



DigitalHouse >
Coding School
DATA SCIENCE

Intro a CARTs

INTRO a CARTs



1

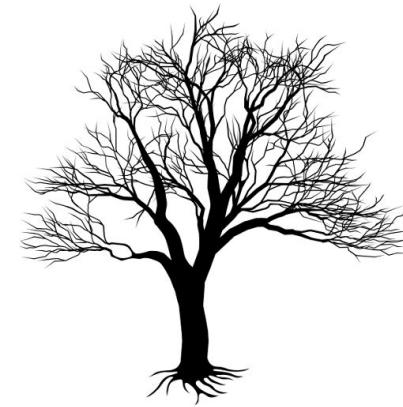
Describir que es un árbol de decisión

2

Explicar cómo funciona un árbol de clasificación

3

Explicar cómo funciona un árbol de regresión



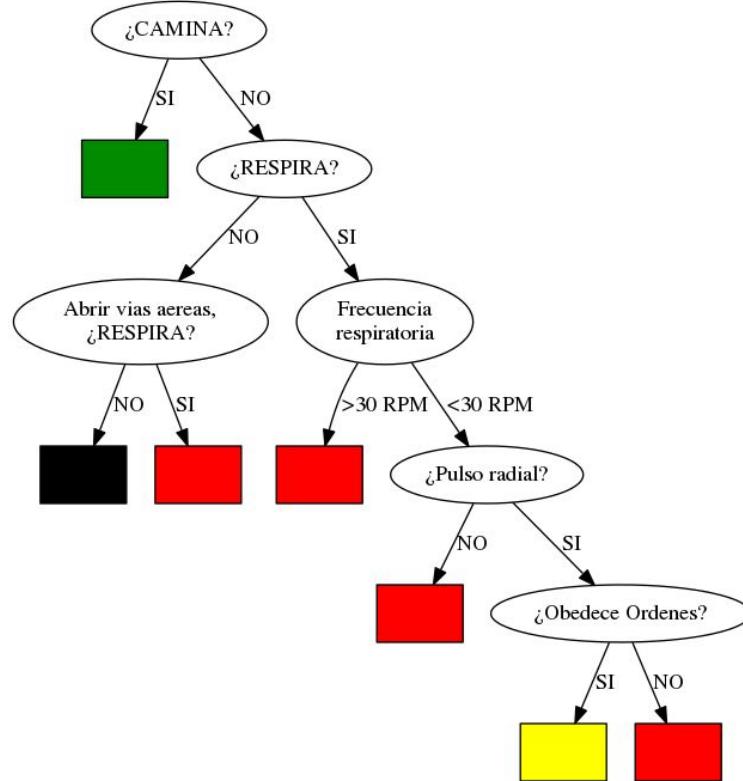
INTRODUCCIÓN



¿Qué es un árbol de decisión?

Consideremos como ejemplo el método Triage para la clasificación de heridos según su gravedad. Aquí los colores determinan la gravedad del paciente.

- El árbol da una **regla precisa para decidir** la gravedad del paciente, dependiendo de los valores tomados por las variables.
- La **interpretación del modelo es extremadamente simple** lo que facilita su comunicación a gente no técnica.



¿Qué es un árbol de decisión?

