Universidad de Aysén

# KNN & K-means

**Oscar Painen Briones**

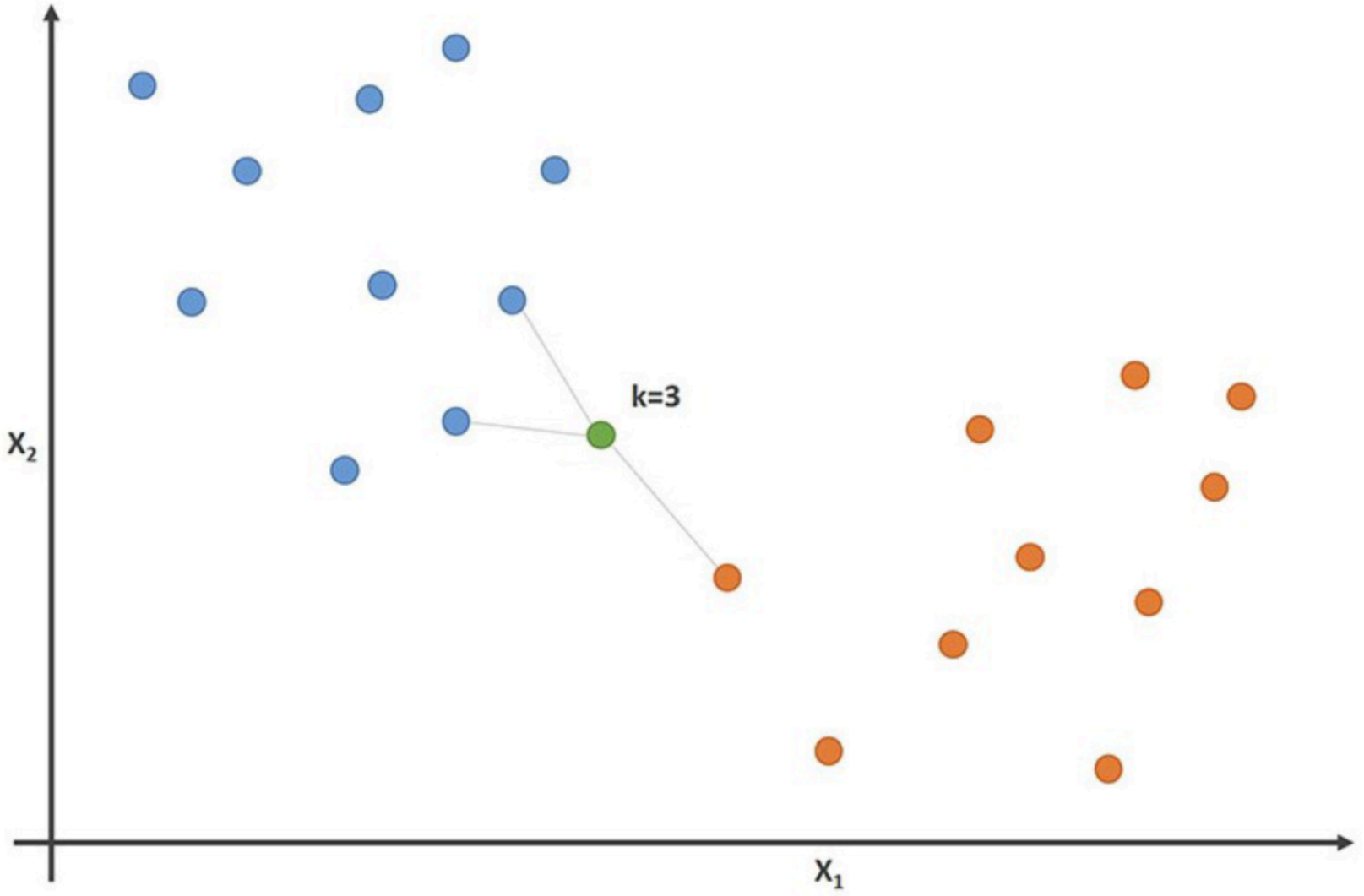
# Vecinos cercanos

Los métodos de vecinos cercanos son la base de muchos otros métodos de aprendizaje.

El principio de los métodos de vecinos cercanos consiste en encontrar un número predefinido de muestras de entrenamiento que están más próximas en distancia a un nuevo punto. El número de muestras puede ser una constante definida por el usuario (k-vecinos más cercanos) o variar según la densidad local de puntos (aprendizaje basado en un radio determinado).

