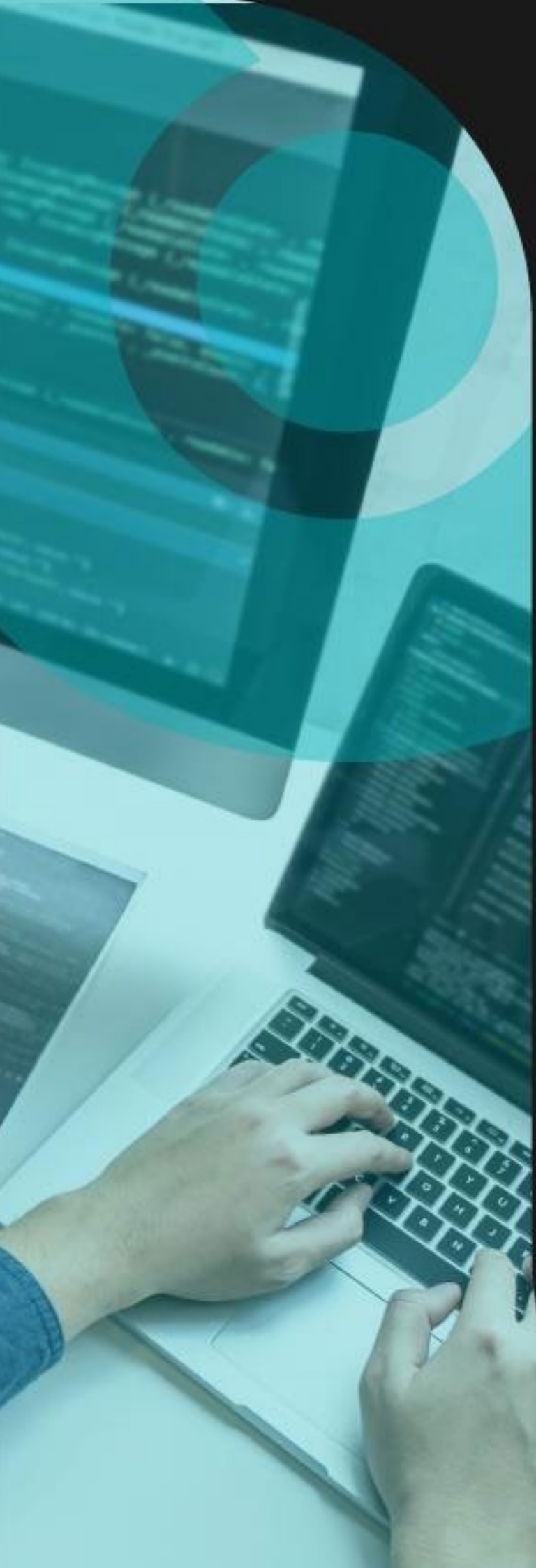


AWAKELAB

BASECAMP

Ciencia de Datos

Módulo: Aprendizaje de Máquina No Supervisado

Aprendizaje Esperado

---

2. Elaborar un modelo predictivo aplicando el algoritmo K-Means utilizando lenguaje Python para resolver un problema de clusterización.

Algoritmos No Supervisados

Los problemas de Aprendizaje No Supervisados se pueden dividir en:

- **Problemas de Agrupación:** Se trata de establecer clústeres o grupos, de forma tal, que los individuos dentro de cada clúster sean similares y que los individuos en distintos clústeres sean heterogéneos. Nos centraremos en este tipo de problemas en esta clase.

- **Problemas de Asociación:** Se trata de descubrir reglas con un nivel de generalización mínimo (soporte), estas reglas nacen de eventos que ocurren repetitivamente y por ende, se asume cierta asociación.

¿En qué consiste la clusterización?

Independientemente del tipo de investigación que esté realizando o de la tarea que realicen sus algoritmos de aprendizaje automático (ML), en algún momento, utilizará técnicas de agrupación en clústeres con bastante liberalidad. La agrupación en clústeres y la preparación de datos van de la mano, ya que muchas veces trabajará, al menos inicialmente, con conjuntos de datos que en gran medida no están estructurados ni clasificados.

Más importante aún, la agrupación en clústeres es una manera fácil de realizar muchos análisis de nivel de superficie que pueden brindarle

ganancias rápidas en una variedad de campos. Los especialistas en marketing pueden realizar un análisis de conglomerados para segmentar rápidamente la demografía de los clientes, por ejemplo. Las aseguradoras pueden profundizar rápidamente en los factores y ubicaciones de riesgo y generar un perfil de riesgo inicial para los solicitantes.

