

Módulo 6 – Aprendizaje de Máquina No Supervisado

K-Means

Especialización en Ciencia de Datos

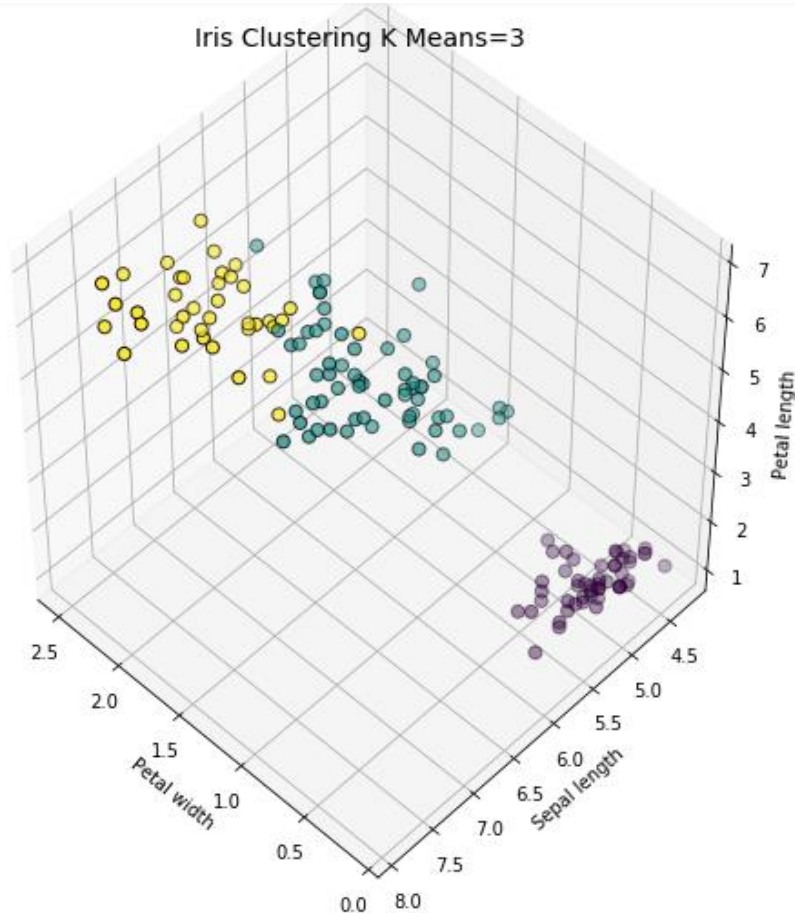
Objetivos



- Utilizar los conceptos básicos de aprendizaje de máquinas no supervisado.
- Conocer los distintos tipos de algoritmos para agrupamiento jerárquico.
- Conocer sobre Dendrogramas.
- Implementación en Python.
- Ventajas y desventajas de la clusterización jerárquica.

K-Means

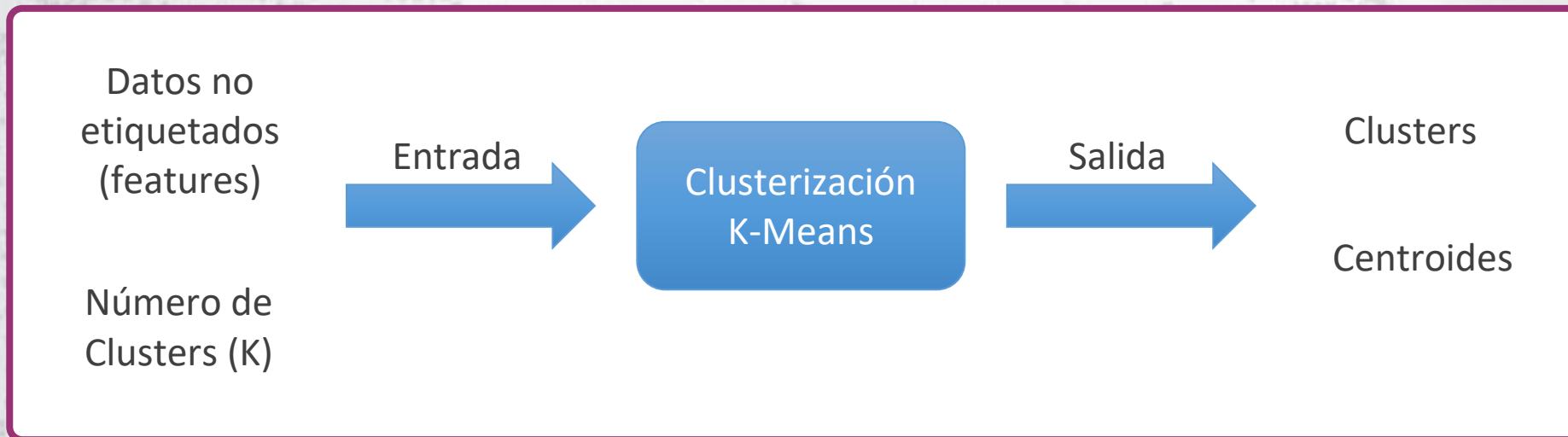
¿Qué es K-Means?



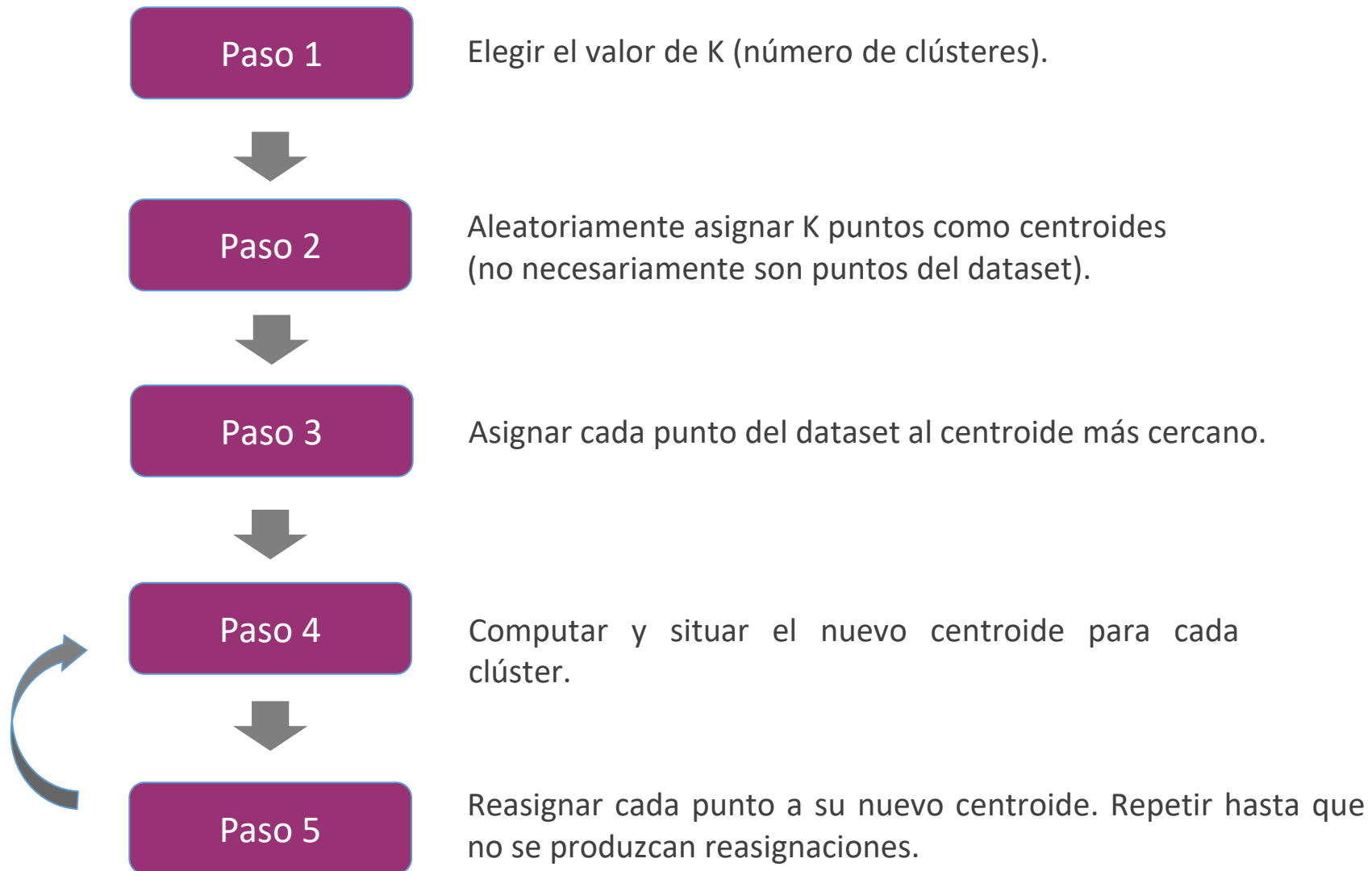
- K-Means es uno de los algoritmos de clusterización más populares utilizados en análisis de datos y aprendizaje automático debido a su simplicidad y eficiencia.
- Este algoritmo agrupa un conjunto de datos en **k grupos** o clústeres basados en la similitud entre los datos, minimizando un criterio llamado inercia (también llamado **suma de cuadrados intracluster**). El "k" representa el número de clústeres deseados y **requiere ser especificado a priori**.
- K-Means **escala bien** en grandes cantidades de datos.

K-Means Clustering

K-Means recibe dos entradas para realizar el proceso de clusterización, datos no etiquetados (features) y la cantidad de clusters buscada (k). La salida del algoritmo, entrega la clusterización de cada instancia y la ubicación del centroide de cada cluster.



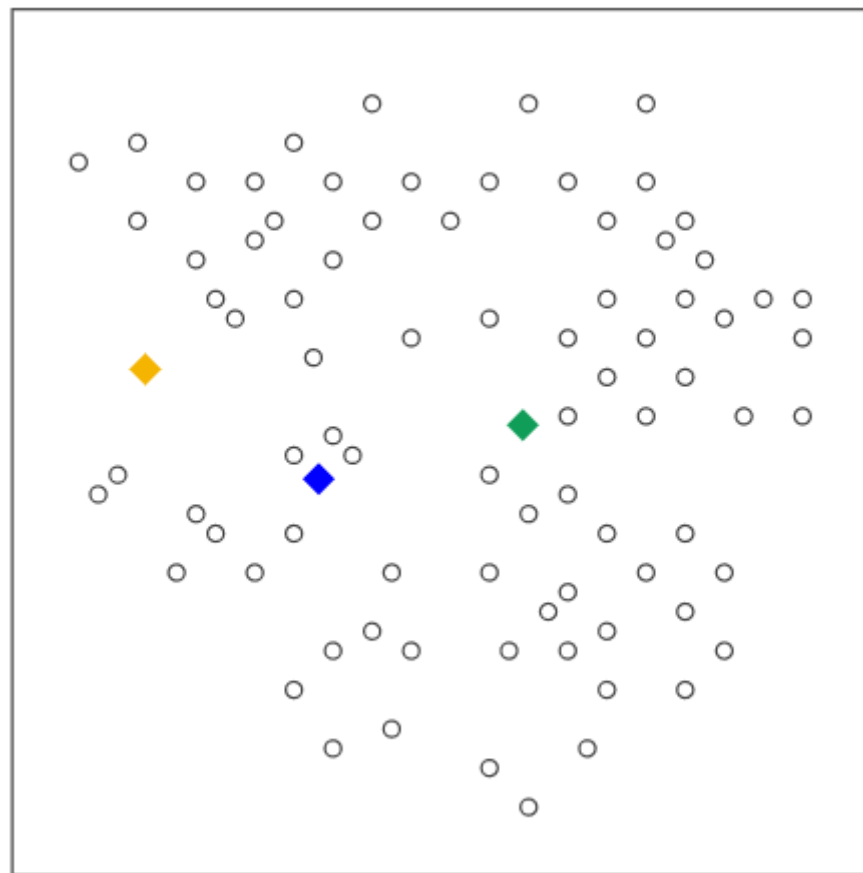
Algoritmo K-Means



Algoritmo K-Means

➤ Paso uno

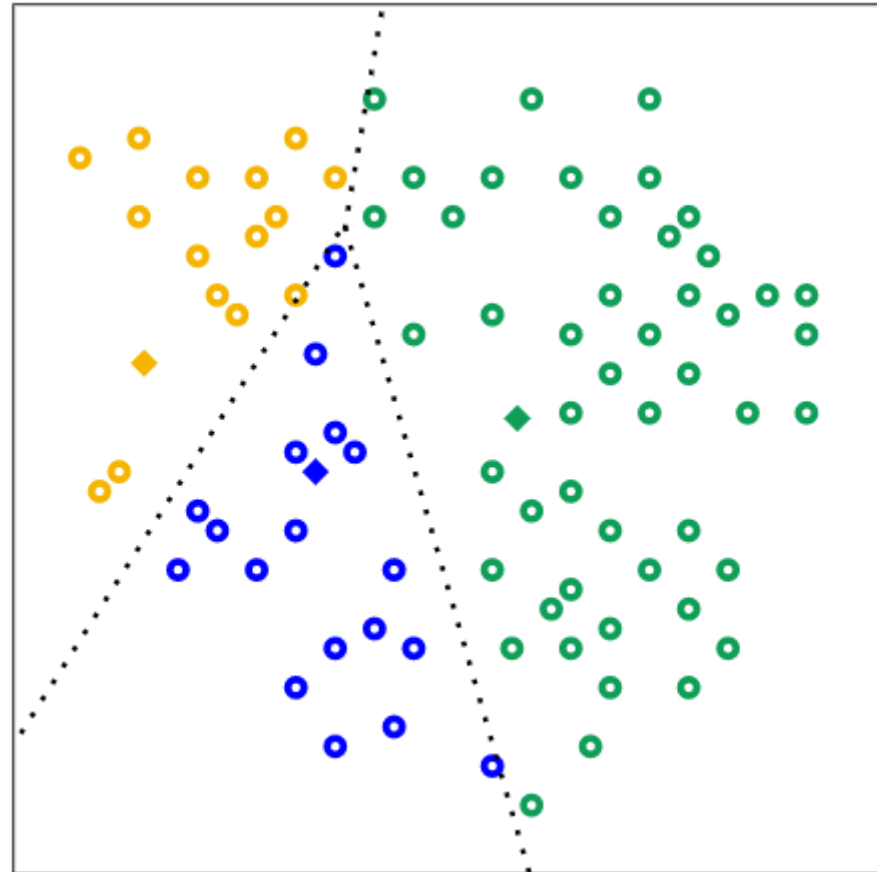
El algoritmo elige al azar un centroide para cada clúster. En nuestro ejemplo, elegimos uno de 3 y, por lo tanto, el algoritmo elige al azar 3 centroides.



Algoritmo K-Means

➤ Paso dos

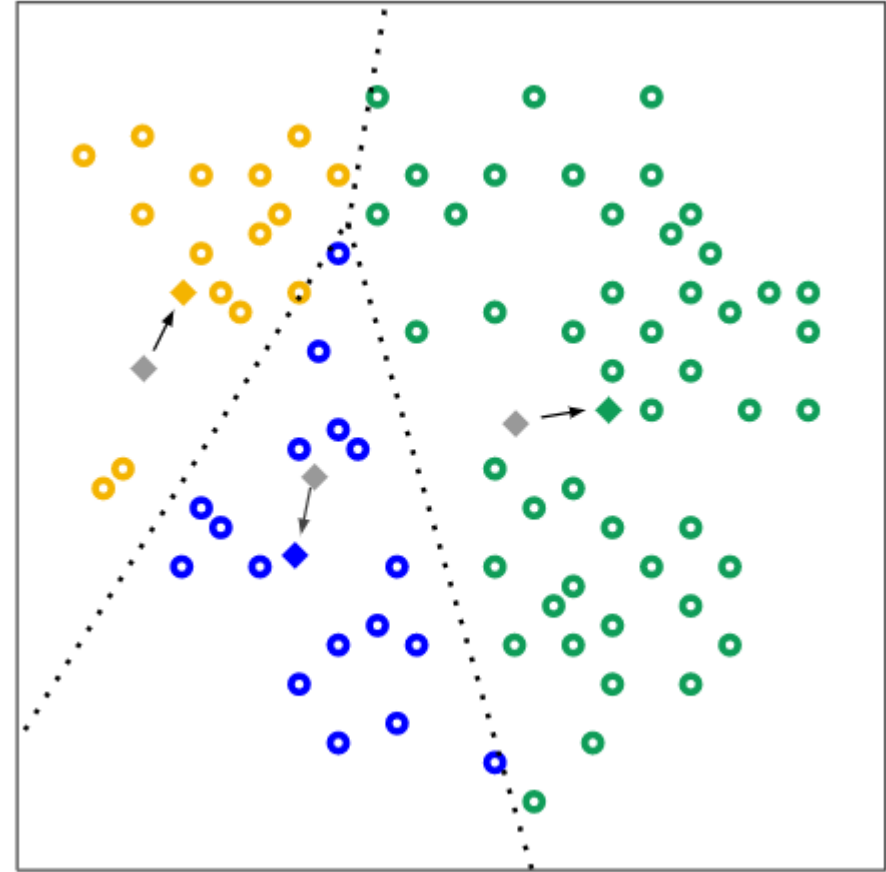
El algoritmo asigna cada punto al centroide más cercano para obtener k clústeres iniciales.



Algoritmo K-Means

➤ Paso tres

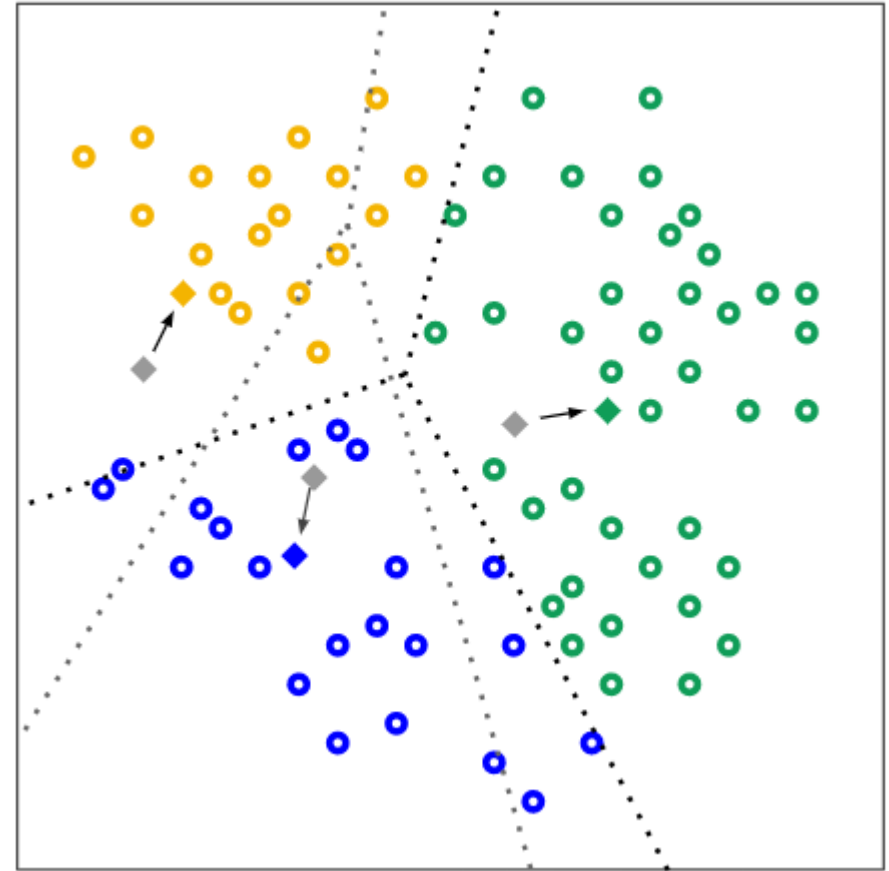
Para cada clúster, el algoritmo recalcula el centroide mediante el promedio de todos los puntos del clúster. Los cambios en los centroides se muestran en la figura 3 con flechas. Como los centroides cambian, el algoritmo vuelve a asignar los puntos al centroide más cercano. En la próxima figura, se muestran los clústeres nuevos después de la reasignación.



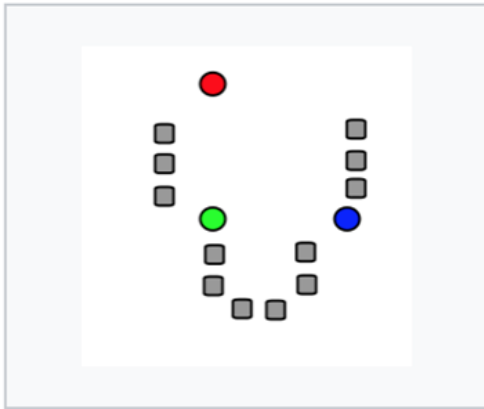
Algoritmo K-Means

➤ Paso cuatro

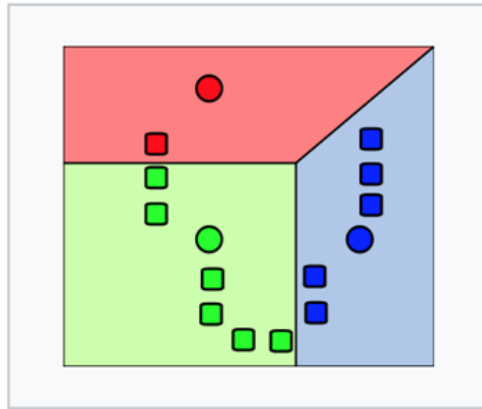
El algoritmo repite el cálculo de los centroides y la asignación de puntos hasta que estos dejen de cambiar de clúster. Cuando agrupas en conjuntos de datos grandes, debes detener el algoritmo antes de alcanzar la convergencia mediante otros criterios.



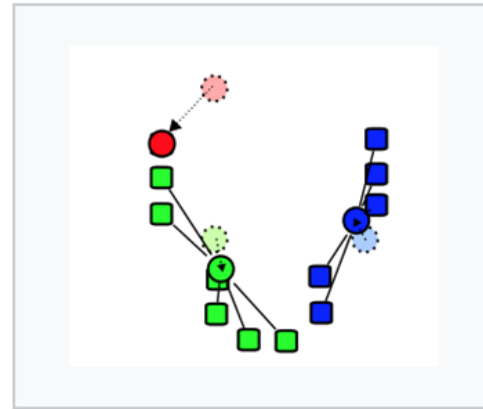
K-Means



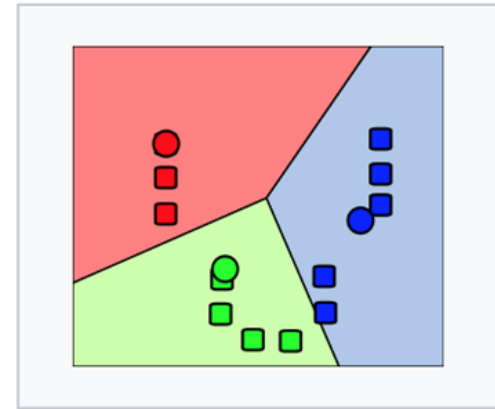
1. Escoger k “medias” iniciales* (en este caso $k=3$) aleatoriamente generadas dentro del dominio de los datos.



2. k clusters son creados asociando cada observación a la media más cercana. Las particiones representan el diagram de Voronoi generado por las medias.