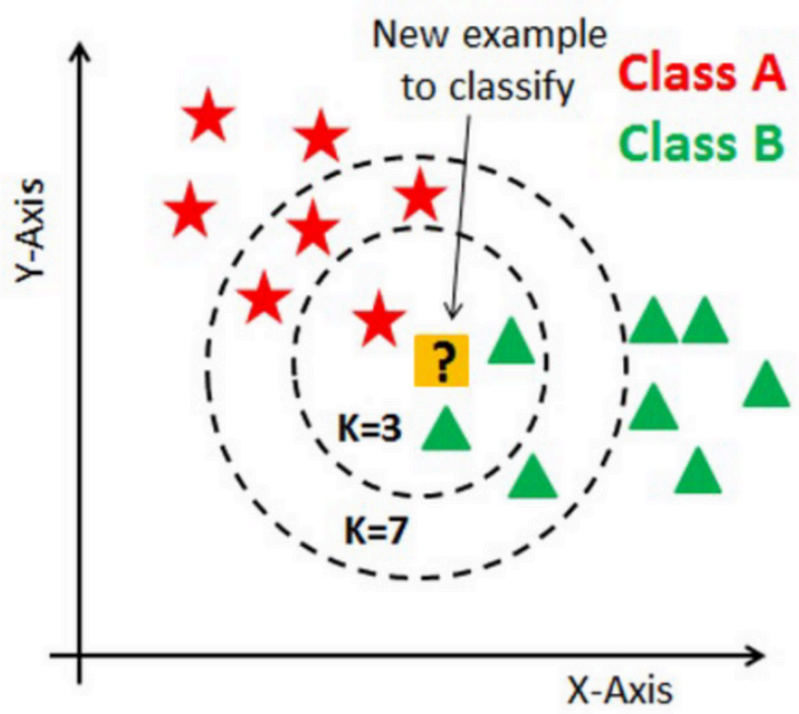
Aunque la distancia puede ser medida con cualquier métrica, la distancia euclidiana es la más utilizada.

A pesar de su simplicidad, los vecinos más cercanos han demostrado ser eficaces en una amplia variedad de problemas, incluidos la **búsqueda**, la **clasificación** y la **detección de anomalías**.


k=3
X2
X1

# Vecinos cercanos para clasificación

- Dado un nuevo ejemplo, buscamos sus k vecinos más cercanos y lo asignamos a la clase mayoritaria.



- Dado un nuevo ejemplo, buscamos sus k vecinos más cercanos.

- Su distancia a los vecinos es usada para estimar su densidad.

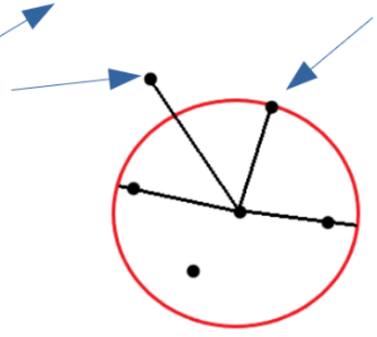
- Se compara su densidad con la densidad de los vecinos.

Definimos k-distance(B): distancia de B a su kNN.

Luego definimos alcanzabilidad:

$$\text{reachability-distance}_k(A,B) = \max\{k\text{-distance}(B), d(A,B)\}$$

Parametriza la distancia según el tamaño del vecindario de B



Si A está afuera del vecindario de B, nos quedamos con ésta

Si A está dentro del vecindario, se reemplaza por la distancia al k vecino de B.

# Vecinos cercanos para detección de outliers

Definimos la densidad de alcanzabilidad local:

La sumatoria se minimiza si el área es densa

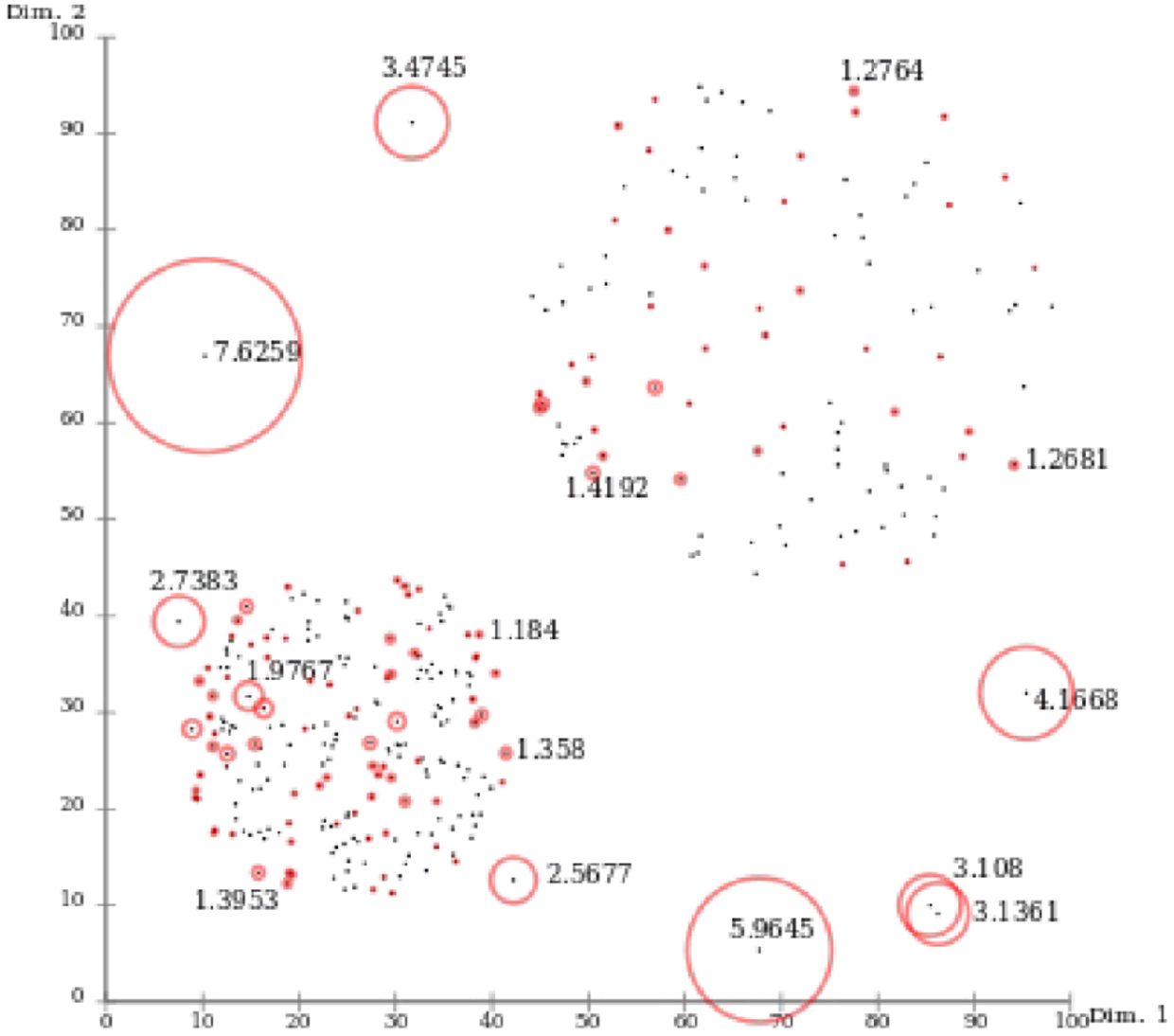
$$LRD(p) = 1/\left(\frac{\sum_{q \epsilon knn(p)} reach-dist(p.q)}{||k-neighborhood||}\right)$$

… pero como es el inverso, LRD aumenta si el área es densa.

Y luego el Local Outlier Factor (LOF):

$$LOF(p) = \left(\frac{\sum_{q\ \epsilon\ knn(p)} \frac{LRD(q)}{LRD(p)}}{||k-neighborhood||}\right)$$

Si p es un outlier, su LRD va a ser menor que el de sus vecinos (q), por lo que LOF(p) aumenta.


