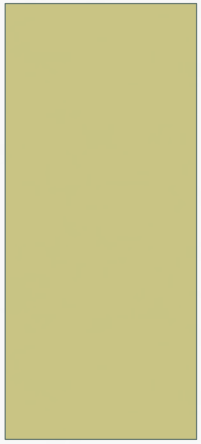
uc3m | Universidad Carlos III de Madrid

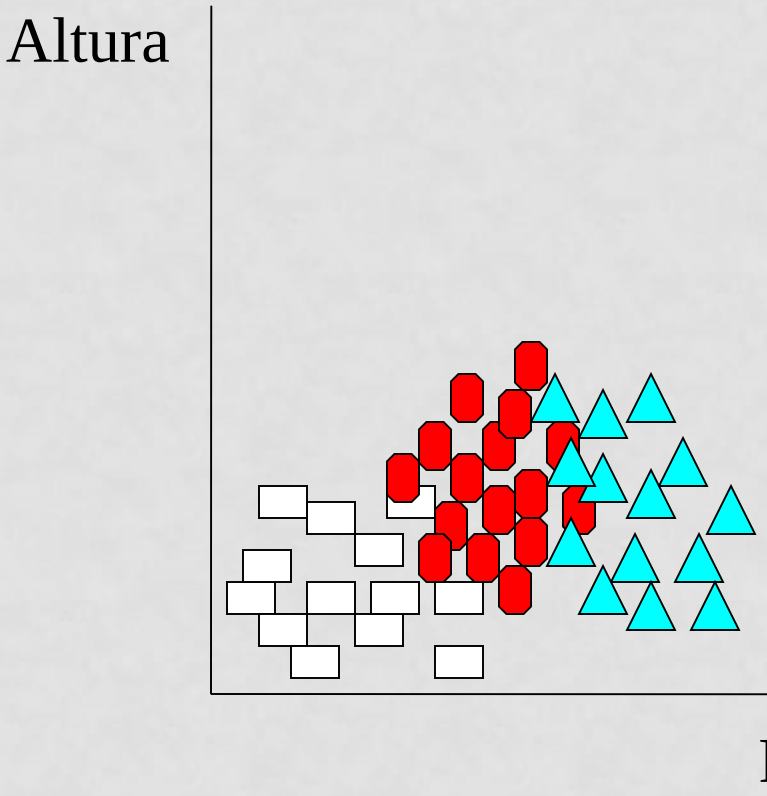
OPENCOURSEWARE
APRENDIZAJE AUTOMÁTICO PARA EL ANÁLISIS DE DATOS
GRADO EN ESTADÍSTICA Y EMPRESA
Ricardo Aler

VECINO(S) MAS CERCANO(S)

