Association Rules - 연관규칙

```
import warnings
warnings.filterwarnings('ignore')
```

I. Read Data_Set and Preprocessing

1) Read 'order.tsv'

- DF.info( )

```
import pandas as pd

url = 'https://raw.githubusercontent.com/rusita-ai/pyData/master/orders.tsv'
DF = pd.read_table(url)

DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   order_id            4622 non-null   int64 
 1   quantity            4622 non-null   int64 
 2   item_name           4622 non-null   object
 3   choice_description  3376 non-null   object
 4   item_price          4622 non-null   object
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

- 'order_id' and 'item_name'

```
DF.head(3)
```

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39

2) 데이터 정보 확인

- DF_1 지정 후 'order_id' 및 'item_name' 종류 확인
- 한 개의 'order_id'가 여러 개의 'item_name'으로 분리되어 지정

```
DF_1 = DF[['order_id', 'item_name']]

DF_1.order_id.unique().shape, DF_1.item_name.unique().shape
```

```
((1834,), (50,))
```

- 'order_id' 1834개

```
order_ID = list(DF_1.order_id.unique())

order_ID[:10], order_ID[-10:]
```

```
([1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
 [1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834])
```

- 'item_name' 50종류

```
item_NAME = list(DF_1.item_name.unique())

item_NAME[:5], item_NAME[-5:]
```

```
(['Chips and Fresh Tomato Salsa',
  'Izze',
  'Nantucket Nectar',
  'Chips and Tomatillo-Green Chili Salsa',
  'Chicken Bowl'],
 ['Barbacoa Salad Bowl',
  'Salad',
  'Veggie Crispy Tacos',
  'Veggie Salad',
  'Carnitas Salad'])
```

3) Preprocessing

- 1835 길이의 2차원 리스트 생성

```
orderItems = [[] for i in range(1835)]

len(orderItems)
```

```
1835
```

- 'order_id' 별로 'item_name' 묶기

```
num = 0

for i in DF_1.item_name :
    orderItems[DF_1.order_id[num]].append(i)
    num = num + 1
```

```
orderItems[:5], orderItems[-5:]
```

```
([[],
  ['Chips and Fresh Tomato Salsa',
   'Izze',
   'Nantucket Nectar',
   'Chips and Tomatillo-Green Chili Salsa'],
  ['Chicken Bowl'],
  ['Chicken Bowl', 'Side of Chips'],
  ['Steak Burrito', 'Steak Soft Tacos']],
 [['Steak Burrito', 'Veggie Burrito'],
  ['Carnitas Bowl', 'Chips', 'Bottled Water'],
  ['Chicken Soft Tacos', 'Chips and Guacamole'],
  ['Steak Burrito', 'Steak Burrito'],
  ['Chicken Salad Bowl', 'Chicken Salad Bowl', 'Chicken Salad Bowl']])
```

- 첫 번째 빈 리스트 제거 및 중복 아이템 단일화

```
orderItems.pop(0)

num = 0
for i in orderItems :
    orderItems[num] = list(set(orderItems[num]))
    num = num + 1
```

```
orderItems[:5], orderItems[-5:]
```

```
([['Nantucket Nectar',
   'Izze',
   'Chips and Fresh Tomato Salsa',
   'Chips and Tomatillo-Green Chili Salsa'],
  ['Chicken Bowl'],
  ['Chicken Bowl', 'Side of Chips'],
  ['Steak Soft Tacos', 'Steak Burrito'],
  ['Chips and Guacamole', 'Steak Burrito']],
 [['Veggie Burrito', 'Steak Burrito'],
  ['Chips', 'Carnitas Bowl', 'Bottled Water'],
  ['Chips and Guacamole', 'Chicken Soft Tacos'],
  ['Steak Burrito'],
  ['Chicken Salad Bowl']])
```

II. TransactionEncoder( )

- Transaction 구조 변환

```
from mlxtend.preprocessing import TransactionEncoder

TSE = TransactionEncoder()
Transac_Array = TSE.fit_transform(orderItems)
```

- pandas DataFrame 구조 변환

```
order_DF = pd.DataFrame(Transac_Array, columns = TSE.columns_)

order_DF.head()
```

	6 Pack Soft Drink	Barbacoa Bowl	Barbacoa Burrito	Barbacoa Crispy Tacos	Barbacoa Salad Bowl	Barbacoa Soft Tacos	Bottled Water	Bowl	Burrito
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False

III. apropri( )

- 지지도(support) 0.05 이상인 주문 추출
- use_colnames : item_name으로 출력
- max_len : 주문의 최대 길이 지정

```
from mlxtend.frequent_patterns import apriori

frequent_itemsets = apriori(order_DF,
                            min_support = 0.05,
                            use_colnames = True,
                            max_len = None)

frequent_itemsets
```

	support	itemsets
0	0.083969	(Bottled Water)
1	0.051254	(Canned Soda)
2	0.150491	(Canned Soft Drink)
3	0.335333	(Chicken Bowl)
4	0.266630	(Chicken Burrito)
5	0.053435	(Chicken Salad Bowl)
6	0.058342	(Chicken Soft Tacos)
7	0.113413	(Chips)
8	0.059978	(Chips and Fresh Tomato Salsa)
9	0.258451	(Chips and Guacamole)
10	0.055071	(Side of Chips)
11	0.102508	(Steak Bowl)
12	0.186478	(Steak Burrito)
13	0.060523	(Canned Soft Drink, Chicken Bowl)
14	0.066521	(Chips, Chicken Bowl)
15	0.081243	(Chips and Guacamole, Chicken Bowl)

IV. association_rules( )

1) 지지도(support)가 최소 0.05 이상인 연관관계 출력

- antecedents(조건절) -> consequents(결과절)
- 전체 주문 중 조건절과 결과절을 포함한 비율
- 방향성 없음

```
from mlxtend.frequent_patterns import association_rules

association_rules(frequent_itemsets,
                  metric = 'support',
                  min_threshold = 0.05)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(Canned Soft Drink)	(Chicken Bowl)	0.150491	0.335333	0.060523	0.402174	1.199328
1	(Chicken Bowl)	(Canned Soft Drink)	0.335333	0.150491	0.060523	0.180488	1.199328

2) 신뢰도(confidence)가 최소 0.3 이상인 연관관계 출력

- 조건절이 있을때 결과절도 있는 비율
- 조건부확률
- 방향성 존재

```
association_rules(frequent_itemsets,
                  metric = 'confidence',
                  min_threshold = 0.3)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(Canned Soft Drink)	(Chicken Bowl)	0.150491	0.335333	0.060523	0.402174	1.199328
1	(Chips)	(Chicken	0.113413	0.335333	0.066521	0.586538	1.749124

1) 향상도(support)가 최소 0.1 이상인 연관관계 출력

- 향상도가 1이라면 조건절과 결과절은 독립관계
- 1보다 크거나 작다면 우연이 아닌 필연적 관계

```
association_rules(frequent_itemsets,
                  metric = 'lift',
                  min_threshold = 0.1)
```

```
antecedent  consequent
```

#

#

#

The End

#

#

#