Practical #10: Association Analysis in Python

Objective:

- Learn how to do association analysis using Python.

Submission:

- *What to submit*: The saved document (**.html**) containing the required steps done in this practical.
- *Deadline of the submission*: 11:59PM (Adelaide Time), Tuesday of **Week 13**.
- *Submission link*: "**Submission Link of Prac #10"** in **Week 12 section** on Learnonline course site.
- *Marks*: Prac#10 (part of the ongoing assessment of the course) is worth 2% of the total marks of the course.

Association Analysis in Python:

Association analysis is a rule-based unsupervised learning method for discovering interesting relations between variables in databases. The association rules can be determined by 3 parameters (support, confidence, and lift) that are used to identify the algorithm's strength. The **Apriori** algorithm is the most popular association rules approach, which can help us to find strong relationships between different items/products in the industry, such as retail industry data in this practical ("**Prac#10-GroceryStore.csv**").

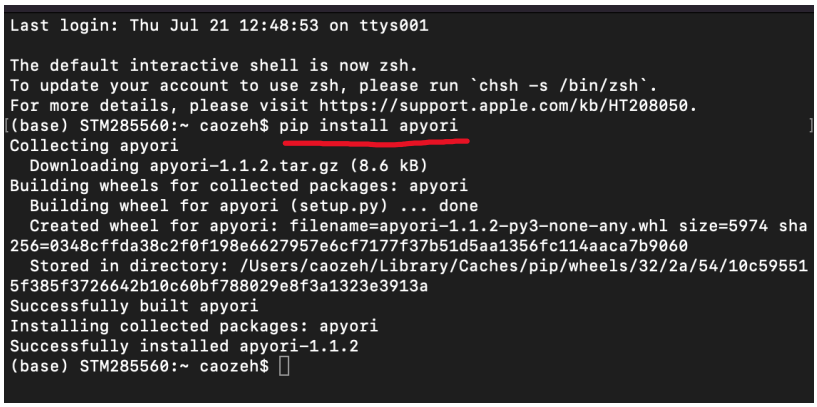
The first step, as always, is to import the required Python libraries. Please execute the following script to do so:

```
import pandas as pd
import numpy as np
from apyori import apriori
from mlxtend.frequent_patterns import apriori, association_rules
```

Note: You must download and install **apyori** library in the default path for your Python libraries before proceeding.

Open the terminal and type in the following command, as shown in the screenshot below,

pip install apyori



Now, we can import the dataset ("**Prac#10-GroceryStore.csv**") and overview of how it is structured.

```
df = pd.read_csv("Prac#10-GroceryStore.csv", names = ['products'], sep = ',')
df.head()
```

	products
0	MILK,BREAD,BISCUIT
1	BREAD,MILK,BISCUIT,CORNFLAKES
2	BREAD,TEA,BOURNVITA
3	JAM,MAGGI,BREAD,MILK
4	MAGGI,TEA,BISCUIT

Note: you need to download the dataset ("**Prac#10-GroceryStore.csv**") from Week 12 section on Learnonline course site and upload it to the Jupyter Notebook home folder.

Next, let us transform the dataset, split the items, and create a list.

```
df.shape
data = list(df["products"].apply(lambda x:x.split(",") ))
data
```

```
[['MILK', 'BREAD', 'BISCUIT'],
 ['BREAD', 'MILK', 'BISCUIT', 'CORNFLAKES'],
 ['BREAD', 'TEA', 'BOURNVITA'],
 ['JAM', 'MAGGI', 'BREAD', 'MILK'],
 ['MAGGI', 'TEA', 'BISCUIT'],
 ['BREAD', 'TEA', 'BOURNVITA'],
 ['MAGGI', 'TEA', 'CORNFLAKES'],
 ['MAGGI', 'BREAD', 'TEA', 'BISCUIT'],
 ['JAM', 'MAGGI', 'BREAD', 'TEA'],
 ['BREAD', 'MILK'],
 ['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
 ['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
 ['COFFEE', 'SUGER', 'BOURNVITA'],
 ['BREAD', 'COFFEE', 'COCK'],
 ['BREAD', 'SUGER', 'BISCUIT'],
 ['COFFEE', 'SUGER', 'CORNFLAKES'],
 ['BREAD', 'SUGER', 'BOURNVITA'],
 ['BREAD', 'COFFEE', 'SUGER'],
 ['BREAD', 'COFFEE', 'SUGER'],
 ['TEA', 'MILK', 'COFFEE', 'CORNFLAKES']]
```

After reviewing the list items, we need to import **TransactionEncoder** that can convert the list to a One-Hot Encoded Boolean list, which is convenient for Apriori's algorithm to transform as True or False (0) values – see the next page.

```
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
```

	BISCUIT	BOURNVITA	BREAD	COCK	COFFEE	CORNFLAKES	JAM	MAGGI	MILK	SUGER	TEA
0	True	0	True	0	0	0	0	0	True	0	0
1	True	0	True	0	0	True	0	0	True	0	0
2	0	True	True	0	0	0	0	0	0	0	True
3	0	0	True	0	0	0	True	True	True	0	0
4	True	0	0	0	0	0	0	True	0	0	True
5	0	True	True	0	0	0	0	0	0	0	True
6	0	0	0	0	0	True	0	True	0	0	True
7	True	0	True	0	0	0	0	True	0	0	True
8	0	0	True	0	0	0	True	True	0	0	True
9	0	0	True	0	0	0	0	0	True	0	0
10	True	0	0	True	True	True	0	0	0	0	0
11	True	0	0	True	True	True	0	0	0	0	0
12	0	True	0	0	True	0	0	0	0	True	0
13	0	0	True	True	True	0	0	0	0	0	0
14	True	0	True	0	0	0	0	0	0	True	0
15	0	0	0	0	True	True	0	0	0	True	0
16	0	True	True	0	0	0	0	0	0	True	0
17	0	0	True	0	True	0	0	0	0	True	0
18	0	0	True	0	True	0	0	0	0	True	0
19	0	0	0	0	True	True	0	0	True	0	True

The next step is to create the **Apriori Model**. We can change all the parameters in the Apriori Model in the mlxtend package (Note: there may exist some warning messages in Python due to different version upgrading issues). I will try to use minimum support parameters for this modelling - For instance, we can set a *min_support* value with a threshold value of 20% and print them on the screen as well.

```
df = apriori(df, min_support = 0.2, use_colnames = True, verbose = 1)
df
```

```
Processing 42 combinations | Sampling itemset size 3
/Users/caozeh/opt/anaconda3/lib/python3.9/site-packages/mlxtend/frequent_
aFrames with non-bool types result in worse computationalperformance and
.Please use a DataFrame with bool type
  warnings.warn(
```

	support	itemsets
0	0.35	(BISCUIT)
1	0.2	(BOURNVITA)
2	0.65	(BREAD)
3	0.4	(COFFEE)
4	0.3	(CORNFLAKES)
5	0.25	(MAGGI)
6	0.25	(MILK)
7	0.3	(SUGER)
8	0.35	(TEA)
9	0.2	(BREAD, BISCUIT)
10	0.2	(BREAD, MILK)
11	0.2	(BREAD, SUGER)
12	0.2	(BREAD, TEA)
13	0.2	(CORNFLAKES, COFFEE)
14	0.2	(COFFEE, SUGER)
15	0.2	(MAGGI, TEA)

The final step is that we can choose the 60% minimum confidence value. In other words, when product X is purchased, we can say that the purchase of product Y is 60% or more.

```
df_ar = association_rules(df, metric = "confidence", min_threshold = 0.6)
df_ar
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(MILK)	(BREAD)	0.25	0.65	0.2	0.800000	1.230769	0.0375	1.75
1	(SUGER)	(BREAD)	0.30	0.65	0.2	0.666667	1.025641	0.0050	1.05
2	(CORNFLAKES)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.80
3	(SUGER)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.80
4	(MAGGI)	(TEA)	0.25	0.35	0.2	0.800000	2.285714	0.1125	3.25

Save the .html document as your submission by following the instruction given on page 1 of the practical document.