# SD21063 TEAN JIN HE Data Mining Lab Report 5

December 27, 2023

## 1 Data Mining Lab Report 5

**NAME: TEAN JIN HE**

**MATRIC ID: SD21063**

**SECTION: 02G**

#### 1.0.1 Case Study:

This process of identifying an association between products/items is called association rule mining. To implement association rule mining, many algorithms have been developed. Apriori algorithm is one of the most popular and arguably the most efficient algorithms among them. Let us discuss what an Apriori algorithm is.

Below are the eight days transaction data from Day 1. This dataset contains 6 items and 22 transaction records.

#### 1.0.2 Question 1

***General Knowledge***

**Discuss the two application of Apriori algorithms in real-world problems. Give reference/ references.** The Apriori algorithm is a data mining technique for frequent item set mining and association rule learning over relational databases. It can be used to discover patterns and trends in various domains, such as market basket analysis, web usage mining, bioinformatics, and text mining. Here are two examples of its applications in real-world problems:

- Market basket analysis: This is the process of finding associations between items that customers buy together frequently. For example, if customers often buy bread and butter together, then the Apriori algorithm can generate a rule that says {bread} -> {butter}, meaning that if a customer buys bread, they are likely to buy butter as well. This can help retailers to design effective marketing strategies, such as placing related items together, offering discounts, or recommending products to customers.
- Web usage mining: This is the process of analyzing the behavior and preferences of web users based on their browsing history. For example, if web users often visit pages A, B, and C in sequence, then the Apriori algorithm can generate a rule that says {A, B} -> {C}, meaning that if a user visits pages A and B, they are likely to visit page C as well. This can help web developers to improve the design and functionality of their websites, such as providing personalized content, optimizing navigation, or detecting anomalies.

Reference:
https://en.wikipedia.org/wiki/Apriori_algorithm
https://www.geeksforgeeks.org/implementing-apriori-algorithm-in-python/
https://botpenguin.com/glossary/apriori-algorithm
https://medium.com/@monocosmo77/applications-of-apriori-algorithm-part1-machine-learning-2023-fa8de125b15d

### 1.0.3 Question 2

***Python***

### 1.0.4 a. Import related libraries and load the Day1.csv dataset.

[1]:
```python
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.frequent_patterns import apriori as ml_apriori
from numpy import nan
```

[2]:
```python
df = pd.read_csv('C:\\Users\\user\\OneDrive\\Desktop\\Sem 5 Slide\\Data␣
 ↪Mining\\dataset\\Day1.csv', header = None)
df
```

[2]:

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | Wine | Chips | Bread | Butter | Milk | Apple |
| 1 | Wine | NaN | Bread | Butter | Milk | NaN |
| 2 | NaN | NaN | Bread | Butter | Milk | NaN |
| 3 | NaN | Chips | NaN | NaN | NaN | Apple |
| 4 | Wine | Chips | Bread | Butter | Milk | Apple |
| 5 | Wine | Chips | NaN | NaN | Milk | NaN |
| 6 | Wine | Chips | Bread | Butter | NaN | Apple |
| 7 | Wine | Chips | NaN | NaN | Milk | NaN |
| 8 | Wine | NaN | Bread | NaN | NaN | Apple |
| 9 | Wine | NaN | Bread | Butter | Milk | NaN |
| 10 | NaN | Chips | Bread | Butter | NaN | Apple |
| 11 | Wine | NaN | NaN | Butter | Milk | Apple |
| 12 | Wine | Chips | Bread | Butter | Milk | NaN |
| 13 | Wine | NaN | Bread | NaN | Milk | Apple |
| 14 | Wine | NaN | Bread | Butter | Milk | Apple |
| 15 | Wine | Chips | Bread | Butter | Milk | Apple |
| 16 | NaN | Chips | Bread | Butter | Milk | Apple |
| 17 | NaN | Chips | NaN | Butter | Milk | Apple |
| 18 | Wine | Chips | Bread | Butter | Milk | Apple |
| 19 | Wine | NaN | Bread | Butter | Milk | Apple |
| 20 | Wine | Chips | Bread | NaN | Milk | Apple |
| 21 | NaN | Chips | NaN | NaN | NaN | NaN |

