Associate Rules – Apriori in Python

Chun-Hao Chen (陳俊豪)
chchen@ntut.edu.tw
TaipeiTech

Mining Problem

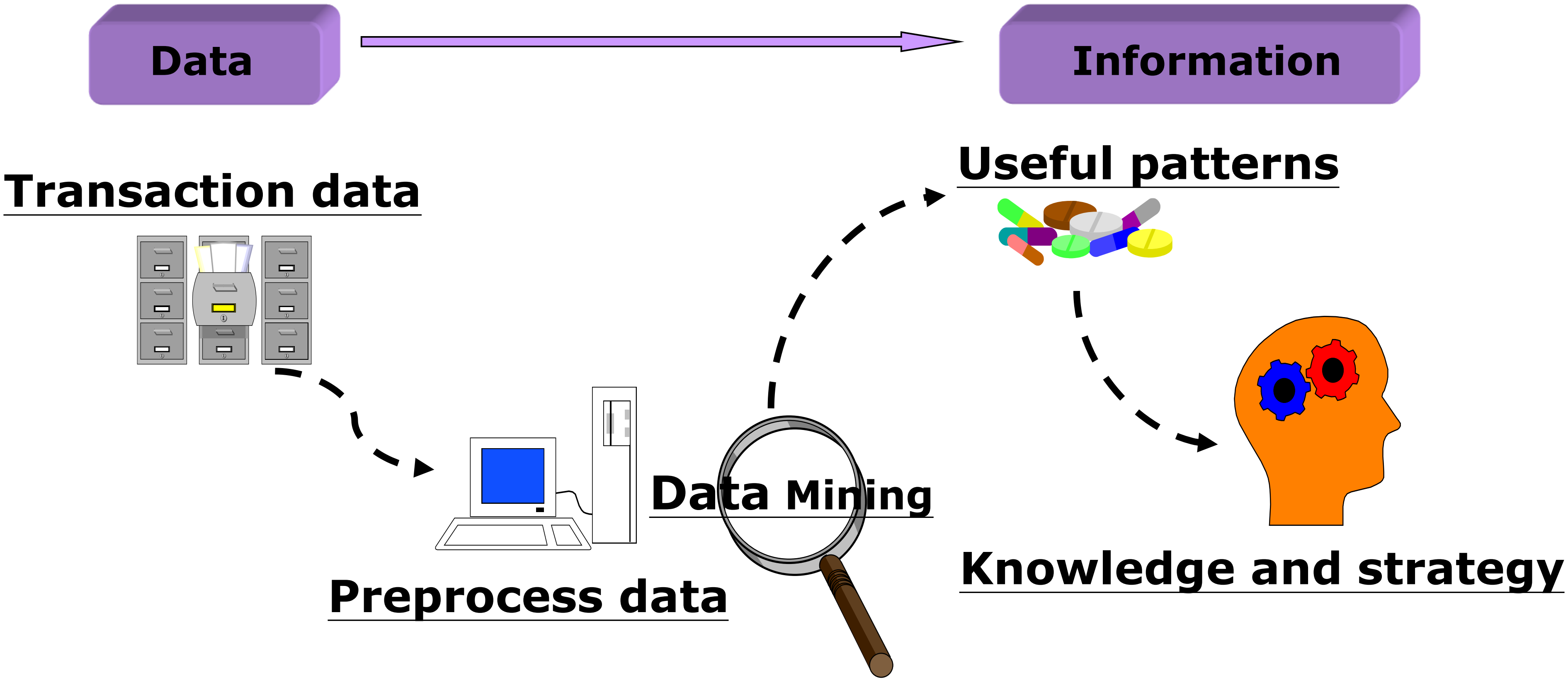
Goods

How to arrange goods into supermarket?

Manager

Supermarket

The Process of Data Mining



Data

Information

Transaction data

Preprocess data

Data Mining

Useful patterns

Knowledge and strategy

Apriori Algorithm

- Proposed by Agrawal et al. in 1994

- Step 1: Define *minsup* and *minconf*
  - Example
    - *minsup* = 50%
    - *minconf* = 50%

- Step 2: Find frequent itemsets with *minsup*

- Step 3: Generate association rules with *minconf*

Example

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Scan Database →

C_1

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

Large itemsets

L_1

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A B}
{A C}
{A E}
{B C}
{B E}
{C E}

Scan Database →

C_2

Itemset	Sup.
{A B}	1
{A C}	2
{A E}	1
{B C}	2
{B E}	3
{C E}	2

L_2

Itemset	Sup.
{A C}	2
{B C}	2
{B E}	3
{C E}	2

C_3

Itemset
{B C E}

Scan Database →

C_3

Itemset	Sup.
{B C E}	2

L_3

Itemset	Sup.
{B C E}	2

Example

Association rules	Confidence
IF BC THEN E	S(BCE)/S(BC)=2/2
IF BE THEN C	S(BCE)/S(BE)=2/3
IF CE THEN B	S(BCE)/S(CE)=2/2
IF B THEN CE	S(BCE)/S(B)=2/3
IF C THEN BE	S(BCE)/S(C)=2/3
IF E THEN BC	S(BCE)/S(E)=2/3
IF A THEN C	S(AC)/S(A)=2/2
IF C THEN A	S(AC)/S(C)=2/3
IF B THEN C	S(BC)/S(B)=2/3
IF C THEN B	S(BC)/S(C)=2/3
IF B THEN E	S(BE)/S(B)=3/3
IF E THEN B	S(BE)/S(E)=3/3
IF C THEN E	S(CE)/S(C)=2/3
IF E THEN C	S(CE)/S(E)=2/3

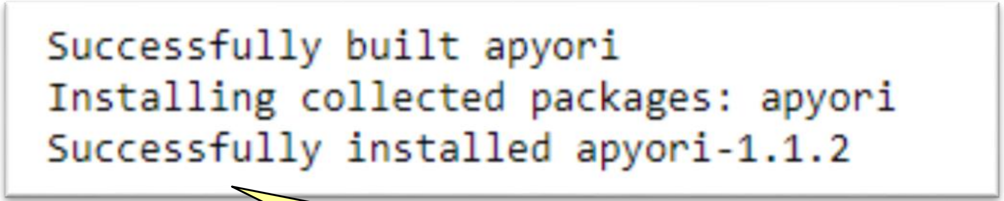
Code - Outline

1. Import library and read dataset (讀取套件以及讀檔)
2. Data Preprocessing (檔案前處理)
3. Uinsg Apriori module (使用Apriori套件)
4. Show results (顯示結果)
5. Observations (解讀結果)

Install apyori in Jupyter

```
!pip install apyori
```

Type this statement to install apyori

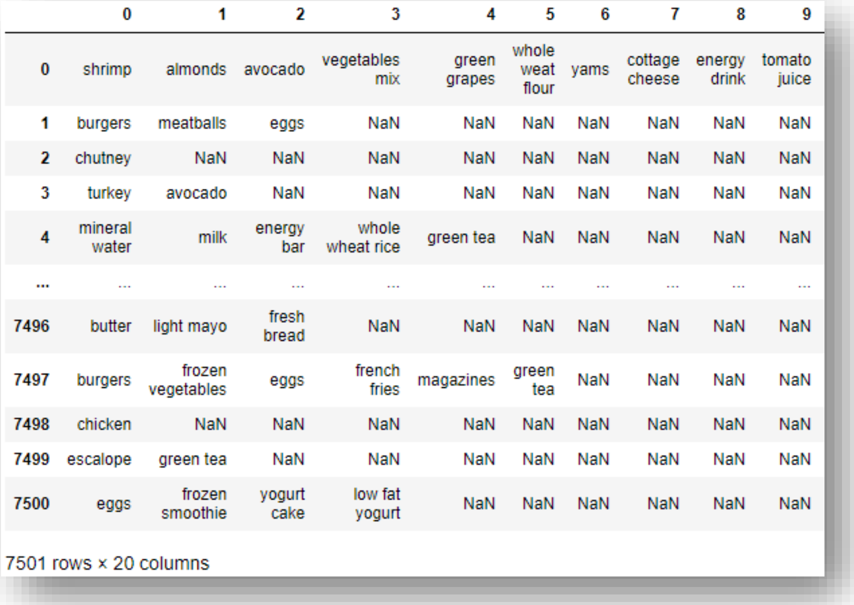


```
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

See ‘successfully’ means install correctly

Dataset - Retail Store

- 7500 transactions over the course of a week at a French retail store
- Number of items: 118
- Dataset source: https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/



	0	1	2	3	4	5	6	7	8	9
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN

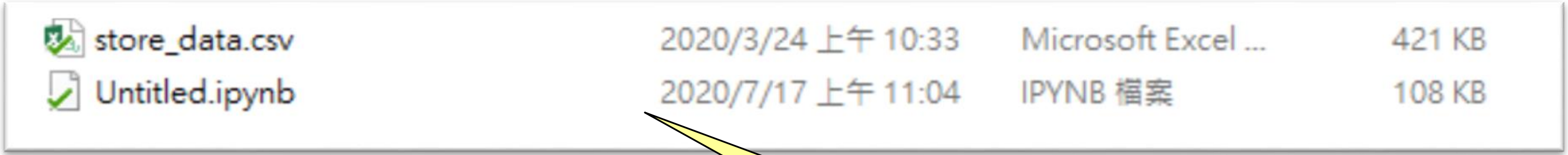
7501 rows × 20 columns

Part of transactions

Import Library and Read Dataset

```
import pandas as pd
from apyori import apriori
store_data = pd.read_csv("store_data.csv"
                                            ,header=None)
```

File name or path

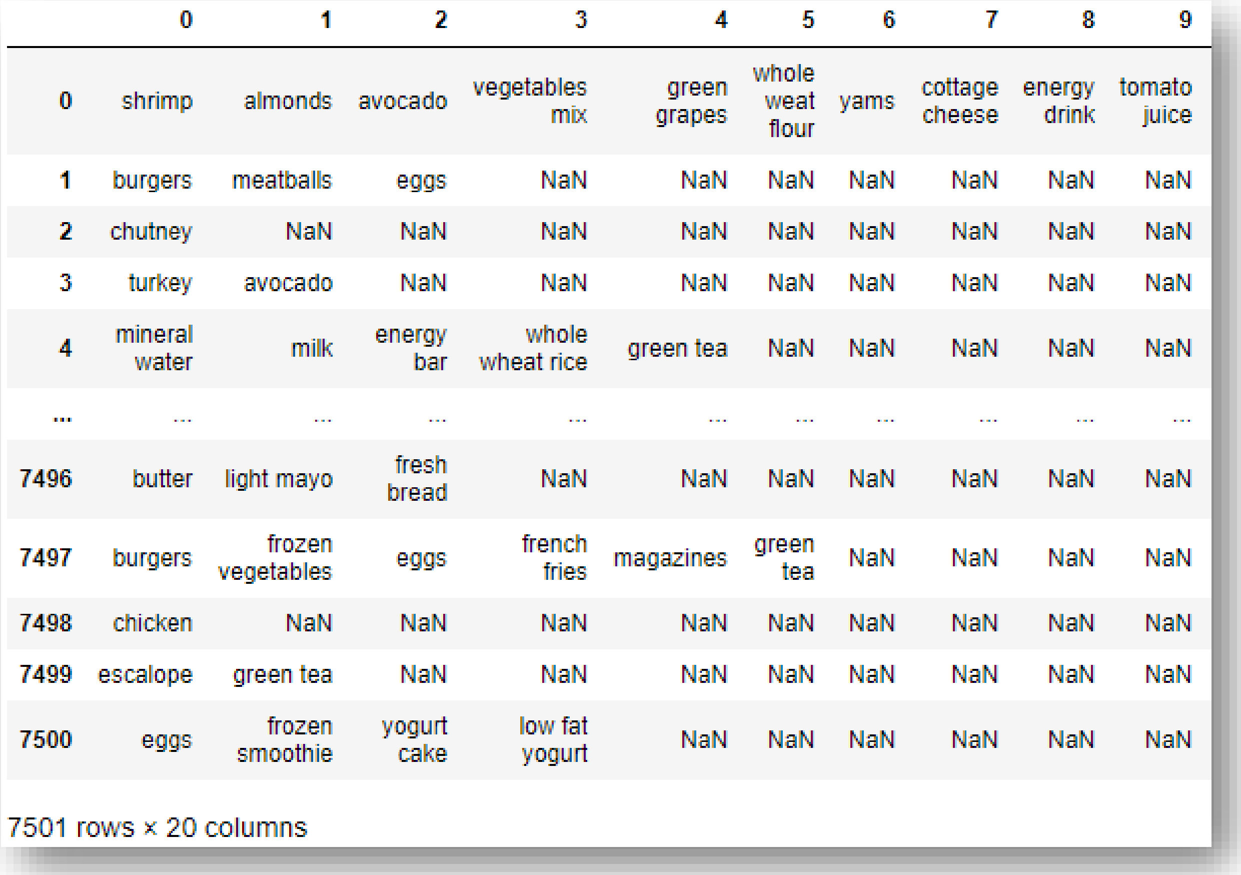


```
store_data.csv        2020/3/24 上午 10:33    Microsoft Excel ...    421 KB
Untitled.ipynb        2020/7/17 上午 11:04    IPYNB 檔案             108 KB
```

Put the dataset and python code in the same folder

Data Preprocessing

- Remove ‘NaN’ values



	0	1	2	3	4	5	6	7	8	9
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN

7501 rows × 20 columns

Data Preprocessing (Code)

```
records = []
for i in range(0, store_data.shape[0]):
    tmp = []

    for j in range(0, store_data.shape[1]):
        if str(store_data.values[i, j]) != 'nan':
            tmp.append([str(store_data.values[i, j])])
        else:
            break

    records.append([str(tmp[k][0]) for k in range(0, len(tmp))])
    tmp = []
```

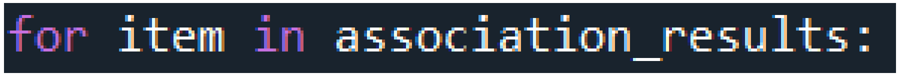
Using Apriori Module

- Call function apriori()
- Parameters
  - dataset
  - min_support
  - min_confidence
  - min_lift
- Transform results into list format

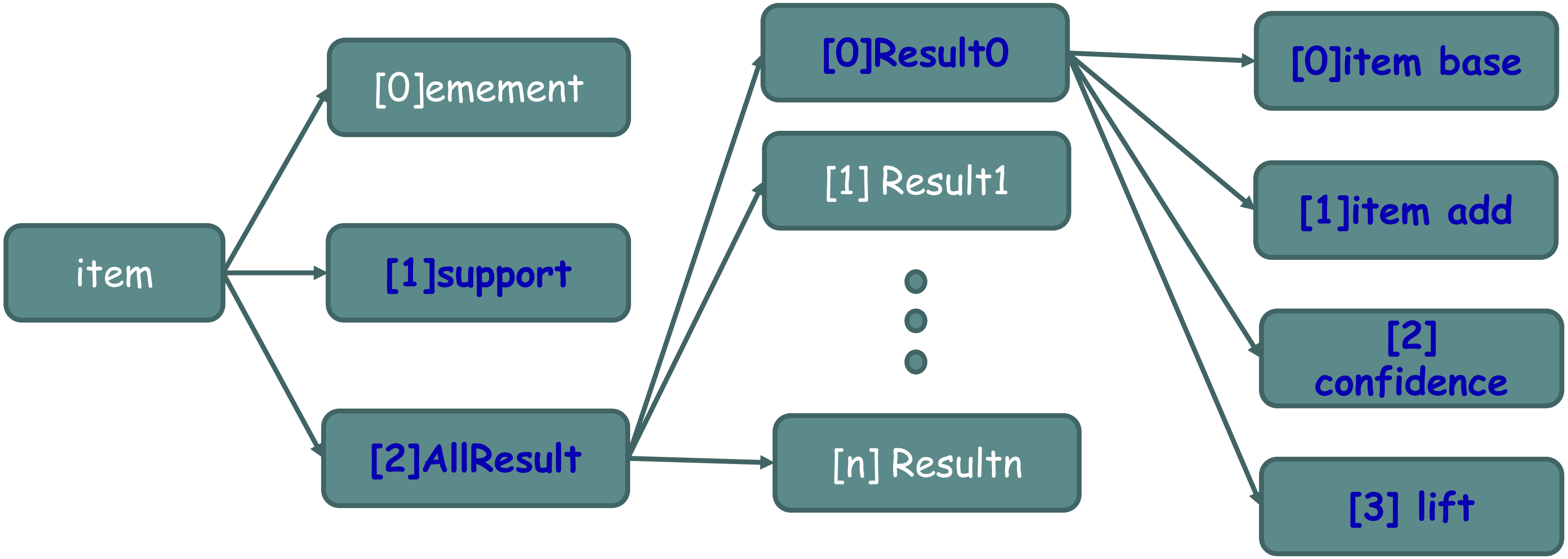
```
association_rules = apriori(records, min_support=0.005,
   min_confidence=0.2, min_lift=3)
association_results = list(association_rules)
```

Show Results

- Use for to show results



```
for item in association_results:
```



item

[0]emement

[1]support

[2]AllResult

[0]Result0

[1] Result1

[n] Resultn

[0]item base

[1]item add

[2] confidence

[3] lift

Show Results

```
RelationRecord(items=frozenset({'mushroom cream sauce', 'escalope'}),
support=0.005732568990801226,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'mushroom cream sauce'}),
items_add=frozenset({'escalope'}),
confidence=0.3006993006993007,
lift=3.790832696715049)])

Rule: (mushroom cream sauce) -> (escalope)
Length: 2
Support: 0.005732568990801226
Confidence: 0.3006993006993007
Lift: 3.790832696715049
```

Show Results (Code 1)

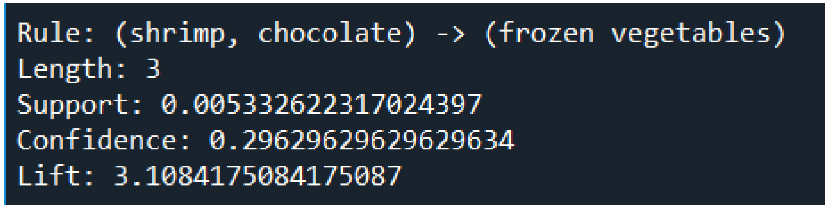
```
count = 0

