

---

# Árboles de Decisión



---

75.06 Organización de Datos

---

# Temario

---

- Árboles de Decisión
  - Ventajas
  - Limitaciones
  - ID3
  - C4.5
  - Ensamblados
    - Bagging
    - Boosting
  - Random Forests
  - XGBoost
-

# Arboles de Decisión

---

- Árbol con ramas para cada valor en la comparación.
  - En cada nodo dividimos el set de datos de acuerdo a un cierto criterio.
  - Objetivo: llegar a nodos hoja en los cuales podamos clasificar correctamente nuestros datos.
  - **Algoritmos**
    - ID3 [Quinlan, 1979]
    - C4.5 [Quinlan, 1993]
    - C5.0 (Última versión de Quinlan, propietaria. Mejora sobre C4.5)
    - CART (similar a C4.5, soporta regresión, ver scikit-learn).
-

# Arboles de Decisión: Ventajas

---

- Simples de entender e interpretar: modelo de caja blanca.
  - Funcionan con datos numéricos o categóricos.
  - Requieren poca preparación de los datos: no requieren normalización.
  - Buena performance para datasets grandes.
  - Ayudan en la selección de features.
-

# Arboles de Decisión: Limitaciones

---

- Encontrar el óptimo es NP-complete.
  - Algoritmos greedy para encontrar óptimos locales.
- Arboles demasiado complejos generan overfitting.
  - Poda

# ID3

---

- Algoritmo de tipo greedy:
    - En cada paso realiza el mejor split posible en dos.
    - Selecciona el atributo que nos da mayor Ganancia de Información (relación con la entropía).
  - Esto se repite recursivamente hasta construir el árbol final.
  - Features deben ser categóricos.
-

# ID3: Ejemplo

---

Candidato	Presencia	Estudios	Experiencia	Contratado
1	Buena	Universitarios	Alta	<b>SI</b>
2	Mala	Universitarios	Media	<b>NO</b>
3	Buena	Secundarios	Alta	<b>SI</b>
4	Mala	Universitarios	Baja	<b>NO</b>
5	Buena	Secundarios	Media	<b>SI</b>
6	Buena	Universitarios	Media	<b>SI</b>
7	Regular	Primarios	Baja	<b>NO</b>
8	Regular	Universitarios	Media	<b>SI</b>

---

# ID3: Ejemplo

---

- Entropia del set de datos:

$$H[5/8; 3/8] = 0.9544 \quad -\sum P_i * \log_2 P_i$$

Contratado

*SI*

*NO*

*SI*

*NO*

*SI*

*SI*

*NO*

*SI*



# ID3: Ejemplo

---

- Atributo ***Presencia***:

$$H(\text{Presencia}=\text{Buena}) = H[4/4; 0/4] = 0$$

$$H(\text{Presencia}=\text{Mala}) = H[0/2; 0/0] = 0$$

$$H(\text{Presencia}=\text{Regular}) = H[1/2; 1/2] = 1$$

$$H(\text{Presencia}) = 4/8 * 0 + 2/8 * 0 + 2/8 * 1 = 0.25$$

$$GI(\text{Presencia}) = 0.9544 - 0.25 = 0.7044$$

Presencia	SI	NO
Buena	4	0
Mala	0	2
Regular	1	1

# ID3: Ejemplo

---

- Atributo ***Estudios***:

Estudios	SI	NO
Univ	3	2
Sec	2	0
Prim	0	1

$$H(\text{Estudios}=\text{Univ}) = H[\frac{3}{5}; \frac{2}{5}] = 0.971$$

$$H(\text{Estudios}=\text{Sec}) = H[\frac{2}{2}; \frac{0}{2}] = 0$$

$$H(\text{Estudios}=\text{Prim}) = H[\frac{0}{1}; \frac{1}{1}] = 0$$

$$H(\text{Estudios}) = \frac{5}{8} * 0.971 + \frac{2}{8} * 0 + \frac{1}{8} * 0 = 0.6069$$

$$GI(\text{Estudios}) = 0.9544 - 0.6069 = 0.3475$$

---

# ID3: Ejemplo

---

- Atributo ***Experiencia:***

$$H(\text{Experiencia}=\text{Alta}) = H[2/2; 0/2] = 0$$

$$H(\text{Experiencia}=\text{Media}) = H[3/4; 1/4] = 0.8113$$

$$H(\text{Experiencia}=\text{Baja}) = H[0/2; 2/2] = 0$$

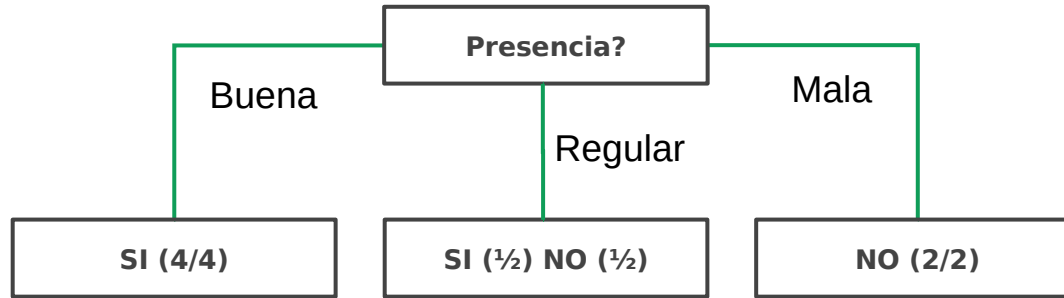
$$H(\text{Experiencia}) = 2/8 * 0 + 4/8 * 0.8113 + 2/8 * 0 = 0.40565$$