### 1.0.5 b. Have a glance at the records.

[3]:
```
# Print the first five rows of the DataFrame
df.head()
```

[3]:

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | Wine | Chips | Bread | Butter | Milk | Apple |
| 1 | Wine | NaN | Bread | Butter | Milk | NaN |
| 2 | NaN | NaN | Bread | Butter | Milk | NaN |
| 3 | NaN | Chips | NaN | NaN | NaN | Apple |
| 4 | Wine | Chips | Bread | Butter | Milk | Apple |

[4]:
```
# change the NaN to blanks
df = df.fillna('')
df
```

[4]:

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | Wine | Chips | Bread | Butter | Milk | Apple |
| 1 | Wine | | Bread | Butter | Milk | |
| 2 | | | Bread | Butter | Milk | |
| 3 | | Chips | | | | Apple |
| 4 | Wine | Chips | Bread | Butter | Milk | Apple |
| 5 | Wine | Chips | | | Milk | |
| 6 | Wine | Chips | Bread | Butter | | Apple |
| 7 | Wine | Chips | | | Milk | |
| 8 | Wine | | Bread | | | Apple |
| 9 | Wine | | Bread | Butter | Milk | |
| 10 | | Chips | Bread | Butter | | Apple |
| 11 | Wine | | | Butter | Milk | Apple |
| 12 | Wine | Chips | Bread | Butter | Milk | |
| 13 | Wine | | Bread | | Milk | Apple |
| 14 | Wine | | Bread | Butter | Milk | Apple |
| 15 | Wine | Chips | Bread | Butter | Milk | Apple |
| 16 | | Chips | Bread | Butter | Milk | Apple |
| 17 | | Chips | | Butter | Milk | Apple |
| 18 | Wine | Chips | Bread | Butter | Milk | Apple |
| 19 | Wine | | Bread | Butter | Milk | Apple |
| 20 | Wine | Chips | Bread | | Milk | Apple |
| 21 | | Chips | | | | |

[5]:
```
df.describe()
```

[5]:

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| count | 22 | 22 | 22 | 22 | 22 | 22 |
| unique | 2 | 2 | 2 | 2 | 2 | 2 |
| top | Wine | Chips | Bread | Butter | Milk | Apple |
| freq | 16 | 14 | 16 | 15 | 17 | 15 |

### 1.0.6 c. Check the shape of the dataset.

[6]:
```
# Print the number of rows and columns of the DataFrame
df.shape
```

[6]:
```
(22, 6)
```

### 1.0.7 d. Convert Pandas DataFrame into a list of lists.

[7]:
```
#Converting the pandas data frame into a list of lists
records =[]
for i in range(0,22):
    records.append([str(df.values[i,j]) for j in range(0,6)])

records
```

[7]:
```
[['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', ''],
 ['', '', 'Bread', 'Butter', 'Milk', ''],
 ['', 'Chips', '', '', '', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', '', '', 'Milk', ''],
 ['Wine', 'Chips', 'Bread', 'Butter', '', 'Apple'],
 ['Wine', 'Chips', '', '', 'Milk', ''],
 ['Wine', '', 'Bread', '', '', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', ''],
 ['', 'Chips', 'Bread', 'Butter', '', 'Apple'],
 ['Wine', '', '', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', ''],
 ['Wine', '', 'Bread', '', 'Milk', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['', 'Chips', '', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', '', 'Milk', 'Apple'],
 ['', 'Chips', '', '', '', '']]
```

### 1.0.8 e. Build the Apriori model for Day1.csv dataset. Explain the output.

[8]:
```
#Building apriori model
from apyori import apriori
rules = apriori(records, min_support=0.50, min_confidence = 0.7, min_lift = 1.
 ↪2, min_length = 2)
results = list(rules)
```

