Luật kết hợp

Phân tích luật kết hợp (Association Rule Analysis) là một kỹ thuật để khai phá cách các mục (item) được liên kết với nhau. Có ba cách phổ biến để đo lường sự liên kết.

1. Độ hỗ trợ (support)
2. Độ tin cậy (confident)
3. Mức tăng (lift)

Dưới đây là giải thích đơn giản về các độ đo trên:

1. **Độ hỗ trợ**: Độ đo này được tính toán để kiểm tra mức độ phổ biến của một mặt hàng nhất định. Nó được đo lường bằng tỷ lệ các giao dịch trong đó một tập hợp mục xuất hiện. Ví dụ, có 100 người mua mặt hàng nào đó từ cửa hàng tạp hóa ngày hôm nay, ví dụ, cứ 100 người thì có 10 người mua chuối. Do đó, độ hỗ trợ của những người đã mua chuối sẽ là $(10/100 = 0.1 = 10\%)$.
2. **Độ tin cậy**: được tính toán để kiểm tra xem mặt hàng X có được mua hay không khi mặt hàng Y được mua. Điều này được đo lường bằng tỷ lệ giao dịch với mặt hàng X, trong đó mặt hàng Y cũng xuất hiện. Giả sử, có 10 người mua táo (trong số 100 người), và từ 10 người đó, 6 người cũng mua sữa chua. Do đó, độ tin cậy của $(táo => sữachua)$ là: $(táo => sữachua)/táo$, tức là $(6/10 = 0.6)$.
3. **Mức tăng**: được tính để đo lường khả năng mặt hàng Y được mua khi mặt hàng X được mua, đồng thời kiểm soát mức độ phổ biến của mặt hàng Y. Công thức cho độ tăng là:
$$lift = support(X -> Y)/(support(X) * support(Y))$$
.

Để thu được các luật kết hợp, ta thường áp dụng 2 tiêu chí: *minimum support (min_sup)* và *minimum confidence (min_conf)*

Các luật thỏa mãn có support và confidence thỏa mãn (lớn hơn hoặc bằng) cả Minimum support và Minimum confidence gọi là các luật mạnh (Strong Rule)

Minimum support và Minimum confidence gọi là các giá trị ngưỡng (threshold) và phải xác định trước khi sinh các luật kết hợp.

Apriori

Thuật toán Apriori được công bố bởi R. Agrawal và R. Srikant vào năm 1994 vì để tìm các tập phổ biến trong một bộ dữ liệu lớn. Tên của thuật toán là Apriori vì nó sử dụng kiến thức đã có từ trước (prior) về các thuộc tính, vật phẩm thường xuyên xuất hiện trong cơ sở dữ liệu. Để cải thiện hiệu quả của việc lọc các mục thường xuyên (frequent itemsets) theo cấp độ, một thuộc tính quan trọng được sử dụng gọi là thuộc tính Apriori giúp giảm phạm vi tìm kiếm của thuật toán.

Tư tưởng chính của thuật toán Apriori là:

- Tìm tất cả frequent itemsets:
  - k-itemset (itemsets gồm k items) được dùng để tìm (k+1)- itemset.

    - Đầu tiên tìm 1-itemset (ký hiệu L1). L1 được dùng để tìm L2 (2-itemsets). L2 được dùng để tìm L3 (3-itemset) và tiếp tục cho đến khi không có k-itemset được tìm thấy.
- Từ frequent itemsets sinh ra các luật kết hợp mạnh (các luật kết hợp thỏa mãn 2 tham số min_sup và min_conf)

Nội dung thực hành:

Một siêu thị thực phẩm có lưu lại thông tin hành vi mua sắm của khách hàng tại siêu thị của họ. Bài toán đặt ra là tìm ra được những sản phẩm thường xuyên được mua cùng nhau bởi các khách hàng, từ đó, họ có thể có một vài chiến lược kích cầu mua sắm, ví dụ, sắp xếp lại gian hàng, tặng kèm sản phẩm ưa thích, ...

- *Dữ liệu sử dụng:* `data/groceries.csv`
- *Mô tả:*
    - Item(s): Số lượng mặt hàng đã mua
    - Item 1: Tên sản phẩm thứ nhất
    - Item 2: Tên sản phẩm thứ hai (nếu có)
    - Item n: Tên sản phẩm thứ n (nếu có)

Tương tự như trong các bài thực hành trước, chúng ta vẫn đi qua các bước cơ bản, đầu tiên, import các thư viện cần thiết và tải dữ liệu.

Entrée [1]:
```
import numpy as np
import pandas as pd

from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
```

Entrée [2]:
```
data = pd.read_csv('data/groceries.csv')
print('Kích thước dữ liệu: ', data.shape)
data.head()
```

```
Kích thước dữ liệu:  (200, 33)
```

Out[2]:

	Item(s)	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	...	Item 23	Item 24	It
0	4	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
1	3	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
2	1	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
3	4	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
4	4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N

5 rows × 33 columns

Trong bài thực hành cơ bản này, chúng ta tạm thời bỏ qua các bước EDA (Phân tích dữ liệu) và đi thẳng vào việc triển khai thuật toán Apriori.

Thuật toán Apriori lấy danh sách các mặt hàng đã được mua cùng nhau làm đầu vào. Do đó, chúng ta cần lấy từng hàng dưới dạng danh sách (ngoại trừ cột đầu tiên và 'NAN' trong các cột).

Entrée [3]:
```
transactions = []

# Thêm tất cả các mục từ mỗi hàng trong một danh sách (Bỏ qua các cột đầu tiên, r
for i in range(0, 200):
    transactions.append([str(data.values[i,u]) for u in range(1, 33)])
```

Entrée [4]:
```
print(transactions[0])
```

```
['citrus fruit', 'semi-finished bread', 'margarine', 'ready soups', 'nan', 'na
n', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'na
n', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'na
n', 'nan', 'nan', 'nan', 'nan']
```

Entrée [5]:
```
te = TransactionEncoder()
te_data = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_data,columns=te.columns_)
del df['nan']
df.head()
```

Out[5]:

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baking powder	bathroom cleaner	beef	berries	beverages	bottled beer	...
0	False	False	False	False	False	False	False	False	False	False	...
1	False	False	False	False	False	False	False	False	False	False	...
2	False	False	False	False	False	False	False	False	False	False	...
3	False	False	False	False	False	False	False	False	False	False	...
4	False	False	False	False	False	False	False	False	False	False	...

5 rows × 118 columns

Triển khai thuật toán với độ hỗ trợ 0.01. Sau đó sắp xếp để thấy được top sản phẩm có độ hỗ trợ cao

Entrée [6]:
```
df1 = apriori(df,min_support=0.01,use_colnames=True)
df1
```

Out[6]:

	support	itemsets
0	0.020	(UHT-milk)
1	0.010	(abrasive cleaner)
2	0.010	(baking powder)
3	0.015	(bathroom cleaner)
4	0.040	(beef)
...	...	...
1396	0.010	(frankfurter, fruit/vegetable juice, pip fruit...
1397	0.010	(sugar, tropical fruit, whole milk, waffles, d...
1398	0.010	(frankfurter, pip fruit, sugar, soft cheese, w...
1399	0.010	(cereals, sugar, tropical fruit, whole milk, w...
1400	0.010	(frankfurter, fruit/vegetable juice, pip fruit...

1401 rows × 2 columns

Entrée [7]:
```
df1.sort_values(by="support",ascending=False)
```

Out[7]:

	support	itemsets
89	0.265	(whole milk)
68	0.200	(rolls/buns)
56	0.185	(other vegetables)
76	0.145	(soda)
90	0.110	(yogurt)
...	...	...
619	0.010	(cream cheese, tropical fruit, domestic eggs)
620	0.010	(whole milk, cream cheese, domestic eggs)
621	0.010	(cream cheese, frankfurter, fruit/vegetable ju...
622	0.010	(pip fruit, cream cheese, frankfurter)
1400	0.010	(frankfurter, fruit/vegetable juice, pip fruit...

1401 rows × 2 columns

Entrée [8]:
```
df1['length'] = df1['itemsets'].apply(lambda x:len(x))
df1
```

Out[8]:

	support	itemsets	length
0	0.020	(UHT-milk)	1
1	0.010	(abrasive cleaner)	1
2	0.010	(baking powder)	1
3	0.015	(bathroom cleaner)	1
4	0.040	(beef)	1
...	...	...	...
1396	0.010	(frankfurter, fruit/vegetable juice, pip fruit...	7
1397	0.010	(sugar, tropical fruit, whole milk, waffles, d...	7
1398	0.010	(frankfurter, pip fruit, sugar, soft cheese, w...	7
1399	0.010	(cereals, sugar, tropical fruit, whole milk, w...	8
1400	0.010	(frankfurter, fruit/vegetable juice, pip fruit...	8

1401 rows × 3 columns

Entrée [9]:
```
df1[(df1['length']==2) & (df1['support']>=0.05)]
```

Out[9]:

	support	itemsets	length
402	0.070	(other vegetables, whole milk)	2
433	0.060	(whole milk, rolls/buns)	2
481	0.055	(whole milk, yogurt)	2

Entrée [10]:
```
df1[(df1['length']==3) & (df1['support']>=0.02)]
```

Out[10]:

	support	itemsets	length
504	0.02	(whole milk, fruit/vegetable juice, bottled beer)	3
723	0.02	(other vegetables, frozen vegetables, whole milk)	3
724	0.02	(whole milk, frozen vegetables, rolls/buns)	3
759	0.02	(other vegetables, ham, whole milk)	3
786	0.02	(root vegetables, other vegetables, whole milk)	3
792	0.02	(other vegetables, sugar, whole milk)	3
846	0.02	(whole milk, tropical fruit, yogurt)	3