KNN = k-nearest neighbour

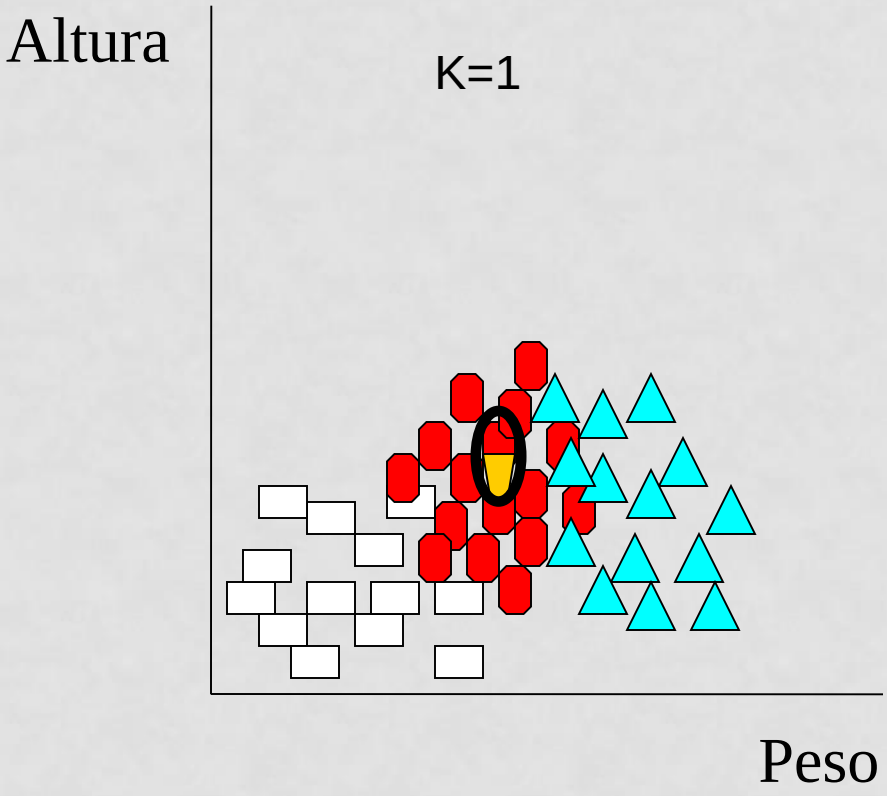


K NEAREST NEIGHBORS (KNN)



Se guardan todos los ejemplos (lazy)

K NEAREST NEIGHBORS (KNN)



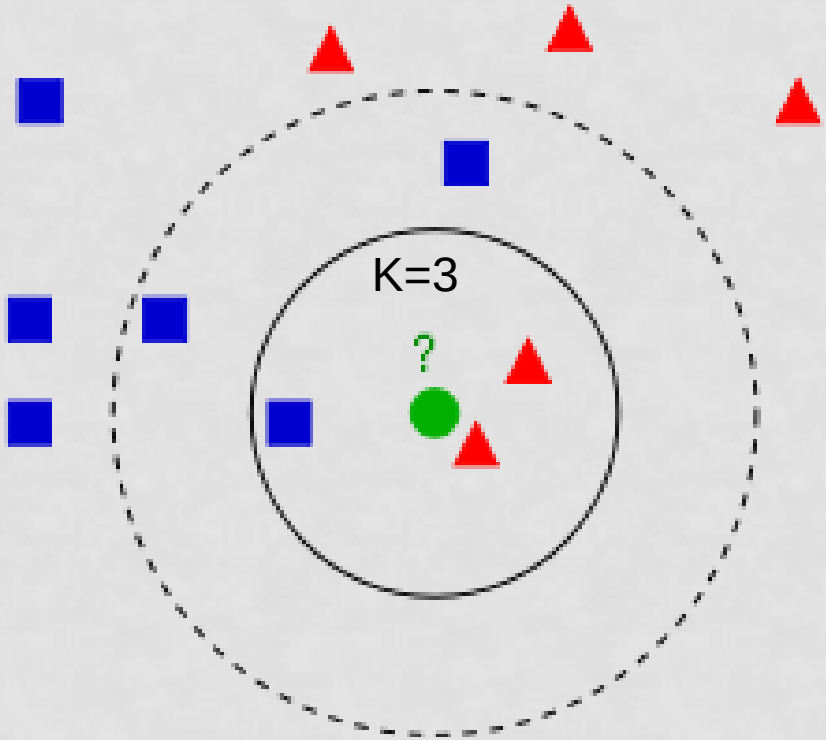
Niño

Adulto

Mayor

Algoritmo “lazy” (perezoso):
No se construye un modelo,
Simplemente se guardan todas las instancias

K NEAREST NEIGHBORS (KNN)



Source: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

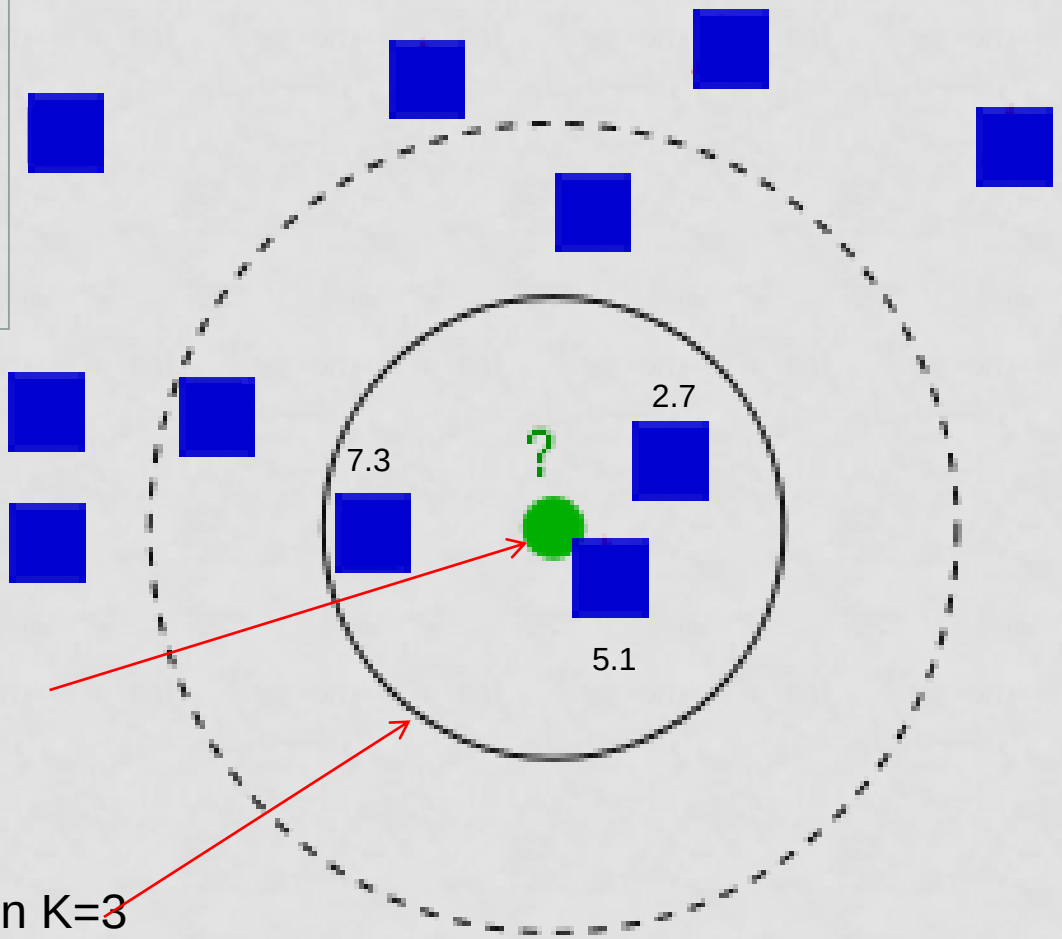
- Para evitar que los vecinos lejanos tengan mucha influencia, se puede hacer que cada vecino vote de manera inversamente proporcional a la distancia 1/d

CARACTERISTICAS KNN

- KNN: Clasifica nuevas instancias como la clase mayoritaria de entre los k vecinos mas cercanos de entre los datos de entrenamiento
- KNN es un algoritmo perezoso (lazy)
  - Durante el entrenamiento, sólo guarda las instancias, no construye ningún modelo (a diferencia de, por ejemplo, los árboles de decisión)
  - La clasificación se hace cuando llega la instancia de test

KNN PARA REGRESIÓN

- Calcular la media de los K vecinos
- Para que las instancias más lejanas tengan menos importancia, se puede hacer una media ponderada por 1/d
- Se puede construir un modelo lineal con los K vecinos



Predicción = (7.3+2.7+5.1)/3

Vecindad con K=3

Source: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

MIDIENDO LA SIMILARIDAD

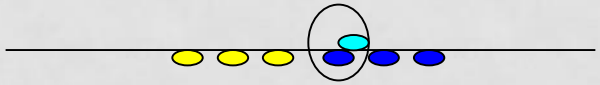
- Normalmente se usa la distancia Euclidea:
  - En 2D: $d(\mathbf{x_i},\mathbf{x_j})^2 = (x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2$
  - En dD: $d(\mathbf{x_i},\mathbf{x_j})^2 = (x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \ldots + (x_{id}-x_{jd})^2$
- Es necesario normalizar los atributos para que atributos con mucho rango no tengan mas peso que los demás (preproceso):
  - Normalización: $x'_{1j} = (x_{1j}-min_j)/(max_j-min_i)$
  - Estandarización: $x'_{1j} = (x_{1j}-\mu_j)/\sigma_j$
- Si los atributos son nominales, usar distancia de Hamming:
  - Si el atributo e es nominal (o discreto o categórico), en lugar del componente $(x_{ie}-x_{je})^2$ se usa:
    - $\delta(x_{ie}, x_{je})$: *0* si $x_{ie}=x_{je}$
    - 1 en caso contrario
  - También, variables "dummy" o "one-hot" (preproceso)

Limitaciones de KNN

- Lento, si hay muchos datos de entrenamiento (en almacenamiento y en tiempo):
  - Eliminación de instancias superfluas (fase de preproceso)
- Muy sensible a los atributos irrelevantes:
  - Selección de atributos (fase de preproceso)
- Muy sensible al ruido:
  - Ajuste del hiper-parámetro del número de vecinos (K)

Atributos irrelevantes

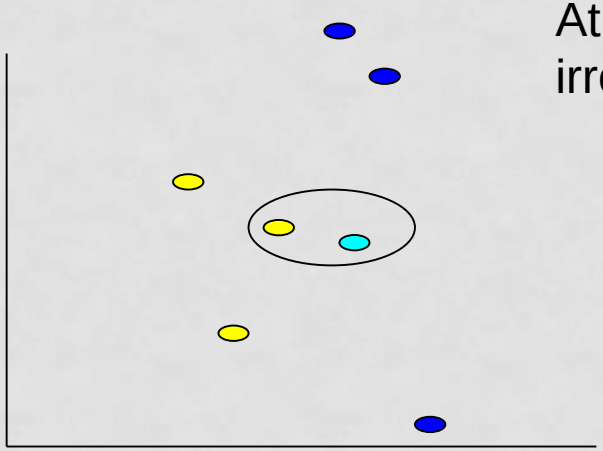
0 atributos irrelevantes



Con el atributo relevante, se clasifica bien

Atributo irrelevante

Atributo irrelevante



Con el atributo irrelevante, se clasifica mal (las distancias cambian)

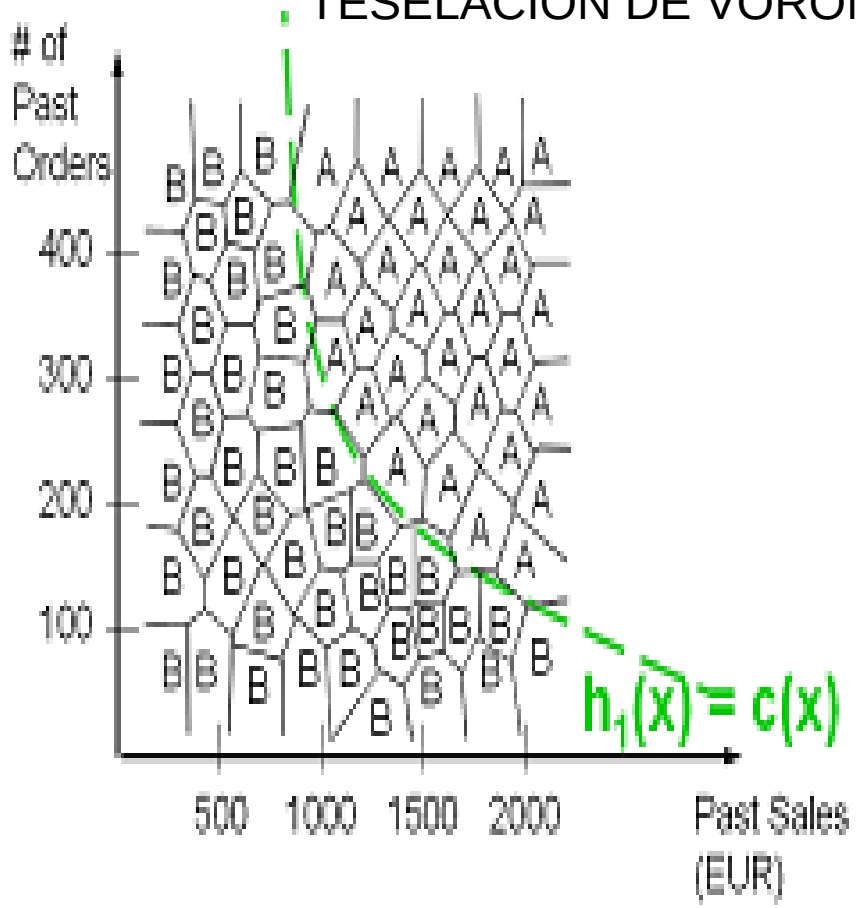
1 atributo irrelevante

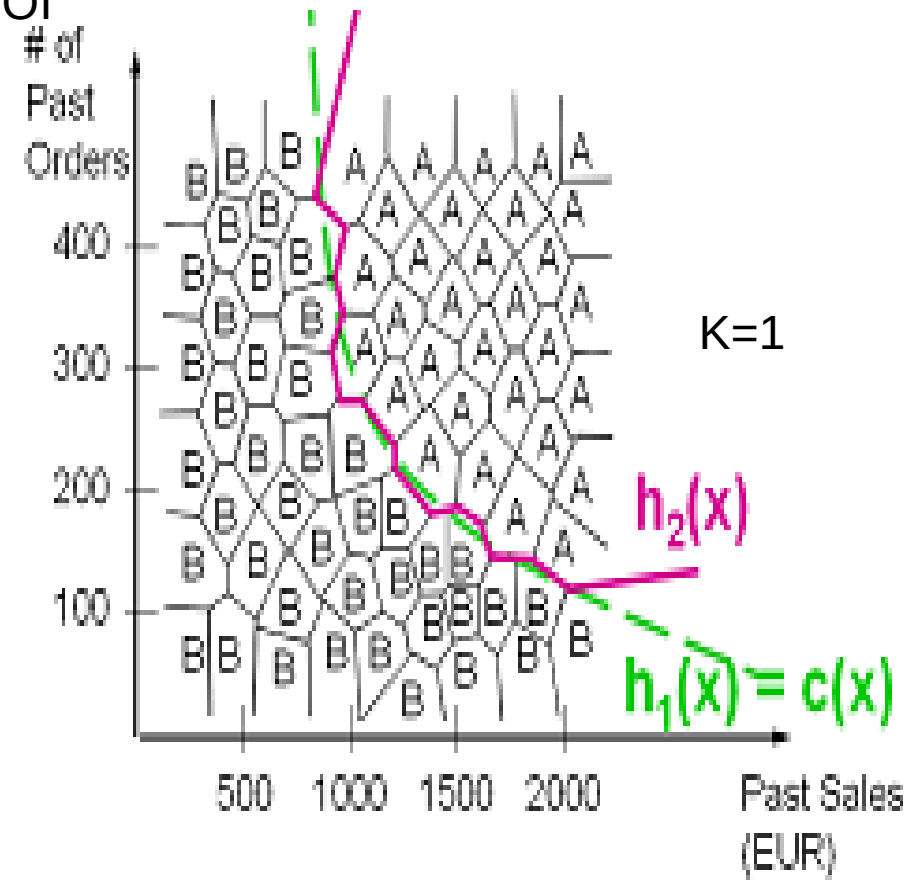
Atributo relevante

Vecino mas cercano k=1

INFLUENCIA DE DATOS CON RUIDO

TESELACIÓN DE VORONOI

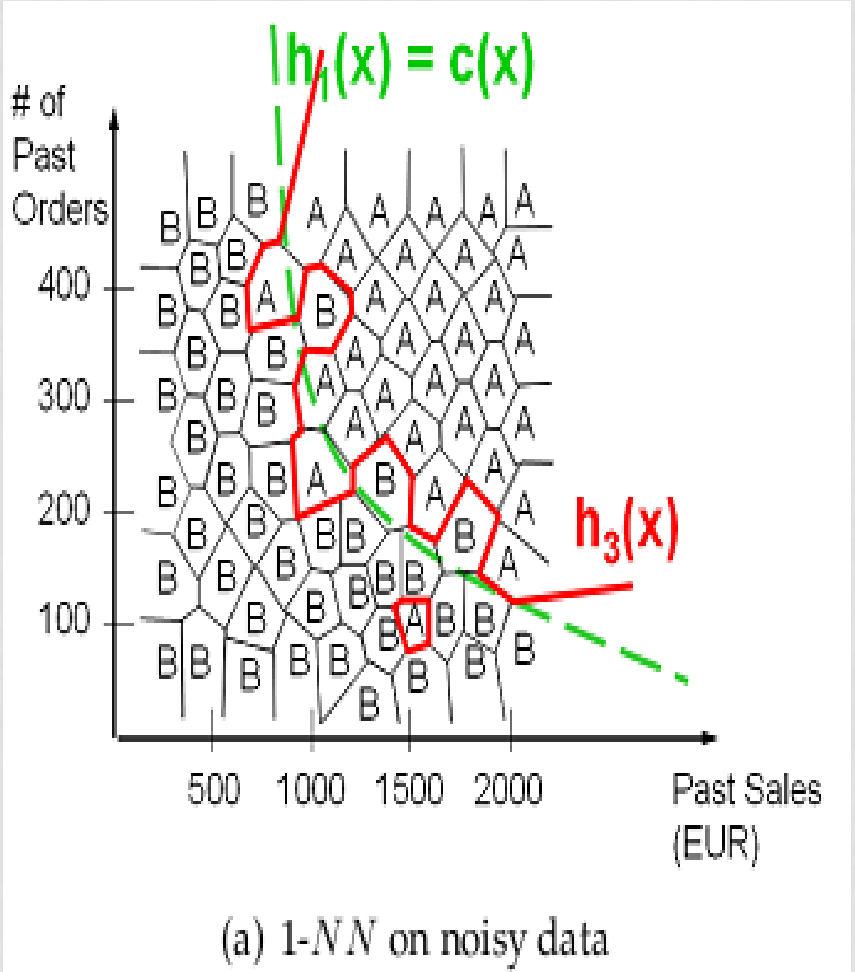




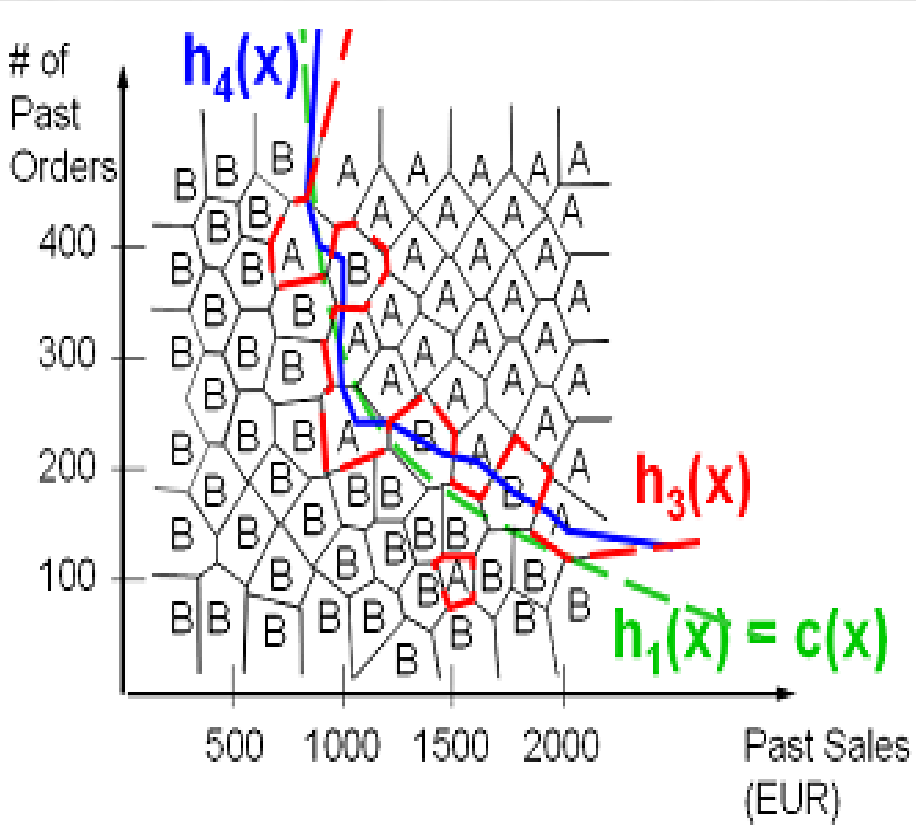
K=1

Source: https://en.wikipedia.org/wiki/Voronoi_diagram

INFLUENCIA DE DATOS CON RUIDO



(a) 1-NN on noisy data



(b) 3-NN and noisy data

Source: https://en.wikipedia.org/wiki/Voronoi_diagram

¿VALORES DE K?

- K es el hiper-parámetro principal de KNN
- Con K=1, las instancias que son ruido (o solape entre clases) tienen mucha influencia
- Con K>1, se consideran mas vecinos y el ruido pierde influencia (es como hacer una promediado)
- Si k es muy alto, se pierde la idea de localidad
  - ¿En que se convierte KNN si K == número de datos?
- Si hay dos clases, usar K impar para deshacer empates