Dim. 2
100
90
80
70
60
50
40
30
20
10
0
3.4745
1.2764
7.6259
1.2681
1.4192
2.7383
1.184
1.9767
4.1668
1.358
2.5677
3.108
1.3953
3.1361
5.9645
0
10
20
30
40
50
60
70
80
90
100
Dim. 1

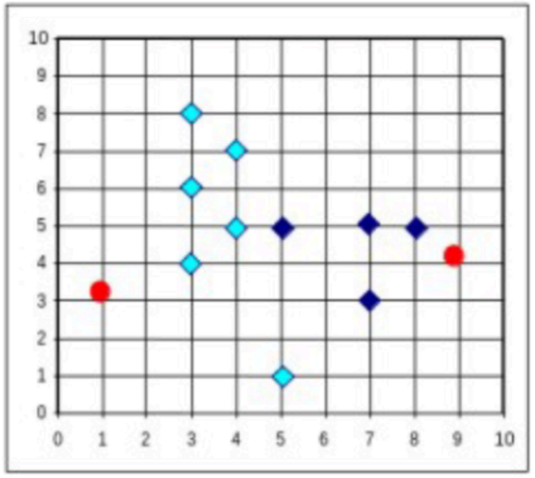
# K-means

## Clustering con k-means

- Cada cluster en $K$-means es definido por un centroide.
- Objetivo: optimizar alguna noción de distancia:
  1. Intra-cluster: (Minimizar) distancia entre objetos de un cluster a su centroide.
  2. Inter-cluster: (Maximizar) distancia entre objetos de clusters distintos
- Centroide:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x$$

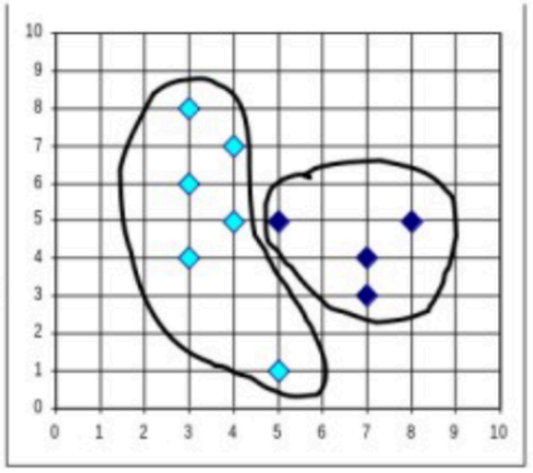
  donde $C_i$ denota un cluster.
- Idea del algoritmo:
  - Asignación inicial: `k` centroides al azar.
  - Reasignación: asignar cada objeto a su centroide más cercano (algoritmo avaro).
  - Recomputación: recalcular los centroides.



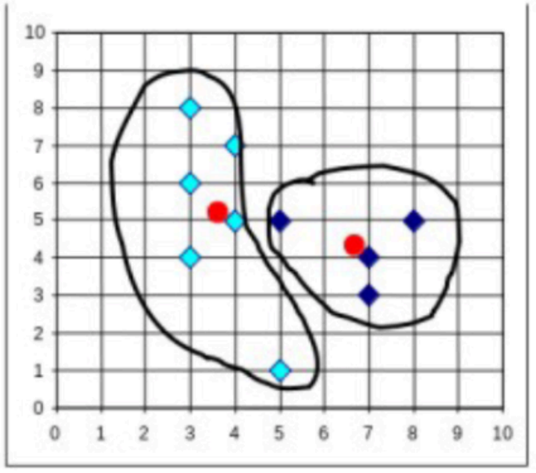
K=2

K centroides al azar

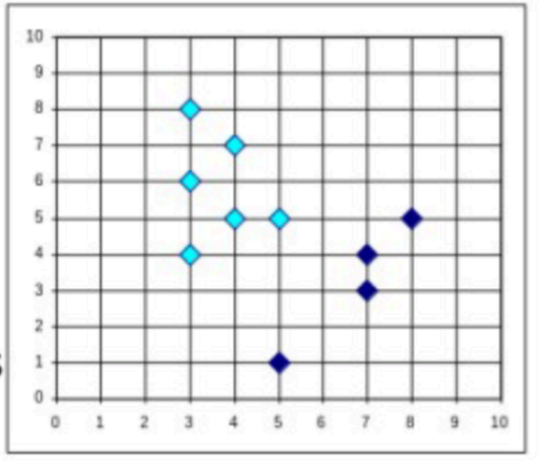
Se asigna cada objeto a su centroide más cercano