3. El centroide de cada cluster, se convierte en la nueva media.



4. Iteraciones: pasos 2 y 3 se repiten hasta que la convergencia se ha alcanzado.

Concepto de Inercia

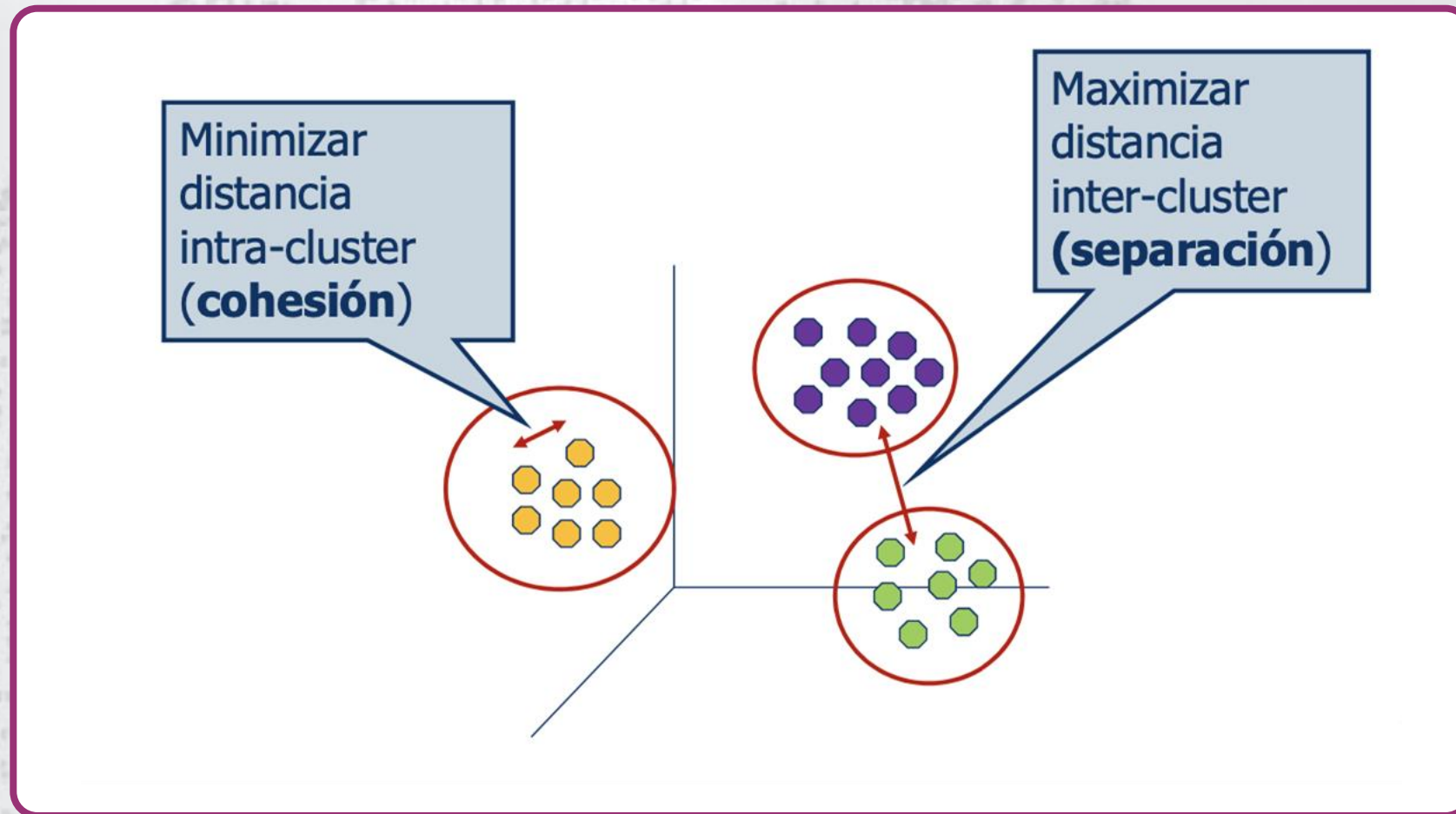
Dado un conjunto de observaciones (x_1, x_2, \dots, x_n), donde cada observación es un vector real de d dimensiones, k-means construye una partición de las observaciones en k conjuntos ($k \leq n$) a fin de minimizar la suma de los cuadrados dentro de cada grupo (WCSS): $S = \{S_1, S_2, \dots, S_k\}$

WCSS: Within Cluster Sum of Squares

$$\arg \min_S \sum_{i=1}^k \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

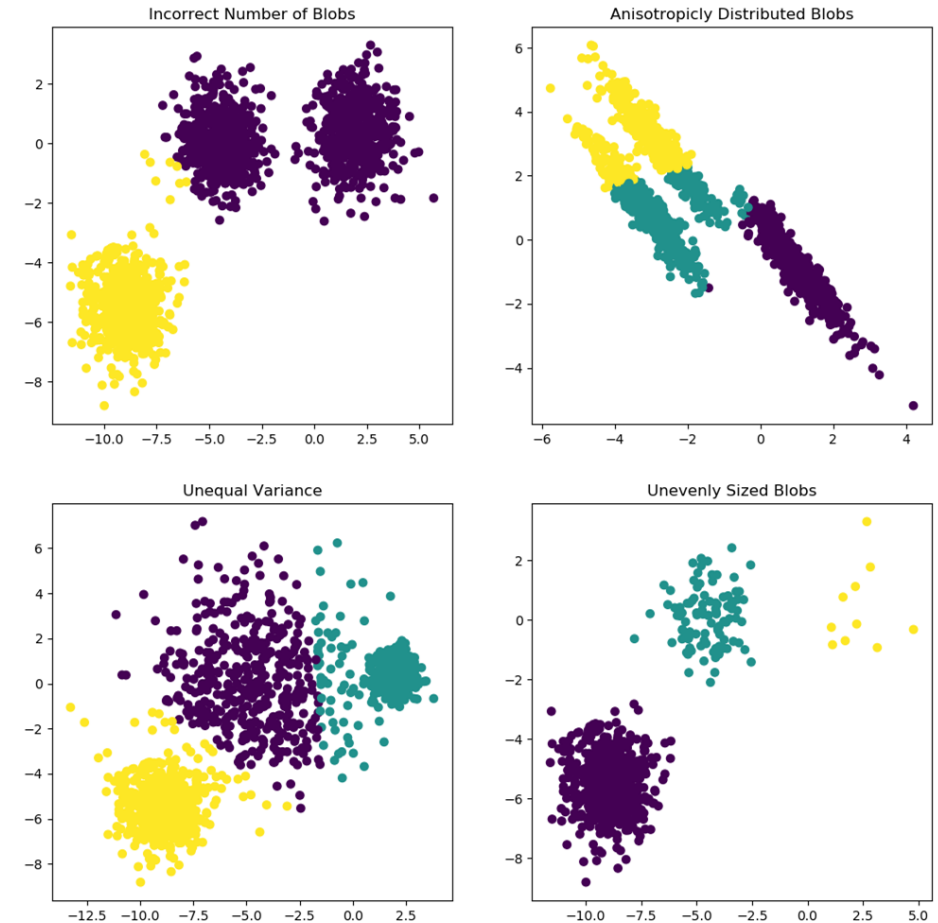
donde μ_i es la media de puntos en S_i .

Evaluación del Resultado



Consideraciones

- La inercia es un criterio para establecer clústeres, pero tiene algunos inconvenientes:
- Asume que los clústeres son convexos e isotrópicos (distribución igual en todas direcciones). No funciona bien con figuras irregulares
- La alta dimensionalidad tiende a inflar las distancias euclidianas (maldición de la dimensionalidad). En este caso, se sugiere ejecutar previamente un algoritmo de reducción de dimensionalidad.



Consideraciones

- K-Means trabaja midiendo distancias entre los puntos hacia los centroides. Esto implica que:
 - Puede utilizar cualquier medida de distancia, siendo la Euclidiana la más simple.
 - Debería considerar escalamiento de los datos dado que los resultados pudieran variar.
- Variables categóricas deben ser transformadas ya que K-Means computa distancias. Esto implica que:
 - Si son variables categóricas ordinales, asignar un número que represente el orden (p. ej. 5 / 10).
 - Si son variables categóricas cardinales, deben crearse variables dummy.

Inicialización Aleatoria

La inicialización aleatoria de centroides en K-Means puede ser problemática porque puede conducir a soluciones subóptimas o no convergentes. Es decir, dependiendo de la selección aleatoria de los centroides iniciales, el algoritmo puede converger a un mínimo local en lugar del mínimo global de la función de costo, lo que resulta en una solución subóptima.



Existen diferentes técnicas para abordar el problema de la inicialización aleatoria en K-Means, como la inicialización con **k-means++** que elige los centroides iniciales de manera más inteligente, o el uso de múltiples inicializaciones aleatorias para aumentar la probabilidad de encontrar la solución óptima.

Ventajas

1. Es eficiente y escalable en términos de tiempo y recursos computacionales. K-means es adecuado para conjuntos de datos grandes y es más rápido que otros algoritmos de clusterización.
2. Es fácil de implementar y entender. El algoritmo tiene pocos parámetros y es fácil de entender cómo funciona.
3. Es útil para identificar patrones en los datos y agrupar objetos similares.
4. Es altamente interpretable, ya que los resultados se presentan en forma de grupos con características similares.

Desventajas

1. K-means requiere que el número de clústeres "k" se especifique de antemano. Esto puede ser un problema si no se conoce el número óptimo de clústeres o si la elección es subjetiva.
2. Es sensible a la inicialización aleatoria de los centroides iniciales, lo que puede conducir a soluciones subóptimas o no convergentes.
3. Es sensible a valores atípicos y ruido en los datos. Estos puntos pueden influir en la posición de los centroides y, por lo tanto, en la estructura final de los clústeres.
4. No funciona bien en conjuntos de datos con formas o tamaños irregulares, y puede haber dificultades en la identificación de clústeres con formas complejas.

Elección del Valor de K

Elección del valor de K

Determinar el número óptimo de clústeres "k" en el algoritmo K-Means puede ser un desafío y depende en gran medida del conjunto de datos y de los objetivos del análisis. A continuación, se describen algunos métodos que se pueden utilizar para determinar el valor de "k":

1. **Método del codo** (elbow method): este método implica trazar el valor de la función de costo en función del número de clústeres y elegir el valor de "k" donde la disminución de la función de costo disminuye significativamente.
1. **Método de silueta** (silhouette method): este método evalúa la calidad de los clústeres a través de la silueta de cada punto de datos, que es una medida de cuán similar es un punto a su clúster en comparación con los clústeres vecinos. El valor de "k" que maximiza el valor medio de la silueta de los puntos de datos se considera como el número óptimo de clústeres.

