Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Apriori algorithm

RADI608: Data Mining and Machine Learning
RADI602: Data Mining and Knowledge Discovery

Lect. Anuchate Pattanateepapon, D.Eng.

Section of Data Science for Healthcare

Department of Clinical Epidemiology and Biostatistics

Faculty of Medicine Ramathibodi Hospital, Mahidol University

Wisdom of the Land

Guideline

- Association Rule Learning
- Apriori Algorithm
- Apriori Algorithm in Python

Association Rule Learning

What's an association rule learning?

A relationship-extraction method for finding relations between features in large or big dataset which based on the concept of strong rules.

An association rule introduced by Rakesh A., Tomasz I. and Arun S. in 1993 [1] for discovering relations between products that retrieved large-scale transaction data from the point-of-sale systems.

Example: the rule {onions, potatoes} $\Rightarrow$ {burgers}

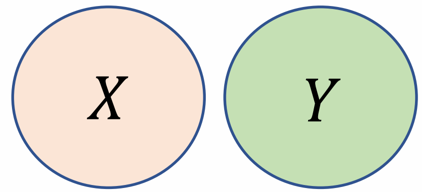
An association rule

Let $I = \{i_1, i_2, \dots, i_n\}$ be a set of n binary attributes called items
Let $T = \{t_1, t_2, \dots, t_n\}$ be a set of transactions called databases

Each transaction in T has a unique transaction ID and contains a subset of the items in I , then a rule is defined as:

$$X \Rightarrow Y \quad \text{,where } X, Y \subseteq I, \quad X \cap Y = \emptyset$$

LHS RHS



An example of transactional database

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs

http://dataminingtrend.com/2014/association-rules/

How to select interesting rules from the set of possible rules

Select by using the minimum thresholds on:

- Support : how frequently an itemset appears in the dataset
  (count from an itemset)
- Confidence: how often the rule has been found to be true
  (count from an interesting rule)
- Lift: the lift of a rule (the lift value is a measure of importance of a rule)
  (the ratio of the observed support to that expected
  if X and Y were independent)

How to calculate the support

From $I = \{i_1, i_2, \dots, i_n\}$
and $T = \{t_1, t_2, \dots, t_n\}$
and $X \Rightarrow Y, \quad X, Y \subseteq I$

how frequently an itemset appears in the dataset

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

Low support rule is also likely to be uninteresting from a business perspective because it may not be profitable to promote items

Range: [close to 0 <-> 1]

How to calculate the support

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

Example: X ={apples} has a frequency : 2 transactions

T equals 4 transactions

the support = 2/4 = 0.50 or 50%

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs

How to calculate the confidence or $P(E_Y|E_X)$

From $supp(X) = |\{t \in T;\ X \subseteq t\}|/|T|$
and $X \Rightarrow Y,\quad X, Y \subseteq I$

how often the rule has been found to be true

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

measure the reliability of the inference made by a rule

Range: [close to 0 <-> 1]

How to calculate the confidence or $P(E_Y|E_X)$

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

Example: X = apple, Y = cereal

{apple} has a $supp(X) = 0.50$

{apple, cereal,…} has a frequency = 2 transactions

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs

$$supp(X \cup Y) = 2/4 = 0.50$$

X Y

{apple} $\Rightarrow$ {cereal} has a confidence $= \frac{0.50}{0.50} = 1$ or 100%

How to calculate the lift

- The lift is a degree to which those two occurrences are dependent on one another
- help us to consider the confidence of the rule and overall data

the lift of a rule

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

0< Range <1 — if $\mathrm{supp}(X \cup Y)$ less than $(\mathrm{supp}(X) \times \mathrm{supp}(Y))$
Y is unlikely to be bought if item X is bought

Range =1 — if $\mathrm{supp}(X \cup Y)$ equals $(\mathrm{supp}(X) \times \mathrm{supp}(Y))$
implying no association between X and Y

Range >1 — if $\mathrm{supp}(X \cup Y)$ larger than $(\mathrm{supp}(X) \times \mathrm{supp}(Y))$
Y is likely to be bought if item X is bought

$$lift(A \Rightarrow B) = \frac{2/4}{3/4 \times 3/4}$$

$$lift(C \Rightarrow D) = \frac{1/4}{2/4 \times 2/4}$$

$$lift(E \Rightarrow F) = \frac{2/4}{2/4 \times 3/4}$$

How to calculate the lift

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

Example:

{cereal,…} has a $supp(X) = \frac{3}{4} = 0.75$ and

{cereal, eggs,…} $supp(X \cup Y) = \frac{2}{4} = 0.50$

{eggs,…} has a $supp(Y) = \frac{3}{4} = 0.75$

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs diapers

X Y

{cereal} $\Rightarrow$ {eggs} has a lift $= \frac{0.50}{0.75 \times 0.75} = 0.88$

lift < 1 , lets us know that Y is unlikely to be bought if item X is bought

How to calculate the lift

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

Example:

{apple,…} has a $supp(X) = \frac{2}{4} = 0.50$ and

{apple, diapers,…} $supp(X \cup Y) = \frac{1}{4} = 0.25$

{diapers,…} has a $supp(Y) = \frac{2}{4} = 0.50$

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs diapers

X Y

{apple} $\Rightarrow$ {diapers} has a lift $= \frac{0.25}{0.50 \times 0.50} = 1$

lift =1 , implying no association between X and Y

How to calculate the lift

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

Example:

{apple,…} has a $supp(X) = 0.50$ and {apple,cereal,…} $supp(X \cup Y) = 0.50$

{cereal,…} has a $supp(Y) = \frac{3}{4} = 0.75$

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs diapers

X Y

{apple} $\Rightarrow$ {cereal} has a lift $= \frac{0.50}{0.50 \times 0.75} = 1.33$

lift > 1 , lets us know that Y is likely to be bought if item X is bought

Common strategy adopted by many association rules

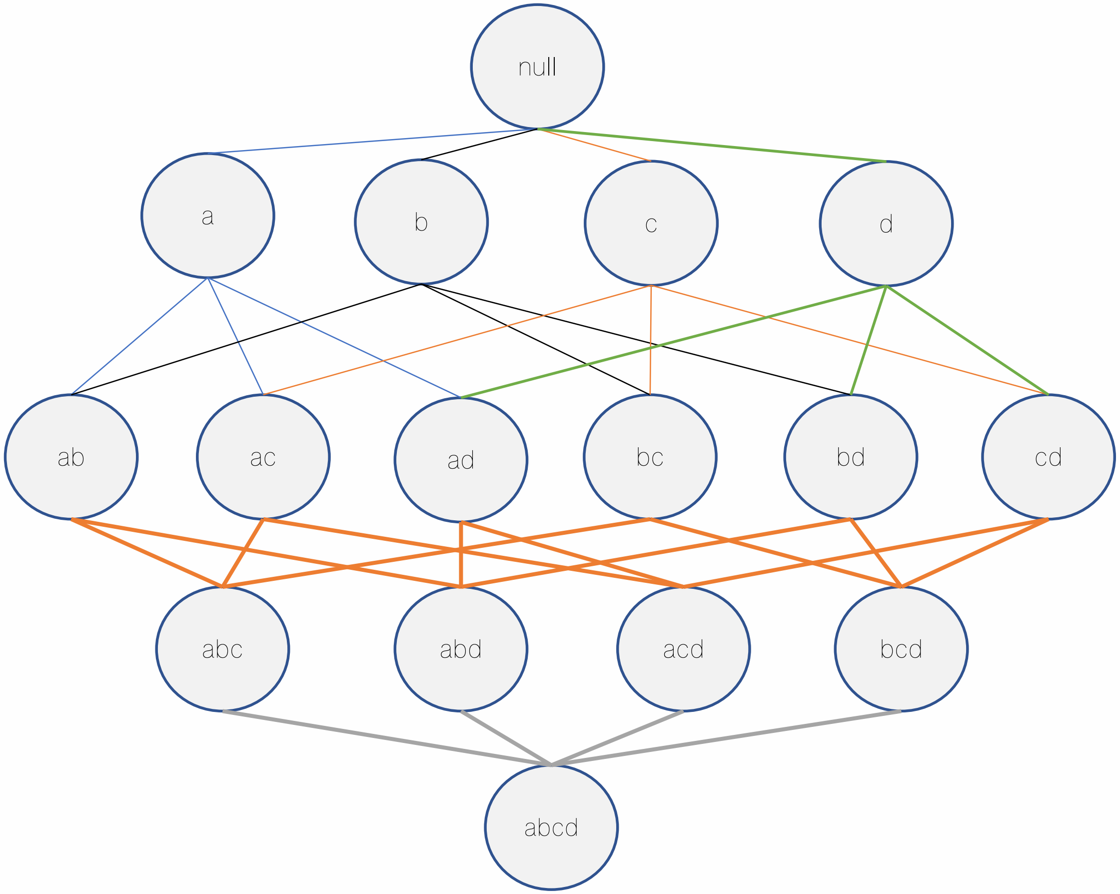
1. **Frequent itemset generation:** generate all the itemset (called the frequent itemsets) that satisfy the minimum support threshold

2. **Rule generation:** extract all the high-confidence rules (called the strong rules) from the frequent itemsets found in the previous step

A lattice structure used to enumerate all possible itemsets

For $I = \{a, b, c, d\}$ that contain k items could generate up to $2^k - 1$ frequent itemsets

If $k = 4$, then Possible item $= 2^4 - 1$ $= 15$ itemsets



Apriori Algorithm

What's an Apriori algorithm?

- an algorithm for frequent itemset mining and association rule learning over transactional databases
- proceed by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those itemsets appear sufficiently often in the database
- The frequent itemsets determined by Apriori can be used to determine association rules which highlight general trends in the database

https://en.wikipedia.org/wiki/Apriori_algorithm

Apriori principle

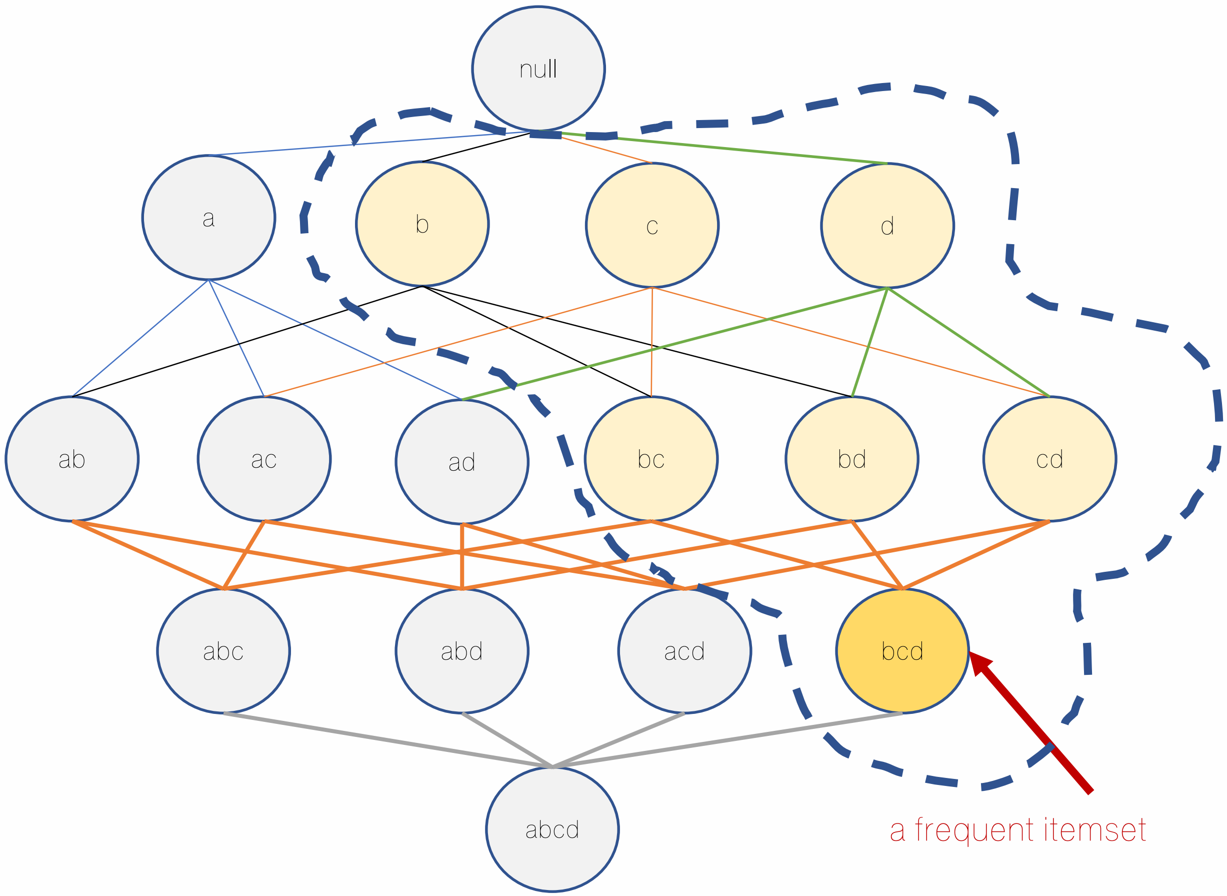
If an itemset is frequent, then all of its subsets must also be frequent:

Example

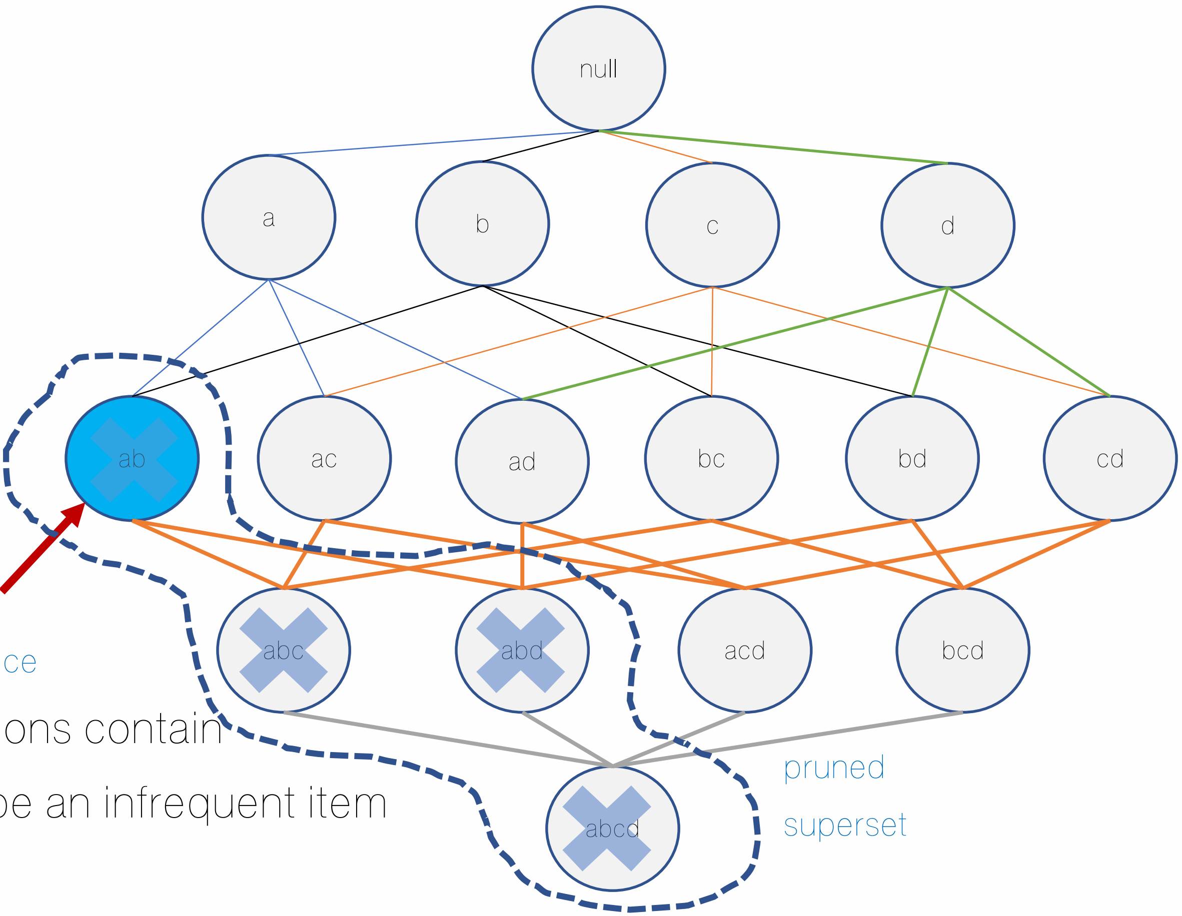
If { b, c, d } is a frequent itemset, all subset of {b, c, d } must also be frequent itemset:

{b, c}, {b, d}, {c, d}, {b}, {c} and {d}

Apriori principle



How an Apriori algorithm pruned the possible itemsets



If {a, b} is an infrequent

Itemset or low confidence

Then, all transactions contain

{a, b } must also be an infrequent item

Apriori algorithm pseudocode

C_k: The candidate itemsets of size k, L_k: The frequent itemsets of size k

U: The candidate rules

min_supp : The minimum support threshold (a support-based pruning)

L_1 = {the frequent 1-itemsets};

for (k = 2; $L_{k-1} \neq \emptyset$; k ++)

  C_{k+1} = GenerateCandidates (L_{k-1})

    for each transaction t in database do

      increment count of candidates in C_{k+1} that are contained in t

    end for

    L_{k+1} = candidates in C_{k+1} with support ≥ min_supp

End for

return $U \leftarrow L_k$

GenerateCandidates ($L_{k\text{-}1}$) function

$C_k \leftarrow \emptyset$;

For all $l_1, l_2 \in L_{k-1}$

With $l_1 = \{i_1, i_2, \dots, i_{k-1}\}$

and $l_2 = \{i_1, i_2, \dots, i'_{k-1}\}$

and $i_{k-1} < i'_{k-1}$ do

$c = \{i_1, i_2, \dots, i'_{k-1}\}$; //join l_1 and l_2

$C_k = C_k \cup \{c\}$;

for each $(k-1)$ subset s of c do

if ($s \notin L_{k-1}$) then

delete c from C_k //prune

end

end

Return C_k

Step-by-step to create the association rules

1. Initial an minimum support threshold (pruning) and apply to find all frequent itemsets in database:

For this example, minimum support threshold = 50%, minimum conf.= 80%

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs

Step-by-step to create the association rules

2. From a frequent item, find a support of each item:

Items	Transaction ID				Support
	1	2	3	4	
apple	1	0	1	0	2/4 = 50%
beer	0	1	1	1	3/4 = 75%
cereal	1	1	1	0	3/4 = 75%
diapers	1	0	0	0	1/4 = 25%
eggs	0	1	1	1	3/4 = 75%

Step-by-step to create the association rules

2. From a frequent item, find a support of each item:

Items	Transaction ID				Support
	1	2	3	4	
apple	1	0	1	0	2/4 = 50%
beer	0	1	1	1	3/4 = 75%
cereal	1	1	1	0	3/4 = 75%
~~diapers~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1/4 = 25%~~
eggs	0	1	1	1	3/4 = 75%

Support-based pruning: an item that have a support < 50%

Step-by-step to create the association rules

3. Generate 2-items per itemset by using the frequent items from a previous table

when {apple, beer} = {beer, apple}

Itemset	Transaction ID				Support
	1	2	3	4	
{apple, beer}	0	0	1	0	1/4 = 25%
{apple, cereal}	1	0	1	0	2/4 = 50%
{apple, eggs}	0	0	1	0	1/4 = 25%
{beer, cereal}	0	1	1	0	2/4 = 50%
{beer, eggs}	0	1	1	1	3/4 = 75%
{cereal, eggs}	0	1	1	0	2/4 = 50%

Step-by-step to create the association rules

3. Generate 2-items per itemset by using the frequent items from a previous table

when {apple, beer} = {beer, apple}

Itemset	Transaction ID				Support
	1	2	3	4	
~~{apple, beer}~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1/4 = 25%~~
{apple, cereal}	1	0	1	0	2/4 = 50%
~~{apple, eggs}~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1/4 = 25%~~
{beer, cereal}	0	1	1	0	2/4 = 50%
{beer, eggs}	0	1	1	1	3/4 = 75%
{cereal, eggs}	0	1	1	0	2/4 = 50%

Support-based pruning: an item that have a support < 50%

Step-by-step to create the association rules

4. Generate 3-items per itemset by using the frequent itemset from a previous table

when {beer, cereal} join {cereal, eggs} = {beer, cereal, eggs}

Itemset	Transaction ID				Support
	1	2	3	4	
{beer, cereal, eggs}	0	1	1	0	2/4 = 50%

Stop create 4-items per set, because it does not have any frequent item to join and build the new itemset

Step-by-step to create the association rules

Transactional DB

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs

Candidates

Items	Transaction ID				Support
	1	2	3	4	
apple	1	0	1	0	2/4 = 50%
beer	0	1	1	1	3/4 = 75%
cereal	1	1	1	0	3/4 = 75%
~~diapers~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1/4 = 25%~~
eggs	0	1	1	1	3/4 = 75%

Support-based pruning: an item that have a support < 50%

Itemset	Transaction ID				Support
	1	2	3	4	
~~{apple, beer}~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1/4 = 25%~~
{apple, cereal}	1	0	1	0	2/4 = 50%
~~{apple, eggs}~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1/4 = 25%~~
{beer, cereal}	0	1	1	0	2/4 = 50%
{beer, eggs}	0	1	1	1	3/4 = 75%
{cereal, eggs}	0	1	1	0	2/4 = 50%

Support-based pruning: an item that have a support < 50%

Itemset	Transaction ID				Support
	1	2	3	4	
{beer, cereal, eggs}	0	1	1	0	2/4 = 50%

5. Generate the association rules by using the frequent itemset that it has the item ≥ 2

Rule no.	Frequent Itemset	Confidence	Lift
1	apple ⇒ cereal	100%	1.33
2	beer ⇒ eggs	100%	1.33
3	eggs ⇒ beer	100%	1.33
4	{beer, cereal} ⇒ eggs	100%	1.33
5	{cereal, eggs} ⇒ beer	100%	1.33
6	cereal ⇒ apple	67%	1.33
7	beer ⇒ {cereal, eggs}	67%	1.33
8	eggs ⇒ {beer, cereal}	67%	1.33
9	beer ⇒ cereal	67%	0.89
10	cereal ⇒ beer	67%	0.89
11	cereal ⇒ eggs	67%	0.89
12	eggs ⇒ cereal	67%	0.89
13	cereal ⇒ {beer, eggs}	67%	0.89
14	{beer, eggs} ⇒ cereal	67%	0.89

Step-by-step to create the association rules

5. Generate the association rules by using the frequent itemset that it has the item ≥ 2

Rule no.	Frequent Itemset	Confidence	Lift
1	apple ⇒ cereal	100%	1.33
2	beer ⇒ eggs	100%	1.33
3	eggs ⇒ beer	100%	1.33
4	{beer, cereal} ⇒ eggs	100%	1.33
5	{cereal, eggs} ⇒ beer	100%	1.33
6	cereal ⇒ apple	67%	1.33
7	beer ⇒ {cereal, eggs}	67%	1.33
8	eggs ⇒ {beer, cereal}	67%	1.33
9	beer ⇒ cereal	67%	0.89
10	cereal ⇒ beer	67%	0.89
11	cereal ⇒ eggs	67%	0.89
12	eggs ⇒ cereal	67%	0.89
13	cereal ⇒ {beer, eggs}	67%	0.89
14	{beer, eggs} ⇒ cereal	67%	0.89

Then, we have 14 candidate rules

We might be designed to remove the rules that have the Confidence below 0.8

Step-by-step to create the association rules

5. Generate the association rules by using the frequent itemset that it has the item ≥ 2

Rule no.	Frequent Itemset	Confidence	Lift
1	apple ⇒ cereal	100%	1.33
2	beer ⇒ eggs	100%	1.33
3	eggs ⇒ beer	100%	1.33
4	{beer, cereal} ⇒ eggs	100%	1.33
5	{cereal, eggs} ⇒ beer	100%	1.33

So, we have final 5 rules

Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Step-by-step to create the association rules

Transactional DB

Transaction ID	Items
1	apple, cereal, diapers
2	beer, cereal, eggs
3	apple, beer, cereal, eggs
4	beer, eggs

Candidates

Items	Transaction ID				Support
	1	2	3	4	
apple	1	0	1	0	2/4 = 50%
beer	0	1	1	1	3/4 = 75%
cereal	1	1	1	0	3/4 = 75%
~~diapers~~	~~1~~	~~0~~	~~0~~	~~0~~	~~1/4 = 25%~~
eggs	0	1	1	1	3/4 = 75%

Support-based pruning: an item that have a support < 50%

Itemset	Transaction ID				Support
	1	2	3	4	
~~{apple, beer}~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1/4 = 25%~~
{apple, cereal}	1	0	1	0	2/4 = 50%
~~{apple, eggs}~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1/4 = 25%~~
{beer, cereal}	0	1	1	0	2/4 = 50%
{beer, eggs}	0	1	1	1	3/4 = 75%
{cereal, eggs}	0	1	1	0	2/4 = 50%

Support-based pruning: an item that have a support < 50%

Itemset	Transaction ID				Support
	1	2	3	4	
{beer, cereal, eggs}	0	1	1	0	2/4 = 50%

Association rules

Rule no.	Frequent Itemset	Confidence	Lift
1	apple ⇒ cereal	100%	1.33
2	beer ⇒ eggs	100%	1.33
3	eggs ⇒ beer	100%	1.33
4	{beer, cereal} ⇒ eggs	100%	1.33
5	{cereal, eggs} ⇒ beer	100%	1.33
6	cereal ⇒ apple	67%	1.33
7	beer ⇒ {cereal, eggs}	67%	1.33
8	eggs ⇒ {beer, cereal}	67%	1.33
9	beer ⇒ cereal	67%	0.89
10	cereal ⇒ beer	67%	0.89
11	cereal ⇒ eggs	67%	0.89
12	eggs ⇒ cereal	67%	0.89
13	cereal ⇒ {beer, eggs}	67%	0.89
14	{beer, eggs} ⇒ cereal	67%	0.89

Lower than minimum confidence 0.80

How to apply the lift of the rule

Focus on two event a) apple ⇒ cereal and b) cereal ⇒ apple

If customer buy apple, they will certainly buy cereal (confidence = 100%)

If customer buy cereal, they will probably buy apple (confidence = 67%)

apple and cereal has a lift = 1.33,

means these two event are independent of each other (The rule can converse)

or

The converse is true for {apple ⇒ cereal}

we may conclude that if someone buys cereal, he is very likely to buy apple as well

How to apply the lift of the rule

Focus on two event a) eggs ⇒ cereal and b) cereal ⇒ eggs

If customer buy eggs, they will probably buy cereal (confidence = 67%)

If customer buy cereal, they will probably buy eggs (confidence = 67%)

eggs and cereal has a lift = 0.89,

means these two event are dependent of each other

or

The converse is false for {eggs ⇒ cereal}

we may conclude that if someone buys cereal, he would likely be averse to eggs.

How to pick appropriate support & confidence & lift

Rule no.	Frequent Itemset	Support	Confidence	Lift
1	apple ⇒ cereal	50%	100%	1.33
4	{beer, cereal} ⇒ eggs	50%	100%	1.33
8	eggs ⇒ {beer, cereal}	75%	67%	1.33
10	cereal ⇒ beer	75%	67%	0.89
14	{beer, eggs} ⇒ cereal	70%	67%	0.89

usually we want all three to be high

high support: should apply to a large amount of cases

high confidence: should be correct often

high lift: indicates it is not just a coincidence

A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

Apriori Algorithm in Python

http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/

Apriori algorithm in Python

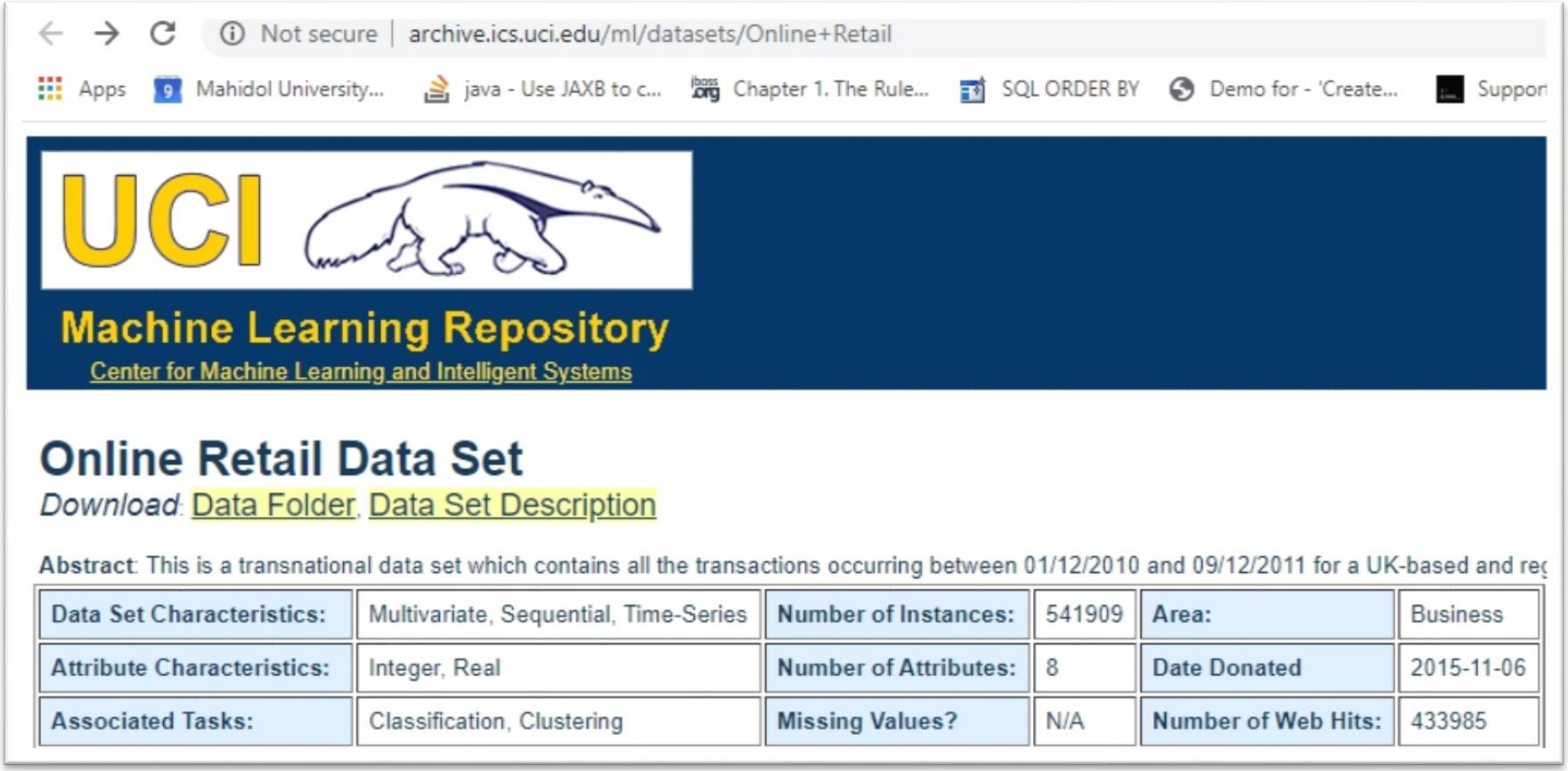
Example 1

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df = pd.read_excel('H:/Online_Retail.xlsx')
df.head()
```

```
pip install mlxtend
```

Apriori algorithm in Python


Not secure | archive.ics.uci.edu/ml/datasets/Online+Retail

Apps · Mahidol University... · java - Use JAXB to c... · Chapter 1. The Rule... · SQL ORDER BY · Demo for - 'Create... · Suppor

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Online Retail Data Set

Download: Data Folder, Data Set Description

Abstract: This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and re

Data Set Characteristics:	Multivariate, Sequential, Time-Series	**Number of Instances:**	541909	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	8	**Date Donated**	2015-11-06
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	433985

Apriori algorithm in Python

df - DataFrame

Index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGH…	6	2010-12-01 08:26:00	2.55	17850	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT …	8	2010-12-01 08:26:00	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WAT…	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE…	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
5	536365	22752	SET 7 BABUSHKA NES…	2	2010-12-01 08:26:00	7.65	17850	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LI…	6	2010-12-01 08:26:00	4.25	17850	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850	United Kingdom

Format Resize Background color Column min/max Save and Close Close

Apriori algorithm in Python

There is a little cleanup, we need to do. First, some of the descriptions have spaces that need to be removed. We'll also drop the rows that don't have invoice numbers and remove the credit transactions (those with invoice numbers containing C).

```
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
```

Remove spaces at the beginning and the end

Drop rows which contain missing values

Change column InvoiceNo type to String

Re-construct df by removing rows that have column "InvoiceNo" contain "C"

Apriori algorithm in Python

After the cleanup, we need to consolidate the items into 1 transaction per row with each product.

```
basket = (df[df['Country'] ==“France"]
        .groupby(['InvoiceNo', 'Description'])
        ['Quantity'].sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))
```

Re-construct df by select all rows that have column “Country” = “France”

Group by column InvoiceNo and Description

Sum column Quantity of each Description of the same index (InvoiceNo)

Select column InvoiceNo to be index.

basket - DataFrame

Index	)LOUR SPACEBOY	)URED PARTY BAI	HOUSE PAINTED	3E CARDS WITH E	. SMALL TUBE WC	3MALL TUBE RED	CILS SMAL
544069	0	0	0	0	0	0	0
544115	24	20	0	0	0	24	24
544200	0	0	0	0	0	0	0
544355	0	0	0	0	0	0	0
544423	0	0	0	0	0	0	0
544470	0	0	0	0	0	0	0
544585	0	0	2	0	0	0	0
544817	0	0	0	0	0	0	0
544818	0	0	0	0	0	0	0

Format | Resize | Background color | Column min/max | Save and Close | Close

Name	Type	Size
basket	DataFrame	(392, 1563)

Apriori algorithm in Python

There are a lot of zeros in the data but we also need to make sure any positive values are converted to a 1 and anything less the 0 is set to 0. This step will complete the one hot encoding of the data and remove the postage column (since that charge is not one we wish to explore):

```
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
```

Apriori algorithm in Python

Now that the data is structured properly, we can generate frequent item sets that have a support of at least 7% (this number was chosen so that I could get enough useful examples):

```
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
```

The final step is to generate the rules with their corresponding support, confidence and lift:

Select all rules that have the lift >= 1.00

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
```

Apriori algorithm in Python

rules - DataFrame

Index	antecedents	consequents	ntecedent suppor	onsequent suppo	support	confidence	lift	leverage	conviction
0	frozenset({'ALARM CLOCK BAKELIKE PINK'})	frozenset({'ALARM CLOCK BAKELIKE GREEN'})	0.102041	0.0969388	0.0739796	0.725	7.47895	0.0640879	3.28386
1	frozenset({'ALARM CLOCK BAKELIKE GREEN'})	frozenset({'ALARM CLOCK BAKELIKE PINK'})	0.0969388	0.102041	0.0739796	0.763158	7.47895	0.0640879	3.79138
2	frozenset({'ALARM CLOCK BAKELIKE RED'})	frozenset({'ALARM CLOCK BAKELIKE GREEN'})	0.0943878	0.0969388	0.0790816	0.837838	8.64296	0.0699318	5.56888
3	frozenset({'ALARM CLOCK BAKELIKE GREEN'})	frozenset({'ALARM CLOCK BAKELIKE RED'})	0.0969388	0.0943878	0.0790816	0.815789	8.64296	0.0699318	4.91618
4	frozenset({'ALARM CLOCK BAKELIKE RED'})	frozenset({'ALARM CLOCK BAKELIKE PINK'})	0.0943878	0.102041	0.0739796	0.783784	7.68108	0.0643482	4.15306
5	frozenset({'ALARM CLOCK BAKELIKE PINK'})	frozenset({'ALARM CLOCK BAKELIKE RED'})	0.102041	0.0943878	0.0739796	0.725	7.68108	0.0643482	3.29314
6	frozenset({'SPACEBOY LUNCH BOX'})	frozenset({'DOLLY GIRL LUNCH BOX'})	0.125	0.0994898	0.0714286	0.571429	5.74359	0.0589923	2.10119
7	frozenset({'DOLLY GIRL LUNCH BOX'})	frozenset({'SPACEBOY LUNCH BOX'})	0.0994898	0.125	0.0714286	0.717949	5.74359	0.0589923	3.10227
8	frozenset({'PLASTERS IN TIN CIRCUS PARADE'})	frozenset({'PLASTERS IN TIN SPACEBOY'})	0.168367	0.137755	0.0892857	0.530303	3.84961	0.0660923	1.83575
9	frozenset({'PLASTERS IN TIN SPACEBOY'})	frozenset({'PLASTERS IN TIN CIRCUS PARADE'})	0.137755	0.168367	0.0892857	0.648148	3.84961	0.0660923	2.36359
10	frozenset({'PLASTERS IN TIN CIRCUS PARADE'})	frozenset({'PLASTERS IN TIN WOODLAND ANIMALS'})	0.168367	0.170918	0.102041	0.606061	3.54591	0.0732637	2.10459
11	frozenset({'PLASTERS IN TIN WOODLAND ANIMALS'})	frozenset({'PLASTERS IN TIN CIRCUS PARADE'})	0.170918	0.168367	0.102041	0.597015	3.54591	0.0732637	2.06368
12	frozenset({'PLASTERS IN TIN SPACEBOY'})	frozenset({'PLASTERS IN TIN WOODLAND ANIMALS'})	0.137755	0.170918	0.104592	0.759259	4.44223	0.081047	3.44388
13	frozenset({'PLASTERS IN TIN WOODLAND ANIMALS'})	frozenset({'PLASTERS IN TIN SPACEBOY'})	0.170918	0.137755	0.104592	0.61194	4.44223	0.081047	2.22194

Format | Resize | Background color | Column min/max | Save and Close | Close

Apriori algorithm in Python

We can filter the dataframe using standard pandas code. In this case, look for a large lift (6) and high confidence (.8):

```
filter_rules = rules[ (rules['lift'] >= 6) & (rules['confidence'] >= 0.8) ]
```

filter_rules - DataFrame

Index	antecedents	consequents	ntecedent suppor	onsequent suppo	support	confidence	lift	leverage	conviction
2	frozenset({'ALARM CLOCK BAKELIKE RED'})	frozenset({'ALARM CLOCK BAKELIKE GREEN'})	0.0943878	0.0969388	0.0790816	0.837838	8.64296	0.0699318	5.56888
3	frozenset({'ALARM CLOCK BAKELIKE GREEN'})	frozenset({'ALARM CLOCK BAKELIKE RED'})	0.0969388	0.0943878	0.0790816	0.815789	8.64296	0.0699318	4.91618
17	frozenset({'SET/6 RED SPOTTY PAPER PLATES'})	frozenset({'SET/20 RED RETROSPOT PAPER NAPKINS'})	0.127551	0.132653	0.102041	0.8	6.03077	0.0851208	4.33673
18	frozenset({'SET/6 RED SPOTTY PAPER CUPS'})	frozenset({'SET/6 RED SPOTTY PAPER PLATES'})	0.137755	0.127551	0.122449	0.888889	6.96889	0.104878	7.85204
19	frozenset({'SET/6 RED SPOTTY PAPER PLATES'})	frozenset({'SET/6 RED SPOTTY PAPER CUPS'})	0.127551	0.137755	0.122449	0.96	6.96889	0.104878	21.5561
20	frozenset({'SET/6 RED SPOTTY PAPER CUPS', 'SET/20 RED RETROSPOT PAPER NAPKINS'})	frozenset({'SET/6 RED SPOTTY PAPER PLATES'})	0.102041	0.127551	0.0994898	0.975	7.644	0.0864744	34.898
21	frozenset({'SET/6 RED SPOTTY PAPER PLATES', 'SET/20 RED RETROSPOT PAPER NAPKINS'})	frozenset({'SET/6 RED SPOTTY PAPER CUPS'})	0.102041	0.137755	0.0994898	0.975	7.07778	0.0854332	34.4898
22	frozenset({'SET/6 RED SPOTTY PAPER PLATES', 'SET/6 RED SPOTTY PAPER CUPS'})	frozenset({'SET/20 RED RETROSPOT PAPER NAPKINS'})	0.122449	0.132653	0.0994898	0.8125	6.125	0.0832466	4.62585

Format | Resize | ☑ Background color | ☑ Column min/max | Save and Close | Close

In looking at the rules, it seems that the green and red alarm clocks are purchased together and the red paper cups, napkins and plates are purchased together in a manner that is higher than the overall probability would suggest.

Apriori algorithm in Python

At this point, you may want to look at how much opportunity there is to use the popularity of one product to drive sales of another. For instance, we can see that we sell 340 Green Alarm clocks but only 316 Red Alarm Clocks so maybe we can drive more Red Alarm Clock sales through recommendations?

```
basket['ALARM CLOCK BAKELIKE GREEN'].sum()
```

```
340.0
```

```
basket['ALARM CLOCK BAKELIKE RED'].sum()
```

```
316.0
```

Apriori algorithm in Python

What is also interesting is to see how the combinations vary by country of purchase. Let's check out what some popular combinations might be in Germany:

```
basket2 = (df[df['Country'] =="Germany"]
        .groupby(['InvoiceNo', 'Description'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))

basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)

filter_rules2 = rules2[ (rules2['lift'] >= 4) & (rules2['confidence'] >= 0.5)]
```

Apriori algorithm in Python

What is also interesting is to see how the combinations vary by country of purchase.

Let's check out what some popular combinations might be in Germany:

filter_rules2 - DataFrame

Index	antecedents	consequents	ntecedent suppor	onsequent suppo	support	confidence	lift	leverage	conviction
0	frozenset({'PLASTERS IN TIN CIRCUS PARADE'})	frozenset({'PLASTERS IN TIN WOODLAND ANIMALS'})	0.115974	0.137856	0.0678337	0.584906	4.24289	0.0518461	2.07698
6	frozenset({'PLASTERS IN TIN SPACEBOY'})	frozenset({'PLASTERS IN TIN WOODLAND ANIMALS'})	0.107221	0.137856	0.0612691	0.571429	4.14512	0.0464881	2.01167
10	frozenset({'RED RETROSPOT CHARLOTTE BAG'})	frozenset({'WOODLAND CHARLOTTE BAG'})	0.0700219	0.126915	0.059081	0.84375	6.64817	0.0501942	5.58775

Apriori algorithm in Python

Example 2

```
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df2 = pd.DataFrame(te_ary, columns=te.columns_)
df2

frequent_itemsets3 = apriori(df2, min_support=0.05, use_colnames=True)
rules3 = association_rules(frequent_itemsets3, metric="lift", min_threshold=1)
```

Import Transaction Encoder Library

Transform the dataset to the transaction encode array

convert transaction encode array to Data Frame

Apriori algorithm in Python

rules3 - DataFrame

Index	antecedents	consequents	ntecedent suppor	onsequent suppo	support	confidence	lift	leverage	conviction
0	frozenset({'Apple'})	frozenset({'Eggs'})	0.2	0.8	0.2	1	1.25	0.04	inf
1	frozenset({'Eggs'})	frozenset({'Apple'})	0.8	0.2	0.2	0.25	1.25	0.04	1.06667
2	frozenset({'Apple'})	frozenset({'Kidney Beans'})	0.2	1	0.2	1	1	0	inf
3	frozenset({'Kidney Beans'})	frozenset({'Apple'})	1	0.2	0.2	0.2	1	0	1
4	frozenset({'Apple'})	frozenset({'Milk'})	0.2	0.6	0.2	1	1.66667	0.08	inf
5	frozenset({'Milk'})	frozenset({'Apple'})	0.6	0.2	0.2	0.333333	1.66667	0.08	1.2
6	frozenset({'Ice cream'})	frozenset({'Corn'})	0.2	0.4	0.2	1	2.5	0.12	inf
7	frozenset({'Corn'})	frozenset({'Ice cream'})	0.4	0.2	0.2	0.5	2.5	0.12	1.6

Format | Resize | ☑ Background color | ☑ Column min/max | Save and Close | Close

Apriori algorithm in Python

filter_rules3 = rules3[(rules3['lift'] >= 4) & (rules3['confidence'] >= 0.5)]

filter_rules3 - DataFrame

Index	antecedents	consequents	ntecedent suppor	onsequent suppo	support	confidence	lift	leverage	conviction
76	frozenset({'Corn', 'Eggs'})	frozenset({'Ice cream'})	0.2	0.2	0.2	1	5	0.16	inf
77	frozenset({'Ice cream'})	frozenset({'Corn', 'Eggs'})	0.2	0.2	0.2	1	5	0.16	inf
93	frozenset({'Onion', 'Corn'})	frozenset({'Ice cream'})	0.2	0.2	0.2	1	5	0.16	inf
96	frozenset({'Ice cream'})	frozenset({'Onion', 'Corn'})	0.2	0.2	0.2	1	5	0.16	inf
111	frozenset({'Corn', 'Milk'})	frozenset({'Unicorn'})	0.2	0.2	0.2	1	5	0.16	inf
114	frozenset({'Unicorn'})	frozenset({'Corn', 'Milk'})	0.2	0.2	0.2	1	5	0.16	inf
124	frozenset({'Yogurt', 'Corn'})	frozenset({'Unicorn'})	0.2	0.2	0.2	1	5	0.16	inf
125	frozenset({'Unicorn'})	frozenset({'Yogurt', 'Corn'})	0.2	0.2	0.2	1	5	0.16	inf
325	frozenset({'Kidney Beans', 'Corn', 'Eggs'})	frozenset({'Ice cream'})	0.2	0.2	0.2	1	5	0.16	inf
326	frozenset({'Kidney Beans', 'Ice cream'})	frozenset({'Corn', 'Eggs'})	0.2	0.2	0.2	1	5	0.16	inf
331	frozenset({'Corn', 'Eggs'})	frozenset({'Kidney Beans', 'Ice cream'})	0.2	0.2	0.2	1	5	0.16	inf
332	frozenset({'Ice cream'})	frozenset({'Kidney Beans', 'Corn', 'Eggs'})	0.2	0.2	0.2	1	5	0.16	inf
338	frozenset({'Onion', 'Corn', 'Eggs'})	frozenset({'Ice cream'})	0.2	0.2	0.2	1	5	0.16	inf
340	frozenset({'Onion', 'Ice cream'})	frozenset({'Corn', 'Eggs'})	0.2	0.2	0.2	1	5	0.16	inf

Apriori algorithm in Python

filter_rules3[filter_rules3['antecedents'] == frozenset(('Corn', 'Eggs'))]

```
     antecedents                         consequents  ...  leverage  conviction
76   (Corn, Eggs)                        (Ice cream)  ...      0.16         inf
331  (Corn, Eggs)           (Ice cream, Kidney Beans)  ...      0.16         inf
345  (Corn, Eggs)                  (Onion, Ice cream)  ...      0.16         inf
779  (Corn, Eggs)  (Onion, Ice cream, Kidney Beans)  ...      0.16         inf
```

filter_rules3[filter_rules3['consequents'] == frozenset(('Corn', 'Eggs'))]

```
                          antecedents   consequents  ...  leverage  convictior
'7                        (Ice cream)  (Corn, Eggs)  ...      0.16         inf
:26         (Ice cream, Kidney Beans)  (Corn, Eggs)  ...      0.16         inf
:40                (Onion, Ice cream)  (Corn, Eggs)  ...      0.16         inf
'66  (Onion, Ice cream, Kidney Beans)  (Corn, Eggs)  ...      0.16         inf
```