Aun así, sería una pena dejar su análisis en la agrupación de datos, ya que no pretende ser una respuesta única a sus preguntas. De hecho, si bien la agrupación en clústeres en el aprendizaje automático es increíblemente útil en una variedad de entornos, no deja de tener algunas limitaciones bastante importantes.

Principales algoritmos

Existen distintos algoritmos para obtener los grupos (o también llamados *clústeres*) deseados, el uso de uno u otro depende en gran medida del tipo de variables en nuestra base de datos:

- **Algoritmo de K-Means:** Se utiliza cuando los datos corresponden a una matriz numérica, es decir, los grupos se obtendrán utilizando variables numéricas.
- **Algoritmo de K-Modes:** Se utiliza cuando los datos corresponden a una dataframe de atributos categóricos, es decir, utilizando variables categóricas.
- **Algoritmo de K-Prototype:** Se utiliza en datos mixtos, es decir, cuando nuestros atributos corresponden a variables numéricas y categóricas.

Estos algoritmo necesitan los datos y el número de grupos K. Veremos más adelante cómo determinar candidatos para K.

Algoritmo K-Means

Es una técnica de Machine Learning de Aprendizaje No Supervisado basado en centroides obtenidos mediante el cálculo iterativo de

distancias. Las distancias son calculadas para asignar cada punto a un grupo. En K-Means, cada grupo está caracterizado por un centroide.

El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o clúster. Se suele usar la distancia cuadrática, por lo cual, en este algoritmo solo se utilizan variables numéricas.

En qué consiste

Paso 1: Se parte con K centros aleatorios.

Paso 2: Para cada punto, se calcula la distancia a estos centros. Y se asigna cada punto al centro más cercano.

Paso 3: EL nuevo centro se calcula como promedio de las observaciones dentro de cada clúster.

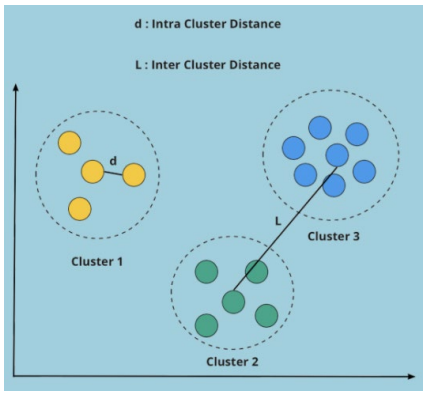
Paso 4: Se vuelven a obtener los centros y se itera hasta que ya no cambien.

Link GIF: shabal.in/visuals/kmeans/4.html

Sumas cuadráticas en K-Means

Dos conceptos muy importantes al realizar clustering es la varianza intra clúster e inter clúster:

Distancia intra clústers (Within Cluster Sums of Squares - WSS)

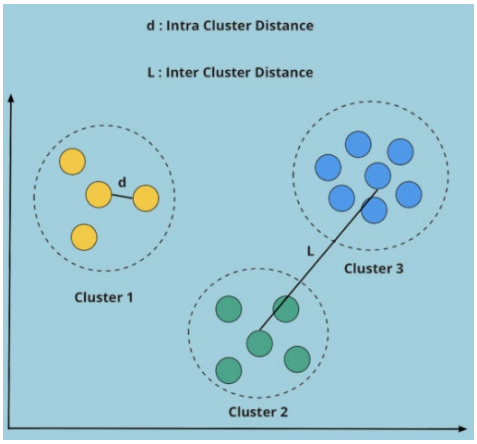


Indica qué tan separados están los elementos del mismo grupo y qué tan lejos están de su centroide. Los grupos más densos con elementos más cercanos tienen mayor similitud y son más probables para compartir realmente la misma calificación.

Se calcula:

$$WSS = \sum_{i=1}^{N_c} \sum_{x \in C_i} d\left(x, \underline{x}_{C_i}\right)^2$$

Donde N_c es el número de clúster, C_i corresponde al clúster, x la observación y $\underline{x}_{C_i}$ el centroide del clúster.



Distancia entre clústers (Between Cluster Sums of Squares - BSS)

Indica qué tan lejos están los grupos entre sí. Es una buena señal cuando están distantes entre sí porque muestran un mayor aislamiento de los grupos y probablemente una clasificación más precisa.

Se calcula:

$$BSS = \sum_{i=1}^{N_c} |C_i| \cdot d\left(\underline{x}_{C_i}, \underline{x}\right)^2$$

Donde $\underline{x}$ es la media global y $|C_i|$ corresponde al clúster.

