MID-TERM PROJECT - CS 634 DATA MINING

Instructed by Professor Jason T.L. Wang

APRIORI ALGORITHM

By : **Roopali Rajaram Sarode**

NJIT ID : 31570495

Email Id : rs366@njit.edu

Table of contents

Introduction

What is association rule mining ?

Association mining is finding frequent patterns, associations, or correlations among sets of items or objects in transactional databases. Association rules are critical in data mining for analyzing and forecasting consumer behavior. Customer analytics, market basket analysis, product clustering, catalog design, and shop layout are all examples of where they're employed.

The main applications of association rule mining:

- Basket data analysis - is to analyze the association of purchased items in a single basket or single purchase as per the examples given above.
- Cross marketing - is to work with other businesses that complement your own, not competitors. For example, vehicle dealerships and manufacturers have cross marketing campaigns with oil and gas companies for obvious reasons.
- Catalog design - the selection of items in a business' catalog are often designed to complement each other so that buying one item will lead to buying of another. So these items are often complements or very related.

What is the Apriori algorithm ?

Apriori Algorithm is one of the algorithms used for transaction data in Association Rule Learning. It allows us to mine the frequent itemset in order to generate association rules between them. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

- Find the sets of items that have minimum support (frequent itemsets) starting from 1-itemsets and expanding to k-itemsets; if a j-itemset is already not frequent, then do not consider any superset of it.
- Use the frequent itemsets to generate all association rules.

Terminologies related to Apriori algorithm :

- Support : refers to how often a given rule appears in the database being mined.
- Confidence : refers to the amount of times a given rule turns out to be true in practice.
- Frequent Itemset : Frequent itemsets are a form of frequent pattern. Given examples that are sets of items and a minimum frequency, any set of items that occurs at least in the minimum number of examples is a frequent itemset.

- Association rules : finding the rules that may govern associations and causal objects between sets of items.

What are the principles of Apriori algorithm ?

The two Apriori principles are :

- Any subset of a frequent itemset must be frequent.
- Any superset of a non-frequent itemset must be non-frequent.

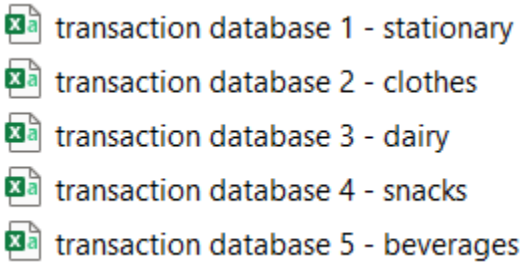
What is the Brute force approach?

A brute force approach is an approach that finds all the possible solutions to find a satisfactory solution to a given problem. The brute force algorithm tries out all the possibilities till a satisfactory solution is not found.

Input Dataset

5 databases containing 20 transactions have been used as a dataset for the algorithm. Each transaction contains a few items.

All transaction files are in csv format.



Content of transaction database 1 - stationary

Pen	Marker	Scissors			
Paper	Folder	Highlighter			
Pencil	Sharpner				
Eraser	Pencil	Sharpner	Stapler	Folder	
Marker	Pen	Highlighter			
Stapler	Pencil	Sharpner	Eraser		
Folder					
Scissors	Paper	Crayons			
Sharpner	Pencil	Watercolors	Eraser	Stapler	Crayons
Highlighter	Notebook	Pen	Pencil	Eraser	
Notepad	Pencil	Pen	Stickynotes	Eraser	
Binder	Folder	Paper	Marker	Labels	
Labels	Notebook	Pen			
Canvas	Watercolors	Crayons	Eraser		
Pen	Notebook	Pencil			
Scissors	Labels	Folder	Marker		
Notepad	Highlighter	Pen	Stapler	Binder	
Paper	Scissors	Folder			
Pencil	Sharpner	Pen			
Ruler	Pencil	Sharpner	Eraser		

Content of transaction database 2- clothes

Shirt	Basket	Gloves	Cap			
Jacket	Gloves	Shirt	Bag	Ball		
Cap	Bag	Ball				
Jacket	Bag	Ball	Gloves	Shirt	Bottle	Cap
Jacket	Gloves	Shirt	Basket			
Gloves	Bag	Ball	Bottle	Cap		
Shoes	Bottle	Socks	Cap			
Jacket	Bottle	Socks				
Socks	Bag	Ball	Bottle			
Jacket	Bottle	Basket	Bag	Ball		
Bag	Ball	Shirt	Socks	Cap		
Socks	Bottle	Gloves	Bag	Ball	Cap	
Basket	Gloves	Jacket	Bottle	Bag	Ball	
Cap	Bottle	Socks	Bag	Ball	Shoes	
Bag	Ball	Gloves	Cap	Basket		
Shirt	Gloves	Bag	Ball	Cap	Shoes	
Bag	Ball	Jacket	Shirt	Socks	Gloves	Shoes
Shirt	Shoes	Socks	Basket	Bottle		
Shirt	Bag	Ball	Cap	Socks	Basket	Jacket
Bag	Ball	Socks	Bottle	Cap		

Content of transaction database 3- dairy

Milk	Dips	Butter					
IcecreamDips	Cheese	Milk	Cream	Butter			
Cheese	Doughs	Yogurt	Icecream	Butter			
Cheese	Eggs	Milk	Icecream	Butter	Cream	Yogurt	Doughs
Icecream	Doughs	Cream	Butter	Dips			
Butter	Dips	Margarine	Doughs				
Cream	Icecream	Doughs	Cheese	Butter	Yogurt		
Cheese							
Yogurt	Icecream	Margarine	Milk	Eggs	Butter		
Margarine	Butter	Dips	Milk				
Doughs	Margarine	Milk					
Yogurt	Margarine	Icecream	Cream				
Dips							
Butter	Yogurt	Margarine					
Eggs	Dips	Yogurt	Milk	Icecream	Cheese	Doughs	
Dips	Milk	Cheese	Cream				
Milk	Butter	Dips	Margarine	Yogurt			
Cheese	Butter						
Eggs	Cheese	Doughs	Milk	Icecream			
Milk	Icecream	Dips	Cheese	Cream			

Content of transaction database 4- snacks

Crackers	Nuts	Chips	Cookies				
Pretzels	Pudding	MeatSticks	Spreads				
Pudding	Pretzels	Nuts	Crackers	Dips			
MeatSticks	Pretzels						
Chips	Cookies	Pudding	Popcorn	Crackers	Dips	Pretzels	Spreads
Chips	Cookies	Pretzels	Pudding				
Pudding	Chips	Cookies	Nuts	Popcorn	Pretzels	Dips	
MeatSticks	Pretzels	Dips					
Popcorn							
MeatSticks	Pudding	Crackers	Popcorn				
MeatSticks	Popcorn	Pretzels	Dips	Chips	Cookies		
Chips	Cookies	Crackers	Dips	Nuts	Pretzels	Popcorn	
MeatSticks							
Pretzels	Pudding	Nuts	Crackers				
Chips	Cookies	Pretzels	Dips	Pudding			
Chips	Cookies	Dips	Nuts	MeatSticks	Pudding	Pretzels	
Chips	Cookies	Dips	Nuts	Crackers			
Crackers							
Pretzels	Chips	Cookies	Crackers	Pudding	Popcorn	Dips	
Popcorn	Chips	Cookies	Nuts				

Content of transaction database 5- beverages

Water	Cocoa	Cider			
Cocoa	Water	Juice			
Cocktail					
Tea	Cocktail	Wine	Coffee	Water	
Cocktail	Juice	ProteinShake	Soda		
Tea	ProteinShake	Coffee	Soda		
Water					
Water	Cocoa	Wine			
Juice	Cocoa	Water	Coffee	Tea	Cider
Cocoa	Tea	Water	Coffee	Wine	
Soda	Tea	Coffee	ProteinShake	Juice	
Cocoa	Cocktail	Juice	ProteinShake	Water	
Coffee	Tea	ProteinShake	Cocktail	Water	Cider
Cocktail	Coffee	ProteinShake	Wine	Juice	
Soda	Coffee	Cider			
ProteinShake	Water	Coffee			
ProteinShake	Cocktail	Ciderss			
Coffee	Cocktail	Cocoa	Wine	ProteinShake	Water
Juice	Coffee	Soda			
Soda					

