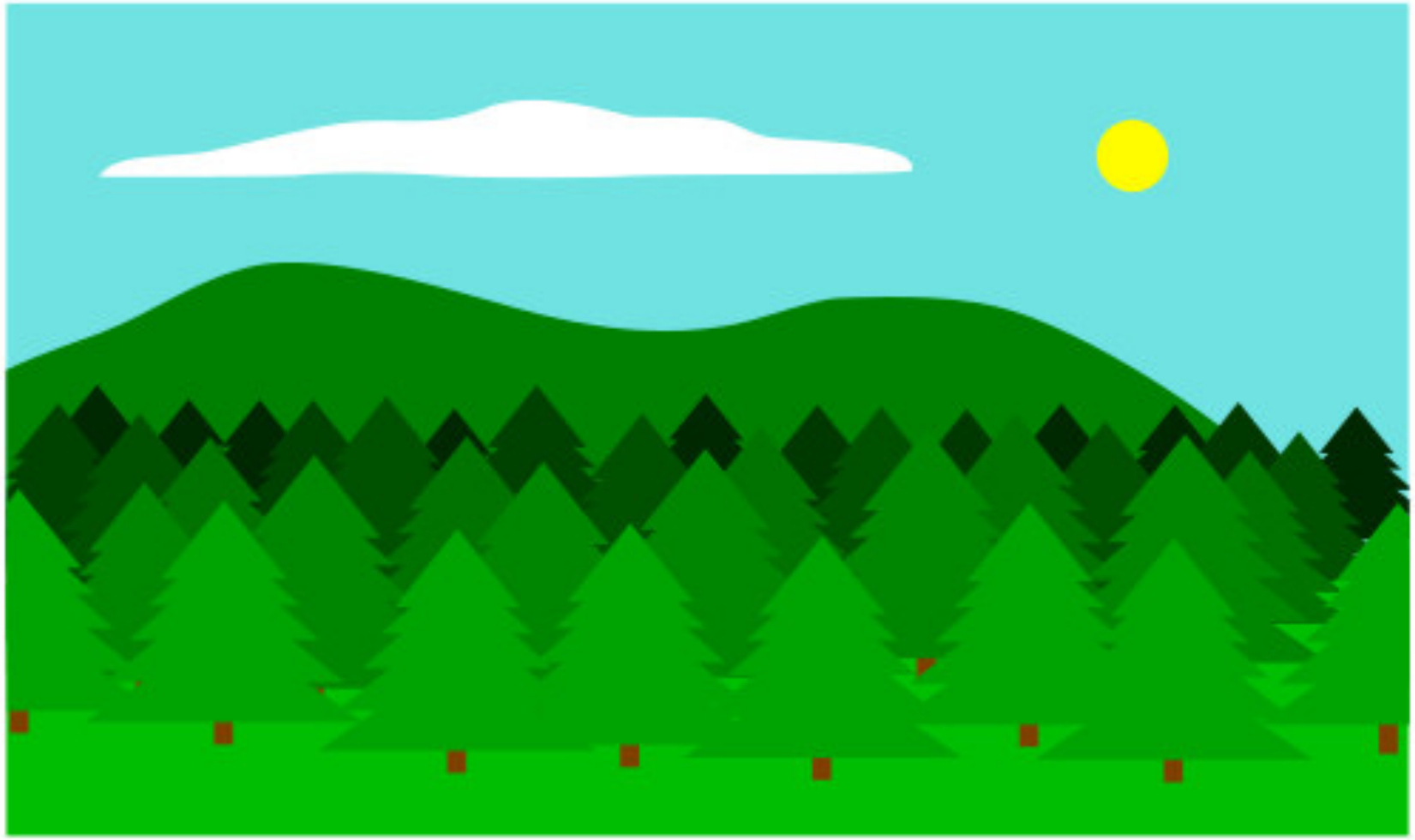
# APRENDIZAJE SUPERVISADO

**Anibal Sosa, PhD**

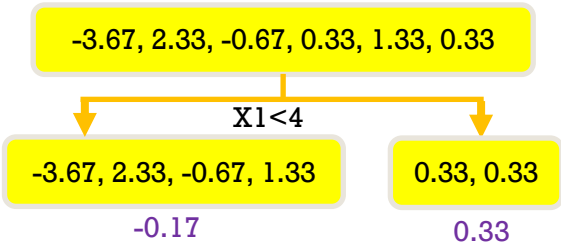
UNIVERSIDAD ICESI

# ENSEMBLE LEARNING



UNIVERSIDAD ICESI

# GRADIENT BOOSTING (REGRESIÓN)




Input: Data $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$, función de pérdida diferenciable $L\big(y_i, F(\boldsymbol{x_i})\big)$.

Para regresión: $L(y_i, F(\boldsymbol{x_i})) = \frac{1}{2}(y_i - F(\boldsymbol{x_i}))^2$

1. Inicializar el modelo con un valor constante $F_0(\boldsymbol{x_i}) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$

   $\sum_{i=1}^{n} -(y_i - \gamma) = 0 \rightarrow \gamma * n = \sum_{i=1}^{n} y_i \rightarrow \gamma = \bar{y}$ , el promedio

2. Para m=1 a M:
   a) Calcular los pseudo residuos de cada instancia con respecto a la predicción anterior (el **gradiente**)

      $r_{im} = -\frac{\partial L(y_i, F_{m-1}(\boldsymbol{x_i}))}{\partial F_{m-1}(\boldsymbol{x_i})} = -\Big(-\big(y_i - F_{m-1}(\boldsymbol{x_i})\big)\Big) = y_i - F_{m-1}(\boldsymbol{x_i})$

   b) Crear un árbol para predecir los $r_{im}$ a partir de las variables independientes y crear sus $J_m$ hojas $R_{jm}$, con $j = 1 \ldots J_m$

   c) Estimar el valor de salida (out) hoja $R_{jm}$, con $j = 1 \ldots J_m$ (análogo al punto 1, pero por hoja)

      $\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma)$

      $\sum_{x_i \in R_{jm}} -(y_i - F_{m-1}(\boldsymbol{x_i}) - \gamma) = 0 \rightarrow \gamma * n = \sum_{x_i \in R_{jm}} \big(y_i - F_{m-1}(\boldsymbol{x_i})\big) \rightarrow \gamma = \overline{r_{im}}$

      → el promedio de los pseudo residuos $r_{im}$ de la hoja en cuestión

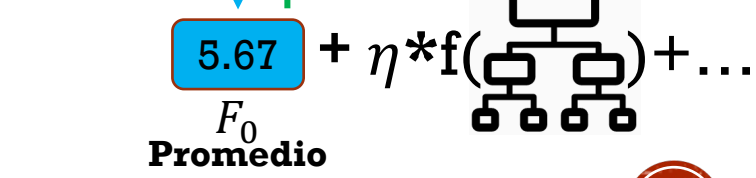
   d) Se actualiza la predicción con incorporando el nuevo modelo con un learning rate $\eta$:

      $F_m = F_{m-1}(x_i) + \eta * r_{im}$

   Se itera

| X1 | X2 | Y | res | out | pred |
|---|---|---|---|---|---|
| 1 | .. | 2 | -3.67 | -0.17 | 5.53 |
| 2 | .. | 8 | 2.33 | -0.17 | 5.53 |
| 3 | .. | 5 | -0.67 | -0.17 | 5.53 |
| 4 | .. | 6 | 0.33 | 0.33 | 5.93 |
| 0 | .. | 7 | 1.33 | -0.17 | 5.53 |
| 5 | .. | 6 | 0.33 | 0.33 | 5.93 |

Con $\eta$=0.8




UNIVERSIDAD ICESI

# GRADIENT BOOSTING (CLASIFICACIÓN)

- Queremos minimizar la función de pérdida (igual a maximizar el log likelihood):

$$
\begin{aligned}
L(y_i, F(x_i)) &= -\left(y_i * log(F(x_i)) + (1 - y_i) * log(1 - F(x_i))\right) \\
&= -\left(y_i * log(p) + (1 - y_i) * log(1 - p)\right) \\
&= -y_i * log(p) - log(1 - p) + y_i * log(1 - p) \\
&= -y_i * \left(log(p) - log(1 - p)\right) - log(1 - p) \\
&= -y_i * log\left(\frac{\boldsymbol{p}}{\boldsymbol{1-p}}\right) - log\left(1 - \frac{e^{log\left(\frac{\boldsymbol{p}}{\boldsymbol{1-p}}\right)}}{1+e^{log\left(\frac{\boldsymbol{p}}{\boldsymbol{1-p}}\right)}}\right) \\
&= -y_i * log(\boldsymbol{odds}) - log\left(1 - \frac{e^{log(\boldsymbol{odds})}}{1+e^{log(\boldsymbol{odds})}}\right) \\
&= -y_i * log(odds) - log\left(\frac{1+e^{log(odds)}}{1+e^{log(odds)}} - \frac{e^{log(odds)}}{1+e^{log(odds)}}\right) \\
&= -y_i * log(odds) - log\left(\frac{1}{1+e^{log(odds)}}\right) \\
&= -y_i * log(odds) + log\left(1 + e^{log(odds)}\right)
\end{aligned}
$$

$$
\begin{aligned}
p &= \frac{e^{log(odds)}}{1+e^{log(odds)}} \\
1 - p &= 1 - \frac{e^{log(odds)}}{1+e^{log(odds)}} \\
&= \frac{1+e^{log(odds)}}{1+e^{log(odds)}} - \frac{e^{log(odds)}}{1+e^{log(odds)}} \\
&= \frac{1}{1+e^{log(odds)}}
\end{aligned}
$$

UNIVERSIDAD ICESI

# GRADIENT BOOSTING (CLASIFICACIÓN)

- Buscamos el gradiente de la función de pérdida

$$L\left(y_i, log(odds)\right) = -y_i * log(odds) + log\left(1 + e^{log(odds)}\right)$$

$$\frac{\partial\left(-y_i*\left(log(odds)\right)+log\left(1+e^{log(odds)}\right)\right)}{\partial(\log(odds))} = -y_i + \frac{1}{1+e^{log(odds)}} * e^{log(odds)}$$

$$= -y_i + \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$= -y_i + p$$

$$p = \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$1 - p = 1 - \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$= \frac{1+e^{log(odds)}}{1+e^{log(odds)}} - \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$= \frac{1}{1+e^{log(odds)}}$$

# GRADIENT BOOSTING (CLASIF.)

Input: Data $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$, función de pérdida diferenciable $L(y_i, F(\boldsymbol{x_i}))$.

$$L(y_i, \log(odds)) = -y_i * log(odds) + log\left(1 + e^{log(odds)}\right)$$

1. Inicializar el modelo con un valor constante

$$F_0(\boldsymbol{x_i}) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$ , se usa el gradiente

$$\sum_{i=1}^{n} -y_i + \frac{1}{1+e^{\gamma}} * e^{\gamma} = 0 \rightarrow \frac{1}{1+e^{\gamma}} * e^{\gamma} * n = \sum_{i=1}^{n} y_i$$

$$\rightarrow \frac{e^{\gamma}}{1+e^{\gamma}} = p \rightarrow \gamma = \log(odds)$$

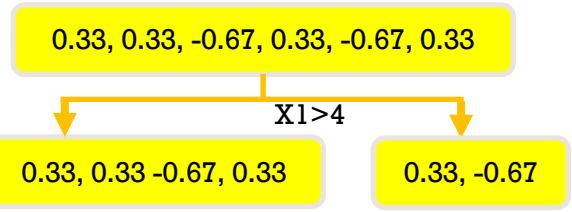
$$F_0(\boldsymbol{x_i}) = \log(odds)$$

2. Para m=1 a M:
   a) Calcular los pseudo residuos de cada instancia con respecto a la predicción anterior (el **gradiente**)

$$r_{im} = -\frac{\partial L(y_i, F_{m-1}(\boldsymbol{x_i}))}{\partial F_{m-1}(\boldsymbol{x_i})} = -(-y_i + p) = y_i - p$$

   b) Crear un árbol para predecir los $r_{im}$ a partir de las variables independientes y crear sus $J_m$ hojas $R_{jm}$, con $j = 1 \dots J_m$
   c) Estimar el valor de salida hoja $R_{jm}$, con $j = 1 \dots J_m$

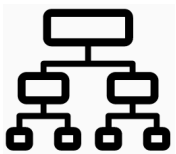
$$\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma)$$




| X1 | X2 | Y | res |
|---|---|---|---|
| 1 | .. | 1 | 0.33 |
| 2 | .. | 1 | 0.33 |
| 3 | .. | 0 | -0.67 |
| 4 | .. | 1 | 0.33 |
| 5 | .. | 0 | -0.67 |
| 6 | .. | 1 | 0.33 |

Proba : $\frac{e^{\log odds}}{1+e^{\log odds}}$ = 0.67

**Log odds:**
**ln(4/2) = 0.69**



UNIVERSIDAD ICESI

# GRADIENT BOOSTING (CLASIFICACIÓN)

- Para obtener el valor de salida de cada hoja necesitamos derivar $L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma)$

$$L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma) = -y_i * log(F_{m-1}(\boldsymbol{x_i}) + \gamma) + log\left(1 + e^{F_{m-1}(\boldsymbol{x_i}) + \gamma}\right)$$

