



# K SCHOOL

## Supervised Learning

Javier Cañadillas - [javier@canadillas.org](mailto:javier@canadillas.org)



Esta presentación:

[bit.ly/ks-sl-day1](https://bit.ly/ks-sl-day1)

Notas adicionales:

[bit.ly/ks-sl-notes](https://bit.ly/ks-sl-notes)

## Sobre mi (*shameless self promotion*)

- **Hoy: Google Cloud Customer Engineer**
- Antes: Arquitecto y Especialista diversas áreas en Oracle, HP y Sun
- Profesor invitado Master Data Science profesional UNED 2018
- Profesor asistente IE - School of Human Sciences & Technology
- Profesor UC3M - IoT Software Engineering & PhD student
- Computer Science Master Degree (UC3M)
- Licenciado en Ciencias Físicas - Física Industrial (UNED)
- Ingeniería aeronáutica - Navegación y Transporte Aéreo (UPM)

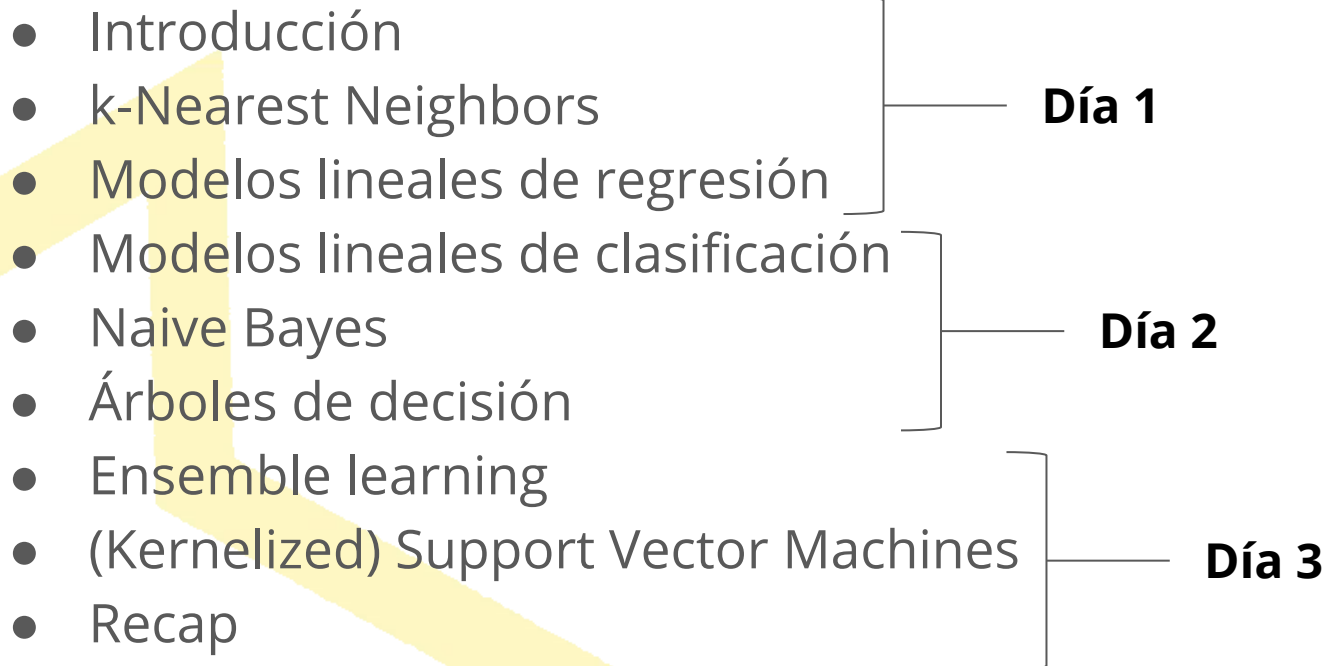
# Temario

- Introducción - Aprendizaje supervisado, regresión y clasificación
- Modelos de aprendizaje supervisado
- Evaluación de modelos - Sobre la marcha
- Laboratorios y ejercicios
- Refrescos, recapitulaciones y conclusiones

## **NOT**(Temario)

- Matemáticas avanzadas y demostraciones
- Ingeniería de características
- Técnicas de descenso de gradiente
- Optimización y evaluación avanzada de modelos