[9]:
```
print(len(results))
```

```
1
```

[10]:
```python
print(results)
```

```
[RelationRecord(items=frozenset({'Butter', 'Bread', 'Milk'}), support=0.5,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'Butter'}),
items_add=frozenset({'Bread', 'Milk'}), confidence=0.7333333333333334,
lift=1.241025641025641), OrderedStatistic(items_base=frozenset({'Bread',
'Milk'}), items_add=frozenset({'Butter'}), confidence=0.8461538461538461,
lift=1.241025641025641)])]
```

The support value for the first rule is 0.5. This number is calculated by dividing the number of transactions containing 'Milk,' 'Bread,' and 'Butter' by the total number of transactions.

The confidence level for the rule is 0.846. This will show that out of all the transactions that contain both "Milk", "Bread" and 'Butter' which is 84.6 %.

The lift of 1.241 tells us that 'Butter' is 1.241 times more likely to be bought by the consumers who buy both 'Milk' and 'Butter' compared to the default likelihood sale of 'Butter.'

### 1.0.9 f. Print out the association rule. Explain the output.

[11]:
```python
data = df.values.tolist()
data
```

[11]:
```
[['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', ''],
 ['', '', 'Bread', 'Butter', 'Milk', ''],
 ['', 'Chips', '', '', '', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', '', '', 'Milk', ''],
 ['Wine', 'Chips', 'Bread', 'Butter', '', 'Apple'],
 ['Wine', 'Chips', '', '', 'Milk', ''],
 ['Wine', '', 'Bread', '', '', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', ''],
 ['', 'Chips', 'Bread', 'Butter', '', 'Apple'],
 ['Wine', '', '', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', ''],
 ['Wine', '', 'Bread', '', 'Milk', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['', 'Chips', '', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', '', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', '', 'Milk', 'Apple'],
 ['', 'Chips', '', '', '', '']]
```

[12]:
```python
#Let's transform the list, with one-hot encoding
from mlxtend.preprocessing import TransactionEncoder
a = TransactionEncoder()
a_data = a.fit(data).transform(data)
df = pd.DataFrame(a_data,columns=a.columns_)
df = df.replace(False,0)
df = df.iloc[:,1:]
df
```

[12]:

| | Apple | Bread | Butter | Chips | Milk | Wine |
|---|---|---|---|---|---|---|
| 0 | True | True | True | True | True | True |
| 1 | 0 | True | True | 0 | True | True |
| 2 | 0 | True | True | 0 | True | 0 |
| 3 | True | 0 | 0 | True | 0 | 0 |
| 4 | True | True | True | True | True | True |
| 5 | 0 | 0 | 0 | True | True | True |
| 6 | True | True | True | True | 0 | True |
| 7 | 0 | 0 | 0 | True | True | True |
| 8 | True | True | 0 | 0 | 0 | True |
| 9 | 0 | True | True | 0 | True | True |
| 10 | True | True | True | True | 0 | 0 |
| 11 | True | 0 | True | 0 | True | True |
| 12 | 0 | True | True | True | True | True |
| 13 | True | True | 0 | 0 | True | True |
| 14 | True | True | True | 0 | True | True |
| 15 | True | True | True | True | True | True |
| 16 | True | True | True | True | True | 0 |
| 17 | True | 0 | True | True | True | 0 |
| 18 | True | True | True | True | True | True |
| 19 | True | True | True | 0 | True | True |
| 20 | True | True | 0 | True | True | True |
| 21 | 0 | 0 | 0 | True | 0 | 0 |

[13]:
```python
#set a threshold value for the support value and calculate the support value.
threshold = ml_apriori(df, min_support = 0.5, use_colnames = True, verbose = 1)
threshold
```

```
Processing 4 combinations | Sampling itemset size 4
```

```
D:\anaconda\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110:
DeprecationWarning: DataFrames with non-bool types result in worse
computationalperformance and their support might be discontinued in the
future.Please use a DataFrame with bool type
  warnings.warn(
```