Se aproxima la función de pérdida con un polinomio de Taylor de orden 2:

$$L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma)$$
$$\approx L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right) + \frac{\partial}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right) * \gamma + \frac{\mathbf{1}}{\mathbf{2}} \frac{\partial^2}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right) * \gamma^2$$

Y ahora si se deriva con respecto a $\gamma$, para optimizarlo:

$$\frac{\partial}{\partial \gamma} L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma) \approx \frac{\partial}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right) + \frac{\partial^2}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right) * \gamma = 0$$

$$\gamma = \frac{-\frac{\partial}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right)}{\frac{\partial^2}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right)} = \frac{y_i - p}{\frac{\partial^2}{\partial F_{m-1}(\boldsymbol{x_i})} L\left(y_i, F_{m-1}(\boldsymbol{x_i})\right)}$$

UNIVERSIDAD ICESI

# GRADIENT BOOSTING (CLASIFICACIÓN)

- Necesitamos obtener la derivada segunda de la función de pérdida

$$\frac{\partial^2}{\partial \log(odds)^2} L(y_i, \log(odds)) = \frac{\partial}{\partial \log(odds)} \left( -y_i + \frac{1}{1+e^{log(odds)}} * e^{log(odds)} \right)$$

$$= \frac{-e^{log(odds)}}{\left(1+e^{log(odds)}\right)^2} * e^{log(odds)} + \frac{1}{1+e^{log(odds)}} * e^{log(odds)}$$

$$= \frac{-e^{2*log(odds)}}{\left(1+e^{log(odds)}\right)^2} + \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$= \frac{-e^{2*log(odds)}}{\left(1+e^{log(odds)}\right)^2} + \frac{e^{log(odds)}+e^{2*log(odds)}}{\left(1+e^{log(odds)}\right)^2}$$

$$= \frac{e^{log(odds)}}{\left(1+e^{log(odds)}\right)^2} = \frac{e^{log(odds)}}{1+e^{log(odds)}} * \frac{1}{1+e^{log(odds)}} = p * (1-p)$$

$$p = \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$1 - p = 1 - \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$= \frac{1+e^{log(odds)}}{1+e^{log(odds)}} - \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

$$= \frac{1}{1+e^{log(odds)}}$$

- Por lo tanto tenemos, cuando consideramos todas las instancias de un nodo

$\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma)$, que se logra cuando $\gamma = \frac{y_i - p}{\frac{\partial^2}{\partial F_{m-1}(\boldsymbol{x_i})} L(y_i, F_{m-1}(\boldsymbol{x_i}))} = \frac{y_i - p}{p*(1-p)}$

$$\gamma_{jm} = \frac{\sum_{i \in R_{jm}} (y_i - p)}{\sum_{i \in R_{jm}} p(1-p)}$$

UNIVERSIDAD ICESI

# GRADIENT BOOSTING (CLASIF.)

Input: Data $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$, función de pérdida diferenciable $L(y_i, F(\boldsymbol{x_i}))$.

$$L(y_i, \log(odds)) = -y_i * log(odds) + log\left(1 + e^{log(odds)}\right)$$

1. Inicializar el modelo con un valor constante

$$F_0(\boldsymbol{x_i}) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$ , se usa el gradiente

$$\sum_{i=1}^{n} -y_i + \frac{1}{1+e^{\gamma}} * e^{\gamma} = 0 \rightarrow \frac{1}{1+e^{\gamma}} * e^{\gamma} * n = \sum_{i=1}^{n} y_i$$

$$\rightarrow \frac{e^{\gamma}}{1+e^{\gamma}} = p \rightarrow \gamma = \log(odds)$$

$$F_0(\boldsymbol{x_i}) = \log(odds)$$

2. Para m=1 a M:
   a) Calcular los pseudo residuos de cada instancia con respecto a la predicción anterior (el **gradiente**)

$$r_{im} = -\frac{\partial L(y_i, F_{m-1}(\boldsymbol{x_i}))}{\partial F_{m-1}(\boldsymbol{x_i})} = -(-y_i + p) = y_i - p$$

   b) Crear un árbol para predecir los $r_{im}$ a partir de las variables independientes y crear sus $J_m$ hojas $R_{jm}$, con $j = 1 \ldots J_m$
   c) Estimar el valor de salida hoja $R_{jm}$, con $j = 1 \ldots J_m$

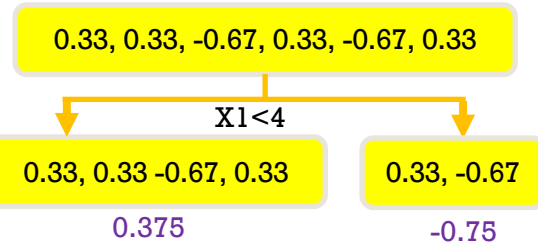
$$\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(\boldsymbol{x_i}) + \gamma) = \frac{\sum y_i - p}{\sum p*(1-p)}$$

$$\frac{0.33 - 0.67}{0.67 * 0.33 + 0.67 * 0.33} = -0.75$$

   d) Se actualiza la predicción con incorporando el nuevo modelo con un learning rate $\eta$:

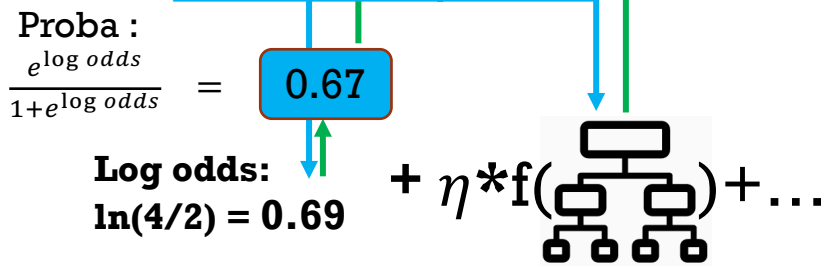
$$F_m = F_{m-1}(x_i) + \eta * r_{im}$$

Se itera




| X1 | X2 | Y | res | out | pred |
|---|---|---|---|---|---|
| 1 | .. | 1 | 0.33 | 0.375 | 0.967 |
| 2 | .. | 1 | 0.33 | 0.375 | 0.967 |
| 3 | .. | 0 | -0.67 | 0.375 | 0.967 |
| 4 | .. | 1 | 0.33 | 0.375 | 0.967 |
| 5 | .. | 0 | -0.67 | -0.75 | 0.067 |
| 6 | .. | 1 | 0.33 | -0.75 | 0.067 |

