K-NN

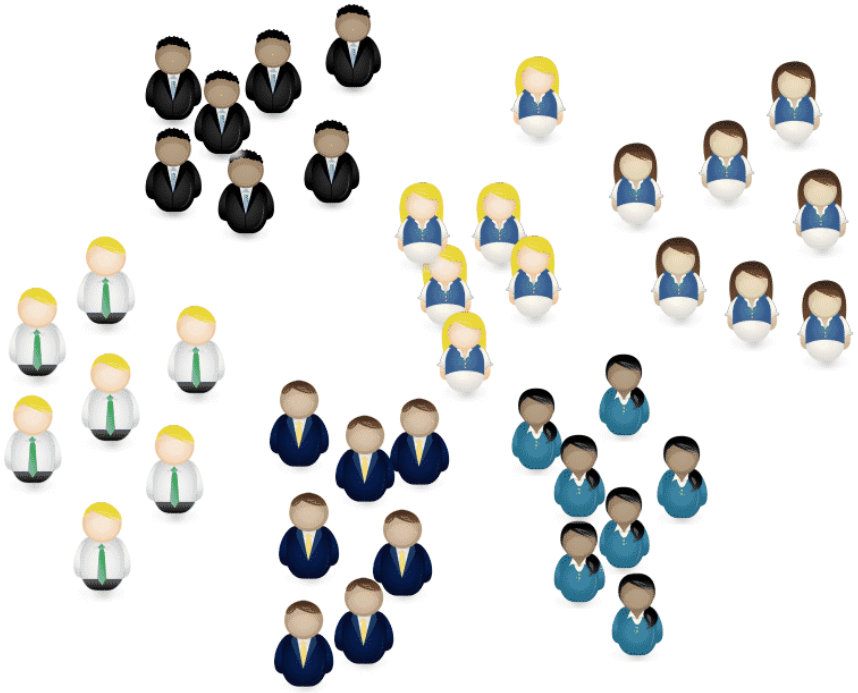
Data Mining

Ester Vidaña Vila

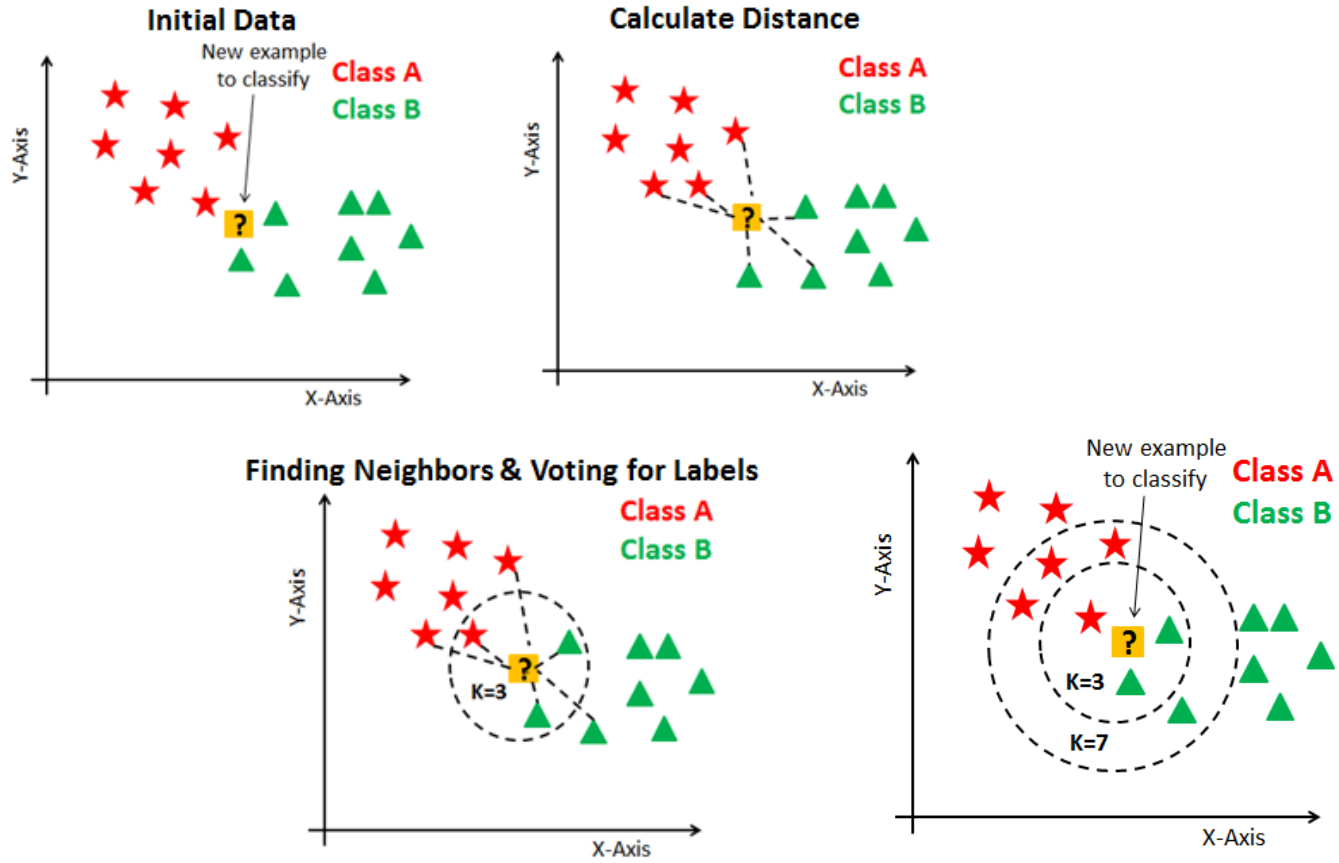
laSalle ENG
Universitat Ramon Llull

K-NN

- La idea es clasificar dependiendo de la distancia entre muestras vecinas.



