Market Basket Insights

Market Basket Analysis using association rules - apriori technique in Two ways :

Association rules analysis is a technique to uncover how items are associated to each other. There are three common ways to measure association.

Measure 1: Support. This says how popular an itemset is, it is number of times appear in total number of transaction. in other word we say frequency of item.

Measure 2: Confidence. This says how likely item Y is purchased when item X is purchased, expressed as {X -> Y}. This is measured by the proportion of transactions with item X, in which item Y also appears.

Measure 3: Lift. it is ratio of expected confidance to observed confidance. it is described as confidance of Y when item X was already known(x/y) to the confidance of Y when X item is unknown. in other words confidance of Y w.r.t. x and confiadnce of Y without X (means both are independent to each other).

support = occurance of item / total no of transaction.

confidance = support (X Union Y) / support(X).

lift = support (X Union Y)/ support(X) * support(Y) .

#External package need to install

!pip install apyori

#import all required packages..

import pandas as pd
import numpy as np
from apyori import apriori

#loading market basket dataset..

df = pd.read_csv('../input/basket-optimisation/Market_Basket_Optimisation.csv',header=None)
df.head()

#replacing empty value with 0.

```
df.fillna(0,inplace=True)
df.head()
```

#for using aprori need to convert data in list format..

```
transaction =
[['apple','almonds'],['apple'],['banana','apple']]....
transactions = []
for i in range(0,len(df)):
   transactions.append([str(df.values[i,j]) for j in
range(0,20) if str(df.values[i,j])!='0'])
transactions[0]
```

OutPut :

```
         ['shrimp',
 'almonds',
 'avocado',
 'vegetables mix',
 'green grapes',
 'whole weat flour',
 'yams',
 'cottage cheese',
 'energy drink',
 'tomato juice',
 'low fat yogurt',
 'green tea',
 'honey',
 'salad',
 'mineral water',
 'salmon',
 'antioxydant juice',
 'frozen smoothie',
 'spinach',
 'olive oil']

#Call apriori function which requires minimum support, confidance a
nd lift, min length is combination of item default is 2".
rules = apriori(transactions,min_support=0.003,min_confidan
ce=0.2,min_lift=3,min_length=2)
```

#it generates a set of rules in a generator file...rules:

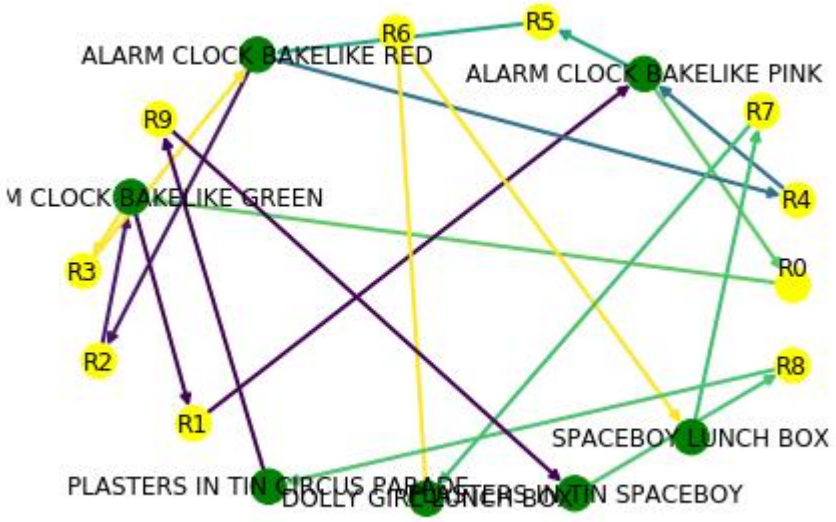
<generator object apriori at 0x7f6401fa89e8>

