DATA MINING

ABHIRUPA MITRA- 17BCE0437

Assessment-3

APRIORI ALGORITHM

Consider the Transaction Dataset given below in the link

https://drive.google.com/file/d/1XS0Q4ozQC60IzTjHBmZqn_qCnLppe96q/view?usp=sharing

Apply the Apriori algorithm to the dataset of transactions and identify the Identifying Frequently-Purchased Groceries.

Rules <- apriori (groceries, parameter = list

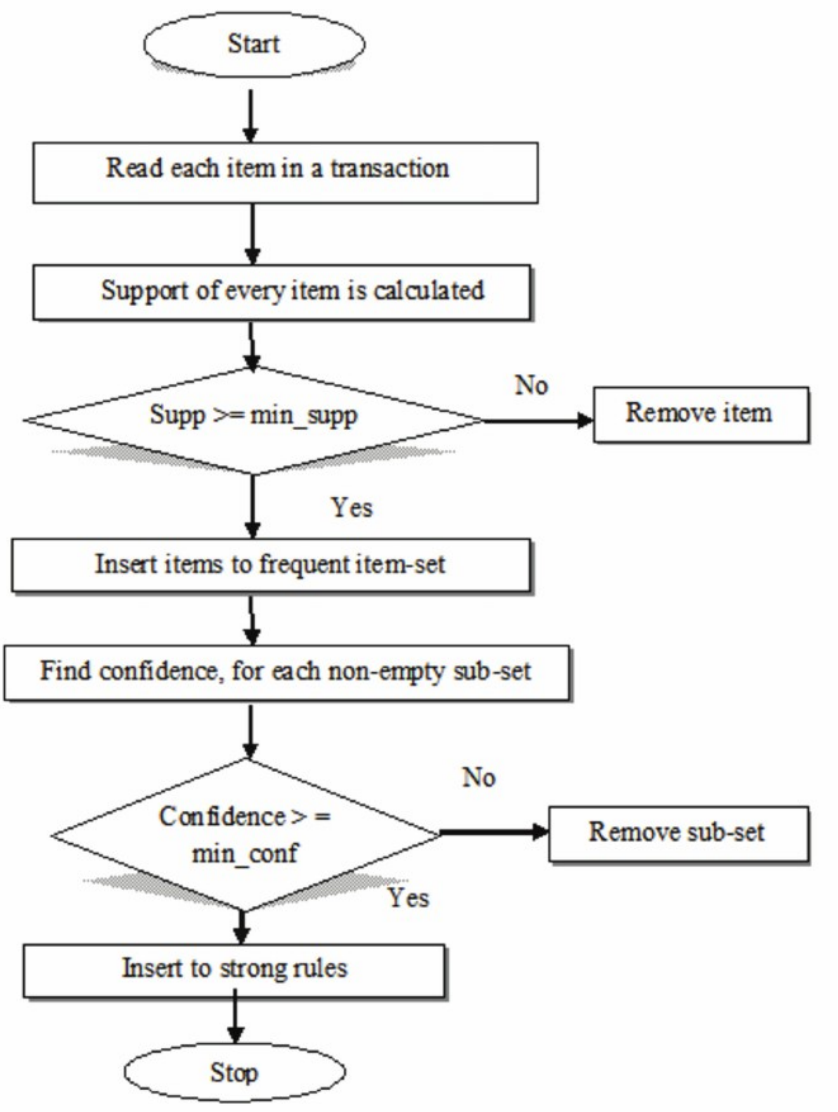
(support = 0.006, confidence = 0.25, minlen = 2))

Apriori Algorithm:
Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space.

Apriori Property –
All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that:

All subsets of a frequent itemset must be frequent(Apriori propertry).
If an itemset is infrequent, all its supersets will be infrequent.



Code:

In [95]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import  apriori

```

In [101]:
```
data = pd.read_csv("groceries.csv", sep=',', names = ["1", "2", "3", "4", "5", "6", "7", "8","9", "10", "11", "12","13"
data0 = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
          ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
          ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
          ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
          ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

# data.head(200)
data.shape
```

Out[101]: (9835, 32)

In [97]:
```
import math
records=data.values.tolist()

j=0
for i in range (0,9835):
    for j in range(0,32):
        if (isinstance(records[i][j],float) and  math.isnan(records[i][j])):
            break
    records[i]=records[i][0:j]
for i in range(0,9835):
    print(records[i],"\n")

```

```
['citrus fruit', 'semi-finished bread', 'margarine', 'ready soups']

['tropical fruit', 'yogurt', 'coffee']

['whole milk']

['pip fruit', 'yogurt', 'cream cheese', 'meat spreads']

['other vegetables', 'whole milk', 'condensed milk', 'long life bakery product']

['whole milk', 'butter', 'yogurt', 'rice', 'abrasive cleaner']

['rolls/buns']

['other vegetables', 'UHT-milk', 'rolls/buns', 'bottled beer', 'liquor (appetizer)']

['potted plants']