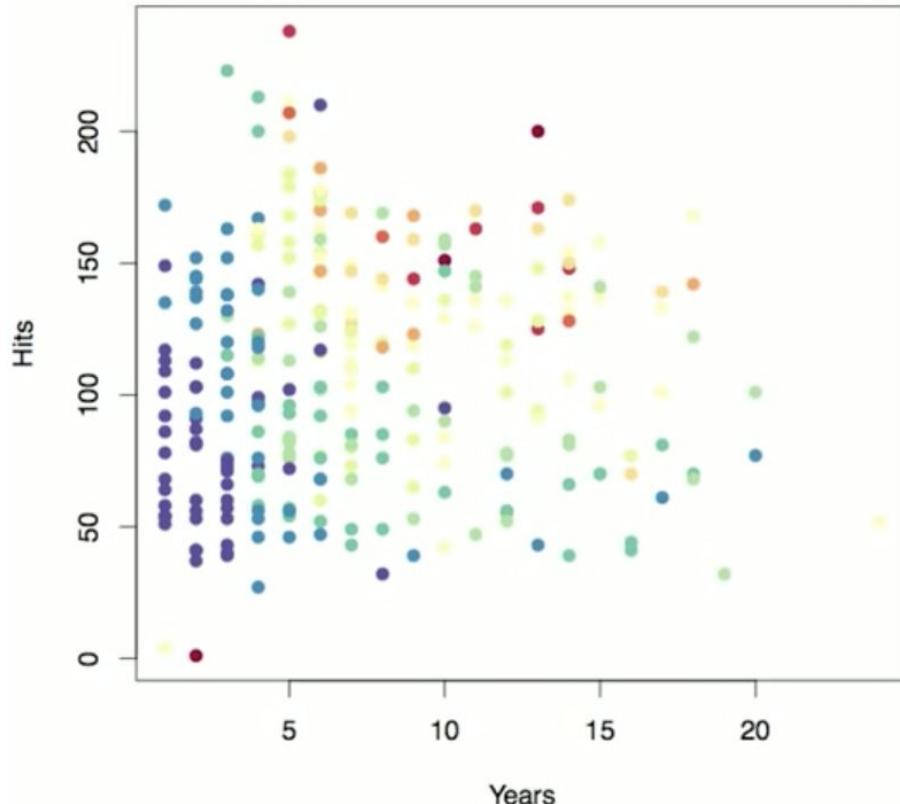
Los árboles de decisión son una **técnica de aprendizaje** estadístico que puede usarse para **regresión** y para **clasificación**.

- Las técnicas de árbol implican **estratificar o segmentar el espacio de predictores** en un número de **regiones simples**.
 - Para hacer una predicción para una observación dada típicamente se usa la media o la moda de la variable de respuesta para la región a la que pertenece la observación.
- Los métodos basados en árboles son simples y útiles para la interpretación. Sin embargo, **típicamente no son competitivos** con los mejores enfoques basados en aprendizaje supervisado en términos de precisión predictiva.
- Veremos que la **combinación de un gran número de árboles puede llevar a mejoras sustanciales en la precisión** de las predicciones a cambio de cierta pérdida en la interpretabilidad de los resultados.

Características de un Árbol de Decisión:

- **No paramétrica:** el modelo no puede describir mediante una lista finita de parámetros.
- **Jerárquica:** el modelo consiste en una secuencia de preguntas que definen una etiqueta de clase cuando se aplica a cualquier dato. En otras palabras, una vez entrenado, el modelo se comporta como una receta, que nos dará un resultado si se sigue exactamente.

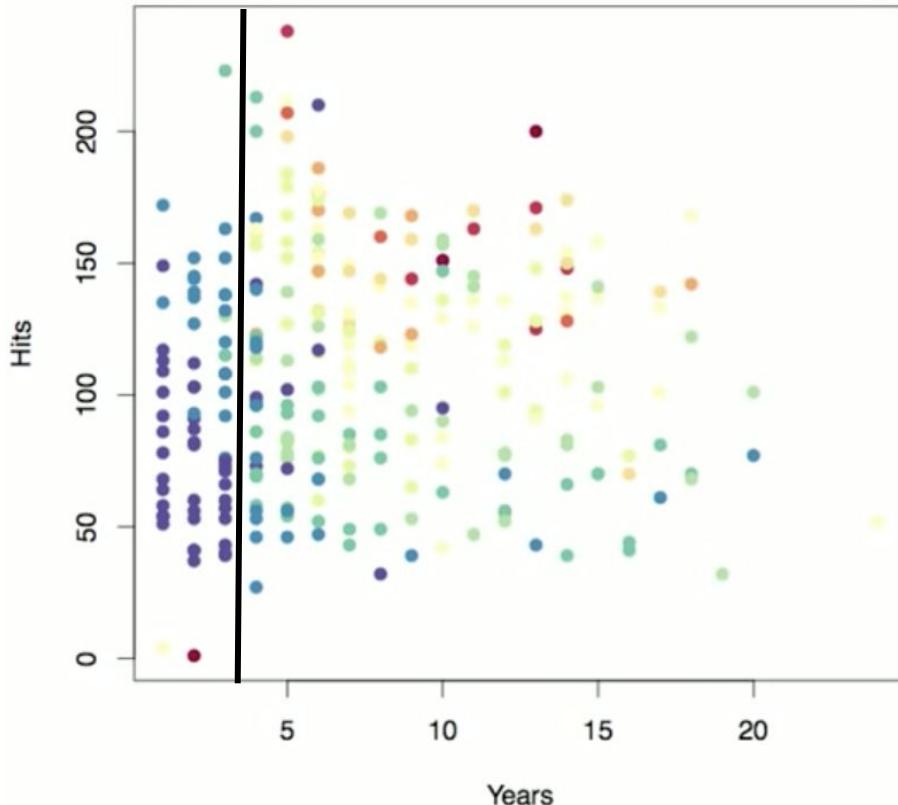
Ejemplo: Árboles de regresión para predecir el salario de un jugador de baseball