$$GI(\text{Experiencia}) = 0.9544 - 0.40565 = 0.54875$$

Experiencia	SI	NO
Alta	2	0
Media	3	1
Baja	0	2

# ID3: Ejemplo

---

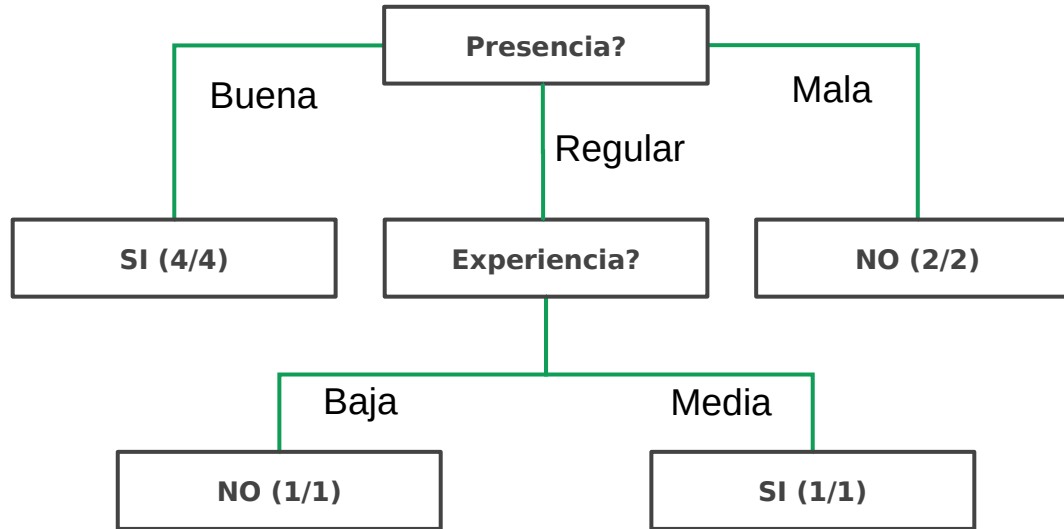


Candidato	Presencia	Estudios	Experiencia	Contratado
7	Regular	Primarios	Baja	<b>NO</b>
8	Regular	Universitarios	Media	<b>SI</b>

---

# ID3: Ejemplo

---



# ID3

---

- Gran peligro de Overfitting.
    - Realizar preguntas hasta que todas las hojas tengan nodos de una sola clase.
  - Uso del hiper-parámetro minbucket.
    - Indica la cantidad mínima de registros que puede tener un nodo no hoja.
  - Hojas del árbol indican la probabilidad de cada clase en base a cuantos registros de cada clase hay en el nodo hoja.
-

## C4.5

---

- Sucesor de ID3.
  - Acepta atributos numéricos.
  - Acepta datos con atributos faltantes.
  - Los atributos pueden tener un peso.
  - Poda del árbol.
-

# C4.5: Ejemplo

---

Candidato	Presencia	Estudios	Experiencia	Edad	Contratado
1	Buena	Universitarios	Alta	33	<i>SI</i>
2	Mala	Universitarios	Media	27	<i>NO</i>
3	Buena	Secundarios	Alta	41	<i>SI</i>
4	Mala	Universitarios	Baja	35	<i>SI</i>
5	Buena	Secundarios	Media	37	<i>SI</i>
6	Buena	Universitarios	Media	28	<i>SI</i>
7	Regular	Primarios	Baja	25	<i>NO</i>
8	Regular	Universitarios	Media	40	<i>SI</i>



