Facultad de Ciencias, UNAM

Segmentación de clientes usando clustering

Reconocimiento de Patrones y Aprendizaje Automatizado

Oscar Andrés Rosas Hernandez

12 de junio de 2020

Índice general

Abstract

En este reporte mostraré como se puede realizar una segmentación de clientes de un centro comercial utilizando varios algoritmos de machine learning que vimos en el curso. Voy a comparar 2 métodos: KMeans y DBSCAN.

Este documento esta dividido en varias secciones: una introducción básica, un análisis de la lectura de datos y el preprocesamiento, análisis de datos exploratorios, aplicación de los algoritmos, comparación de ellos y una pequeña discusión.

Parte I

Marco Teórico

Capítulo 1

Aprendizaje No Supervisado

Definimos al machine Learning como el área que estudia como hacer que las computadoras puedan aprender de manera automática usando experiencias (data) del pasado para predecir el futuro.

A diferencia del aprendizaje supervisado, en el no supervisado solo se le otorgan las características, sin proporcionarle al algoritmo ninguna etiqueta. Su función es la agrupación, por lo que el algoritmo debería catalogar por similitud y poder crear grupos, sin tener la capacidad de definir cómo es cada individualidad de cada uno de los integrantes del grupo.

1.1. ¿Por qué es importante?¿Porque no usar solo supervisado?

Hasta ahora, en los 2 trabajos pasados habíamos explorado algoritmos y técnicas de aprendizaje automático supervisado para desarrollar modelos en los que los datos tenían etiquetas previamente conocidas.

En otras palabras, nuestros datos tenían algunas variables objetivo con valores específicos que utilizamos para entrenar nuestros modelos.

Sin embargo, cuando se trata de problemas del mundo real, la mayoría de las veces, los datos no vienen con etiquetas predefinidas, así que vamos a querer desarrollar modelos de aprendizaje automático que puedan clasificar correctamente estos datos, encontrando por sí mismos algunos puntos en común en las características, que se utilizarán para predecir las clases sobre nuevos datos.

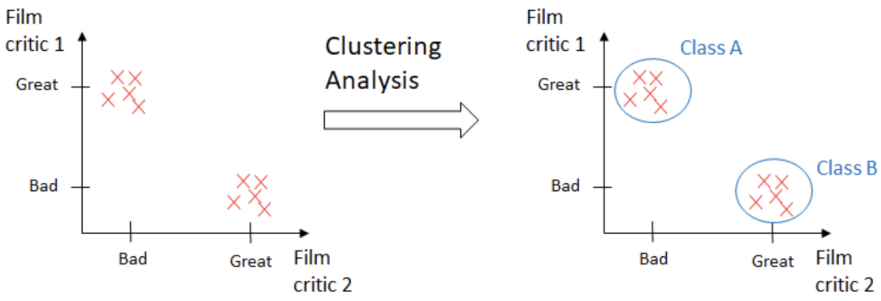
Capítulo 2

Clustering

La idea de usar clústers pertenece a técnicas de aprendizaje automático no supervisadas.

La tarea principal del clustering es descubrir grupos "naturales" (o en otras palabras agrupar a los datos según su similitud (la idea de saber que tan similares son dos elementos es algo muy interesante) en un conjunto de datos que no tienen etiquetas. Esta es una tarea es muy importante en el análisis de datos, ya que se utiliza en muchas aplicaciones científicas, de ingeniería y comerciales. En general se usa siempre que no tengas datos que no tengas etiquetas, y eso pasa muchas veces en la vida real.

La aplicación más conocida de clustering es la segmentación de clientes (para una segmentar clientes y hacer publicidad especifica eficiente), la segmentación de imágenes, la agrupación de documentos.



Existen muchos algoritmos de clustering que se pueden dividir en dos tipos principales: jerárquicos y particionales.

- Los algoritmos jerárquicos dividen recursivamente un conjunto de datos en un subconjunto más pequeño hasta que un subconjunto contenga solo un elemento. Esto se puede representar con un dendrograma que se parece a un árbol.

  Se puede construir desde las hojas hasta la raíz (enfoque aglomerativo) o desde la raíz hasta las hojas (enfoque divisivo). En la agrupación jerárquica, no tiene que especificar la cantidad de agrupaciones, sino que debe definir una condición de terminación para el proceso de división / fusión.

- Los algoritmos particionales dividen un conjunto de datos en varios subconjuntos (grupos) según un criterio dado. Para algunos algoritmos, el número de grupos debe definirse a priori (por ejemplo, K-Means) y para algunos no (DBSCAN).

  La definición del número de clústeres antes de ejecutar un algoritmo a menudo requiere un conocimiento de dominio específico que a menudo es desafiante (o incluso imposible) en muchas aplicaciones. Esto condujo al desarrollo de muchas heurísticas y enfoques simplificados que ayudaron a los analistas sin conocimiento de dominio a elegir el número apropiado de grupos.

  Y si, se que algoritmos como DBSCAN puede verse como otra categoria pero si lo piensas bien, no es mas una subcategoría.