Interpretación de las sumas cuadráticas

- **Distancia intra clústers (WSS):** Mide la variabilidad de las observaciones dentro de cada grupo. Una menor suma cuadrática indica que los grupos son más más compactos o pequeños. El WSS también está influenciado por el número de observaciones, a medida que aumenta el número de observaciones, notará que la suma de cuadrados aumenta. Esto significa que WCSS a menudo no es directamente comparable entre grupos con diferentes números de observaciones.
- **Distancia entre clústers (BSS):** Mide la distancia promedio al cuadrado entre todos los centroides, lo cual representa la dispersión entre todos los grupos. Un valor grande puede indicar clústeres que están dispersos, mientras que un valor pequeño puede indicar clústeres que están más cerca unos de otros.

Estos dos criterios pueden utilizarse como complemento para la elección del número de grupos según el *criterio de inercia* que indica que el agrupamiento será óptimo, cuando los grupos formados tengan una distancia mínima intra clúster y una distancia máxima entre clúster.

Elección del mejor K

Abordaremos dos métodos que pueden ser útiles para encontrar el valor adecuado de K en K-Means.

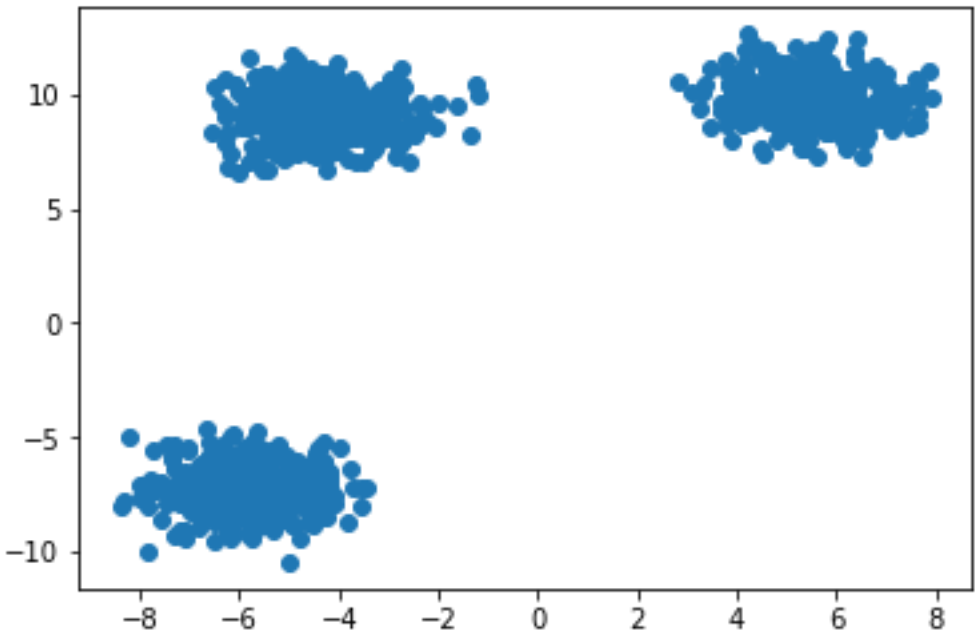
El método del codo

Usaremos nuestro propio conjunto de datos generado por el siguiente código para una ilustración de los dos métodos:

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

```
# Crear un conjunto de datos con 3 centros de cluster aleatorios y 1000 puntos de datos

x, y = make_blobs(n_samples = 1000, centers = 3, n_features=2, shuffle=True, random_state=31)
```

Así es como se ven los datos gráficamente:



Claramente, el conjunto de datos tiene 3 grupos. Validaremos nuestros dos métodos en este conjunto de datos.

Este es probablemente el método más conocido para determinar el número óptimo de grupos. *También es un poco ingenuo en su enfoque.*

Calcule la ***suma dentro del grupo de*** *errores al cuadrado (WSS) para* ***diferentes valores de k*** *y elija la k para la que WSS se convierte en el primero que comienza a disminuir. En el gráfico de WSS-versus-k, esto se ve como un* ***codo.***

La suma dentro del grupo de errores al cuadrado suena un poco compleja. Vamos a desglosarlo:

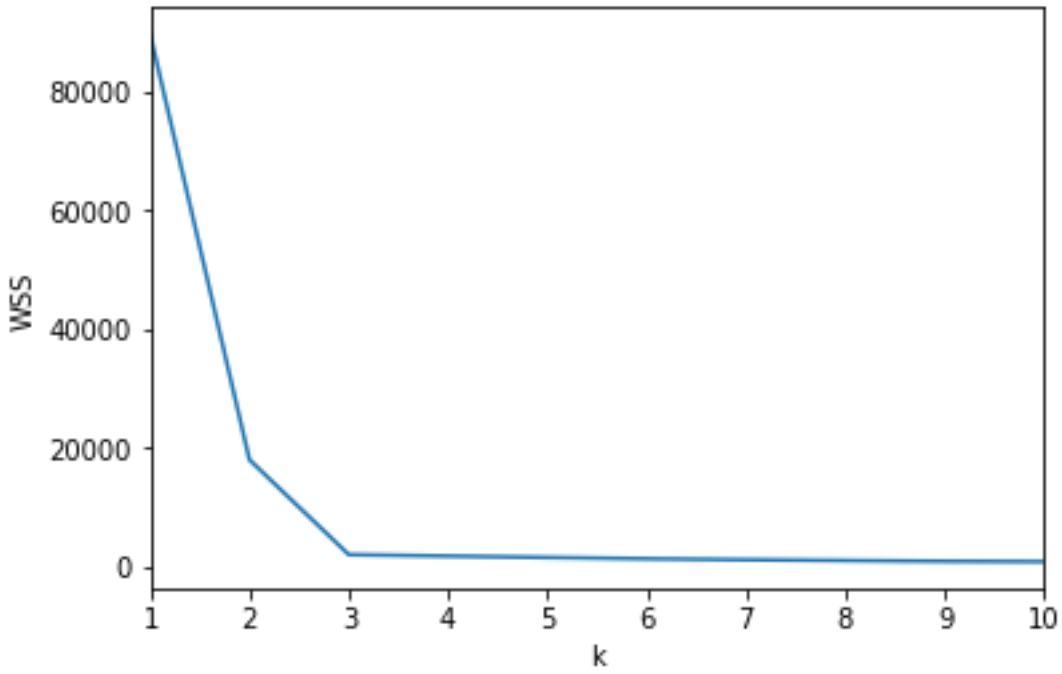
- El error cuadrático de cada punto es el cuadrado de la distancia del punto desde su representación, es decir, su centro de agrupación predicho.
- La puntuación WSS es la suma de estos errores al cuadrado para todos los puntos.
- Se puede utilizar cualquier métrica de distancia, como la distancia euclidiana o la distancia de Manhattan.

Implementemos esto en Python usando la biblioteca ***sklearn*** y nuestra propia función para calcular WSS para un rango de valores para k.

```
from sklearn.cluster import KMeans

# la función devuelve la puntuación WSS para los
valores k de 1 a kmax

def calculate_WSS(points, kmax):

  sse = []

  for k in range(1, kmax+1):

    kmeans = KMeans(n_clusters = k).fit(points)

    centroids = kmeans.cluster_centers_

    pred_clusters = kmeans.predict(points)

    curr_sse = 0


    # calcular el cuadrado de la distancia euclidiana
de cada punto desde su centro de cluster y añadirlo
al WSS actual
```

```
    for i in range(len(points)):
      curr_center = centroids[pred_clusters[i]]
      curr_sse += (points[i, 0] - curr_center[0]) **
2 + (points[i, 1] - curr_center[1]) ** 2


    sse.append(curr_sse)
  return sse
```

Obtenemos la siguiente gráfica para WSS-vs-k para nuestro conjunto de datos.



Como era de esperar, la **trama parece un brazo con un codo despejado en k = 3.**

Desafortunadamente, no siempre tenemos datos tan claramente agrupados. Esto significa que el codo puede no estar claro y afilado.

Para el conjunto de datos A, el codo está claro en k = 3. Sin embargo, esta elección es ambigua para el conjunto de datos B. Podríamos elegir que k sea 3 o 4.

Desventajas del K-Means

Ya hemos visto la potencia que tiene este algoritmo. Por lo sencillo que es de aplicar y la valiosa información sobre nuestros datos que nos aporta. Como no es oro todo lo que reluce, tengo que comentaros también las desventajas que ofrece:

- **Tenemos que elegir *k*** nosotros mismos. Es muy posible que nosotros cometamos un error, o que sea imposible escoger una k óptima.
- Es **sensible a outliers**. Los casos extremos hacen que el clúster se vea afectado. Aunque esto puede ser algo positivo a la hora de detectar anomalías.
- Es un algoritmo que sufre de la maldición de la dimensionalidad.

Referencias

[1] K-means

https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html

[2] Elección del valor de K

https://jarroba.com/seleccion-del-numero-optimo-clusters/

[3] Qué es el Tradeoff Bias – Variance

https://www.themachinelearners.com/tradeoff-bias-variance/

Material Complementario

[1] Algoritmo K-means

https://www.youtube.com/watch?v=kvKxsFfsVTg&feature=youtu.be

[2] Cluster de k-means

https://www.youtube.com/watch?v=EZOab1vkFmI