K-NN

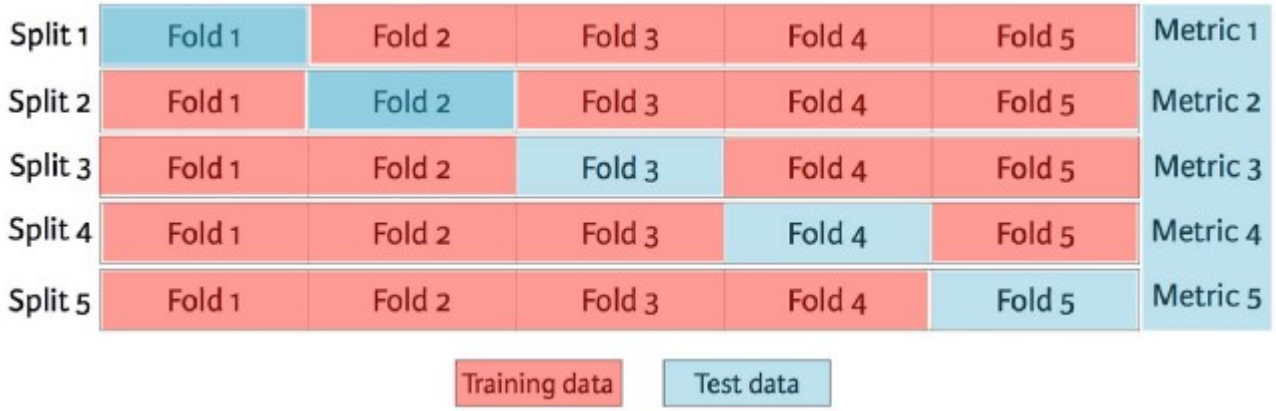

Initial Data
New example
to classify
Class A
Class B
Y-Axis
X-Axis
?
Calculate Distance
Class A
Class B
Y-Axis
X-Axis
?
Finding Neighbors & Voting for Labels
Class A
Class B
Y-Axis
X-Axis
?
K=3
New example
to classify
Class A
Class B
Y-Axis
X-Axis
?
K=3
K=7

¿Cómo se implementa en Python?

- Función: sklearn.neighbors.KNeighborsClassifier
- Documentación:
- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Cross validation

- Cross-validation es la técnica de dividir de forma aleatoria nuestros datos en **K** grupos.
- Un grupo se utiliza como test, y el resto de grupos se utilizan como train.
- El modelo se entrena con los datos de train y se evalúa con los datos de test.
- El proceso se repite hasta que todos los grupos se han utilizado como test.



https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a

Función sklearn: cross_val_score

¿Se puede usar para problemas de regresión?

- ¡Sí! En este caso, asignaríamos (por ejemplo) el promedio de las variables numéricas de los vecinos.
- Ejemplo:

Age	Loan	House Price Index	Distance	
25	$40,000	135	102000	
35	$60,000	256	82000	
45	$80,000	231	62000	
20	$20,000	267	122000	
35	$120,000	139	22000	2
52	$18,000	150	124000	
23	$95,000	127	47000	
40	$62,000	216	80000	
60	$100,000	139	42000	3
48	$220,000	250	78000	
33	$150,000	264	8000	1
48	**$142,000**	**?**		

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Vecino más próximo:

$$D = \sqrt{(48-33)^2 + (142000-150000)^2} =$$

8000.01 -> HPI = 264

HPI = (264+139+139)/3 = 180.7

¿Se puede usar para problemas de regresión?

- ¡Problema! ¿Qué pasa si las unidades entre variables son muy distintas?
- Deberíamos estandarizarlas:

Age	Loan	House Price Index	Distance	
25	$40,000	135	102000	
35	$60,000	256	82000	
45	$80,000	231	62000	
20	$20,000	267	122000	
35	$120,000	139	22000	2
52	$18,000	150	124000	
23	$95,000	127	47000	
40	$62,000	216	80000	
60	$100,000	139	42000	3
48	$220,000	250	78000	
33	$150,000	264	8000	1
48	**$142,000**	**?**		

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Age	Loan	House Price Index	Distance
0.125	0.11	135	0.7652
0.375	0.21	256	0.5200
0.625	0.31	231	0.3160
0	0.01	267	0.9245
0.375	0.50	139	0.3428
0.8	0.00	150	0.6220
0.075	0.38	127	0.6669
0.5	0.22	216	0.4437
1	0.41	139	0.3650
0.7	1.00	250	0.3861
0.325	0.65	264	0.3771
0.7	**0.61**	**?**	

$$X_s = \frac{X - Min}{Max - Min}$$

¿Cómo se implementa en Python?

- Función: sklearn.neighbors.KNeighborsRegressor
- Documentación:
- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor
- ¿Cómo transformamos los datos en Python?
- Función: sklearn.preprocessing.StandardScaler
- ¿Qué hace?
- Hace que cada *feature* tenga media 0 y desviación estándar 1 [z = (x - u) / s].
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html