Hay una gran cantidad de algoritmos de clustering y actualmente no hay uno solo que domine a otros. Elegir la mejor depende de la base de datos en sí, un dominio de los datos y los requisitos y expectativas del análisis y demás parámetros específicos del problema.

El objetivo del uso de clusters es identificar patrones o grupos de objetos similares dentro de un conjunto de datos de interés.

Cada grupo contiene observaciones con un perfil similar de acuerdo con un criterio específico. La similitud entre las observaciones se define utilizando algunas medidas de distancia entre observaciones, incluidas las medidas de distancia euclidiana y de correlación.

Estas son muy usadas en muchos campos, algunos son:

- En la investigación del cáncer, para clasificar a los pacientes en subgrupos según su perfil de expresión génica. Esto puede ser útil para identificar el perfil molecular de pacientes con pronóstico bueno o malo, así como para comprender la enfermedad.

- En marketing, para la segmentación del mercado mediante la identificación de subgrupos de clientes con perfiles similares y que podrían ser receptivos a una forma particular de publicidad.

- En la planificación urbana, para identificar grupos de casas según su tipo, valor y ubicación.

Como ya vimos la idea de clustering no hace referencia a algoritmos específicos, pero es un proceso para crear grupos basados en medidas de similitud. El análisis de clustering utiliza un algoritmo de aprendizaje no supervisado para crear estos clusters.

Los algoritmos de clustering generalmente funcionan según el principio simple de maximización de similitudes intragrupo y minimización de similitudes entre grupos. La medida de similitud determina cómo se deben formar los grupos.

La similitud es una caracterización de la proporción del número de atributos que comparten dos objetos en común en comparación con la lista total de atributos entre ellos.

Los objetos que tienen todo en común son idénticos y tienen una similitud de 1.0. Los objetos que no tienen nada en común tienen una similitud de 0.0.

La agrupación se puede adaptar ampliamente en el análisis de las empresas. Por ejemplo, un departamento de marketing puede usar la agrupación para segmentar a los clientes por atributos personales. Como resultado de esto, se pueden diseñar diferentes campañas de marketing dirigidas a varios tipos de clientes.

Parte II

Segmentación de clientes

Capítulo 3

El problema

El problema elegido fue el de segmentar clientes en cluster que los representen, buscamos comprender mejor a los clientes y brindar información de segmentación para que por ejemplo un equipo de marketing pudiera planificar una estrategia efectiva basada en nuestros clusters.

3.1. Importancia de resolverlo / Relevancia

Si bien las tácticas de marketing masivo aún pueden obtener resultados (la idea que usa DuckDuckGo), la suposición de que simplemente todos estarán interesados en comprar lo que está vendiendo es una estrategia costosa, ineficiente y una forma bastante mala de tirar dinero de marketing a la basura.

En lugar de un enfoque de "talla única", la segmentación exitosa agrupa los datos de sus clientes en grupos que comparten las mismas propiedades o características de comportamiento, lo que ayuda a impulsar el contenido dinámico y las tácticas de personalización para comunicaciones de marketing más oportunas, relevantes y efectivas (algo como lo que hace Facebook o Google).

Sin embargo, para que la segmentación se use correctamente, debe tener en cuenta que diferentes clientes compran por diferentes razones, y los especialistas en marketing deben aplicar de manera inteligente una serie de consideraciones que podrían afectar sus decisiones de compra.

Un profesor de la Harvard Business School incluso llegó a decir que, de 30,000 nuevos lanzamientos de productos de consumo cada año, el 95 % falla debido a la segmentación ineficaz del mercado.

[3]

Capítulo 4

El dataset / Business Understanding

4.1. ¿De donde salieron los datos?

Para poder solucionar este problema busque un dataset que fuera apropiedo en Kaggle, llegando a este:

`https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python`

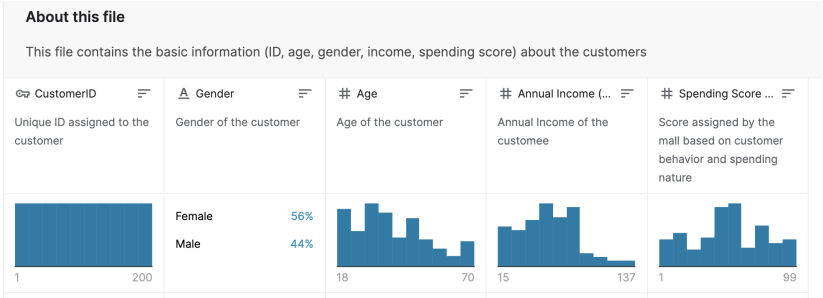
Ahora veamos un poco de contexto de este dataset:

Este conjunto de datos se creo solo con el propósito de aprender los conceptos de segmentación de clientes, también conocidos como análisis de la carrito de compras :v.

Este fue diseñado para usarse con la técnica de ml no supervisada (algoritmo de kmeans) en la forma más simple. Es decir es un dataset "didactico" publicado por el creador de un curso que explicaba kmeans.

Este dataset se supone que es de un centro comercial y, a través de tarjetas de membresía, se tienen algunos datos básicos sobre los clientes, como identificación del cliente, edad, sexo, ingresos anuales y puntaje de gastos.

