Clustering: algoritmos de agrupamiento

Aprendizaje no supervisado

Aprendizaje NO supervisado: no tenemos los datos de salida, solo de entrada. Para darnos info de los datos para tomar decisiones. Para predecir cosas también.

Objetivo: encontrar grupos de instancias tales que las instancias en un cluster sean similares entre sí y diferentes de las instancias en otros clusters. Sirve para tener una mejor perspectiva de los datos. Queremos minimizar la distancia intracluster y maximizar la distancia intercluster.

La definición de cluster depende de la naturaleza de los datos y los resultados deseados.

Mientras más información hay sobre nuestros datos, más difícil es visualizarlos de manera clara. Para muchas dimensiones se pueden aplicar algoritmos de reducción dimensional.

Cluster globular: clusters con forma de globos.

Algoritmos de clustering:

K-means: se divide los datos en k clusters.
Algoritmo iterativo. Condiciones de corte: cantidad de iteraciones y convergencia de los centroides.

1. Elegir k centroides al azar
2. Asignar cada instancia al centroide más cercano
3. Recomputar el centroide de cada cluster
4. Repetir paso 2 y 3 hasta que los centroides no cambien

Busca minimizar la ecuación de distorsión de clustering:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \cdot ||x^{(i)} - \mu_k||^2$$

$$r_{ik} = \begin{cases} 1 & \text{si } x^{(i)} \text{ está asignada al cluster } k \\ 0 & \text{en caso contrario} \end{cases}$$

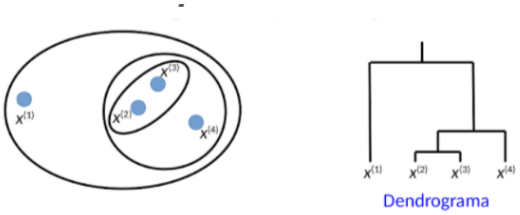
J es la suma de las distancias al cuadrado de cada instancia $x^{(i)}$ a su centroide μ_k

Ventajas: Simple, eficiente, escala muy bien. Funciona muy bien para clusterings globulares. Si no no se garantiza el correcto funcionamiento.
Desventajas: Hay que especificar k, Sensible a ruido y outliers, Muy sensible a elección de centroides iniciales, sólo puede encontrar clusters globulares.

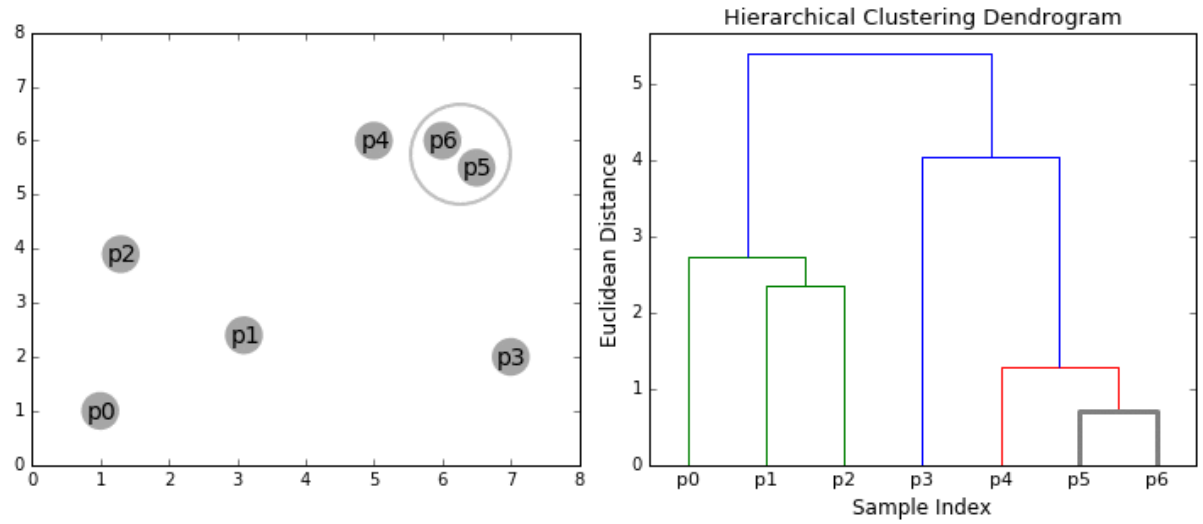
Los centroides se eligen de manera random. Se hacen varias corridas de K means y nos quedamos con el que mejor score nos de, el que menor error nos da. Para cada corrida (aprox 3) se eligen random. Si elegimos un K malo tal vez el resultado no tiene sentido y no tendrá uso.

Clustering jerárquico aglomerativo: construye una jerarquía de clusters.