# Agenda y reparto aproximado

- Introducción
  - k-Nearest Neighbors
  - Modelos lineales de regresión
  - Modelos lineales de clasificación
  - Naive Bayes
  - Árboles de decisión
  - Ensemble learning
  - (Kernelized) Support Vector Machines
  - Recap
- Día 1**
- Día 2**
- Día 3**
- 

# Repositorio de código

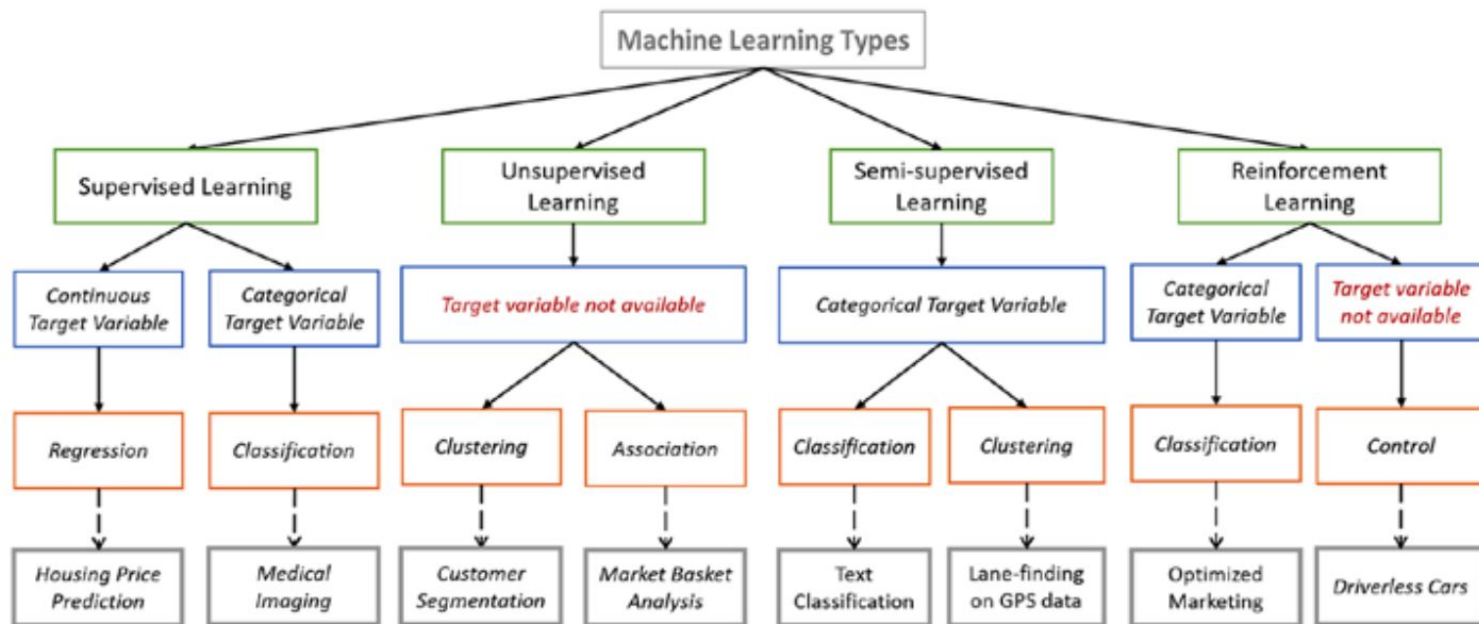
**[gitlab.com/Sh3llD00m/kschool-supervised-learning.git](https://gitlab.com/Sh3llD00m/kschool-supervised-learning.git)**



# 0. Introducción



# Aprendizaje supervisado en ML



# Aprendizaje Supervisado vs No Supervisado

- Tenemos conocimiento previo de valores de salida (labels o etiquetas)
- Objetivo: encontrar función que, dado un dataset de entrada y unas salidas, aproxime de la mejor forma la relación existente entre ellos.
- Tipos de algoritmos supervisados:
  - Clasificación (binaria o multiclase)
  - Regresión (predicción número continuo)

# Motores de reglas ↔ Aprendizaje Supervisado

```
import numpy as np
from sklearn.datasets import make_classification

rs = np.random.RandomState(42)
X, y = make_classification(n_samples = 10, random_state = rs)
```

## Motores de reglas ↔ Aprendizaje Supervisado

```
def tomar_decision_super_importante(X):  
    """  
    Decidir si pasa algo gordo  
    """  
    row_sums = X.sum(axis=1)  
    return (row_sums > 0).astype(int)  
  
tomar_decision_super_importante(X)
```