La puntuación de gasto es algo que asigna al cliente en función de sus parámetros definidos, como el comportamiento del cliente y los datos de compra.



Capítulo 5

La propuesta

Mi hipotesis sería que usando este dataset y mediante aprendizaje no supervisado (kmeans y dbscan) se podría crear un sistema que fuera capaz de clasificar a los clientes en un grupos significativos.

El problema elegido fue el de segmentar clientes en cluster que los representen, buscamos comprender mejor a los clientes y brindar información de segmentación para que por ejemplo un equipo de marketing pudiera planificar una estrategia efectiva basada en nuestros clusters.

Capítulo 6

La implementación

6.1. Preprocesamiento

6.1.1. Preparación y conocimiento de los datos

Lo primero que tuvimos que hacer fue preparar los datos, el primer paso fue descargar el csv y abrirlo para explorar un poco la naturaleza del dataset.

Figura 6.1: Primero cargemos las librerias

```
In [19]: mall_data = read.csv('./mall_customers.csv', header = TRUE)
         mall_data = data.frame(mall_data)
         print sprintf("Dataset of %d row with %d columns each", nrow(mall_data), ncol(mall_data))

[1] "Dataset of 200 row with 5 columns each"
```

Figura 6.2: Leemos la información y la almacenamos en un dataframe

```
In [20]: head(mall_data)
         summary mall_data
```

A data.frame: 6 × 5

	CustomerID	Gender	Age	Annual.Income..k..	Spending.Score..1.100.
	<int>	<chr>	<int>	<int>	<int>
1	1	Male	19	15	39
2	2	Male	21	15	81
3	3	Female	20	16	6
4	4	Female	23	16	77
5	5	Female	31	17	40
6	6	Female	22	17	76

```
   CustomerID        Gender              Age        Annual.Income..k..
 Min.   :  1.00   Length:200         Min.   :18.00   Min.   : 15.00
 1st Qu.: 50.75   Class :character   1st Qu.:28.75   1st Qu.: 41.50
 Median :100.50   Mode  :character   Median :36.00   Median : 61.50
 Mean   :100.50                      Mean   :38.85   Mean   : 60.56
 3rd Qu.:150.25                      3rd Qu.:49.00   3rd Qu.: 78.00
 Max.   :200.00                      Max.   :70.00   Max.   :137.00
 Spending.Score..1.100.
 Min.   : 1.00
 1st Qu.:34.75
 Median :50.00
 Mean   :50.20
 3rd Qu.:73.00
 Max.   :99.00
```

Figura 6.3: Usamos las funciones de r para ver un poco mejor el dataset

6.1.2. Conociendo los datos

Hay 5 columnas dentro de nuestra información:

- ID de cliente: número de cliente numérico único
- Género: categórico, binario (hombre / mujer)
- Edad - numérica, entero
- Ingreso anual (k $) - numérico, entero
- Puntaje de gasto (1-100) - numérico, entero

6.1.3. Cosa curiosa: usar kmeans sobre columnas binaria

Hay una columna binaria, categórica: género. Recuerdo haber hablado con el profesor sobre que era una mala idea usar un numero para poner esa columna (por ejemplo hombre 1 y mujer 0), despues de eso pense en usar algo parecido al one hot encoding.

Y llegue a esta conclusion:

- técnicamente posible
- teóricamente no prohibido
- prácticamente no recomendado

Por qué no se recomienda, se explica muy bien en el sitio de soporte de IBM.

`https://www.ibm.com/support/pages/clustering-binary-data-k-means-should-be-avoided`

6.1.4. Limpieza de datos

No hay datos faltantes. Esto simplifica el análisis, pero es un escenario muy poco probable en la vida real, donde los científicos de datos se pueden pasan una cantidad significativa de tiempo limpiando sus datos antes de realizar el análisis central.

Eso si, es importante notar que para que todo funcione en los algoritmos que usaremos después vamos a quedarnos solo con las columnas numéricas, es decir, adiós el género como un dato categorico y al solo ser binario podemos usar valores numéricos.

In [248]:
```
# only numerical data
mall_data$Gender[mall_data$Gender == 'Male'] <- -1
mall_data$Gender[mall_data$Gender == 'Female'] <- 1
mall_data$Gender <- as.integer(as.character(mall_data$Gender))
head mall_data
```

A data.frame: 6 × 5

	CustomerID	Gender	Age	Annual.Income..k..	Spending.Score..1.100.
	<int>	<int>	<int>	<int>	<int>
1	1	-1	19	15	39
2	2	-1	21	15	81
3	3	1	20	16	6
4	4	1	23	16	77
5	5	1	31	17	40
6	6	1	22	17	76

6.2. Exploración de datos

Esta sección contiene una investigación estadística básica de una base de datos dada. Es un punto crucial en cualquier análisis, ya que permite una mejor comprensión de los datos.

Los datos los voy a separar por género, la única variable categórica.