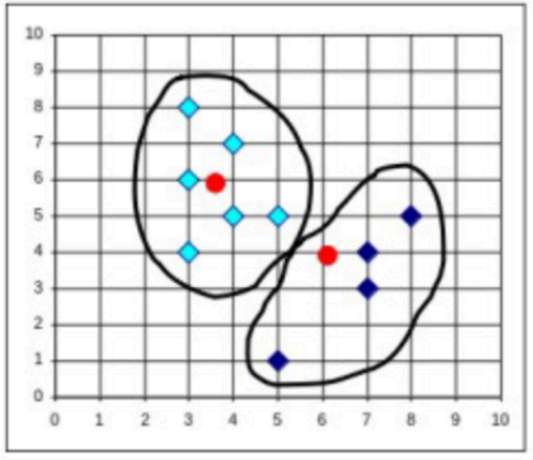
Se recalculan los centroides



reasignación



Se recalculan los centroides

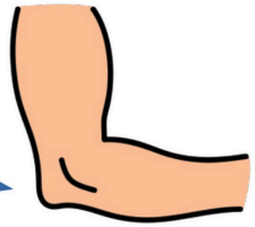


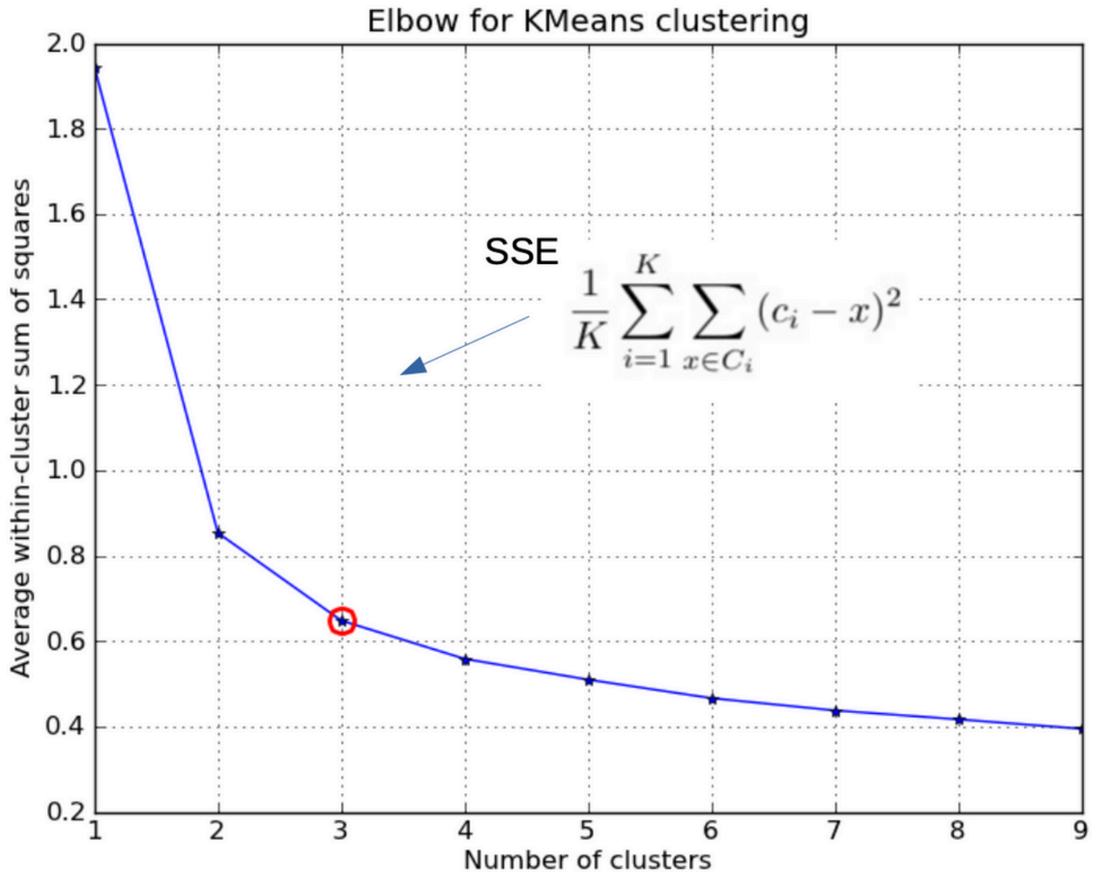
reasignación

# Algoritmo para entender como agrupamos los datos. ¿Que K usamos?

ELBOW (codo):

