2020

Lab 8: Association Rules

Assoc. Prof. Dr. Mohammed Al-Sarem
Taibah University, Information System Department
3/19/2020

Lab 8: Association Rules

Lab Objectives:

The goal of this lab is to explain the role of association rules in mining hidden relationships in data. After completion of this lab, you will be able to:

- Generate association rules using Apriori algorithm
- Evaluate the goodness of the findings rules

Methodology

First you have to install the required library that allow you to run Apriori algorithm. Then, as usual, you have to load and read dataset. After executing the algorithm, a set of generated rules are found. Your task is to analyze these rules and reports only the most interesting.

In class task:

At the end of this lab, the student will be able to:

- Explain how the Apriori algorithm works.
- Extract rules from the generated set.
- Evaluate the findings using Support and Confidence.

home task:

Complete your **Course Project** (See Home task in lab 5).

References:

- Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." Proc. 20th int. conf. very large data bases, VLDB. Vol. 1215. 1994.
- https://en.wikipedia.org/wiki/Apriori_algorithm

Lab 8: Association Rules

1. Frequent Itemsets via Apriori Algorithm

Apriori function to extract frequent itemsets for association rule mining. To work with `apriori` algorithm, first you have to install, in this time, `mlxtend` library as follows:

- add a new jupyter cell
- write down the following code: `pip install mlxtend`

Now, you can find the required algorithm by import it using the following:

```
from mlxtend.frequent_patterns import apriori
```

Generating Frequent Itemsets

Suppose we have the following transaction data as depicted in the Table below.

Table 1: Transaction Data

Tid	Items bought
1	Drink, Nuts, Diaper
2	Drink, Coffee, Diaper
3	Drink, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

Exercise 1.1: The apriori function expects data in a one-hot encoded pandas DataFrame. Use the transaction data that is presented in the Table 1 and convert the them into 2 dimensional array. Display the output here!

OUTPUT

```
[['Drink', 'Nuts', 'Diaper'],
 ['Drink', 'Coffee', 'Diaper'],
 ['Drink', 'Diaper', 'Eggs'],
 ['Nuts', 'Eggs', 'Milk'],
 ['Nuts', 'Coffee', 'Diaper', 'Eggs', 'Milk']]
```

```
from mlxtend.preprocessing import TransactionEncoder

TranEncod = TransactionEncoder()
te_ary = TranEncod.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=TranEncod.columns_)
```

Exercise 1.2: Display the dataset below! What do you see?

```
In [5]: import pandas as pd
        from mlxtend.preprocessing import TransactionEncoder
        TranEncod = TransactionEncoder()
        te_ary = TranEncod.fit(dataset).transform(dataset)
        df = pd.DataFrame(te_ary, columns=TranEncod.columns_)

In [9]: df
```

Out[9]:

	Coffee	Diaper	Drink	Eggs	Milk	Nuts
0	False	True	True	False	False	True
1	True	True	True	False	False	False
2	False	True	True	True	False	False
3	False	False	False	True	True	True
4	True	True	False	True	True	True

Remember that to work with association rules, the algorithm required as input beside the dataset, the minimum support value ε:

$$
\begin{array}{l}
\text{Apriori}(T, \epsilon) \\
\quad L_1 \leftarrow \{\text{large } 1 - \text{itemsets}\} \\
\quad k \leftarrow 2 \\
\quad \textbf{while } L_{k-1} \neq \emptyset \\
\quad\quad C_k \leftarrow \{c = a \cup \{b\} \mid a \in L_{k-1} \wedge b \notin a, \{s \subseteq c \mid |s| = k-1\} \subseteq L_{k-1}\} \\
\quad\quad \textbf{for } \text{transactions } t \in T \\
\quad\quad\quad D_t \leftarrow \{c \in C_k \mid c \subseteq t\} \\
\quad\quad\quad \textbf{for } \text{candidates } c \in D_t \\
\quad\quad\quad\quad count[c] \leftarrow count[c] + 1 \\
\quad\quad L_k \leftarrow \{c \in C_k \mid count[c] \geq \epsilon\} \\
\quad\quad k \leftarrow k + 1 \\
\quad \textbf{return } \bigcup_k L_k
\end{array}
$$

So, let us return the items and itemsets with at least 60% support.

```
apriori(df, min_support=0.6)
```

By default, `apriori` returns the column indices of the items, which may be useful in downstream operations such as association rule mining. For better readability, we can set `use_colnames=True` to convert these integer values into the respective item names:

```
apriori(df, min_support=0.6, use_colnames=True)
```

	support	itemsets
0	0.8	(Diaper)
1	0.6	(Drink)
2	0.6	(Eggs)
3	0.6	(Nuts)
4	0.6	(Diaper, Drink)

Exercise 1.3: Find the possible rules and calculate the confidence of each rule you find?

EXE: 1.3

P(B|A) = |B∩A| / |A|

Drink=3 Nuts=3 Eggs=3 Diaper=4 Milk=2 Coffee=2

Drink-Diaper=3/3*100=100%

Diaper- Drink=3/4*100=75%

Diaper-Nuts=2/4*100=50%

Diaper-coffee=2/4*100=50%

Diaper-eggs=2/4*100=50%

Nuts- Diaper=2/3*100=66.5%

Nuts-Milke=2/3*100=66.5%

Nuts-eggs=2/3*100=66.5%

Coffee- Diaper=2/2*100=100%

Eggs- Diaper=2/3*100=66.5%

Eggs-Nuts=2/3*100=66.5%

Eggs-Milke =2/3*100=66.5%

2. Selecting and Filtering Results

Before moving ahead, let us decrease the minimum support to 30%. This leads to increase the number of returned items and itemsets. Let us now filter out these rules.The advantage of working with pandas `DataFrames` is that we can use its convenient features to filter the results. For instance, let's assume we are only interested in `itemsets` of length 2 that have a support of at least 50% percent. First, we create the frequent itemsets via `apriori` and add a new column that stores the length of each itemset:

```
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

	support	itemsets	length
0	0.8	(Diaper)	1
1	0.6	(Drink)	1
2	0.6	(Eggs)	1
3	0.6	(Nuts)	1
4	0.6	(Diaper, Drink)	2

Then, you can select the results that satisfy our desired criteria as follows:

```
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.5) ]
```

	support	itemsets	length
4	0.6	(Diaper, Drink)	2

Similarly, using the Pandas API, we can select entries based on the "itemsets" column:

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'Diaper', 'Drink'} ]
```

Good luck