BỘ GIÁO DỤC ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SÀI GÒN

Khoa Công nghệ Thông tin

PHÂN TÍCH DỮ LIỆU - BUỔI 06,7
Phân tích Luật kết hợp – Thực hành

CBGD: Phan Thành Huấn

✉ : pthuan112358@gmail.com

: 097 882 8842

Nội dung

1. Một số thư viện hỗ trợ phân tích luật kết hợp
2. Khai thác tập phổ biến
3. Khai thác luật kết hợp

1-Một số thư viện hỗ trợ phân tích luật kết hợp

Thư viện Python phổ biến được sử dụng để phân tích tập phổ biến và luật kết hợp:

1. mlxtend: Cung cấp một loạt các chức năng cho phân tích tập phổ biến và luật kết hợp; gồm thuật toán *Apriori* và *Eclat* cho phân tích tập phổ biến, cũng như thuật toán Association Rules cho LKH;
2. pyfpgrowth: Cung cấp thuật toán *FP-Growth* cho phân tích tập phổ biến và luật kết hợp - xây dựng trên cơ sở cấu trúc dữ liệu *FP-Tree*;

1-Một số thư viện hỗ trợ phân tích luật kết hợp

3. Orange: Cung cấp một loạt các công cụ và thuật toán cho phân tích tập phổ biến và luật kết hợp, bao gồm *Apriori* và *FP-Growth*;
4. pymining: Thư viện nhỏ gọn cho khai thác dữ liệu và LKH; cung cấp các hàm khai thác tập phổ biến và LKH. Hàm **frequent_itemsets()** để khai thác tập phổ biến và **rules()** cho khai thác LKH.

1-Một số thư viện hỗ trợ phân tích luật kết hợp

Một số hàm phổ biến trong thư viện mlxtend:

1. **apriori()**: Hàm này được sử dụng để áp dụng thuật toán *Apriori* để tìm tập phổ biến từ một *ma trận nhị phân*. Trả về một DataFrame chứa các tập phổ biến và độ phổ biến tương ứng;
2. **association_rules()**: Hàm này được sử dụng để áp dụng thuật toán *Association Rules* từ tập phổ biến đã tìm được. Trả về một DataFrame chứa các luật kết hợp và các thông số như độ tin cậy, độ phổ biến, lift, và leverage;

1-Một số thư viện hỗ trợ phân tích luật kết hợp

3. **TransactionEncoder()**: sử dụng để chuyển đổi dữ liệu thành *ma trận nhị phân*. Hàm sử dụng phương pháp *one-hot encoding* để chuyển đổi các mục thành các cột nhị phân;
4. **one_hot()**: chuyển đổi danh sách các thuộc tính thành một ma trận nhị phân. Trả về một *pandas DataFrame* có các cột nhị phân tương ứng với các thuộc tính trong danh sách;

1-Một số thư viện hỗ trợ phân tích luật kết hợp

5. **fpgrowth()**: sử dụng để áp dụng thuật toán FP-Growth để tìm tập phổ biến từ một *ma trận nhị phân*. Trả về một *DataFrame* chứa các tập phổ biến và độ phổ biến tương ứng;
6. **generate_association_rules()**: sử dụng để áp dụng thuật toán Association Rules từ tập phổ biến đã tìm được bằng FP-Growth. Nó trả về một *DataFrame* chứa các luật kết hợp và các thông số như độ tin cậy, độ phổ biến, lift, và leverage.

1-Một số thư viện hỗ trợ phân tích luật kết hợp

- Độ đo Lift là độ đo quan trọng trong phân tích luật kết hợp (association rules) và được sử dụng để đánh giá mức độ tương quan giữa các item trong một luật kết hợp.

$$lift(X \rightarrow Y) = \frac{\sup(X \cup Y)}{\sup(X)\sup(Y)}$$

Giá trị Lift > 1 cho thấy mức độ tương quan tích cực giữa X và Y, trong khi giá trị Lift < 1 cho thấy mức độ tương quan tiêu cực. Nếu giá trị Lift = 1, thì không có mức độ tương quan giữa X và Y.

1-Một số thư viện hỗ trợ phân tích luật kết hợp

- Độ đo Leverage là độ đo trong phân tích LKH (association rules) được sử dụng để đánh giá mức độ tương quan giữa các item trong một LKH. Đo lường mức độ tương quan tuyến tính giữa việc xuất hiện của các item trong LKH so với việc chúng xuất hiện độc lập.

$$leverage(X \rightarrow Y) = \sup(X \cup Y) - \sup(X)\sup(Y)$$

Giá trị Leverage thuộc [-1, 1]. Giá trị Leverage = 0 không có tương quan tuyến tính giữa X và Y. Giá trị Leverage > 0 mức độ tương quan **dương**, trong khi Leverage < 0 mức độ tương quan **âm**.

1-Một số thư viện hỗ trợ phân tích luật kết hợp

Ví dụ: Sử dụng hàm apriori khai thác tập phổ biến thuộc mlxtend

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Dữ liệu mẫu
dataset = [['Bread', 'Milk', 'Eggs'],
           ['Bread', 'Diapers', 'Beer', 'Eggs'],
           ['Milk', 'Diapers', 'Beer', 'Coke'],
           ['Bread', 'Milk', 'Diapers', 'Beer'],
           ['Bread', 'Milk', 'Diapers', 'Coke']]

# Chuyển đổi dữ liệu thành ma trận nhị phân
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Áp dụng thuật toán Apriori để tìm tập phổ biến
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)

# In ra tập phổ biến
print(frequent_itemsets)
```

	support	itemsets
0	0.6	(Beer)
1	0.8	(Bread)
2	0.8	(Diapers)
3	0.8	(Milk)
4	0.6	(Diapers, Beer)
5	0.6	(Diapers, Bread)
6	0.6	(Milk, Bread)
7	0.6	(Diapers, Milk)