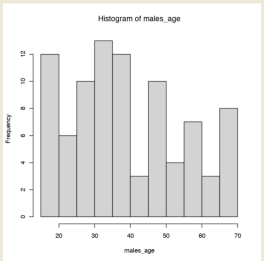
6.2.1. Edades

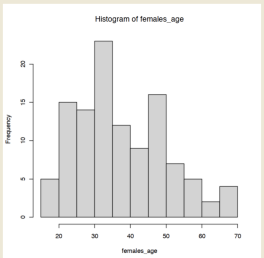
Distributions and exploring data

```
In [41]: males_age <- (mall_data %>% filter(Gender == 'Male') %>% select(Age))$Age
         females_age <- (mall_data %>% filter(Gender == 'Female') %>% select(Age))$Age

         ks.test(males_age, females_age)
         mean(males_age)
         mean(females_age)
```

```
Warning message in ks.test(males_age, females_age):
"cannot compute exact p-value with ties"


        Two-sample Kolmogorov-Smirnov test

data:  males_age and females_age
D = 0.11526, p-value = 0.5294
alternative hypothesis: two-sided


39.8068181818182


38.0982142857143
```

```
In [42]: hist males_age
         hist(females_age)
```

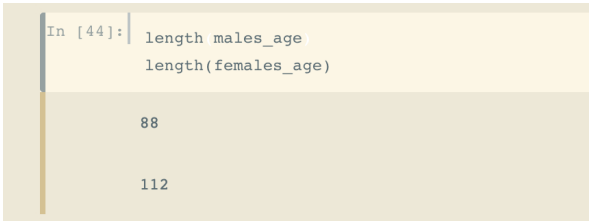




La edad promedio de los clientes masculinos es ligeramente más alta que la de las mujeres (39.8 versus 38.1). La distribución de la edad masculina es más uniforme que la femenina, donde podemos observar que el grupo de edad más grande es de 30 a 35 años.

Sin embargo, la prueba K-S muestra que las diferencias entre estos dos grupos son estadísticamente insignificantes.

Aquí hay un poco más de clientes femeninos que masculinos (112 vs. 88). Las mujeres representan el 56 % del total de clientes.



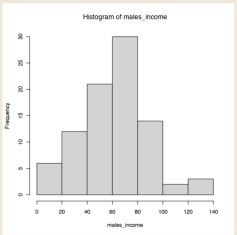
6.2.2. Ingresos

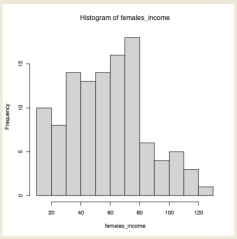
El ingreso promedio de los hombres es mayor que el de las mujeres (62.2k $ vs. 59.2k $). También el ingreso medio de los clientes masculinos (62.5k $) es mayor que el de las mujeres (60k $).

La desviación estándar es similar para ambos grupos. Hay un valor atípico en el grupo masculino con un ingreso anual de aproximadamente 140k $. La prueba K-S muestra que estos dos grupos no son estadísticamente diferentes.

```
In [51]: males_income <- (mall_data %>% filter(Gender == 'Male')
         females_income <- (mall_data %>% filter(Gender == 'Fema]

         mean(males_income)
         mean(females_income)
         hist(males_income)
         hist females_income
```

```
62.2272727272727

59.25
```



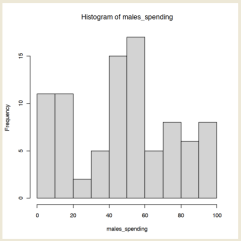


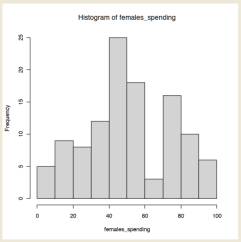
6.2.3. Gasto

Elgasto promedio para las mujeres (51.5) es más alto que el de los hombres (48.5). El valor p de la prueba K-S indica que no hay evidencia para rechazar la hipótesis nula, sin embargo, la evidencia no es tan fuerte como en comparaciones anteriores.

