Data Mining (CS634)

Mid Term Project

Kelvinkumar Gandhi UCID - ksg27 NJIT\_ID - 31457955

# **Apriori Algorithm:**

I have implemented apriori algorithm in python language. I've read the transaction data from text file.

For my implementation, I want briefly explain steps that I followed to implemented this algorithm.

- 1. Read transaction from text file.
- 2. Getting min support and min confidence from user input.
- 3. Finding frequent 1 itemset.
- 4. From frequent 1 item, finding frequent k itemsets.
- 5. From frequent k itemsets, I've generated association rules.
- 6. Display rules which satisfy min support and min confidence.

## **Program**

```
from itertools import combinations
def generateSupportCount(trans):
  item_list=[]
  item_dict={ }
  for item in trans:
     i = item.strip().split(',')
     for j in i:
       if j not in item_list:
          item_list.append(j)
          item_dict[j]=1
       else:
          item\_dict[j]=item\_dict[j] + 1
  #print(item_list)
  #print(item_dict)
  return item_list, item_dict
#Compute support frequent 1 itemset
def satisfySupport(tr_dict, min_sup,trans_dataset, tran_len):
  new_list = []
  for i,c in tr_dict.items():
     if c/tran_len >= min_sup:
       new_list.append([i])
       print(f'[{i}] => {c/tran_len} (support)')
  return new_list
```

```
def associateTrans(candidateItemSet, k):
  newAssociatedList=[]
  for i in range(len(candidateItemSet)):
     iList = candidateItemSet[i];
     for j in range(i+1,len(candidateItemSet)):
       ¡List = candidateItemSet[j];
       if k == 2:
          join = iList[(len(iList)-1)],jList[len(jList)-1]
          newAssociatedList.append(list(join))
       elif(iList[:len(iList)-1] == jList[:len(jList)-1]):
          join = iList[0:(len(iList)-1)],iList[len(iList)-1],jList[len(jList)-1]
          new = list(iList[0:(len(iList)-1)])
          new.append(iList[len(iList)-1])
          new.append(jList[len(jList)-1])
          newAssociatedList.append(new)
  return newAssociatedList
#Method use to compute candidate itemset and return list that satisfy min sup
def reduceBySupport(candidateItemSet, min_sup, trans_len):
  itemSet_dict = { }
  #first count support
  for t in trans:
    for n,itemSet in enumerate(candidateItemSet):
       index = itemSet
       if n not in itemSet dict:
          itemSet_dict[n] = 0
       c=0
```

```
for i in itemSet:
          #print("item value:", i)
          if i in t:
            c=c+1;
       if c == len(itemSet):
          itemSet\_dict[n] = itemSet\_dict[n] + 1
  #compare with min support
  reduced_list = []
  for index,count in itemSet_dict.items():
    if count/trans_len >= min_sup:
       reduced\_list.append(candidateItemSet[index])
       print(f'{candidateItemSet[index]} => {count/trans_len} (support)')
  return reduced_list
#check support of transactions in association rules
def checkSupportOfList(itemSet):
  #print(itemSet)
  \sup_{\cdot} count = 0;
