K-Nearest Neighbours y Árboles de Decisión

Introducción a Machine Learning
Modelos no paramétricos

Modelos paramétricos vs. no paramétricos

En general, los modelos de aprendizaje pueden dividirse en dos grandes grupos(esta clasificación no es exacta y puede ser un poco ambigua):

- **Paramétricos:** Cantidad de parámetros **no dependen** del tamaño de la muestra con la que estemos trabajando.
- **No paramétricos:** la cantidad de parámetros que definen al modelo no están fijos y pueden depender del tamaño de la muestra

Los modelos vistos hasta ahora eran **modelos paramétricos**, entre ellos:

- Regresión lineal
- Regresión logística
- Redes neuronales

Hoy veremos dos modelos **no paramétricos**: *árboles de decisión* y *k-vecinos más cercanos*.

K-Nearest Neighbours

KNN - Introducción

K-Nearest Neighbours (KNN) o **K vecinos más cercanos** es un modelo no paramétrico para clasificación y regresión.

La idea detrás de KNN es utilizar los **k** datos más cercanos (**vecindad**) para predecir datos nuevos haciendo una votación/promedio de sus vecinos.

KNN es un algoritmo *basado en las instancias*, es decir que memoriza los datos y las distancias entre ellos.

En este algoritmo el hiperparámetro **k** es el que definirá la cantidad de vecinos a utilizar.

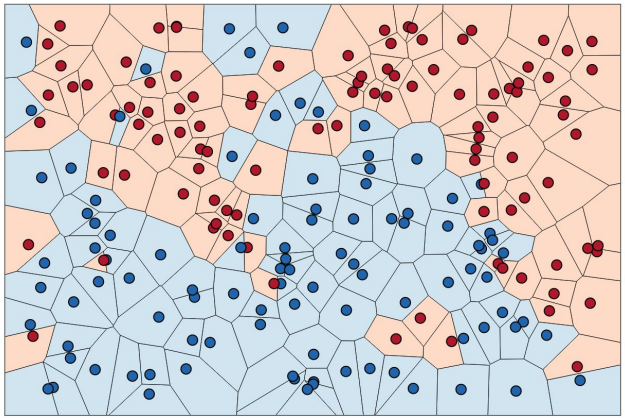
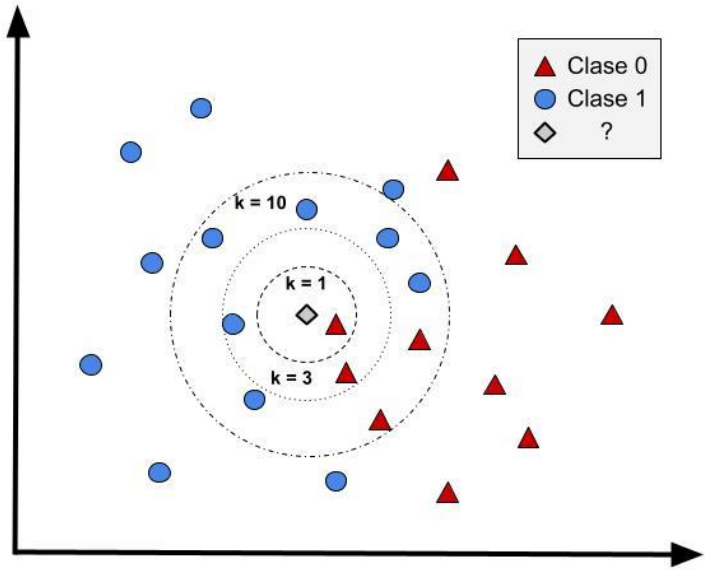


Diagrama de Voronoi. Fuente: Kevin Zakka's Blog

KNN - Ejemplo

El esquema de KNN para predecir la etiqueta **Y** una instancia de test **X**:

1. Calcular la distancia de todos los datos al dato **X**
2. Determinar los **k** datos de entrenamiento más cercanos a **X** (**vecindad**)
3. Etiquetar **X** usando algún criterio por ejemplo, predecir la moda (mayoría) de etiquetas (clasificación) o el promedio de los target **Y** (regresión) de la **vecindad**



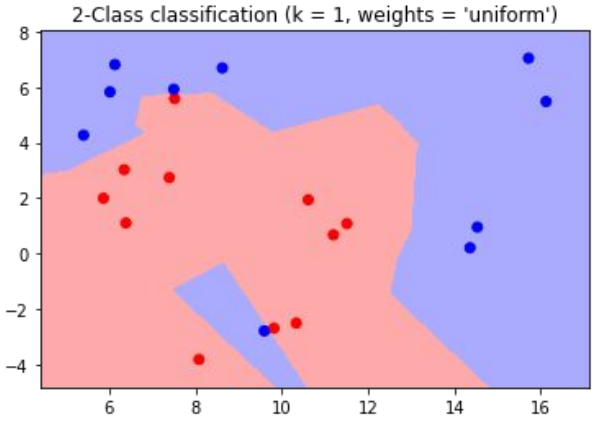
Con k = 1

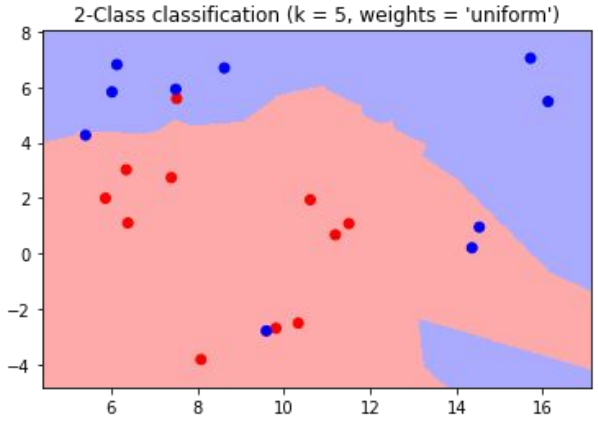
Con k = 3

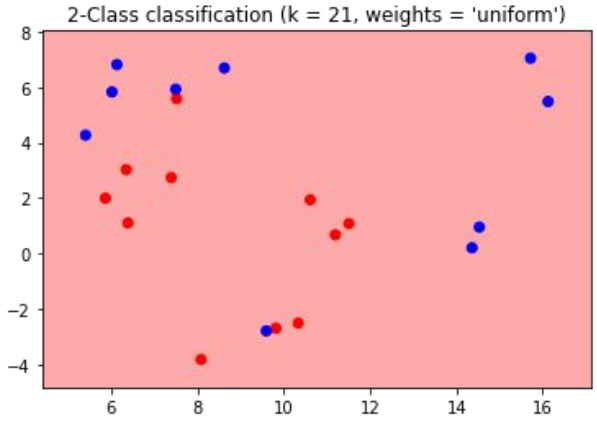
Con k = 10

KNN - El rol de *k* y la frontera de decisión

El valor de **k** regula el trade-off entre sesgo y varianza modificando la frontera de decisión.
¿Que pasa si elegimos k=n el tamaño de la muestra? ¿Y si k=1?







+ Varianza
- Sesgo

- Varianza
+ Sesgo

KNN - Haciendo predicciones

Una vez que determinamos el vecindario para poder hacer la predicción tenemos dos principales estrategias:

Clasificación

La instancia nueva la podemos predecir cómo la etiqueta más frecuente en el vecindario donde todos votan con peso 1 - *uniform weights.*

Podemos hacer una votación pesada por la distancia del vecino x_i al dato nuevo x - *distance weights* - cada instancia pesa entonces: $\frac{1}{d(x,x_i)}$

Regresión

uniform weights:

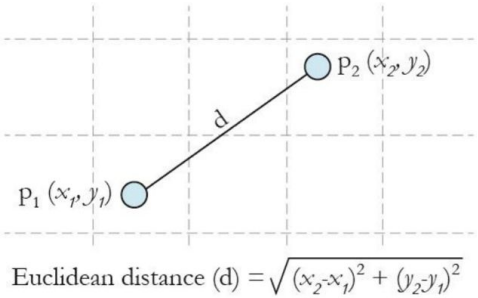
$$\hat{y} = \frac{1}{k} \sum_{i \in Vec(x)} y_i$$

distance weights:

$$\hat{y} = \frac{1}{k} \sum_{i \in Vec(x)} \frac{y_i}{d(x,x_i)}$$

KNN - La elección de la distancia

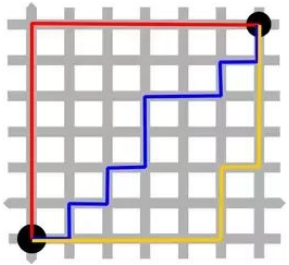
La distancia más utilizada es la distancia **Euclídea**:



Sin embargo, existen muchas otras distancias que podemos usar entre ellas:

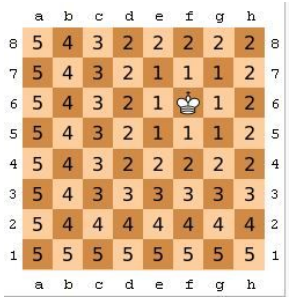
Manhattan o distancia City-Block

$$\sum_{i=1}^{m} |x_i - y_i|$$



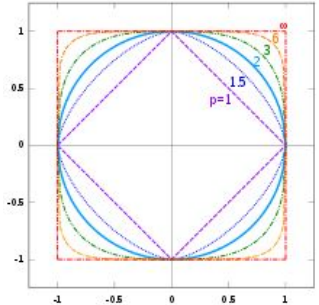
Chebyshev

$$\max_{i=1,\ldots,m} \{|x_i - y_i|\}$$



Minkowski *p*

$$\sqrt[p]{\sum_{i=1}^{m} |x_i - y_i|^p}$$



Fuente imágenes: Wikipedia Commons

La elección de la distancia depende de muchos factores entre ellos depende si tenemos datos en la misma escala o no, de la dimensionalidad del problema, del tipo de dato de cada feature, etc.

Ver más en: sklearn.neighbors.DistanceMetric — scikit-learn 0.24.2 documentation.

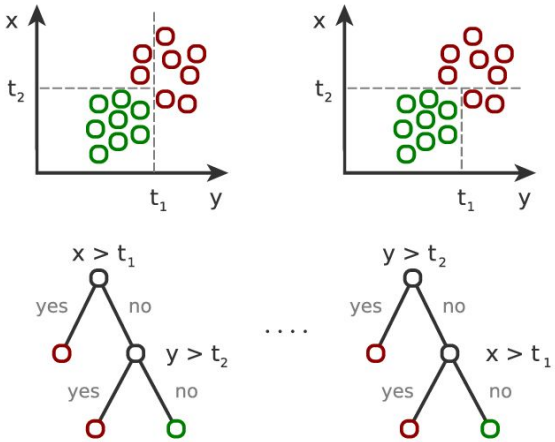
KNN - Ventajas y desventajas

- Ventajas
  - Casi no utilizamos suposiciones de los datos, sólo necesitamos determinar *k* y la medida de distancia
  - Fácil de actualizar en cuando el modelo está en producción: solo necesitamos agregar un nuevo elemento al conjunto de capacitación (no necesitamos entrenamiento)
- Desventajas
  - Necesito manejar los valores perdidos
  - Sensible a valores atípicos y datos de entrenamiento mal etiquetados
  - Sensible a atributos irrelevantes, ya que afecta el cálculo de la distancia
  - Costo computacional:
    - Espacio: es necesario almacenar todos los ejemplos de entrenamiento
    - Tiempo: es necesario calcular la distancia a todos los ejemplos *O (nd)*, donde n= número de conjunto de entrenamiento y d = costo de la distancia de cálculo (el cálculo es pesado en el conjunto de test *(O (nd))*, no en el entrenamiento *(O (d))*).