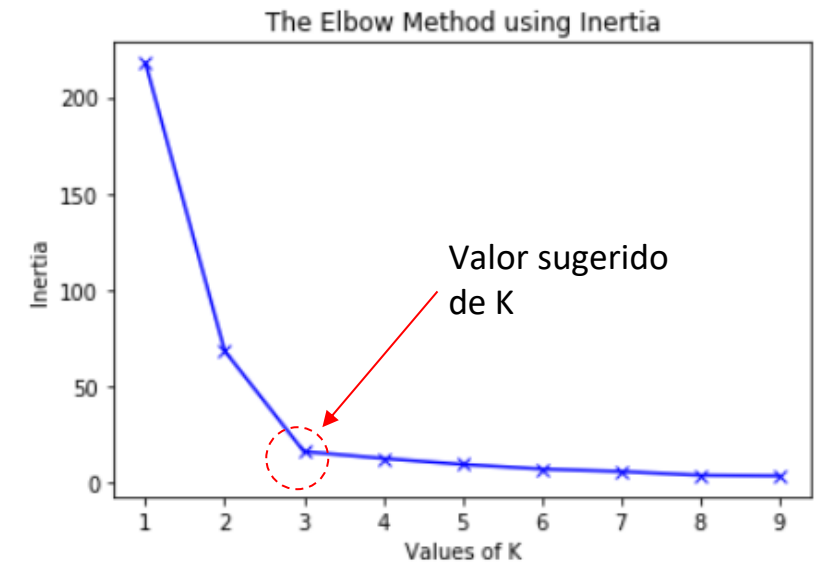
Elección del valor de K

- 3. **Métodos basados en criterios externos:** en algunos casos, se puede utilizar la información a priori sobre los datos para determinar el número óptimo de clústeres. Por ejemplo, si se están agrupando imágenes de caras, se puede elegir el número de personas en las imágenes como el número de clústeres. Este enfoque requiere conocimiento previo del dominio y puede no ser aplicable en todos los casos.
- 3. **Métodos basados en criterios internos:** evalúan la calidad de los clústeres en función de la estructura de los datos y no requieren conocimiento previo del dominio. Por ejemplo, el índice de Dunn y el índice de Calinski-Harabasz.

En general, es importante tener en cuenta que la elección del número óptimo de clústeres es un proceso iterativo que puede requerir la exploración de varios métodos y la consideración de los objetivos específicos del análisis.

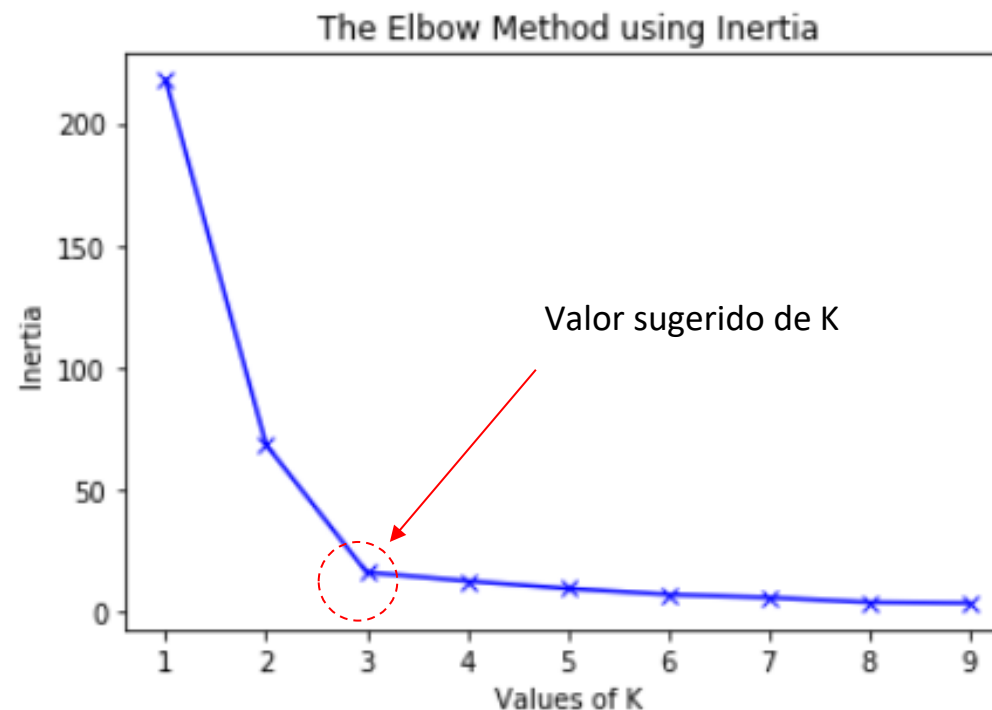
Método del Codo

- El método se basa en la idea de que a medida que aumenta el número de clústeres, la varianza dentro de cada clúster disminuye, lo que resulta en una disminución en la función de costo.
- Sin embargo, a medida que se agregan más clústeres, la mejora en la varianza dentro del clúster se vuelve menos significativa, lo que resulta en una disminución menos pronunciada en la función de costo.
- Para utilizar el método del codo, se traza la suma de los cuadrados de la distancia de cada punto de datos al centroide de su clúster correspondiente en función del número de clústeres. Luego, se busca el punto en el gráfico donde se produce una disminución significativa en la función de costo, que se parece a un "codo" en el gráfico. Este punto se considera como el número óptimo de clústeres.



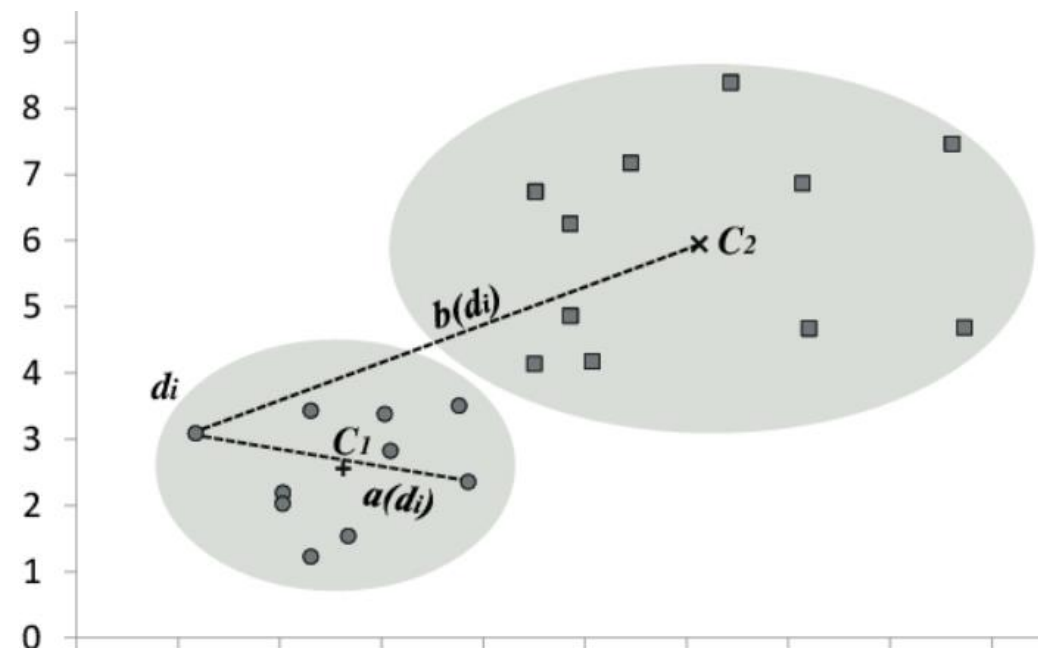
Método del Codo

Es importante tener en cuenta que el **método del codo** puede no siempre ser efectivo en la determinación del número óptimo de clústeres, especialmente en conjuntos de datos complejos o con formas irregulares. En tales casos, pueden ser necesarios otros métodos para determinar el número óptimo de clústeres, como el método de silueta o métodos basados en criterios internos y externos.



Análisis del Coeficiente de Silueta

- Este método evalúa la calidad de los clústeres utilizando la silueta de cada punto de datos, que es una medida de cuán similar es un punto a su clúster en comparación con los clústeres vecinos.
- Para cada punto de datos, se calcula su coeficiente de silueta, que es una medida de la similitud del punto con su propio clúster en comparación con otros clústeres. Los coeficientes de silueta varían entre -1 y 1, donde un valor alto indica que el punto está bien ajustado a su propio clúster y mal ajustado a otros clústeres, y un valor bajo indica lo contrario.



Análisis del Coeficiente de Silueta

El coeficiente de la silueta mide “cuán bien pertenece un punto a un clúster”, esto es, comparando las distancias a su centroide y al del centroide del cluster más próximo. El coeficiente es calculado como promedio de todos los puntos.