Variar k buscando el codo





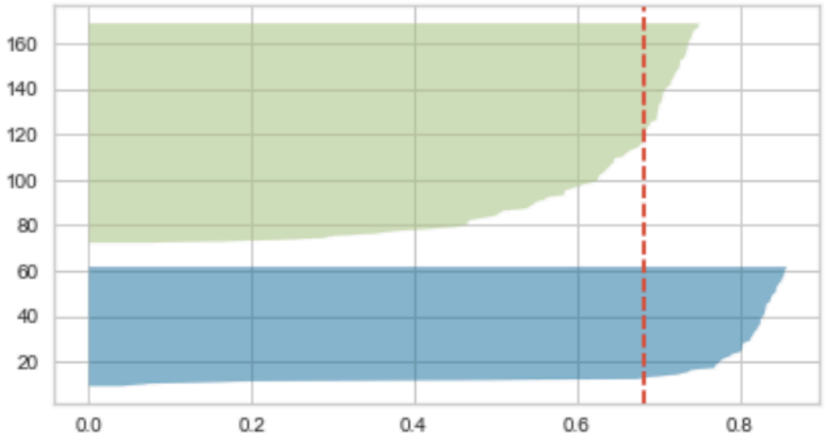

SSE

$$\frac{1}{K}\sum_{i=1}^{K}\sum_{x\in C_i}(c_i - x)^2$$

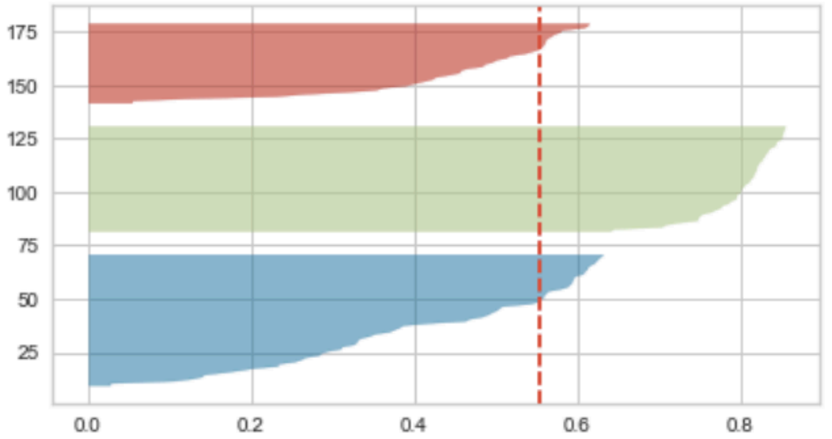
Esto es mala idea ya que SSE es monótono decreciente con k

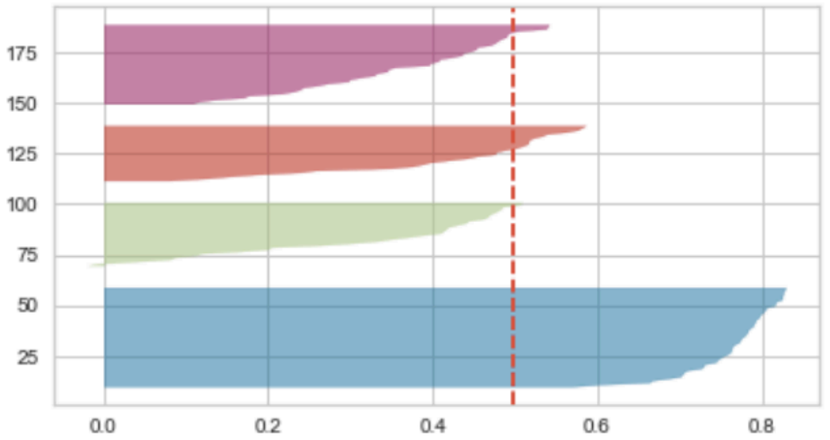
Silhouette promedio:

k=2

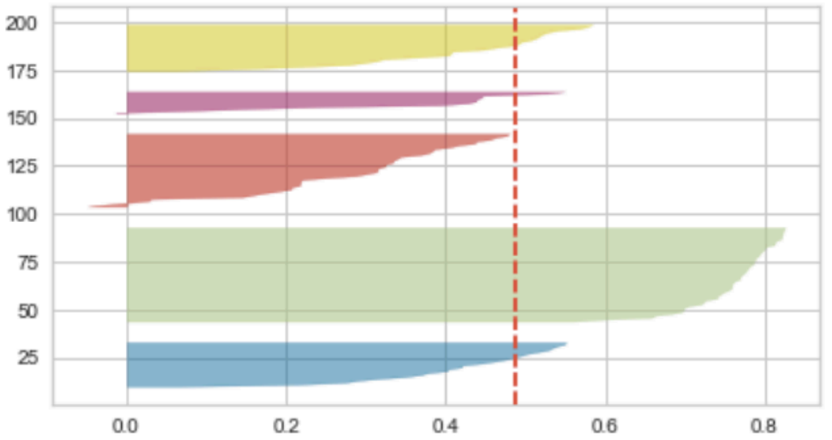



k=3



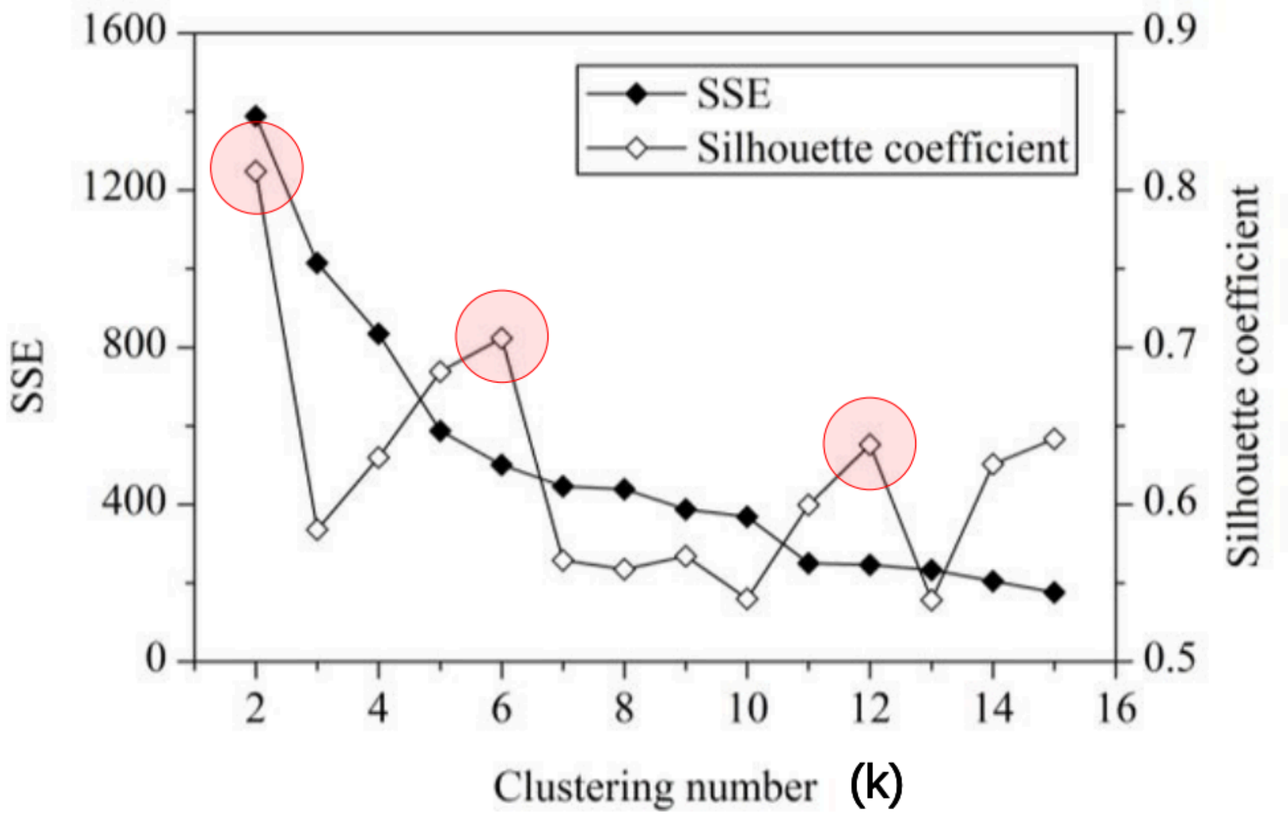




k=4




k=5

Silhouette v/s ELBOW:




¿Con cuál **k** se quedan?