## Ejemplo de aprendizaje supervisado

```
from sklearn.linear_model import LogisticRegression

def aprender_leccion_vital(X,y):
    """
    Aprender una lección y aplicarla en el futuro
    """
    model = LogisticRegression().fit(X,y)
    return (lambda x: model.predict(x))

# Aprender una lección y aplicarla
decision_informada = aprender_leccion_vital(X,y)(X)
print(decision_informada)
```

## ¿Qué es aprendizaje supervisado?

Es un método de aprendizaje que **aprende una función** a partir de un **conjunto de muestras ya etiquetada** que **aproxima valores futuros de  $y$**

$$\hat{y} = f(X, y; \theta)$$

Valor futuro

Función de  
predicción

Matriz de  
muestras

Vector de  
etiquetas

Parámetros

## Función de coste (*loss function*)

- Cuantifica un coste que el algoritmo minimiza
- Es una medida de lo bien que un modelo se ajusta a un problema

$$L(y, \hat{y}) \in \mathbb{R}$$

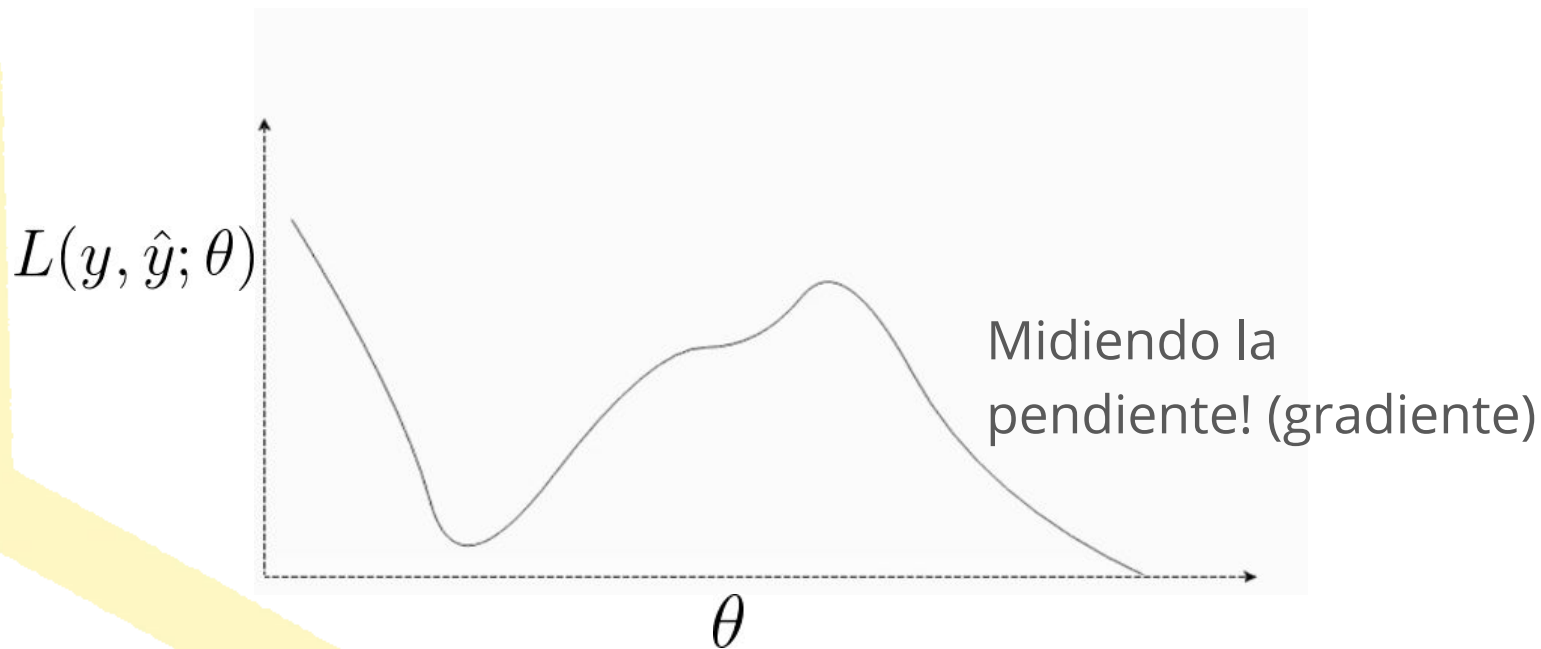
$$L = - \sum_x p(x) \log(1 - p(x))$$

Entropía cruzada (clasificación)