Coeficiente de Silueta:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

En donde,

$a(i)$ es la distancia promedio entre i y todos los otros puntos del cluster al cual pertenece.

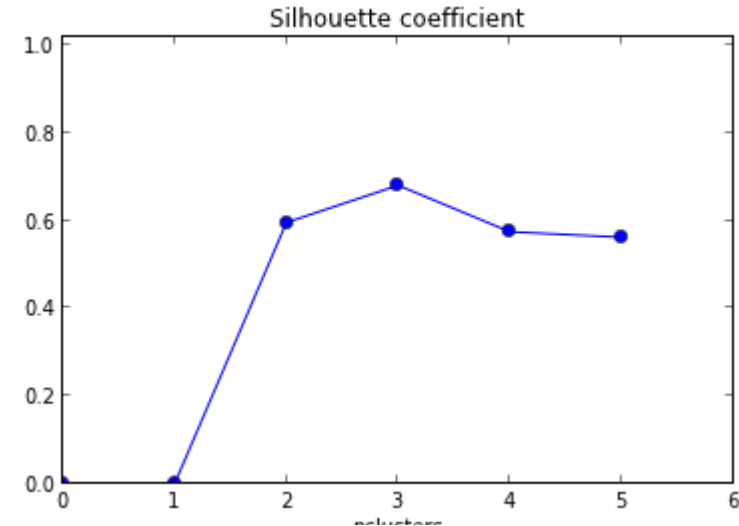
$b(i)$ es la distancia entre i y el centroide del siguiente cluster más cercano.

- Mientras mayor sea la diferencia, el coeficiente se acerca a 1.
- Mientras menor sea la diferencia, el coeficiente se acerca a 0.
- Si el coeficiente es negativo, significa que el punto fue mal clasificado (valor atípico)

Análisis del Coeficiente de Silueta

Luego, se calcula el valor medio de los coeficientes de silueta de todos los puntos de datos en cada clúster. El valor medio de la silueta del conjunto de datos se define como la media de los valores medios de los coeficientes de silueta de todos los clústeres.

El número óptimo de clústeres se elige como el que maximiza el valor medio de la silueta del conjunto de datos. En general, cuanto mayor sea el valor medio de la silueta, mejor será la calidad de los clústeres.

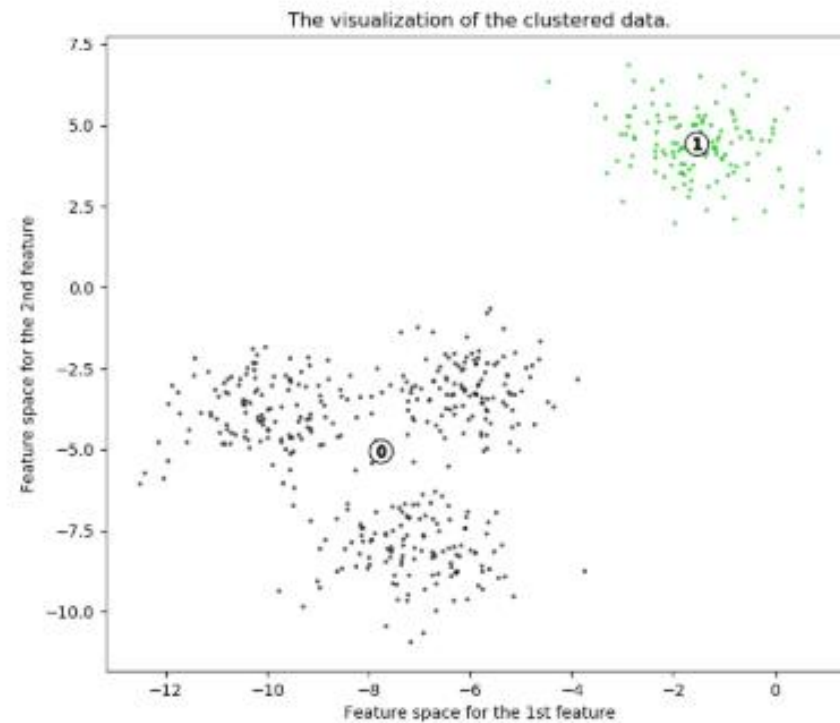
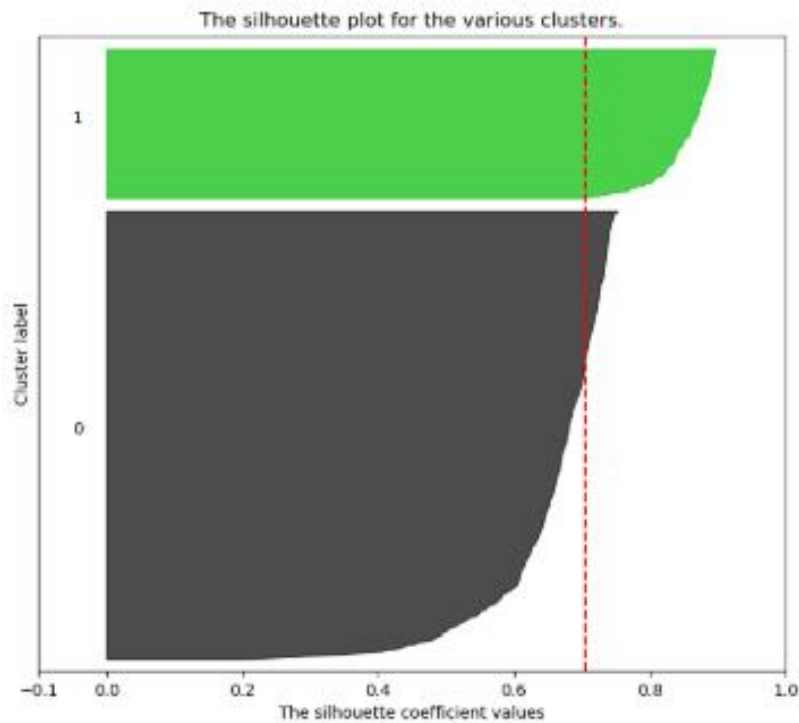


```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.5745566973301872
For n_clusters = 6 The average silhouette_score is : 0.4387644975296138
```

(Silhouette coefficient: coeficiente de silueta)

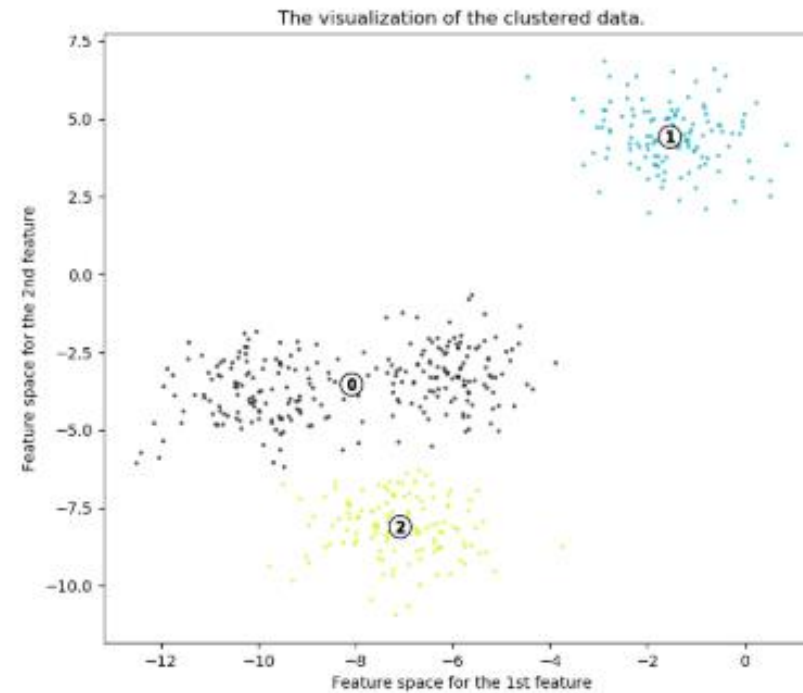
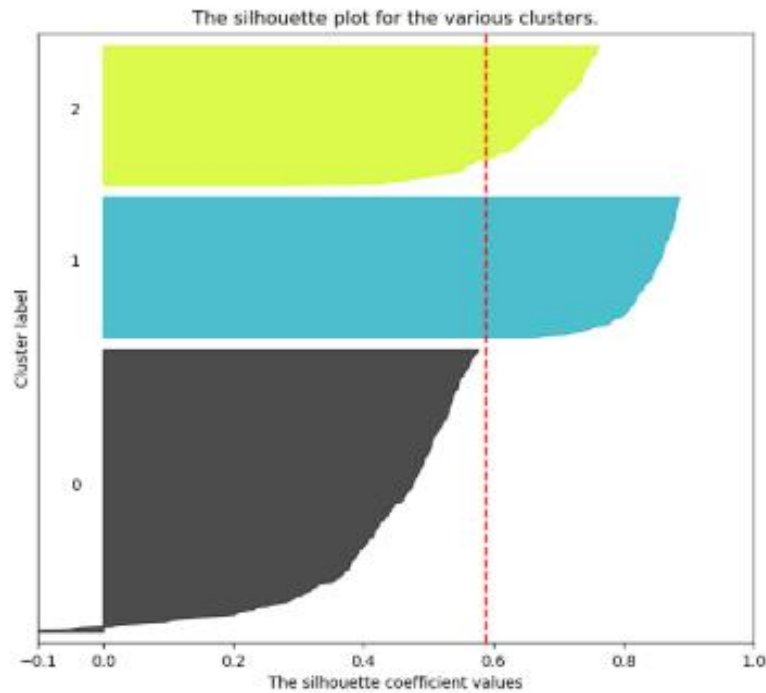
Análisis del Coeficiente de Silueta