for item in association_results:
    #[2][0][0]=>item base
    for item_num in range(0,len(item[2])):
        if item[2][item_num][0] == frozenset():
            continue
        else:
            pairBase = item[2][item_num][0]
            items = [x for x in pairBase]
            r="Rule: ("
            for x in range(0,len(items)):
                if x==0 :
                    r=r+items[x]
                else:
                    r=r+", "+items[x]
            r=r+") -> "
            #[2][0][1]=>item add
            pairAdd = item[2][item_num][1]
            items = [x for x in pairAdd]
```

Show Results (Code 2)

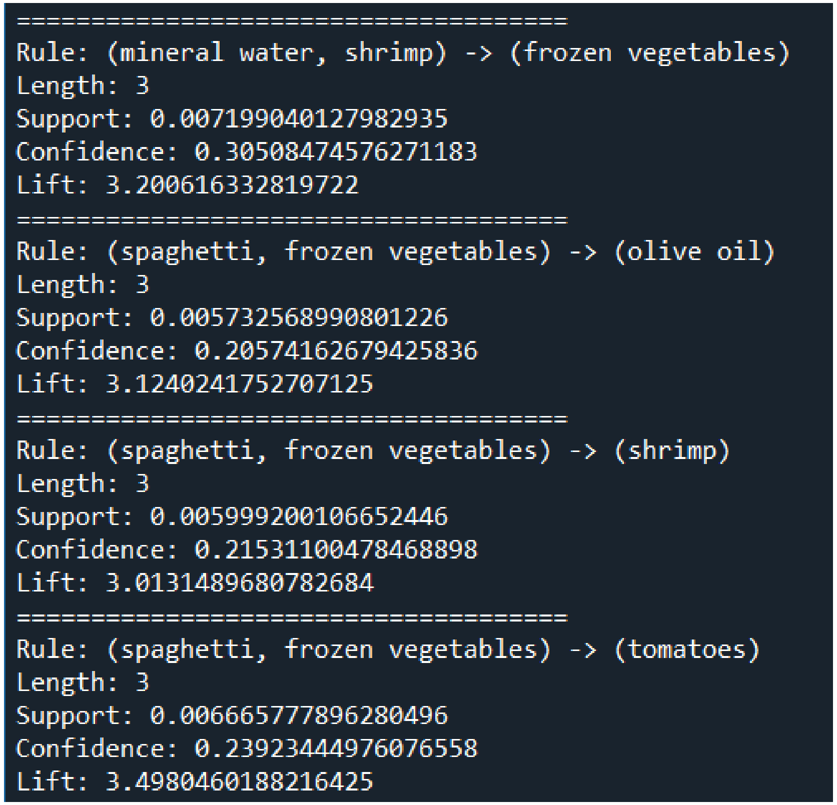
```
for x in range(0, len(items)):
    if x==0 :
        r=r+"("+items[x]
    else:
        r=r+", "+items[x]
r=r+")"

#print rule
print(r)
#[0] => all items in the rule
print("Length: "+str(len(item[0])))
#[1] => support
print("Support: " + str(item[1]))
#[2][0][2] => confidence
print("Confidence: " + str(item[2][item_num][2]))
#[2][0][3] => lift
print("Lift: " + str(item[2][item_num][3]))
print("=====================================")
count=count+1
```

Observations

- Generate 21 rules
  - Length 2: 6 rules
  - Length 3: 15 rules
- Rules are related to food
- Frequency of spaghetti and frozen vegetables is the largest
- Interesting rule:



```
Rule: (shrimp, chocolate) -> (frozen vegetables)
Length: 3
Support: 0.005332622317024397
Confidence: 0.29629629629629634
Lift: 3.1084175084175087
```

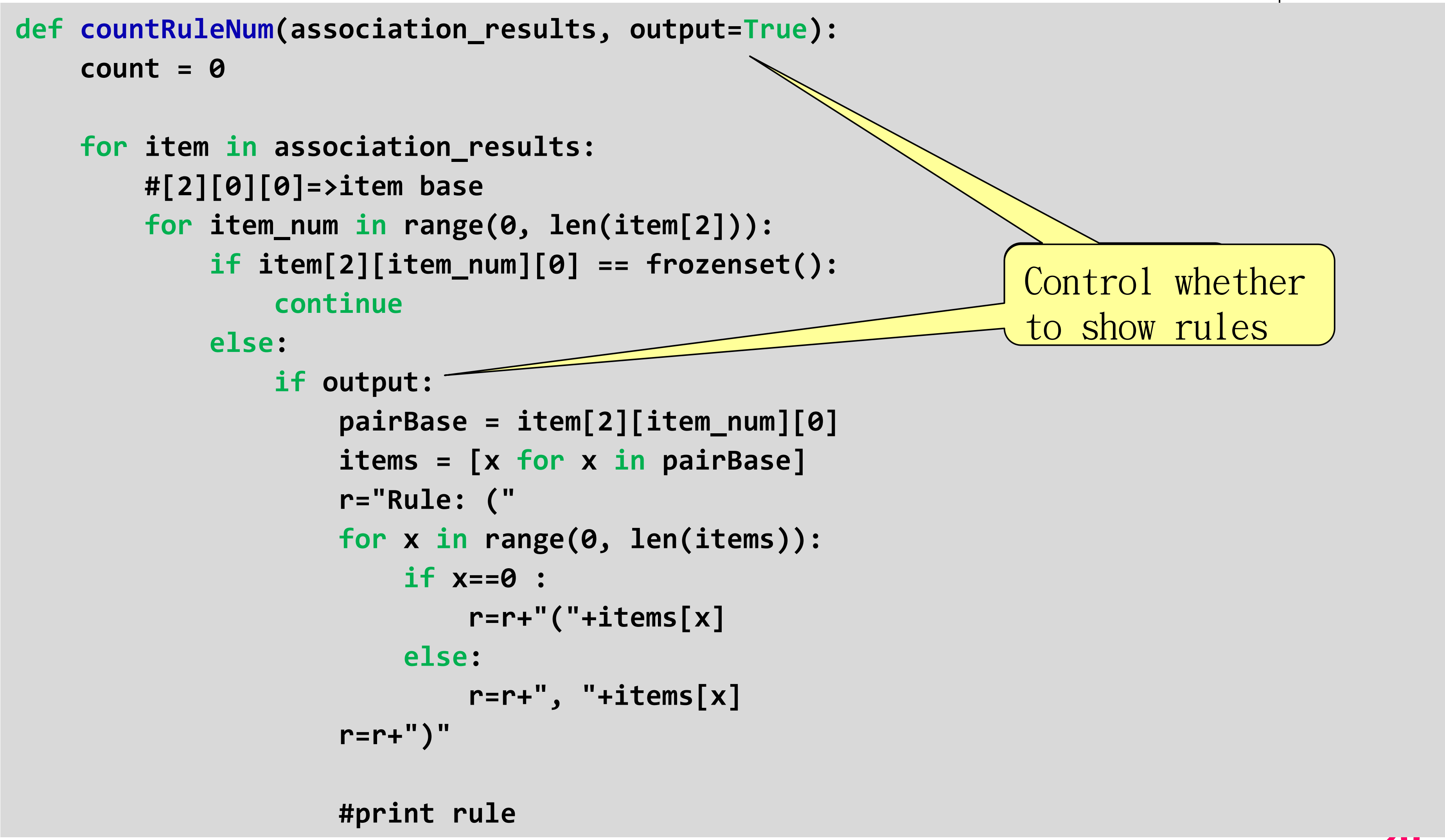


```
=====================================
Rule: (mineral water, shrimp) -> (frozen vegetables)
Length: 3
Support: 0.007199040127982935
Confidence: 0.30508474576271183
Lift: 3.200616332819722
=====================================
Rule: (spaghetti, frozen vegetables) -> (olive oil)
Length: 3
Support: 0.005732568990801226
Confidence: 0.20574162679425836
Lift: 3.1240241752707125
=====================================
Rule: (spaghetti, frozen vegetables) -> (shrimp)
Length: 3
Support: 0.005999200106652446
Confidence: 0.21531100478468898
Lift: 3.0131489680782684
=====================================
Rule: (spaghetti, frozen vegetables) -> (tomatoes)
Length: 3
Support: 0.006665777896280496
Confidence: 0.23923444976076558
Lift: 3.4980460188216425
```

Relationship between #Rules and Different Minimum Supports

- Min. Support: from 0.001 to 0.01
  - Increase by 0.001 each run
- Prepare a function '**`countRuleNum`**' to count rules

The countRuleNum() (Code 1)



```
def countRuleNum(association_results, output=True):
    count = 0

    for item in association_results:
        #[2][0][0]=>item base
        for item_num in range(0, len(item[2])):
            if item[2][item_num][0] == frozenset():
                continue
            else:
                if output:
                    pairBase = item[2][item_num][0]
                    items = [x for x in pairBase]
                    r="Rule: ("
                    for x in range(0, len(items)):
                        if x==0 :
                            r=r+"("+items[x]
                        else:
                            r=r+", "+items[x]
                    r=r+")"

                    #print rule
```

Control whether to show rules

The countRuleNum() (Code 2)