  for t in trans:
    c=0:
    for i in itemSet:
       i = str(i)
       if i in t:
          c=c+1
    if c == len(itemSet):
       sup\_count = sup\_count + 1
  sup_val = sup_count/len(trans)
  #print(f'{itemSet} -> {sup_count}')
  return sup_val
```

```
#Check confidence of association rules
def checkConf(candi rule):
  temp_list = []
  combine_list = []
  for i in candi_rule:
     temp_list.append(list(i))
  for j in temp_list:
    for k in j:
       combine_list.append(k)
  rule_confidence = checkSupportOfList(combine_list)/checkSupportOfList(temp_list[0])
  if rule_confidence >= min_conf:
     conf_val=rule_confidence
     sup_val=checkSupportOfList(combine_list)
     return candi_rule, sup_val, conf_val
  return [],","
#Generate combinations of association rules from frequent itemsets
def gen_combi(itemlists):
  for ilist in itemlists:
    li_len = len(ilist)
    listset = set(tuple(ilist))
     tempFreqItemset = listset
     while li_len > 1:
       mainset = set()
       for c in combinations(tempFreqItemset, li_len - 1):
         mainset = set(c)
         rule = (mainset, listset - mainset)
```

```
valid_rules,sup_val,conf_val = checkConf(list(rule))
          #Result the final rules which satisfy min confidence
          if valid_rules != []:
            print(f'{valid_rules[0]} -> {valid_rules[1]} => confidence={conf_val} ,
support={sup_val}')
       li_len=li_len-1;
while True:
  try:
    inp_file = input("Please enter file name: ")
     file=open(inp_file, 'r')
     trans_dict={}
     trans=[]
     for line in file:
       (k,v) = line.strip().split('-')
       trans_dict[k]= v
       trans.append(v)
  except FileNotFoundError:
     print('Either file is missing or filename is wrong')
  print("\nDataset Transactions:\n")
  for tr in trans_dict:
    print(f'{tr} - {trans_dict[tr]}')
  min_sup= float(input("Please enter minimum support="))
  min_conf=float(input("Please enter minimum confidence="))
```

```
tr_list, tr_dict = generateSupportCount(trans)
print("\n")
print("Frequent 1 itemset:")
frequentOne = satisfySupport(tr_dict, min_sup, trans, len(trans))
k=2
freqSetList = []
while(frequentOne != []):
  #make join operation to associate the transactions
  candidate_k_itemSet = associateTrans(frequentOne, k)
  print(f'\nFrequent {k} itemset:')
  #Generate frequent k itemset
  frequentOne = reduceBySupport(candidate_k_itemSet, min_sup, len(trans))
  if frequentOne != []:
     frequent_k_itemSet = frequentOne
    freqSetList.append(frequent_k_itemSet)
  else:
    print("NULL")
  k = k + 1
#Generate Rules from frequent k itemsets
print('\n\nRules:\n')
if freqSetList != []:
  for iList in freqSetList:
     gen_combi(iList)
```

```
else:
    print("No association rules found")

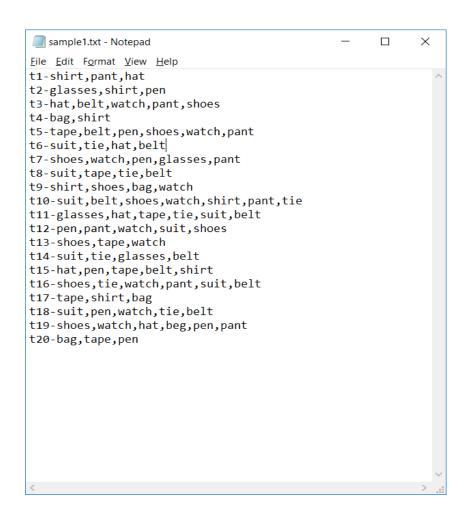
keepOn = input("Press any key to run again. | Press q to quit = ")

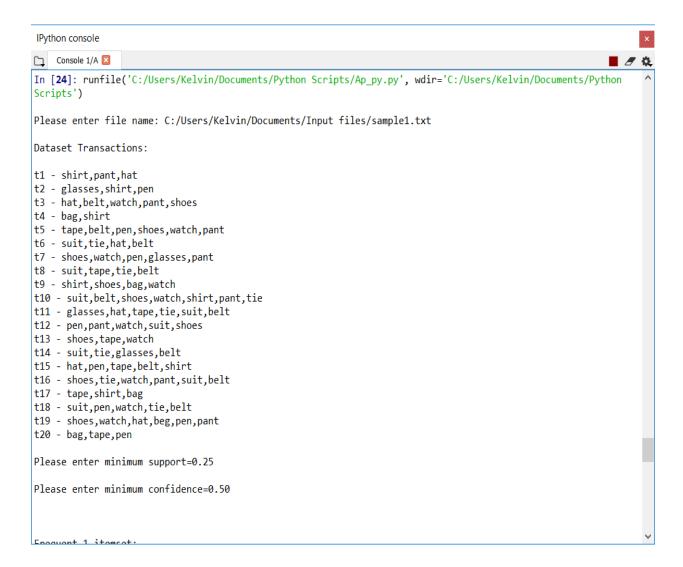
if keepOn in ('q','Q'):
    break;

else:
    continue;
```