['whole milk', 'cereals']
```

In [98]:
```
from mlxtend.preprocessing import TransactionEncoder
```

In [108]:
```
# print(records)
# print(data0)
te = TransactionEncoder()
te_ary = te.fit(records).transform(records)
df = pd.DataFrame(te_ary, columns=te.columns_)
# print(df)
from mlxtend.frequent_patterns import apriori
apriori(df, min_support=0.006)
```

Out[108]:

	support	itemsets
0	0.008033	(0)
1	0.033452	(1)
2	0.017692	(7)
3	0.052466	(9)
4	0.033249	(10)
5	0.026029	(11)
6	0.080529	(12)
7	0.110524	(13)
8	0.064870	(15)
9	0.055414	(16)

10	0.027961	(17)
11	0.013218	(18)
12	0.008846	(19)
13	0.029893	(20)
14	0.077682	(21)
15	0.015048	(22)
16	0.010778	(24)
17	0.023284	(25)
18	0.021047	(27)
19	0.042908	(28)
20	0.049619	(29)
21	0.009049	(30)
22	0.082766	(31)
23	0.011388	(33)
24	0.058058	(35)
25	0.010269	(36)
26	0.039654	(40)
27	0.053279	(41)
28	0.037112	(45)
29	0.019217	(46)
...	...	...
717	0.009354	(123, 131, 166)
718	0.006304	(123, 139, 134)
719	0.008846	(123, 139, 166)
720	0.008643	(123, 139, 167)
721	0.010981	(123, 166, 158)
722	0.008744	(123, 158, 167)
723	0.007829	(162, 123, 166)
724	0.015557	(123, 166, 167)
725	0.007728	(131, 124, 166)
726	0.008134	(139, 124, 166)
727	0.011998	(124, 166, 158)
728	0.008134	(124, 158, 167)
729	0.009456	(162, 124, 166)
730	0.006406	(162, 124, 167)
731	0.014540	(124, 166, 167)
732	0.006711	(131, 139, 166)
733	0.007219	(131, 166, 158)
734	0.008744	(131, 166, 167)
735	0.006812	(139, 166, 134)
736	0.007829	(139, 166, 158)
737	0.006609	(139, 158, 167)
738	0.010473	(139, 166, 167)
739	0.007931	(162, 166, 158)
740	0.006202	(162, 158, 167)
741	0.015150	(166, 158, 167)
742	0.010880	(162, 166, 167)
743	0.006202	(123, 124, 166, 103)
744	0.007016	(124, 166, 158, 103)
745	0.007829	(167, 124, 166, 103)
746	0.007626	(167, 166, 158, 103)

747 rows × 2 columns

In [111]:
```
apriori(df, min_support=0.006, use_colnames=True)
```

Out[111]:

	support	itemsets
0	0.008033	(Instant food products)
1	0.033452	(UHT-milk)
2	0.017692	(baking powder)
3	0.052466	(beef)
4	0.033249	(berries)
5	0.026029	(beverages)
6	0.080529	(bottled beer)
7	0.110524	(bottled water)
8	0.064870	(brown bread)
9	0.055414	(butter)
10	0.027961	(butter milk)
11	0.013218	(cake bar)
12	0.008846	(candles)
13	0.029893	(candy)
14	0.077682	(canned beer)
15	0.015048	(canned fish)
16	0.010778	(canned vegetables)
17	0.023284	(cat food)
18	0.021047	(chewing gum)
19	0.042908	(chicken)
20	0.049619	(chocolate)
21	0.009049	(chocolate marshmallow)
22	0.082766	(citrus fruit)
23	0.011388	(cling film/bags)
24	0.058058	(coffee)
25	0.010269	(condensed milk)
26	0.039654	(cream cheese)
27	0.053279	(curd)
28	0.037112	(dessert)
29	0.019217	(detergent)
...	...	...
717	0.009354	(rolls/buns, sausage, whole milk)
718	0.006304	(shopping bags, rolls/buns, soda)
719	0.008846	(rolls/buns, soda, whole milk)
720	0.008643	(rolls/buns, yogurt, soda)
721	0.010981	(rolls/buns, tropical fruit, whole milk)
722	0.008744	(yogurt, rolls/buns, tropical fruit)
723	0.007829	(rolls/buns, whipped/sour cream, whole milk)
724	0.015557	(rolls/buns, yogurt, whole milk)
725	0.007728	(sausage, root vegetables, whole milk)
726	0.008134	(soda, root vegetables, whole milk)
727	0.011998	(root vegetables, tropical fruit, whole milk)
728	0.008134	(yogurt, root vegetables, tropical fruit)
729	0.009456	(root vegetables, whipped/sour cream, whole milk)
730	0.006406	(root vegetables, yogurt, whipped/sour cream)
731	0.014540	(root vegetables, yogurt, whole milk)
732	0.006711	(sausage, soda, whole milk)
733	0.007219	(sausage, tropical fruit, whole milk)
734	0.008744	(sausage, yogurt, whole milk)
735	0.006812	(shopping bags, soda, whole milk)
736	0.007829	(soda, tropical fruit, whole milk)
737	0.006609	(yogurt, soda, tropical fruit)
738	0.010473	(soda, yogurt, whole milk)
739	0.007931	(tropical fruit, whipped/sour cream, whole milk)
740	0.006202	(yogurt, tropical fruit, whipped/sour cream)
741	0.015150	(yogurt, tropical fruit, whole milk)
742	0.010880	(yogurt, whipped/sour cream, whole milk)
743	0.006202	(rolls/buns, root vegetables, other vegetables...
744	0.007016	(root vegetables, other vegetables, tropical f...
745	0.007829	(root vegetables, other vegetables, yogurt, wh...
746	0.007626	(tropical fruit, other vegetables, yogurt, who...

747 rows × 2 columns