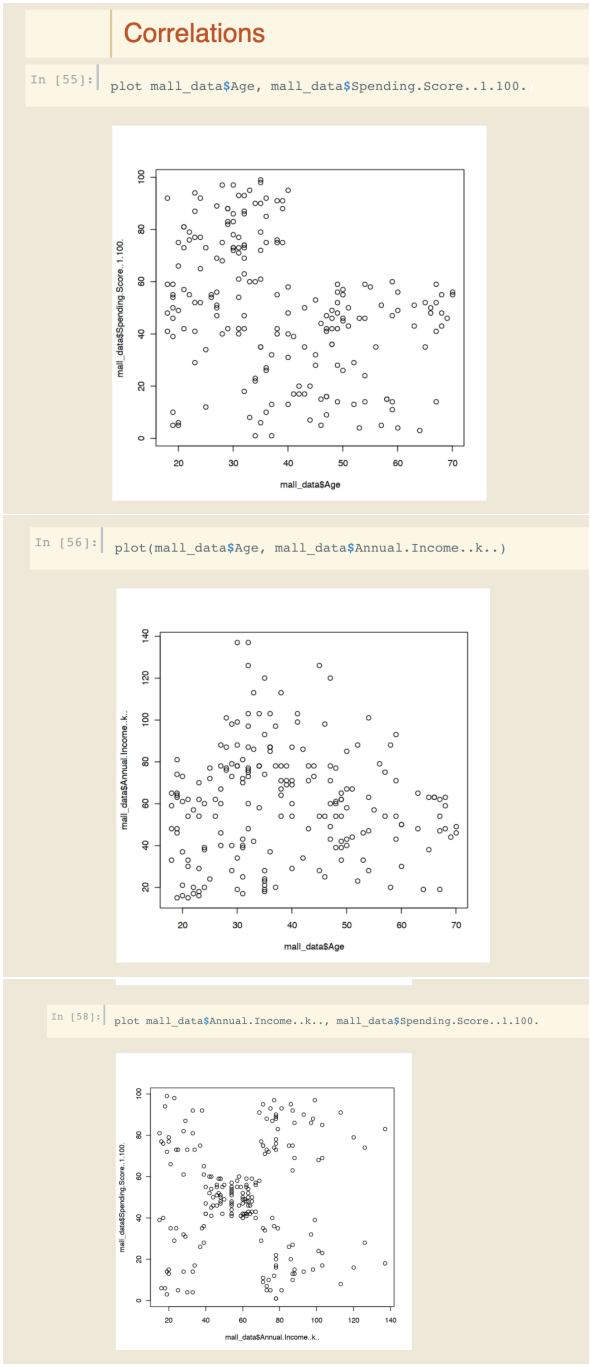
```
In [52]: males_spending <- (mall_data %>% filter(Gender == 'Male'
         females_spending <- (mall_data %>% filter(Gender == 'Fem

         mean(males_spending)
         mean females_spending
         hist(males_spending)
         hist(females_spending)
```

```
48.5113636363636

51.5267857142857
```





6.2.4. Posibles correlaciones que no encontre



Volvamos un poco atras y hablemos de los clasificadores que vamos a usar:

6.3. Técnica: K means

6.3.1. Historia y funcionamiento

La agrupación particional (o agrupación de partición) son métodos de agrupación utilizados para clasificar los elementos dentro de un conjunto de datos en múltiples grupos en función de su similitud. Los algoritmos requieren que el analista especifique el número de clústeres que se generarán.

K-Means, el algoritmo por excelencia de clustering, muy popular y el que se enseña en la mayoría de los cursos de aprendizaje automático.

Es ademas un algoritmo de agrupación particional. Fue desarrollado independientemente en muchos lugares en los años 50 y 60 y ganó gran popularidad debido a su facilidad de implementación, simplicidad y muchos éxitos empíricos (por ejemplo, en negocios, medicina y ciencia).

Hay 3 pasos principales en el algoritmo K-Means (también conocido como algoritmo de Lloyd):

- Dividir las muestras en grupos iniciales mediante el uso de puntos de semillas. Las muestras más cercanas a estos puntos de origen crearán grupos iniciales.
- Calcule las distancias de las muestras a los puntos centrales de los grupos (centroides) y asigne las muestras más cercanas a su grupo.
- El tercer paso es calcular los centroides de clúster recién creados (actualizados).

Luego simplemente hay que repetir los pasos 2 y 3 hasta que el algoritmo converja. Como se mencionó anteriormente, el objetivo de K-Means es minimizar la función objetivo (inercia) en todos los grupos.

La función objetivo comunmene esta definida como:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

Esto se conoce como problema NP-fuerte, lo que significa que es un algoritmo greeady y converge al mínimo local. El costo computacional del algoritmo K-Means de Lloyd es O (kn), donde k es una cantidad de grupos y n es una cantidad de muestras.

Esto no es malo en comparación con otros algoritmos de agrupación. A pesar de converger generalmente a un mínimo local, K-means. es relativamente rápido y cuando los

grupos están bien aislados unos de otros, es probable que converga al mínimo global. Debido a que el resultado de la agrupación depende de los criterios de inicialización, es común ejecutar el análisis para varios puntos de inicialización y elegir el que tenga la mínima inercia resultante.

Hay algunas mejoras en el algoritmo para resolver el problema de los mínimos locales. Una mejora ejemplar es utilizar algoritmos mejorados de Firefly.

Esto se conoce como problema NP-hard, lo que significa que es un algoritmo codicioso y converge al mínimo local. El costo computacional del algoritmo K-Means de Lloyd es O (kn), donde k es una cantidad de grupos yn es una cantidad de muestras. Esto no es malo en comparación con otros algoritmos de agrupación. A pesar de converger generalmente a un mínimo local, K-means es relativamente rápido y cuando los grupos están bien aislados unos de otros, es probable que converja al mínimo global. Debido a que el resultado de la agrupación depende de los criterios de inicialización, es común ejecutar el análisis para varios puntos de inicialización y elegir el que tenga la mínima inercia resultante. Hay algunas mejoras en el algoritmo para resolver el problema de los mínimos locales. Una mejora ejemplar es utilizar Algoritmos mejorados de Firefly sobre los cuales puede leer aquí.

En general, se requiere definir tres parámetros principales:

- Criterios de inicialización

  En muchas librerias, se implementa un ingenioso esquema de inicialización: "k-means ++"propuesto por Arthur y Vassilvitskii.

