tp_ra_mf_exo1_correction

November 24, 2023

1 TP Règles d'associations et motifs fréquents (correction)

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

1.1 Exercice 1. Analyse du ticket de caisse

1.1.1 1. Chargement des données

Les données que nous allons utiliser sont déjà préparées (fichier "market-basket.csv").

Chaque ligne correspond à une transaction (i.e., un panier/ticket) et chaque colonne à un item (i.e., un produit). Un "1" (ou "True") code la présence d'un item (i.e., produit acheté), un "0" ou "False" son absence (i.e., produit non acheté).

[2]:
```
dataset = pd.read_csv('market_basket.csv', header=0)
print(dataset.shape)
print(dataset.columns)
```

```
(1360, 303)
Index(['100_Watt_Lightbulb', '2pct_Milk', '40_Watt_Lightbulb',
       '60_Watt_Lightbulb', '75_Watt_Lightbulb', '98pct_Fat_Free_Hamburger',
       'AA_Cell_Batteries', 'Apple_Cinnamon_Waffles', 'Apple_Drink',
       'Apple_Fruit_Roll',
       …
       'White_Bread', 'White_Wine', 'White_Zinfandel_Wine', 'Whole_Corn',
       'Whole_Green_Beans', 'Whole_Milk', 'Window_Cleaner', 'Wood_Polish',
       'flav_Fruit_Bars', 'flav_Ice'],
      dtype='object', length=303)
```

[3]:
```
# transformation en valeurs booléennes (étape non obligatoire mais recommandée␣
 ↪par MLxtend)
for col in dataset.columns:
    dataset[col] = dataset[col].map({1 : True, 0 : False})
    dataset[col].astype("boolean")
```

1.1.2 2. Extraction des motifs fréquents

Pour extraire les motifs fréquents, nous utilisons apriori() sur les données chargées précédemment, avec un support minimum (min_support) fixé à 0,025 (i.e., 2,5%, soit 34 transactions), une cardinalité max (max_len) pour un motif fixée à 4 items et une utilisation des noms d'item (use_colnames) à vrai.

Nous remarquons que les résultats (l'ensemble des motifs fréquents) sont stockés dans un DataFrame et qu'un motif est stockée dans une Serie.

Pour chaque motif, nous avons la valeur de son support en relatif (%) et les items qu'il contient.

Notons que les items sont stockés dans un frozenset, i.e., un ensemble non modifiable (non-mutable).

[4]:
```
taillemax = 4
freq_itemsets = apriori(dataset, min_support=0.025, max_len=taillemax,␣
 ↪use_colnames=True, verbose=0)
```

[5]:
```
# type du résultat
type(freq_itemsets)
```

[5]: pandas.core.frame.DataFrame

[6]:
```
# nombre de motifs obtenus
print(freq_itemsets.shape)
```

```
(603, 2)
```

[7]:
```
# liste des colonnes
print(freq_itemsets.columns)
```

```
Index(['support', 'itemsets'], dtype='object')
```

[8]:
```
# affichage des 15 premiers motifs
print(freq_itemsets.head(15))
```

```
     support                    itemsets
0   0.030147        (100_Watt_Lightbulb)
1   0.109559                 (2pct_Milk)
2   0.037500         (60_Watt_Lightbulb)
3   0.031618         (75_Watt_Lightbulb)
4   0.093382  (98pct_Fat_Free_Hamburger)
5   0.031618         (AA_Cell_Batteries)
6   0.025735    (Apple_Cinnamon_Waffles)
7   0.026471               (Apple_Drink)
8   0.031618          (Apple_Fruit_Roll)
9   0.032353                 (Apple_Jam)
10  0.033088               (Apple_Jelly)
11  0.032353               (Apple_Sauce)
12  0.053676                    (Apples)
```

```
13  0.066912                   (Aspirin)
14  0.027941                (Avocado_Dip)
```

[9]:
```
# type de la colonne 'itemsets'
print(type(freq_itemsets.itemsets))
```

```
<class 'pandas.core.series.Series'>
```

[10]:
```
# affichage du premier élément
print(freq_itemsets.itemsets[0])
```

```
frozenset({'100_Watt_Lightbulb'})
```

1.1.3 3. Génération des règles d'association

Afin de produire les règles d'association, il faut utiliser association_rules(). Nous allons choisir la mesure (metric) de la confiance et fixer le seuil minimal (min_threshold) à 0,75 (i.e., 75%).

Les règles obtenues sont stockées dans un DataFrame. Pour chaque règle, nous avons l'antécédent, le conséquent, le support de la règle, la confiance, le lift, le leverage et la conviction. Nous avons aussi le support de l'antécédent et celui du conséquent.

[11]:
```
arules = association_rules(freq_itemsets, metric="confidence", min_threshold=0.
↪7)
```

[12]:
```
# type du résultat
print(type(arules))
```

```
<class 'pandas.core.frame.DataFrame'>
```

[13]:
```
# nombre de règles générées
print(arules.shape)
```

```
(105, 10)
```

[14]:
```
# liste des colonnes
print(arules.columns)
```

```
Index(['antecedents', 'consequents', 'antecedent support',
       'consequent support', 'support', 'confidence', 'lift', 'leverage',
       'conviction', 'zhangs_metric'],
      dtype='object')
```

[15]:
```
# affichage des 5 premières règles du résultat
print(arules.iloc[:5,:])
```

```
                          antecedents     consequents  antecedent support  \
0                       (Hot_Dog_Buns)      (Hot_Dogs)            0.058824
1  (2pct_Milk, 98pct_Fat_Free_Hamburger)          (Eggs)            0.038235
2  (2pct_Milk, 98pct_Fat_Free_Hamburger)  (Potato_Chips)            0.038235
3                  (Aspirin, 2pct_Milk)          (Eggs)            0.034559
4                  (Aspirin, 2pct_Milk)  (Potato_Chips)            0.034559
```

```
   consequent support   support  confidence      lift  leverage  conviction  \
0            0.092647  0.041912    0.712500  7.690476  0.036462    3.156010
1            0.122794  0.027206    0.711538  5.794565  0.022511    3.040980
2            0.097794  0.027206    0.711538  7.275882  0.023467    3.127647
3            0.122794  0.025735    0.744681  6.064467  0.021492    3.435723
4            0.097794  0.025000    0.723404  7.397216  0.021620    3.261821

   zhangs_metric
0       0.924342
1       0.860319
2       0.896851
3       0.864998
4       0.895771
```

1.1.4 4. Post-traitement des règles découvertes

A présent, nous allons trier les règles selon une mesure, filtrer les règles selon une mesure, et sélectionner des règles en fonction de la présence de certains items dans l'antécédent et le conséquent.

[16]:
```
# pour un affichage plus lisible
pd.set_option('display.max_columns', 6)
pd.set_option('display.max_rows', 10)
pd.set_option('display.precision', 3)
```

[17]:
```
# sélection des colonnes à afficher
my_rules = arules.loc[:,['antecedents', 'consequents', 'lift', 'conviction']]
print(my_rules.shape)
```

```
(105, 4)
```

[19]:
```
# affichage des 10 premières règles
print(my_rules[:10])
```

```
                            antecedents     consequents   lift  conviction
0                          (Hot_Dog_Buns)      (Hot_Dogs)  7.690       3.156
1  (2pct_Milk, 98pct_Fat_Free_Hamburger)          (Eggs)  5.795       3.041
2  (2pct_Milk, 98pct_Fat_Free_Hamburger)  (Potato_Chips)  7.276       3.128
3                    (Aspirin, 2pct_Milk)          (Eggs)  6.064       3.436
4                    (Aspirin, 2pct_Milk)  (Potato_Chips)  7.397       3.262
5                    (Aspirin, 2pct_Milk)   (White_Bread)  6.609       4.140
6                  (White_Bread, Bananas)     (2pct_Milk)  7.261       4.353
7                    (2pct_Milk, Bananas)   (White_Bread)  6.833       4.735
8                           (Eggs, Cola)     (2pct_Milk)  6.638       3.265
9                      (Cola, 2pct_Milk)          (Eggs)  5.923       3.216
```

[20]:
```
# affichage des règles avec un lift supérieur ou égal à 7
print(my_rules[my_rules['lift'].ge(7.0)])
```

```
                                antecedents     consequents   lift  \
0                             (Hot_Dog_Buns)      (Hot_Dogs)  7.690
2      (2pct_Milk, 98pct_Fat_Free_Hamburger)  (Potato_Chips)  7.276
4                       (Aspirin, 2pct_Milk)  (Potato_Chips)  7.397
6                      (White_Bread, Bananas)     (2pct_Milk)  7.261
11                        (Wheat_Bread, Cola)     (2pct_Milk)  7.261
..                                        …             …    …
98            (Eggs, White_Bread, Toothpaste)     (2pct_Milk)  7.261
101    (Potato_Chips, White_Bread, 2pct_Milk)    (Toothpaste)  9.514
102  (Potato_Chips, White_Bread, Toothpaste)     (2pct_Milk)  7.569
103    (Potato_Chips, 2pct_Milk, Toothpaste)   (White_Bread)  7.319
104     (White_Bread, 2pct_Milk, Toothpaste)  (Potato_Chips)  7.726

     conviction
0         3.156
2         3.128
4         3.262
6         4.353
11        4.353
..          …
98        4.353
101       3.766
102       5.215
103       6.871
104       3.691

[22 rows x 4 columns]
```

[21]:
```
# tri des règles selon le lift (ordre décroissant) et affichage des 10␣
  ↪meilleures règles
print(my_rules.sort_values(by='lift', ascending=False)[:10])
```

```
                                antecedents                consequents  \
101    (Potato_Chips, White_Bread, 2pct_Milk)               (Toothpaste)
78               (Sweet_Relish, Hot_Dog_Buns)                 (Hot_Dogs)
97             (Eggs, White_Bread, 2pct_Milk)               (Toothpaste)
79                   (Hot_Dog_Buns, Hot_Dogs)              (Sweet_Relish)
43              (White_Bread, Hamburger_Buns)  (98pct_Fat_Free_Hamburger)
104     (White_Bread, 2pct_Milk, Toothpaste)              (Potato_Chips)
0                             (Hot_Dog_Buns)                 (Hot_Dogs)
29                      (Onions, Wheat_Bread)               (2pct_Milk)
102  (Potato_Chips, White_Bread, Toothpaste)               (2pct_Milk)
45   (Wheat_Bread, 98pct_Fat_Free_Hamburger)              (White_Bread)

      lift  conviction
101  9.514       3.766
78   9.031       5.558
97   8.995       3.222
```

```
79   8.433       3.259
43   8.202       3.874
104  7.726       3.691
0    7.690       3.156
29   7.574       5.231
102  7.569       5.215
45   7.556       8.809
```

[22]:
```
# affichage des règles contenant 'Eggs' dans le conséquent
print(my_rules[my_rules['consequents'].ge({'Eggs'})])
```

```
                              antecedents consequents   lift  conviction
1     (2pct_Milk, 98pct_Fat_Free_Hamburger)      (Eggs)  5.795       3.041
3                     (Aspirin, 2pct_Milk)      (Eggs)  6.064       3.436
9                        (Cola, 2pct_Milk)      (Eggs)  5.923       3.216
14    (2pct_Milk, Pepperoni_Pizza_-_Frozen)      (Eggs)  6.064       3.436
16                (2pct_Milk, Popcorn_Salt)      (Eggs)  6.696       4.934
..                                      …         …    …          …
70                  (White_Bread, Potatoes)      (Eggs)  5.897       3.180
72             (Sugar_Cookies, White_Bread)      (Eggs)  6.277       3.828
73               (Sweet_Relish, Toothpaste)      (Eggs)  5.749       2.983
96   (Potato_Chips, White_Bread, 2pct_Milk)      (Eggs)  6.515       4.386
100    (White_Bread, 2pct_Milk, Toothpaste)      (Eggs)  6.334       3.947

[33 rows x 4 columns]
```

[23]:
```
# affichage des règles où le conséquent est exactement 'Eggs'
print(my_rules[my_rules['consequents'].eq({'Eggs'})])
```

```
                              antecedents consequents   lift  conviction
1     (2pct_Milk, 98pct_Fat_Free_Hamburger)      (Eggs)  5.795       3.041
3                     (Aspirin, 2pct_Milk)      (Eggs)  6.064       3.436
9                        (Cola, 2pct_Milk)      (Eggs)  5.923       3.216
14    (2pct_Milk, Pepperoni_Pizza_-_Frozen)      (Eggs)  6.064       3.436
16                (2pct_Milk, Popcorn_Salt)      (Eggs)  6.696       4.934
..                                      …         …    …          …
70                  (White_Bread, Potatoes)      (Eggs)  5.897       3.180
72             (Sugar_Cookies, White_Bread)      (Eggs)  6.277       3.828
73               (Sweet_Relish, Toothpaste)      (Eggs)  5.749       2.983
96   (Potato_Chips, White_Bread, 2pct_Milk)      (Eggs)  6.515       4.386
100    (White_Bread, 2pct_Milk, Toothpaste)      (Eggs)  6.334       3.947

[33 rows x 4 columns]
```

[24]:
```
# idem mais en triant selon le lift (ordre décroissant)
print(my_rules[my_rules['consequents'].eq({'Eggs'})].sort_values(by='lift',␣
 ↪ascending=False))
```

```
                              antecedents consequents   lift  conviction
```

```
16               (2pct_Milk, Popcorn_Salt)      (Eggs)  6.696       4.934
62                (Popcorn_Salt, Potatoes)      (Eggs)  6.593       4.605
96  (Potato_Chips, White_Bread, 2pct_Milk)      (Eggs)  6.515       4.386
69                (Sweet_Relish, Potatoes)      (Eggs)  6.482       4.298
21                   (2pct_Milk, Tomatoes)      (Eggs)  6.478       4.289
..                                     …         …    …          …
58                 (White_Bread, Hot_Dogs)      (Eggs)  5.775       3.015
51                    (Potato_Chips, Cola)      (Eggs)  5.768       3.008
73              (Sweet_Relish, Toothpaste)      (Eggs)  5.749       2.983
47                  (White_Bread, Aspirin)      (Eggs)  5.715       2.941
25                (White_Bread, 2pct_Milk)      (Eggs)  5.701       2.924

[33 rows x 4 columns]
```

[25]:
```
# affichage des règles avec '2pct_Milk' contenu dans l'antécédent et 'Eggs'␣
 ↪correspondant au conséquent
print(my_rules[my_rules['antecedents'].ge({'2pct_Milk'}) &␣
 ↪my_rules['consequents'].eq({'Eggs'})]
      .sort_values(by='lift', ascending=False))
```

```
                              antecedents consequents   lift  conviction
16                (2pct_Milk, Popcorn_Salt)      (Eggs)  6.696       4.934
96   (Potato_Chips, White_Bread, 2pct_Milk)      (Eggs)  6.515       4.386
21                   (2pct_Milk, Tomatoes)      (Eggs)  6.478       4.289
100    (White_Bread, 2pct_Milk, Toothpaste)      (Eggs)  6.334       3.947
17                (Potato_Chips, 2pct_Milk)      (Eggs)  6.141       3.567
..                                      …         …    …          …
24                 (Wheat_Bread, 2pct_Milk)      (Eggs)  5.897       3.180
19                (Sweet_Relish, 2pct_Milk)      (Eggs)  5.817       3.070
22                  (2pct_Milk, Toothpaste)      (Eggs)  5.797       3.044
1     (2pct_Milk, 98pct_Fat_Free_Hamburger)      (Eggs)  5.795       3.041
25                 (White_Bread, 2pct_Milk)      (Eggs)  5.701       2.924

[14 rows x 4 columns]
```

1.1.5 5. À vous de jouer

Les règles que nous avons ne sont pas vraiment très intéressantes. La valeur du support minimum est peut-être trop élevée.

Déterminez les règles d'association avec cette fois minsup=0,015 et minconf=0,7 (la cardinalité max reste fixée à 4). On a alors 4 759 motifs et 3 265 règles.

Trouvez quelques règles qui vous semblent intéressantes.

[26]:
```
taillemax = 4
freq_itemsets = apriori(dataset, min_support=0.015, max_len=taillemax,␣
 ↪use_colnames=True, verbose=0)
print(freq_itemsets.shape)
```

```
print(freq_itemsets.head(15))
```

```
(4759, 2)
    support                    itemsets
0     0.030        (100_Watt_Lightbulb)
1     0.110                  (2pct_Milk)
2     0.037         (60_Watt_Lightbulb)
3     0.032         (75_Watt_Lightbulb)
4     0.093  (98pct_Fat_Free_Hamburger)
..      …                          …
10    0.033               (Apple_Jelly)
11    0.024               (Apple_Juice)
12    0.032               (Apple_Sauce)
13    0.054                    (Apples)
14    0.067                   (Aspirin)

[15 rows x 2 columns]
```

[30]:
```
arules = association_rules(freq_itemsets, metric="confidence", min_threshold=0.
↪7)
print(arules.shape)
```

```
(3265, 10)
```

Il y a beaucoup de règles, essayons avec minconf=0.9 (90%)

[35]:
```
arules = association_rules(freq_itemsets, metric="confidence", min_threshold=0.
↪9)
print(arules.shape)
```

```
(154, 10)
```

[36]:
```
print(arules.iloc[:5,:])
```

```
                  antecedents  consequents  antecedent support  …  \
0   (Wheat_Bread, Apple_Jelly)  (2pct_Milk)                0.02  …
1        (Corn_Oil, 2pct_Milk)       (Eggs)                0.02  …
2  (Oranges, Hair_Conditioner)  (2pct_Milk)                0.02  …
3       (Hot_Dogs, Trash_Bags)  (2pct_Milk)                0.02  …
4    (White_Bread, Trash_Bags)  (2pct_Milk)                0.02  …

   leverage  conviction  zhangs_metric
0      0.02       11.58           0.90
1      0.01       19.30           0.89
2      0.01       10.24           0.90
3      0.01       21.37           0.90
4      0.01       10.24           0.90

[5 rows x 10 columns]
```

[40]:
```
my_rules = arules.loc[:,['antecedents', 'consequents', 'lift', 'conviction']]
print(my_rules.shape)
```

```
(154, 4)
```

[41]:
```
pd.set_option('display.max_columns', 6)
pd.set_option('display.max_rows', 10)
pd.set_option('display.precision', 2)

# affichage des 5 premières règles
print(my_rules[:10])
```

```
                                antecedents    consequents  lift  conviction
0                (Wheat_Bread, Apple_Jelly)    (2pct_Milk)  8.43       11.58
1                     (Corn_Oil, 2pct_Milk)         (Eggs)  7.77       19.30
2               (Oranges, Hair_Conditioner)    (2pct_Milk)  8.33       10.24
3                    (Hot_Dogs, Trash_Bags)    (2pct_Milk)  8.75       21.37
4                  (White_Bread, Trash_Bags)    (2pct_Milk)  8.33       10.24
5  (Wheat_Bread, 98pct_Fat_Free_Hamburger)  (White_Bread)  7.56        8.81
6          (Apple_Fruit_Roll, Popcorn_Salt)         (Eggs)  8.14         inf
7                    (Apples, Popcorn_Salt)  (White_Bread)  7.67       10.13
8               (C_Cell_Batteries, Aspirin)  (White_Bread)  7.75       11.45
9                 (Aspirin, Cottage_Cheese)  (White_Bread)  7.67       10.13
```

[42]:
```
# affichage des règles avec un LIFT supérieur ou égal à 7
print(my_rules[my_rules['lift'].ge(7.0)])
```

```
                                    antecedents    consequents   lift  \
0                     (Wheat_Bread, Apple_Jelly)    (2pct_Milk)   8.43
1                          (Corn_Oil, 2pct_Milk)         (Eggs)   7.77
2                    (Oranges, Hair_Conditioner)    (2pct_Milk)   8.33
3                         (Hot_Dogs, Trash_Bags)    (2pct_Milk)   8.75
4                       (White_Bread, Trash_Bags)    (2pct_Milk)   8.33
..                                            ...            ...    ...
149    (Potato_Chips, Toothpaste, Popcorn_Salt)  (White_Bread)   7.72
150    (Toilet_Paper, Toothpaste, Popcorn_Salt)  (White_Bread)   7.70
151     (Sugar_Cookies, Potato_Chips, Tomatoes)  (White_Bread)   7.72
152  (Sweet_Relish, Sugar_Cookies, White_Bread)   (Toothpaste)  11.50
153   (Sweet_Relish, Sugar_Cookies, Toothpaste)  (White_Bread)   7.67

     conviction
0         11.58
1         19.30
2         10.24
3         21.37
4         10.24
..          ...
149       11.01
150       10.57
```

```
151       11.01
152       10.59
153       10.13

[154 rows x 4 columns]
```

[43]:
```
# tri des règles selon le lift (ordre décroissant) et affichage des 10␣
 ↪meilleures règles
print(my_rules.sort_values(by='lift', ascending=False)[:10])
```

```
                                     antecedents     consequents   lift  \
152    (Sweet_Relish, Sugar_Cookies, White_Bread)    (Toothpaste)  11.50
145            (Hot_Dog_Buns, Hot_Dogs, Potatoes)  (Sweet_Relish)  10.70
15                    (Hot_Dog_Buns, Canned_Tuna)      (Hot_Dogs)   9.89
50                  (Apples, Onions, Wheat_Bread)     (2pct_Milk)   9.13
66            (White_Bread, Wheat_Bread, Bananas)     (2pct_Milk)   8.80
48                   (Eggs, Potato_Chips, Apples)     (2pct_Milk)   8.79
82   (Eggs, Pepperoni_Pizza_-_Frozen, Toothpaste)     (2pct_Milk)   8.78
99          (Onions, Wheat_Bread, Ramen_Noodles)     (2pct_Milk)   8.76
3                           (Hot_Dogs, Trash_Bags)     (2pct_Milk)   8.75
96   (Potato_Chips, Toothpaste, Hair_Conditioner)     (2pct_Milk)   8.73

     conviction
152       10.59
145       10.52
15        10.89
50          inf
66        24.93
48        24.04
82        23.15
99        22.26
3         21.37
96        20.48
```

[44]:
```
# filtrage des règles contenant 'Aspirin' dans l'antécédent
print(my_rules[my_rules['antecedents'].ge({'Aspirin'})])
print(my_rules[my_rules['antecedents'].ge({'Hamburger_Buns'})])
print(my_rules[my_rules['antecedents'].ge({'Tissues'})])
```

```
                           antecedents    consequents  lift  conviction
8             (C_Cell_Batteries, Aspirin)  (White_Bread)  7.75       11.45
9               (Aspirin, Cottage_Cheese)  (White_Bread)  7.67       10.13
10                     (Aspirin, Donuts)  (White_Bread)  8.03       20.26
11                    (Tissues, Aspirin)        (Eggs)  7.77       19.30
12                     (Garlic, Aspirin)  (White_Bread)  8.40         inf
..                                   …            …   …        …
116         (Eggs, Aspirin, Popcorn_Salt)  (White_Bread)  7.72       11.01
117       (Potato_Chips, Onions, Aspirin)  (White_Bread)  7.72       11.01
118        (Aspirin, Onions, Wheat_Bread)  (White_Bread)  7.67       10.13
```

```
119  (Potato_Chips, Wheat_Bread, Aspirin)  (White_Bread)  7.72       11.01
120      (Wheat_Bread, Aspirin, Potatoes)  (White_Bread)  7.70       10.57

[17 rows x 4 columns]
Empty DataFrame
Columns: [antecedents, consequents, lift, conviction]
Index: []
               antecedents consequents  lift  conviction
11       (Tissues, Aspirin)      (Eggs)  7.77        19.3
33  (Sweet_Relish, Tissues)      (Eggs)  7.77        19.3
```

[45]:
```
print(my_rules[my_rules['conviction'].ge(100000000)].sort_values(by='lift',␣
 ↪ascending=False)[:10])
```

```
                                          antecedents    consequents  lift  \
50                      (Apples, Onions, Wheat_Bread)    (2pct_Milk)  9.13
12                                  (Garlic, Aspirin)  (White_Bread)  8.40
6                     (Apple_Fruit_Roll, Popcorn_Salt)         (Eggs)  8.14
80   (Sweet_Relish, 2pct_Milk, Pepperoni_Pizza_-_Fr…         (Eggs)  8.14
84              (White_Bread, 2pct_Milk, Plain_Bagels)         (Eggs)  8.14
136                 (Sweet_Relish, Tomatoes, Potatoes)         (Eggs)  8.14

     conviction
50          inf
12          inf
6           inf
80          inf
84          inf
136         inf
```

[46]:
```
# conséquents des regles
c = my_rules['consequents']
mon_set = set()
for i in c:
    mon_set.add(i)
print(mon_set)
```

```
{frozenset({'Sweet_Relish'}), frozenset({'Toothpaste'}),
frozenset({'2pct_Milk'}), frozenset({'Hot_Dogs'}), frozenset({'Eggs'}),
frozenset({'White_Bread'})}
```

[47]:
```
itemsconsequents = ['Pepperoni_Pizza_-_Frozen', 'Eggs', 'Tomatoes', 'Cola',␣
 ↪'White_Bread', 'Onions',
                    '2pct_Milk', 'Wheat_Bread', 'Toothpaste', 'Toilet_Paper',␣
 ↪'Cola', 'Potato_Chips',
                    'Sugar_Cookies', 'Bananas', 'Sweet_Relish', 'Aspirin',␣
 ↪'Popcorn_Salt', 'Cream_Cheese',
```

```
                    'Hot_Dogs', '98pct_Fat_Free_Hamburger', 'Ramen_Noodles',␣
 ↪'Potatoes']

for item in itemsconsequents:
    regles_select = my_rules[my_rules['consequents'].ge({item})]
    if regles_select.size > 0:
        print("\n", item)
        print(regles_select)
```

```
 Eggs
                                antecedents consequents  lift  conviction
1                      (Corn_Oil, 2pct_Milk)      (Eggs)  7.77       19.30
6           (Apple_Fruit_Roll, Popcorn_Salt)      (Eggs)  8.14         inf
11                         (Tissues, Aspirin)      (Eggs)  7.77       19.30
18                           (Corn_Oil, Cola)      (Eggs)  7.77       19.30
19                        (Corn_Oil, Potatoes)      (Eggs)  7.77       19.30
..                                         …         …   …         …
133  (Potato_Chips, Sugar_Cookies, Tomatoes)      (Eggs)  7.49       10.97
134       (Salsa_Dip, White_Bread, Potatoes)      (Eggs)  7.44       10.09
136       (Sweet_Relish, Tomatoes, Potatoes)      (Eggs)  8.14         inf
137    (Sugar_Cookies, Toothpaste, Tomatoes)      (Eggs)  7.52       11.40
139   (Sugar_Cookies, White_Bread, Tomatoes)      (Eggs)  7.38        9.36

[42 rows x 4 columns]

 White_Bread
                                    antecedents    consequents  lift  \
5      (Wheat_Bread, 98pct_Fat_Free_Hamburger)  (White_Bread)  7.56
7                        (Apples, Popcorn_Salt)  (White_Bread)  7.67
8                   (C_Cell_Batteries, Aspirin)  (White_Bread)  7.75
9                     (Aspirin, Cottage_Cheese)  (White_Bread)  7.67
10                           (Aspirin, Donuts)  (White_Bread)  8.03
..                                         …            …   …
148          (Sweet_Relish, Plums, Toothpaste)  (White_Bread)  7.67
149   (Potato_Chips, Toothpaste, Popcorn_Salt)  (White_Bread)  7.72
150   (Toilet_Paper, Toothpaste, Popcorn_Salt)  (White_Bread)  7.70
151    (Sugar_Cookies, Potato_Chips, Tomatoes)  (White_Bread)  7.72
153  (Sweet_Relish, Sugar_Cookies, Toothpaste)  (White_Bread)  7.67

     conviction
5          8.81
7         10.13
8         11.45
9         10.13
10        20.26
..          …
148       10.13
```

```
149       11.01
150       10.57
151       11.01
153       10.13

[70 rows x 4 columns]

 2pct_Milk
                                     antecedents  consequents  lift  \
0                      (Wheat_Bread, Apple_Jelly)  (2pct_Milk)  8.43
2                     (Oranges, Hair_Conditioner)  (2pct_Milk)  8.33
3                         (Hot_Dogs, Trash_Bags)  (2pct_Milk)  8.75
4                       (White_Bread, Trash_Bags)  (2pct_Milk)  8.33
45                (Apples, White_Bread, Bananas)  (2pct_Milk)  8.33
..                                           …          …   …
90              (Eggs, Sandwich_Bags, Toothpaste)  (2pct_Milk)  8.33
96   (Potato_Chips, Toothpaste, Hair_Conditioner)  (2pct_Milk)  8.73
99           (Onions, Wheat_Bread, Ramen_Noodles)  (2pct_Milk)  8.76
101     (White_Bread, Wheat_Bread, Ramen_Noodles)  (2pct_Milk)  8.43
102           (Sweet_Relish, Ravioli, Toothpaste)  (2pct_Milk)  8.33

     conviction
0         11.58
2         10.24
3         21.37
4         10.24
45        10.24
..          …
90        10.24
96        20.48
99        22.26
101       11.58
102       10.24

[39 rows x 4 columns]

 Toothpaste
                                   antecedents   consequents  lift  \
152  (Sweet_Relish, Sugar_Cookies, White_Bread)  (Toothpaste)  11.5

     conviction
152       10.59

 Sweet_Relish
                           antecedents     consequents  lift  conviction
145  (Hot_Dog_Buns, Hot_Dogs, Potatoes)  (Sweet_Relish)  10.7       10.52

 Hot_Dogs
```

```
                    antecedents consequents  lift  conviction
15  (Hot_Dog_Buns, Canned_Tuna)  (Hot_Dogs)  9.89       10.89
```

Certains items (2pct_Milk, white_Bread, ...) peuvent "poller" les résultats car ils correspondent à des produits achetés couramment. On pourrait filtrer les règles ou les enlever des données avant la découverte des règles.

[48]:
```
dataset2 = dataset.copy(deep=True)
```

[49]:
```
dataset2.drop("2pct_Milk", axis=1, inplace=True)
dataset2.drop("White_Bread", axis=1, inplace=True)
dataset2.drop("Wheat_Bread", axis=1, inplace=True)
dataset2.drop("Eggs", axis=1, inplace=True)
```

[50]:
```
taillemax = 4
freq_itemsets2 = apriori(dataset2, min_support=0.015, max_len=taillemax,␣
 ↪use_colnames=True, verbose=0)
print(freq_itemsets2.shape)
```

```
(1933, 2)
```

[51]:
```
arules2 = association_rules(freq_itemsets2, metric="confidence",␣
 ↪min_threshold=0.8)
print(arules2.shape)
```

```
(38, 10)
```

[52]:
```
pd.set_option('display.max_rows', 50)
my_rules2 = arules2.loc[:,['antecedents', 'consequents', 'lift']]
print(my_rules2.sort_values(by='lift', ascending=False))
```

```
                                    antecedents                consequents  \
18               (Potato_Chips, Graham_Crackers)                (Toothpaste)
7                                 (Apples, Hot_Dogs)                (Toothpaste)
34            (Hot_Dog_Buns, Hot_Dogs, Potatoes)               (Sweet_Relish)
5                                  (Apples, Bananas)                (Toothpaste)
37              (Toothpaste, Hot_Dogs, Potatoes)               (Sweet_Relish)
27                        (Sweet_Relish, Ravioli)                (Toothpaste)
14                         (Cantaloupe, Tomatoes)                (Toothpaste)
29                (Sweet_Relish, Cola, Potatoes)                (Toothpaste)
12                   (Hot_Dog_Buns, Canned_Tuna)                   (Hot_Dogs)
36            (Toilet_Paper, Hot_Dogs, Potatoes)               (Sweet_Relish)
17                  (Sweet_Relish, French_Fries)                   (Potatoes)
13                       (Cantaloupe, Hot_Dogs)               (Sweet_Relish)
25                          (Plums, Toothpaste)               (Sweet_Relish)
32       (Cream_Cheese, Hot_Dog_Buns, Hot_Dogs)               (Sweet_Relish)
30                  (Cola, Toothpaste, Potatoes)               (Sweet_Relish)
15                          (Hot_Dog_Buns, Cola)                   (Hot_Dogs)
31   (Cream_Cheese, Hot_Dog_Buns, Sweet_Relish)                   (Hot_Dogs)
21                  (Toilet_Paper, Hot_Dog_Buns)                   (Hot_Dogs)
```

22	(Sour_Cream, Sweet_Relish)	(Hot_Dogs)	
4	(Hamburger_Buns, Ramen_Noodles)	(98pct_Fat_Free_Hamburger)	
33	(Sweet_Relish, Hot_Dog_Buns, Potatoes)	(Hot_Dogs)	
10	(Hot_Dog_Buns, Aspirin)	(Hot_Dogs)	
16	(Cream_Cheese, Hot_Dog_Buns)	(Hot_Dogs)	
20	(Sweet_Relish, Hot_Dog_Buns)	(Hot_Dogs)	
6	(Apples, Canned_Tuna)	(Potato_Chips)	
23	(Licorice, Popcorn_Salt)	(Potato_Chips)	
2	(Corn_Oil, 98pct_Fat_Free_Hamburger)	(Potato_Chips)	
35	(Toilet_Paper, Sweet_Relish, Potatoes)	(Hot_Dogs)	
19	(Hot_Dog_Buns, Popcorn_Salt)	(Hot_Dogs)	
0	(98pct_Fat_Free_Hamburger, BBQ_Potato_Chips)	(Potato_Chips)	
1	(Corn_Oil, Potato_Chips)	(98pct_Fat_Free_Hamburger)	
28	(Aspirin, Toothpaste, Potatoes)	(Potato_Chips)	
11	(Aspirin, Waffles)	(Potato_Chips)	
3	(Garlic, 98pct_Fat_Free_Hamburger)	(Potato_Chips)	
26	(Waffles, Tomatoes)	(Potato_Chips)	
9	(French_Fries, Aspirin)	(Potato_Chips)	
24	(Plums, Popcorn_Salt)	(Potato_Chips)	
8	(Cream_Cheese, Aspirin)	(Potato_Chips)	

	lift
18	11.02
7	10.79
34	10.70
5	10.66
37	10.37
27	10.34
14	10.17
29	10.17
12	9.89
36	9.85
17	9.82
13	9.70
25	9.63
32	9.47
30	9.47
15	9.44
31	9.44
21	9.25
22	9.19
4	9.12
33	9.07
10	9.07
16	9.05
20	9.03
6	9.00
23	8.95

```
2    8.95
35   8.72
19   8.72
0    8.65
1    8.65
28   8.59
11   8.59
3    8.58
26   8.40
9    8.33
24   8.26
8    8.26
```

Les données proviennent d'un magasin US et sans surprise on retrouve des produits tels que les hot dogs, popcorn, cola, etc. Avec des règles comme "(Cream_Cheese, Hot_Dog_Buns, Hot_Dogs) -> (Sweet_Relish)", nous pouvons suggérer au gérant du magasin de faire une promotion sur la sauce sweet_relish ou de mettre quelques exemplaires proche de cream_cheese ou de hot_dogs. On pourrait aussi avoir une promo sur l'achat de l'ensemble des items présents dans la règle. L'analyse des résultats peut même amener le gérant à réorganiser certains rayons.

Remarque finale : ici on a analysé ce qui se vend bien ensemble, mais on pourrait aussi regarder ce qui ce vend moins bien et donc inciter les clients à les acheter via des promotions, par exemple.

[ ]: