Facultad de Ciencias, UNAM

Segmentación de clientes usando clustering

Reconocimiento de Patrones y Aprendizaje Automatizado

Oscar Andrés Rosas Hernandez

7 de junio de 2020

Índice general

Parte I

Marco Teórico

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.

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.

Hasta ahora, tan en los 2 trabajos pasados habiamos 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.

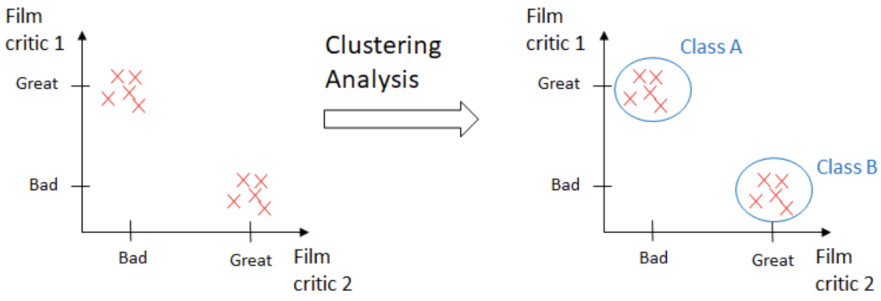
Capítulo 2

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 comercialización 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.

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 analisis.

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.

El modelo de clustering es una noción utilizada para indicar qué tipo de clustering estamos tratando de identificar. Los cuatro modelos más comunes de métodos de agrupación son la agrupación jerárquica, la agrupación de k-means, la agrupación basada en modelos y la agrupación basada en densidad.

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

Si bien las tácticas de marketing masivo aún pueden obtener resultados, 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.

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.

[?]

Capítulo 4

El dataset / Business Understanding

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

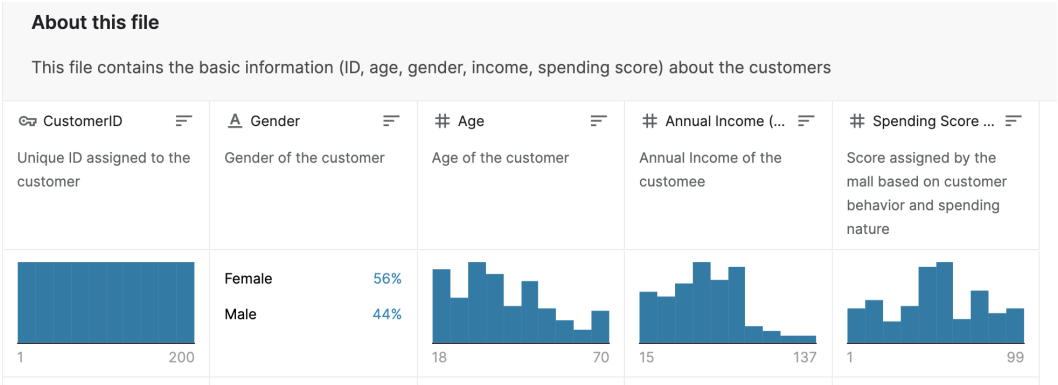
`https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python`

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.

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. 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.

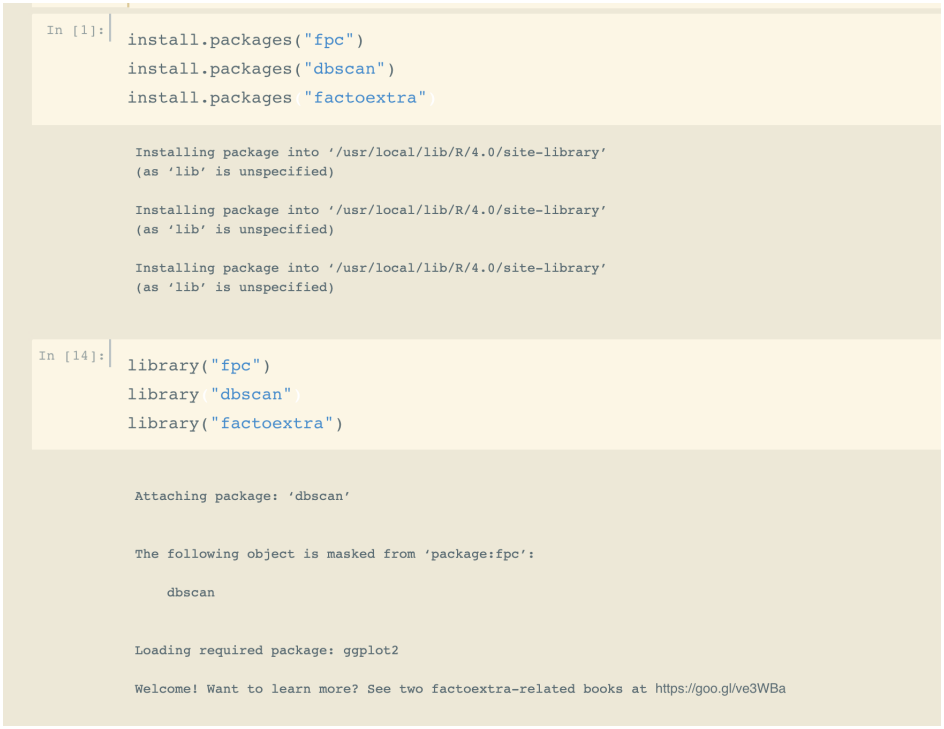


Figura 6.1: Primero cargemos las librerias

```
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

6.1.1. 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

6.1.2. Resultados, 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`

6.1.3. Creación del clasificador

Esto fue mucho mas sencillo e igualmente buscamos crear el modelo mas sencillo que fuera capaz de resolver el problema, llegando a que el parametro que mas afectba el desempeño del sistema era la cantidad de elementos necesario para crear una hoja, un valor de 15 resulto ser mucho mas efectivo en reducir el tamaño del arbol y aumentar el desempeño.

Ahora tambien fue importante cambiar el tamaño del bache a 64, siendo una creencia común que conocía y comprobé que usar una potencia de 2 aumenta significativamente el tiempo de entrenamiento.

6.1.4. Evaluación

Lo que hicimos para evaluar al sistema fue partir nuestros datos, elegimos un 80 %, algo muy importante (y que me tomo dos horas darme cuenta y cambio el resultado del sistema de un 40 % a un 85 %) es que antes de partir los datos lso "revolvimos", para evitar que cierto grupo de etiquetas quedara sobrerepresentado en algun conjunto.

6.1.5. interpretación

Con esto listo creamos un evaluador y medimos el resultado de nuestro clasificador.

```
@attribute body string
@attribute urgency {0,1,2,3}
48549
101

Correctly Classified Instances        8295               85.4274 %
Incorrectly Classified Instances      1415               14.5726 %
Kappa statistic                          0.6738
Mean absolute error                      0.0867
Root mean squared error                  0.218
Relative absolute error                 37.9172 %
Root relative squared error             64.5128 %
Total Number of Instances             9710
```

En este podemos ver inmediatamente que nuestro sistema cumplio nuestro objetivo, logran un mas de 85 % de accuracy en el conjunto de prueba.

Ademas viendo a la matriz de confusión vemos que los errores se distribuyen de manera informe. Finalmente y como una posible forma de llevarlo a las manos de los usuarios cree un programa que usa los modelos y al que tu le pasas un texto por la terminal y te regresa su nivel estimado de urgencia.

```
→  Code git:(master) ✗ javac -cp weka.jar Tests.java && java -cp weka.jar:. Tests hello man
3
→  Code git:(master) ✗
```

6.1.6. Codigo

Codigo principal

Compilarse usando: javac -cp weka.jar Project.java && java -cp weka.jar:. Project y java v13

```
@attribute body string
@attribute urgency {0,1,2,3}
48549
101

Correctly Classified Instances        8295               85.4274 %
Incorrectly Classified Instances      1415               14.5726 %
Kappa statistic                          0.6738
Mean absolute error                      0.0867
Root mean squared error                  0.218
Relative absolute error                 37.9172 %
Root relative squared error             64.5128 %
Total Number of Instances             9710
```

Codigo de pruebas del usuario final

6.1.7. Posibles mejoras a futuro

- Por un lado seria interesante ver como usando $IDFTransform/TFTransform$ podriamos alterar los resultados.
- Debido a que los datos no estan balanceados usar el accuracy no es la mejor idea y se debe hacer con cuidado.

6.1.8. Conclusión

Gracias a los resultados vimos que es posible crear un algoritmo con la ayuda de los árboles de decisión que nos permite clasificar con un gran desempeño la urgencia de un "support ticket".

Bibliografía

[1] *Aidan Wilson*, Sep 29, 2019
`https://towardsdatascience.com/a-brief-introduction-to-supervised-learning`

[2] *Diego Lopez Yse*, Apr 17, 2019
`https://towardsdatascience.com/the-complete-guide-to-decision-trees`

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