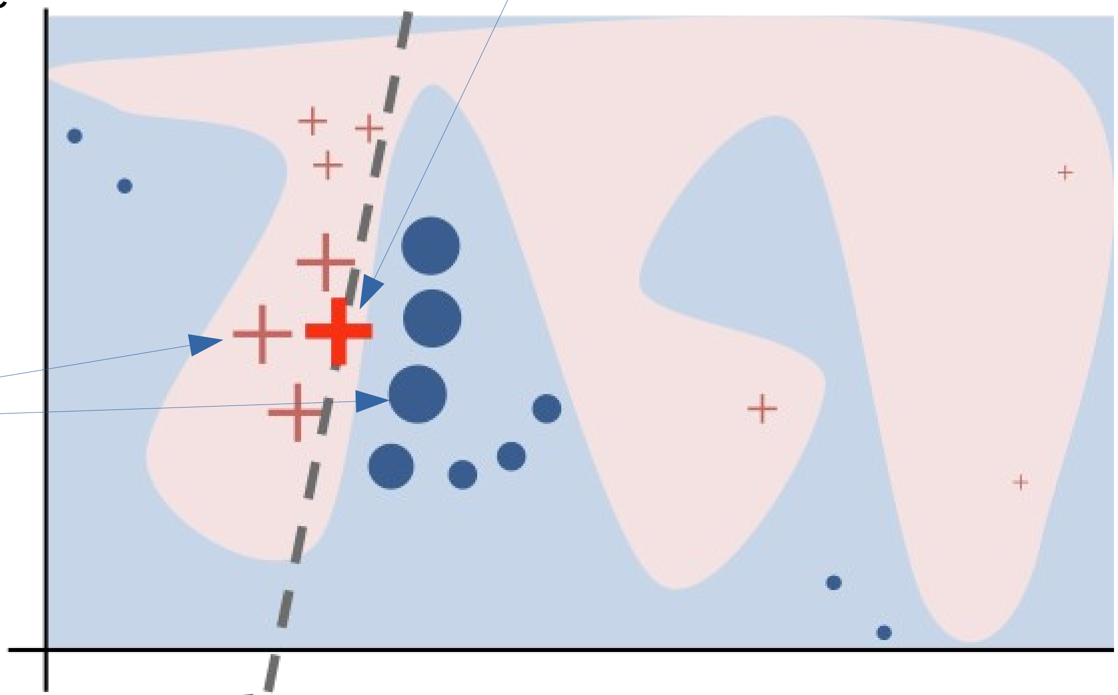


Explainers

LIME (Local Interpretable Model-Agnostic Explanations)

Instancia a explicar

1. Se perturban (eliminan features) de la instancia



2. Se obtienen las predicciones sobre los ejemplos perturbados

3. Se construye un modelo lineal que aproxima al modelo original usando las predicciones (noisy label)

4. Se identifica cuales perturbaciones inclinan la decisión

LIME (Local Interpretable Model-Agnostic Explanations)