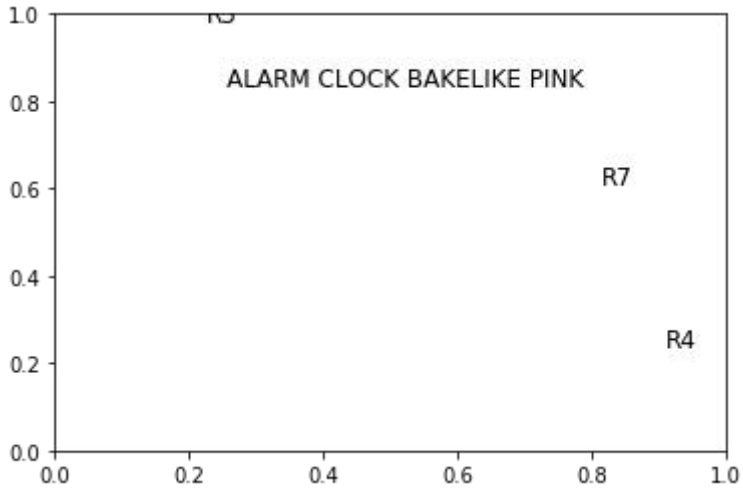
 # all rules need to be converted in a list..Results = list(rules)Res ults

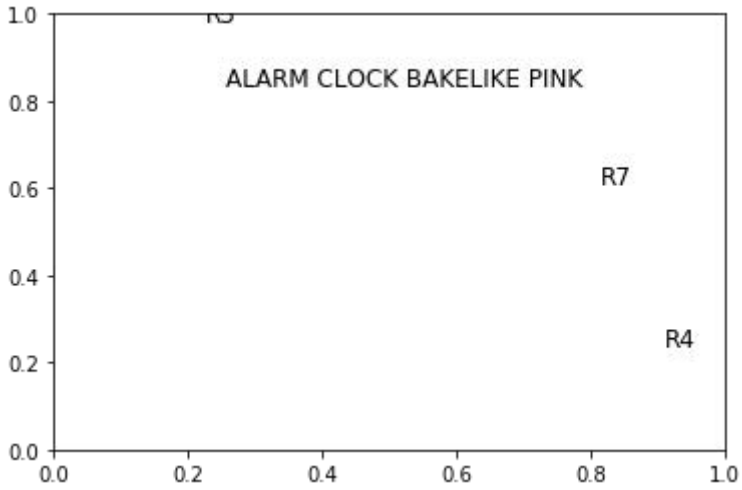
Complete Program :

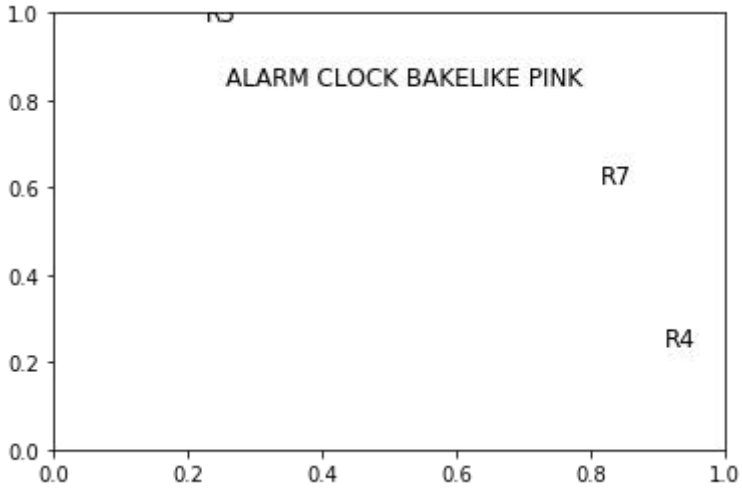
```
#plotting output in a graph plot.

import networkx as nx
import matplotlib.pyplot as plt

def draw_graph(rules, rules_to_show):
   G1 = nx.DiGraph()
   color_map=[]
   N = 50
   colors = np.random.rand(N)
   strs=['R0', 'R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8', 'R9', 'R10', 'R
11']

   for i in range(rules_to_show):
      G1.add_nodes_from(["R"+str(i)])
      for a in rules.iloc[i]['antecedents']:
         G1.add_nodes_from([a])
         G1.add_edge(a, "R"+str(i), color=colors[i] , weight = 2)
      for c in rules.iloc[i]['consequents']:
         G1.add_nodes_from([c])
         G1.add_edge("R"+str(i), c, color=colors[i],  weight=2)

   for node in G1:
      found_a_string = False
      for item in strs:
         if node==item:
            found_a_string = True
      if found_a_string:
         color_map.append('yellow')
      else:
         color_map.append('green')

   edges = G1.edges()
   colors = [G1[u][v]['color'] for u,v in edges]
   weights = [G1[u][v]['weight'] for u,v in edges]
```

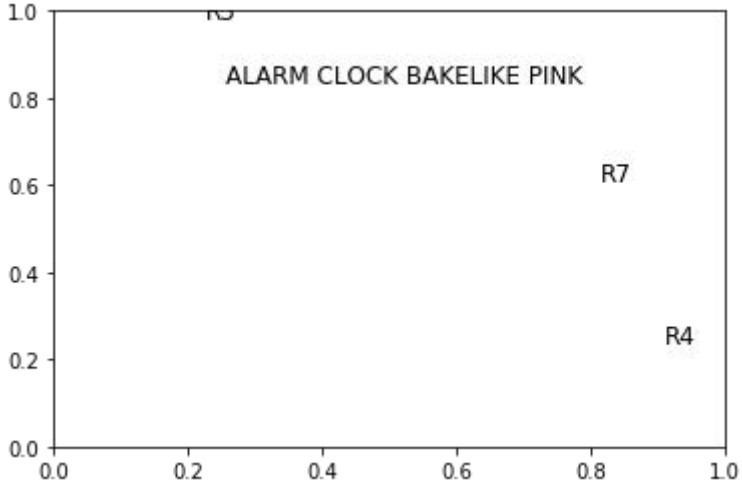
```
    pos = nx.spring_layout(G1, k=16, scale=1)
    nx.draw(G1, pos, edges=edges, node_color = color_map, edge_c
olor=colors, width=weights, font_size=16,
            with_labels=False)

    for p in pos:  # raise text positions
        pos[p][1] += 0.07
        nx.draw_networkx_labels(G1, pos)
        plt.show()

draw_graph (rules_mlxtend, 10)
```

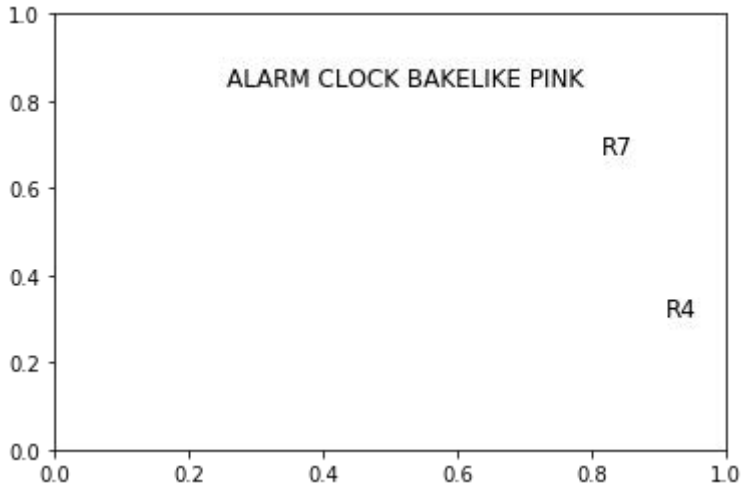





R5
ALARM CLOCK BAKELIKE PINK
R7
R4
0.0
0.2
0.4
0.6
0.8
1.0


R5
ALARM CLOCK BAKELIKE PINK
R7
R4
0.0
0.2
0.4
0.6
0.8
1.0


R5
ALARM CLOCK BAKELIKE PINK
R7
R4
0.0
0.2
0.4
0.6
0.8
1.0


ALARM CLOCK BAKELIKE PINK
R7
R4
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0