Published in Geek Culture

George Pipis Follow

May 25, 2021 · 4 min read ★ · Listen

Market Basket Analysis and Association Rules from Scratch

How to get the Association Rules in Python

We have provided a tutorial of Market Basket Analysis in Python working with the `mlxtend` library. Today, we will provide an example of how you can get the association rules from scratch. Let's recall the 3 most common association rules:

Association Rules

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction. For example, we can extract information on purchasing behavior like " *If someone buys beer and sausage, then is likely to buy mustard with high probability* "

Let's define the main Associaton Rules:

Support

It calculates how often the product is purchased and is given by the formula:

$$Support(X) = \frac{Frequency(X)}{N(\#of\ Transactions)}$$

$$Support(X \rightarrow Y) = \frac{Frequency(X \bigcap Y)}{N(\#of\ Transactions)}$$

Confidence

It measures how often items in Y appear in transactions that contain X and is given by the formula.

$$Confidence(X \rightarrow Y) = \frac{Support(X \rightarrow Y)}{}$$

greater than one indicate that the items are likely to be purchased together. It tells us how much better a rule is at predicting the result than just assuming the result in the first place. When lift > 1 then the rule is better at predicting the result than guessing. When lift < 1, the rule is doing worse than informed guessing. It can be given by the formula:

$$Lift(X \rightarrow Y) = \frac{Support(X \rightarrow Y)}{Support(X) \times Support(Y)}$$

Coding Part

By 2 Products

Assume that we are dealing with the following `groceries.xlsx` file:

3	1	bread,milk,biscuit,cereal
4	2	bread,tea
5	3	jam,bread,milk
6	4	tea,biscuit
7	5	bread,tea
8	6	tea,cereal
9	7	bread,tea,biscuit
10	8	jam,bread,tea
11	9	bread,milk
12	10	coffee,orange,biscuit,cereal
13	11	coffee,orange,biscuit,cereal
14	12	coffee,sugar
15	13	bread,coffee,orange
16	14	bread,sugar,biscuit
17	15	coffee,sugar,cereal
18	16	bread,sugar,biscuit
19	17	bread,coffee,sugar
20	18	bread,coffee,sugar
21	19	tea,milk,coffee,cereal

We want to transform the data into order id and product id.

```
import pandas as pd

df = pd.read_excel("groceries.xlsx")
df['items'] = df['items'].apply(lambda x: x.split(","))

df = df.explode('items')
df.columns = ['oid', 'pid']
df.reset_index(drop=True, inplace=True)

df
```

	oid	pid
0	0	milk
1	0	bread
2	0	biscuit
3	1	bread
4	1	milk
5	1	biscuit
6	1	cereal
7	2	bread

Write the function which returns the three association rules such as **support**, **confidence**, and **lift** for every possible pair. The `my_pid` is the **`antecedent`** and the `y` is the **`consequent`**.

```
def all_x_y(df, my_pid, y):
    df = df.copy()
    N = len(df.oid.unique())

    tmp = pd.DataFrame({'XY':[my_pid,y]})
    tmp = df.merge(tmp, how='inner', left_on='pid', right_on='XY' )

    numerator = sum(tmp.groupby('oid').size()==2)/N
    a = len(df.loc[df.pid==my_pid].oid.unique())/N
    b = len(df.loc[df.pid==y].oid.unique())/N
    denominator = a * b


    lift = numerator/denominator
    confidence = numerator/a
    support = numerator

    return (support, confidence, lift)
```

Let's see some examples by considering the **(milk, bread)** and **(orange, coffee)**:

```
all_x_y(df, 'orange', 'coffee')
```

```
(0.15, 1.0, 2.5)
```

You can confirm that we get the same results with that from the `mlxtend` module:

```
from mlxtend.frequent_patterns import association_rules, apriori

# compute frequent items using the Apriori
algorithmfrequent_itemsets = apriori(onehot, min_support = 0.01,
max_len = 2, use_colnames=True)

# compute all association rules for frequent_itemsets
rules = association_rules(frequent_itemsets, min_threshold=0.01)

rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(bread)	(biscuit)	0.65	0.40	0.25	0.384615	0.961538	-0.0100	0.975000
1	(biscuit)	(bread)	0.40	0.65	0.25	0.625000	0.961538	-0.0100	0.933333
2	(biscuit)	(cereal)	0.40	0.30	0.15	0.375000	1.250000	0.0300	1.120000
3	(cereal)	(biscuit)	0.30	0.40	0.15	0.500000	1.250000	0.0300	1.200000
4	(coffee)	(biscuit)	0.40	0.40	0.10	0.250000	0.625000	-0.0600	0.800000
5	(biscuit)	(coffee)	0.40	0.40	0.10	0.250000	0.625000	-0.0600	0.800000
6	(milk)	(biscuit)	0.25	0.40	0.10	0.400000	1.000000	0.0000	1.000000
7	(biscuit)	(milk)	0.40	0.25	0.10	0.250000	1.000000	0.0000	1.000000
8	(orange)	(biscuit)	0.15	0.40	0.10	0.666667	1.666667	0.0400	1.800000
9	(biscuit)	(orange)	0.40	0.15	0.10	0.250000	1.666667	0.0400	1.133333
10	(sugar)	(biscuit)	0.30	0.40	0.10	0.333333	0.833333	-0.0200	0.900000

Now, let's see how we can get all the possible pairs.

```
unique_products = df.pid.unique()
```


```
                output.append({
                    'antecedents':i,
                    'consequents':j,
                    'support':tmp[0],
                    'confidence':tmp[1],
                    'lift':tmp[2]
                              })

output = pd.DataFrame(output)
output
```

	antecedents	consequents	support	confidence	lift
0	milk	bread	0.20	0.800000	1.230769
1	milk	biscuit	0.10	0.400000	1.000000
2	milk	cereal	0.10	0.400000	1.333333
3	milk	tea	0.05	0.200000	0.571429
4	milk	jam	0.05	0.200000	2.000000
...	...	...	...	...	...
67	sugar	cereal	0.05	0.166667	0.555556
68	sugar	tea	0.00	0.000000	0.000000
69	sugar	jam	0.00	0.000000	0.000000
70	sugar	coffee	0.20	0.666667	1.666667
71	sugar	orange	0.00	0.000000	0.000000

72 rows × 5 columns

By 3 Products

The Market Basket Analysis and the Association rules are becoming more complicated when we examine more combinations. Let's say that we want to get all the association rules when the **antecedents are 2** and the **consequent is 1**. I.e we have already two items in the basket, what are the association rules of the extra item. The first that we will need to do is to generate all the possible combinations by 3 (or even by 2, and then to add the right-hand side). For example:


```
[('milk', 'bread', 'biscuit'),
 ('milk', 'bread', 'cereal'),
 ('milk', 'bread', 'tea'),
 ('milk', 'bread', 'jam'),
 ('milk', 'bread', 'coffee'),
 ('milk', 'bread', 'orange'),
 ('milk', 'bread', 'sugar'),
 ('milk', 'biscuit', 'cereal'),
 ('milk', 'biscuit', 'tea'),
 ('milk', 'biscuit', 'jam'),
 ('milk', 'biscuit', 'coffee'),
 ('milk', 'biscuit', 'orange'),
 ('milk', 'cereal', 'jam'),
 ('milk', 'cereal', 'coffee'),
 ('milk', 'cereal', 'orange'),
 ('milk', 'cereal', 'sugar'),
 ('milk', 'tea', 'jam'),
 ('milk', 'tea', 'coffee'),
```


In another tutorial, we will show you how you can generate the association rules for