El algoritmo cuando empieza asigna a cada instancia a un cluster propio.
Cada instancia comienza en un cluster distinto y se une con el más cercano a medida que va subiendo en la jerarquía.



Termina cuando todo queda agrupado en un solo cluster. El enfoque es bottom-up.
En base a la cantidad de clusters que tenemos podemos ver cuándo cortar.



Clusters iniciales -> cada dato.

Distancia entre clusters:

- Distancia mínima entre clusters
- Distancia máxima entre clusters
- Distancia entre centroides
- Promedio de las distancias del grupo

Cada definición tiene sus pros y sus contras respecto a sensibilidad a ruido y outliers y forma de los clusters que pueden manejar.

Ventajas: No hay que especificar k. Dendrograma: útil para crear taxonomías
Desventajas: No busca optimizar una función objetivo global. O(n2 log n) en tiempo O(n2) en espacio. Sensible a ruido y outliers

Clustering jerárquico divisivo: Todas las instancias comienzan en un único cluster. Los clusters se van dividiendo a medida que se va bajando de jerarquía. Enfoque top-down.

DBSCAN: crea clusters basándose en las densidades de los datos y las densidades de cada clase.

Usa dos parámetros, la cantidad de puntos y densidad. A es el centro y el radio es el Ɛ.

Los core points son los puntos que tienen densidad alta.
Los border points son los puntos que no son core pero son vecinos de un punto core.
Los noise points son los puntos que no son ni core ni border.

1. Etiquetar cada punto como *core*, *border* o *noise*
2. Eliminar todos los puntos *noise*
3. Poner una arista entre cada par de puntos *core* que son vecinos entre sí
4. Cada componente conexa corresponde a un cluster
5. Asignar los puntos *border* a uno de los clusters vecinos. (Puede ser necesario desempatar entre 2+ clusters)

https://dashee87.github.io/images/DBSCAN_tutorial.gif

Ventajas: No hay que especificar K, Puede encontrar clusters de formas arbitrarias, Es robusto ante el ruido.
Desventajas: Elegir el valor de Ɛ y Min Pts requiere conocimiento de los datos, difícil en casos de alta dimensionalidad, funciona mal con datos con densidad variable.

No es bueno tener un solo Ɛ con distintas densidades. DBSCAN no va a funcionar bien. A las zonas muy densas las toma como clusters y a las zonas poco densas las toma como ruido.

En caso de cluster no globular, mejor que k means.

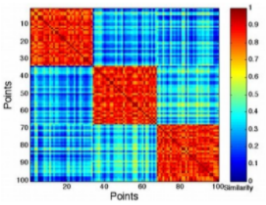
Un Ɛ muy grande considera un único cluster.

HDBSCAN: hereda de DBSCAN, pero con Ɛ variable y densidad variable. Más rápido.
Se adapta más al dataset que tenemos. Es una mejora de DBSCAN.

Evaluación de clusters:

Matriz de similitud:

El clustering es bueno si la MS es diagonal por bloques (la matriz ordenada).
Es cuadrada y simétrica.
Muestra la similitud entre cada par de instancias.
La matriz ordenada está ordenada según el cluster de instancias. Cada bloque de la matriz es un cluster.
Se grafica con un heatmap.



Métricas basadas en cohesión y separación:

Cohesión: cuán estrechamente relacionados están los elementos de un mismo cluster. Queremos que en un cluster los elementos sean muy parecidos entre sí.

Separación: cuán distintos son los clusters entre sí. Queremos que la separación de los cluster sea grande, que un cluster sea muy diferente a otro.

Para medir esto tenemos el coeficiente de Silhouette. Para cada punto i.
a: la distancia promedio de xi hacia otros puntos del mismo cluster
b: la distancia promedio de xi hacia los puntos del cluster más cercano
Si (coeficiente) se mueve de [-1,1].

- Mientras más cerca de 1 mejor.
- Queremos que b >> a. La distancia de un cluster a otro sea mayor que la distancia entre los puntos de un mismo cluster. La cuenta es:

$$s_i = \frac{b_i - a_i}{max\{b_i, a_i\}}$$

Si es cercano a -1, no esta tan buena esa categorización. No hay una buena agrupación.

Rand index:

Mide la similitud entre dos resultados de clustering. Comparamos los resultados obtenidos de haber corrido los algoritmos de antes, o a mano también podría ser. Usamos rand index para ver que clustering anda mejor.
a-> pares que están en el mismo cluster en X y en el mismo en Y
b -> pares que están en distintos clusters en X y en distintos en Y
c -> pares que están en el mismo cluster en X pero en distintos en Y
d-> pares que están en distintos clusters en X y en el mismo en Y

0 -> los dos resultados no tienen elementos en común

1 -> los dos resultados son exactamente los mismos

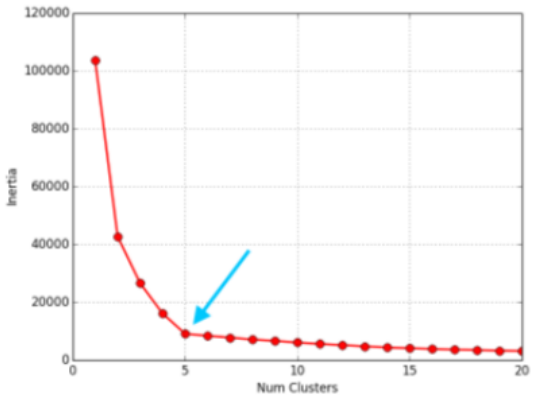
$$R = \frac{a+b}{a+b+c+d}$$

Va agrupando de a pares y nos da el index para ver qué tan similares son.

Elbow method:

Nos sirve para encontrar el número ideal de clusters. Se grafican los resultados para distintos números de clusters. Corremos K- means por ejemplo. Se elige el número a partir del cual los resultados no mejoran significativamente, dónde va disminuyendo el error. Es el codo de la curva. De acá obtenemos un K.

Voy calculando el error y cuando veo un codo paro.



Las métricas se acomodan al problema que tenemos. Hay una métrica óptima para cada caso.
Es más fácil aplicar Silhouette que rand index porque para el primero necesito un solo algoritmo para obtener el coeficiente, en cambio rand index no te da un valor para ese algoritmo, si no que necesita dos.

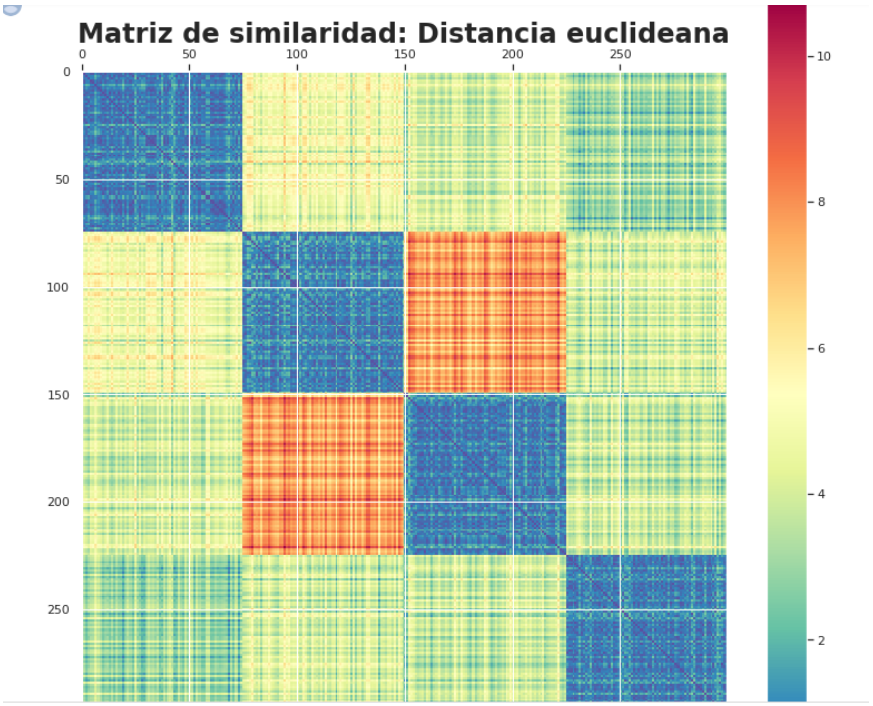
Práctica

La desviación estándar nos dice qué tan separados están los datos.
Si es alta, puede haber ruido y outliers, lo cual hace fallar a k means.
Los clusters del notebook son globulares.

Matriz de similaridad

Hay que ver si está en función de la **distancia** o de la **similaridad**

Chequear la escala.
A partir de estos colores podemos ver cuales clusters hay, En el gráfico del notebook hay 4.