[13]:

| | support | itemsets |
|---|---|---|
| 0 | 0.681818 | (Apple) |
| 1 | 0.727273 | (Bread) |

```
2   0.681818              (Butter)
3   0.636364               (Chips)
4   0.772727                (Milk)
5   0.727273                (Wine)
6   0.545455        (Bread, Apple)
7        0.5       (Butter, Apple)
8        0.5         (Milk, Apple)
9        0.5         (Apple, Wine)
10  0.590909       (Butter, Bread)
11  0.590909         (Bread, Milk)
12  0.590909         (Bread, Wine)
13  0.590909        (Butter, Milk)
14       0.5        (Butter, Wine)
15  0.636364          (Milk, Wine)
16       0.5  (Butter, Bread, Milk)
17       0.5    (Bread, Wine, Milk)
```

[14]:

```
#Let's view our interpretation values using the Associan rule function.
df_association_rules = association_rules(threshold, metric = "confidence",␣
 ↪min_threshold = 0.8)
df_association_rules
```

[14]:

```
        antecedents consequents  antecedent support  consequent support  \
0           (Apple)     (Bread)            0.681818            0.727273
1          (Butter)     (Bread)            0.681818            0.727273
2           (Bread)    (Butter)            0.727273            0.681818
3           (Bread)      (Milk)            0.727273            0.772727
4           (Bread)      (Wine)            0.727273            0.727273
5            (Wine)     (Bread)            0.727273            0.727273
6          (Butter)      (Milk)            0.681818            0.772727
7            (Milk)      (Wine)            0.772727            0.727273
8            (Wine)      (Milk)            0.727273            0.772727
9   (Butter, Bread)      (Milk)            0.590909            0.772727
10   (Butter, Milk)     (Bread)            0.590909            0.727273
11    (Bread, Milk)    (Butter)            0.590909            0.681818
12    (Bread, Wine)      (Milk)            0.590909            0.772727
13    (Bread, Milk)      (Wine)            0.590909            0.727273

     support  confidence      lift  leverage  conviction  zhangs_metric
0   0.545455    0.800000  1.100000  0.049587    1.363636       0.285714
1   0.590909    0.866667  1.191667  0.095041    2.045455       0.505495
2   0.590909    0.812500  1.191667  0.095041    1.696970       0.589744
3   0.590909    0.812500  1.051471  0.028926    1.212121       0.179487
4   0.590909    0.812500  1.117188  0.061983    1.454545       0.384615
5   0.590909    0.812500  1.117188  0.061983    1.454545       0.384615
6   0.590909    0.866667  1.121569  0.064050    1.704545       0.340659
7   0.636364    0.823529  1.132353  0.074380    1.545455       0.514286
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | 0.636364 | 0.875000 | 1.132353 | 0.074380 | 1.818182 | 0.428571 |
| 9 | 0.500000 | 0.846154 | 1.095023 | 0.043388 | 1.477273 | 0.212121 |
| 10 | 0.500000 | 0.846154 | 1.163462 | 0.070248 | 1.772727 | 0.343434 |
| 11 | 0.500000 | 0.846154 | 1.241026 | 0.097107 | 2.068182 | 0.474747 |
| 12 | 0.500000 | 0.846154 | 1.095023 | 0.043388 | 1.477273 | 0.212121 |
| 13 | 0.500000 | 0.846154 | 1.163462 | 0.070248 | 1.772727 | 0.343434 |

Based on the index value:

The probability of seeing milk sales is seen as 77% and the wine sales intake is seen as 73%.

From this point, we can say that the support of both of them is measured as 64% and 82% of those who will buys milk and wine as well.

Since the consumers who buy milk will likely consume 13% more wine than consumers who don't buy milk.

So, their correlation with each other is seen as 1.55.