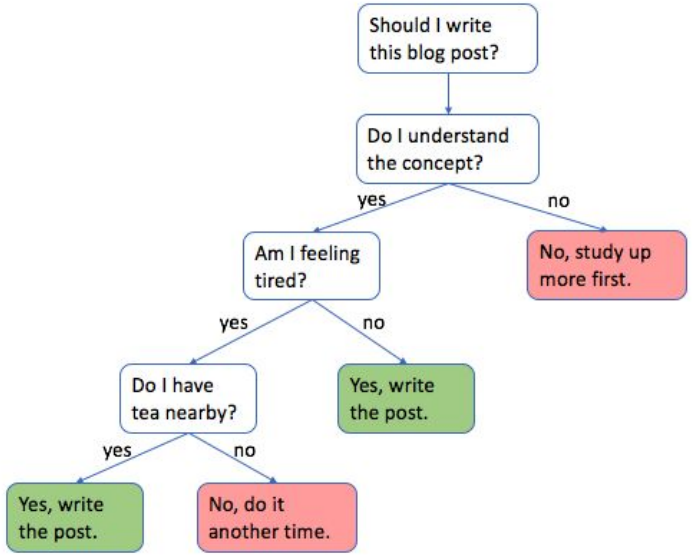
Árboles de Decisión - CARTs

Árboles de decisión - Introducción

Los **árboles de decisión** (o CART por *Classification and Regression Tree*) son modelos no paramétricos que se utilizan tanto para clasificación como regresión. Son modelos altamente interpretables y flexibles.



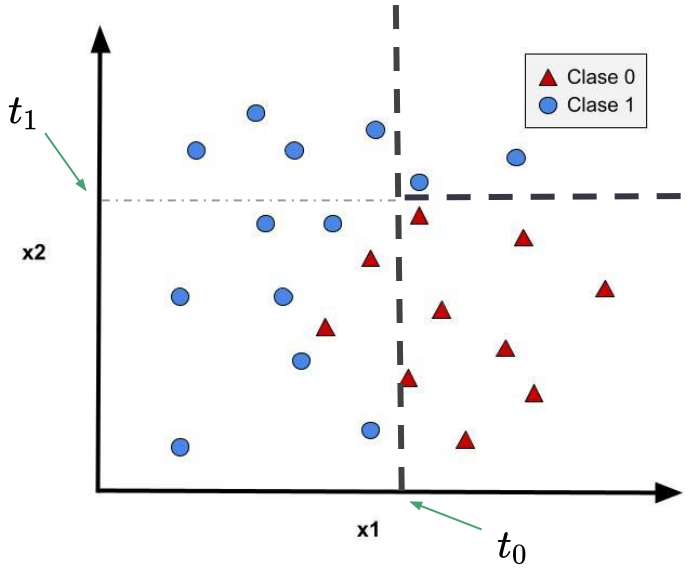
La idea es segmentar los datos en regiones que se van delimitando por cortes ortogonales, utilizando solo un feature a la vez.

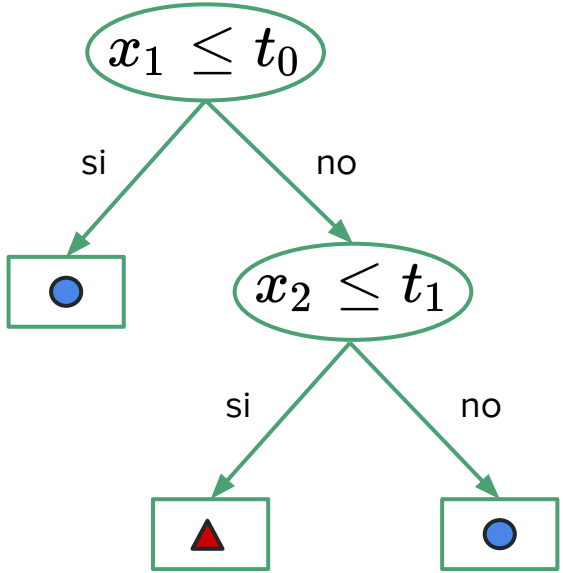


Fuente https://www.jeremyjordan.me/decision-trees/

Fuente: Figure 4: Two exemplary decision trees that form the random forest...

Árboles de decisión - Introducción





Lo que acabamos de realizar recibe el nombre de **Recursive Binary Splitting**, el algoritmo que se utiliza par ir construyendo el árbol.

Árboles de decisión - Entrenamiento

Para realizar la segmentación del espacio *recursive binary splitting* nos propone ir separando la muestra en cada nivel del árbol en dos conjuntos diferentes.

Necesitamos entonces un criterio para decir que un *corte* o *split* es bueno. Para eso en clasificación utilizaremos la noción de *impureza* para cuantificar qué tan mezcladas están las distintas clases (mayor valor de impureza es pero). Las dos maneras que utilizaremos son la **Entropía** y el **índice de Gini**.

Entropía

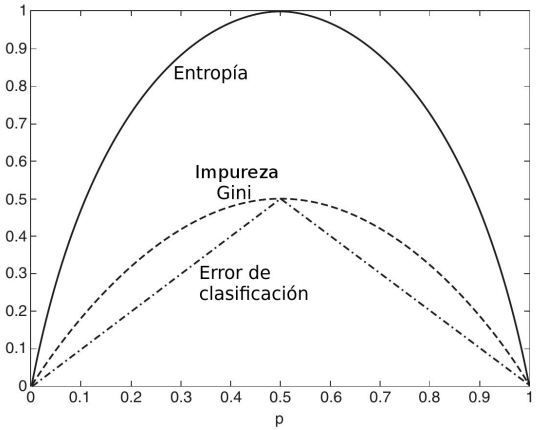
$$Entropia(S) = \sum_{c \in Clases(S)} -p_c \cdot \log_2(p_c)$$

$$\text{donde } p_c = \frac{|\{x \in S, clase(x) = c\}|}{|S|}$$

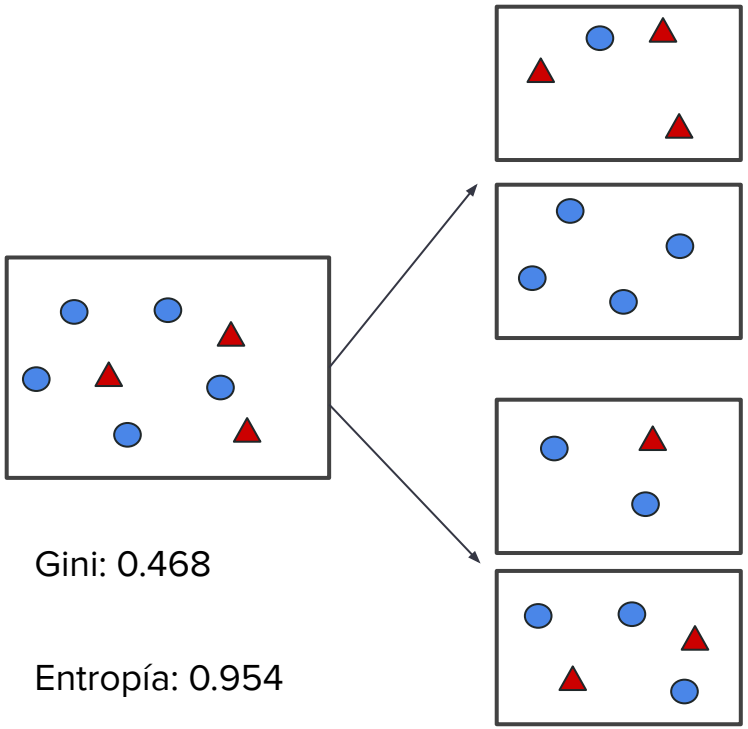
Índice de Gini

$$Gini(S) = \sum_{c \in Clases(S)} p_c(1 - p_c)$$

$$= 1 - \sum_{c \in Clases(S)} p_c^2$$



Árboles de decisión - Índice de gini y entropía



Gini: 0.468

Entropía: 0.954

Gini:
$1-(1/4)^2-(3/4)^2= 3/8$

Entropía:
$-(1/4).\log(1/4)-(3/4).\log(3/4)=$
$= 0.811$

Gini:
$1-(4/4)^2-(0/4)^2=0$

Entropía:
$-(4/4).\log(4/4)-(0/4).\log(0/4)=$
$= 0$

Gini:
$1-(2/3)^2-(1/3)^2= 4/9$

Entropía:
$-(2/3).\log(2/3)-(1/3).\log(1/3)=$
$= 0.918$

Gini:
$1-(3/5)^2-(2/5)^2=12/25$

Entropía:
$-(3/5).\log(3/5)-(2/5).\log(2/5)=$
$= 0.971$

Árboles de decisión - Information Gain

Para determinar cuál es el mejor corte queremos *minimizar la impureza*. Podemos medir la reducción de la impureza usando la entropía como:

Entropía(nodo padre) - Promedio pesado(Entropía (nodos hijos))

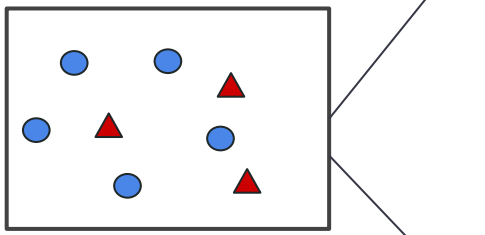
Esto se conoce como **Ganancia de Información** (Information Gain), llamemos V al nodo padre con N datos, X al feature, s al corte en el feature X (X < s), entonces:

$$IG(X_j, s) = Entropy(V) - \sum_{V_i} \frac{N_i}{N} Entropy(V_i)$$

$$V_i = \{X \in V | X_j \leq s\} \text{ o } \{X \in V | X_j > s\}$$

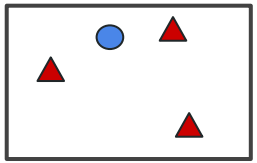
Un mayor valor de information gain implica una mayor reducción de entropía. Si se usa el índice de Gini en vez de entropía se suele llamar **Ganancia de Gini**.

Árboles de decisión - Índice de gini y entropía



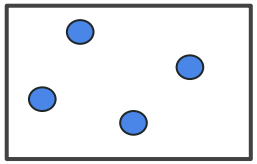
Gini: 0.468

Entropía: 0.954



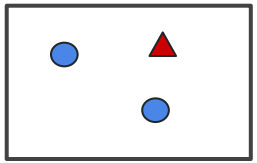
Gini:
$1-(1/4)^2-(3/4)^2= 3/8$

Entropía: 0.811



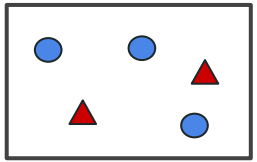
Gini:
$1-(4/4)^2-(0/4)^2=0$

Entropía: 0



Gini:
$1-(2/3)^2-(1/3)^2= 4/9$

Entropía: 0.918



Gini:
$1-(3/5)^2-(2/5)^2=12/25$

Entropía: 0.971

Mejor corte

Information Gain:
$0.954-1/2*0.811-1/2*0$
$= 0.5485$

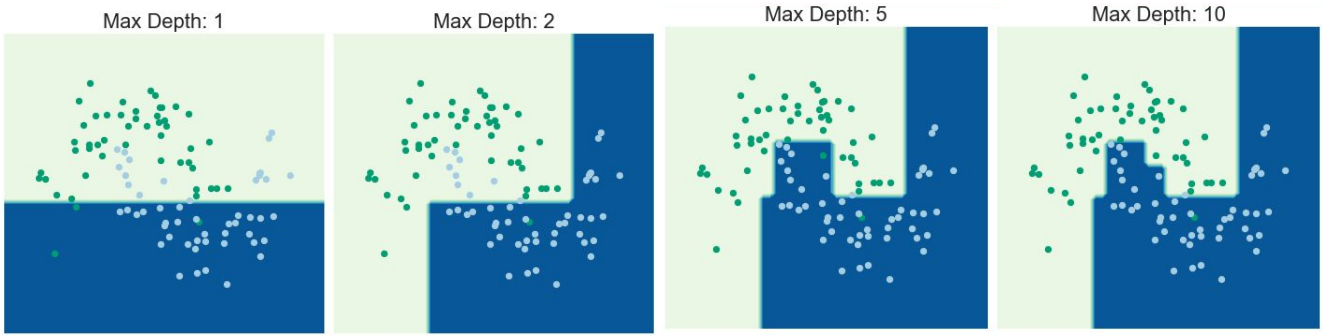
Information Gain:
$0.954-3/8*0.918-5/8*0.917$
$= 0.0366$

Árboles de decisión - Entrenamiento

Entonces podemos tener una idea de cómo se entrenará un árbol:

- Para cada feature se probarán distintos cortes (como máximo tantos como datos en entrenamiento)
- Se elige el mejor split con Information Gain
- Se chequea condiciones de parada:
  - Cota de profundidad máxima
  - Mínimo de datos para realizar un split
  - Mínimo de datos en una hoja para considerar un split

Sin la condición de parada, el árbol crecerá hasta separar cada uno de las instancias de entrenamiento de forma perfecta, generando un modelo mas complejo.



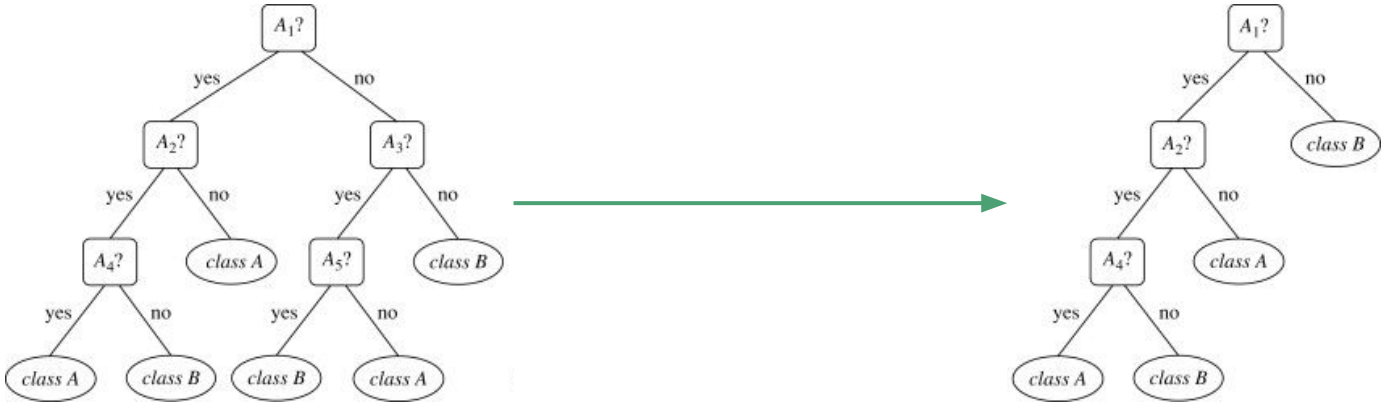
Fuente: https://www.programmersought.com/article/17315849174/

Árboles de decisión - Poda y overfitting

La profundidad del árbol nos controlará la complejidad del modelo, sirviendonos como control para el trade-off entre sesgo y varianza.