Apriori algorithm in Python

filter_rules3[filter_rules3['consequents'] == frozenset({'Ice cream'})]

	antecedents	consequents	...	leverage	conviction
76	(Corn, Eggs)	(Ice cream)	...	0.16	inf
93	(Onion, Corn)	(Ice cream)	...	0.16	inf
325	(Corn, Eggs, Kidney Beans)	(Ice cream)	...	0.16	inf
338	(Onion, Corn, Eggs)	(Ice cream)	...	0.16	inf
362	(Onion, Corn, Kidney Beans)	(Ice cream)	...	0.16	inf
760	(Onion, Corn, Eggs, Kidney Beans)	(Ice cream)	...	0.16	inf

rules3[rules3['consequents'] == frozenset({'Ice cream'})]

	antecedents	consequents	...	leverage	conviction
7	(Corn)	(Ice cream)	...	0.12	1.600000
23	(Eggs)	(Ice cream)	...	0.04	1.066667
31	(Kidney Beans)	(Ice cream)	...	0.00	1.000000
32	(Onion)	(Ice cream)	...	0.08	1.200000
76	(Corn, Eggs)	(Ice cream)	...	0.16	inf
88	(Corn, Kidney Beans)	(Ice cream)	...	0.12	1.600000
93	(Onion, Corn)	(Ice cream)	...	0.16	inf
190	(Eggs, Kidney Beans)	(Ice cream)	...	0.04	1.066667
195	(Onion, Eggs)	(Ice cream)	...	0.08	1.200000
241	(Onion, Kidney Beans)	(Ice cream)	...	0.08	1.200000
325	(Corn, Eggs, Kidney Beans)	(Ice cream)	...	0.16	inf
338	(Onion, Corn, Eggs)	(Ice cream)	...	0.16	inf
362	(Onion, Corn, Kidney Beans)	(Ice cream)	...	0.16	inf
572	(Onion, Eggs, Kidney Beans)	(Ice cream)	...	0.08	1.200000
760	(Onion, Corn, Eggs, Kidney Beans)	(Ice cream)	...	0.16	inf

Explore prescription data

CrossMark

Evaluation of rational nonsteroidal anti-inflammatory drugs and gastro-protective agents use; association rule data mining using outpatient prescription patterns

Oraluck Pattanaprateep[1*], Mark McEvoy[2], John Attia[2] and Ammarin Thakkinstian[1]

Abstract

Background: Nonsteroidal anti-inflammatory drugs (NSAIDs) and gastro-protective agents should be co-prescribed following a standard clinical practice guideline; however, adherence to this guideline in routine practice is unknown. This study applied an association rule model (ARM) to estimate rational NSAIDs and gastro-protective agents use in an outpatient prescriptions dataset.

Methods: A database of hospital outpatients from October 1st, 2013 to September 30th, 2015 was searched for any of following drugs: oral antacids (A02A), peptic ulcer and gastro-oesophageal reflux disease drugs (GORD, A02B), and anti-inflammatory and anti-rheumatic products, non-steroids or NSAIDs (M01A). Data including patient demographics, diagnoses, and drug utilization were also retrieved. An association rule model was used to analyze co-prescription of the same drug class (i.e., prescriptions within A02A-A02B, M01A) and between drug classes (A02A-A02B & M01A) using the Apriori algorithm in R. The lift value, was calculated by a ratio of confidence to expected confidence, which gave information about the association between drugs in the prescription.

Results: We identified a total of 404,273 patients with 2,575,331 outpatient visits in 2 fiscal years. Mean age was 48 years and 34% were male. Among A02A, A02B and M01A drug classes, 12 rules of associations were discovered with support and confidence thresholds of 1% and 50%. The highest lift was between Omeprazole and Ranitidine (340 visits); about one-third of these visits (118) were prescriptions to non-GORD patients, contrary to guidelines. Another

Explore prescription data

Raw data

HNDate	DrugCode
AA20141001	OMPZ
AA20141001	XAND

Input data

HNDate	DrugCode1	DrugCode2
AA20141001	OMPZ	XAND

Explore prescription data

Input data

HNDate	DrugCode1	DrugCode2
AA20141001	OMPZ	XAND

Apriori algorithm (Association)

Association rules

Explore prescription data

prescriptionDB.csv

Item ID	Code.1	Code.2	Code.3	Code.4
1	ALGY	PARI	NA	NA
2	ALGY	PARI	NA	NA
3	ALGY	PRVF	NA	NA
4	ALGY	PARI	NA	NA
5	ALGY	PARI	NA	NA
6	ALGY	DEXI	NA	NA
7	ALGY	ARCX	NA	NA
8	ALGY	PARI	NA	NA
9	ALGY	PARI	NA	NA
10	ALGY	NEXM	NA	NA
11	ALGY	XAND	NA	NA
12	ALGY	OMPZ	NA	NA
13	ALGY	OMPZ	NA	NA
14	ALGY	PARI	XAND	NA
15	ALGY	PARI	NA	NA

Reference

[1] Rakesh A., Tomasz I. and Arun S. (1993). "Mining association rules between sets of items in large databases". Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93. p. 207.

[2] Rakesh Agrawal and Ramakrishnan Srikant (1994). “Fast algorithms for mining association rules”. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile.

Assignment: due date November 21, 2022

a. (4 points) From the below transaction database, generate the frequent itemset and association rules.

Student#1: use a minimum support threshold = 50%

Transaction ID	Items
1	onions, potatoes, burger, cereal
2	potato chip, burger, beer
3	onions, potatoes, potato chip, burger, beer
4	potato chip, beer, onions
5	eggs, cereal, potato chip

Assignment:

Student#2: use a minimum support threshold = 40%

Transaction ID	Items
1	onions, potatoes, burger, cereal
2	potato chip, burger, beer, apple
3	onions, potatoes, potato chip, burger, beer
4	potato chip, beer, onions
5	eggs, cereal, potato chip, apple

Assignment:

Student#3: use a minimum support threshold = 50%

Transaction ID	Items
1	onions, potatoes, burger, cereal
2	potato chip, burger, beer
3	onions, potatoes, potato chip, burger, beer
4	eggs, onions, potatoes, cereal, potato chip, burger
5	potato chip, beer, onions

Assignment:

Student#4: use a minimum support threshold = 50%

Transaction ID	Items
1	onions, potatoes, burger, cereal
2	potato chip, burger, beer, apple
3	onions, potatoes, potato chip, burger, beer
4	eggs, onions, potatoes, cereal, potato chip, burger
5	potato chip, beer, onions

Assignment:

Student#5: use a minimum support threshold = 50%

Transaction ID	Items
1	onions, eggs, burger, cereal
2	potato chip, potatoes, beer
3	onions, potatoes, potato chip, burger, beer
4	potato chip, beer, onions, cereal
5	eggs, cereal, potato chip

Assignment:

Student#6: use a minimum support threshold = 50%

Transaction ID	Items
1	onions, potatoes, burger, cereal
2	potato chip, burger, beer, eggs
3	onions, potatoes, potato chip, burger, beer
4	potato chip, beer, onions
5	eggs, cereal, potato chip

Assignment:

b. (3 points)

- From prescriptionDB.csv
- Perform an Apriori algorithm to generate the association rules by using the follow conditions
- Student#1

  support=0.001, confidence=0.5, and find top 10 of the

  RHS="OMPZ"
- Student#2

  support=0.0001, confidence=0.4, and find top 20 of the

  RHS="OMPZ"

Assignment:

- Student#3

  support=0.0001, confidence=0.30, and find the rule of RHS=“XAND”

- Student#4

  support=0.01, confidence=0.50, and find the rule of is LHS=“ANTC” or “ARCX” or “XAND”

- Student#5

  support=0.001, confidence=0.20, and find the rule of LHS=“ALGY” or “ANTT” or “CAPN”

Assignment:

- Student#6

  support=0.0001, confidence=0.30, and find the rule of RHS="ALGY" or "ASA."

Assignment:

c. (3 points)

- From prescriptionDB.csv
- Perform an Apriori algorithm to generate the association rules by selecting the top 20 rules at support=0.0001 and the LHS has at least two drugs in the basket.