  Crea centroides iniciales generalmente distantes entre sí, lo que aumenta la probabilidad de obtener mejores resultados. También existe la posibilidad de utilizar un generador de puntos aleatorios. Hay esfuerzos continuos para crear el método de siembra más eficiente para el algoritmo K-Means, uno de ellos se basa en el análisis de componentes independientes.

- Numero de clusters

  La selección de varios grupos es la parte más difícil de configurar este algoritmo. No existen criterios matemáticos estrictos para esto y se han desarrollado muchos enfoques heurísticos / simplificados. Uno de los más simples y populares es el método de codo que se muestra en este análisis.

  También hay otras opciones, a menudo avanzadas, para elegir la cantidad óptima de clústeres, por ejemplo:

  - Longitud mínima del mensaje (MML)
  - Longitud mínima de descripción (MDL)
  - Criterio de información de Bayers (BIC)
  - Criterio de información de Akiake (AIC)

    - Proceso de Dirichlet
    - Estadísticas de brecha

- Una métrica de distancia (no se requiere en la implementación)

  Hay varias opciones para calcular la distancia entre puntos. La más popular es simplemente la métrica euclidiana y es la implementada en R. A menudo se le llama modelo esférico de k-medias. Tiene el inconveniente de que solo encuentra grupos de tipo esférico y tiende a inflarse en análisis altamente multidimensionales ("maldición de la dimensionalidad"). Hay otras opciones pero no implementadas en R por defecto, por ejemplo:

    - Distancia de Mahalonobis (alto costo de cómputo)
    - Distancia Itakura-Saito
    - L1 distancia
    - Distancia coseno
    - Distancia de Bregman

Algunas conclusiones sobre K-Means:

- Se utilizan distancias euclidianas
- Se debe definir el número de grupos para el algoritmo
- El centroide se calcula utilizando la distancia media a los miembros del grupo
- Los grupos se suponen isotrópicos y convexos.
- Algoritmo estocástico: los resultados dependen de los criterios de inicialización
- Crea grupos de igual varianza (minimiza la inercia)
- Propenso a la "maldición de la dimensionalidad"
- Se puede ejecutar en paralelo, por lo que se escala bien

6.3.2. Ventajas y desventajas

K-means es uno de los métodos de clustering más utilizados. Destaca por la sencillez y velocidad de su algoritmo, sin embargo, presenta una serie de limitaciones que se deben tener en cuenta.

Requiere que se ponga de antemano el número de clusters que se van a crear. Esto puede ser complicado si no se tiene la información adicional sobre los datos con los que

se trabaja. Se han desarrollado varias estrategias para ayudar a identificar potenciales valores óptimos de K (ver más adelante), aunque todas ellas son orientativas.

Las grupos resultantes pueden variar dependiendo de la asignación aleatoria inicial de los centroides. Para minimizar este problema se recomienda repetir el proceso de clustering entre 25-50 veces y seleccionar como resultado definitivo el que tenga menor suma total de varianza interna. Aun así, solo se puede garantizar la reproducibilidad de los resultados si se emplean semillas.

Presenta problemas de robustez frente a outliers. La única solución es excluirlos o recurrir a otros métodos de clustering más robustos como K-medoids (PAM).

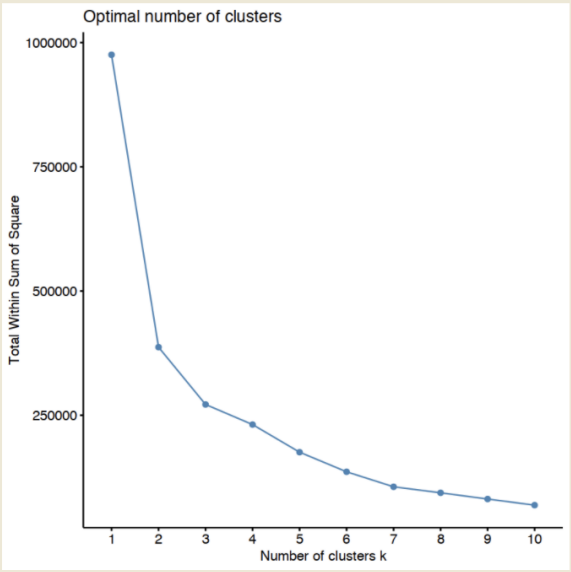
6.3.3. Apliquemoslo a los datos

¿Qué parametros usar?

Para encontrar un número apropiado de clusters, se utilizará una metrica bastante conocida, la del codo:

K means

```
In [69]: mall_data$Gender <- NULL
         fviz_nbclust(mall_data, kmeans, method = c("wss"))
```



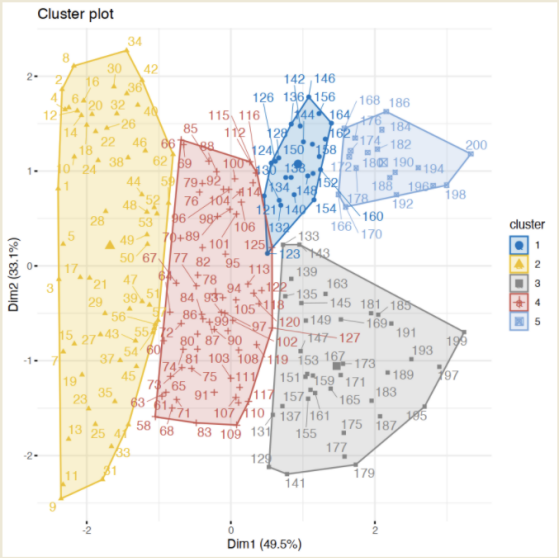
Esta de una resultado entre el 5 y el 6, asi que intentaremos con ambos para ver.