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

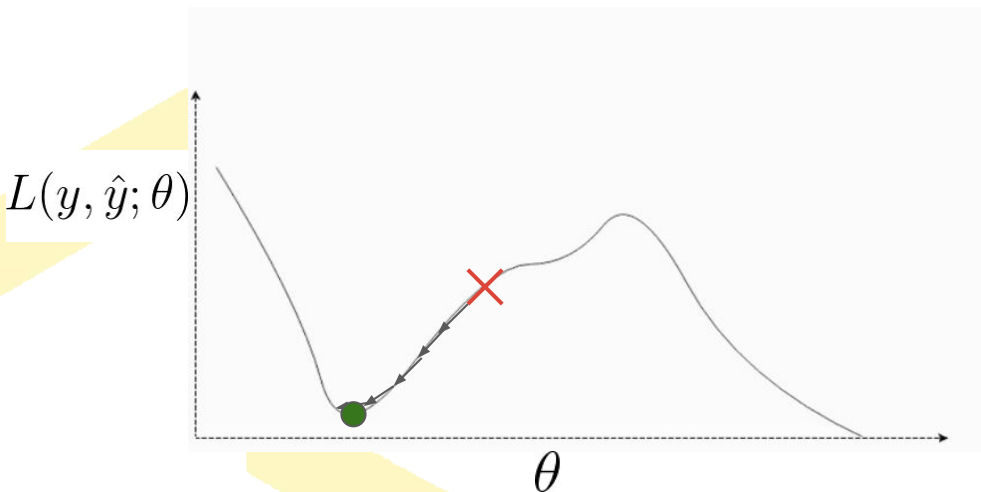
Sumatorio error cuadrático (regresión)

## Queremos minimizar $L$ - ¿cómo?





# Algoritmo general de minimización de gradiente



1. Inicializamos los parámetros/pesos ( $\theta$ ) de manera aleatoria
2. Calculamos el gradiente  $\mathbf{G}$  de la función de coste  $\mathbf{L}$  con respecto a los parámetros
3. Actualizamos los pesos con una cantidad proporcional al gradiente:  $\mathbf{w} = \mathbf{w} - (\text{learning\_rate}) * \mathbf{G}$
4. Repetimos hasta que  $L$  ya no se reduce más o alcanzamos otro criterio de parada.

## ¿Quién querrá comprar un barco?

Edad	# coches en propiedad	Tiene casa	# niños	Estado civil	Tiene perro	Tiene barco
66	1	sí	2	viudo	no	sí
52	2	sí	3	casado	no	sí
22	0	no	0	casado	sí	sí
25	1	no	1	soltero	no	no
44	0	no	2	divorciado	sí	no
39	1	sí	2	casado	sí	no
26	1	no	2	soltero	no	no
40	3	sí	1	casado	sí	no
53	2	sí	2	divorciado	no	sí
64	2	sí	3	divorciado	no	no
58	2	sí	2	casado	sí	sí
33	1	no	1	soltero	no	no

## Modelo 1

Edad	# coches en propiedad	Tiene casa	# niños	Estado civil	Tiene perro	Tiene barco
66	1	sí	2	viudo	no	sí
52	2	sí	3	casado	no	sí
22	0	no	0	casado	sí	sí
25	1	no	1	soltero	no	no
44	0	no	2	divorciado	sí	no
39	1	sí	2	casado	sí	no
26	1	no	2	soltero	no	no
40	3	sí	1	casado	sí	no
53	2	sí	2	divorciado	no	sí
64	2	sí	3	divorciado	no	no
58	2	sí	2	casado	sí	sí
33	1	no	1	soltero	no	no

(> 45 años) y (< 3 niños o (no divorciado)) → Sí

## Modelo 2

Edad	# coches en propiedad	Tiene casa	# niños	Estado civil	Tiene perro	Tiene barco
66	1	sí	2	viudo	no	sí
52	2	sí	3	casado	no	sí
22	0	no	0	casado	sí	sí
25	1	no	1	soltero	no	no
44	0	no	2	divorciado	sí	no
39	1	sí	2	casado	sí	no
26	1	no	2	soltero	no	no
40	3	sí	1	casado	sí	no
53	2	sí	2	divorciado	no	sí
64	2	sí	3	divorciado	no	no
58	2	sí	2	casado	sí	sí
33	1	no	1	soltero	no	no

> 50 años → Sí

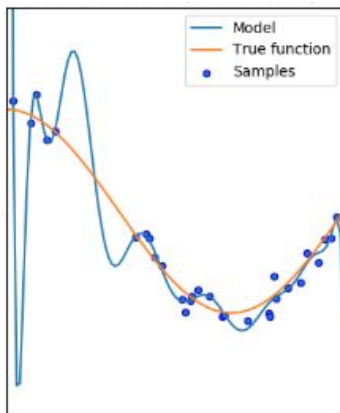
¿Cuál es mejor?