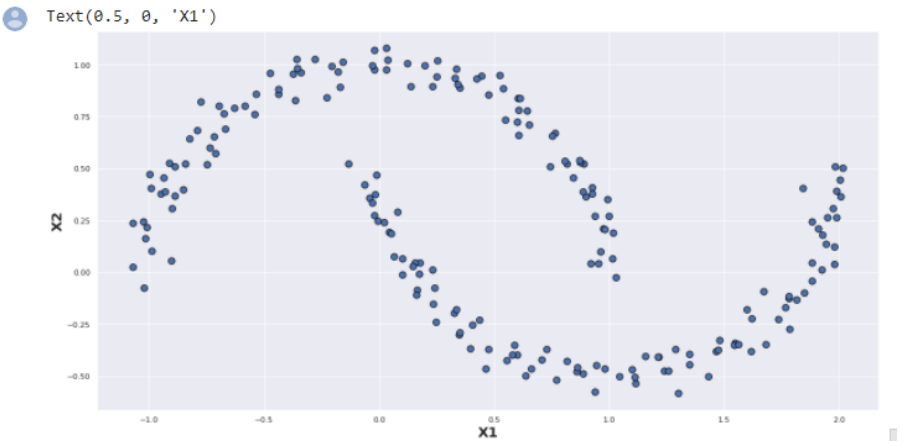


Los algoritmos con una semilla inicial nos puede dar un mínimo local en lugar de darnos el mínimo global. Hay que correrlo varias veces.

Como saber que cluster es con cual: tamaño del cuadrado. También que tan cerca están, la distancia de los cuadrados con respecto a los otros. En la dirección horizontal, con los números de arriba vemos la cantidad de puntos en el cluster.

<u>Coeficiente de silhouette</u>

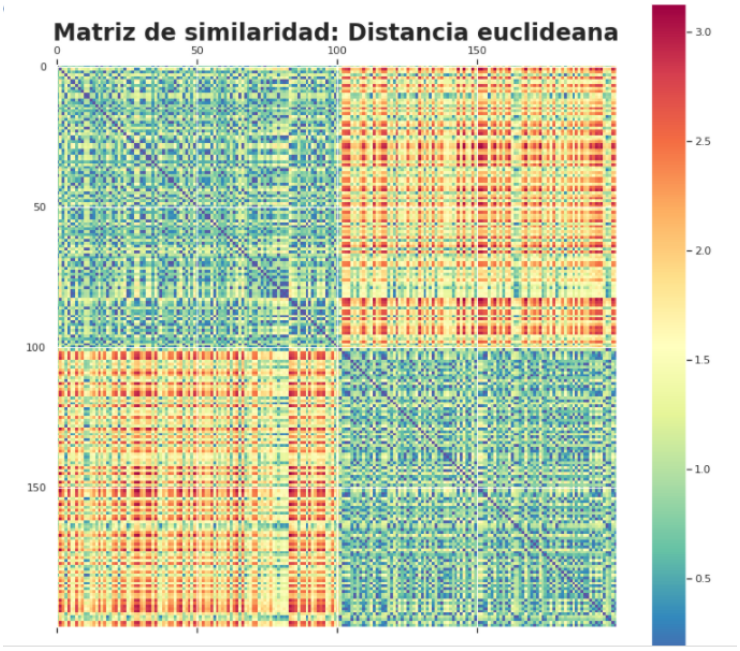
Va de -1 a 1. Si es más cercano a 1 es bueno. Cada matriz tiene su coeficiente.



Clustering aglomerativo

Este método nos permite tener un ordenamiento jerárquico de las observaciones en lo que llamamos un dendrograma: nos indica qué grupo de observaciones es más parecida a otra. Observaciones que se unen más abajo en el dendrograma son más similares. Podemos definir qué función de distancia usa el algoritmo aglomerativo para considerar unir o no un cluster.

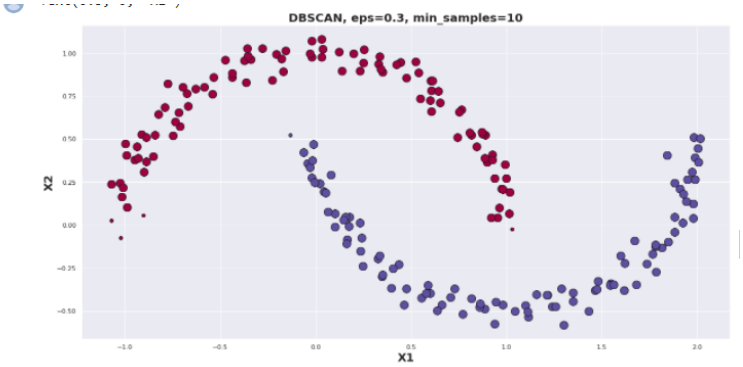
Kmeans no da bien por no ser globular.



Hay intersecciones de rojo y verde y azul con verde, cuando en realidad debería darnos todo rojo y todo azul. Nos sirve para ver que el clustering es malo. Sabemos que hay dos clusters grandes que están muy mal definidos, ya que tienen puntos muy lejanos entre sí y puntos muy cercanos entre los dos.
La distancia euclidiana no sirve.

El agglomerative clustering sigue dando mal por estar usando distancias euclidianas.

DBSCAN



Separa bien.

Cuando aumenta la cantidad de samples falla. Hay que ir jugando con el epsilon y los min samples.