Con $\eta$=0.8




UNIVERSIDAD ICESI

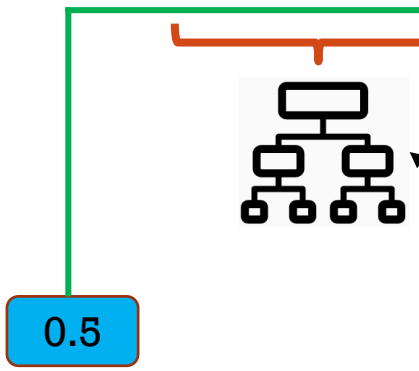
# XGBOOST (REGRESIÓN)

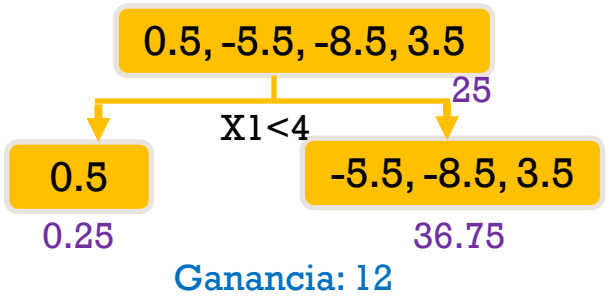
- Algoritmo de árboles de tipo XGBoost:

1. Se establecen los residuos con respecto a la predicción anterior (por defecto se empieza prediciendo un valor de **0.5**)
2. Calcular los pseudo residuos de cada instancia con respecto a la predicción anterior.
3. Para cada nodo evaluar las particiones posibles
4. Calcular la similitud de las instancias de cada nodo: $Sim(nodo) = \frac{(\sum_{i \in nodo} residuo_i)^2}{cardinalidad(hoja)+\lambda}$, con $\lambda$ que controla la regularización (0 para este ejemplo)
5. Calcular la ganancia de cada partición: $Ganancia(árbol) = Sim(izq) + Sim(der) - Sim(raiz)$
6. Se escoge el particionamiento de mayor ganancia
7. Se itera con los siguientes particionamientos de cada hoja

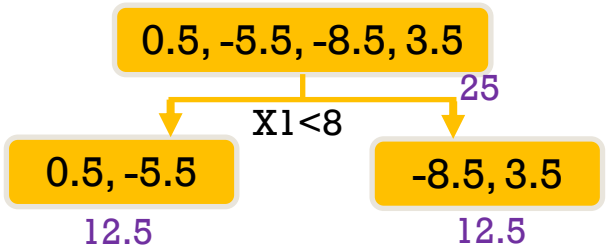
| X1 | X2 | Y |
|---|---|---|
| 13 | .. | 4 |
| 9 | .. | -8 |
| 7 | .. | -5 |
| 1 | .. | 1 |

| X1 | X2 | res |
|---|---|---|
| 13 | .. | 3.5 |
| 9 | .. | -8.5 |
| 7 | .. | -5.5 |
| 1 | .. | 0.5 |

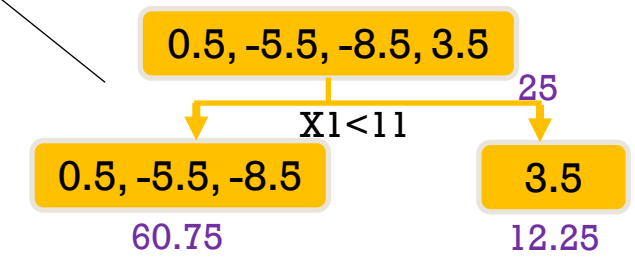



Ganancia: 12




Ganancia: 0




Ganancia: 48

UNIVERSIDAD ICESI

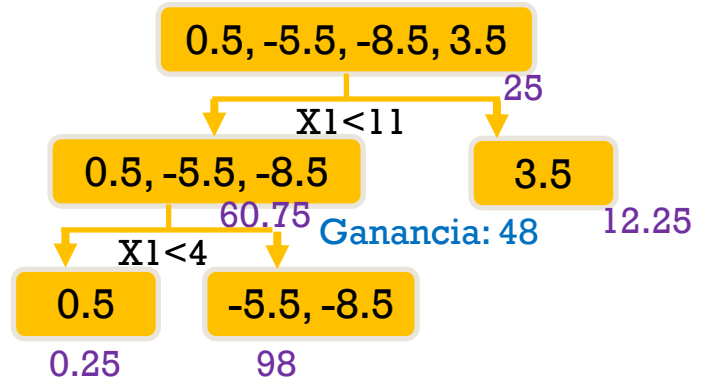
# XGBOOST (REGRESIÓN)

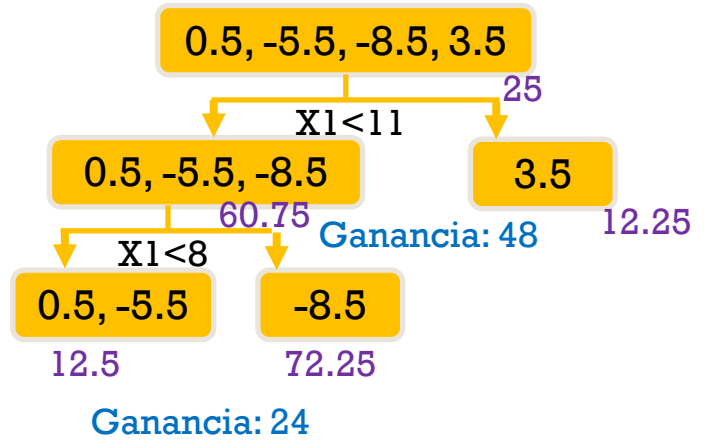
- Algoritmo de árboles de tipo XGBoost:

7. Se itera con los siguientes particionamientos recursivamente en cada hoja hasta un criterio de parada (e.g. **max depth**=2, es este caso)
8. Una vez desarrollado el árbol, se poda (post-pruning), de las hojas hacia la raíz, teniendo en cuenta un umbral mínimo $\gamma$ (**gamma**) de ganancia para particionar un nodo.

| X1 | X2 | Y |
|---|---|---|
| 13 | .. | 4 |
| 9 | .. | -8 |
| 7 | .. | -5 |
| 1 | .. | 1 |

| X1 | X2 | res |
|---|---|---|
| 13 | .. | 3.5 |
| 9 | .. | -8.5 |
| 7 | .. | -5.5 |
| 1 | .. | 0.5 |


$$Sim(nodo) = \frac{(\sum_{i \in nodo} residuo_i)^2}{cardinalidad(hoja) + \lambda}$$

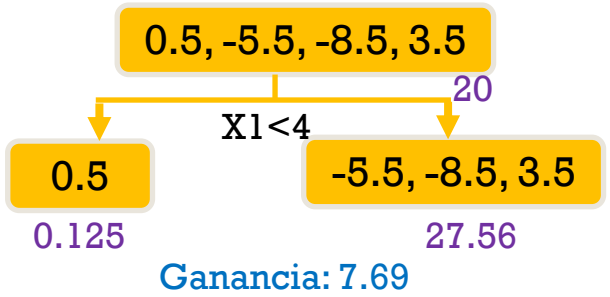
- Si $\gamma$ =30, cómo sería el árbol?
- Si $\gamma$ =40, cómo sería el árbol?
- Si $\gamma$ =30 y $\lambda$ =1, cómo sería el árbol?