Análisis de Silueta para k-Means clustering en datos de muestra con $n_clusters = 2$



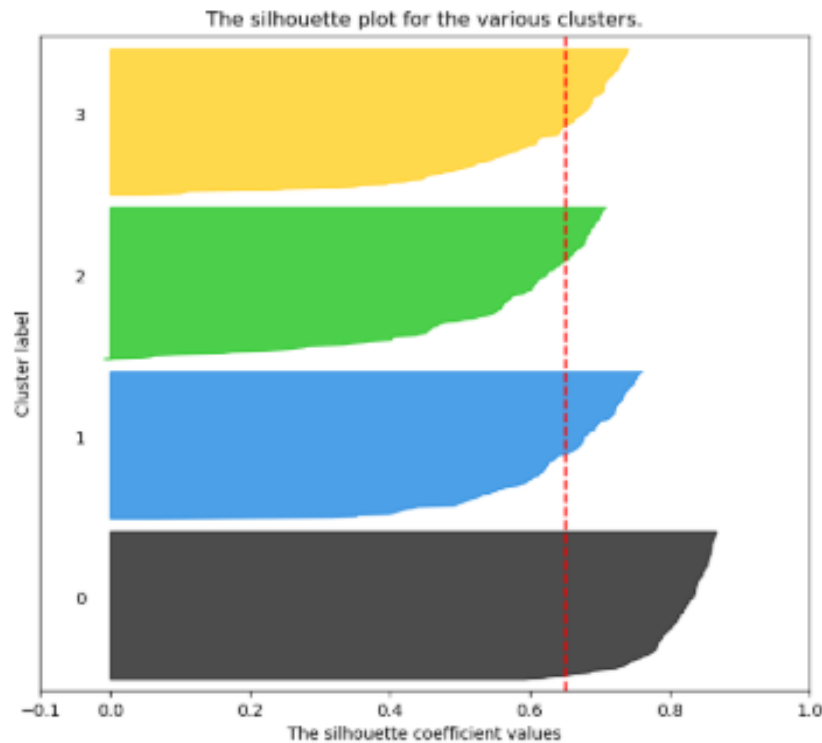
Análisis del Coeficiente de Silueta

Análisis de Silueta para k-Means clustering en datos de muestra con $n_clusters = 3$



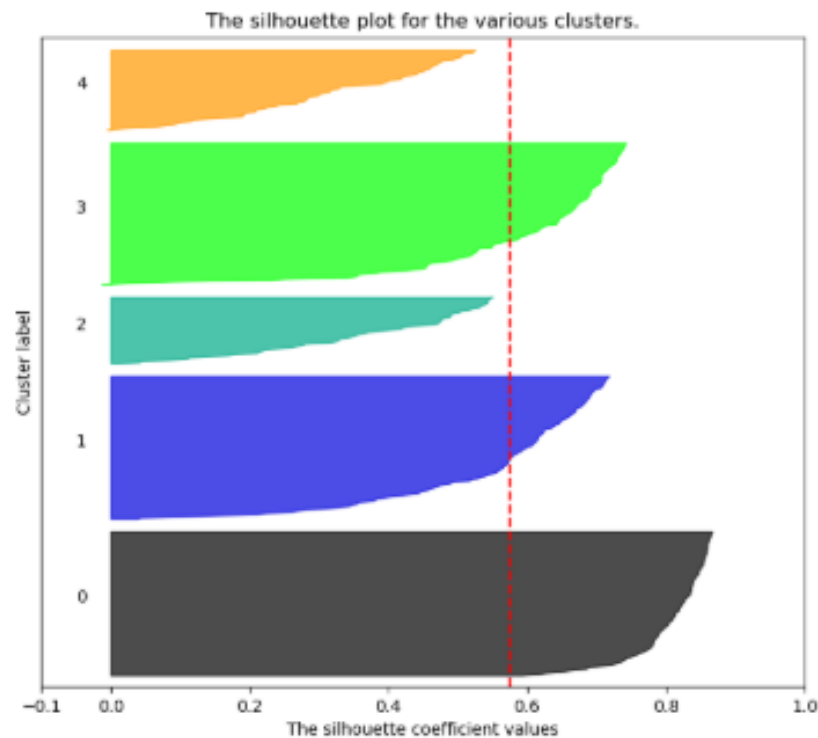
Análisis del Coeficiente de Silueta

Análisis de Silueta para k-Means clustering en datos de muestra con $n_clusters = 4$



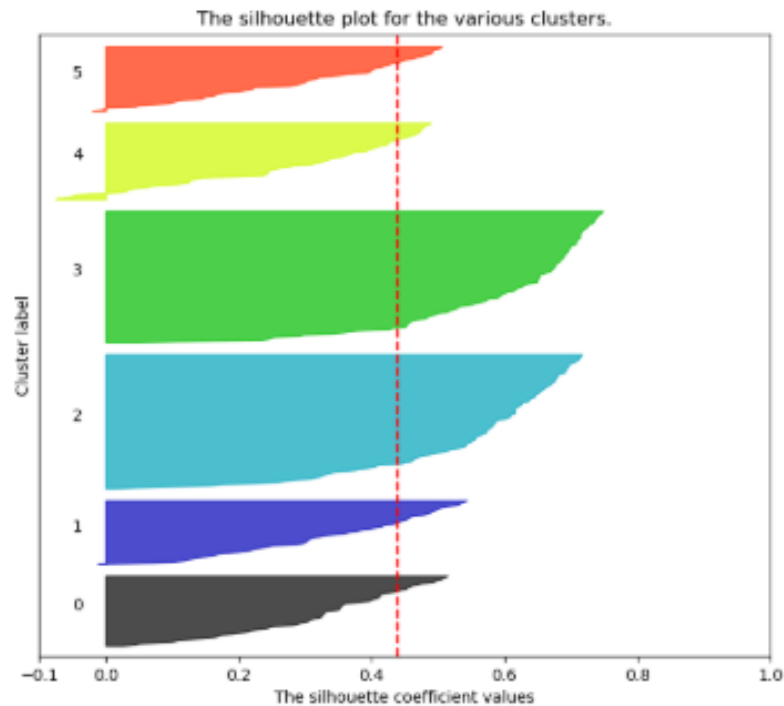
Análisis del Coeficiente de Silueta

Análisis de Silueta para k-Means clustering en datos de muestra con $n_clusters = 5$



Análisis del Coeficiente de Silueta

Análisis de Silueta para k-Means clustering en datos de muestra con $n_clusters = 6$




Análisis del Coeficiente de Silueta

El método de la silueta tiene la ventaja de ser una técnica más robusta y confiable que el método del codo para determinar el número óptimo de clústeres en conjuntos de datos complejos o con formas irregulares. Sin embargo, también tiene algunas limitaciones y puede no ser efectivo en todos los casos. En general, es importante considerar múltiples métodos y criterios al determinar el número óptimo de clústeres en un análisis de clusterización.

Implementación en Python

Librería Sk-Learn

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More ▾](#)

scikit-learn

Machine Learning in Python

[Getting Started](#) [Release Highlights for 0.23](#) [GitHub](#)

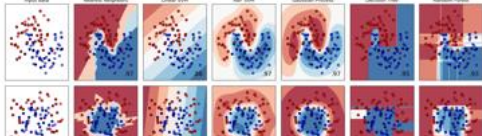
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...

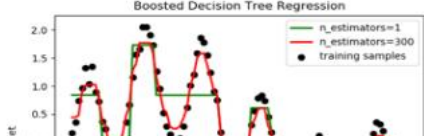


Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...




Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



<https://scikit-learn.org>

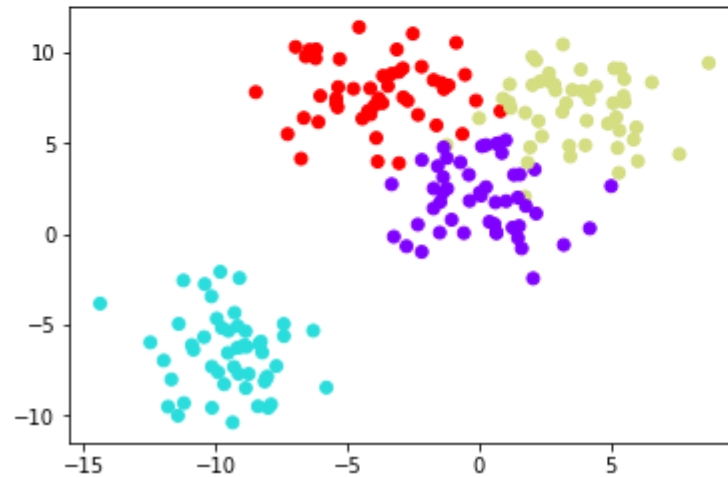
K-Means en Python

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.datasets import make_blobs
```

```
data = make_blobs(n_samples=200, n_features=2, centers=4, cluster_std=1.8, random_state=101)
```

```
p = plt.scatter(data[0][:,0], data[0][:,1], c=data[1], cmap='rainbow')
```



K-Means en Python

```
from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=2)
```

```
kmeans.fit(data[0])
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,  
       n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',  
       random_state=None, tol=0.0001, verbose=0)
```

```
kmeans.cluster_centers_
```

```
array([[ -9.46941837,  -6.56081545],  
       [-0.0336134 ,   5.54542558]])
```

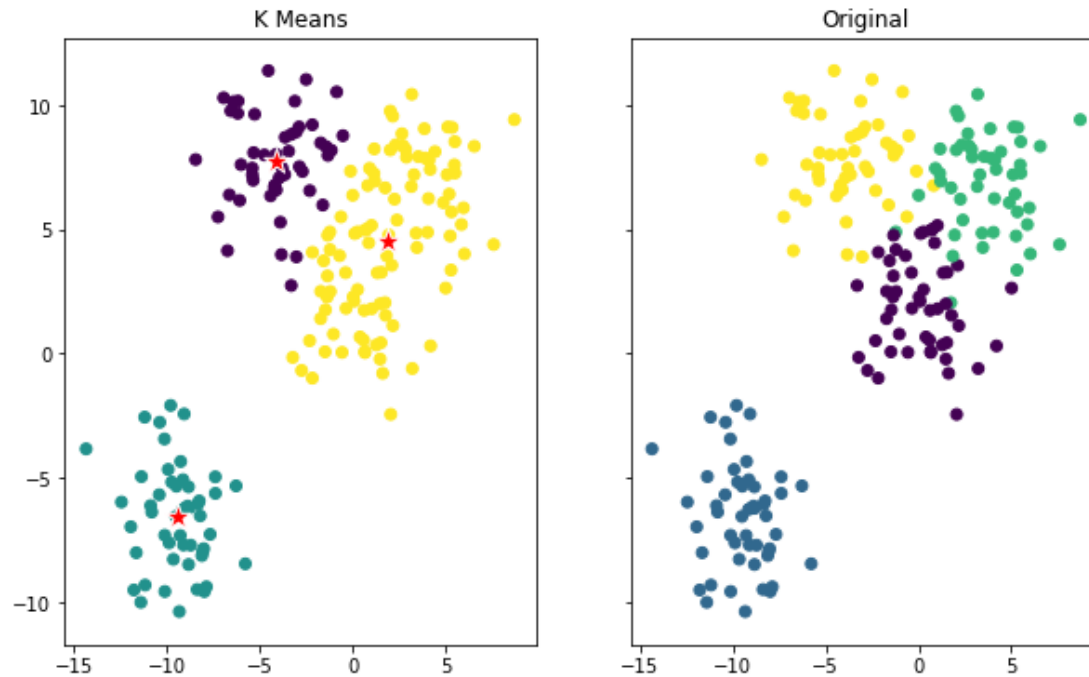
```
kmeans.labels_
```

```
array([1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,  
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1,  
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,  
       0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,  
       0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,  
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,  
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,  
       0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1])
```

K-Means en Python

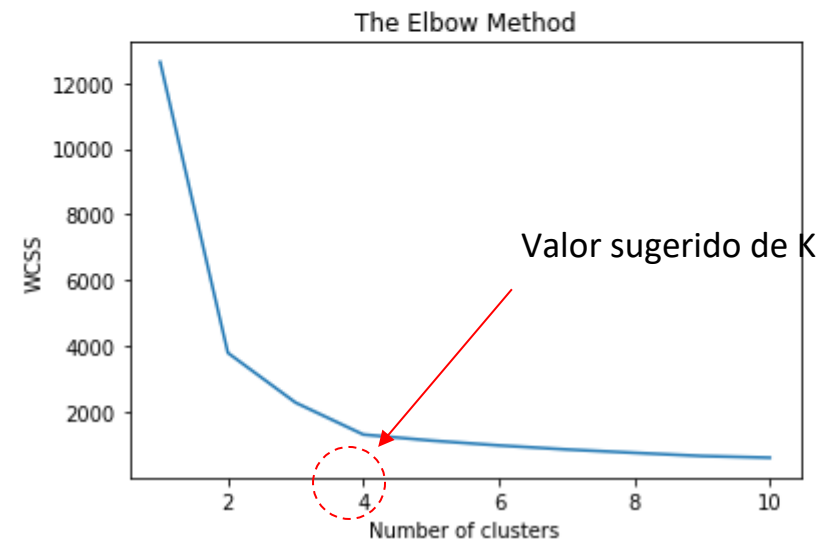
```
fig, (axes1, axes2) = plt.subplots(1, 2, sharey=True, figsize=(10, 6))
axes1.set_title('K Means')
axes1.scatter(data[0][:, 0], data[0][:, 1], c=kmeans.labels_)
axes1.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='r', marker='*', s=200, edgecolors='w')
axes2.set_title('Original')
axes2.scatter(data[0][:, 0], data[0][:, 1], c=data[1])
```

<matplotlib.collections.PathCollection at 0x21bebf86358>



Método del Codo

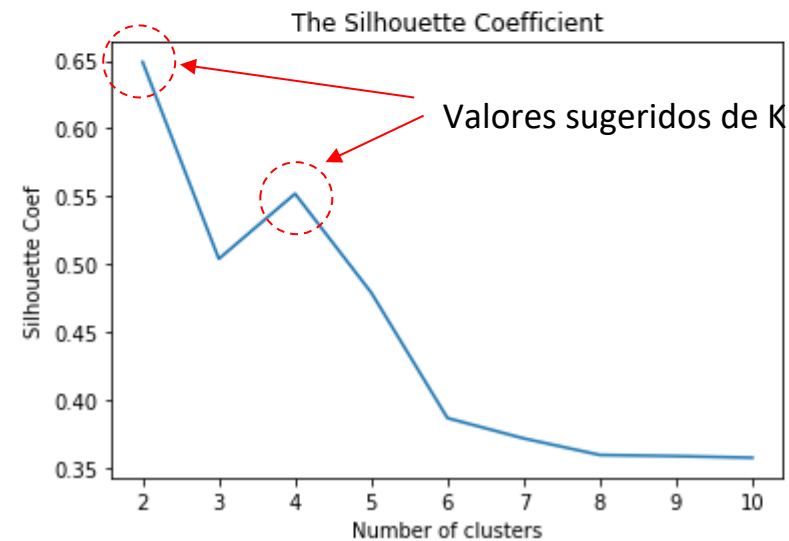
```
: wcss = []  
for i in range(1, 11):  
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)  
    kmeans.fit(X)  
    wcss.append(kmeans.inertia_)  
plt.plot(range(1, 11), wcss)  
plt.title('The Elbow Method')  
plt.xlabel('Number of clusters')  
plt.ylabel('WCSS')  
plt.show()
```



Coeficiente de Silueta

```
from sklearn.metrics import silhouette_samples, silhouette_score
```

```
savg = []  
for i in range(2, 11):  
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)  
    kmeans.fit(X)  
    silhouette_avg = silhouette_score(X, kmeans.labels_)  
    savg.append(silhouette_avg)  
plt.plot(range(2, 11), savg)  
plt.title('The Silhouette Coefficient')  
plt.xlabel('Number of clusters')  
plt.ylabel('Silhouette Coef')  
plt.show()
```



Dudas y consultas

Fin Presentación