# K - Vecinos más cercanos

(K-Nearest Neighbours, KNN)

Sirve: V. CATEGÓRICAS → CLASIFICACIÓN
V. NUMÉRICAS → REGRESIÓN

Training set $(X \mid y)$

FEATURES → $X = \begin{pmatrix} \vec{x}^{(1)} \\ \vdots \\ \vec{x}^{(m)} \end{pmatrix} = \begin{pmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ x_1^{(m)} & \cdots & x_d^{(m)} \end{pmatrix}$

TARGET → $y = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$

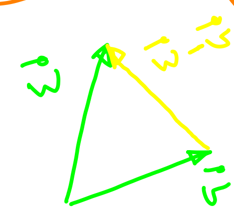
Considero: $\vec{x}^{(1)}, \dots, \vec{x}^{(m)}$ son puntos en $\mathbb{R}^d$

$\vec{x}^{(1)} = (x_1^{(1)}, \dots, x_d^{(1)})$

Distancia:

$\vec{v} = (v_1, \dots, v_d)$

$\vec{w} = (w_1, \dots, w_d)$




$$d(\vec{v}, \vec{w}) = \sqrt{(v_1-w_1)^2 + \dots + (v_d-w_d)^2} = \|\vec{v}-\vec{w}\|$$

Test: $\vec{x} \rightsquigarrow$ ¿$\hat{y}$?

$N_K(\vec{x}) = \{\vec{x}^{(1)}, \dots, \vec{x}^{(K)}$ los más cercanos a $\vec{x}\}$

Sus K vecinos más cercanos


CLASIFICACIÓN: Si $y$ es categórica predigo

$\hat{y}$ = El más común en $N_K(\vec{x})$

REGRESIÓN: Si $y$ es numérica predigo el promedio de los vecinos:

$$\hat{y} = \frac{1}{K} \sum_{\vec{x}^{(i)} \in N_K(\vec{x})} y_i$$



- **Lazy learning**

Lazy learning: Se realiza la predicción en el momento que aparecen los test data y no antes (la predicción se demora hasta que se hace una pregunta al modelo).

→ Se aproxima localmente

Problemas:

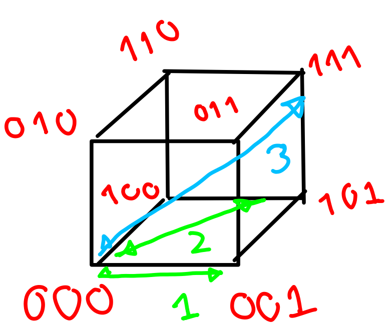
1. Distancia: ¿Qué pasa con las features que no están en $\mathbb{R}^d$?

Por ejemplo, ¿si son categóricas?

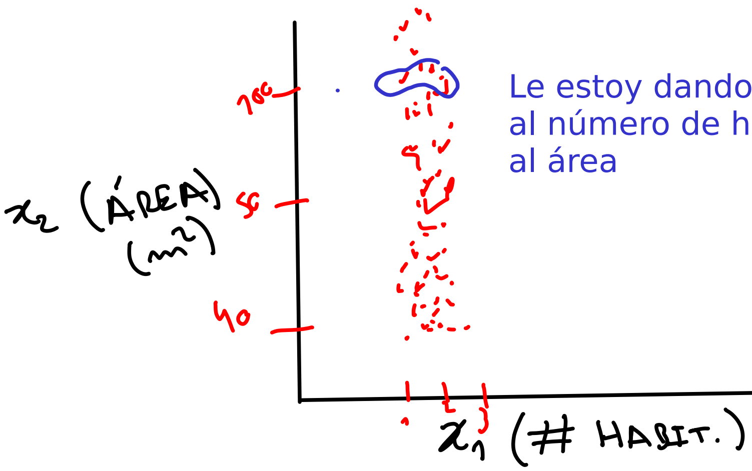
└→ Dummies (binarias)

**Hamming distance:**

La distancia entre dos números binarios es el número de bits distintos



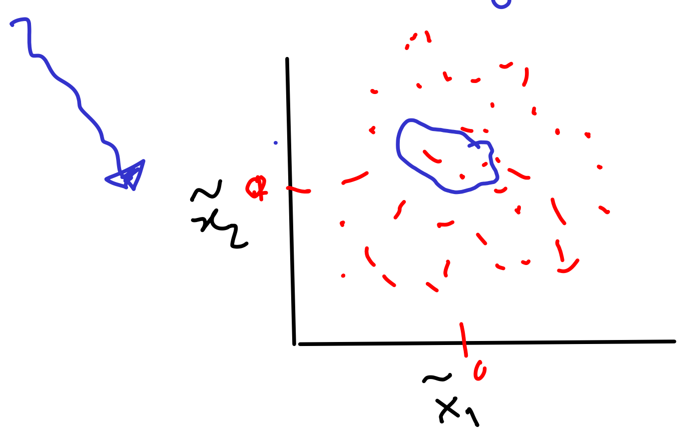
2 Normalización: ¿Qué pasa si tengo features en unidades distintas?




Le estoy dando más importancia
al número de habitaciones que
al área

Solución:

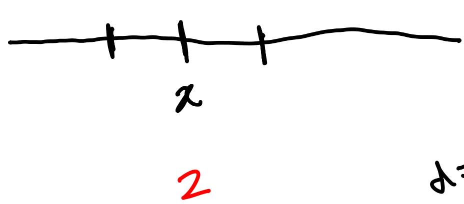
Reescalar a $\mu = 0$, $\sigma = 1$



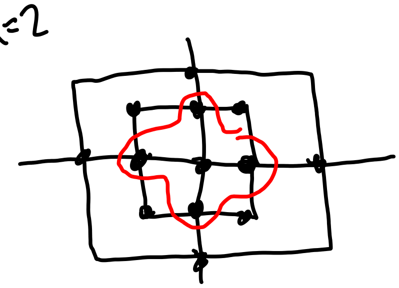

d = # features

3. Muchas features: El número de vecinos aumenta y las predicciones se vuelven más aleatorias

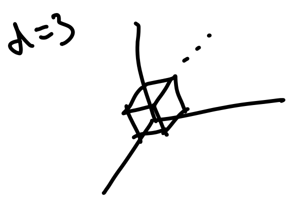
d=1




d=2



d=3



Para aplicar KNN se recomienda usar algoritmos de reducción de dimensión (p. ej. PCA)

Mejoras: $N_K(\vec{x}) = \{\vec{x}^{(1)}, \dots, \vec{x}^{(n)}\}$

$$d(\vec{x}^{(i)}, \vec{x}) = d_i$$

PESOS

$$w_i = \frac{1}{d_i}$$

$\boxed{\text{VOTO PONDERADO}}$

El error en KNN:

Error de predicción esperado:

$$Err = E\left[(y - \hat{y})^2\right]$$

Si suponemos que el $y$ real es de la forma $y = f(\vec{x}) + \varepsilon$, para cierta función $f$ y un error aleatorio $\varepsilon \sim N(0, \sigma_\varepsilon)$

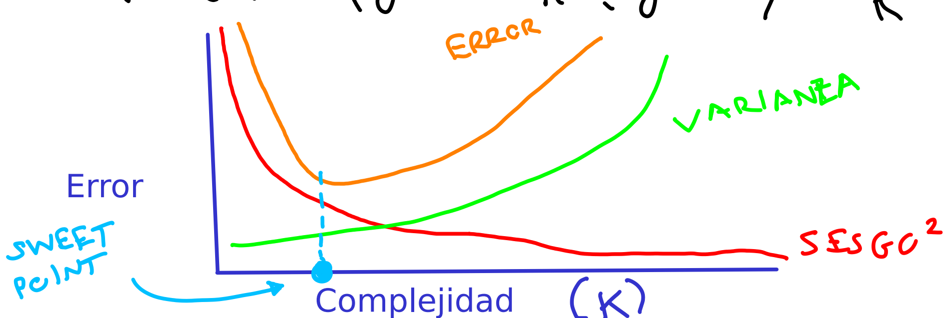
$$Err(\vec{x}) = \underbrace{\left(E[\hat{y}] - f(\vec{x})\right)^2}_{\text{SESGO (BIAS)}} + \underbrace{E\left[\left(\hat{y} - E[\hat{y}]\right)^2\right]}_{\text{VARIANZA DE } \hat{y}} + \underbrace{\sigma_\varepsilon^2}_{\text{ERROR ALEATORIO}}$$

En KNN: $\hat{f}(\vec{x}) = \frac{1}{K} \sum_{i=1}^{K} f(\vec{x}^{(i)})$

$$Err(\vec{x}) = \left(f(\vec{x}) - \frac{1}{n} \sum_i f(\vec{x}^{(i)})\right) + \frac{\sigma_\varepsilon^2}{K} + \sigma_\varepsilon^2$$

Bias-Variance

TRADE OFF



Comparación entre Regresión Lineal y KNN

| RL | KNN |
| --- | --- |
| Varianza baja | Varianza alta |
| Sesgo bajo | Sesgo alto |
| | Depende de K |

$$\hat{y} = \vec{\theta} \cdot \vec{x} = \theta_1 x_1 + \ldots + \theta_n x_n$$

Se asume que y se puede aproximar por una función lineal

$$\hat{y} = \frac{1}{K} \sum_{N_k(x)} y_i$$

Se asume que y se puede aproximar por una función localmente constante