pip install apyori

Note: you may need to restart the kernel to use updated packages.Collecting apyori  
 Downloading apyori-1.1.2.tar.gz (8.6 kB)  
 Installing build dependencies: started  
 Installing build dependencies: finished with status 'done'  
 Getting requirements to build wheel: started  
 Getting requirements to build wheel: finished with status 'done'  
 Preparing metadata (pyproject.toml): started  
 Preparing metadata (pyproject.toml): finished with status 'done'  
Building wheels for collected packages: apyori  
 Building wheel for apyori (pyproject.toml): started  
 Building wheel for apyori (pyproject.toml): finished with status 'done'  
 Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5976 sha256=bbf2f2273c1097de00565ac3715eec52e802864b6722882b4687ebd0745b54b6  
 Stored in directory: c:\users\nikhi\appdata\local\pip\cache\wheels\77\3d\a6\d317a6fb32be58a602b1e8c6b5d6f31f79322da554cad2a5ea  
Successfully built apyori  
Installing collected packages: apyori  
Successfully installed apyori-1.1.2

[notice] A new release of pip is available: 23.2.1 -> 24.0  
[notice] To update, run: python.exe -m pip install --upgrade pip

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from apyori import apriori

store\_data = pd.read\_csv("Market Basket\_Small dataset.csv", header=None)  
display(store\_data.head(15))  
print(store\_data.shape)

0 1 2 3 4 5  
0 Wine Chips Bread Butter Milk Apple  
1 Wine Chips Bread Butter Milk Apple  
2 Wine Chips Bread Butter Milk NaN  
3 Wine Chips NaN Butter Milk NaN  
4 Wine NaN Bread NaN NaN Apple  
5 NaN NaN NaN Butter Milk NaN  
6 NaN Chips Bread NaN NaN Apple  
7 Wine Chips NaN Butter Milk NaN  
8 Wine NaN Bread NaN NaN Apple  
9 Wine NaN Bread NaN Milk NaN  
10 NaN Chips Bread Butter NaN Apple  
11 Wine NaN NaN Butter Milk Apple  
12 Wine Chips Bread Butter Milk NaN  
13 Wine NaN Bread NaN Milk Apple  
14 Wine NaN Bread Butter Milk Apple

(22, 6)

transactions = []  
for i in range(0, len(store\_data)):  
 transactions.append([str(store\_data.values[i,j]) for j in range(0, len(store\_data.columns))])

association\_rules = apriori(transactions, min\_support=0.5, min\_confidence=0.7, min\_lift=1.2, min\_length=2)  
association\_results = list(association\_rules)

print(len(association\_results ))

3

print(association\_results )

[RelationRecord(items=frozenset({'Butter', 'Milk '}), support=0.6363636363636364, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Butter'}), items\_add=frozenset({'Milk '}), confidence=0.9333333333333333, lift=1.207843137254902), OrderedStatistic(items\_base=frozenset({'Milk '}), items\_add=frozenset({'Butter'}), confidence=0.8235294117647058, lift=1.2078431372549019)]), RelationRecord(items=frozenset({'Wine ', 'Milk ', 'Bread'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Milk ', 'Bread'}), items\_add=frozenset({'Wine '}), confidence=0.9166666666666667, lift=1.2604166666666667)]), RelationRecord(items=frozenset({'Wine ', 'Butter', 'Milk '}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Wine ', 'Butter'}), items\_add=frozenset({'Milk '}), confidence=1.0, lift=1.2941176470588236)])]

print("There are {} Relation derived.".format(len(association\_results)))

There are 3 Relation derived.

for i in range(0, len(association\_results)):  
 print(association\_results[i][0])

frozenset({'Butter', 'Milk '})  
frozenset({'Wine ', 'Milk ', 'Bread'})  
frozenset({'Wine ', 'Butter', 'Milk '})

# Import the transaction encoder function from mlxtend  
from mlxtend.preprocessing import TransactionEncoder  
  
# Instantiate transaction encoder and identify unique items  
encoder = TransactionEncoder().fit(transactions)  
  
# One-hot encode transactions  
onehot = encoder.transform(transactions)  
  
# Convert one-hot encoded data to DataFrame  
onehot = pd.DataFrame(onehot, columns = encoder.columns\_).drop('nan', axis=1)  
  
# Print the one  
onehot.head()

Apple Bread Butter Chips Milk Wine   
0 True True True True True True  
1 True True True True True True  
2 False True True True True True  
3 False False True True True True  
4 True True False False False True

# Import the association rules function  
from mlxtend.frequent\_patterns import apriori, association\_rules  
  
# Compute frequent itemsets using the Apriori algorithm  
frequent\_itemsets = apriori(onehot, min\_support = 0.5,  
 max\_len = 2, use\_colnames = True)  
  
# Compute all association rules using confidence  
rules = association\_rules(frequent\_itemsets,  
 metric = "confidence",  
 min\_threshold = 0.7)  
  
# Print association rules  
rules.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 16 entries, 0 to 15  
Data columns (total 10 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 antecedents 16 non-null object   
 1 consequents 16 non-null object   
 2 antecedent support 16 non-null float64  
 3 consequent support 16 non-null float64  
 4 support 16 non-null float64  
 5 confidence 16 non-null float64  
 6 lift 16 non-null float64  
 7 leverage 16 non-null float64  
 8 conviction 16 non-null float64  
 9 zhangs\_metric 16 non-null float64  
dtypes: float64(8), object(2)  
memory usage: 1.4+ KB

rules.head()

antecedents consequents antecedent support consequent support support \  
0 (Apple) (Bread) 0.681818 0.727273 0.590909   
1 (Bread) (Apple) 0.727273 0.681818 0.590909   
2 (Apple) (Milk ) 0.681818 0.772727 0.500000   
3 (Apple) (Wine ) 0.681818 0.727273 0.500000   
4 (Milk ) (Bread) 0.772727 0.727273 0.545455   
  
 confidence lift leverage conviction zhangs\_metric   
0 0.866667 1.191667 0.095041 2.045455 0.505495   
1 0.812500 1.191667 0.095041 1.696970 0.589744   
2 0.733333 0.949020 -0.026860 0.852273 -0.144444   
3 0.733333 1.008333 0.004132 1.022727 0.025974   
4 0.705882 0.970588 -0.016529 0.927273 -0.117647