Sin embargo, tenemos otra forma de reducir el overfitting de un árbol ya entrenado: las técnicas de **post-poda** o **post-pruning**.

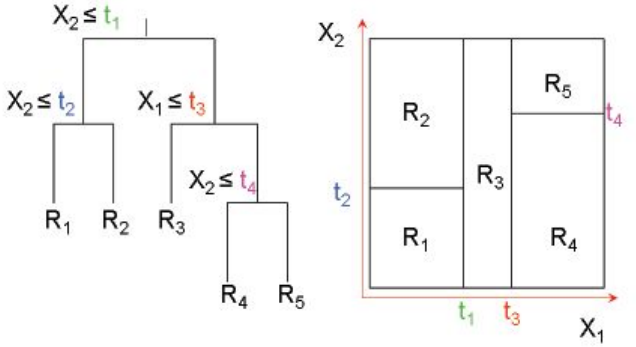
Estas técnicas penalizarán modelos más complejos y cortarán las hojas menos necesarias del árbol mejorando su generalización. Uno de los algoritmos más utilizados para tal efecto es el **Minimal Cost-Complexity Pruning**.



Árboles de decisión - Regresión

Para el caso de regresión lo que tratamos de minimizar es el **Error Cuadrático Medio**, aproximando al target *y* como el promedio de cada una de las regiones *R* determinadas por el árbol

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



Fuente: Hastie et al., 2009

Para evaluar qué tan bueno es un corte queremos elegir el que minimice el error cuadrático

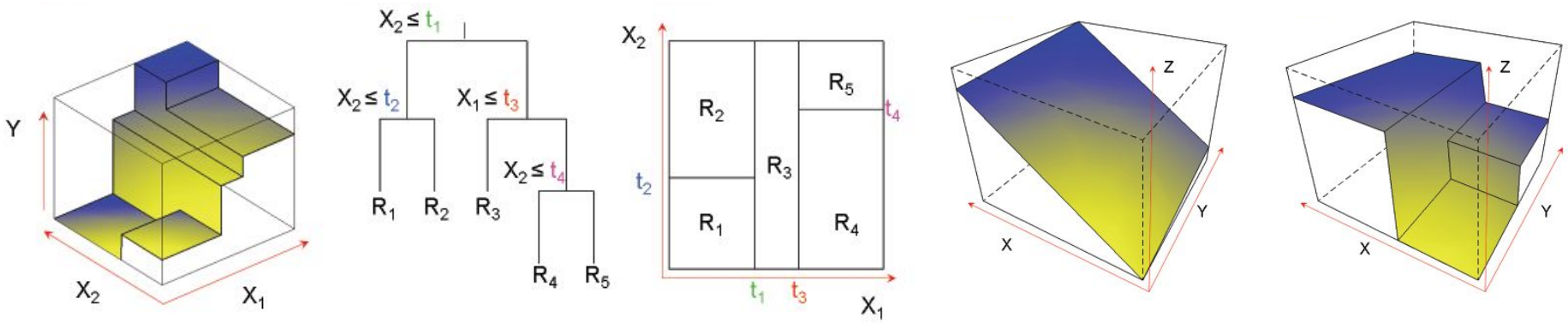
$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

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

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

Árboles de decisión - Regresión

Finalmente el Árbol de Regresión tiene la forma de una función escalonada



Fuente: Hastie et al., 2009

Fuente: Carey et al., 2005

Árboles de decisión - Ventajas y desventajas

- Ventajas
  - Modelos interpretables
  - Rápido entrenamiento
  - Pueden manejar distintos tipos de datos: continuos o discretos (dependiendo de la implementación) y datos faltantes
- Desventajas
  - Tienden al overfitting si no se limita el crecimiento del árbol
  - Tienen un rendimiento generalmente inferior a otros modelos clásicos ya que les cuesta encontrar fronteras de decisión que no sean ortogonales a los ejes de coordenadas