# C4.5: Ejemplo

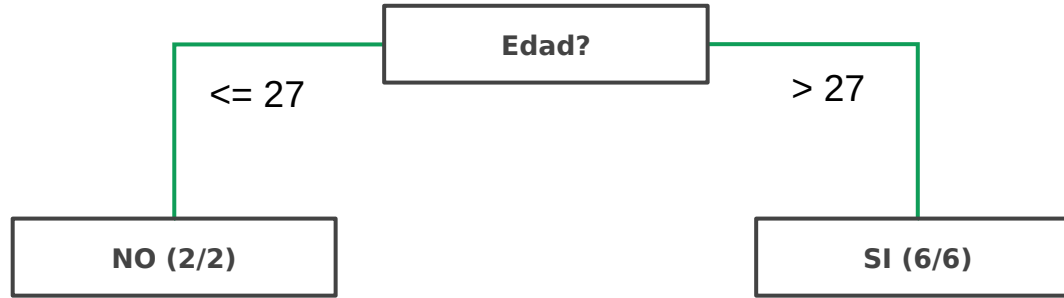
---

Edad	Contratado	
25	<i>NO</i>	Split por $\leq 25 \rightarrow$ Calculo GI
27	<i>NO</i>	$H(\text{Edad} \leq 25) = H(0/1, 1/1) = 0$
28	<i>SI</i>	$H(\text{Edad} > 25) = H(6/7, 1/7) = 0.5917$
33	<i>SI</i>	
35	<i>SI</i>	Split por $\leq 27 \rightarrow$ Calculo GI
37	<i>SI</i>	$H(\text{Edad} \leq 27) = H(0/2, 2/2) = 0$
40	<i>SI</i>	$H(\text{Edad} > 27) = H(6/6, 0/6) = 0$
41	<i>SI</i>	

---

# ID3: Ejemplo

---



# Ensamblés

---

- Los mejores algoritmos de ML son combinaciones de varios algoritmos.
  - Aunque no siempre sea lo mejor para usar en la práctica.
-

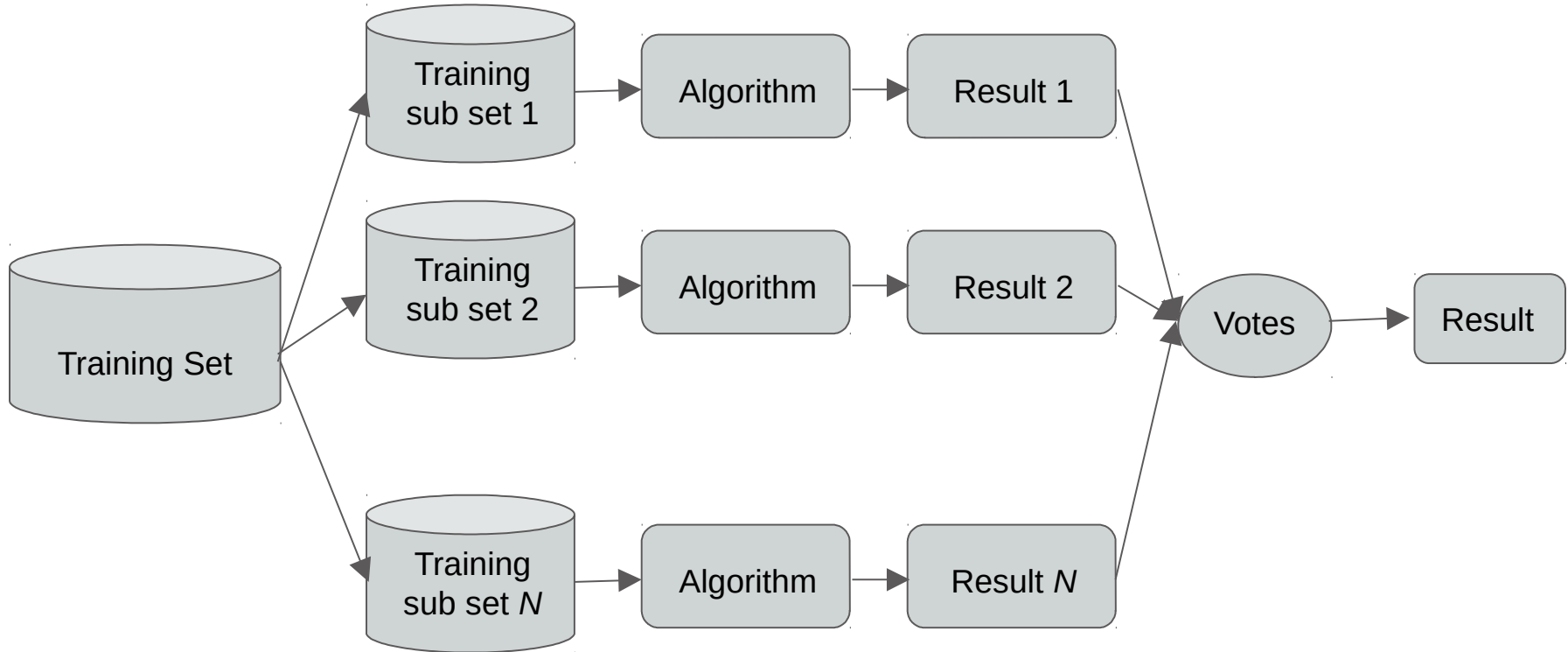
# Bagging

---

- Aplicar el mismo clasificador  $n$  veces usando **Bootstrapping**:
    - Tomamos muestras del set de entrenamiento (con reemplazo) de igual tamaño que este.
  - Promediar sus resultados.
-

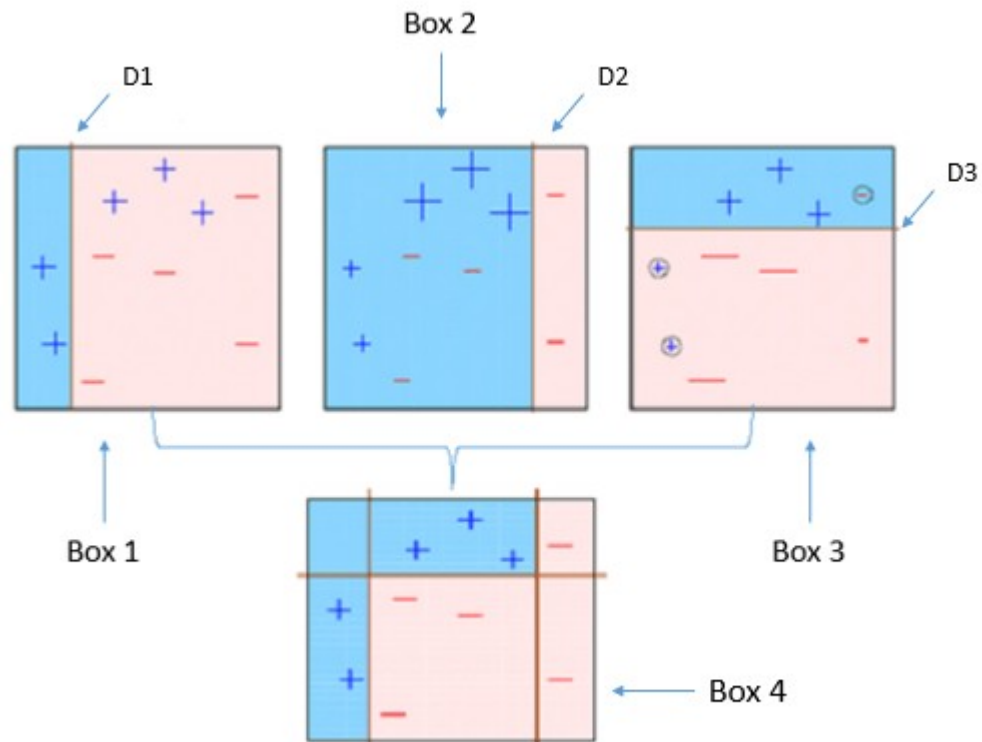
# Bagging

---



# Bagging

---



# Bagging

---

- Disminuye la posibilidad de overfitting
    - Cada clasificador no ve la totalidad de los registros del set de entrenamiento.
  - Registros OOB (Out of Bag)
    - Se usan para ver la precisión del algoritmo (como si fuera un set de test).
-

# Boosting

---

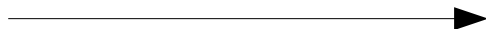
- Entrenar algoritmo simple.
  - Analizar sus resultados.
  - Entrenar otro algoritmo simple en donde se le da mayor peso a los resultados para los cuales el anterior tuvo peor performance.
  - Resultado final en base a un promedio ponderado.
-



# Boosting

---

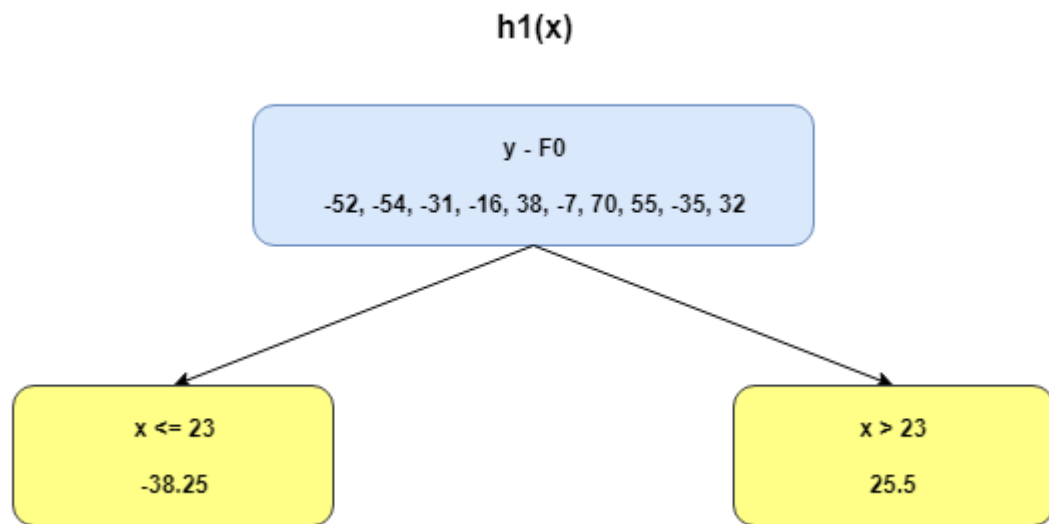
Years	Salary
5	82
7	80
12	103
23	118
25	172
28	127
29	204
34	189
35	99
40	166



x	y	F0	y - F0
5	82	134	-52
7	80	134	-54
12	103	134	-31
23	118	134	-16
25	172	134	38
28	127	134	-7
29	204	134	70
34	189	134	55
35	99	134	-35
40	166	134	32

# Boosting

---



# Boosting

---

x	y	F0	y-F0	h1	F1
5	82	134	-52	-38.25	95.75
7	80	134	-54	-38.25	95.75
12	103	134	-31	-38.25	95.75
23	118	134	-16	-38.25	95.75
25	172	134	38	25.50	159.50
28	127	134	-7	25.50	159.50
29	204	134	70	25.50	159.50
34	189	134	55	25.50	159.50
35	99	134	-35	25.50	159.50
40	166	134	32	25.50	159.50

# Boosting

---

- Se genera nuevo arbol para predecir lo que se predijo mal en el arbol anterior.
  - La profundidad de los árboles generados con Boosting es menor a la generada con Bagging.
-

# Ensamblas: Majority Voting

---

- Cada clasificador da un voto a cada clase.
- Mejor si se usan resultados con poca correlación.
- Se le puede dar un peso distinto a cada modelo.

1111111100 = 80%

1111111100 = 80%

1011111100 = 70%

-----

1111111100 = 80%

1111111100 = 80%

0111011101 = 70%

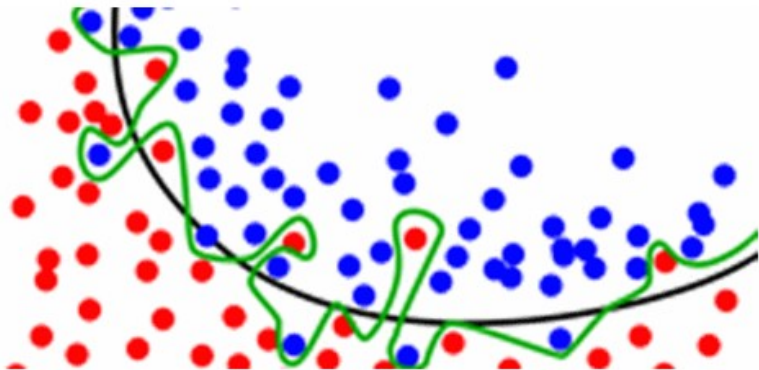
1000101111 = 60%

-----

1111111101 = 90%

# Ensamblas: Averaging

- Promediar el resultado de varios clasificadores.
- Sirve para regresión y clasificación (con clases o probabilidades).
- Ajuste de escala de probabilidades de cada modelo usando rango entre 1 y n (n: total de registros).  $\text{Prob} = 1 - (x - \min) / (\max - \min)$



	C1	R1	C2	R2	Prom	Prob
1	0.57	3	0.3605	1	2	0.75
2	0.04	4	0.3502	3	3.5	0
3	0.96	2	0.35	4	3	0.25
4	0.99	1	0.36	2	1.5	1

# Ensamblas: Blending (o Stacked)

---

- Separar parte pequeña del set de entrenamiento (10%).
  - Con los clasificadores entrenados con el resto del set, lo usamos para predecir sobre el set anterior.
  - Con esto se entrena un nuevo modelo para que aprenda cómo combinar los resultados de los clasificadores de forma tal que estos nos den la predicción final.
-

# Random Forests

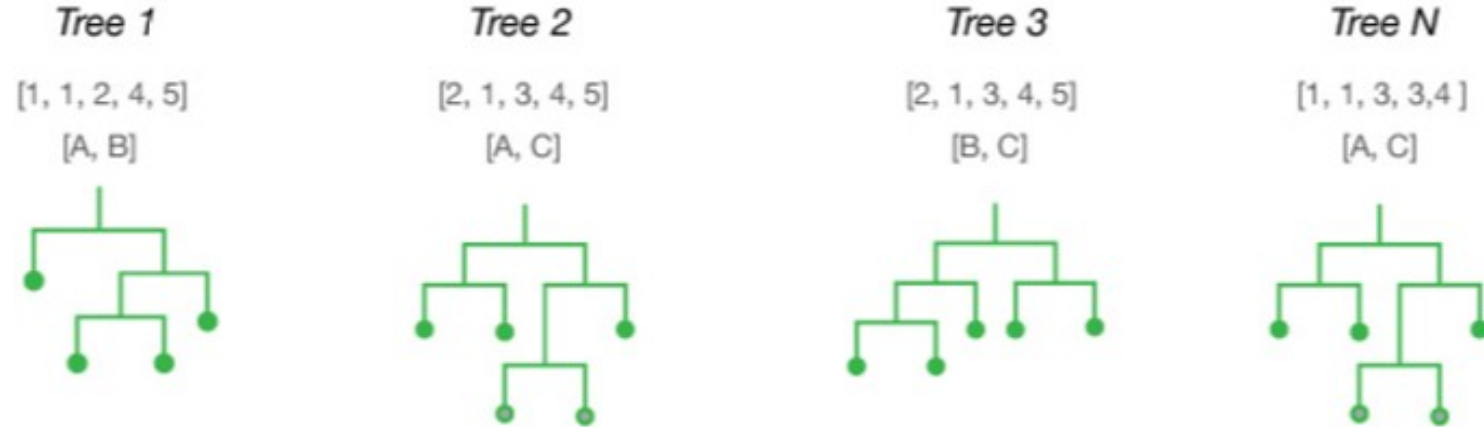
---

- Uno de los algoritmos más populares en clasificación.
    - Buenos resultados para la mayoría de los sets de datos.
  - Bagging sobre árboles de decisión.
  - Cada árbol:
    - Usa un subconjunto de los atributos.
    - Usa un bootstrap del set de entrenamiento.
-



# Random Forests

---



Conjunto de árboles de decisión donde cada uno usa un **bootstrap** del set de entrenamiento y un cierto conjunto de atributos **tomados al azar**.

---

# Random Forests

---

- Hiper-parámetros:
    - Cantidad de árboles a crear.
      - Más árboles mejores resultados, pero trade-off de performance.
    - Cantidad de atributos por árbol.
      - Mas critico.
      - Buscarse mediante grid-search usando OOB (out of bounds) precisión.
-

# Random Forests (Bonus)

---

- Distancia Random Forest:
    - Clasificamos dos puntos con cada árbol.
    - La distancia es el número de árboles en los cuales la predicción de clase son diferentes.
    - Normalizar entre 0 y 1 dividiendo por el total de árboles.
-

# XGBoost

---

- Boosting de árboles de decisión.

# XGBoost - Teoría

---

$$\text{Obj}(\Theta) = L(\Theta) + \Omega(\Theta)$$

$\Theta$ : parámetros a aprender

$L$ : error del modelo

$\Omega$ : factor de regularización

---

# XGBoost - Teoría

---

$$L(\Theta) = \sum_{i=1}^n l(y_i, \hat{y}_i)$$

$l$  es el error en la predicción de  $Y$

Para regresión puede ser  $\rightarrow l = (y_i - \hat{y}_i)^2$

Para clasificación binaria  $\rightarrow l = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$

---

# XGBoost – Teoría

---

La predicción es la sumatoria de la predicción de varios árboles:

$$\hat{y}_i = \sum_{j=1}^k f_j(x_i)$$

# XGBoost - Teoría

---

Cada nuevo arbol intenta corregir los errores del anterior:

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

...

$$\hat{y}_i^{(t)} = \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$



# XGBoost - Teoría

---

$$\text{Obj}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{constant}$$

Reemplazando  $\rightarrow l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$

$$\text{Obj}^{(t)} = \sum_{i=1}^n [2(\hat{y}_i^{(t-1)} - y_i)f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + \text{constant}$$

---

# XGBoost - Teoría

---

Aplicamos la serie de Taylor:

$$\text{Obj}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

Con:

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

$$h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$$

---

# XGBoost - Teoría

---

Complejidad:

$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

T: cantidad de hojas del árbol

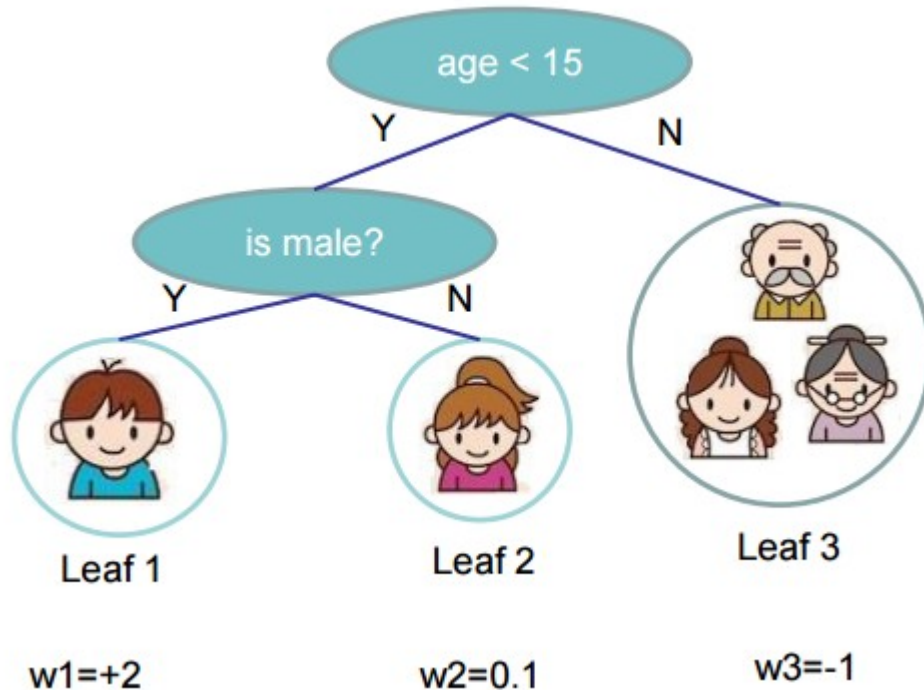
Wj: valor de la j-ésima hoja

Gamma y Lambda son hiper-parámetros

---

# XGBoost - Teoría

---



$$\Omega = \gamma_3 + \frac{1}{2} \lambda (4 + 0.01 + 1)$$

# XGBoost - Teoría

---

Conjunto de instancias de la hoja  $j$ :  $I_j = \{i | q(x_i) = j\}$

$q(x_i)$ : función que mapea instancia  $x_i$  a su nro de hoja

$$\text{Obj}^{(t)} = \sum_{j=1}^T [(\sum_{i \in I_j} g_i)w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda)w_j^2] + \gamma T$$

Suma de  $T$  cuadráticas independientes

---

# XGBoost - Teoría

---

$$G_j = \sum_{i \in I_j} g_i$$

$$H_j = \sum_{i \in I_j} h_i$$

$$\text{Obj}^{(t)} = \sum_{j=1}^T [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T$$

---

# XGBoost - Teoría

---

Peso óptimo de cada hoja:



$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

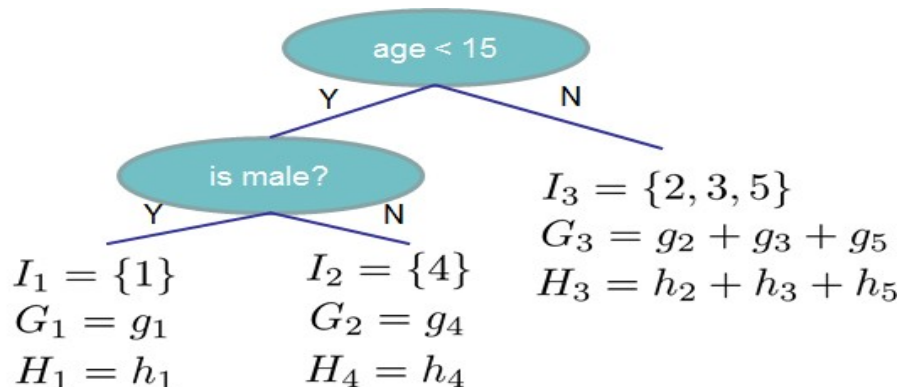
$$\text{Obj} = -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

---

# XGBoost - Teoría

Instance index      gradient statistics

1		$g_1, h_1$
2		$g_2, h_2$
3		$g_3, h_3$
4		$g_4, h_4$
5		$g_5, h_5$



$$Obj = - \sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is



# XGBoost - Teoría

---

Como hay muchas estructuras posibles para el árbol →

XGBoost crece el árbol de forma greedy

$$Gain = \frac{1}{2} \left[ \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

Primer término: puntaje hijo izquierdo.

Segundo término: puntaje hijo derecho.

Tercer término: puntaje en la hoja original.

Gamma: costo de agregar otra hoja.

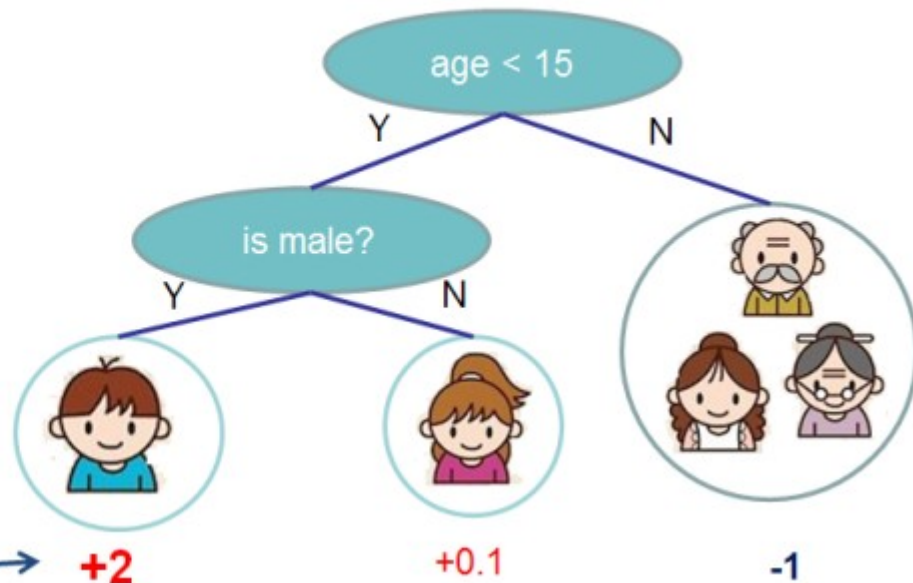
---

# XGBoost

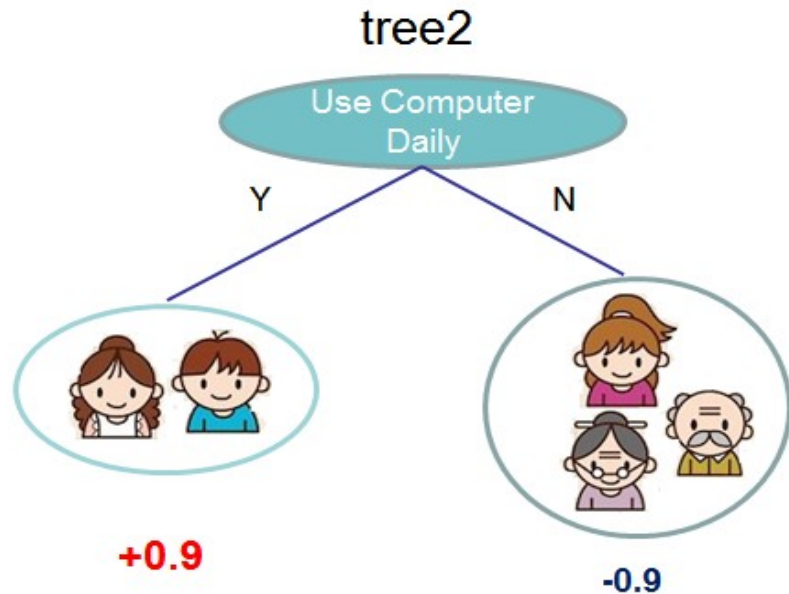
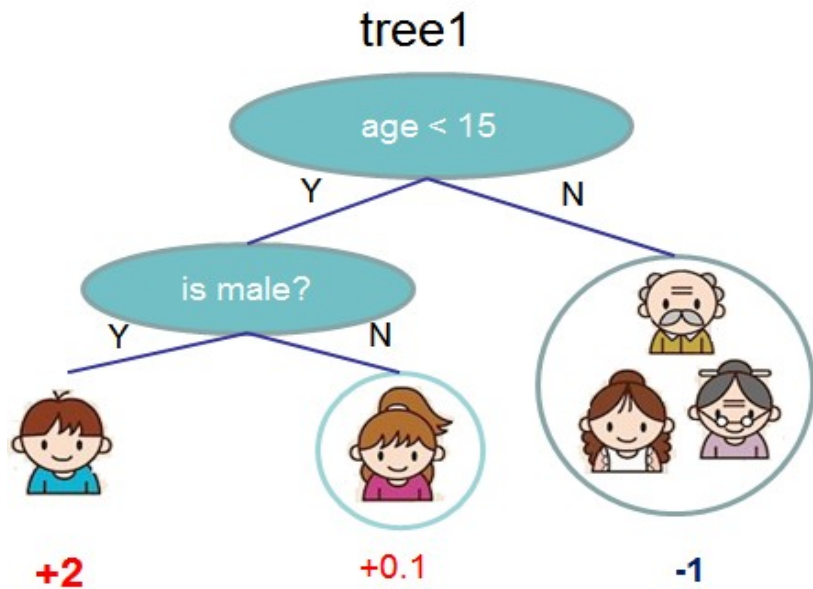
---

Input: age, gender, occupation, ...

Does the person like computer games



# XGBoost



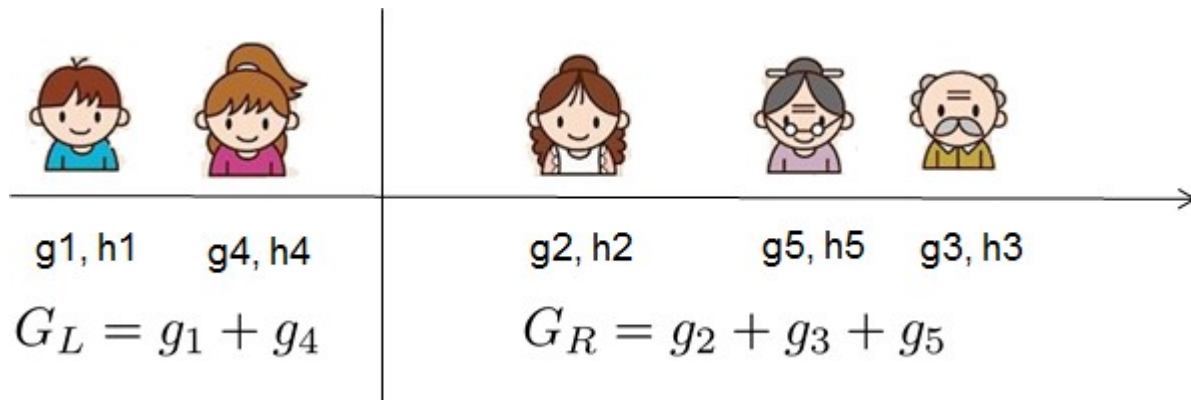
$$f(\text{boy}) = 2 + 0.9 = 2.9$$

$$f(\text{old man}) = -1 - 0.9 = -1.9$$

# XGBoost

---

Para atributos numéricos se ordenan los valores del atributo y se busca el split de mayor ganancia.



Los atributos categóricos deben ser convertidos a variables binarias usando one-hot encoding.