S5_LSC1_MBA

January 14, 2022

ESPACIO PARA BANNER DE LA MAESTRIA

---

1 Análisis de Canasta de Compra. Fundamentos Teóricos.

Este *cuaderno* trata sobre el análisis de canasta de compra. Este análisis ayuda a los vendedores a comprender las preferencias de sus clientes y mejorar sus sistemas de recomendación. El objetivo del *cuaderno* es que usted aprenda que es el análisis de canasta de compra, los conceptos teóricos y el algoritmo que lo implementa, reconociendo sus características y funcionamiento, y cómo ejecutarlo.

NO es necesario editar el archivo o hacer una entrega. Sin embargo, los ejemplos contienen celdas con código ejecutable (`en gris`), que podrá modificar libremente. Esta puede ser una buena forma de aprender nuevas funcionalidades del *cuaderno*, o experimentar variaciones en los códigos de ejemplo.

1.1 Introducción

Las preferencias de los individuos suelen seguir patrones que los sistemas de recomendación pueden aprovechar, por ejemplo, si hemos visto recientemente la película de Iron Man, y la disfrutamos, es muy probable que también disfrutemos la película de Thor. Los sistemas de recomendación entonces encuentran y utilizan estos patrones para predecir que otros productos podrian gustarnos y generar sugerencias. Por ejemplo, cuando compramos en línea es común ver sugerencias de otros productos bajo titulos como "Comprados juntos habitualmente" o "Los clientes que compraron X también compraron Y". Según un estudio del 2013 de McKinsey, el 35% de los artículos comprados en Amazon surge de estos sistemas de recomendación.

En este *cuaderno*, vamos a explorar un algoritmo de sistema de recomendación conocido como análisis de canasta de compra o market basket analysis en inglés. Si bien este algoritmo puede no estar a la vanguardia de los algoritmos de aprendizaje no supervizado, es omnipresente y es innegable su impacto en los negocios.

1.2 ¿Qué es análisis de canasta de compra?

El análisis de canasta de compra consiste en buscar reglas de asociación en los consumos de individuos. La idea general es identificar y cuantificar qué items, o grupos de items, son consumidos juntos con la suficiente frequencia que nos permita etender el comportamiento de los clientes.

Antes de sumergirnos en los detalles matemáticos y del algoritmo, tenemos que definir el término canasta de compra: *Una canasta de compra es un conjunto permanente de productos en un sistema*

económico. Permanente no significa necesariamente permanente en el sentido tradicional. Significa, que hasta el momento en que el producto sea retirado del catálogo, estará siempre disponible para su compra. El producto al que se hace referencia en la definición anterior es cualquier bien, servicio o elemento de un grupo, incluida una bicicleta, escuchar una canción, pintar su casa o un sitio web. Por último, un sistema económico puede ser una empresa, un conjunto de actividades, o un país.

Este tipo de análisis suele aplicarse en tres casos principalmente:

1. Recomendaciones de productos, cupones y descuentos.
2. Mejora de precios.
3. Distribución espacial de los productos en las tiendas.

Las recomendaciones de productos, cupones y descuentos, y la mejora de precios son dos caras de la misma moneda. Imaginemos el caso de dos artículos fuertemente relacionados, lo más probable es que estos dos artículos se compren en la misma transacción. Una forma de aumentar la rentabilidad de esa transacción sería aumentando el precio de uno de los productos. Si la asociación entre los dos artículos es lo suficientemente fuerte, el aumento de precio puede ser tal que no incurra en el riesgo que no se compre ninguno de los artículos.

De manera similar, los comerciantes pueden alentar a los clientes a comprar un artículo débilmente asociado con otro a través de descuentos o cupones. Por ejemplo, al estudiar los historiales de compras de ciertos clientes se podría encontrar que estos están comprando artículos que estan débilmene asociados con otros productos que no están comprando. Con esta información, se pueden ofrecer descuentos e inducir a que los compren, aumentando por lo tanto las ventas.

Asimismo, las asociaciones de productos se pueden explotar para colocar productos estrategicamente en las tiendas y lograr que los clientes compren más artículos, gastando más dinero.

El ejemplo más sencillo de una canasta de compras es un supermercado, que es un sistema compuesto por una colección de alimentos y bebidas:

Incluso sin usar ningún modelo o análisis, ciertas relaciones de productos son obvias. Tomemos la relación entre la carne y las verduras (por lo general, los modelos de análisis de la canasta de compra arrojan relaciones más específicas que la que existe entre carne y las verduras, pero, supongamos por un momento que esta es así)

Dada esta relación, ¿qué podemos hacer? Bueno, sabemos que estos son artículos se compran juntos frequentemente. Podemos entonces aprovechar esta información colocando las verduras y las carnes en lados opuestos de la tienda, obligando a los consumidores a caminar toda la tienda, aumentando la probabilidad de que compren artículos adicionales que no hubiesen comprado de otra forma. Obviamente, son muchos factores que determinan el diseño de la tienda, pero el análisis de la canasta de compra es definitivamente uno de esos ellos.

En el análisis de la canasta de compra entonces estamos buscando grupos de productos que ocurren con frecuencia. Por lo tanto, se podría pensar a este análisis como uno de clustering. Sin embargo, la principales diferencias son que los grupos en el análisis de la canasta de mercado son micro (solo unos pocos productos por grupo) y el orden de los artículos en el grupo es importante cuando se trata de calcular métricas probabilísticas.

El mensaje final de esta sección, es que el analista puede descubrir relaciones, obvias y sorprendentes, entre productos y que una vez descubiertas, estas se pueden explotar para para informar y mejorar el proceso de toma de decisiones.

1.3 Reglas de asociación

Como dijimos anteriormente, el análisis de la canasta de compra busca de manera eficiente reglas de asociación en los datos. Pero antes de seguir tenemos que definir que es una regla de asociación:

Una regla de asociación es un caso en el que la probabilidad condicional del producto A dado que también compra el producto B es alta, mucho más alta que la probabilidad incondicional del producto A. Por ejemplo, tal vez cuando compra leche, también compra pan. Si sólo compra pan después de comprar leche, entonces habrá una regla de asociación: leche → pan. Sin embargo, si siempre compra pan independientemente de leche, entonces la probabilidad de comprar pan será alta, pero esta no es una regla de asociación, porque la leche no hace una diferencia.

Cada una de estas probabilidades que componenen la regla de asociación recibe un nombre particular en este contexto, las 5 métricas más importantes son: soporte, confianza, "lift", levante, y convicción.

- El **soporte** es la probabilidad de comprar el artículo:

$$soporte = Pr(X) \tag{1}$$

- La **confianza** es la probabilidad condicional

$$confianza = Pr(X|Y) \tag{2}$$

- El **lift** es un indicador que busca responder la pregunta: si un individuo compra el producto Y, ¿podemos decir algo acerca de si comprará o no el producto X con cierto nivel de confianza?. Se calcula como el cociente de la confianza y el soporte

$$lift = \frac{Pr(X|Y)}{Pr(X)} = \frac{Pr(X,Y)}{Pr(X)Pr(Y)} \tag{3}$$

  Su rango $[0, \infty]$ y nos da una medida del aumento de la probabilidad conjunta relativo a lo que esperariamos si X y Y fuesen independientes.
- El **levante** o **apalancamiento** por otro lado es un indicador que toma la diferencia

$$apalancamiento = Pr(X,Y) - Pr(X)Pr(Y) \tag{4}$$

  Su rango es $[-1, 1]$ y nos va a dar la dirección de la relación entre los items: valores postivos implican una asociación positiva, mientras que valores negativos indican una asociación negativa.
- La **convicción** es un indicador que se forma con el ratio de:
  - la probabilidad de que X ocurra sin Y, si X e Y son independientes.
  - la frecuencia observada de las predicciones incorrectas

$$conviccin = \frac{1 - Pr(Y)}{1 - Pr(X|Y} \tag{5}$$

Su rango [0, Infinito], valores del indicador superiores a 1 son ideales porque implica que la asociación entre X e Y es incorrecta más a menudo que si la asociación entre X e Y es aleatoria ($X \perp Y$). En otras palabras, esto quiere decir que existe la asociación entre X e Y. Un valor de 1 implica independencia, y un valor menor a 1 significa que la relación que indica que X e Y son independientes es correcta con más frecuencia que la relación X e Y que se ha definido como X→ Y. En esta situación, la relación podría ser inversa Y →X.

1.3.1 Ejemplo cálculo de las métricas

Ilustremos cómo calcular estas métricas en `Python` utilizando datos de 10 transacciones ficticias. Cada una de las entradas de la lista es una transacción y cada elemento son los productos que se compró en cada transaccion.

[1]:
```
transacciones = [
    ['leche', 'pan', 'manzanas', 'cereales', 'gelatina', 'galletas',␣
↪'ensalada', 'tomates'],
    ['cerveza', 'leche', 'papas', 'salsa', 'uvas', 'vino', 'papas', 'huevos',␣
↪'zanahorias'],
    ['pañales', 'fórmula para bebés', 'leche', 'pan', 'pollo', 'espárragos',␣
↪'galletas'],
    ['leche', 'galletas', 'pollo', 'espárragos', 'brócoli', 'cereales', 'jugo␣
↪de naranja'],
    ['filete', 'espárragos', 'brócoli', 'papas fritas', 'salsa', 'ketchup',␣
↪'patatas', 'ensalada'],
    ['cerveza', 'salsa', 'espárragos', 'vino', 'queso', 'galletas', 'fresas',␣
↪'galletas'],
    ['torta de chocolate', 'frutillas', 'vino', 'queso', 'cerveza', 'leche',␣
↪'jugo de naranja'],
    ['pollo', 'porotos', 'brócoli', 'leche', 'pan', 'huevos', 'patatas',␣
↪'ketchup', 'galletas saladas'],
    ['huevos', 'pan', 'queso', 'pavo', 'ensalada', 'tomates', 'vino', 'tapa de␣
↪asado', 'zanahorias'],
    ['pan', 'leche', 'tomates', 'cereal', 'pollo', 'pavo', 'papas fritas',␣
↪'salsa', 'pañales']
]
```

Podemos ver que hay 10 transacciones

[2]:
```
N = len(transacciones)
N
```

[2]:
```
10
```

La regla de asociación que queremos evaluar en este ejemplo es la de leche → pan. Para ello, necesitamos calcular las frecuencias de compras de cada uno de los artículos

[1]:
```
# frecuencia de compra de leche
f_x = sum(['leche' in i for i in transacciones])
# frecuencia de compra de pan
f_y = sum(['pan' in i for i in transacciones])
# frecuencia de compra de leche y pan
f_x_y = sum([
    all(w in i for w in ['leche', 'pan'])
    for i in transacciones
])
```

```
print(
    "N = {}\n".format(N) +
    "Freq(leche) = {}\n".format(f_x) +
    "Freq(pan) = {}\n".format(f_y) +
    "Freq(leche, pan) = {}".format(f_x_y)
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/7_/4x3t27892rv_5_f7bks427h80000gn/T/ipykernel_40507/2344945505.py␣
 ↪in <module>
      1 # frecuencia de compra de leche
----> 2 f_x = sum(['leche' in i for i in transacciones])
      3 # frecuencia de compra de pan
      4 f_y = sum(['pan' in i for i in transacciones])
      5 # frecuencia de compra de leche y pan

NameError: name 'transacciones' is not defined
```

Calculamos las métricas:

[4]:
```
# soporte (supp)
soporte = f_x_y / N
print("Soporte = {}".format(round(soporte, 4)))
```

```
Soporte = 0.4
```

[5]:
```
# confianza: x -> y

confianza = soporte / (f_x / N)
print("Confianza = {}".format(round(confianza, 4)))
```

```
Confianza = 0.5714
```

[6]:
```
# lift: x -> y

lift = confianza / (f_y / N)
print("Lift = {}".format(round(lift, 4)))
```

```
Lift = 1.1429
```

[7]:
```
# apalancamiento: x -> y

apalancamiento = soporte - ((f_x / N) * (f_y / N))
print("Apalancamiento = {}".format(round(apalancamiento, 4)))
```

```
Apalancamiento = 0.05
```

[8]:
```
# convicción: x -> y

conviccion = (1 - (f_y / N)) / (1 - confianza)
print("Convicción = {}".format(round(conviccion, 4)))
```

```
Convicción = 1.1667
```

1.4 Algortimo Apriori

Las reglas de asociación con un indicador de *lift* alto son las más útiles, pues suelen revelar asociaciones que no conomecemos de antemano. No existe, sin embargo, una teoría subyacente sobre estas regalas de asociación. En la práctica, lo que hacemos, es explorar los datos y buscar pares de productos, o grupos de productos que tengan indicadores altos de *confianza* y *lift*.

Operacionalmente esta exploración se hace utilizando el algoritmo de Apriori, que tiene dos etapas:

```
1. Identificar todos los productos que ocurren con una frecuencia por encima de un determinado
2. Convertir esos productos frecuentes en reglas de asociación.
```

Es importante notar que este límite debe ser predeterminado, es un hiper-parámetro del algoritmo, y puede ser ajustado por quien lo ejecuta. Sin embargo, esto no es un valor que pueda ser optimizado ya que no hay una métrica de evaluación para este algoritmo. Este hiper-parámetro suele especificarse en función de los datos, el uso particular, o la experiencia del analista.

La idea principal detrás de este algoritmo es el principio Apriori: cualquier subconjunto de un conjunto de elementos frecuentes debe ser frecuente. De forma paralela, ningún superconjunto de un conjunto de elementos infrecuentes puede ser frecuente.

Para illustar como funciona, veamos un ejercicio simple. Supongamos que tenemos los siguientes datos de un comercio que vende 4 productos (A,B,C,D), tenemos 7 transacciones y podemos ver que productos se compraron en la misma transaccion (en la misma canasta de compra).

Transacción	Productos
1	{A, B, C, D}
2	{A, B, D}
3	{A, B}
4	{B, C, D}
5	{B, C}
6	{C, D}
7	{B, D}

De la tabla de datos podemos ver que el producto A aparece en 3 de las 7 transacciones, el artículo B en 6 y ambos artículos juntos en 3. El soporte del producto {A} es por lo tanto del 43%, el del producto {B} del 86% y del conjunto {A, B} del 43%. De las 3 transacciones que incluyen A, las 3 incluyen B, por lo tanto, la regla "clientes que compran el artículo A también compran B", se cumple, acorde a los datos, un 100% de las veces. Esto significa que la confianza de la regla A $\rightarrow$ B es del 100%.

Encontrar conjuntos de artículos frecuentes (conjuntos de artículos con una frecuencia mayor o igual a un determinado soporte mínimo) no es un proceso trivial debido ya que los conjuntos

posibles crecen exponencialmente. Para illustarlo veamos todas las posibles combinaciones de estos 4 artículos:

Las flechas indican que dos o más conuntos puden ser combinados para formar un conjunto más grande. Recordemos que nuestro objetivo es encontrar los productos que se compran juntos frecuentemente; y que la frecuencia la mediamos con el indicador de soporte que contaba el porcentaje de transacciones que contienen este conjunto. Entonces, ¿cómo calculamos el soporte del conjunto {A,B}? En este caso tendríamos que examinar todas las transacciones y contar el total de veces que aparece este cojunto, para luego dividirlo por el total de transacciones.

Esto deberíamos además hacerlo para todos los conjuntos posibles. En la figura podemos ver que para cada conjunto posible debemos ir a través de los datos 15 veces. Es más, para un conjunto de datos que contiene N items posibles puede generar $2^N - 1$ conjuntos posibles. Por ejemplo, para una tienda que vende solo 100 artículos, uno puede generar 1.26×10^{30} posibles conjutos. Esto demandaría demasiados recursos computacionales y de tiempo.

Para reducir los costos computacionales podemos aplicar el principio Apriori: cualquier subconjunto de un conjunto de elementos frecuentes debe ser frecuente. Esto implica por ejemplo que si {A,B} es frecuente, también lo son {A} y {B}. Entonces, lo que hace el algoritmo Apriori es buscar exhaustivamente por niveles de complejidad (de menor a mayor tamaño de conjunto de productos).

El funcionamiento del algoritmo es relativamente sencillo, comienza identificando los productos individuales que aparecen en el total de transacciones con una frecuencia por encima de un límite minimo establecido por el analista.

Supongamos que el límite que establecemos es que el soporte tiene que ser igual o superior a 3/7 = 0.43, es decir que el item aparezca al menos en 3 de las 7 transacciones. A continuación lo que hacemos es calcular el soporte para los conjuntos de productos que contienen los items individuales:

Conjunto de Productos	Ocurrencias	Soporte
{A}	3	0.43
{B}	6	0.86
{C}	4	0.57
{D}	5	0.71

Dado que todos los items superan este umbral se mantienen en la base. Luego generamos los conjuntos que tienen todos las combinaciones de tamaño 2 y calculamos soporte:

Conjunto de Productos	Ocurrencias	Soporte
{A, B}	3	0.43
{A, C}	1	0.14
{A, D}	2	0.29
{B, C}	3	0.43
{B, D}	4	0.57
{C, D}	3	0.43

Los conjutos {A, B}, {B, C}, {B, D} y {C, D} superan el límite establecido, por lo que son frecuentes. Los restantes conjuntos se descartan, lo que implica además que cualquier conjunto

futuro que los contenga no van a ser frecuentes.

Conjunto de Productos	Ocurrencias	Soporte
{A, B}	3	0.43
{B, C}	3	0.43
{B, D}	4	0.57
{C, D}	3	0.43

Continuamos el proceso calculando para los conjuntos con 3 elementos que contienen A, B y C:

Conjunto de Productos
{A, B, C}
{A, B, D}
{B, C, D}
{C, D, A}

Los conuntos {A, B, C}, {A, B, D} y {C, D, A} contienen subconjuntos infrecuentes, por lo que los descartamos. Para el conjunto restante calculamos el soporte:

Conjunto de Productos	Ocurrencias	Soporte
{B, C, D}	2	0.29

El soporte del conjunot {B, C, D} no supera el umbral que establecimos y lo declaramos infrecuente. Al no existir ningún nuevo conjunto frecuente, se detiene el algoritmo.

Como resultado de la búsqueda se han identificado los siguientes itemsets frecuentes:

Conjuntos frecuentes
{A, B}
{B, C}
{B, D}
{C, D}

Identificados los conjuntos frequentes en el paso siguiente, calculamos las reglas de asociación. Nuevamete, este paso es demandante puesto que para cada conjunto frequente se generan tantas reglas como combinaciones binarias. Imponer un límite en la confianza reduce el conjunto de items a verificar. En concreto, el proceso a seguir es el siguiente:

- `Por cada conjunto frecuente I, obtenemos todos los posibles subconjuntos de I.`
- `Para cada subconjuntos s de I, creamos la regla "s => (I-s)"`
- `Descartamos todas las reglas que no superen un mínimo de confianza.`

Continuando con nuestro ejemplo, supongamos que solo queremos las reglas que tienen una confianza superior al 0.7.

Reglas	Confianza	Confianza
$A \rightarrow B$	$Pr(A,B)/Pr(A)$	$0.43 \ / \ 0.43 = 1$
$B \rightarrow A$	$Pr(A,B)/Pr(B)$	$0.43 \ / \ 0.86 = 0.5$
$B \rightarrow C$	$Pr(B,C)/Pr(B)$	$0.43 \ / \ 0.86 = 0.5$
$C \rightarrow B$	$Pr(B,C)/Pr(C)$	$0.43 \ / \ 0.57 = 0.75$
$B \rightarrow D$	$Pr(B,D)/Pr(B)$	$0.43 \ / \ 0.86 = 0.5$
$D \rightarrow B$	$Pr(B,D)/Pr(D)$	$0.43 \ / \ 0.71 = 0.6$
$C \rightarrow D$	$Pr(C,D)/Pr(C)$	$0.43 \ / \ 0.57 = 0.75$
$D \rightarrow C$	$Pr(C,D)/Pr(D)$	$0.43 \ / \ 0.71 = 0.6$

De todas las posibles reglas, únicamente $C \rightarrow D$ y $C \rightarrow B$ superan el límite de confianza.

1.5 Ejemplo implementación algoritmo Apriori

Para ilustrar la implementación en `Phyton` vamos a utilizar datos de `lastfm` que es un servicio de radio online. Estos datos contienen la lista de reproducción para 15.000 usuarios

Para esta tarea vamos a usar la libreria `apriori`.

[9]:
```
#Cargamos las librerías a utilizar
import pandas as pd
import numpy as np
from apyori import apriori
```

Cargamos y visualizamos la primeras observaciones de los datos

[10]:
```
lastfm = pd.read_csv('Data/lastfm_es.csv')
lastfm.head()
```

[10]:

	usuario	artista
0	1	red hot chili peppers
1	1	the black dahlia murder
2	1	goldfrapp
3	1	dropkick murphys
4	1	le tigre

La librería `apriori` requiere que los datos estén en un formato de listas de listas, donde todo el conjunto de datos es una lista grande y cada transacción es una lista interna dentro de la lista grande externa. Actualmente tenemos datos en forma `data.frame` de pandas. Así, con el siguiente bucle lo convertimos en el formato deseado:

[11]:
```
records = []
for i in lastfm['usuario'].unique():
    records.append(list(lastfm[lastfm['usuario'] == i]['artista'].values))
```

Por ejemplo el primer usuario escuchó a los siguientes artistas:

[12]:
```
records[0]
```

[12]:
```
['red hot chili peppers',
 'the black dahlia murder',
 'goldfrapp',
 'dropkick murphys',
 'le tigre',
 'schandmaul',
 'edguy',
 'jack johnson',
 'eluveitie',
 'the killers',
 'judas priest',
 'rob zombie',
 'john mayer',
 'the who',
 'guano apes',
 'the rolling stones']
```

El siguiente paso es aplicar el algoritmo Apriori en el conjunto de datos. Para hacerlo, podemos usar la clase `apriori` que importamos de la libreria `apyori`. La clase a priori requiere algunos valores de parámetros para funcionar:

1. El primer parámetro es la lista de lista de la que se desea extraer reglas.
2. El segundo parámetro es el parámetro `min_support`. Este parámetro se utiliza para seleccionar los elementos con valores de soporte superiores al valor especificado por el parámetro.
3. El parámetro `min_confidence` filtra aquellas reglas que tienen una confianza superior al umbral de confianza especificado por el parámetro.
4. De manera similar, el parámetro `min_lift` especifica el valor de elevación mínimo para las reglas preseleccionadas.
5. Finalmente, el parámetro `min_length` especifica la cantidad mínima de elementos que desea en sus reglas.

Supongamos que queremos reglas sólo para aquellos artistas que se escuchan al menos 1% de las veces. La confianza mínima para las reglas es 50% o 0.5. De manera similar, especificamos el valor de lift como 8 y finalmente `min_length` es 5 ya que queremos al menos cinco artistas escuchados en nuestras reglas. Recordemos que la elección de estos valores es un tanto arbitraria y por lo tanto los invito a experimentar con distintos valores

[13]:
```
association_rules = apriori(records, min_support=0.01, min_confidence=0.5,␣
↪min_lift=8,min_length=5)
association_results = list(association_rules)
```

Podemos ver cuantas reglas de asociación derivamos a partir de estos parámetros:

[14]:
```
print("Derivamos {} reglas de asociación.".format(len(association_results)))
```

Derivamos 4 reglas de asociación.

Cada elemento que obtuvimos contiene tres items. Primero muestra la regla de asociación, por ejemplo en este caso Judas Priest se escucha a menudo con Iron Maiden. El valor de soporte es 0.01, que nos dice que el 1% de los usuarios escucharon judas priest. El nivel de confianza es .5 que muestra que de los usuarios que escucharon Judas Priest, el 50% también escucho Iron Maiden. Finalmente, el lift de 8.56 nos dice que Iron Maiden es 8.56 veces más probables que sea escuchado por los clientes que escuchan Judas Priest, comparado a la probabilidad base de escuchar Iron Maiden.

[15]:
```
print(association_results[0])
```

```
RelationRecord(items=frozenset({'judas priest', 'iron maiden'}),
support=0.013533333333333333,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'judas priest'}),
items_add=frozenset({'iron maiden'}), confidence=0.5075,
lift=8.56299212598425)])
```

Los sistemas de recomendación pueden entonces recoger estas reglas de asociación y sugerir contenido basado en ellas. A los clientes que escucharon Judas Priest, se les puede entonces recomendar Iron Maiden. Finalmente podemos ver todas las reglas de asociación derivadas anteriormente:

[16]:
```
for item in association_results:

    # first index of the inner list
    # Contains base item and add item
    pair = item[0]
    items = [x for x in pair]
    print("Regla: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Soporte: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confianza: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("====================================")
```

```
Regla: judas priest -> iron maiden
Soporte: 0.013533333333333333
Confianza: 0.5075
Lift: 8.56299212598425
====================================
Regla: t.i. -> kanye west
Soporte: 0.0104
Confianza: 0.5672727272727273
Lift: 8.854413016743923
```

```
====================================
Regla: nightwish -> sonata arctica
Soporte: 0.013466666666666667
Confianza: 0.51010101010101
Lift: 8.236291874612649
====================================
Regla: the pussycat dolls -> rihanna
Soporte: 0.0104
Confianza: 0.5777777777777778
Lift: 13.415892672858618
====================================
```

2 Referencias

- Amat Rodrigo, Joaquín. (2018). Reglas de asociación y algoritmo Apriori con R, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/43_reglas_de_asociacion. Accedido el 12 de Enero de 2022
- Harrington, Peter (2012). Machine learning in action. Simon and Schuster.
- Jones, Aaron; Kruger, Christopher; Johnston, Benjamin. The Unsupervised Learning Workshop: Get started with unsupervised learning algorithms and simplify your unorganized data to help make future predictions. Packt Publishing. Kindle Edition.
- Taddy, Matt; Taddy, Matt. Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions. McGraw-Hill Education. Kindle Edition.