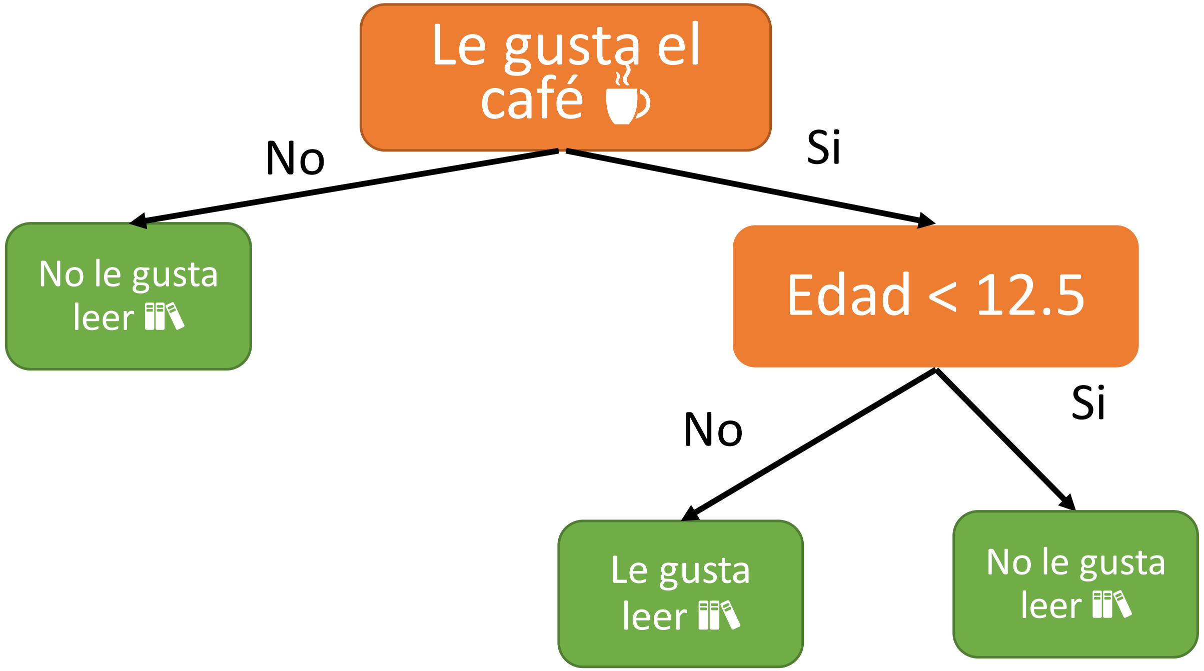
Árboles de Decisión y Métodos de Ensemble

Árboles de Decisión

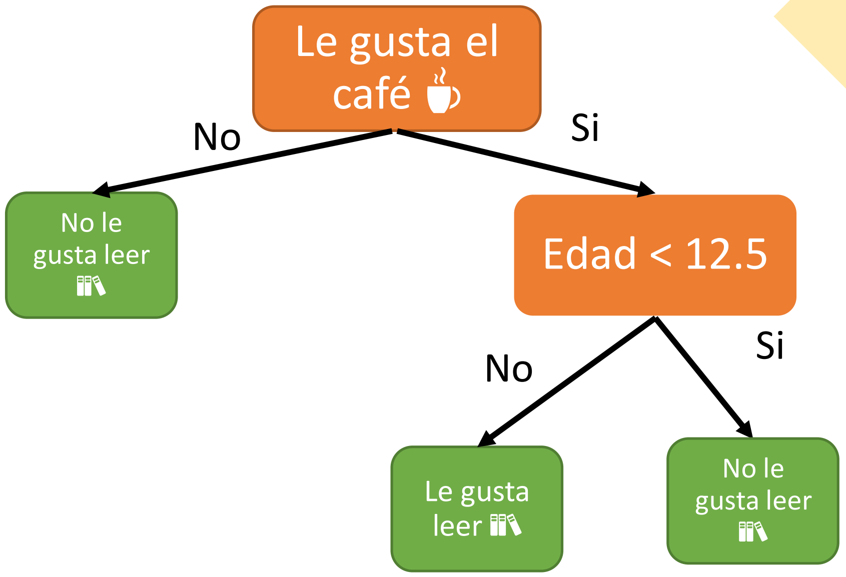
Árboles de Decisión

- Quizá uno de los algoritmos de ML más fáciles de entender
- Individualmente no es muy potente, pero es intuitivo
- Similar a lo que nosotros hacemos cuando decidimos algo



¿Sobre cuál variable debería decidir primero?

Le gusta la TV	Le gusta el café	Edad	Le gusta leer
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**



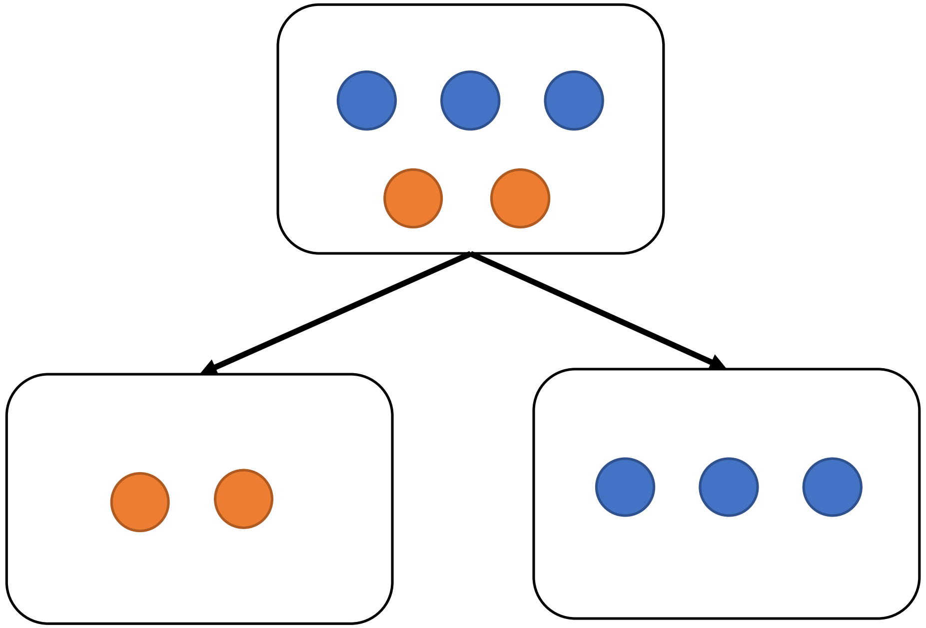
Pureza e información

Tenemos que decidir cuál es el feature que nos va a separar la data con mayor diferencia

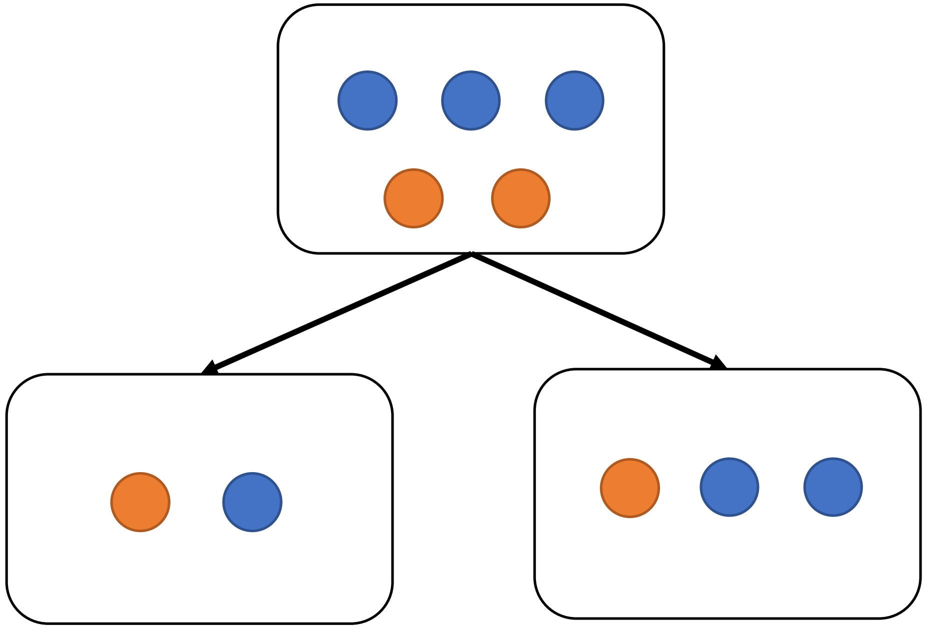
El que nos dé mayor **pureza** de clases

Con el que ganemos más **información, menor entropía**

Puro



Impuro



Por lo general se desea que la división sea lo más pura posible

Cómo medir

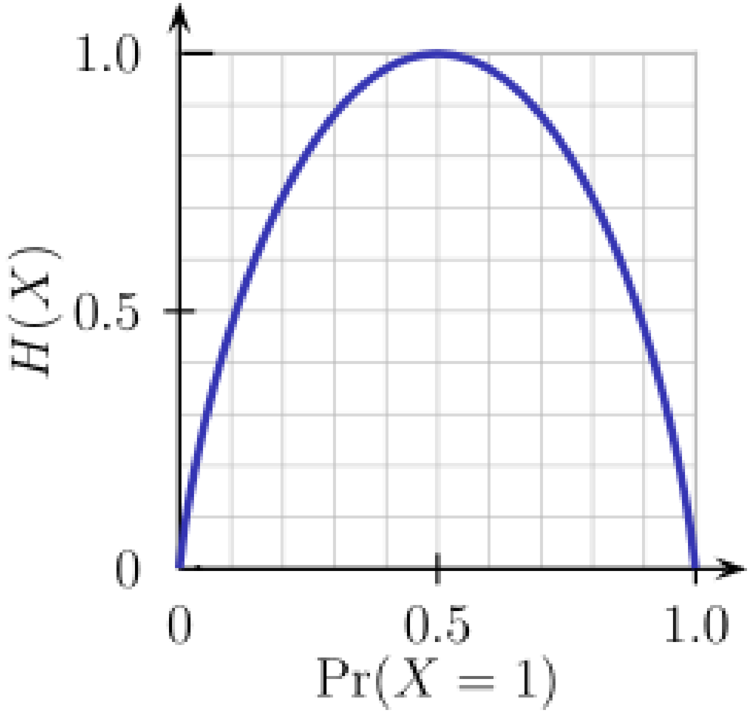
Entropía (Information Gain)

- Mide la diferencia que existe entre dos distribuciones de probabilidad de la misma variable. Es decir, cuanta información se pierde al dividir.

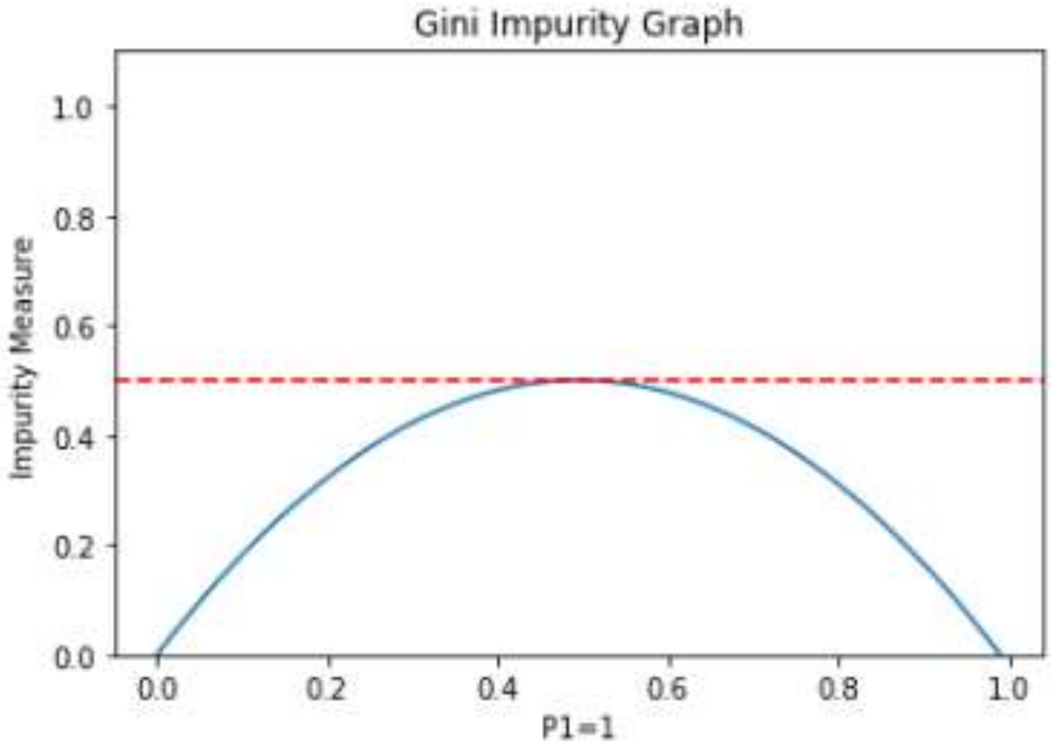
Impureza Gini

- Nos indica cuál es la probabilidad de clasificar mal una observación.

Entropía



Impureza Gini



Ejemplo Usando Gini

Le gusta la TV	**Le gusta el café**	**Edad**	**Le gusta leer**
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

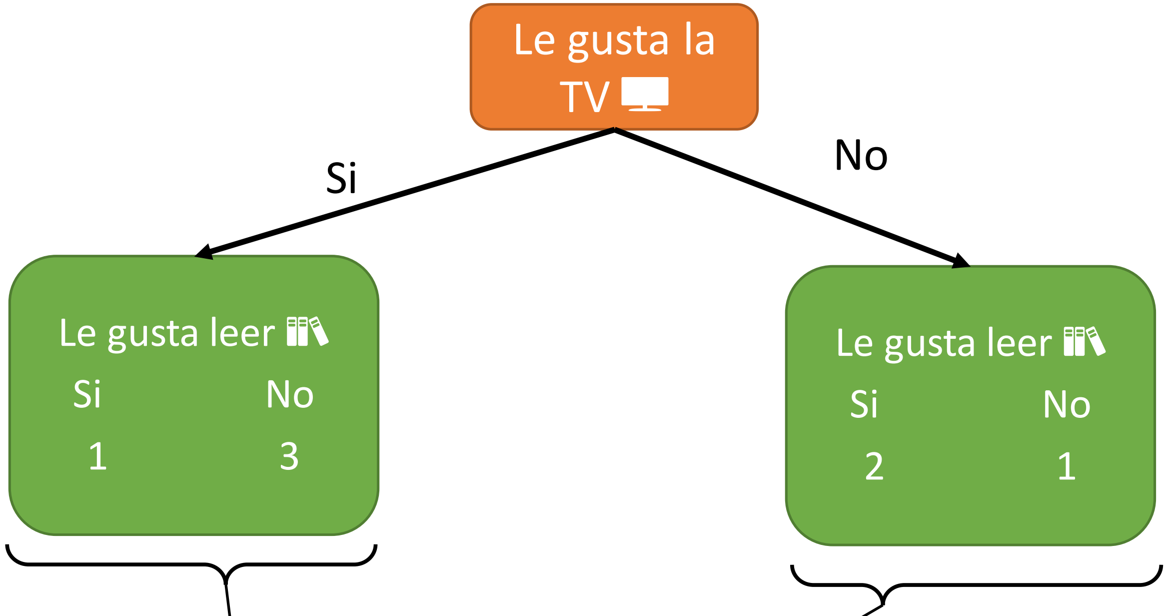
Le gusta la TV

Le gusta el café

Edad

Ejemplo Usando Gini

Le gusta la TV	Le gusta leer
Si	**No**
Si	**No**
No	**Si**
No	**Si**
Si	**Si**
Si	**No**
No	**No**

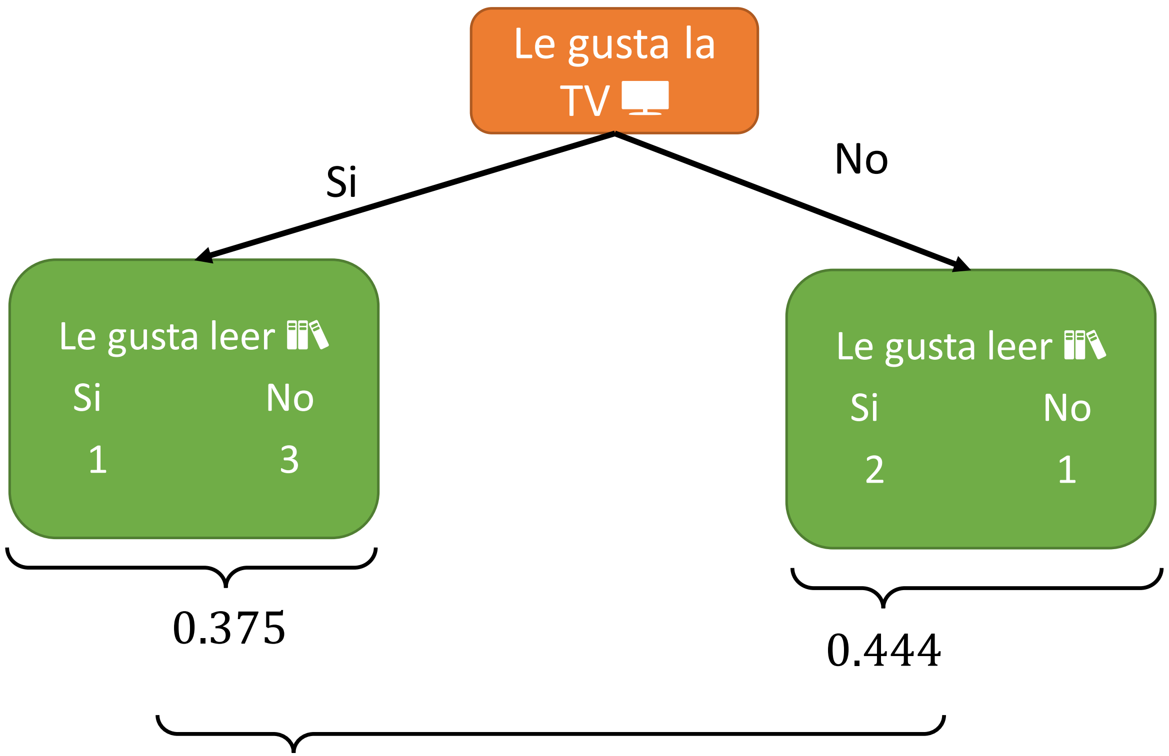


$$I(H) = \mathbf{1} - \sum_{\boldsymbol{i}}^{\boldsymbol{Clases}} \boldsymbol{P_i}^{\mathbf{2}} = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375$$

$$1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 = 0.444$$

Ejemplo Usando Gini

Le gusta la TV	Le gusta leer
Si	**No**
Si	**No**
No	**Si**
No	**Si**
Si	**Si**
Si	**No**
No	**No**



$$Impureza\ Gini\ Total(R) = \left(\frac{4}{4+3}\right)0.375 + \left(\frac{3}{4+3}\right)0.444 = 0.405$$

Ejemplo Usando Gini

Le gusta la TV	Le gusta el café	Edad	Le gusta leer
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

$Impureza\ Gini = 0.405$

Le gusta la TV

$Impureza\ Gini = 0.214$

Le gusta el café

$Impureza\ Gini = ?$

Edad

Para calcular valores numericos?

Ejemplo Usando Gini

Edad	Le gusta leer 📚
7	**No**
12	**No**
18	**Si**
35	**Si**
38	**Si**
50	**No**
83	**No**

9.5 → Impureza Gini = 0.429

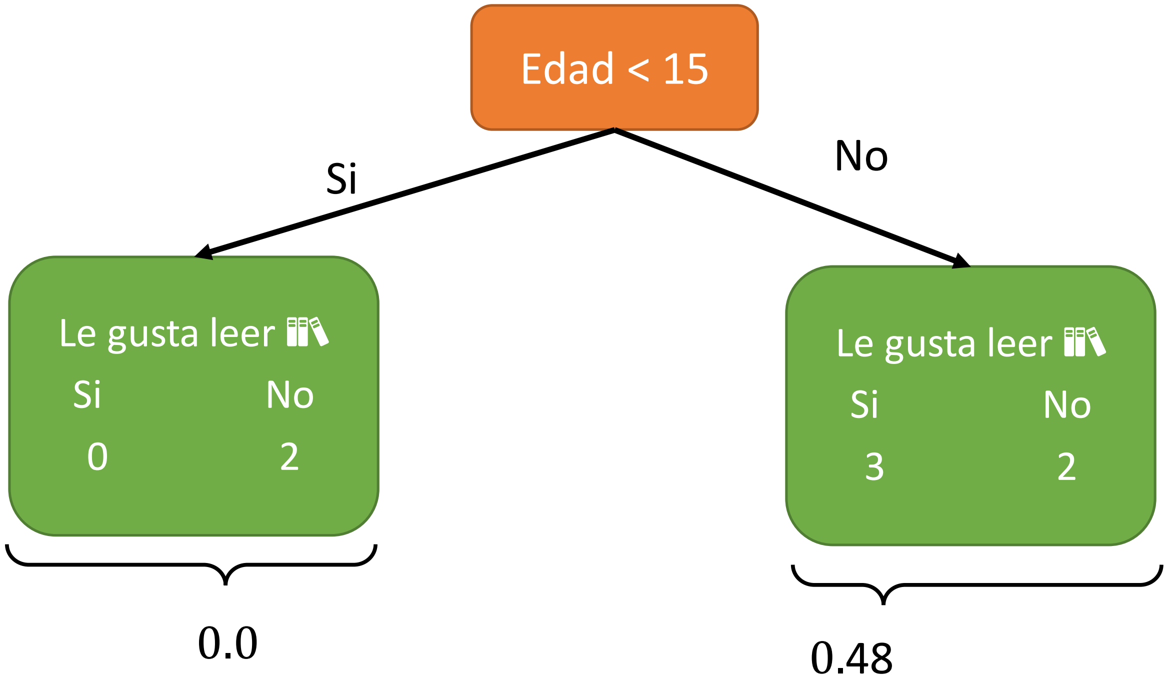
15 → Impureza Gini = 0.343

26.5 → Impureza Gini = 0.476

36.5 → Impureza Gini = 0.476

44 → Impureza Gini = 0.343

66.5 → Impureza Gini = 0.429



Escogemos el que tiene menor impureza

$$Impureza\ Gini\ Total(R) = 0.343$$

Ejemplo Usando Gini

Le gusta la TV	Le gusta el café	Edad	Le gusta leer
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

$Impureza\ Gini = 0.405$

Le gusta la TV

$Impureza\ Gini = 0.214$

Le gusta el café

$Impureza\ Gini = 0.343$

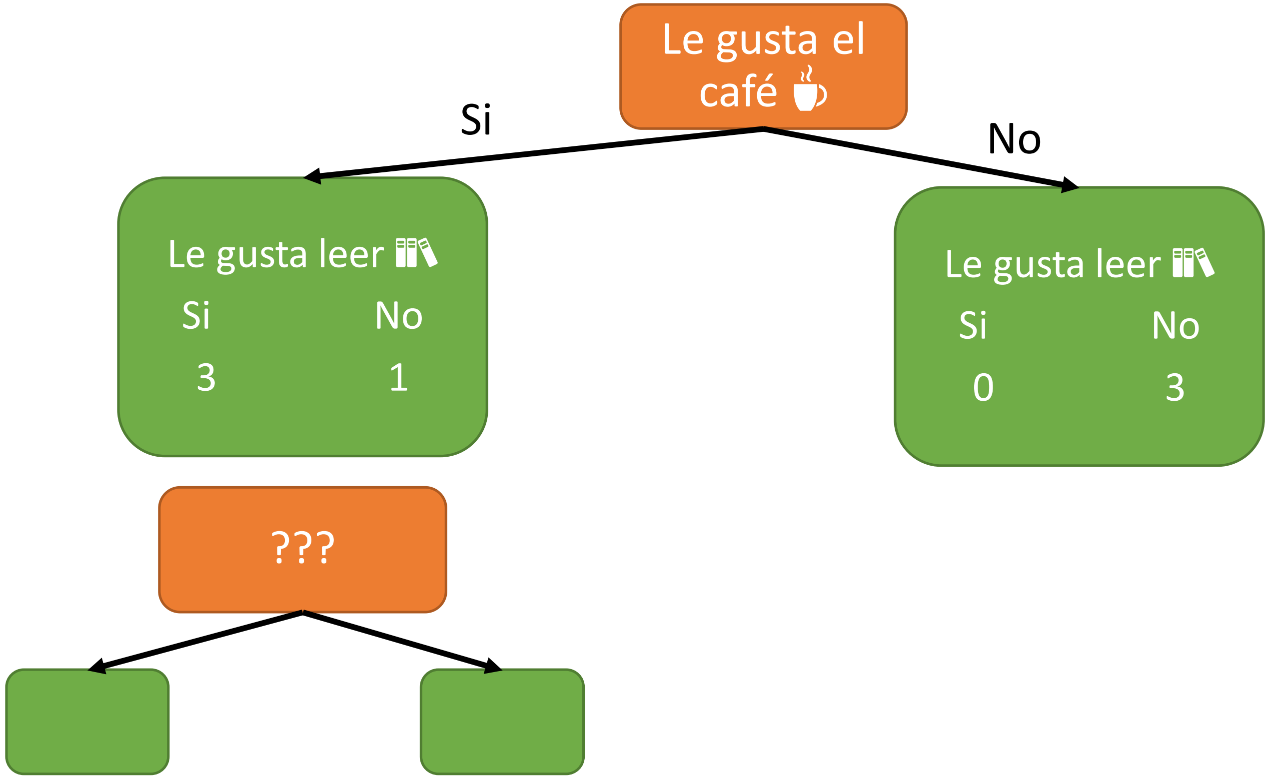
Edad

Escogemos el que tiene menor impureza

Ejemplo Usando Gini

Le gusta la TV	Le gusta el café	Edad	Le gusta leer
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

$Impureza\ Gini = 0.214$



Repetimos con el subconjunto

Usando Entropia

$$Information\ Gain = Entropia(Padre) - \sum_{clases}^{Total\ Clases} Entropia(clases, nodo)$$

Le gusta la TV	**Le gusta el café**	**Edad**	**Le gusta leer**
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

$$Entropia(S) = -\sum_i p_i \ln p_i$$

S es el estado actual
p_i el porcentaje de la clase i en el nodo del estado S

Usando Entropia

Le gusta la TV	Le gusta el café	Edad	Le gusta leer
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

$$Entropia(Leer) = -p_{si} \ln p_{si} - p_{no} \ln p_{no}$$
$$= -0.43 \ln 0.43 - 0.57 \ln 0.57$$
$$= 0.683$$

Primero debemos calcular la entropía inicial del sistema

Para TV

$$Entropia_{TV} = P_{Si\text{🖥}}\Big(E(Si\text{📚}, Si\text{🖥}) + E(No\text{📚}, Si\text{🖥})\Big) + P_{No\text{🖥}}\Big(E(Si\text{📚}, No\text{🖥}) + E(No\text{📚}, \mathrm{No}\text{🖥})\Big)$$

Le gusta la TV 🖥	**Le gusta el café ☕**	**Edad**	**Le gusta leer 📚**
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

$$Entropia_{TV} = \frac{4}{7}\left(-\frac{1}{4}\ln\frac{1}{4} - \frac{3}{4}\ln\frac{3}{4}\right) + \frac{3}{7}\left(-\frac{2}{3}\ln\frac{2}{3} - \frac{1}{3}\ln\frac{1}{3}\right)$$

$$Entropia_{TV} = 0.594$$

$$IG_{TV} = Entropia(Padre) - Entropia_{TV}$$

$$IG_{TV} = 0.089$$

Para Cafe

Le gusta la TV	Le gusta el café	Edad	Le gusta leer
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

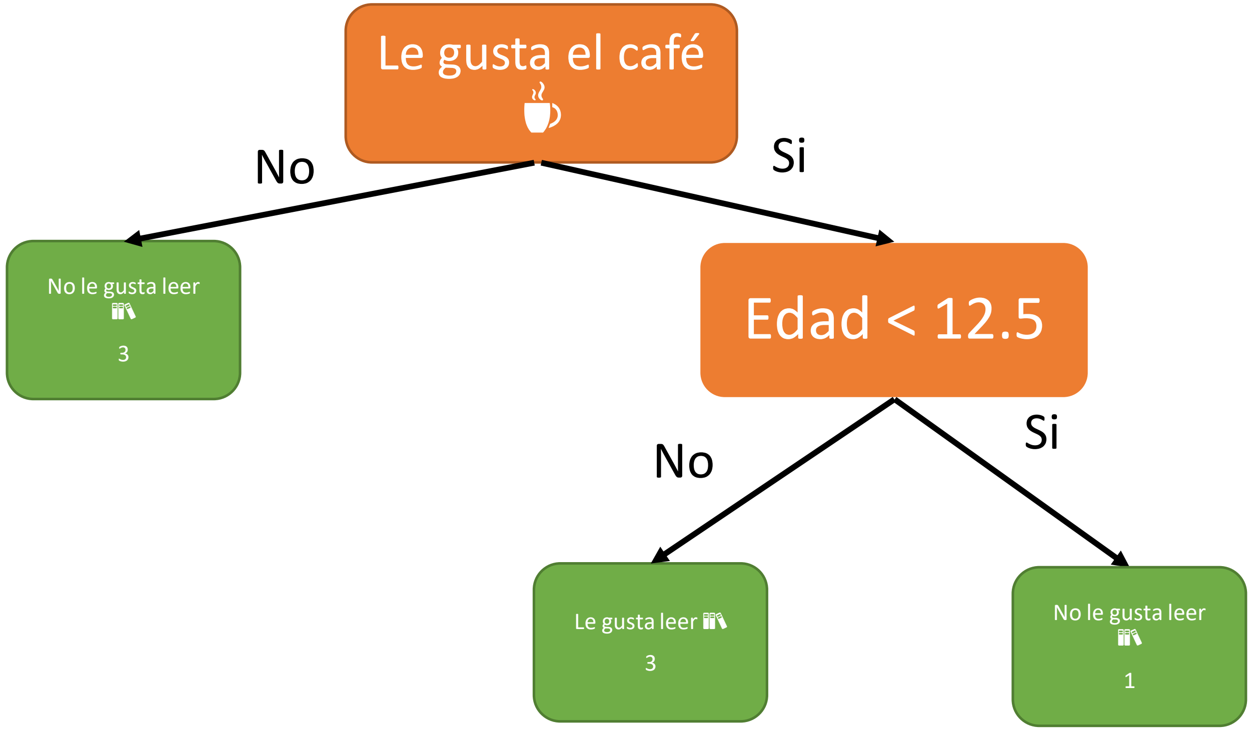
$$Entropia_{Cafe} = \frac{4}{7}\left(-\frac{3}{4}\ln\frac{3}{4} - \frac{1}{4}\ln\frac{1}{4}\right) + \frac{3}{7}\left(-\frac{0}{3}\ln\frac{0}{3} - \frac{3}{3}\ln\frac{3}{3}\right)$$

$$Entropia_{Cafe} = 0.321$$

$$IG_{Cafe} = Entropia(Padre) - Entropia_{Cafe}$$

$$IG_{Cafe} = 0.362$$

Resultado en Ambos casos



Overfitting

- Cuando todas las hojas son puras, es decir solo tienen una clase, es probable que el modelo haga overfitting.
- Para prevenir se hace poda del árbol.
  - Se define un número de elementos mínimos por cada hoja, la clase de la hoja es la clase con más elementos.
  - Se define una penalización por complejidad del árbol

Arboles de Regresión

- No se puede usar entropía o Gini
- Se puede utilizar el sum of squared errors (SSE)

$$\sum_{clase\ A} (\bar{y}_{clase\ A} - y^{(n)})^2 + \sum_{clase\ B} (\bar{y}_{clase\ B} - y^{(n)})^2$$

Una especie de desviación estándar

Arboles de Decisión

Ventajas 👍

- Interpretable
- Funciona variables numéricas y categóricas
- No requiere de mucho preprocesamiento
- No requiere asumir forma (distribución) de la data

Desventajas 👎

- Suele hacer overfitting si no se tiene cuidado
- La predicción puede ser limitada
- Regresión limitada
- Data desbalanceada impacta resultados

Métodos de Ensemble

Bagging y Boosting

Bagging

- **B**ootstrap **Agg**regat**ing**
- Reduce la varianza en los algoritmos.
  - Evita ovefitting
- Es el promedio de varios modelos que entrenan sobre una muestra aleatoria de la data.
- La muestra puede contener elementos repetidos

Bagging

Bootstrapping

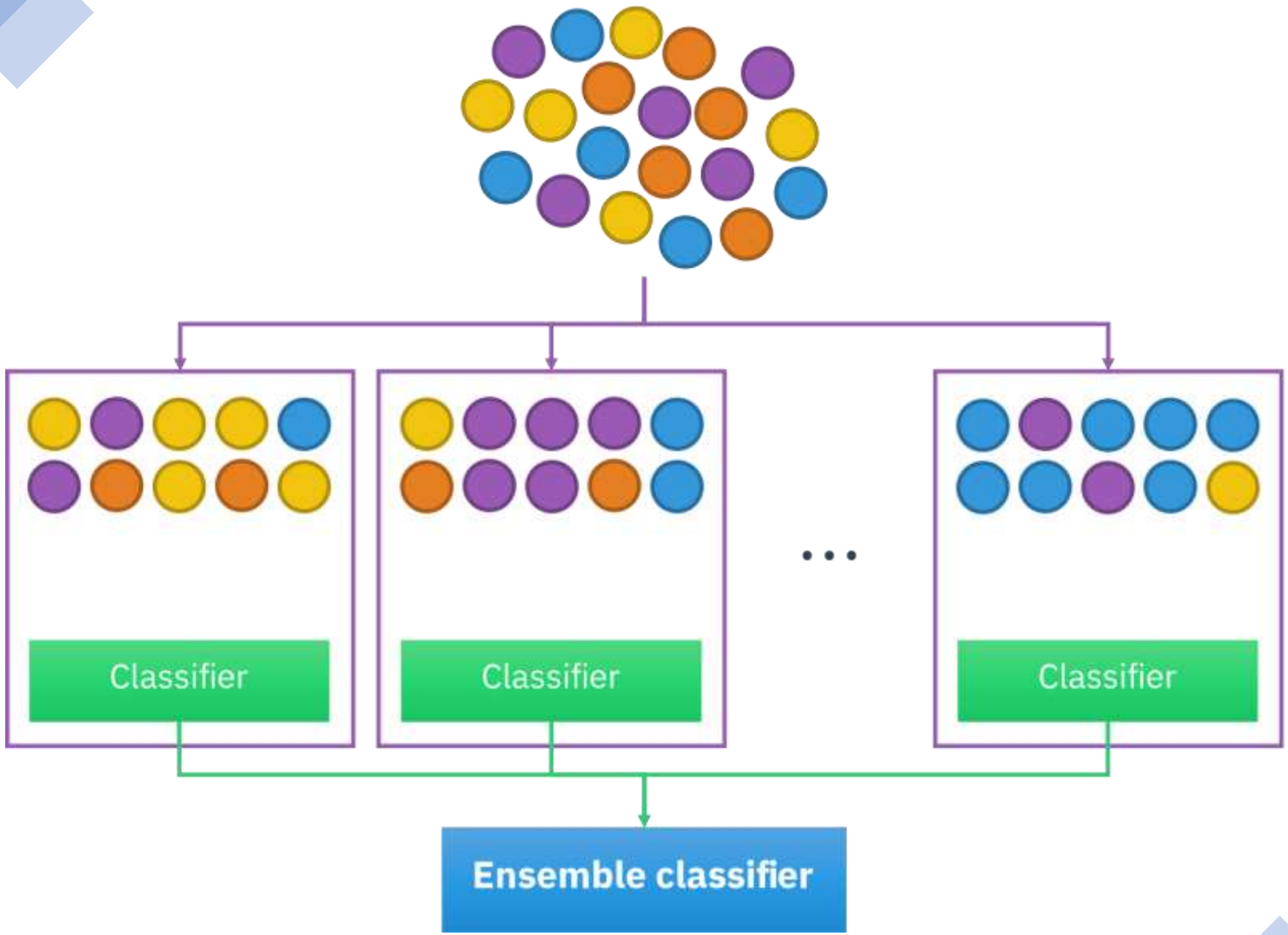
- Crear múltiples datasets usando técnica de muestreo con reposición

Entrenamiento Paralelo

- Por cada una de las N muestras, se crea un modelo para entrenar sobre ese dataset.

Promedio

- Para una clasificación, el resultado es la clase mayoritaria que se obtenga de los N sistemas
- Para regresión, es el promedio



Original Data

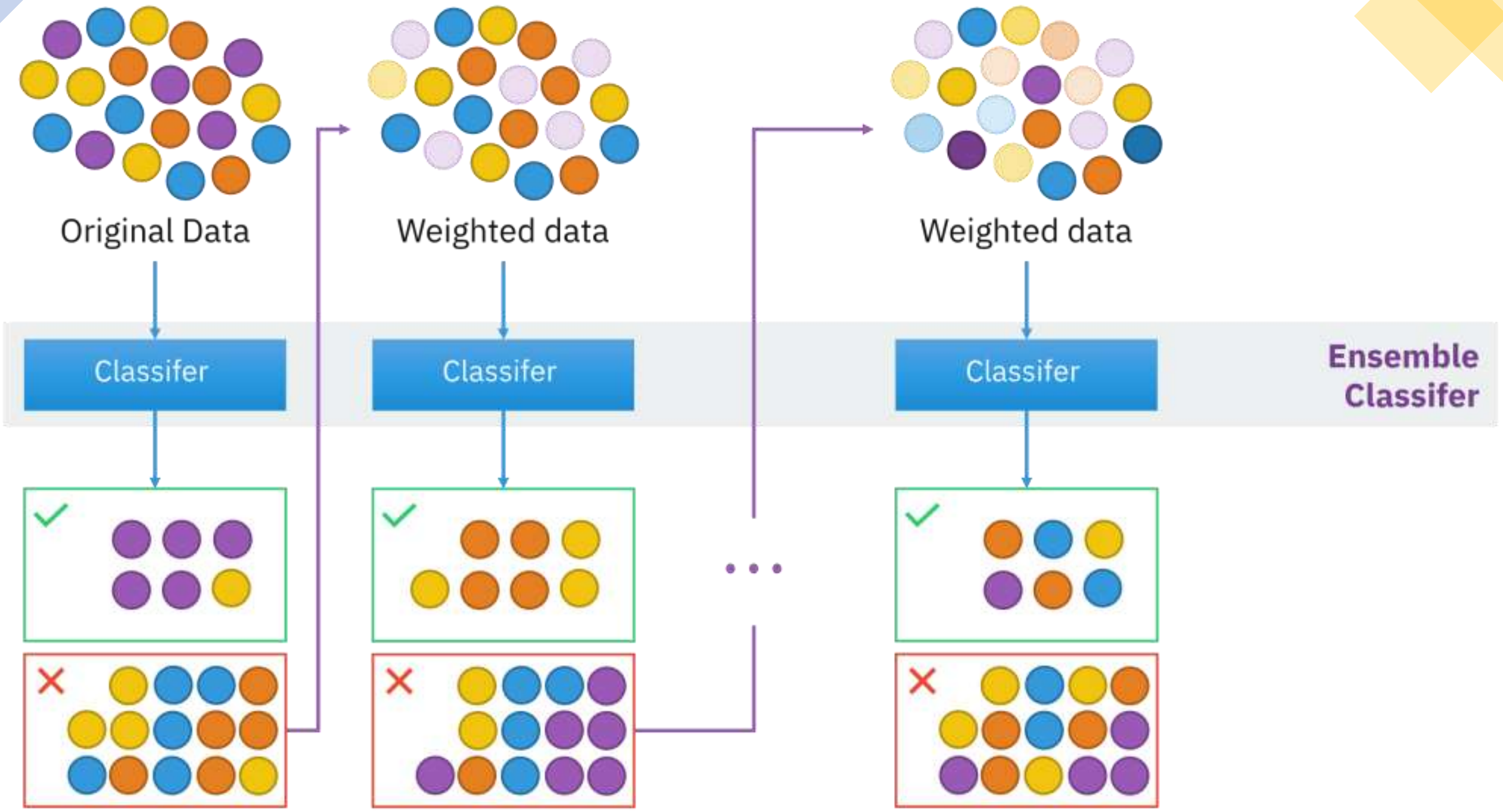
Bootstrapping

Aggregating

Bagging

Boosting

- Un conjunto de clasificadores “débiles” combinados que forman un clasificador más potente
- Se entrena secuencialmente
- Los errores del clasificador previo son tomados en cuenta para la métrica de error del siguiente clasificador


Original Data
Weighted data
Weighted data
Classifer
Classifer
Classifer
Ensemble
Classifer

Adaboost

- **Ada**ptative **Boost**ing
- Clasificación binaria
- Cada datapoint va a tener un peso asociado $w_n^{(m)}$ donde n es el n-esimo datapoint y m es el m-esimo modelo de M modelos en total

- Inicializamos los pesos con $w_n^{(1)} = \frac{1}{N}$ donde N es el número de ítems en el dataset.
- Todos los pesos tienen el mismo valor porque no queremos enfatizar uno sobre otro

Adaboost

Para cada $m = 1 \;\rightarrow\; M$

- Entrenar el modelo m donde la función de costo estará definida como

$$Costo^{(m)} = \sum_{n=1}^{N} w_n^{(m)}\, I(x_n)$$

Donde I es una función Indicador. Tendrá valor 0 si clasifica correctamente, 1 si clasifica incorrectamente.

Para el primer clasificador, todos los pesos son iguales. Para los siguientes clasificadores, el peso será mas grande si el clasificador anterior se equivocó

Adaboost

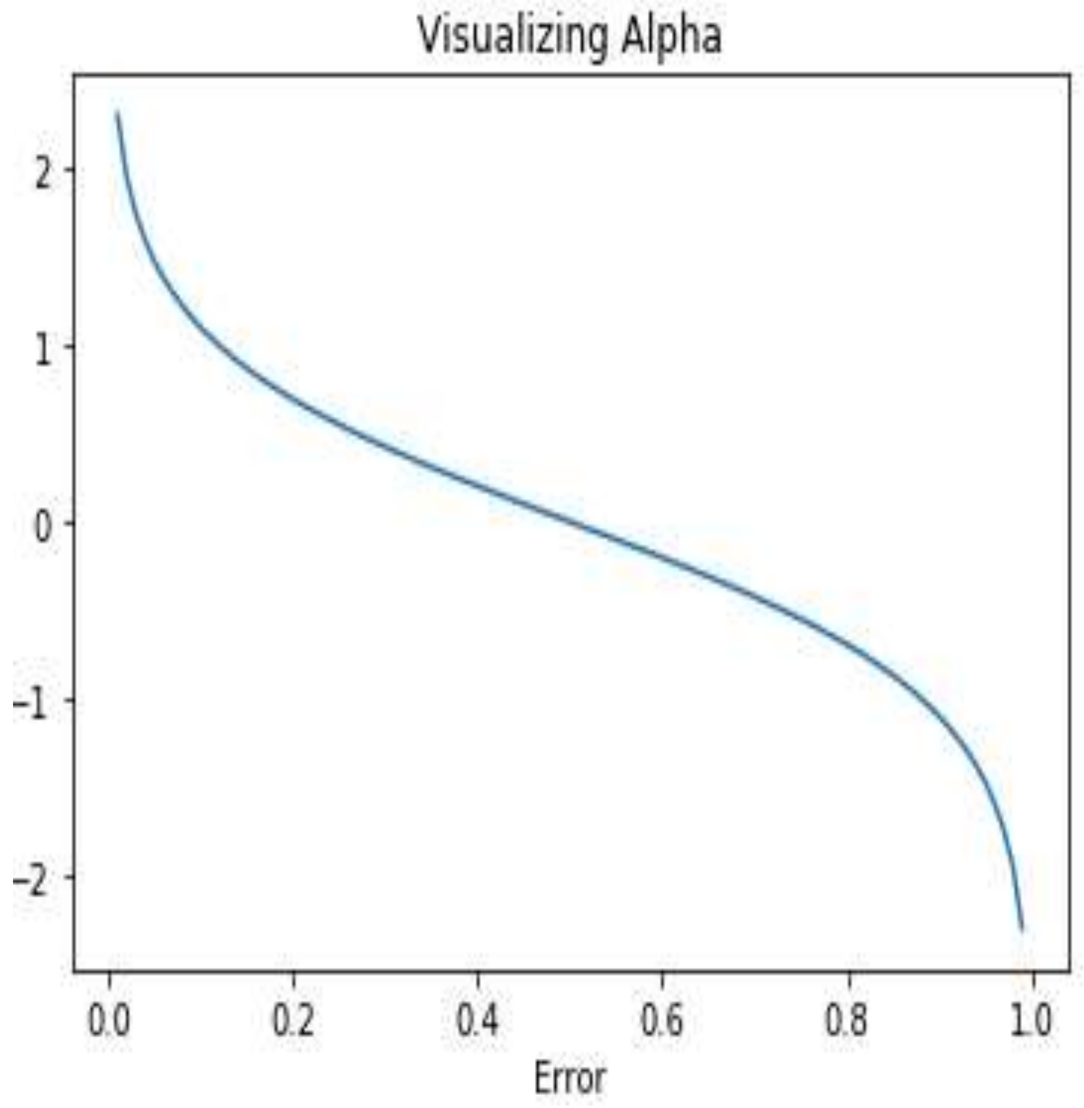
- Calculamos el error y la contribución (α)

$$error^{(m)} = \frac{Costo^{(m)}}{\sum_{n=1}^{N} w_n^{(m)}}$$

$$\alpha^{(m)} = \ln \frac{1 - error^{(m)}}{error^{(m)}}$$

Si todas las muestras fueron clasificadas correctamente, el error es 0, y alfa es +

Si no, alfa es -



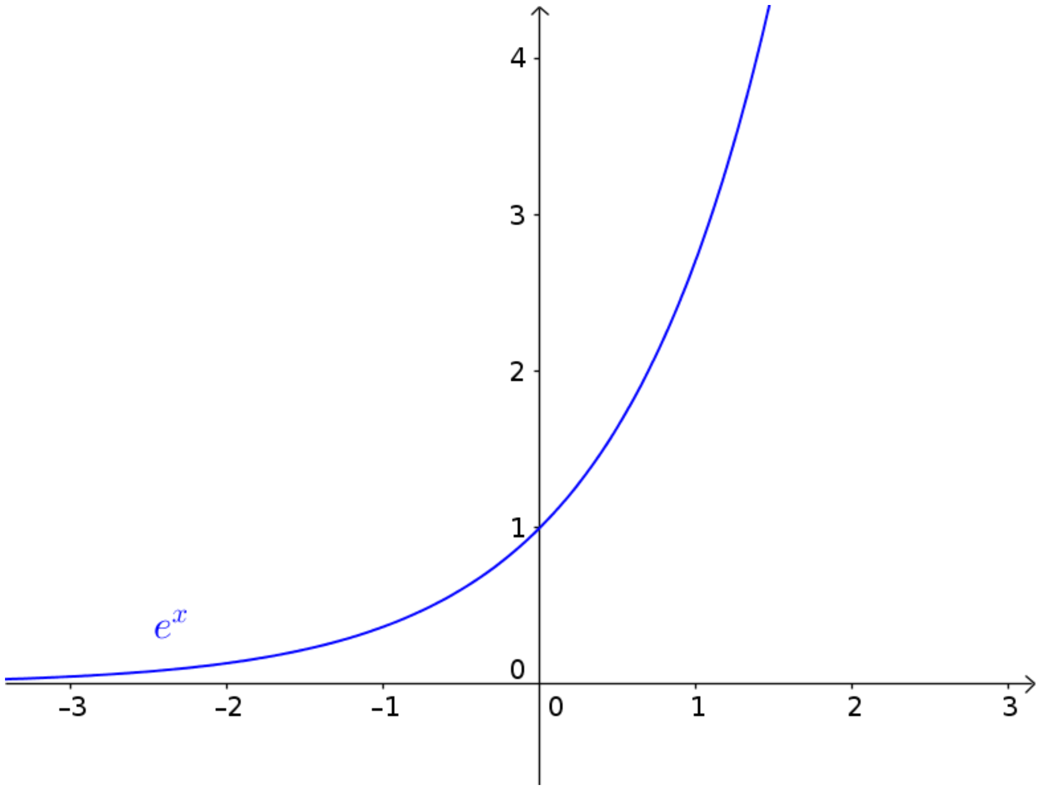
Adaboost

- Actualizamos los pesos

$$w_n^{(m+1)} = w_n^{(m)} e^{\alpha^{(m)} I(x_n)}$$

Recordando que I es una función Indicador. Tendrá valor 0 si clasificó correctamente, 1 si clasificó incorrectamente

Aquellas muestras erradas crecerán en $e^{\alpha^{(m)}}$, las correctas se mantienen igual



Adaboost

- Prediciendo

$$Prediccion\ (x) = \sum_{m=1}^{M} \alpha^{(m)} modelo^{(m)}(x)$$

Los mejores modelos tendrán un $\alpha^{(m)}$ más grande, por lo tanto tendrán mayor contribución en la decisión final.

Random Forest

Random Forest

- Combina los conceptos de Árboles de Decisión con Bagging y adicionalmente random subspace (selección aleatoria de features)

1) Generar múltiples datasets usando el Bootstrap.
2) Construir árboles de decisición a una profundidad predeterminada
    1) Seleccionar aleatoriamente una muestra de **features**
    2) Dividir el dataset en base a esos features
    3) Construir el árbol hasta la profundidad predeterminada
3) Regresar el conjunto de arboles.
4) Predecir usando el promedio o voto mayoritario

Random Forest

- Es relativamente insensible a variables no informativas
- Pueden caer en overfitting si se tiene demasiada complejidad
- Son ideales para entrenamiento en parallelo, en contraste con el boosting
- Tienen buena performance con poca necesidad de ajustar hiperparámetros.