## **Documentation:**

- My program is starting with asking user to give file name (with path). Then, program will show input transactions.
- Next user have to give min support and min confidence value at run time.
- Based on min support value, method named satisfySupport(tr\_dict, min\_sup, trans, len(trans)) will computer support of all transactions and return frequent 1 item set who satisfy min support.
- Now, we've frequent 1 itemset. So, starts with k=2, in while loop program will repeatedly call **reduceBySupport(candidate\_k\_itemSet, min\_sup, len(trans))** method which will generate frequent k itemset.
- reducedBySupport() method will take each list of transaction from candidate k itemset and compare it with original transaction. Then, it check for min support condition. Only returns those transactions who satisfy min support.
- All frequent k itemsets are stored in **freqSetList** List.
- After generating frequent k itemsets, for each itemset **gen\_combi(iList)** will generate association rules.
- Inside gen\_combi(iList) method, for each transaction it will call checkSupportOfList(itemSet) method and checkConf(candi\_rule) method to check min confidence of each association rule.
- Finally it will return association rules which satisfy both min support and min confidence.
- At the end, my program also ask user to continue with other dataset file or want to quit.





```
IPython console
Console 1/A 🗵
                                                                                                                                                                                   Please enter minimum confidence=0.50
 Frequent 1 itemset:
 [shirt] => 0.35 (support)
 [pant] => 0.4 (support)
 [hat] => 0.3 (support)
 [pen] => 0.4 (support)
 [belt] => 0.5 (support)
 [watch] => 0.5 (support)
 [shoes] => 0.45 (support)
 [tape] => 0.35 (support)
 [suit] => 0.4 (support)
 [tie] => 0.35 (support)
Frequent 2 itemset:
['pant', 'watch'] => 0.35 (support)
['pant', 'shoes'] => 0.35 (support)
['pen', 'watch'] => 0.25 (support)
['belt', 'watch'] => 0.25 (support)
['belt', 'suit'] => 0.35 (support)
['belt', 'tie'] => 0.35 (support)
['watch', 'shoes'] => 0.45 (support)
['suit', 'tie'] => 0.35 (support)
Frequent 2 itemset:
Frequent 3 itemset:
['pant', 'watch', 'shoes'] => 0.35 (support)
['belt', 'suit', 'tie'] => 0.35 (support)
Frequent 4 itemset:
NULL
```

```
IPython console
Console 1/A 🛛
                                                                                                                          Frequent 4 itemset:
NULL
Rules:
{'watch'} -> {'pant'} => confidence=0.7 , support=0.35
{\text{'watch'}} \rightarrow {\text{'pen'}} \Rightarrow \text{confidence=0.5}, \text{ support=0.25}
{'pen'} -> {'watch'} => confidence=0.625 , support=0.25 
{'watch'} -> {'belt'} => confidence=0.5 , support=0.25
{'belt'} -> {'watch'} => confidence=0.5 , support=0.25
{'belt'} -> {'suit'} => confidence=0.7 , support=0.35 {'tie'} -> {'belt'} => confidence=1.0 , support=0.35
{'belt'} -> {'tie'} => confidence=0.7 , support=0.35
{'shoes'} -> {'watch'} => confidence=1.0 , support=0.45 {'watch'} -> {'shoes'} => confidence=0.9 , support=0.45
{'tie'} -> {'suit'} => confidence=1.0 , support=0.35
{\suit', 'tie'} -> {\belt'} => \confidence=0.7, \support=0.35

{\suit', 'tie'} -> {\telt'} => \confidence=1.0, \support=0.35

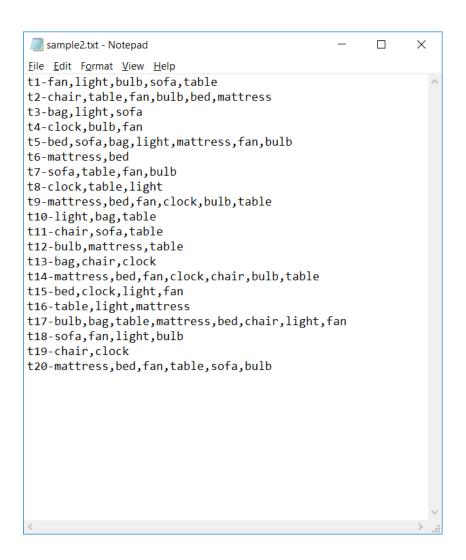
{\telt', 'belt'} -> {\telt'} => \confidence=1.0, \support=0.35

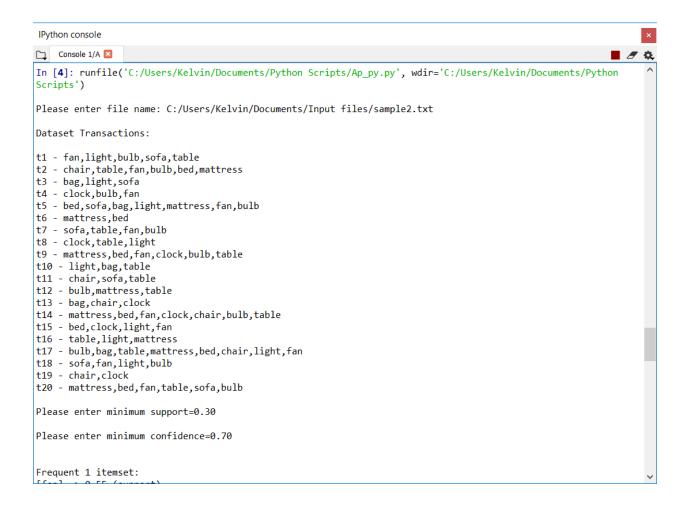
{\telt', 'belt'} -> {\telt'} => \confidence=0.874999999999999, \support=0.35

{\telt'} -> {\telt'} => \confidence=0.874999999999999, \support=0.35

{\telt'} -> {\telt'} => \confidence=1.0, \support=0.35

{\telt'} -> {\telt'} => \confidence=0.7, \support=0.35
```





```
IPython console
Console 1/A 🗵
                                                                                                                                                                                                                       Frequent 1 itemset:
 [fan] => 0.55 (support)
 [light] => 0.45 (support)
 [bulb] => 0.55 (support)
 [sofa] => 0.35 (support)
 [table] => 0.6 (support)
 [chair] => 0.3 (support)
 [bed] => 0.4 (support)
 [mattress] => 0.45 (support)
[clock] => 0.35 (support)
 Frequent 2 itemset:
Frequent 2 itemset:
['fan', 'bulb'] => 0.5 (support)
['fan', 'table'] => 0.35 (support)
['fan', 'bed'] => 0.35 (support)
['fan', 'mattress'] => 0.3 (support)
['bulb', 'table'] => 0.4 (support)
['bulb', 'bed'] => 0.3 (support)
['bulb', 'mattress'] => 0.35 (support)
['table', 'mattress'] => 0.35 (support)
['bed', 'mattress'] => 0.35 (support)
Frequent 3 itemset:
['fan', 'bulb', 'table'] => 0.35 (support)
['fan', 'bulb', 'bed'] => 0.3 (support)
['fan', 'bulb', 'mattress'] => 0.3 (support)
['fan', 'bed', 'mattress'] => 0.3 (support)
['bulb', 'table', 'mattress'] => 0.3 (support)
['bulb', 'bed', 'mattress'] => 0.3 (support)
 Frequent 4 itemset:
['fan', 'bulb', 'bed', 'mattress'] => 0.3 (support)
Frequent 5 itemset:
```

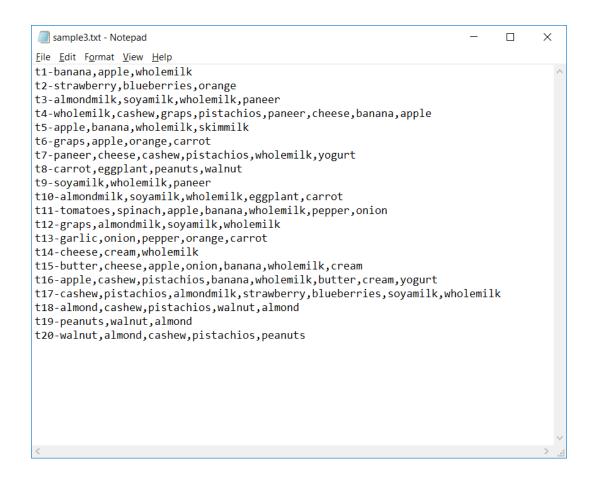
IPython console Console 1/A 🗵 **■** Ø ₺ Frequent 5 itemset: NULL Rules: {'fan'} -> {'bulb'} => confidence=0.9090909090909091, support=0.5 {'bulb'} -> {'fan'} => confidence=0.9090909090909091, support=0.5 {'bulb'} -> {'table'} => confidence=0.72727272727273, support=0.4 { 'table', 'mattress'} -> {'table'} => confidence=0.85/14285/14203/2, 50pp.

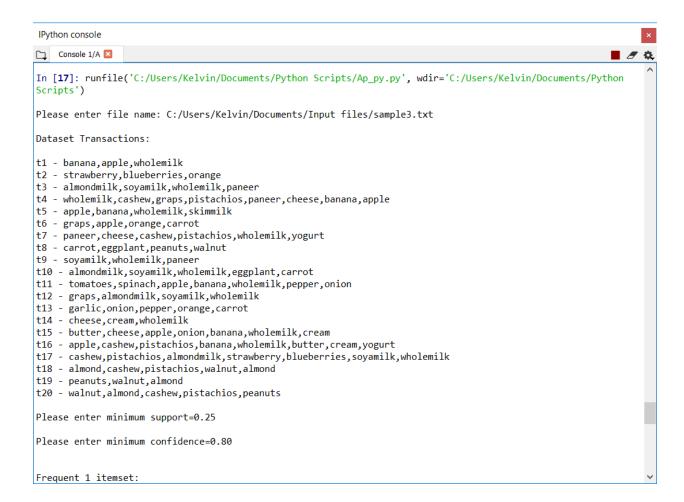
{'bulb', 'mattress'} -> {'mattress'} => confidence=1.0, support=0.3

{'bed', 'mattress'} -> {'bulb'} => confidence=0.8571428571428572, support=0.3

{'bulb', 'mattress'} -> {'bed'} => confidence=0.8571428571428572, support=0.3

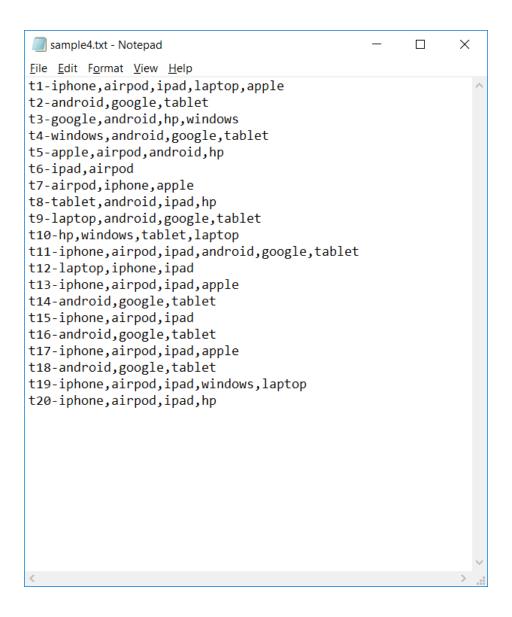
{'bed'} -> {'bulb', 'mattress'} => confidence=0.74999999999999, support=0.3

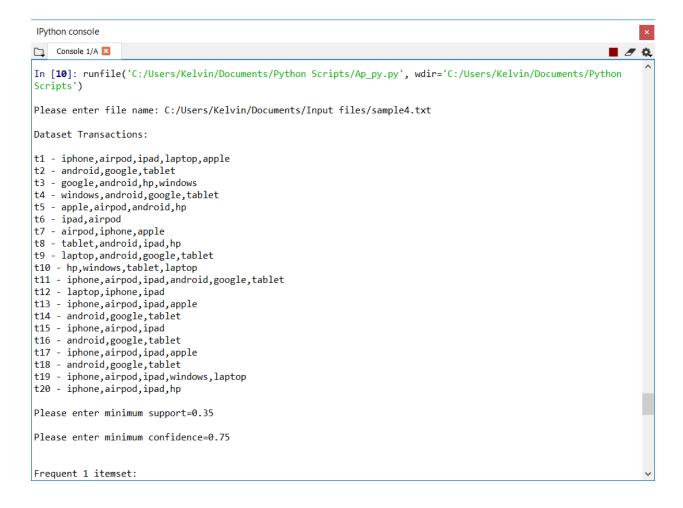




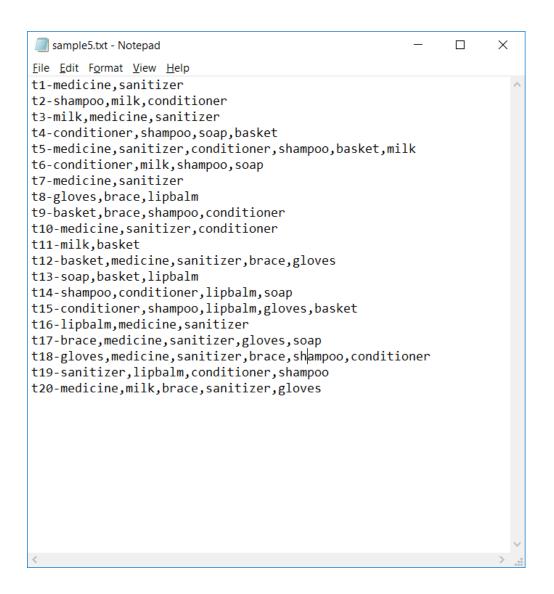
```
IPython console
Console 1/A 🗵
                                                                                                                                                Frequent 1 itemset:
[banana] => 0.3 (support)
 [apple] => 0.35 (support)
 [wholemilk] => 0.65 (support)
 [soyamilk] => 0.25 (support)
[cashew] => 0.3 (support)
[pistachios] => 0.3 (support)
Frequent 2 itemset:
['banana', 'apple'] => 0.3 (support)
['banana', 'wholemilk'] => 0.3 (support)
['apple', 'wholemilk'] => 0.3 (support)
['wholemilk', 'soupport)
['wholemilk', 'soupport)
['cashew', 'pistachios'] => 0.3 (support)
Frequent 3 itemset:
['banana', 'apple', 'wholemilk'] => 0.3 (support)
Frequent 4 itemset:
NULL
Rules:
{'banana'} -> {'apple'} => confidence=1.0 , support=0.3 
{'banana'} -> {'wholemilk'} => confidence=1.0 , support=0.3
{'apple'} -> {'wholemilk'} => confidence=0.8571428571428572 , support=0.3
{'soyamilk'} -> {'wholemilk'} => confidence=1.0 , support=0.25
{'pistachios'} -> {'cashew'} => confidence=1.0 , support=0.3
{'cashew'} -> {'pistachios'} => confidence=1.0 , support=0.3
{'apple', 'banana'} -> {'wholemilk'} => confidence=1.0 , support=0.3
{'apple', 'wholemilk'} -> {'banana'} => confidence=1.0 , support=0.3
{'banana', 'wholemilk'} -> {'apple'} => confidence=1.0 , support=0.3
```

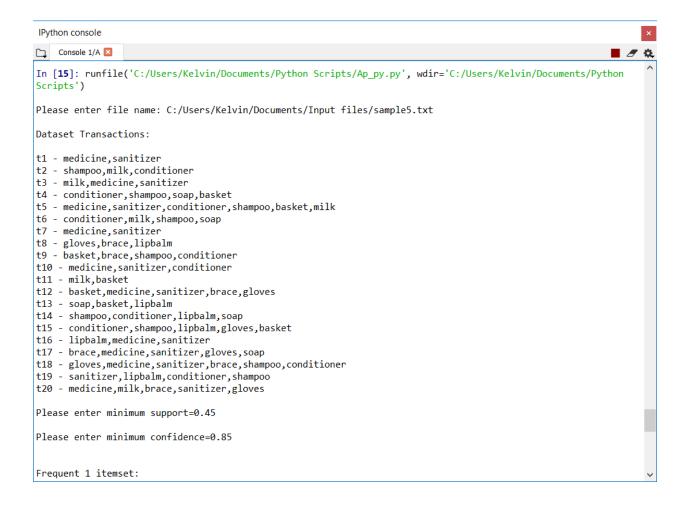
```
IPython console
Console 1/A
['banana', 'apple'] => 0.3 (support)
['banana', 'wholemilk'] => 0.3 (support)
['apple', 'wholemilk'] => 0.3 (support)
['wholemilk', 'soyamilk'] => 0.25 (support)
['cashew', 'pistachios'] => 0.3 (support)
Frequent 3 itemset:
['banana', 'apple', 'wholemilk'] => 0.3 (support)
Frequent 4 itemset:
NULL
Rules:
{'apple'} -> {'banana'} => confidence=0.8571428571428572 , support=0.3
{'banana'} -> {'apple'} => confidence=1.0 , support=0.3
{'banana'} -> {'wholemilk'} => confidence=1.0 , support=0.3 
{'apple'} -> {'wholemilk'} => confidence=0.8571428571428572 , support=0.3
{'soyamilk'} -> {'wholemilk'} => confidence=1.0 , support=0.25
{'pistachios'} -> {'cashew'} => confidence=1.0 , support=0.3
{'cashew'} -> {'pistachios'} => confidence=1.0 , support=0.3
{'cashew'} -> {'pistachios'} => confidence=1.0 , support=0.3
{'apple', 'banana'} -> {'wholemilk'} => confidence=1.0 , support=0.3
{'apple', 'wholemilk'} -> {'banana'} => confidence=1.0 , support=0.3
{'banana', 'wholemilk'} -> {'apple'} => confidence=1.0 , support=0.3
{'apple'} -> {'banana', 'wholemilk'} => confidence=0.8571428571428572 , support=0.3
{'banana'} -> {'apple', 'wholemilk'} => confidence=1.0 , support=0.3
Press any key to run again. | Press q to quit = |
```





```
IPython console
Console 1/A 🛚
                                                                                                                                                                 t20 - iphone,airpod,ipad,hp
Please enter minimum support=0.35
Please enter minimum confidence=0.75
Frequent 1 itemset:
[iphone] => 0.45 (support)
 [airpod] => 0.5 (support)
 [ipad] => 0.5 (support)
[android] => 0.5 (support)
[google] => 0.4 (support)
[tablet] => 0.45 (support)
Frequent 2 itemset:
Frequent 2 itemset:
['iphone', 'airpod'] => 0.4 (support)
['iphone', 'ipad'] => 0.4 (support)
['airpod', 'ipad'] => 0.4 (support)
['android', 'google'] => 0.4 (support)
['android', 'tablet'] => 0.4 (support)
['google', 'tablet'] => 0.35 (support)
Frequent 3 itemset:
['iphone', 'airpod', 'ipad'] => 0.35 (support)
['android', 'google', 'tablet'] => 0.35 (support)
Frequent 4 itemset:
NULL
Rules:
{'airpod'} -> {'iphone'} => confidence=0.8 , support=0.4
```





```
IPython console
Console 1/A 🗵
                                                                                                                                                             Please enter minimum support=0.45
Please enter minimum confidence=0.85
Frequent 1 itemset:
[medicine] => 0.5 (support)
[sanitizer] => 0.55 (support)
[shampoo] => 0.45 (support)
[conditioner] => 0.5 (support)
Frequent 2 itemset:
['medicine', 'sanitizer'] => 0.5 (support)
['shampoo', 'conditioner'] => 0.45 (support)
Frequent 3 itemset:
NULL
Rules:
{'medicine'} -> {'sanitizer'} => confidence=1.0 , support=0.5
{'sanitizer'} -> {'medicine'} => confidence=0.9090909090909091 , support=0.5
{'shampoo'} -> {'conditioner'} => confidence=1.0 , support=0.45
{'conditioner'} -> {'shampoo'} => confidence=0.9 , support=0.45
Press any key to run again. | Press q to quit =
```