Para 5 paso esto:

El algoritmo K-Means generó los siguientes 5 grupos:

- clientes con bajo ingreso anual y alto puntaje de gasto
- clientes con ingreso medio y puntaje de gasto medio
- clientes con alto ingreso anual y bajo puntaje de gasto
- clientes con altos ingresos anuales y alto puntaje de gastos

In [63]:
```
set.seed(0)
km.res <- kmeans mall_data, 5
fviz_cluster(km.res, data = mall_data,
             ellipse.type = "convex",
             palette = "jco",
             repel = TRUE,
             ggtheme = theme_minimal())
```



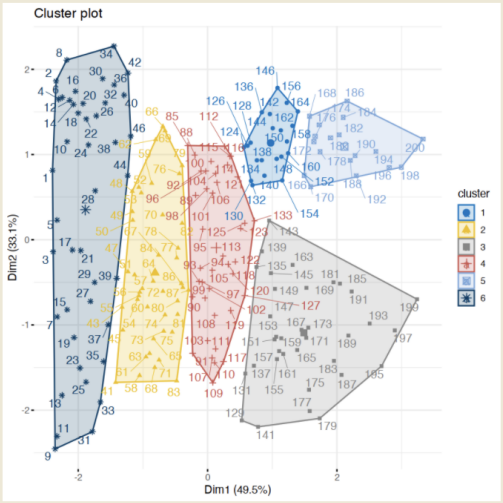

```
K-means clustering with 5 clusters of sizes 23, 59, 36, 64, 18

Cluster means:
  CustomerID      Age Annual.Income..k.. Spending.Score..1.100.
1   142.0870 33.17391           74.47826                80.52174
2    30.0678 36.10169           29.86441                49.86441
3   164.0000 40.80556           87.91667                17.88889
4    91.5625 44.20312           57.40625                49.01562
5   183.0000 32.16667           99.88889                81.38889

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2 4 4 2 4 4 4 4 4 4 4 4 4 4 4
 [75] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[112] 4 4 4 4 4 4 4 4 4 1 4 1 1 4 1 4 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1
[149] 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3
[186] 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5

Within cluster sum of squares by cluster:
[1]  7476.609 78832.949 33247.944 47088.531  7806.556
 (between_SS / total_SS =  82.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

- clientes con bajo ingreso anual y bajo puntaje de gasto

No hay grupos distintos en términos de edad de los clientes.

Para 6 paso esto:

```
In [64]: set.seed(0)
         km.res <- kmeans(mall_data, 6)
         fviz_cluster(km.res, data = mall_data,
                      ellipse.type = "convex",
                      palette = "jco",
                      repel = TRUE,
                      ggtheme = theme_minimal())
```




```
K-means clustering with 6 clusters of sizes 21, 43, 35, 40, 18, 43

Cluster means:
  CustomerID      Age Annual.Income..k.. Spending.Score..1.100.
1  144.00000 33.14286           75.09524                82.76190
2   64.90698 46.76744           47.32558                49.27907
3  164.88571 41.25714           88.37143                17.42857
4  106.70000 39.37500           62.87500                47.97500
5  183.00000 32.16667           99.88889                81.38889
6   22.13953 34.06977           25.44186                50.90698

Clustering vector:
  [1] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 [38] 6 6 6 2 6 2 6 2 6 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 2 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[112] 4 4 4 4 4 4 4 4 4 4 4 4 1 4 1 4 1 3 1 3 1 4 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1
[149] 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3
[186] 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5

Within cluster sum of squares by cluster:
[1]  5198.190 20269.395 31474.971 19785.125  7806.556 60410.186
 (between_SS / total_SS =  85.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

El algoritmo K-Means generó los siguientes 6 grupos:

- clientes más jóvenes con puntaje de gasto medio anual y medio

- clientes con alto ingreso anual y bajo puntaje de gasto
- clientes más jóvenes con puntaje de gasto medio anual y medio
- clientes con altos ingresos anuales y alto puntaje de gastos
- clientes con bajo ingreso anual y bajo puntaje de gasto
- clientes con bajo ingreso anual y alto puntaje de gasto

No hay grupos distintos en términos de edad de los clientes.

6.4. Técnica: DBSCAN

6.4.1. Historia y funcionamiento

El clustering basado en la densidad utiliza la idea de de densidad y conectividad de densidad (como alternativa a la medición de distancia), lo que la hace muy útil para descubrir un grupo en formas no lineales.

Este método encuentra un área con una densidad más alta que el área restante. Uno de los métodos más famosos es la Agrupación espacial basada en densidad de aplicaciones con ruido (DBSCAN). Utiliza el concepto de accesibilidad de densidad y conectividad de densidad.