Source Code for Apriori :

```
import math
from time import process_time

# function to calculate the support value for item and itemsets
def itemsupport(total_transaction = 0, matching_transaction = 0):
    return ((matching_transaction/total_transaction)*100)

# function to calculate the confidence value for rules
def itemconfidence(matching_itemset = 0, matching_left_item = 0):
    return((matching_itemset/matching_left_item)*100)

def apriori_algorithm():
    t1_start = process_time()
    #dictionary containing transaction from the user created file
    original_transactions = {}
    # list of unique items from all transactions
    unique_items = []

    file_path= input("\n Enter transaction filename including extension of the file :
")
    transaction_file=open(file_path,"r")

    k=0
    for items in transaction_file:
        original_transactions[k] = [temp.strip() for temp in (items.split(','))]

        for element in original_transactions[k]:
            if(element not in unique_items):
                unique_items.append(element)
        k+=1

    print("\n****************Transactions****************\n")
    temp = 0

    for items in original_transactions:
        print("\nTransaction " + str(temp) + ": " + str(original_transactions[items])
+ " ")
        temp+=1

    min_support = int(input("\nMinimum support value: "))
    min_confidence = int(input("\nMinimum confidence value: "))

    #list of combinations of items that satify minimum support
    candidate_list=[]
    temp_queue1=[]

    for item in unique_items:
        # adding item in temp queue as a list of lists
        temp_queue1.append([item])
```

```
    exit_flag =0

    # This loop will find candidate sets satisfying minimum support for the
association rule
    while(len(temp_queue1)>=2):
        temp_queue2=[]
        #resetting exit flag inside the loop
        exit_flag=0

        for item in temp_queue1:
            # variable to keep track of number of item sets
            count=0
            for element in original_transactions:
                if(all(data in original_transactions[element] for data in item)):
                    count = count + 1

            #ignore if the support for the itemset is less than minimum support given
by user
            if(itemsupport(k,count))<min_support:
                continue

            # if support of itemset is more than user given support then append the
item in candidate queue and temp2 queue
            candidate_list.append(item)
            temp_queue2.append(item)

        #checking for case where item set is 1
        if(len(temp_queue2))<2:
            temp_queue1=temp_queue2
            continue
        else:
            temp_queue1=[]

        # Generating possible combinations of itemsets
        for i in range(0,len(temp_queue2)-1):
            for j in range(i+1,len(temp_queue2)):
                queue1=temp_queue2[i].copy()
                queue1.extend(temp_queue2[j])
                queue1=list(dict.fromkeys(queue1))

                # checking fro redundancy
                duplicate_flag1=0
                for index in range(0,len(temp_queue1)):
                    if(len(queue1)==len(temp_queue1[index]) and all (data in
temp_queue1[index] for data in queue1)):
                        duplicate_flag1=1 # set duplicate flag to one if one
combination is repeated in queue1
                        break

                if duplicate_flag1 == 0:
                    temp_queue1.append(queue1)
                    exit_flag=1 # setting exit flag to come out of the loop if no
duplicate found

    if exit_flag ==1: # if no duplicates found then enter the below loop
```

```
        for item in temp_queue1:
            count=0 # calculating the number of item sets in order to find out the
support of the set
            for element in original_transactions:
                if(all(data in original_transactions[element] for data in item)):
                    count += 1

            if(itemsupport(k,count))<min_support:
                break

            # If Itemset matches minimum support criteria update the candidate queue
            candidate_list.append(item)

    # Printing candidate for association rule.
    print("\nCandidate list: \n")
    for data in candidate_list:
        print(data)

    # maintiaining a dictionary for the resulting association rules from frequent
itemsets
    # key will be left hand side of rule and value will be right hand side of rule.
    rules={}

    for candidate in candidate_list:
        # since we can not take itemset with length less than 2 for creating
association rules
        if(len(candidate))<2:
            continue

        # Each iteration  of 'i' specifies the items on left hand side.
        for i in range(1,len(candidate)):
            # to store the left hand side of the association rule
            rule_LHS= []
            # to store the right hand side of the association rule
            rule_RHS= []
            # to store the current index of the item of LHS of the rule
            current_item_index= []

            for j in range(i):
                current_item_index.append(j)

            for j in current_item_index:
                rule_LHS.append(candidate[j])

            for j in range(len(candidate)):
                if j not in current_item_index:
                    rule_RHS.append(candidate[j])

            #count of total number of transaction itemsets in all
            count_all=0
            # count of number of appeareance of left hand side item set in transaction
            count_left=0

            for element in original_transactions:
                if(all(data in original_transactions[element] for data in rule_LHS)):
                    count_left+=1
```

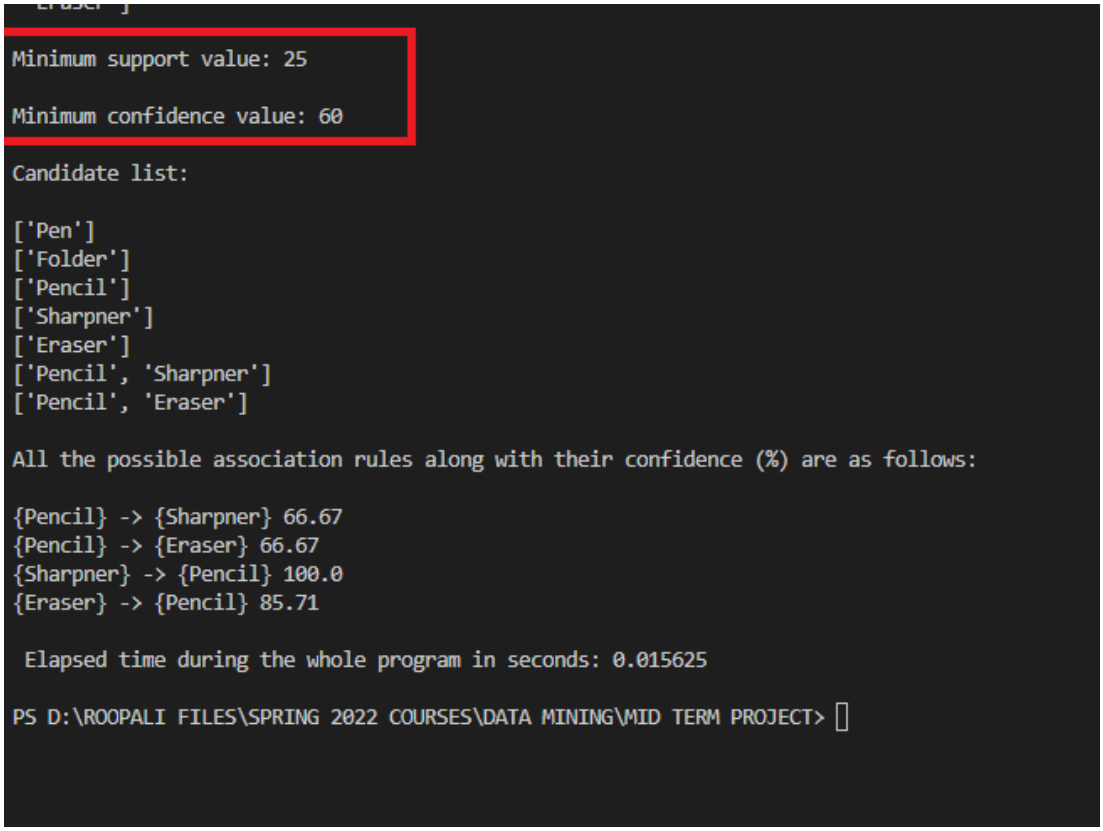
```
                    if(all(data in original_transactions[element] for data in
rule_RHS)):
                        count_all+=1

            #Calculating confidence of the rule
            if(itemconfidence(itemsupport(k,count_all), itemsupport(k,count_left))) >=
min_confidence:
                # Storing association rules if minimum confidence criteria is
achieved.
                key = ""
                value = ""
                rule_confidence = round(itemconfidence(itemsupport(k,count_all),
itemsupport(k,count_left)),2)

                for j in rule_LHS:
                    key = key + " " + j

                key = key.strip()

                for j in rule_RHS:
                    value = value + " " + j

                value = value.strip()

                if key in rules.keys():
                    rules[key].append([value,rule_confidence])

                else:
                    temp=[]
                    temp.append([value,rule_confidence])
                    rules[key]=temp

            # last index position from list of items currently referred to
            last_index_position=len(current_item_index)-1

            # Calculating possible combinations.
            item_combinations= math.factorial(len(candidate))/math.factorial(i)

            while(item_combinations>0):
                # Resetting LHS and RHS variables
                rule_LHS=[]
                rule_RHS=[]

                for j in range(i-1,-1,-1):
                    if(current_item_index[j] <
(len(candidate))-(last_index_position-j)-1):
                        temp_position= current_item_index[j] + 1

                        for m in range(j,i):
                            current_item_index[m] = temp_position
                            temp_position = temp_position + 1
                        break
                    else:
                        continue

                # Storing left hand side items of association rules
```

```
                for j in current_item_index:
                    rule_LHS.append(candidate[j])

                # Storing right hand side items of association rules
                for j in range(len(candidate)):
                    if j not in current_item_index:
                        rule_RHS.append(candidate[j])

                # Checking confidence constraint.
                # calculating number of transaction items set in whole
                count_all = 0
                # calculating number of transaction items set on left
                count_left = 0
                for element in original_transactions:
                    if(all(data in original_transactions[element] for data in
rule_LHS)):
                        count_left += 1
                        if(all(data in original_transactions[element] for data in
rule_RHS)):
                            count_all += 1

                if (itemconfidence(itemsupport(k, count_all),itemsupport(k,
count_left)) < min_confidence):
                    item_combinations = item_combinations - 1
                    continue

                # Storing association rules if minimum confidence criteria is
achieved.
                key = ""
                value = ""
                for j in rule_LHS:
                    key = key + " " + j

                key = key.strip()

                for j in rule_RHS:
                    value = value + " " + j

                value = value.strip()

                rule_confidence = round(itemconfidence(itemsupport(k,count_all),
itemsupport(k,count_left)),2)

                if key in rules.keys():
                    if (value not in [item[0] for item in rules[key]]):
                        rules[key].append([value,rule_confidence])

                else:
                    temp = []
                    temp.append([value,rule_confidence])
                    rules[key] = temp

                item_combinations = item_combinations - 1

    print("\nAll the possible association rules along with their confidence (%) are as
follows: \n")
```

```
    for key in rules:
        for value in rules[key]:
            print("{" + key + "} -> {" + value[0] + "} " + str(value[1]))

    t1_stop = process_time()
    print("Elapsed time during the whole program in seconds:",t1_stop-t1_start)

apriori_algorithm()
```

Apriori Output 1:

Dataset → transaction database 1 - stationary.csv
Support → 25 %
Confidence → 60 %



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Transaction 14: ['Pen', 'Notebook', 'Pencil']

Transaction 15: ['Scissors', 'Labels', 'Folder'

Transaction 16: ['Notepad', 'Highlighter', 'Pen

Transaction 19: ['Ruler', 'Pencil', 'Sharpner',
 'Eraser']

Minimum support value: 25

Minimum confidence value: 60

Candidate list:

['Pen']
['Folder']
['Pencil']
['Sharpner']
['Eraser']
['Pencil', 'Sharpner']
['Pencil', 'Eraser']

All the possible association rules along with their confidence (%) are as follows:

{Pencil} -> {Sharpner} 66.67
{Pencil} -> {Eraser} 66.67
{Sharpner} -> {Pencil} 100.0
{Eraser} -> {Pencil} 85.71

 Elapsed time during the whole program in seconds: 0.015625
```

```
Transaction 14: ['Pen', 'Notebook', 'Pencil']

Transaction 15: ['Scissors', 'Labels', 'Folder', 'Marker']

Transaction 16: ['Notepad', 'Highlighter', 'Pen', 'Stapler', 'Binder']

Transaction 17: ['Paper', 'Scissors', 'Folder']

Transaction 18: ['Pencil', 'Sharpner', 'Pen']

Transaction 19: ['Ruler', 'Pencil', 'Sharpner', 'Eraser']

Candidate list:

['Pen']
['Folder']
['Pencil']
['Sharpner']
['Eraser']
['Pencil', 'Sharpner']
['Pencil', 'Eraser']

All the possible association rules along with their confidence (%) are as follows:

{Pencil} -> {Sharpner} 66.67
{Pencil} -> {Eraser} 66.67
{Sharpner} -> {Pencil} 100.0
{Eraser} -> {Pencil} 85.71

Apriori took  0.0090024471282958 98 seconds
```

Apriori Output 2:
Dataset → transaction database 2 - clothes.csv
Support → 30 %
Confidence → 70 %

```
 Enter transaction filename including extension of the file : transaction database 2 - clothes.csv
***************Transactions***************

Transaction 0: ['Shirt', 'Basket', 'Gloves', 'Cap']
Transaction 1: ['Jacket', 'Gloves', 'Shirt', 'Bag', 'Ball']
Transaction 2: ['Cap', 'Bag', 'Ball']
Transaction 3: ['Jacket', 'Bag', 'Ball', 'Gloves', 'Shirt', 'Bottle', 'Cap']
Transaction 4: ['Jacket', 'Gloves', 'Shirt', 'Basket']
Transaction 5: ['Gloves', 'Bag', 'Ball', 'Bottle', 'Cap']
Transaction 6: ['Shoes', 'Bottle', 'Socks', 'Cap']
Transaction 7: ['Jacket', 'Bottle', 'Socks']
Transaction 8: ['Socks', 'Bag', 'Ball', 'Bottle']
Transaction 9: ['Jacket', 'Bottle', 'Basket', 'Bag', 'Ball']
Transaction 10: ['Bag', 'Ball', 'Shirt', 'Socks', 'Cap']
Transaction 11: ['Socks', 'Bottle', 'Gloves', 'Bag', 'Ball', 'Cap']
Transaction 12: ['Basket', 'Gloves', 'Jacket', 'Bottle', 'Bag', 'Ball']
Transaction 13: ['Cap', 'Bottle', 'Socks', 'Bag', 'Ball', 'Shoes']
Transaction 14: ['Bag', 'Ball', 'Gloves', 'Cap', 'Basket']
Transaction 15: ['Shirt', 'Gloves', 'Bag', 'Ball', 'Cap', 'Shoes']
Transaction 16: ['Bag', 'Ball', 'Jacket', 'Shirt', 'Socks', 'Gloves', 'Shoes']
Transaction 17: ['Shirt', 'Shoes', 'Socks', 'Basket', 'Bottle']
Transaction 18: ['Shirt', 'Bag', 'Ball', 'Cap', 'Socks', 'Basket', 'Jacket']
Transaction 19: ['Bag', 'Ball', 'Socks', 'Bottle', 'Cap']
```

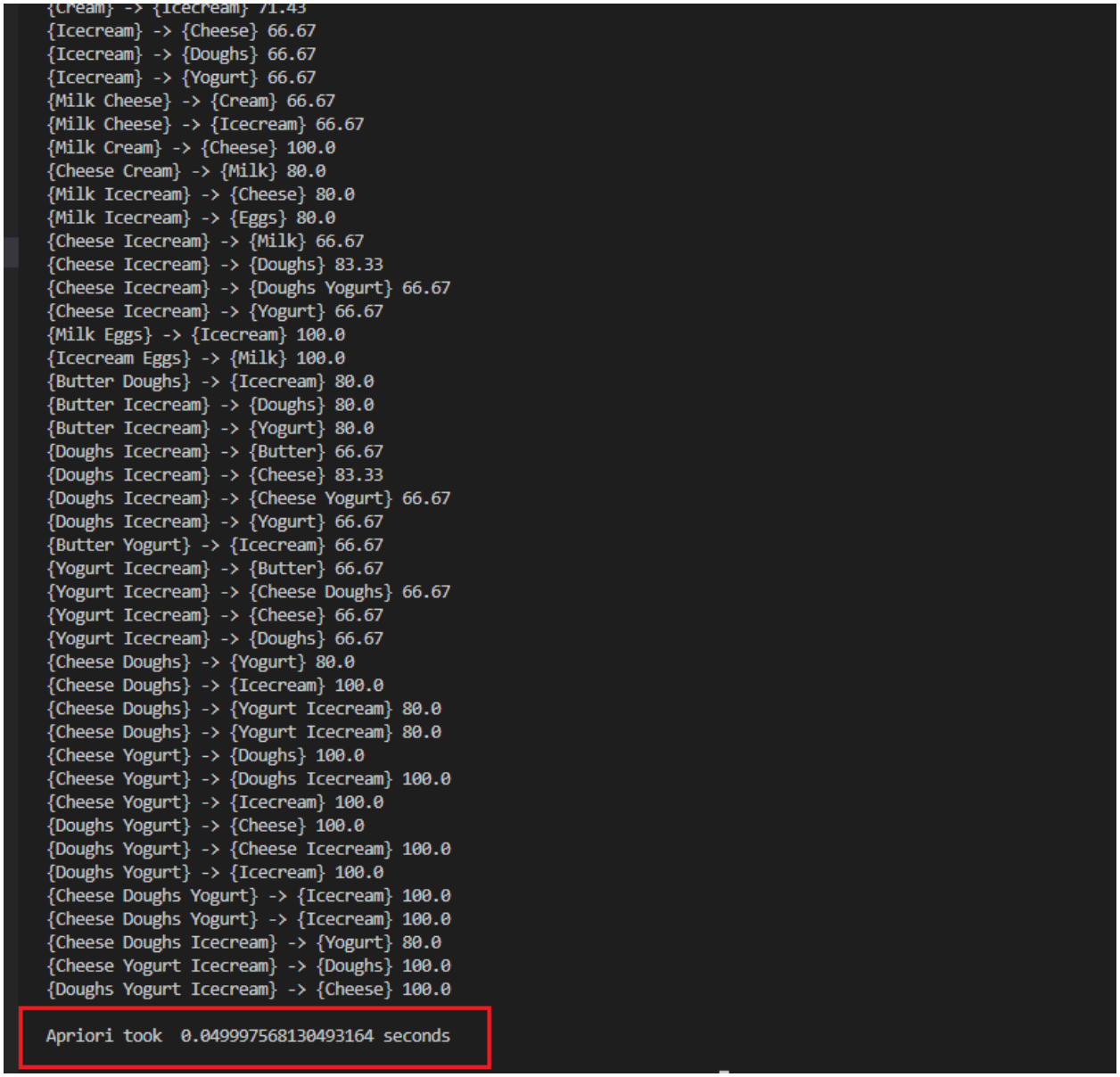
```
Minimum support value: 30

Minimum confidence value: 70

Candidate list:

['Shirt']
['Basket']
['Gloves']
['Cap']
['Jacket']
['Bag']
['Ball']
['Bottle']
['Socks']
['Shirt', 'Gloves']
['Shirt', 'Bag']
['Shirt', 'Ball']
['Gloves', 'Cap']
['Gloves', 'Bag']
['Gloves', 'Ball']
['Cap', 'Bag']
['Cap', 'Ball']
['Cap', 'Bottle']
['Cap', 'Socks']
['Jacket', 'Bag']
['Jacket', 'Ball']
['Bag', 'Ball']
['Bag', 'Bottle']
['Bag', 'Socks']
['Ball', 'Bottle']
['Ball', 'Socks']
['Bottle', 'Socks']
['Shirt', 'Bag', 'Ball']
['Gloves', 'Bag', 'Ball']
['Cap', 'Bag', 'Ball']
['Jacket', 'Bag', 'Ball']
['Bag', 'Ball', 'Bottle']
['Bag', 'Ball', 'Socks']
```

```
['Shirt', 'Ball']
['Gloves', 'Cap']
['Gloves', 'Bag']
['Gloves', 'Ball']
['Cap', 'Bag']
['Cap', 'Ball']
['Cap', 'Bottle']
['Cap', 'Socks']
['Jacket', 'Bag']
['Jacket', 'Ball']
['Bag', 'Ball']
['Bag', 'Bottle']
['Bag', 'Socks']
['Ball', 'Bottle']
['Ball', 'Socks']
['Bottle', 'Socks']
['Shirt', 'Bag', 'Ball']
['Gloves', 'Bag', 'Ball']
['Cap', 'Bag', 'Ball']
['Jacket', 'Bag', 'Ball']
['Bag', 'Ball', 'Bottle']
['Bag', 'Ball', 'Socks']

All the possible association rules along with their confidence (%) are as follows:

{Gloves} -> {Bag} 80.0
{Gloves} -> {Ball} 80.0
{Gloves} -> {Bag Ball} 80.0
{Cap} -> {Bag} 83.33
{Cap} -> {Ball} 83.33
{Cap} -> {Bag Ball} 83.33
{Jacket} -> {Bag} 75.0
{Jacket} -> {Ball} 75.0
{Jacket} -> {Bag Ball} 75.0
{Bag} -> {Ball} 100.0
{Ball} -> {Bag} 100.0
{Bottle} -> {Bag} 72.73
{Bottle} -> {Ball} 72.73
{Bottle} -> {Bag Ball} 72.73
{Socks} -> {Bag} 70.0
{Socks} -> {Ball} 70.0
{Socks} -> {Bottle} 70.0
{Socks} -> {Bag Ball} 70.0
{Shirt Bag} -> {Ball} 100.0
{Shirt Ball} -> {Bag} 100.0
{Gloves Bag} -> {Ball} 100.0
{Gloves Ball} -> {Bag} 100.0
{Cap Bag} -> {Ball} 100.0
{Cap Ball} -> {Bag} 100.0
{Jacket Bag} -> {Ball} 100.0
{Jacket Ball} -> {Bag} 100.0
{Bag Bottle} -> {Ball} 100.0
{Ball Bottle} -> {Bag} 100.0
{Bag Socks} -> {Ball} 100.0
{Ball Socks} -> {Bag} 100.0

Apriori took  0.023998260498046875 seconds
```

Apriori Output 3:
Dataset → transaction database 3 - dairy.csv
Support → 20 %
Confidence → 60 %

```
Enter transaction filename including extension of the file : transaction database 3 - dairy.csv
**********Transactions**********
Transaction 0: ['Milk', 'Dips', 'Butter']
Transaction 1: ['IcecreamDips', 'Cheese', 'Milk', 'Cream', 'Butter']
Transaction 2: ['Cheese', 'Doughs', 'Yogurt', 'Icecream', 'Butter']
Transaction 3: ['Cheese', 'Eggs', 'Milk', 'Icecream', 'Butter', 'Cream', 'Yogurt', 'Doughs']
Transaction 4: ['Icecream', 'Doughs', 'Cream', 'Butter', 'Dips']
Transaction 5: ['Butter', 'Dips', 'Margarine', 'Doughs']
Transaction 6: ['Cream', 'Icecream', 'Doughs', 'Cheese', 'Butter', 'Yogurt']
Transaction 7: ['Cheese']
Transaction 8: ['Yogurt', 'Icecream', 'Margarine', 'Milk', 'Eggs', 'Butter']
Transaction 9: ['Margarine', 'Butter', 'Dips', 'Milk']
Transaction 10: ['Doughs', 'Margarine', 'Milk']
Transaction 11: ['Yogurt', 'Margarine', 'Icecream', 'Cream']
Transaction 12: ['Dips']
Transaction 13: ['Butter', 'Yogurt', 'Margarine']
Transaction 14: ['Eggs', 'Dips', 'Yogurt', 'Milk', 'Icecream', 'Cheese', 'Doughs']
Transaction 15: ['Dips', 'Milk', 'Cheese', 'Cream']
Transaction 16: ['Milk', 'Butter', 'Dips', 'Margarine', 'Yogurt']
Transaction 17: ['Cheese', 'Butter']
Transaction 18: ['Eggs', 'Cheese', 'Doughs', 'Milk', 'Icecream']
Transaction 19: ['Milk', 'Icecream', 'Dips', 'Cheese', 'Cream']
```

```
Minimum support value: 20

Minimum confidence value: 60

Candidate list:

['Milk']
['Dips']
['Butter']
['Cheese']
['Cream']
['Doughs']
['Yogurt']
['Icecream']
['Eggs']
['Margarine']
['Milk', 'Dips']
['Milk', 'Butter']
['Milk', 'Cheese']
['Milk', 'Cream']
['Milk', 'Doughs']
['Milk', 'Yogurt']
['Milk', 'Icecream']
['Milk', 'Eggs']
['Milk', 'Margarine']
['Dips', 'Butter']
['Butter', 'Cheese']
['Butter', 'Cream']
['Butter', 'Doughs']
['Butter', 'Yogurt']
['Butter', 'Icecream']
['Butter', 'Margarine']
['Cheese', 'Cream']
['Cheese', 'Doughs']
['Cheese', 'Yogurt']
['Cheese', 'Icecream']
['Cream', 'Icecream']
['Doughs', 'Yogurt']
['Doughs', 'Icecream']
['Yogurt', 'Icecream']
['Yogurt', 'Margarine']
['Icecream', 'Eggs']
['Milk', 'Cheese', 'Cream']
['Milk', 'Cheese', 'Icecream']
['Milk', 'Icecream', 'Eggs']
['Butter', 'Doughs', 'Icecream']
['Butter', 'Yogurt', 'Icecream']
['Cheese', 'Doughs', 'Yogurt']
['Cheese', 'Doughs', 'Icecream']
['Cheese', 'Doughs', 'Yogurt', 'Icecream']
['Cheese', 'Yogurt', 'Icecream']
['Doughs', 'Yogurt', 'Icecream']
['Cheese', 'Doughs', 'Yogurt', 'Icecream']
```



Apriori Output 4:
Dataset → transaction database 4 - snacks.csv
Support → 45 %
Confidence → 65 %

```
Enter transaction filename including extension of the file : transaction database 4 - snacks.csv
**********Transactions**********

Transaction 0: ['Crackers', 'Nuts', 'Chips', 'Cookies']
Transaction 1: ['Pretzels', 'Pudding', 'MeatSticks', 'Spreads']
Transaction 2: ['Pudding', 'Pretzels', 'Nuts', 'Crackers', 'Dips']
Transaction 3: ['MeatSticks', 'Pretzels']
Transaction 4: ['Chips', 'Cookies', 'Pudding', 'Popcorn', 'Crackers', 'Dips', 'Pretzels', 'Spreads']
Transaction 5: ['Chips', 'Cookies', 'Pretzels', 'Pudding']
Transaction 6: ['Pudding', 'Chips', 'Cookies', 'Nuts', 'Popcorn', 'Pretzels', 'Dips']
Transaction 7: ['MeatSticks', 'Pretzels', 'Dips']
Transaction 8: ['Popcorn']
Transaction 9: ['MeatSticks', 'Pudding', 'Crackers', 'Popcorn']
Transaction 10: ['MeatSticks', 'Popcorn', 'Pretzels', 'Dips', 'Chips', 'Cookies']
Transaction 11: ['Chips', 'Cookies', 'Crackers', 'Dips', 'Nuts', 'Pretzels', 'Popcorn']
Transaction 12: ['MeatSticks']
Transaction 13: ['Pretzels', 'Pudding', 'Nuts', 'Crackers']
Transaction 14: ['Chips', 'Cookies', 'Pretzels', 'Dips', 'Pudding']
Transaction 15: ['Chips', 'Cookies', 'Dips', 'Nuts', 'MeatSticks', 'Pudding', 'Pretzels']
Transaction 16: ['Chips', 'Cookies', 'Dips', 'Nuts', 'Crackers']
Transaction 17: ['Crackers']
Transaction 18: ['Pretzels', 'Chips', 'Cookies', 'Crackers', 'Pudding', 'Popcorn', 'Dips']
Transaction 19: ['Popcorn', 'Chips', 'Cookies', 'Nuts']
```

```
Transaction 10: ['MeatSticks', 'Popcorn', 'Pretzels', 'Dips', 'Chips', 'Cookies']

Transaction 11: ['Chips', 'Cookies', 'Crackers', 'Dips', 'Nuts', 'Pretzels', 'Popcorn']

Transaction 12: ['MeatSticks']

Transaction 13: ['Pretzels', 'Pudding', 'Nuts', 'Crackers']

Transaction 14: ['Chips', 'Cookies', 'Pretzels', 'Dips', 'Pudding']

Transaction 15: ['Chips', 'Cookies', 'Dips', 'Nuts', 'MeatSticks', 'Pudding', 'Pretzels']

Transaction 16: ['Chips', 'Cookies', 'Dips', 'Nuts', 'Crackers']

Transaction 17: ['Crackers']

Transaction 18: ['Pretzels', 'Chips', 'Cookies', 'Crackers', 'Pudding', 'Popcorn', 'Dips']

Transaction 19: ['Popcorn', 'Chips', 'Cookies', 'Nuts']

Candidate list:

['Crackers']
['Chips']
['Cookies']
['Pretzels']
['Pudding']
['Dips']
['Chips', 'Cookies']
['Pretzels', 'Pudding']
['Pretzels', 'Dips']

All the possible association rules along with their confidence (%) are as follows:

{Chips} -> {Cookies} 100.0
{Cookies} -> {Chips} 100.0
{Pretzels} -> {Pudding} 69.23
{Pretzels} -> {Dips} 69.23
{Pudding} -> {Pretzels} 90.0
{Dips} -> {Pretzels} 90.0

Apriori took  0.010975122451782227 seconds
```

Apriori Output 5:
Dataset → transaction database 5 - beverages.csv
Support → 15 %
Confidence → 90 %

```
Enter transaction filename including extension of the file : transaction database 5 - beverages.csv
**********Transactions**********

Transaction 0: ['Water', 'Cocoa', 'Cider']
Transaction 1: ['Cocoa', 'Water', 'Juice']
Transaction 2: ['Cocktail']
Transaction 3: ['Tea', 'Cocktail', 'Wine', 'Coffee', 'Water']
Transaction 4: ['Cocktail', 'Juice', 'ProteinShake', 'Soda']
Transaction 5: ['Tea', 'ProteinShake', 'Coffee', 'Soda']
Transaction 6: ['Water']
Transaction 7: ['Water', 'Cocoa', 'Wine']
Transaction 8: ['Juice', 'Cocoa', 'Water', 'Coffee', 'Tea', 'Cider']
Transaction 9: ['Cocoa', 'Tea', 'Water', 'Coffee', 'Wine']
Transaction 10: ['Soda', 'Tea', 'Coffee', 'ProteinShake', 'Juice']
Transaction 11: ['Cocoa', 'Cocktail', 'Juice', 'ProteinShake', 'Water']
Transaction 12: ['Coffee', 'Tea', 'ProteinShake', 'Cocktail', 'Water', 'Cider']
Transaction 13: ['Cocktail', 'Coffee', 'ProteinShake', 'Wine', 'Juice']
Transaction 14: ['Soda', 'Coffee', 'Cider']
Transaction 15: ['ProteinShake', 'Water', 'Coffee']
Transaction 16: ['ProteinShake', 'Cocktail', 'Ciderss']
Transaction 17: ['Coffee', 'Cocktail', 'Cocoa', 'Wine', 'ProteinShake', 'Water']
Transaction 18: ['Juice', 'Coffee', 'Soda']
Transaction 19: ['Soda']
```

```
['Water', 'Wine']
['Water', 'Coffee']
['Water', 'ProteinShake']
['Cocoa', 'Juice']
['Cocoa', 'Wine']
['Cocoa', 'Coffee']
['Cider', 'Coffee']
['Juice', 'Cocktail']
['Juice', 'Coffee']
['Juice', 'ProteinShake']
['Juice', 'Soda']
['Cocktail', 'Wine']
['Cocktail', 'Coffee']
['Cocktail', 'ProteinShake']
['Tea', 'Coffee']
['Tea', 'ProteinShake']
['Wine', 'Coffee']
['Coffee', 'ProteinShake']
['Coffee', 'Soda']
['ProteinShake', 'Soda']
['Water', 'Cocoa', 'Juice']
['Water', 'Cocoa', 'Wine']
['Water', 'Cocoa', 'Coffee']
['Water', 'Cocktail', 'Coffee']
['Water', 'Cocktail', 'ProteinShake']
['Water', 'Tea', 'Coffee']
['Water', 'Wine', 'Coffee']
['Water', 'Coffee', 'ProteinShake']
['Juice', 'Cocktail', 'ProteinShake']
['Cocktail', 'Wine', 'Coffee']
['Cocktail', 'Coffee', 'ProteinShake']
['Tea', 'Coffee', 'ProteinShake']

All the possible association rules along with their confidence (%) are as follows:

{Cocoa} -> {Water} 100.0
{Tea} -> {Coffee} 100.0
{Water Juice} -> {Cocoa} 100.0
{Cocoa Juice} -> {Water} 100.0
{Cocoa Wine} -> {Water} 100.0
{Cocoa Coffee} -> {Water} 100.0
{Water Tea} -> {Coffee} 100.0
{Juice Cocktail} -> {ProteinShake} 100.0
{Cocktail Wine} -> {Coffee} 100.0
{Tea ProteinShake} -> {Coffee} 100.0

Apriori took  0.03748917579650879 seconds
```

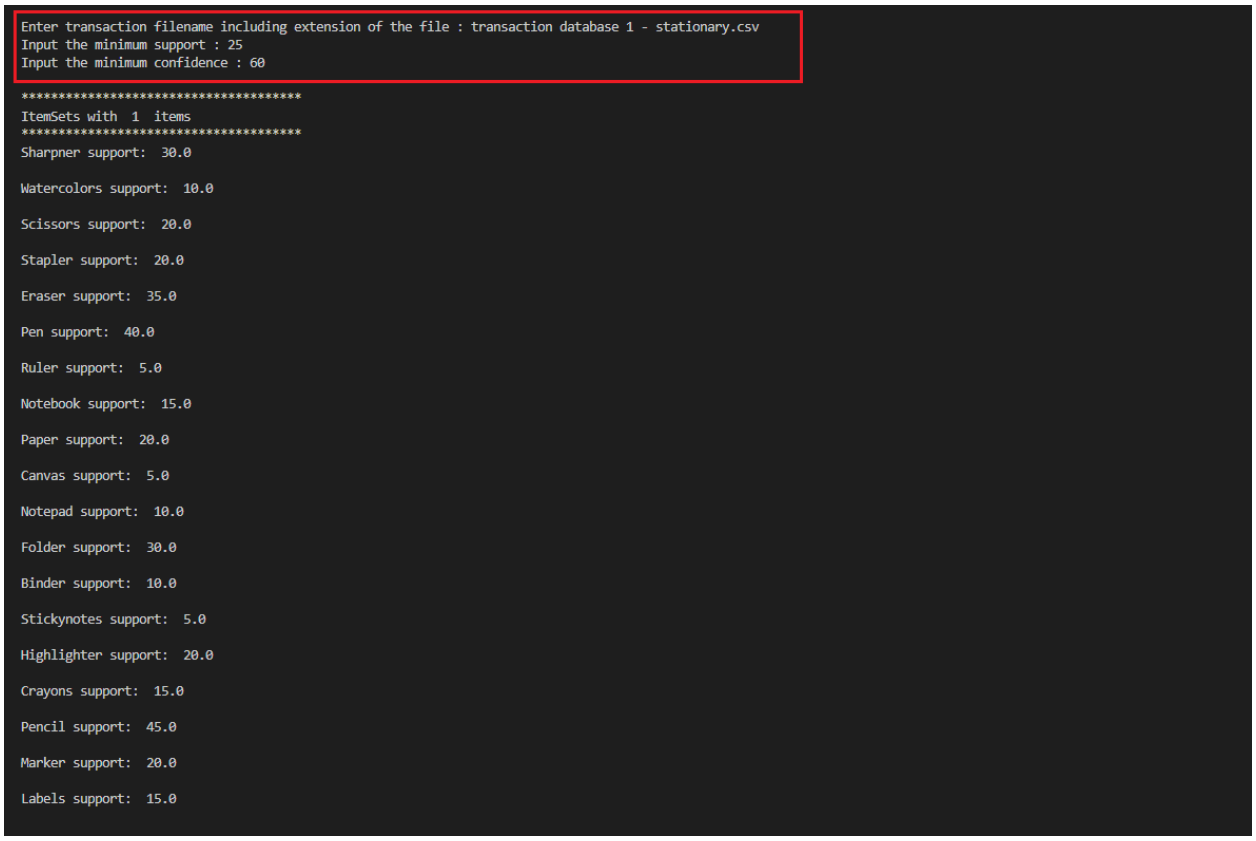
Source Code for Brute Force :

```
from itertools import combinations
import time


file_path= input("\nEnter transaction filename including extension of the file : ")
min_support = int(input("Input the minimum support : "))
min_confidence = int(input("Input the minimum confidence : "))
start = time.time()
file_object = open(file_path, "r")
l_of_data = file_object.readlines()
allItemList = []
items_list = []
d = {}
n_list = 0

#pre processing of the transaction dataset
support_of_all_item_set={}
for q in l_of_data:
  q = q.replace("\n", "")
  allItemList.append("".join(q.split(" ")[0].split(",")))
  s = set()
  for i in "".join(q.split(" ")[0:]).split(","):
    if (i,) in d:
      d[(i,)] += 1
    else:
      d[(i,)] = 1
    s.add(i)
  items_list.append(s)
  n_list+=1
k = []

for i in items_list:
  i = list(i)
  k.extend(i)

k = list(set(k))


#function to find support
def find_support(l):
  dict_sup = {}
  length = len(items_list)
  for i in l:
    c = 0
    for j in items_list:
      if type(i) != tuple:
        if set([i]).issubset(set(j)):
          c += 1
      else:
        if set(i).issubset(set(j)):
```

```
            c += 1
        dict_sup[i] = c*100/length
    return dict_sup

individual_support = find_support(k)

# create association rules and check if their support and confidence satisfies the
user given values
def Assoc_rules(frequentSet):
    for item in frequentSet.keys():
        SizeOfSet=len(item)
        itemset=set(item)
        while SizeOfSet-1>0:
            combos = combinations(item, SizeOfSet-1)
            for i in combos:
                lefts=i[0]
                rights=tuple(itemset-set(i))

item_confidence=round(current_support[item]*100/individual_support[lefts],2)
            if item_confidence >= min_confidence and current_support[item] >=
min_support:
                print("Association rules of the item: ", item, "\nSupport:
",current_support[item])
                print(lefts," -->", rights, "Confidence: ", item_confidence)
                print()
            SizeOfSet -=1

current_size = 1
f = []

while current_size <= len(items_list[0]):
    print("\n************************************")
    print("ItemSets with ", current_size, " items")
    print("************************************")
    if current_size == 1:
        for i in individual_support.keys():
            print(i, "support: ", individual_support[i])
            print()
    elif current_size > 1:
        current_support = find_support(list(combinations(k,current_size)))
        for i in current_support.keys():
            lefts = i[0]
            item_confidence=round(current_support[i]*100/individual_support[lefts],2)
            if current_support[i] >= min_support:
                f.append(i)
                print(i, "with support: ", current_support[i], "and Confidence:
",item_confidence)
                print()
        Assoc_rules(current_support)
    if current_size > 1:
        if len(f) > 1:
            f = []
        else:
            break
    current_size += 1
print()
```