A large, stylized yellow geometric shape, resembling a thick, open 'L' or a corner of a square, is positioned in the bottom-left corner of the slide. It is composed of two thick yellow lines that meet at a right angle, with the lines extending towards the bottom and left edges of the frame.

# Underfitting, Overfitting y generalización

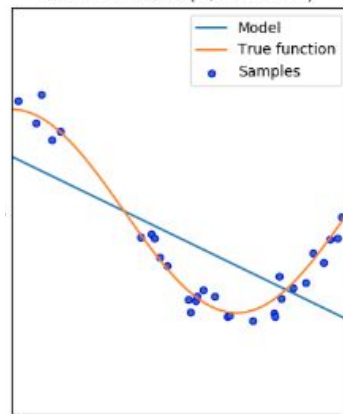
## Sobreajuste (overfitting)

- Alto grado de complejidad
- Funciona bien en training set
- No generaliza bien

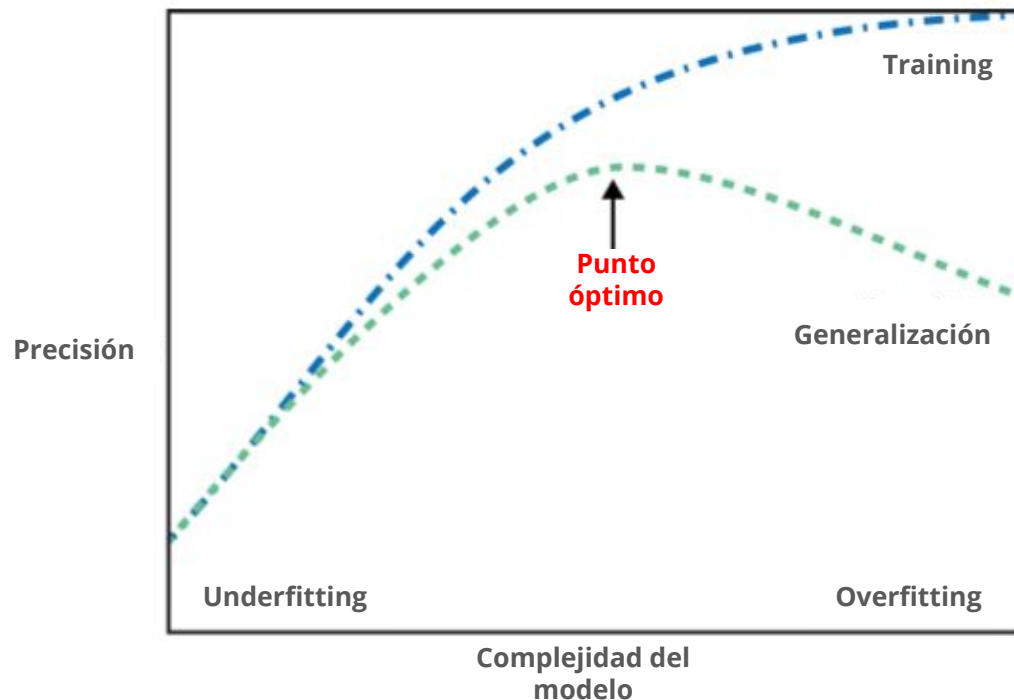


## Subajuste (underfitting)

- Demasiado sencillo
- Predice mal, incluso en training set
- Generaliza demasiado



# Se trata de buscar un compromiso



## Complejidad del modelo y tamaño del dataset

- $\uparrow$  Dataset  $\rightarrow$   $\uparrow$  Complejidad
  - ¿Y si tenemos 100k entradas de posibles compradores de barcos?
- **Recoger más datos** funciona muy bien para los modelos de Aprendizaje Supervisado





# Lab 0 - Datasets de ejemplo

# 1. Conclusiones

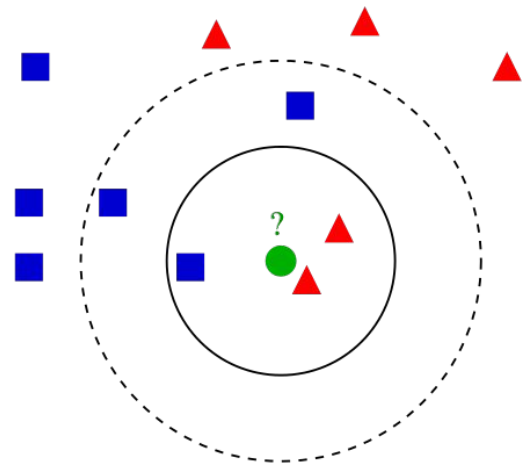
- El uso de **datasets sencillos** es útil
- Hay que probar con **datasets reales conocidos**
- Importante desarrollar una **intuición**
- Realizar siempre **trabajo exploratorio** de los datos



# 1. k-Nearest Neighbors

# k-Nearest Neighbors

- Algoritmo sencillo
  - No paramétrico
  - Basado en instancias
- Clasificación, o regresión
  - Training mínimo: construir modelo → almacenar dataset
- Predicción: encontrar **datapoints más próximos** en training set
  - Por defecto: min(métrica *Minkowski* o distancia euclídea)

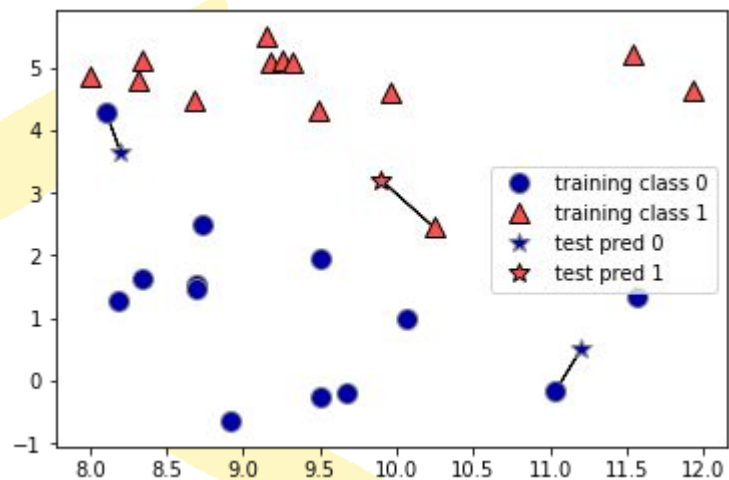


Manhattan 
$$d(p, q) = \sum_{i=1}^n |q_i - p_i|$$

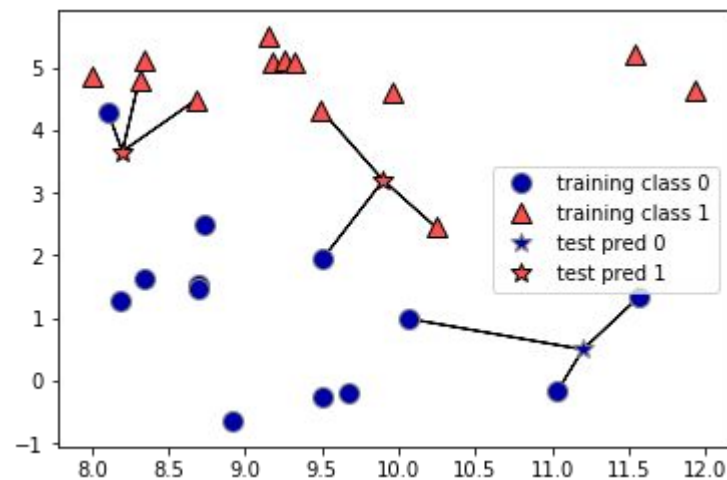
Euclídea 
$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

Minkowski 
$$d(p, q) = \sum_{i=1}^n ((|q_i - p_i|)^q)^{\frac{1}{q}}$$

# k-Nearest Neighbors



Neighbors = 1



Neighbors = 3



# Lab 1 - k-Nearest Neighbors



# 1. Conclusiones

- Dos parámetros importantes
  - Número de vecinos → 3 a 5, pero probar
  - Métrica de distancia → Fuera de alcance
- Fácil de entender, rendimiento razonable
  - Training rápido
  - Testing lento para grandes datasets
- Se usa menos que la **regresión lineal**