```
                    print(r)
                    #[0] => all items in the rule
                    print("Length: "+str(len(item[0])))
                    #[1] => support
                    print("Support: " + str(item[1]))
                    #[2][0][2] => confidence
                    print("Confidence: " + str(item[2][item_num][2]))
                    #[2][0][3] => lift
                    print("Lift: " + str(item[2][item_num][3]))
                    print("=====================================")
                count=count+1

    return count
```

Show Final Results

```python
import numpy as np
import matplotlib.pyplot as plt


x = []
y = []
for i in np.arange(0.001, 0.01+0.001, 0.001):
    association_rules = apriori(records, min_support=i, min_confidence=0.2, min_lift=3)
    association_results = list(association_rules)

    x.append("{:.3f}".format(i))
    y.append(countRuleNum(association_results, output=False))

print(x)
print(y)
```

Increase by 0.001

將X設定成字串格式，以便後續重新命名X軸的數值

```
Minsup = ['0.001', '0.002', '0.003', '0.004', '0.005', '0.006', '0.007', '0.008', '0.009', '0.010’]
# Rules = [2814,    347,     130,     36,      21,      9,       5,       2,       1,       1]
```

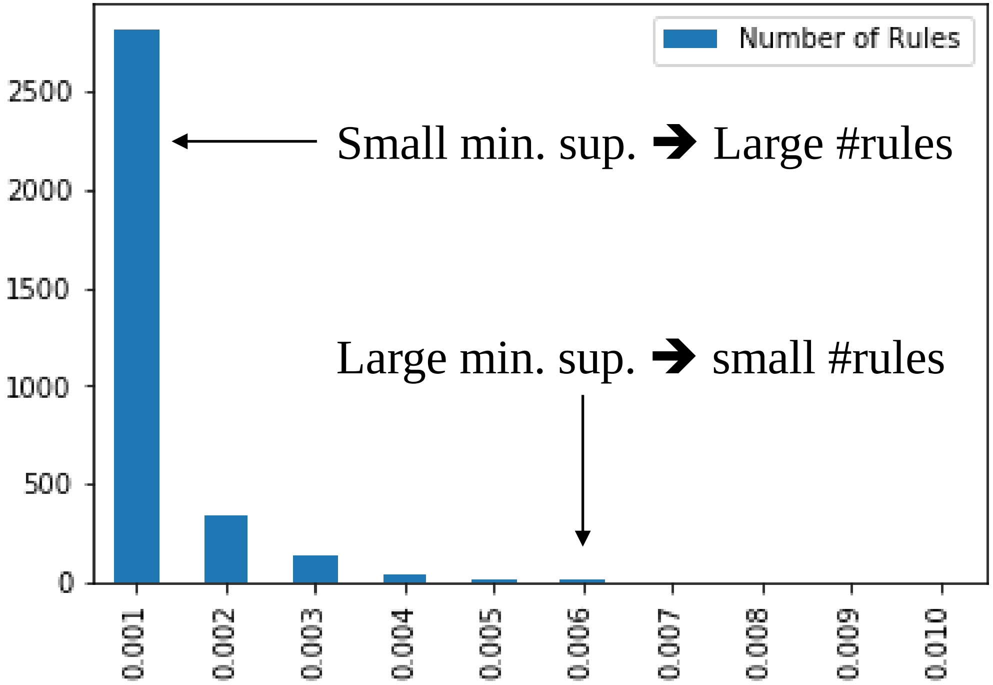
Use Pandans to Draw A Figure

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

x = []
y = []
for i in np.arange(0.001, 0.01+0.001, 0.001):
    association_rules = apriori(records, min_support=i, min_confidence=0.2, min_lift=3)
    association_results = list(association_rules)

    x.append("{:.3f}".format(i))
    y.append(countRuleNum(association_results, output=False))

df = pd.DataFrame(y,columns=['Number of Rules'], index = x)

df.plot(kind='bar')
plt.show()
```

Increase by 0.001

Results

- Relationship Between Minimum Support & Number of Rules



Small min. sup. ➔ Large #rules

Large min. sup. ➔ small #rules