Name-Rebecca Dias Class- TE-CMPNA Roll No-019 PID-182027
EXPERIMENT NO. 6

Aim: Implementation of Apriori algorithm as Association Rule Mining

Theory:

What is Association Mining?
Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly, so that it can be more profitable. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.

What is Apriori Algorithm?
This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the item set efficiently. It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

Explain Apriori Algorithm (pseudo code) with example, solve the below problem manually.

Algorithm:

Step-1: K=1
 Create a table containing support count of each item present in dataset – Called C1(candidate set)
 compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Step-2: K=2

 Generate candidate set C2 using L1 (this is called join step).

 Condition of joining Lk-1 and Lk-1 is that it should have (K-2) elements in common.

 Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent. Check for each itemset)

 Now find support count of these itemsets by searching in dataset.

 compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

Step-3:

 Generate candidate set C3 using L2 (join step). Condition of joining Lk-1 and Lk-1 is that it should have (K-2) elements in common.

 So here, for L2, first element should match.So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}

 Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)

 find support count of these remaining itemset by searching in dataset.

 Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Step-4:

 Generate candidate set C4 using L3 (join step). Condition of joining Lk-1 and Lk-1 (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.

 Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent).

- So no itemset in C4.
- We stop here because no frequent itemsets are found further.

- Consider the five transactions given below. If minimum support is 30% and minimum confidence is 80%, determine the frequent item sets and association rules using Apriori algorithm.

Transaction	Items
T1	Bread, Jelly, Butter
T2	Bread, Butter
T3	Bread, Milk, Butter
T4	Coke, Bread
T5	Coke, Milk

Solution:

Q → min support = 30%, min confi = 80%.

Transaction	Items
T1	Bread, Jelly, Butter
T2	Bread, Butter
T3	Bread, Milk, Butter
T4	Coke, Bread
T5	Coke, Milk

C_1

↓ scan D

	Sup
Bread	4
Jelly	1
Butter	3
Milk	2
Coke	2

Br = Bread
Bt = Butter
M = Milk
C = Coke

↓ min sup 30%.

$\{Br\}$ L_1

$\{Br, Bt\}$ 3 scan D

$\{Br, M\}$ 1 ← C_2

$\{Br, C\}$ 1

$\{Bt, M\}$ 1

$\{Bt, C\}$ 0

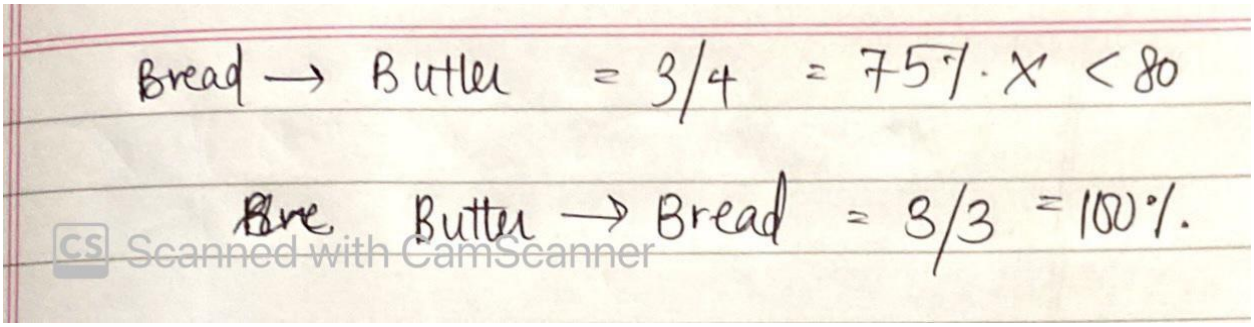
$\{M, C\}$ 1

$\{Bt\}$	
$\{M\}$	
$\{C\}$	

↓ min sup 30%.

L_2

$\{Br, Bt\}$ 3



Implementation:

- Write a program to recognize frequent buying patterns from a given set of transactions and association rule. (use the above data given for implementation)
- Take dataset, support and confidence threshold at run time
- Display frequent item set (display the steps as well)
- Display association rule with confidence value for each
- Prune and display final result

Code:

```
from tabulate import tabulate
num=80
uniques=set()
items=[] #List of list of items
fp=input("Enter the file path: ")
with open(fp,'r') as file_ref:
    for line in file_ref.readlines():
        t_items=[]
        fline=line.strip().split(",")
        for item in fline[1:]:
            uniques.add(item)
            t_items.append(item)
        items.append(t_items)

support=[]
min_support=float(input("Enter minimum support: "))
min_confidence=float(input("Enter minimum confidence: "))
for i in uniques:
    count=0
    for item_list in items:
```

```
        for item in item_list:
            if (i==item):
                count+=1
    support.append([i,(float(count)/len(items))*100])

def display(support,filtering):
    print("\nSupport "+filtering+" filtering:")
    print(tabulate(support,headers=["Item","Support (in %)"],tablefmt="pretty"))

display(support,"before")

support=[s_list for s_list in support if s_list[1]>=min_support]

display(support,"after")

uniques_list=[set([s_list[0]]) for s_list in support]

def apriori(uniques_list,support):
    while(len(support)!=1):
        u_list=[]
        u_list_s=[]
        for item1 in uniques_list:
            for item2 in uniques_list:
                if (item1!=item2 and list(item1.union(item2)) not in u_list):
                    u_list.append(list(item1.union(item2)))
                    u_list_s.append(item1.union(item2))
        uniques_list=u_list_s
        support=[]
        for item_list in u_list:
            count=0
            for t_list in items:
                if (all(x in t_list for x in item_list)):
                    count+=1
            support.append([item_list,(float(count)/len(items))*100])
        display(support,"before")
        support=[s_list for s_list in support if s_list[1]>=min_support]
        display(support,"after")
    return support

support=apriori(uniques_list,support)

def association(support,num_t,min_confidence):
    a_list=[]
    for s_list in support:
        for i in range(len(s_list[0])):
```

```
            count_1=0
            count_2=0
            item=s_list[0][i]
            another_list=[x for x in s_list[0] if x!=item]
            for i_list in items:
                if(item in i_list):
                    count_1+=1
                if(all(x in i_list for x in another_list)):
                    count_2+=1
            if ([[item],another_list,(s_list[1]*num_t)/count_1] not in a_list):
                a_list.append([[item],another_list,(s_list[1]*num_t)/count_1])
            if ([another_list,[item],(s_list[1]*num_t)/count_2] not in a_list):
                a_list.append([another_list,[item],(s_list[1]*num_t)/count_2])
    print("\nConfidence before filtering:")
    print(tabulate(a_list,headers=["Items","Associated_items","Confidence (in %)"
],tablefmt="pretty"))
    print("\nConfidence after filtering:")
    a_list=[x for x in a_list if x[2]>=min_confidence]
    print(tabulate(a_list,headers=["Items","Associated_items","Confidence (in %)"
],tablefmt="pretty"))

association(support,len(items),min_confidence)
```

Output:

```
C:\Users\toshiba\Desktop\Downloads>python exp6.py
Enter the file path: C:\Users\toshiba\Desktop\Downloads\data.txt
Enter minimum support: 30
Enter minimum confidence: 80

Support before filtering:
+--------+----------------+
|  Item  | Support (in %) |
+--------+----------------+
| Butter |      60.0      |
|  Milk  |      40.0      |
|  Coke  |      40.0      |
| Jelly  |      20.0      |
| Bread  |      80.0      |
+--------+----------------+

Support after filtering:
+--------+----------------+
|  Item  | Support (in %) |
+--------+----------------+
| Butter |      60.0      |
|  Milk  |      40.0      |
|  Coke  |      40.0      |
| Bread  |      80.0      |
+--------+----------------+

Support before filtering:
+---------------------+----------------+
|        Item         | Support (in %) |
+---------------------+----------------+
| ['Butter', 'Milk']  |      20.0      |
| ['Coke', 'Butter']  |      0.0       |
| ['Butter', 'Bread'] |      60.0      |
|  ['Coke', 'Milk']   |      20.0      |
|  ['Milk', 'Bread']  |      20.0      |
|  ['Coke', 'Bread']  |      20.0      |
+---------------------+----------------+

Support after filtering:
+---------------------+----------------+
|        Item         | Support (in %) |
+---------------------+----------------+
| ['Butter', 'Bread'] |      60.0      |
+---------------------+----------------+
```

```
Confidence before filtering:
+------------+-------------------+--------------------+
|   Items    | Associated_items  | Confidence (in %)  |
+------------+-------------------+--------------------+
| ['Butter'] |    ['Bread']      |       100.0        |
| ['Bread']  |    ['Butter']     |       75.0         |
+------------+-------------------+--------------------+

Confidence after filtering:
+------------+-------------------+--------------------+
|   Items    | Associated_items  | Confidence (in %)  |
+------------+-------------------+--------------------+
| ['Butter'] |    ['Bread']      |       100.0        |
+------------+-------------------+--------------------+
```

Conclusion:

Conclusion:

→ Summary of Experiment understanding

Apriori algorithm is applied for a level wise search where K-frequent itemsets are used to find K+1 itemsets. During this experiment, we wrote a program of the apriori algorithm in which we obtained the required output

→ Importance

Apriori algorithm is used for mining frequent itemsets and designing association rules from dataset.

→ Applications

- Used in recommender systems and auto complete features
- Used for analysis of patient database.