```
print()
print("TIme for Brute force completion ", time.time() - start, "seconds")
```

Brute force Output 1:
Dataset → transaction database 1 - stationary.csv
Support → 25 %
Confidence → 60 %



```
************************************
ItemSets with  2  items
************************************
('Sharpner', 'Pencil') with support:  30.0 and Confidence:  100.0

('Eraser', 'Pencil') with support:  30.0 and Confidence:  85.71

Association rules of the item:  ('Sharpner', 'Pencil')
Support:  30.0
Sharpner  --> ('Pencil',) Confidence:  100.0

Association rules of the item:  ('Sharpner', 'Pencil')
Support:  30.0
Pencil  --> ('Sharpner',) Confidence:  66.67

Association rules of the item:  ('Eraser', 'Pencil')
Support:  30.0
Eraser  --> ('Pencil',) Confidence:  85.71

Association rules of the item:  ('Eraser', 'Pencil')
Support:  30.0
Pencil  --> ('Eraser',) Confidence:  66.67


************************************
ItemSets with  3  items
************************************

TIme for Brute force completion  0.02602839469909668 seconds
```

Brute Apriori Output 2:
Dataset → transaction database 2 - clothes.csv
Support → 30 %
Confidence → 70 %

```
Enter transaction filename including extension of the file : transaction database 2 - clothes.csv
Input the minimum support : 30
Input the minimum confidence : 70

**************************************
ItemSets with  1  items
**************************************
Bottle support:  55.0

Shirt support:  45.0

Jacket support:  40.0

Basket support:  35.0

Gloves support:  50.0

Shoes support:  25.0

Socks support:  50.0

Bag support:  75.0

Ball support:  75.0

Cap support:  60.0


**************************************
ItemSets with  2  items
**************************************
('Bottle', 'Socks') with support:  35.0 and Confidence:  63.64

('Bottle', 'Bag') with support:  40.0 and Confidence:  72.73

('Bottle', 'Ball') with support:  40.0 and Confidence:  72.73

('Bottle', 'Cap') with support:  30.0 and Confidence:  54.55

('Shirt', 'Gloves') with support:  30.0 and Confidence:  66.67

('Shirt', 'Bag') with support:  30.0 and Confidence:  66.67

('Shirt', 'Ball') with support:  30.0 and Confidence:  66.67

('Jacket', 'Bag') with support:  30.0 and Confidence:  75.0

('Jacket', 'Ball') with support:  30.0 and Confidence:  75.0

('Gloves', 'Bag') with support:  40.0 and Confidence:  80.0

('Gloves', 'Ball') with support:  40.0 and Confidence:  80.0

('Gloves', 'Cap') with support:  30.0 and Confidence:  60.0

('Socks', 'Bag') with support:  35.0 and Confidence:  70.0
```

```
Support:  40.0
Bottle  --> ('Ball',) Confidence:  72.73

Association rules of the item:  ('Bottle', 'Bag', 'Ball')
Support:  40.0
Bottle  --> ('Bag',) Confidence:  72.73

Association rules of the item:  ('Bottle', 'Bag', 'Ball')
Support:  40.0
Bottle  --> ('Bag', 'Ball') Confidence:  72.73

Association rules of the item:  ('Jacket', 'Bag', 'Ball')
Support:  30.0
Jacket  --> ('Ball',) Confidence:  75.0

Association rules of the item:  ('Jacket', 'Bag', 'Ball')
Support:  30.0
Jacket  --> ('Bag',) Confidence:  75.0

Association rules of the item:  ('Jacket', 'Bag', 'Ball')
Support:  30.0
Jacket  --> ('Bag', 'Ball') Confidence:  75.0

Association rules of the item:  ('Gloves', 'Bag', 'Ball')
Support:  40.0
Gloves  --> ('Ball',) Confidence:  80.0

Association rules of the item:  ('Gloves', 'Bag', 'Ball')
Support:  40.0
Gloves  --> ('Bag',) Confidence:  80.0

Association rules of the item:  ('Gloves', 'Bag', 'Ball')
Support:  40.0
Gloves  --> ('Bag', 'Ball') Confidence:  80.0

Association rules of the item:  ('Socks', 'Bag', 'Ball')
Support:  35.0
Socks  --> ('Ball',) Confidence:  70.0

Association rules of the item:  ('Socks', 'Bag', 'Ball')
Support:  35.0
Socks  --> ('Bag',) Confidence:  70.0

Association rules of the item:  ('Socks', 'Bag', 'Ball')
Support:  35.0
Socks  --> ('Bag', 'Ball') Confidence:  70.0

Association rules of the item:  ('Bag', 'Ball', 'Cap')
Support:  50.0
Cap  --> ('Bag', 'Ball') Confidence:  83.33


************************************
ItemSets with  4  items
************************************

TIme for Brute force completion  0.03800153732299805 seconds
```

Brute Apriori Output 3:
Dataset → transaction database 3 - dairy.csv
Support → 20 %
Confidence → 60 %

```
Enter transaction filename including extension of the file : transaction database 3 - dairy.csv
Input the minimum support : 20
Input the minimum confidence : 60

************************************
ItemSets with  1  items
************************************
Eggs support:  20.0

Milk support:  55.0

Icecream support:  45.0

Cheese support:  50.0

IcecreamDips support:  5.0

Butter support:  60.0

Yogurt support:  40.0

Margarine support:  35.0

Cream support:  35.0

Doughs support:  40.0

Dips support:  45.0


************************************
ItemSets with  2  items
************************************
('Eggs', 'Milk') with support:  20.0 and Confidence:  100.0

('Eggs', 'Icecream') with support:  20.0 and Confidence:  100.0

('Milk', 'Icecream') with support:  25.0 and Confidence:  45.45

('Milk', 'Cheese') with support:  30.0 and Confidence:  54.55

('Milk', 'Butter') with support:  30.0 and Confidence:  54.55

('Milk', 'Yogurt') with support:  20.0 and Confidence:  36.36

('Milk', 'Margarine') with support:  20.0 and Confidence:  36.36

('Milk', 'Cream') with support:  20.0 and Confidence:  36.36

('Milk', 'Doughs') with support:  20.0 and Confidence:  36.36

('Milk', 'Dips') with support:  30.0 and Confidence:  54.55

('Icecream', 'Cheese') with support:  30.0 and Confidence:  66.67

('Icecream', 'Butter') with support:  25.0 and Confidence:  55.56

('Icecream', 'Yogurt') with support:  30.0 and Confidence:  66.67

('Icecream', 'Cream') with support:  25.0 and Confidence:  55.56

('Icecream', 'Doughs') with support:  30.0 and Confidence:  66.67

('Cheese', 'Butter') with support:  25.0 and Confidence:  50.0

('Cheese', 'Yogurt') with support:  20.0 and Confidence:  40.0
```

```
Support:  30.0
Doughs  --> ('Icecream',) Confidence:  75.0

Association rules of the item:  ('Cheese', 'Cream')
Support:  25.0
Cream  --> ('Cheese',) Confidence:  71.43

Association rules of the item:  ('Cheese', 'Doughs')
Support:  25.0
Doughs  --> ('Cheese',) Confidence:  62.5

Association rules of the item:  ('Butter', 'Yogurt')
Support:  30.0
Yogurt  --> ('Butter',) Confidence:  75.0

Association rules of the item:  ('Butter', 'Margarine')
Support:  25.0
Margarine  --> ('Butter',) Confidence:  71.43

Association rules of the item:  ('Butter', 'Doughs')
Support:  25.0
Doughs  --> ('Butter',) Confidence:  62.5