El algoritmo de agrupamiento basado en densidad DBSCAN es una técnica fundamental de agrupamiento de datos para encontrar grupos de formas arbitrarias, así como para detectar valores atípicos.

A diferencia de K-Means, DBSCAN no requiere el número de clústeres como parámetro. Más bien, infiere el número de grupos basados en los datos, y puede descubrir grupos de forma arbitraria (en comparación, K-Means generalmente descubre grupos esféricos). Los métodos de partición (K-means, agrupación PAM) y la agrupación jerárquica son adecuados para encontrar agrupaciones de forma esférica o agrupaciones convexas. En otras palabras, funcionan bien para grupos compactos y bien separados. Además, también se ven gravemente afectados por la presencia de ruido y valores atípicos en los datos.

6.4.2. Ventajas y desventajas

Las ventajas de la agrupación basada en la densidad son:

- No se asume el número de grupos. El número de grupos a menudo se desconoce de antemano. Además, en un flujo de datos en evolución, el número de grupos naturales a menudo está cambiando.
- Descubrimiento de racimos con forma arbitraria. Esto es muy importante para muchas aplicaciones de flujo de datos.
- Capacidad para manejar valores atípicos (resistente al ruido).

Las desventajas de la agrupación basada en la densidad son:

- Si hay variación en la densidad, no se detectan puntos de ruido
- Sensible a los parámetros, es decir, difícil de determinar el conjunto correcto de parámetros.

- La calidad de DBSCAN depende de la medida de distancia.
- DBSCAN no puede agrupar bien los conjuntos de datos con grandes diferencias de densidad.

Una limitación de DBSCAN es que es sensible a la elección de e, en particular si los grupos tienen densidades diferentes. Si e es demasiado pequeño, los grupos más dispersos se definirán como ruido. Si e es demasiado grande, los grupos más densos pueden fusionarse.

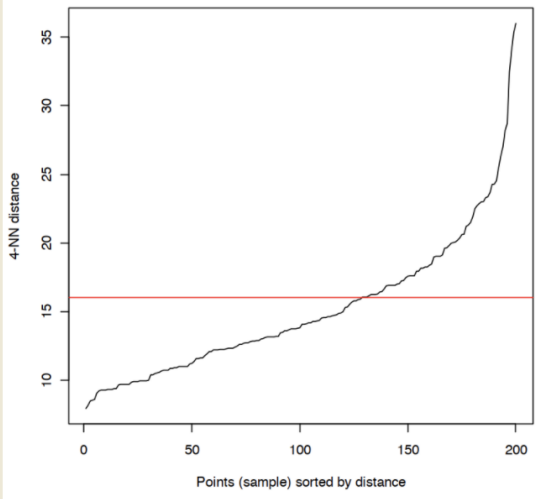
6.4.3. Apliquemoslo a los datos

¿Qué parametros usar?

Para encontrar el parametro de dbscan usaremos una funcion ya establecida:

DBSCAN

```
In [97]: library(dbscan)
         matrix = as.matrix(mall_data)
         kNNdistplot(matrix, k=4)
         abline(h=16, col="red")
```



Esta de una resultado entre el que e $= 13 + -3$, veamos los resultados de usarlo:

DBSCAN encontro entonces 6 clusters y uno de ruido, estos tienen una variacion muy grande, hay 25 puntos que clasifico como ruido. En la grafica podemos ver cuales fueron esos outliers, puntos que no tienen la distancia ni la cantidad de datos necesarios para ser un cluster.

Y aqui es donde la cosa se puso fea, porque no encontre algo que uniera a los clusters, por un lado por falta de tiempo pero tambien porque no encuentro una característica comun para los clusters que DBSCAN hizo.

```
In [102]: mall_data = dbscan(matrix, 16, 4)
          mall_data

          DBSCAN clustering for 200 objects.
          Parameters: eps = 16, minPts = 4
          The clustering contains 6 cluster(s) and 25 noise points.

            0   1   2   3   4   5   6
           25 101  10   7  33   4  20

          Available fields: cluster, eps, minPts
```

```
In [103]: hullplot matrix, mall_data$cluster
```

6.5. Posibles mejoras a futuro

- Me puse a investigar mucho sobre un algoritmo conocido como .ªffinity Propagationz siento que seria una gran comparación usarlo.
- Encontrar un patrón sobre los clusters de DBSCAN.

6.6. Conclusión

Gracias a los resultados vimos está claro que DBSCAN no pudo generar grupos razonables. Se debe a sus problemas para reconocer grupos de varias densidades (que están presentes en este caso).

A su vez, el algoritmo de K-Means creo 5-6 grupos razonables y que se pudieron con algo de análisis encontrar a que grupos correspondían.

6.7. Evidencia

Por si quieres corroborar todo lo que aquí se experimento deje adjunto una libreta que puedes usar para jugar con le código

Bibliografía

[1] *k MEANS*, https://www.datanovia.com/en/courses/advanced-clustering/

[2] *DBSCAN*, https://www.datanovia.com/en/lessons/dbscan-density-based-clustering-essentials/

[3] *Anthony Botibol*, The Importance of Customer Segmentation https://www.bluevenn.com/blog/the-importance-of-customer-segmentation