El (logaritmo del) salario está representado por los colores.
Los puntos **AZULES** representan los salarios más bajos y los **ROJOS** los más altos.

¿Cómo segmentarían el espacio para separar los salarios?

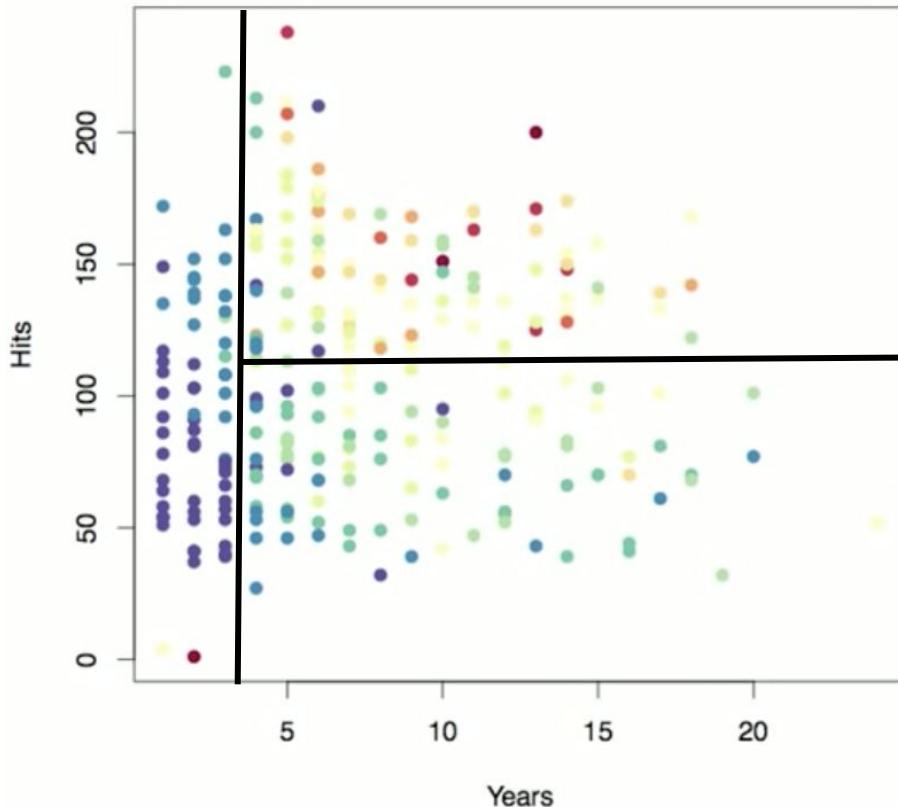
Ejemplo: Árboles para regresión para predecir el salario de un jugador de baseball



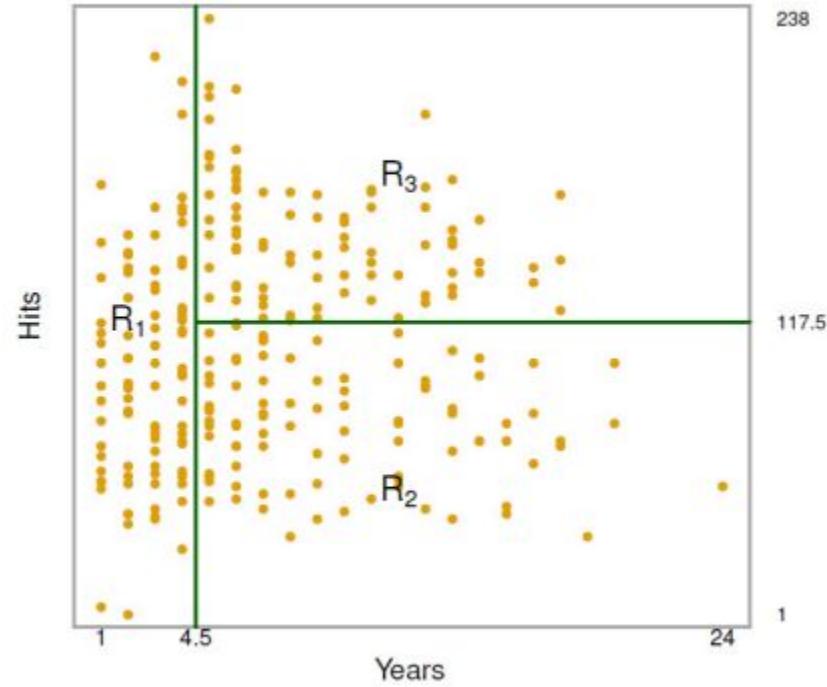
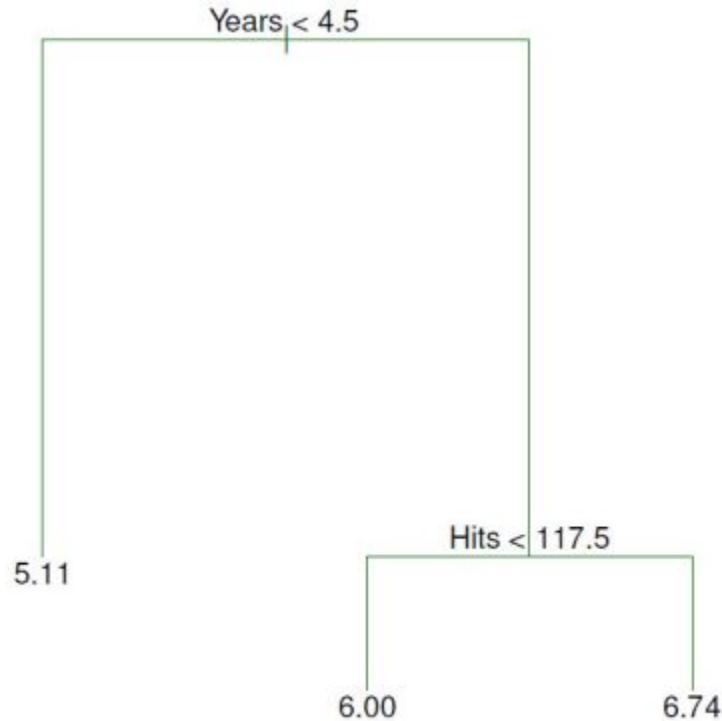
Podemos hacer una primera
partición en los Años.

Separamos las observaciones
con menos de
aproximadamente 5 años.

Ejemplo: Árboles para regresión para predecir el salario de un jugador de baseball



Luego podemos dividir por la cantidad de Hits.



Ejemplo: árboles para regresión

Tratemos de predecir el salario de un jugador de baseball en base a su experiencia (años en la liga) y registros de hits la temporada pasada.

- Para los jugadores con menos de 4.5 años de experiencia el salario pronosticado está dado por la media de los salarios de jugadores con menos de 4.5 años de experiencia. Para esos jugadores la media de los logaritmos de los salario es 5.107 por lo que valor pronosticado para el salario es $\exp(5.107)$ miles de dólares o \$165,174,
- Para los jugadores con experiencia ≥ 4.5 existen dos grupos posibles en base a sus hits: aquellos con menos de 118 la temporada pasada, con salario esperado de $\exp(5.999) = \$402,834$, y otro grupo con al menos 118 hits la temporada pasada $\exp(6.740) = \$845,346$.
- Estas tres regiones pueden escribirse como $R1 = \{X \mid \text{Years} < 4.5\}$, $R2 = \{X \mid \text{Years} \geq 4.5, \text{Hits} < 117.5\}$, and $R3 = \{X \mid \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$.

Ejemplo: árboles para regresión

Analogía árboles

- Las regiones R1, R2, y R3 se conocen como **nodos terminales o hojas del árbol**.
- Los puntos a lo largo del árbol donde el espacio de predictores es particionado se conocen como **nodos internos**.
- Los segmentos de los árboles que conectan los nodos se denominan **ramas**.

¿Como construimos las regiones?

El objetivo es encontrar cajas R_1, \dots, R_J que minimizan la suma de residuos al cuadrado (RSS) dada por:

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

Donde \hat{y}_{R_j} es la media de y de la caja R_j .

¿Como construimos las regiones?

- El problema es que no es *factible computacionalmente* considerar todas las posibles particiones del espacio de atributos en J cajas. Por esta razón se usa un enfoque de arriba hacia abajo “greedy” que es conocido como **recursive binary splitting**.
 - El **recursive binary splitting** comienza en la parte de arriba del árbol (donde todas las observaciones pertenecen a una sola región) y sucesivamente partitiona el espacio de predictores. Cada nueva división se indica mediante dos nuevas ramas que sigue hacia abajo en el árbol

¿Cómo construir un árbol de decisión?

- Características del método ***recursive binary splitting***:
 - **recursivo**: divide el trabajo en partes, y resuelve cada parte dividiéndolas a su vez en partes más pequeñas.
 - **voraz** (greedy): porque en cada paso de la construcción del árbol se busca la mejor división en ese punto en particular en lugar de mirar hacia adelante y elegir una división que llevaría a un mejor árbol en un paso futuro.
 - **óptimo local**: no alcanza la mejor solución de todas las posibles, sino una localmente óptima.
- El algoritmo funciona dividiendo recursivamente los registros en subconjuntos cada vez más pequeños. La decisión de partición se realiza en cada nodo de acuerdo con una métrica llamada pureza. Se dice que un nodo es 100% puro cuando todos sus registros pertenecen a una sola clase (o tienen el mismo valor).

¿Como construimos las regiones?

- Para hacer el **recursive binary splitting** primero seleccionamos el predictor X_j y el punto de corte s tal que separar el espacio de predictores en las regiones $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ lleva a la mayor reducción posible de la RSS.
- Para esto debemos considerar todos los predictores X_1, \dots, X_p , y todos los valores posibles de los puntos de corte s para cada uno de los predictores y luego elegir el predictor y punto de corte tal que el árbol resultante tiene el menor RSS.
- En más detalle definimos las regiones como

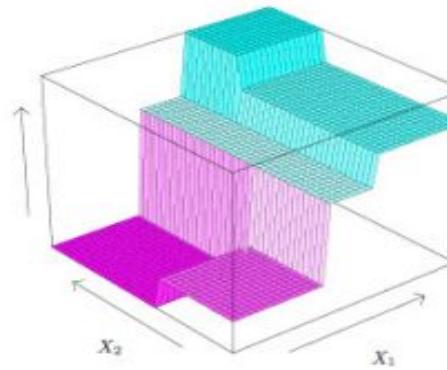
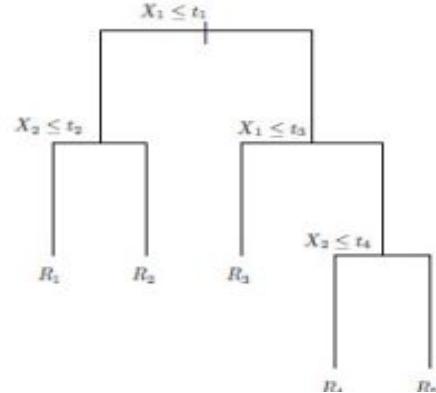
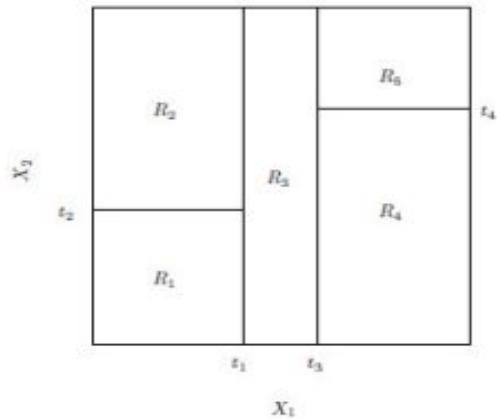
$$R_1(j, s) = \{X|X_j < s\} \text{ and } R_2(j, s) = \{X|X_j \geq s\},$$

- Buscamos minimizar la siguiente expresión eligiendo X_j y s dado que los valores pronosticados para cada región están dados por el promedio de la y en esa partición.

$$\sum_{i: x_i \in R_1(j, s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j, s)} (y_i - \hat{y}_{R_2})^2,$$

¿Como construimos las regiones?

- A continuación repetimos el proceso buscando el mejor predictor y el mejor punto de corte para volver a dividir los datos con el objetivo de minimizar el RSS para la nueva partición. Pero esta vez en lugar de dividir el espacio de predictores entero, dividimos una de las regiones identificadas previamente.
- Ahora tenemos 3 regiones. Volvemos a buscar una división de estas 3 regiones para minimizar el RSS.
- **El proceso continúa hasta que se cumple con algún criterio de detención** (cantidad de observaciones en cada región por ejemplo)
- Una vez que tenemos creadas las regiones R₁,...,R_J **predecimos la respuesta de una observación de test en base la media de las observaciones de training en la región a la que pertenece la observación de test.**



Árboles para clasificación

- Un árbol de clasificación es muy similar a uno de regresión excepto que se utiliza para predecir una variable cualitativa en lugar de una cuantitativa.
- La predicción se obtiene como **la etiqueta mayoritaria para las observaciones de training dentro de la región a la que pertenece una observación**.
- Para construir el árbol usamos recursive binary splitting pero con un criterio distinto a RSS. Una alternativa natural es la tasa de error de clasificación. Como vamos a clasificar a una observación de acuerdo a la clase más frecuente en su región, la **tasa de error de clasificación** es simplemente la **fracción de las observaciones de training en esa región que no pertenecen a la clase mayoritaria**.

$$E = 1 - \max_k(\hat{p}_{mk}).$$

Árboles para clasificación

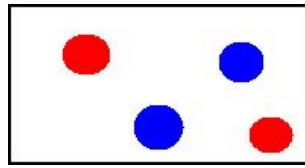
- La tasa de error de clasificación no es suficientemente sensible para construir el árbol y en la práctica se prefieren otras dos medidas.
- El índice de Gini.

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

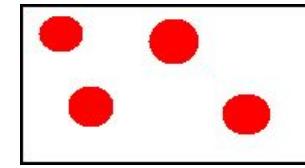
- El índice de Gini mide la varianza total a lo largo de las K clases para cada nodo terminal m . El índice de Gini, toma valores chicos cuando las proporciones son cercanas a 1 o 0.
- Por esto el **índice de Gini se considera como una medida de pureza** de los nodos terminales, un valor pequeño indica que un nodo contiene predominantemente observaciones de una misma clase.

El Concepto de Impureza

Supongamos que queremos hacer una clasificación según el color de las figuras.



Impureza Máxima



Impureza Mínima

Árboles para clasificación

- Otra alternativa está dada por la **entropía**.

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}.$$

- Como el índice de Gini la entropía tomará un valor pequeño si los nodos son “puros”.
- Cuando se construye un árbol de clasificación típicamente el Gini index o la entropía son usados para evaluar la calidad de una división en particular porque estas medidas son más sensibles a la pureza de los nodos que la tasa de clasificación.

Funciones de optimización

- **¿Cómo determinamos la mejor división entre todas las divisiones posibles?**
- Considerando que no es necesario dividir cuando todos los registros pertenecen a la misma clase pura, buscaremos en cada paso crear la partición con la **pureza más alta posible**.
- Para hacer esto, necesitaremos una función objetivo para optimizar, que mida la ganancia en pureza de una división particular. Por lo tanto, queremos que dependa de la distribución de clases sobre los nodos (antes y después de la división).

Funciones de optimización

Error de clasificación:

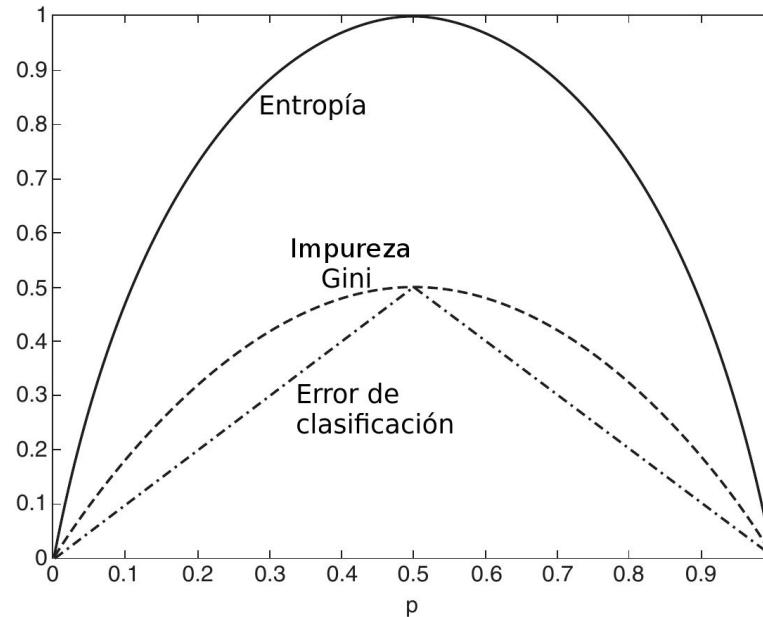
$$E = 1 - \max_k(\hat{p}_{mk}).$$

Índice de Gini:

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

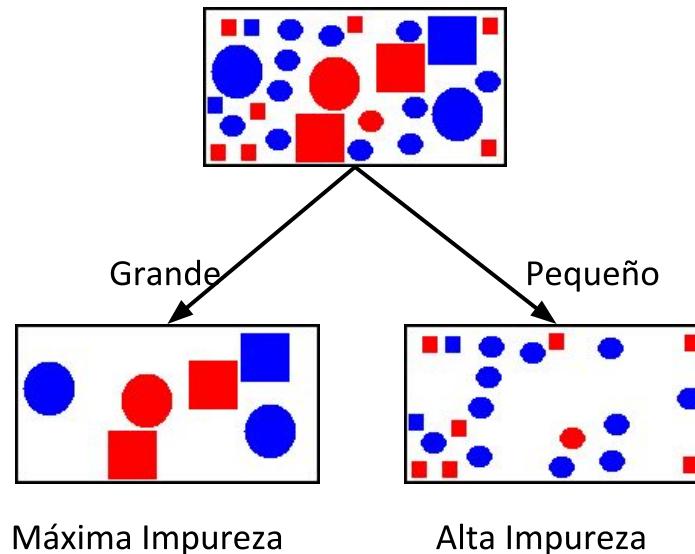
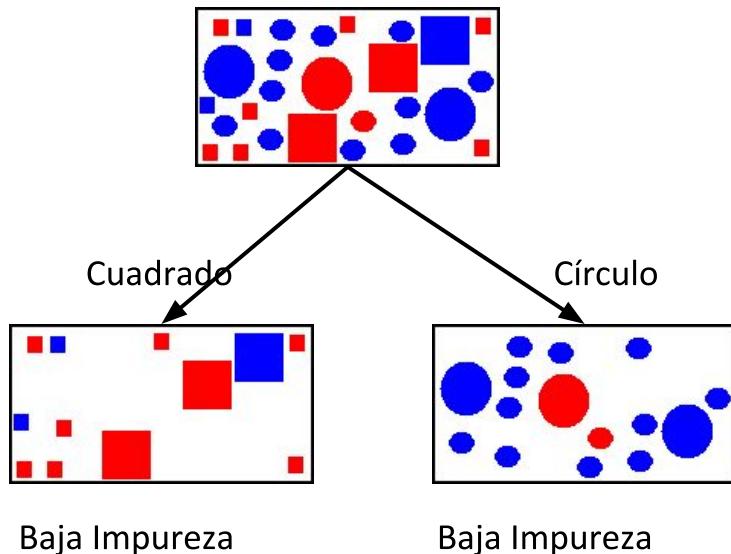
Entropía:

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}.$$



Observar que las tres funciones tienen su máximo en 0.5 y sus mínimos en 0 y 1.

Siguiendo con el ejemplo de la clasificación según el color de las figuras. Si podemos seleccionar la forma y el tamaño, ¿que partición conviene realizar?



Concepto de Ganancia

- Las medidas de impureza por sí solas, no son suficientes para decirnos cómo funcionará la división. Todavía **tenemos que mirar la impureza antes y después de la división**. Podemos hacer esta comparación usando **la ganancia**:

$$\Delta = I(\text{padre}) - \sum_{j \in \text{hijos}} \frac{N_j}{N} I(\text{hijo}_j)$$

- Donde I es la medida de impureza, N_j es el número de registros en el nodo hijo j y N es el número de registros en el nodo padre.
- Cuando I es la entropía, esta cantidad se llama **ganancia de información**.

Concepto**de****Ganancia**

- En el **caso de regresión**, la variable de resultado no es una categoría sino un valor continuo. Por lo tanto, no podemos usar la misma medida de impureza que utilizamos para la clasificación.
- No resulta difícil ver que si el objetivo es predecir un valor numérico, la varianza de dicho valor en un nodo nos dá una medida de impureza de dicho valor.
- Por ello **utilizaremos el Error Cuadrático Medio como medida de impureza** y la función a maximizar seguirá siendo la ganancia.

$$\Delta = ECM(\text{padre}) - \sum_{j \in \text{hijos}} \frac{N_j}{N} ECM(\text{hijo}_j)$$

- El objetivo es buscar la máxima ganancia, donde ECM es el Error Cuadrático Medio, Nj es el número de registros en el nodo hijo j y N es el número de registros en el nodo padre.

Podando los árboles...

- El proceso anterior puede producir buenas predicciones en el training set pero es **probable que haga un overfit a los datos de entrenamiento**. Esto ocurre porque el árbol resultante puede ser demasiado complejo.
- Un **árbol más pequeño** con menos divisiones (menos regiones R₁, .. R_J) podría tener **menos varianza** y ser más interpretable al costo de un poco **más de sesgo**.

Pre-prada:

Una técnica para evitar el overfitting es limitar el crecimiento del árbol. Esta técnica se llama **pre-poda**.

Posibles criterios de pre-poda:

- Reducción mínima de impureza
- Establecer un límite máximo a la profundidad
- Establecer un número mínimo de observaciones por hoja
- Establecer un número máximo de hojas

Post-poda:

- Otra técnica consiste en armar el árbol más grande posible T_0 y luego podarlo (post-poda).
- En lugar de considerar cada posible subárbol podríamos incluir una penalización por la complejidad regulada por un parámetro de tuning alpha no negativo. Para cada alpha se corresponde un subárbol $T \subset T_0$ tal que:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

Post-podando los árboles...

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- Aquí $|T|$ indica el número de nodos terminales del árbol. El parámetro de tuning alpha controla el trade-off entre la complejidad del subárbol y su ajuste a los datos de entrenamiento.
- Con alpha= 0, el subárbol T será igual a T0. Al aumentar alpha el precio a pagar por tener más nodos terminales aumenta por lo que la expresión anterior tenderá a ser minimizada para un subárbol más pequeño. ¿Lasso?
- Podemos elegir alpha por CV y luego usamos el data set entero y obtenemos el subárbol correspondiente a alpha.

Ventajas y Desventajas

Ventajas

- Los árboles son muy **fáciles de explicar a las personas.**
- Parecen más cercanos a la forma en la que las personas toman decisiones
- Pueden ser representados gráficamente, pueden ser interpretados por no expertos fácilmente (especialmente si son pequeños)
- Pueden **manejar fácilmente predictores cualitativos sin necesidad de crear variables dummy.**

Desventajas

- En general **no tienen el mismo nivel de precisión** en la predicción comparados con otros enfoques para regresión y clasificación vistos previamente.
- Además **pueden ser poco robustos**. Un pequeño cambio en los datos puede generar un gran cambio el árbol final estimado. Sin embargo **al agregar muchos árboles de decisión usando métodos como bagging, random forests, y boosting la performance predictiva de los árboles puede mejorarse sustancialmente.**