************************************
ItemSets with  3  items
************************************
('Eggs', 'Milk', 'Icecream') with support:  20.0 and Confidence:  100.0

('Milk', 'Icecream', 'Cheese') with support:  20.0 and Confidence:  36.36

('Milk', 'Cheese', 'Cream') with support:  20.0 and Confidence:  36.36

('Icecream', 'Cheese', 'Yogurt') with support:  20.0 and Confidence:  44.44

('Icecream', 'Cheese', 'Doughs') with support:  25.0 and Confidence:  55.56

('Icecream', 'Butter', 'Yogurt') with support:  20.0 and Confidence:  44.44

('Icecream', 'Butter', 'Doughs') with support:  20.0 and Confidence:  44.44

('Icecream', 'Yogurt', 'Doughs') with support:  20.0 and Confidence:  44.44

('Cheese', 'Yogurt', 'Doughs') with support:  20.0 and Confidence:  40.0

Association rules of the item:  ('Eggs', 'Milk', 'Icecream')
Support:  20.0
Eggs  --> ('Icecream',) Confidence:  100.0

Association rules of the item:  ('Eggs', 'Milk', 'Icecream')
Support:  20.0
Eggs  --> ('Milk',) Confidence:  100.0

Association rules of the item:  ('Eggs', 'Milk', 'Icecream')
Support:  20.0
Eggs  --> ('Milk', 'Icecream') Confidence:  100.0

Association rules of the item:  ('Icecream', 'Cheese', 'Doughs')
Support:  25.0
Doughs  --> ('Cheese', 'Icecream') Confidence:  62.5


TIme for Brute force completion  0.04099726676940918 seconds
PS D:\ROOPALI FILES\SPRING 2022 COURSES\DATA MINING\MID TERM PROJECT>
```

Brute Force Output 4:
Dataset → transaction database 4 - snacks.csv
Support → 45 %
Confidence → 65 %

```
Enter transaction filename including extension of the file : transaction database 4 - snacks.csv
Input the minimum support : 45
Input the minimum confidence : 65

************************************
ItemSets with  1  items
************************************
Spreads support:  10.0

Cookies support:  55.0

Pretzels support:  65.0

MeatSticks support:  35.0

Popcorn support:  40.0

Crackers support:  45.0

Dips support:  50.0

Chips support:  55.0

Nuts support:  40.0

Pudding support:  50.0


************************************
ItemSets with  2  items
************************************
('Cookies', 'Chips') with support:  55.0 and Confidence:  100.0

('Pretzels', 'Dips') with support:  45.0 and Confidence:  69.23

('Pretzels', 'Pudding') with support:  45.0 and Confidence:  69.23

Association rules of the item:  ('Cookies', 'Chips')
Support:  55.0
Cookies  --> ('Chips',) Confidence:  100.0

Association rules of the item:  ('Cookies', 'Chips')
Support:  55.0
Chips  --> ('Cookies',) Confidence:  100.0

Association rules of the item:  ('Pretzels', 'Dips')
Support:  45.0
Pretzels  --> ('Dips',) Confidence:  69.23

Association rules of the item:  ('Pretzels', 'Dips')
Support:  45.0
Dips  --> ('Pretzels',) Confidence:  90.0

Association rules of the item:  ('Pretzels', 'Pudding')
Support:  45.0
Pretzels  --> ('Pudding',) Confidence:  69.23

Association rules of the item:  ('Pretzels', 'Pudding')
Support:  45.0
Pudding  --> ('Pretzels',) Confidence:  90.0


************************************
ItemSets with  3  items
************************************


TIme for Brute force completion  0.013995647430419922 seconds
```

Brute Force Output 5:
Dataset → transaction database 5 - beverages.csv
Support → 15 %
Confidence → 90 %

```
Enter transaction filename including extension of the file : transaction database 5 - beverages.csv
Input the minimum support : 15
Input the minimum confidence : 90

************************************
ItemSets with  1  items
************************************
Juice support:  35.0

Cocktail support:  40.0

Soda support:  30.0

Wine support:  25.0

Cocoa support:  35.0

Cider support:  20.0

Water support:  55.0

ProteinShake support:  45.0

Coffee support:  55.0

Tea support:  30.0

Ciderss support:  5.0


************************************
ItemSets with  2  items
************************************
('Juice', 'Cocktail') with support:  15.0 and Confidence:  42.86

('Juice', 'Soda') with support:  15.0 and Confidence:  42.86

('Juice', 'Cocoa') with support:  15.0 and Confidence:  42.86

('Juice', 'Water') with support:  15.0 and Confidence:  42.86

('Juice', 'ProteinShake') with support:  20.0 and Confidence:  57.14

('Juice', 'Coffee') with support:  20.0 and Confidence:  57.14

('Cocktail', 'Wine') with support:  15.0 and Confidence:  37.5

('Cocktail', 'Water') with support:  20.0 and Confidence:  50.0

('Cocktail', 'ProteinShake') with support:  30.0 and Confidence:  75.0

('Cocktail', 'Coffee') with support:  20.0 and Confidence:  50.0

('Soda', 'ProteinShake') with support:  15.0 and Confidence:  50.0

('Soda', 'Coffee') with support:  20.0 and Confidence:  66.67

('Wine', 'Cocoa') with support:  15.0 and Confidence:  60.0

('Wine', 'Water') with support:  20.0 and Confidence:  80.0

('Wine', 'Coffee') with support:  20.0 and Confidence:  80.0

('Cocoa', 'Water') with support:  35.0 and Confidence:  100.0

('Cocoa', 'Coffee') with support:  15.0 and Confidence:  42.86

('Cider', 'Water') with support:  15.0 and Confidence:  75.0

('Cider', 'Coffee') with support:  15.0 and Confidence:  75.0
```

```
('ProteinShake', 'Coffee') with support:  30.0 and Confidence:  66.67

('ProteinShake', 'Tea') with support:  15.0 and Confidence:  33.33

('Coffee', 'Tea') with support:  30.0 and Confidence:  54.55

Association rules of the item:  ('Cocoa', 'Water')
Support:  35.0
Cocoa  --> ('Water',) Confidence:  100.0

Association rules of the item:  ('Coffee', 'Tea')
Support:  30.0
Tea  --> ('Coffee',) Confidence:  100.0


************************************
ItemSets with  3  items
************************************
('Juice', 'Cocktail', 'ProteinShake') with support:  15.0 and Confidence:  42.86

('Juice', 'Cocoa', 'Water') with support:  15.0 and Confidence:  42.86

('Cocktail', 'Wine', 'Coffee') with support:  15.0 and Confidence:  37.5

('Cocktail', 'Water', 'ProteinShake') with support:  15.0 and Confidence:  37.5

('Cocktail', 'Water', 'Coffee') with support:  15.0 and Confidence:  37.5

('Cocktail', 'ProteinShake', 'Coffee') with support:  15.0 and Confidence:  37.5

('Wine', 'Cocoa', 'Water') with support:  15.0 and Confidence:  60.0

('Wine', 'Water', 'Coffee') with support:  15.0 and Confidence:  60.0

('Cocoa', 'Water', 'Coffee') with support:  15.0 and Confidence:  42.86

('Water', 'ProteinShake', 'Coffee') with support:  15.0 and Confidence:  27.27

('Water', 'Coffee', 'Tea') with support:  20.0 and Confidence:  36.36

('ProteinShake', 'Coffee', 'Tea') with support:  15.0 and Confidence:  33.33


TIme for Brute force completion  0.02599954605102539 seconds
PS D:\ROOPALI FILES\SPRING 2022 COURSES\DATA MINING\MID TERM PROJECT>
```

CPU runtime of both algorithms :

	Apriori (seconds)	Brute Force(seconds)
Dataset 1	0.009	0.026
Dataset 2	0.023	0.038
Dataset 3	0.049	0.049
Dataset 4	0.010	0.013
Dataset 5	0.037	0.025