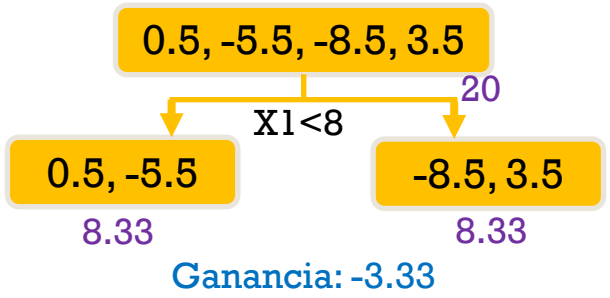
UNIVERSIDAD ICESI

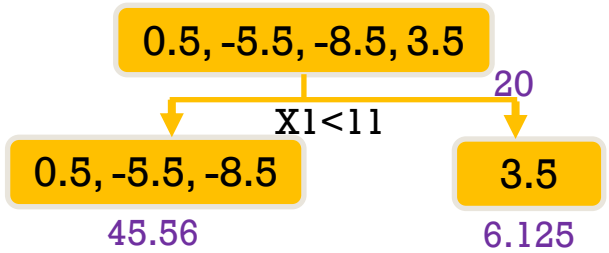
# XGBOOST

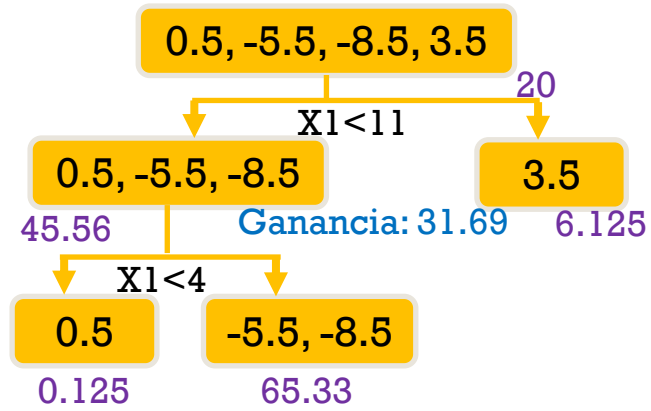
$$Sim(nodo) = \frac{(\sum_{i \in nodo} residuo_i)^2}{cardinalidad(hoja) + \lambda}$$

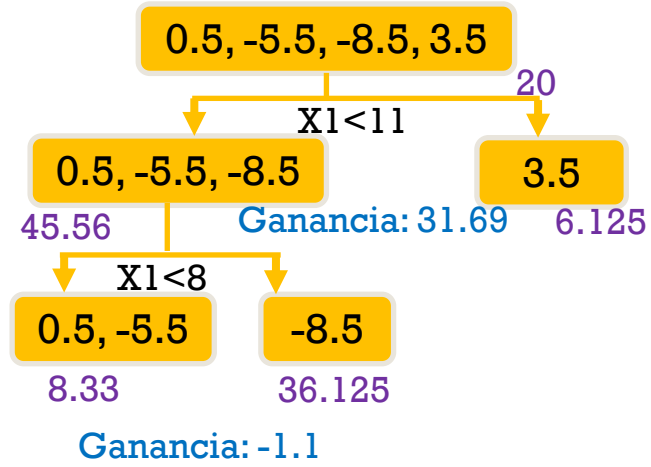
- Si $\gamma$ =30 y $\lambda$ =1,
  ¿Cómo sería el árbol?




UNIVERSIDAD ICESI

# XGBOOST (REGRESIÓN)

- Algoritmo de árboles de tipo XGBoost:

7. Se itera con los siguientes particionamientos recursivamente en cada hoja hasta un criterio de parada (e.g. **max depth**=2, es este caso)
8. Una vez desarrollado el árbol, se poda (post-pruning), de las hojas hacia la raíz, teniendo en cuenta un umbral mínimo $\gamma$ (**gamma**) de ganancia para particionar un nodo.
9. Calculamos los valores predichos por el árbol nodo por nodo:

$$Pred(nodo) = \frac{\sum_{i \in nodo} residuo_i}{cardinalidad(hoja) + \lambda}$$

10. Combinamos el nuevo árbol usando un único **learning rate**, (llamado **eta**, por defecto 0.3)
11. Se crean nuevos árboles hasta llegar a un criterio de parada (e.g. # árboles)

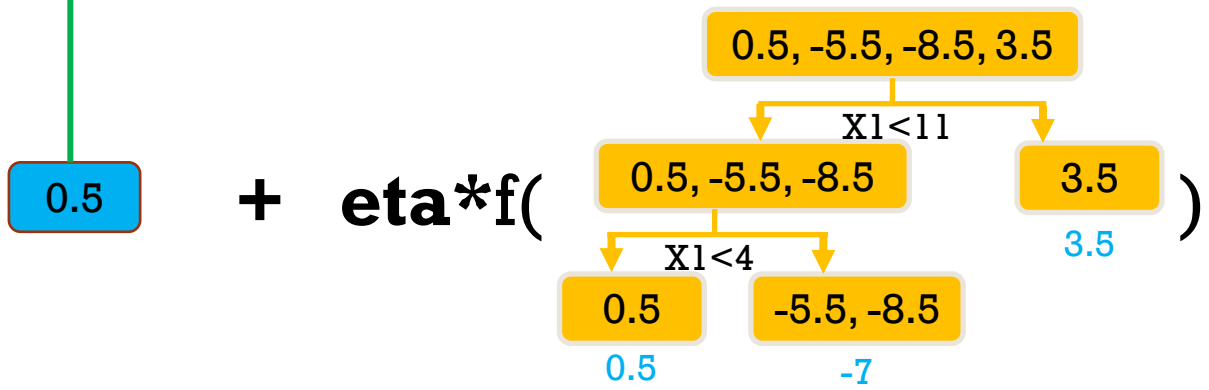
| X1 | X2 | Y |
|---|---|---|
| 13 | .. | 4 |
| 9 | .. | -8 |
| 7 | .. | -5 |
| 1 | .. | 1 |

| X1 | X2 | res |
|---|---|---|
| 13 | .. | 3.5 |
| 9 | .. | -8.5 |
| 7 | .. | -5.5 |
| 1 | .. | 0.5 |

| PRED | res |
|---|---|
| 1.55 | 2.45 |
| -1.6 | -6.4 |
| -1.6 | -3.4 |
| 0.65 | 0.35 |

...

Con $\eta$=0.3




UNIVERSIDAD ICESI

# XGBOOST (REGRESIÓN)

- Para regresión se usa la función de pérdida $L(y_i, F(\boldsymbol{x_i})) = \frac{1}{2}\left(y_i - F(\boldsymbol{x_i})\right)^2$
  - Su gradiente **g** (primera derivada) es: $\boldsymbol{g} = \frac{\partial L(y_i, F(\boldsymbol{x_i}))}{\partial F(\boldsymbol{x_i})} = -\left(y_i - F(\boldsymbol{x_i})\right)$, es decir el residuo negativo
  - Su hessiano **h** (segunda derivada) es: $\boldsymbol{h} = \frac{\partial^2 L(y_i, F(\boldsymbol{x_i}))}{\partial F(\boldsymbol{x_i})^2} = \frac{\partial -\left(y_i - F(\boldsymbol{x_i})\right)}{\partial F(\boldsymbol{x_i})} = 1$
- Para cada hoja buscamos la salida $O$ que minimice $\left(\sum_{i=1}^{n} L(y_i, F_{m-1}(\boldsymbol{x_i}) + O)\right) + \frac{1}{2}\boldsymbol{\lambda} * O^2$, con $\boldsymbol{\lambda}$ para regularización
  - Usamos aproximación de Taylor de orden 2.

$$\sum_{i=1}^{n} L(y_i, F_{m-1}(\boldsymbol{x_i}) + O) = \sum_{i=1}^{n} L(y_i, F(\boldsymbol{x_i})) + \sum_{i=1}^{n} \frac{\partial L(y_i, F(\boldsymbol{x_i}))}{\partial F(\boldsymbol{x_i})} * O + \frac{1}{2} * \sum_{i=1}^{n} \frac{\partial^2 L(y_i, F(\boldsymbol{x_i}))}{\partial F(\boldsymbol{x_i})^2} * O^2$$

  - Vamos a minimizar con respecto a $O$

$$\sum_{i=1}^{n} L(y_i, F(\boldsymbol{x_i})) + \sum_{i=1}^{n} \boldsymbol{g} * O + \frac{1}{2} * \sum_{i=1}^{n} \boldsymbol{h} * O^2 + \frac{1}{2}\boldsymbol{\lambda} * O^2$$

$$\frac{\partial \sum_{i=1}^{n} L(y_i, F(\boldsymbol{x_i})) + \sum_{i=1}^{n} \boldsymbol{g} * O + \frac{1}{2} * \sum_{i=1}^{n} \boldsymbol{h} * O^2 + \frac{1}{2}\boldsymbol{\lambda} * O^2}{\partial O} = 0$$

$$\sum_{i=1}^{n} \boldsymbol{g} + \sum_{i=1}^{n} \boldsymbol{h} * O + \boldsymbol{\lambda} * O = 0$$

$$O = \frac{-\sum_{i=1}^{n} \boldsymbol{g}}{\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}} = \frac{suma\ de\ los\ residuos\ de\ la\ hoja}{cardinalidad\ de\ la\ hoja + \boldsymbol{\lambda}}$$

UNIVERSIDAD ICESI

# XGBOOST (REGRESIÓN)

- Vamos ahora a remplazar el output en la aproximación de Taylor, considerando solo los términos que varían con respecto a $O$. El cambio de signo cambia el problema de minimizar a maximizar

$$-\sum_{i=1}^{n} \boldsymbol{g} * O - \frac{1}{2} * \sum_{i=1}^{n} \boldsymbol{h} * O \quad - \frac{1}{2}\boldsymbol{\lambda} * O^2 = -\sum_{i=1}^{n} \boldsymbol{g} * \frac{-\sum_{i=1}^{n} \boldsymbol{g}}{\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}} - \frac{1}{2} * \left(\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}\right) * \left(\frac{-\sum_{i=1}^{n} \boldsymbol{g}}{\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}}\right)^2$$

$$= \frac{(\sum_{i=1}^{n} \boldsymbol{g})^2}{\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}} - \frac{1}{2} * \frac{(\sum_{i=1}^{n} \boldsymbol{g})^2}{\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}}$$

$$= \frac{1}{2} * \frac{(\sum_{i=1}^{n} \boldsymbol{g})^2}{\sum_{i=1}^{n} \boldsymbol{h} + \boldsymbol{\lambda}}$$

  XGBoost toma el doble de este valor para simplificar (ya que se trata de una medida relativa) como la métrica de similitud usada para particionar los nodos del árbol

- En el caso de regresión tenemos entonces la medida de similitud de los nodos

$$Sim(nodo) = \frac{(\sum_{i \in nodo} residuo_i)^2}{cardinalidad(hoja) + \boldsymbol{\lambda}}$$

- Se define la cobertura de una hoja como $\sum_{i=1}^{n} \boldsymbol{h}$. En el caso de regresión, se trata del número de instancias en una hoja. XGBoost define el parámetro de pre poda **min child weight** como la mínima cobertura posible para permitir el particionamiento de un nodo. Parar el particionamiento cuando los nodos sean muy pequeños.

UNIVERSIDAD ICESI

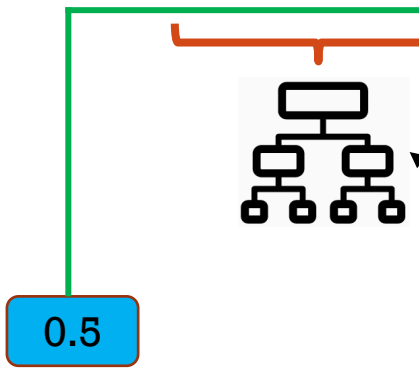
# XGBOOST (CLASIFICACIÓN)

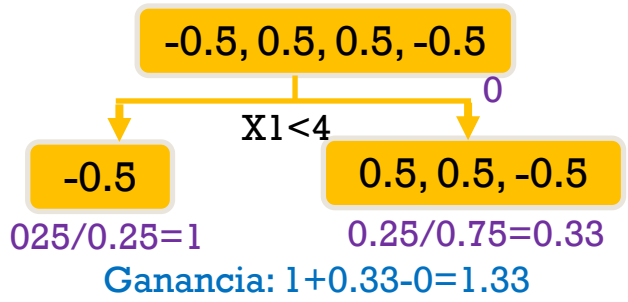
- Algoritmo de árboles de tipo XGBoost:

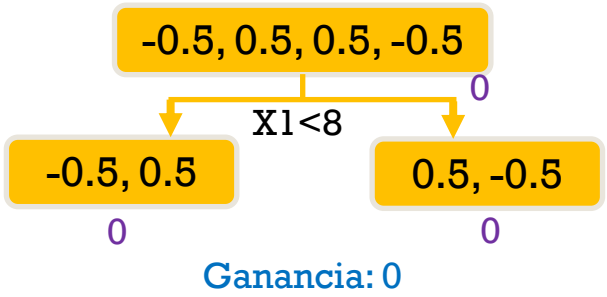
1. Se establecen los residuos con respecto a la predicción anterior (por defecto se empieza prediciendo un valor de **0.5 de probabilidad**)
2. Calcular los pseudo residuos de cada instancia con respecto a la predicción anterior.
3. Para cada nodo evaluar las particiones posibles
4. Calcular la similitud de las instancias de cada nodo: $Sim(nodo) = \frac{(\sum_{i \in nodo} residuo_i)^2}{\sum_{i \in nodo} p_i * (1-p_i) + \lambda}$, con $p_i$ la probabilidad anterior del nodo $i$, y $\lambda$ que controla la regularización (0 para este ejemplo)
5. Calcular la ganancia de cada partición: $Ganancia(árbol) = Sim(izq) + Sim(der) - Sim(raiz)$
6. Se escoge el particionamiento de mayor ganancia
7. Se itera con los siguientes particionamientos de cada hoja

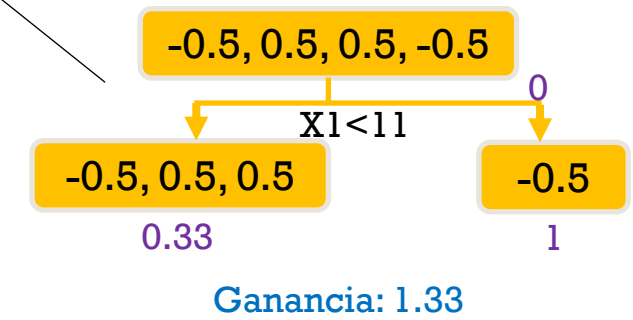
| X1 | X2 | Y |
|---|---|---|
| 13 | .. | 0 |
| 9 | .. | 1 |
| 7 | .. | 1 |
| 1 | .. | 0 |

| X1 | X2 | res |
|---|---|---|
| 13 | .. | -0.5 |
| 9 | .. | 0.5 |
| 7 | .. | 0.5 |
| 1 | .. | -0.5 |




UNIVERSIDAD ICESI

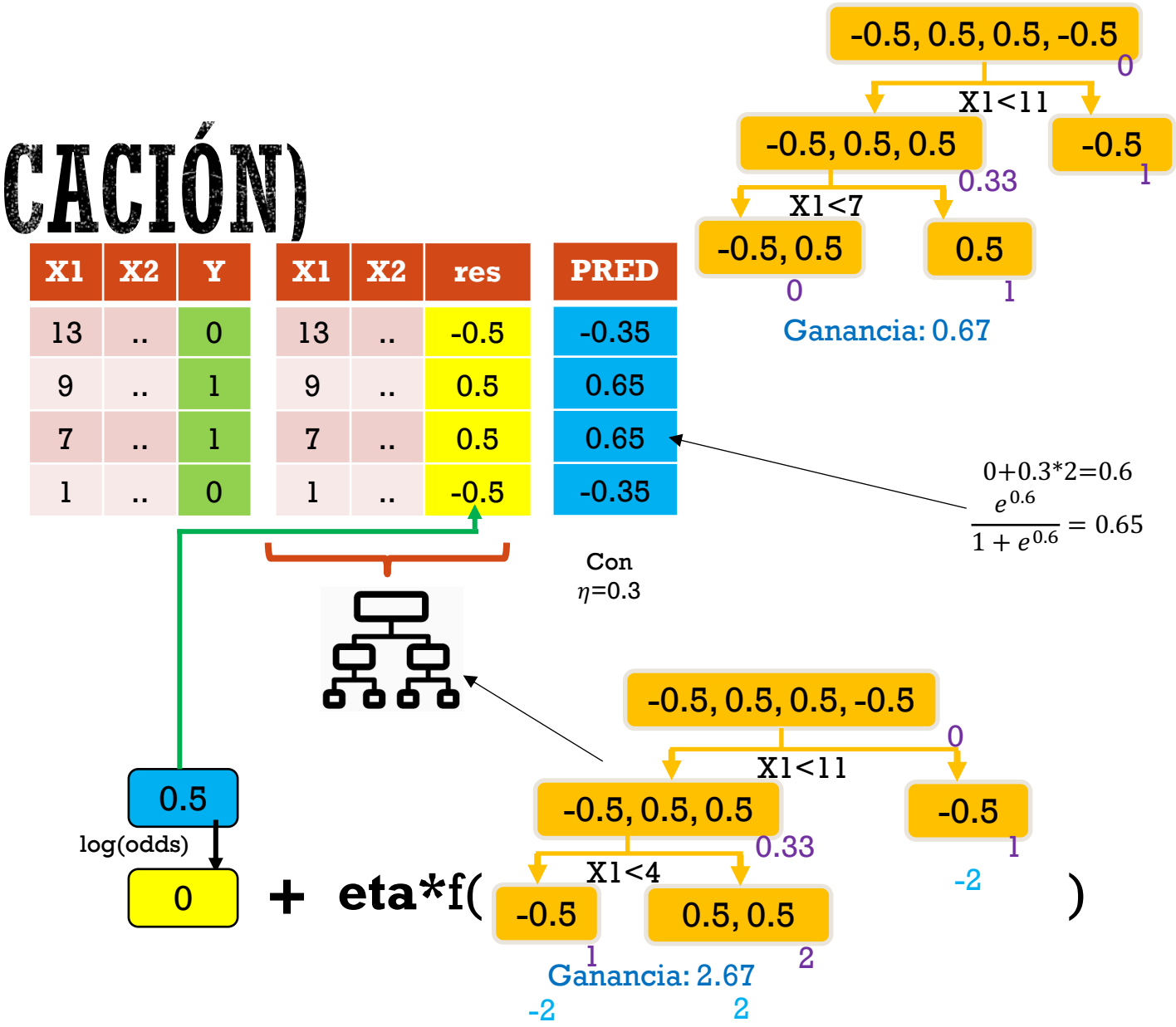
# XGBOOST (CLASIFICACIÓN)

- Algoritmo de árboles de tipo XGBoost:

7. Se itera con los siguientes particionamientos recursivamente en cada hoja hasta un criterio de parada (e.g. **max depth**=2, es este caso)
8. Una vez desarrollado el árbol, se poda (post-pruning), de las hojas hacia la raíz, teniendo en cuenta un umbral mínimo $\gamma$ (**gamma**) de ganancia para particionar un nodo.
9. Calculamos los valores predichos por el árbol nodo por nodo:

$$Pred(nodo) = \frac{\sum_{i \in nodo} residuo_i}{\sum_{i \in nodo} p_i * (1 - p_i) + \lambda}$$

10. Combinamos el nuevo árbol usando un único **learning rate**, (llamado **eta**, por defecto 0.3). Debemos hacer la combinación sobre los log(odds), y luego transformarlos log(odds) a probabilidades
11. Se crean nuevos árboles hasta llegar a un criterio de paro (e.g. # árboles)




UNIVERSIDAD ICESI

# XGBOOST (CLASIFICACIÓN)

- Para clasificación se usa la función de pérdida

$$L(y_i, F(x_i)) = -\big(y_i * log(F(x_i)) + (1 - y_i) * log(1 - F(x_i))\big)$$
$$L\big(y_i, log(odds)\big) = -y_i * log(odds) + log\big(1 + e^{log(odds)}\big)$$

- Su gradiente **g** (primera derivada) es:

$$g = \frac{\partial L(y_i, \log(odds))}{\partial \log(odds)} = -y_i + \frac{e^{\log(odds)}}{1+e^{\log(odds)}} = -(y_i - p_i)$$

- Su hessiano **h** (segunda derivada) es:

$$h = \frac{\partial^2 L(y_i, \log(odds))}{\partial \log(odds)^2} = \frac{e^{\log(odds)}}{1+e^{\log(odds)}} * \frac{1}{1+e^{\log(odds)}} = p_i * (1 - p_i)$$

UNIVERSIDAD ICESI

# XGBOOST (CLASIFICACIÓN)

- Para cada hoja buscamos la salida $O$ que minimice $(\sum_{i=1}^{n} L(y_i, F_{m-1}(\boldsymbol{x_i}) + O)) + \frac{1}{2}\lambda * O^2$, con $\lambda$ para regularización

  Usamos aproximación de Taylor de orden 2, como ya fue ilustrado en el caso de regresión:

$$O = \frac{-\sum_{i=1}^{n} \boldsymbol{g}}{\sum_{i=1}^{n} \boldsymbol{h} + \lambda} = \frac{\sum_{i=1}^{n}(y_i - p_i)}{\sum_{i=1}^{n} p_i * (1 - p_i) + \lambda} = \frac{suma\ de\ los\ residuos\ de\ la\ hoja}{suma\ de\ proba\ anterior * (1 - proba\ anterior) + \lambda}$$

- En el caso de clasificación tenemos entonces la medida de similitud de los nodos

$$Sim(nodo) = \frac{(\sum_{i=1}^{n} \boldsymbol{g})^2}{\sum_{i=1}^{n} \boldsymbol{h} + \lambda} = \frac{(suma\ de\ los\ residuos\ de\ la\ hoja)^2}{suma\ de\ proba\ anterior * (1 - proba\ anterior) + \lambda}$$

- Se define la cobertura de una hoja como $\sum_{i=1}^{n} \boldsymbol{h}$. En el caso de clasificación, la cobertura es $\sum_{i=1}^{n} p_i * (1 - p_i)$. El parámetro de pre poda **min child weight** se utilizaría para parar el particionamiento de los árboles cuando se llega a un grado de pureza dado.



# XGBOOST

Control del overfitting a partir de varios parámetros:

- $\lambda$ (**lambda**): regularización L2 con respecto a la similitud y al output de los árboles

  → A mayor $\lambda$, menor overfitting porque se reducen las ganancias y actualizaciones de las predicciones

- $\gamma$ (**gamma**): parámetro de post-poda, controla el tamaño de los árboles.

  → A mayor $\gamma$, más difícil particionar un nodo. Así $\gamma$ sea 0, se impiden particionamientos con ganancia negativa.

- **max depth:** parámetro de pre poda, limita el desarrollo de los árboles (6, por defecto).

  → A menor **max depth**, menor el overfitting

- **min child weight:** parámetro de pre poda, limita el desarrollo de los árboles.
  **OJO:** Por defecto es 1, lo cual no afecta modelos de regresión, pero sí de clasificación.

  → A mayor **min child weigth**, menor el overfitting



# XGBOOST

Control del overfitting a partir de varios parámetros:

- $\eta$ (**eta**): learning rate que controla la velocidad de aprendizaje

  → A menor $\eta$, menor overfitting, pero se necesitarán más iteraciones

- $\alpha$ (**alpha**): regularización L1

  → A mayor $\alpha$, menor overfitting

- **subsample**: fracción de la muestra considerada para creación de cada árbol (1 por defecto)

  → A menor valor, menor overfitting (Principio de Bagging)

- **colsample_bytree**: fracción de features considerados para creación de las ramas (1 por defecto)

  → A menor valor, menor overfitting (Principio de Random Subspaces)

- **scale_pos_weight**: ayuda a controlar el desbalanceo (1 por defecto), se multiplica al gradiente de la clase minoritaria. Si se exagera en su valor, puede llevar al overfitting



# XGBOOST

- Optimización para grandes volúmenes de datos:
  - No se consideran todos los posibles particionamientos, se utilizan solo los **cuantiles ponderados** (cada dato tiene un peso asociado a la confianza de su predicción – el hessiano **h**)
  - Un algoritmo distribuido aproximadamente greedy ("sketch") divide los datos y calcula una aproximación de los cuantiles de manera **paralela**, para acelerar el entrenamiento
  - Hay 33 cuantiles por defecto (parámetro **sketch_eps**)
- Permite el manejo de **valores faltantes (VFs)**:
  - Se crean las particiones sin incluir los VFs se calculan 2 ganancias, una con los VFs en la rama izquierda, otra con los VFs en la rama derecha
  - Se escoge la partición con mayor ganancia, asociando los VFs a la rama correspondiente



# XGBOOST

- Permite hacer **Cross-Validation** en cada iteración
- Permite **retomar el entreno** de un modelo después de haber parado el entrenamiento
- Consideración del **hardware** disponible para el cómputo:
  - Utiliza **memoria cache** para almacenar los gradientes y hessianos para optimizar la velocidad del cálculo de similitudes, ganancias y valores predichos
  - En casos de volúmenes de datos grandes (no caben en RAM) organiza y comprime pedazos del dataset de manera óptima, sirviéndose de los discos duros disponibles (**out of core**).
- Permite el manejo de **valores faltantes (VFs)**:
  - Se crean las particiones sin incluir los VFs se calculan 2 ganancias, una con los VFs en la rama izquierda, otra con los VFs en la rama derecha
  - Se escoge la partición con mayor ganancia, asociando los VFs a la rama correspondiente