Entrée [11]:
```
rules = association_rules(df1, metric = "confidence", min_threshold = 0)
rules
```

Out[11]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
0	(citrus fruit)	(UHT-milk)	0.080	0.020	0.01	0.125000	6.250000	0.0084
1	(UHT-milk)	(citrus fruit)	0.020	0.080	0.01	0.500000	6.250000	0.0084
2	(other vegetables)	(UHT-milk)	0.185	0.020	0.01	0.054054	2.702703	0.0063
3	(UHT-milk)	(other vegetables)	0.020	0.185	0.01	0.500000	2.702703	0.0063
4	(rolls/buns)	(UHT-milk)	0.200	0.020	0.01	0.050000	2.500000	0.0060
...	...	...	...	...	...	...	...	
19857	(sugar)	(frankfurter, fruit/vegetable juice, pip fruit...	0.060	0.010	0.01	0.166667	16.666667	0.0094
19858	(tropical fruit)	(frankfurter, fruit/vegetable juice, pip fruit...	0.105	0.010	0.01	0.095238	9.523810	0.0089
19859	(whole milk)	(frankfurter, fruit/vegetable juice, pip fruit...	0.265	0.010	0.01	0.037736	3.773585	0.0073
19860	(cream cheese)	(frankfurter, fruit/vegetable juice, pip fruit...	0.050	0.010	0.01	0.200000	20.000000	0.0095
19861	(domestic eggs)	(frankfurter, fruit/vegetable juice, pip fruit...	0.035	0.010	0.01	0.285714	28.571429	0.0096

19862 rows × 9 columns

Whole Milk có độ hỗ trợ cao nhất, do đó, chúng ta lựa chọn whole milk để phân tích chi tiết hơn

Entrée [12]:
```
rules_sel = rules[rules["antecedents"].apply(lambda x: "whole milk" in x)]
rules_sel.sort_values('confidence', ascending=False)
```

Out[12]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
12261	(whole milk, ham, fruit/vegetable juice, bottl...	(other vegetables, tropical fruit)	0.010	0.040	0.01	1.000000	25.000000	0.(
19161	(whole milk, waffles, domestic eggs, yogurt)	(root vegetables, sugar, tropical fruit)	0.010	0.010	0.01	1.000000	100.000000	0.(
19011	(whole milk, frankfurter, tropical fruit, frui...	(pip fruit, sugar, domestic eggs)	0.010	0.010	0.01	1.000000	100.000000	0.(
19162	(root vegetables, whole milk, waffles, yogurt)	(sugar, tropical fruit, domestic eggs)	0.010	0.015	0.01	1.000000	66.666667	0.(
11226	(whole milk, frankfurter, sugar, soft cheese)	(ham)	0.010	0.030	0.01	1.000000	33.333333	0.(
...	...	...	...	...	...	...	...	
9303	(whole milk)	(cream cheese, sugar, fruit/vegetable juice, d...	0.265	0.010	0.01	0.037736	3.773585	0.(
9333	(whole milk)	(cream cheese, tropical fruit, fruit/vegetable...	0.265	0.010	0.01	0.037736	3.773585	0.(
9393	(whole milk)	(pip fruit, cream cheese, sugar, domestic eggs)	0.265	0.010	0.01	0.037736	3.773585	0.(
9423	(whole milk)	(pip fruit, cream cheese, tropical fruit, dome...	0.265	0.010	0.01	0.037736	3.773585	0.(
19859	(whole milk)	(frankfurter, fruit/vegetable juice, pip fruit...	0.265	0.010	0.01	0.037736	3.773585	0.(

6352 rows × 9 columns

Entrée [13]:
```
# Lấy ra 5 sản phẩm quan trọng nhất sau khi khách hàng mua whole milk
rules_support = rules_sel['support'] >= rules_sel['support'].quantile(q = 0.95)
rules_confi = rules_sel['confidence'] >= rules_sel['confidence'].quantile(q = 0.9
rules_lift = rules_sel['lift'] > 1
rules_best = rules_sel[rules_support & rules_confi & rules_lift]
rules_best.head()
```

Out[13]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
793	(whole milk, bathroom cleaner)	(rolls/buns)	0.01	0.200	0.01	1.0	5.000000	0.00800
1032	(whole milk, white bread)	(bottled beer)	0.01	0.065	0.01	1.0	15.384615	0.00935
1326	(candy, whole milk)	(soda)	0.01	0.145	0.01	1.0	6.896552	0.00855
1338	(candy, whole milk)	(waffles)	0.01	0.040	0.01	1.0	25.000000	0.00960
1572	(whole milk, coffee)	(yogurt)	0.01	0.110	0.01	1.0	9.090909	0.00890

Entrée [ ]: