breadbasket

September 6, 2023

[7]:
```python
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/apriorifp/BreadBasket (1).xlsx
/kaggle/input/apriorifp2/BreadBasket.csv
```

[5]:
```
pip install pandas mlxtend pyfpgrowth
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages
(2.0.3)
Requirement already satisfied: mlxtend in /opt/conda/lib/python3.10/site-
packages (0.22.0)
Requirement already satisfied: pyfpgrowth in /opt/conda/lib/python3.10/site-
packages (1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas) (2023.3)
```

```
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-
packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in /opt/conda/lib/python3.10/site-
packages (from pandas) (1.23.5)
Requirement already satisfied: scipy>=1.2.1 in /opt/conda/lib/python3.10/site-
packages (from mlxtend) (1.11.2)
Requirement already satisfied: scikit-learn>=1.0.2 in
/opt/conda/lib/python3.10/site-packages (from mlxtend) (1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in
/opt/conda/lib/python3.10/site-packages (from mlxtend) (3.7.2)
Requirement already satisfied: joblib>=0.13.2 in /opt/conda/lib/python3.10/site-
packages (from mlxtend) (1.3.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.10/site-
packages (from mlxtend) (68.0.0)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib>=3.0.0->mlxtend)
(1.1.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-
packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib>=3.0.0->mlxtend)
(4.40.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib>=3.0.0->mlxtend)
(1.4.4)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/site-
packages (from matplotlib>=3.0.0->mlxtend) (9.5.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib>=3.0.0->mlxtend)
(3.0.9)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn>=1.0.2->mlxtend)
(3.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[12]:
```python
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
import pyfpgrowth


# Read the CSV file into a DataFrame
df = pd.read_csv("/kaggle/input/apriorifp2/BreadBasket.csv")
df
```

```
[12]:     Tx                          products
      0    0                MILK,BREAD,BISCUIT
      1    1     BREAD,MILK,BISCUIT,CORNFLAKES
      2    2                BREAD,TEA,BOURNVITA
      3    3               JAM,MAGGI,BREAD,MILK
      4    4                  MAGGI,TEA,BISCUIT
      5    5                BREAD,TEA,BOURNVITA
      6    6               MAGGI,TEA,CORNFLAKES
      7    7            MAGGI,BREAD,TEA,BISCUIT
      8    8                JAM,MAGGI,BREAD,TEA
      9    9                         BREAD,MILK
      10  10   COFFEE,COCK,BISCUIT,CORNFLAKES
      11  11   COFFEE,COCK,BISCUIT,CORNFLAKES
      12  12          COFFEE,SUGER,BOURNVITA
      13  13                BREAD,COFFEE,COCK
      14  14              BREAD,SUGER,BISCUIT
      15  15          COFFEE,SUGER,CORNFLAKES
      16  16            BREAD,SUGER,BOURNVITA
      17  17               BREAD,COFFEE,SUGER
      18  18               BREAD,COFFEE,SUGER
      19  19       TEA,MILK,COFFEE,CORNFLAKES
```

```
[13]: df['products'] = df['products'].apply(lambda x: x.split(','))
      df
```

```
[13]:     Tx                                products
      0    0                [MILK, BREAD, BISCUIT]
      1    1     [BREAD, MILK, BISCUIT, CORNFLAKES]
      2    2                [BREAD, TEA, BOURNVITA]
      3    3              [JAM, MAGGI, BREAD, MILK]
      4    4                  [MAGGI, TEA, BISCUIT]
      5    5                [BREAD, TEA, BOURNVITA]
      6    6               [MAGGI, TEA, CORNFLAKES]
      7    7           [MAGGI, BREAD, TEA, BISCUIT]
      8    8               [JAM, MAGGI, BREAD, TEA]
      9    9                          [BREAD, MILK]
      10  10   [COFFEE, COCK, BISCUIT, CORNFLAKES]
      11  11   [COFFEE, COCK, BISCUIT, CORNFLAKES]
      12  12           [COFFEE, SUGER, BOURNVITA]
      13  13                 [BREAD, COFFEE, COCK]
      14  14               [BREAD, SUGER, BISCUIT]
      15  15          [COFFEE, SUGER, CORNFLAKES]
      16  16             [BREAD, SUGER, BOURNVITA]
      17  17                [BREAD, COFFEE, SUGER]
      18  18                [BREAD, COFFEE, SUGER]
      19  19       [TEA, MILK, COFFEE, CORNFLAKES]
```

```
[16]: # Apriori Algorithm
min_support = 0.02

# Encode the items as binary values
oht = df['products'].str.join('|').str.get_dummies()
frequent_itemsets_apriori = apriori(oht, min_support=min_support,␣
↪use_colnames=True)

# Generate association rules using Apriori
rules_apriori = association_rules(frequent_itemsets_apriori, metric='lift',␣
↪min_threshold=1.0)

# FP-Growth Algorithm
# Create a list of transactions
transactions = df['products'].tolist()

# Find frequent itemsets using FP-Growth
patterns = pyfpgrowth.find_frequent_patterns(transactions, support_threshold=2)

# Generate association rules using FP-Growth
rules_fpgrowth = pyfpgrowth.generate_association_rules(patterns,␣
↪confidence_threshold=0.5)

print("Apriori Rules:")
print(rules_apriori)

# Print the association rules for FP-Growth
print("FP-Growth Rules:")
for antecedent, consequent in rules_fpgrowth.items():
    print(f"Rule: {antecedent} -> {consequent[0]}, Confidence: {consequent[1]}")
```

```
Apriori Rules:
           antecedents                  consequents  antecedent support  \
0            (BISCUIT)                       (COCK)                0.35
1               (COCK)                    (BISCUIT)                0.15
2            (BISCUIT)                 (CORNFLAKES)                0.35
3         (CORNFLAKES)                    (BISCUIT)                0.30
4            (BISCUIT)                      (MAGGI)                0.35
..                 …                           …                 …
257  (TEA, CORNFLAKES)               (MILK, COFFEE)                0.10
258             (MILK)    (COFFEE, CORNFLAKES, TEA)                0.25
259           (COFFEE)      (MILK, CORNFLAKES, TEA)                0.40
260       (CORNFLAKES)          (MILK, COFFEE, TEA)                0.30
261              (TEA)   (MILK, COFFEE, CORNFLAKES)                0.35

     consequent support  support  confidence      lift  leverage  conviction  \
0                  0.15     0.10    0.285714  1.904762    0.0475    1.190000
```

```
1                 0.35     0.10    0.666667   1.904762    0.0475    1.950000
2                 0.30     0.15    0.428571   1.428571    0.0450    1.225000
3                 0.35     0.15    0.500000   1.428571    0.0450    1.300000
4                 0.25     0.10    0.285714   1.142857    0.0125    1.050000
..                 …        …         …         …          …          …
257               0.05     0.05    0.500000  10.000000    0.0450    1.900000
258               0.05     0.05    0.200000   4.000000    0.0375    1.187500
259               0.05     0.05    0.125000   2.500000    0.0300    1.085714
260               0.05     0.05    0.166667   3.333333    0.0350    1.140000
261               0.05     0.05    0.142857   2.857143    0.0325    1.108333

     zhangs_metric
0         0.730769
1         0.558824
2         0.461538
3         0.428571
4         0.192308
..             …
257       1.000000
258       1.000000
259       1.000000
260       1.000000
261       1.000000

[262 rows x 10 columns]
FP-Growth Rules:
Rule: ('JAM',) -> ('BREAD', 'MAGGI'), Confidence: 1.0
Rule: ('BREAD', 'JAM') -> ('MAGGI',), Confidence: 1.0
Rule: ('BREAD', 'MAGGI') -> ('TEA',), Confidence: 0.6666666666666666
Rule: ('JAM', 'MAGGI') -> ('BREAD',), Confidence: 1.0
Rule: ('COCK',) -> ('BISCUIT', 'COFFEE', 'CORNFLAKES'), Confidence:
0.6666666666666666
Rule: ('BISCUIT', 'COCK') -> ('COFFEE', 'CORNFLAKES'), Confidence: 1.0
Rule: ('BISCUIT', 'CORNFLAKES') -> ('COFFEE',), Confidence: 0.6666666666666666
Rule: ('COCK', 'CORNFLAKES') -> ('BISCUIT', 'COFFEE'), Confidence: 1.0
Rule: ('COCK', 'COFFEE') -> ('BISCUIT', 'CORNFLAKES'), Confidence:
0.6666666666666666
Rule: ('COFFEE', 'CORNFLAKES') -> ('BISCUIT',), Confidence: 0.5
Rule: ('BISCUIT', 'COFFEE') -> ('CORNFLAKES',), Confidence: 1.0
Rule: ('BISCUIT', 'COCK', 'COFFEE') -> ('CORNFLAKES',), Confidence: 1.0
Rule: ('BISCUIT', 'COCK', 'CORNFLAKES') -> ('COFFEE',), Confidence: 1.0
Rule: ('BISCUIT', 'COFFEE', 'CORNFLAKES') -> ('COCK',), Confidence: 1.0
Rule: ('COCK', 'COFFEE', 'CORNFLAKES') -> ('BISCUIT',), Confidence: 1.0
Rule: ('BOURNVITA', 'BREAD') -> ('TEA',), Confidence: 0.6666666666666666
Rule: ('BOURNVITA', 'TEA') -> ('BREAD',), Confidence: 1.0
Rule: ('BREAD', 'TEA') -> ('MAGGI',), Confidence: 0.5
Rule: ('BISCUIT', 'BREAD') -> ('MILK',), Confidence: 0.5
Rule: ('BISCUIT', 'MILK') -> ('BREAD',), Confidence: 1.0
```

```
Rule: ('BREAD', 'MILK') -> ('BISCUIT',), Confidence: 0.5
Rule: ('BISCUIT', 'MAGGI') -> ('TEA',), Confidence: 1.0
Rule: ('BISCUIT', 'TEA') -> ('MAGGI',), Confidence: 1.0
Rule: ('MAGGI', 'TEA') -> ('BREAD',), Confidence: 0.5
Rule: ('TEA',) -> ('BREAD',), Confidence: 0.5714285714285714
Rule: ('COFFEE',) -> ('SUGER',), Confidence: 0.5
Rule: ('BREAD', 'COFFEE') -> ('SUGER',), Confidence: 0.6666666666666666
Rule: ('BREAD', 'SUGER') -> ('COFFEE',), Confidence: 0.5
Rule: ('COFFEE', 'SUGER') -> ('BREAD',), Confidence: 0.5
```

```
/opt/conda/lib/python3.10/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(
```