

# **CS634 midterm project**

Wen Jiao

February 20, 2021

## Part 1: Apriori algorithm:

### Source code:

```
package cs634_midterm;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;

public class Apriori {
    private static double min_support;
    private static double min_confidence;
    static List<List<String>> record=new
ArrayList<List<String>>();
    static List<List<String>> frequentIS=new
ArrayList<List<String>>();
    static boolean endTag;
    static int times=2;

    public static void main(String[] args) {
        System.out.println("please input the min_support
value");
        Scanner in=new Scanner(System.in);
        min_support=in.nextDouble();
        System.out.println("the min_support value is
"+min_support);
        System.out.println("please input the min_confidence
value");
        min_confidence=in.nextDouble();
        System.out.println("the min_confidence value is
"+min_confidence);
        System.out.println();
        in.close();
        long start=System.currentTimeMillis(); //count time
from here
    }
}
```

```

record=getrecord("./DataSet01.txt");//scan data-set
//showData(record); for test

    apriori_alg();//use Apriori algorithm to generate
frequent item-sets
    //System.out.println("Frequent item-sets is
following:"); for test
    //showData(frequentIS); for test
    System.out.println();

        AssociationRulesMining();//generate all the
association rules
        long end=System.currentTimeMillis();//count time until
here
        System.out.println("Executing time: "+(end-start)
+"ms");

    }

    /**
     * method description: read transaction records from
data_set file
     * @param path
     * @return
     */
    private static List<List<String>> getrecord(String path){
        try {
            File dataset=new File(path);
            if(dataset.isFile()&&dataset.exists()) {
                InputStreamReader read=new
InputStreamReader(new FileInputStream(dataset));
                BufferedReader buffered_reader=new
BufferedReader(read);
                String one_line;
                while((one_line=buffered_reader.readLine())!=
>null) {
                    String[] one_record=one_line.split(",",
|:");
                    List<String> lineList=new
ArrayList<String>();
                    for(int i=1;i<one_record.length;i++) {
                        lineList.add(one_record[i]);
                    }
                }
            }
        }
    }
}

```

```

        Collections.sort(lineList);
        record.add(lineList);
    }
    read.close();
}
else if(dataset.isDirectory()&&dataset.exists())
{
    System.out.println("Path is a directory!");
}

else System.out.println("Can't find the file");
}
catch(Exception e){
    e.printStackTrace();
}
System.out.println("The input data is following:");
showData(record);
System.out.println();
return record;
}

/**
 * method description: implementation of Apriori Algorithm
 */
private static void apriori_alg() {
    List<List<String>> tempFList;

    //definition of OneD_CandidateSet: all the candidate
item-sets only have one item
    List<List<String>>
OneD_CandidateSet=get1stCandidateSet();
    //showData(OneD_CandidateSet)
    //definition of OneD_FrequentSet: all the frequent
item-sets only have one item
    List<List<String>>
OneD_FrequentSet=getNthFrequentSet(OneD_CandidateSet);
    frequentIS.addAll(OneD_FrequentSet);
    //System.out.println("1D frequent item-set:");
    for
test
    //showData(OneD_FrequentSet); for test
    tempFList=OneD_FrequentSet;

    while(endTag==false) {
        List<List<String>>
nextCandidateSet=getNextCandidateSet(tempFList);

```

```

        List<List<String>>
nextFrequentSet=getNthFrequentSet(nextCandidateSet);
        frequentIS.addAll(nextFrequentSet);
        if(endTag==false) {
            System.out.println(times+"D frequent item-
set:");
            showData(nextFrequentSet);
            times++;
            tempFList=nextFrequentSet;
        }
    }

    /**
     * method description: find the 1st candidate set(all the
unique items)
     * @return
     */
    private static List<List<String>> get1stCandidateSet(){
        List<List<String>> tablelist=new
ArrayList<List<String>>();
        HashSet<String> hs=new HashSet<String>();
        for(int i=0;i<record.size();i++) {
            for(int j=0;j<record.get(i).size();j++) {
                hs.add(record.get(i).get(j));
            }
        }

        Iterator<String> iterator=hs.iterator();
        while(iterator.hasNext()) {
            List<String> tempList=new ArrayList<String>();

            tempList.add(iterator.next());
            tablelist.add(tempList);
        }
        return tablelist;
    }

    /**
     * get frequent-sets from nth candidate-set
     * @param NthcandidateSet
     * @return
     */
    private static List<List<String>>
getNthFrequentSet(List<List<String>> NthcandidateSet){

```

```

        int count;
        boolean endTag_=true;
        List<List<String>> nthFrequentSet=new
ArrayList<List<String>>();

        for(int i=0;i<NthcandidateSet.size();i++) {
            count=countFrequency(NthcandidateSet.get(i));
            if(count>=min_support*record.size()) {
                nthFrequentSet.add(NthcandidateSet.get(i));
                //map.add(new
MyMap(NthcandidateSet.get(i),count));
                endTag_=false;
            }
        }
        endTag=endTag_;
        if(endTag==true) {
            System.out.println("can't find more frequent-
sets!");
        }
        return nthFrequentSet;
    }

    /**
     * method description: count the times that one item-set in
     * nth-candidateSet occurs
     * @param candidateSet
     * @return
     */
    private static int countFrequency(List<String>
candidateSet) {
        int count=0;
        for(int i=0;i<record.size();i++) {
            boolean countTag=true;
            for(int j=0;j<candidateSet.size();j++) {
                String temp=candidateSet.get(j);
                if(!record.get(i).contains(temp)) {
                    countTag=false;
                    break;
                }
            }
            if(countTag==true) count++;
        }
        return count;
    }

    /**

```

```

    * method description: use the last frequent item-sets to
    generate the next candidate item-sets
    * @param LastFrequentSet
    * @return
    */
private static List<List<String>>
getNextCandidateSet(List<List<String>> LastFrequentSet){
    List<List<String>> nextCandidateSet=new
ArrayList<List<String>>();
    List<List<String>> afterJoining=new
ArrayList<List<String>>();
    int dimension=LastFrequentSet.get(0).size();
    for(int i=0;i<LastFrequentSet.size();i++) {//join
        for(int j=i+1;j<LastFrequentSet.size();j++) {
            boolean joinable=true;
            for(int k=0;k<dimension-1;k++) {//Step 1 of
    checking if two item-set can be joined
            if(LastFrequentSet.get(i).get(k) !=
    =LastFrequentSet.get(j).get(k)) {
                joinable=false;
            }
        }

if(LastFrequentSet.get(i).get(dimension-1)==LastFrequentSet.get(
j).get(dimension-1)) {//Step 2 of checking two item-set can be
joined
                joinable=false;
}
if(joinable==true) {
    List<String> temp=new
ArrayList<String>();
    temp.addAll(LastFrequentSet.get(i));

temp.add(LastFrequentSet.get(j).get(dimension-1));
    Collections.sort(temp);
    afterJoining.add(temp);
}
}
}
//showData(afterJoining); For test

for(int i=0;i<afterJoining.size();i++) {//remove item-
set whose subsets are not all in the last frequent item-set
    boolean allSubsetOccur=true;
    for(int j=0;j<dimension;j++) {
        String item=afterJoining.get(i).remove(j);
    }
}
}

```

```

        List<String> temp=afterJoining.get(i);
        if(!LastFrequentSet.contains(temp)) {
            allSubsetOccur=false;
        }
        afterJoining.get(i).add(j,item); //return
    afterJoining to the original
    }
    if(allSubsetOccur==true) {
        nextCandidateSet.add(afterJoining.get(i));
    }
}
if(nextCandidateSet.isEmpty()) {
    System.out.println("Can't generate next
candidateSet!");
}
return nextCandidateSet;
}

private static void showData(List<List<String>> list) {
    for(int i=0;i<list.size();i++) {
        int temp=list.get(i).size()-1;
        for(int j=0;j<temp;j++) {
            System.out.print(list.get(i).get(j)+", ");
        }
        System.out.print(list.get(i).get(temp));
        System.out.println();
    }
}

private static void AssociationRulesMining() {
    for(int i=0;i<frequentIS.size();i++) {
        List<String> temp=frequentIS.get(i);
        if(temp.size()>1) {
            List<String> tempClone=new
ArrayList<String>();
            tempClone.addAll(temp);
            List<List<String>>
allSubset=getSubsets(tempClone);

            for(int j=0;j<allSubset.size();j++) {
                List<String> s1=allSubset.get(j);
                List<String> s2=getRest(temp,s1);
                double
confidence=isAssociationRule(s1,s2,temp);
                if(confidence>=min_confidence) {

```

```

                    double support=getCount(temp)*1.0/
record.size();
                    System.out.println("[Support=
"+support+", confidence= "+confidence+"]");
                }
            }
        }
    }

private static List<List<String>> getSubsets(List<String>
list) {
    List<List<String>> result=new
ArrayList<List<String>>();
    int length=list.size();
    int number0fSubsets=length==0?0:1<<(length);

    for(int i=1;i<number0fSubsets-1;i++) {//Starting from
1 because we don't need empty set
        List<String> subset=new ArrayList<String>();

        int index=i;
        for(int j=0;j<list.size();j++) {
            if((index&1)==1) {
                subset.add(list.get(j));
            }
            index=index>>1;
        }
        Collections.sort(subset);
        result.add(subset);
    }
    return result;
}

/**
 * method description: extract s1 from temp
 * @param temp
 * @param s1
 * @return
 */
private static List<String> getRest(List<String> temp,
List<String> s1){
    List<String> result=new ArrayList<String>();

```

```

        for(int i=0;i<temp.size();i++) {
            String item=temp.get(i);
            if(!s1.contains(item))
                result.add(item);
        }
        return result;
    }

    /**
     * method description: check if a subset of a item-set can
     * generate an association rule
     * @param s1
     * @param s2
     * @param temp
     * @return
     */
    private static double isAssociationRule(List<String>
s1,List<String> s2,List<String> temp) {
        double confidence;
        int counts1;
        int counttemp;

        if(temp.size()!=0&&temp!=null) {
            counts1=getCount(s1);
            //System.out.println(counts1);
            counttemp=getCount(temp);
            //System.out.println(counttemp);
            confidence=counttemp*1.0/counts1;

            if(confidence>=min_confidence) {
                System.out.print("Association rule:
"+s1.toString()+"-->"+s2.toString()+" ");
                return confidence;
            }
            else return 0;
        }
        else return 0;
    }

    private static int getCount(List<String> list) {
        int count=0;
        for(int i=0;i<record.size();i++) {
            boolean tag=true;
            for(int j=0;j<list.size();j++) {
                String temp=list.get(j);
                if(!record.get(i).contains(temp)) {

```

```

        tag=false;
        break;
    }
    if(tag==true) count++;
}
return count;
}

}

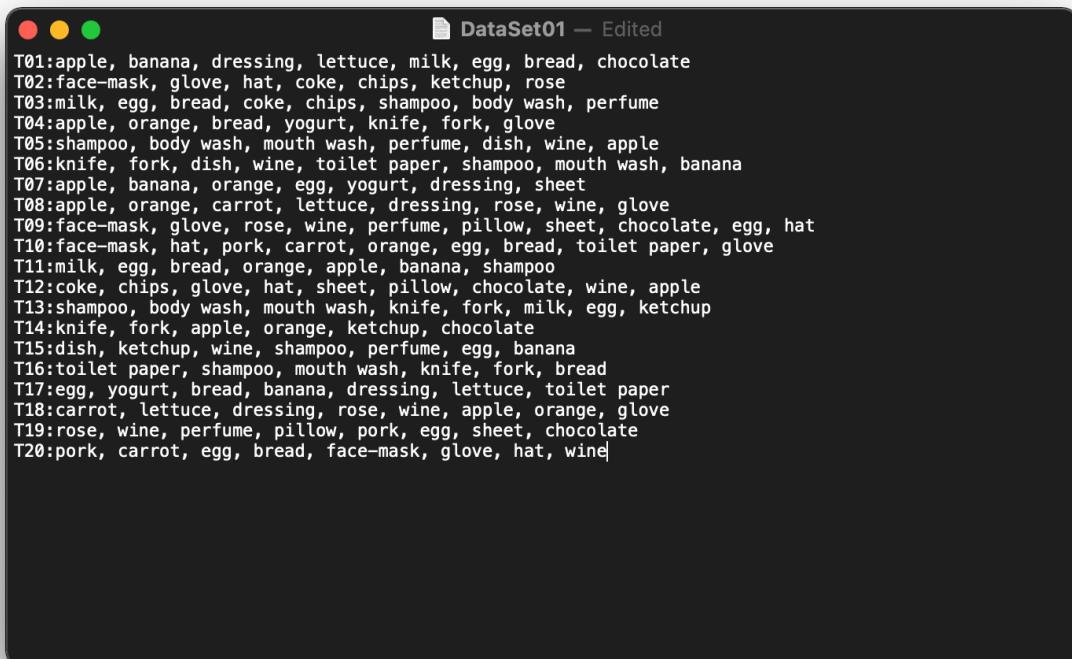
```

## Screenshots:

Data set 01:

Scenario: a normal day in Walmart

Screenshot of DataSet01.txt:



```

DataSet01 — Edited
T01:apple, banana, dressing, lettuce, milk, egg, bread, chocolate
T02:face-mask, glove, hat, coke, chips, ketchup, rose
T03:milk, egg, bread, coke, chips, shampoo, body wash, perfume
T04:apple, orange, bread, yogurt, knife, fork, glove
T05:shampoo, body wash, mouth wash, perfume, dish, wine, apple
T06:knife, fork, dish, wine, toilet paper, shampoo, mouth wash, banana
T07:apple, banana, orange, egg, yogurt, dressing, sheet
T08:apple, orange, carrot, lettuce, dressing, rose, wine, glove
T09:face-mask, glove, rose, wine, perfume, pillow, sheet, chocolate, egg, hat
T10:face-mask, hat, pork, carrot, orange, egg, bread, toilet paper, glove
T11:milk, egg, bread, orange, apple, banana, shampoo
T12:coke, chips, glove, hat, sheet, pillow, chocolate, wine, apple
T13:shampoo, body wash, mouth wash, knife, fork, milk, egg, ketchup
T14:knife, fork, apple, orange, ketchup, chocolate
T15:dish, ketchup, wine, shampoo, perfume, egg, banana
T16:toilet paper, shampoo, mouth wash, knife, fork, bread
T17:egg, yogurt, bread, banana, dressing, lettuce, toilet paper
T18:carrot, lettuce, dressing, rose, wine, apple, orange, glove
T19:rose, wine, perfume, pillow, pork, egg, sheet, chocolate
T20:pork, carrot, egg, bread, face-mask, glove, hat, wine

```

The min\_support value which is decided by user is 0.2

The min\_confidence value which is decided by user is 0.5

## Screenshots of output:

```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 7:36:19 AM)
please input the min_support value
0.2
the min_support value is 0.2
please input the min_confidence value
0.5
the min_confidence value is 0.5

The input data is following:
apple, banana, bread, chocolate, dressing, egg, lettuce, milk
chips, coke, face-mask, glove, hat, ketchup, rose
body wash, bread, chips, coke, egg, milk, perfume, shampoo
apple, bread, fork, glove, knife, orange, yogurt
apple, body wash, dish, mouth wash, perfume, shampoo, wine
banana, dish, fork, knife, mouth wash, shampoo, toilet paper, wine
apple, banana, dressing, egg, orange, sheet, yogurt
apple, carrot, dressing, glove, lettuce, orange, rose, wine
chocolate, egg, face-mask, glove, hat, perfume, pillow, rose, sheet, wine
bread, carrot, egg, face-mask, glove, hat, orange, pork, toilet paper
apple, banana, bread, egg, milk, orange, shampoo
apple, chips, chocolate, egg, glove, hat, pillow, sheet, wine
body wash, egg, fork, ketchup, knife, milk, mouth wash, shampoo
apple, chocolate, fork, ketchup, knife, orange
banana, dish, egg, ketchup, perfume, shampoo, wine
bread, fork, knife, mouth wash, shampoo, toilet paper
banana, bread, dressing, egg, lettuce, toilet paper, yogurt
apple, carrot, dressing, glove, lettuce, orange, rose, wine
chocolate, egg, perfume, pillow, pork, rose, sheet, wine
```

```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 7:36:19 AM)
apple, carrot, dressing, glove, lettuce, orange, rose, wine
chocolate, egg, perfume, pillow, pork, rose, sheet, wine
bread, carrot, egg, face-mask, glove, hat, pork, wine

2D frequent item-set:
egg, shampoo
mouth wash, shampoo
egg, perfume
banana, egg
bread, egg
egg, milk
egg, wine
apple, glove
glove, hat
glove, orange
face-mask, glove
glove, rose
glove, wine
carrot, glove
perfume, wine
apple, dressing
apple, orange
apple, wine
dressing, lettuce
face-mask, hat
fork, knife
rose, wine
```

```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 1:00:44 PM)
please input the min_support value
0.2
the min_support value is 0.2
please input the min_confidence value
0.5
the min_confidence value is 0.5

3D frequent item-set:
face-mask, glove, hat
Can't generate next candidateSet!
can't find more frequent-sets!

Association rule: [shampoo]-->[egg] [Support= 0.2, confidence= 0.5714285714285714]
Association rule: [mouth wash]-->[shampoo] [Support= 0.2, confidence= 1.0]
Association rule: [shampoo]-->[mouth wash] [Support= 0.2, confidence= 0.5714285714285714]
Association rule: [perfume]-->[egg] [Support= 0.2, confidence= 0.8]
Association rule: [apple]-->[banana] [Support= 0.2, confidence= 0.3333333333333334]
Association rule: [bread]-->[egg] [Support= 0.3, confidence= 0.75]
Association rule: [egg]-->[bread] [Support= 0.3, confidence= 0.5454545454545454]
Association rule: [milk]-->[egg] [Support= 0.2, confidence= 1.0]
Association rule: [glove]-->[apple] [Support= 0.2, confidence= 0.5]
Association rule: [glove]-->[hat] [Support= 0.25, confidence= 0.625]
Association rule: [glove]-->[apple] [Support= 0.2, confidence= 1.0]
Association rule: [glove]-->[orange] [Support= 0.2, confidence= 0.5]
Association rule: [orange]-->[glove] [Support= 0.2, confidence= 0.5714285714285714]
Association rule: [face-mask]-->[glove] [Support= 0.2, confidence= 1.0]
Association rule: [glove]-->[face-mask] [Support= 0.2, confidence= 0.5]
Association rule: [apple]-->[orange] [Support= 0.2, confidence= 0.8]
Association rule: [orange]-->[apple] [Support= 0.2, confidence= 0.8]
Association rule: [glove]-->[wine] [Support= 0.25, confidence= 0.625]
Association rule: [wine]-->[glove] [Support= 0.25, confidence= 0.5555555555555556]
Association rule: [carrot]-->[glove] [Support= 0.2, confidence= 1.0]
Association rule: [glove]-->[carrot] [Support= 0.2, confidence= 0.5]
Association rule: [glove]-->[wine] [Support= 0.2, confidence= 0.8]
Association rule: [dressing]-->[apple] [Support= 0.2, confidence= 0.8]
Association rule: [apple]-->[orange] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [orange]-->[apple] [Support= 0.3, confidence= 0.8571428571428571]
Association rule: [dressing]-->[lettuce] [Support= 0.2, confidence= 0.8]
Association rule: [lettuce]-->[dressing] [Support= 0.2, confidence= 0.8]
Association rule: [apple]-->[lettuce] [Support= 0.2, confidence= 0.9]
Association rule: [hat]-->[face-mask] [Support= 0.2, confidence= 0.8]
Association rule: [fork]-->[knife] [Support= 0.25, confidence= 1.0]
Association rule: [knife]-->[fork] [Support= 0.25, confidence= 1.0]
Association rule: [rose]-->[wine] [Support= 0.2, confidence= 0.8]
Association rule: [face-mask]-->[glove] [Support= 0.2, confidence= 1.0]
Association rule: [glove]-->[face-mask] [Support= 0.2, confidence= 0.5]
Association rule: [face-mask, glove]-->[hat] [Support= 0.2, confidence= 0.5]
Association rule: [hat]-->[face-mask, glove] [Support= 0.2, confidence= 0.8]
Association rule: [face-mask, hat]-->[glove] [Support= 0.2, confidence= 0.8]
Association rule: [glove, hat]-->[face-mask] [Support= 0.2, confidence= 0.8]

Executing time: 15ms
```

## Data set 02:

Scenario: another normal day in Walmart

Screenshot of DataSet02.txt



```
DataSet02
T01:milk, egg, bread, orange, apple, banana, shampoo, knife, yogurt, dish
T02:coke, chips, glove, hat, sheet, pillow, chocolate, wine, apple, yogurt
T03:shampoo, body wash, mouth wash, knife, fork, milk, egg, ketchup, glove
T04:knife, fork, apple, orange, ketchup, chocolate, egg, wine, lettuce, pillow
T05:dish, ketchup, wine, shampoo, perfume, egg, banana, pillow, dressing, hat
T06:toilet paper, shampoo, mouth wash, knife, fork, bread, orange, banana, rose
T07:egg, yogurt, bread, banana, dressing, lettuce, toilet paper, rose, milk
T08:carrot, lettuce, dressing, rose, wine, apple, orange, glove, perfume, body wash
T09:rose, wine, perfume, pillow, pork, egg, sheet, chocolate, mouth wash, body wash
T10:pork, carrot, egg, bread, face-mask, glove, hat, wine, rose, toilet paper
T11:apple, banana, dressing, lettuce, milk, egg, bread, chocolate, glove, hat
T12:face-mask, glove, hat, coke, chips, ketchup, rose, apple, wine, fork
T13:milk, egg, bread, coke, chips, shampoo, body wash, perfume, knife, carrot
T14:apple, orange, bread, yogurt, knife, fork, glove, carrot, hat, dressing
T15:shampoo, body wash, mouth wash, perfume, dish, wine, apple, orange, egg
T16:knife, fork, dish, wine, toilet paper, shampoo, mouth wash, banana, glove
T17:apple, banana, orange, egg, yogurt, dressing, sheet, glove, perfume, chocolate
T18:apple, orange, carrot, lettuce, dressing, rose, wine, glove, mouth wash, egg
T19:face-mask, glove, rose, wine, perfume, pillow, sheet, chocolate, egg, hat
T20:face-mask, hat, pork, carrot, orange, egg, bread, toilet paper, glove, banana
```

The min\_support value is 0.25

The min\_confidence value is 0.8

## Screenshots of output:

```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 7:47:57 AM)
please input the min_support value
0.25
the min_support value is 0.25
please input the min_confidence value
0.8
the min_confidence value is 0.8

The input data is following:
apple, banana, bread, dish, egg, knife, milk, orange, shampoo, yogurt
apple, chips, chocolate, coke, glove, hat, pillow, sheet, wine, yogurt
body wash, egg, fork, glove, ketchup, knife, milk, mouth wash, shampoo
apple, chocolate, egg, fork, ketchup, knife, lettuce, orange, pillow, wine
banana, dish, dressing, egg, hat, ketchup, perfume, pillow, shampoo, wine
banana, bread, fork, knife, mouth wash, orange, rose, shampoo, toilet paper
banana, bread, dressing, egg, lettuce, milk, rose, toilet paper, yogurt
apple, body wash, carrot, dressing, glove, lettuce, orange, perfume, rose, wine
body wash, chocolate, egg, mouth wash, perfume, pillow, pork, rose, sheet, wine
body wash, carrot, egg, face-mask, glove, hat, pork, rose, toilet paper, wine
apple, banana, bread, chocolate, dressing, egg, glove, hat, lettuce, milk
apple, chips, coke, glove, hat, knife, ketchup, rose, wine
body wash, bread, carrots, chips, coke, egg, knife, milk, perfume, shampoo
apple, bread, carrots, dressing, fork, glove, hat, knife, orange, yogurt
apple, body wash, dish, egg, mouth wash, orange, perfume, shampoo, wine
banana, dish, fork, glove, knife, mouth wash, shampoo, toilet paper, wine
apple, banana, chocolate, dressing, egg, glove, orange, perfume, sheet, yogurt
apple, carrot, dressing, egg, glove, lettuce, mouth wash, orange, rose, wine
chocolate, egg, face-mask, glove, hat, perfume, pillow, rose, sheet, wine
```

```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 7:47:57 AM)
apple, carrot, dressing, egg, glove, lettuce, mouth wash, orange, rose, wine
chocolate, egg, face-mask, glove, hat, perfume, pillow, rose, sheet, wine
banana, bread, carrot, egg, face-mask, glove, hat, orange, pork, toilet paper

2D frequent item-set:
egg, shampoo
knife, shampoo
egg, glove
egg, perfume
apple, egg
dressing, egg
egg, hat
banana, egg
bread, egg
egg, milk
egg, orange
egg, rose
egg, wine
chocolate, egg
apple, glove
dressing, glove
glove, hat
glove, orange
glove, rose
glove, wine
carrot, glove
perfume, wine
```

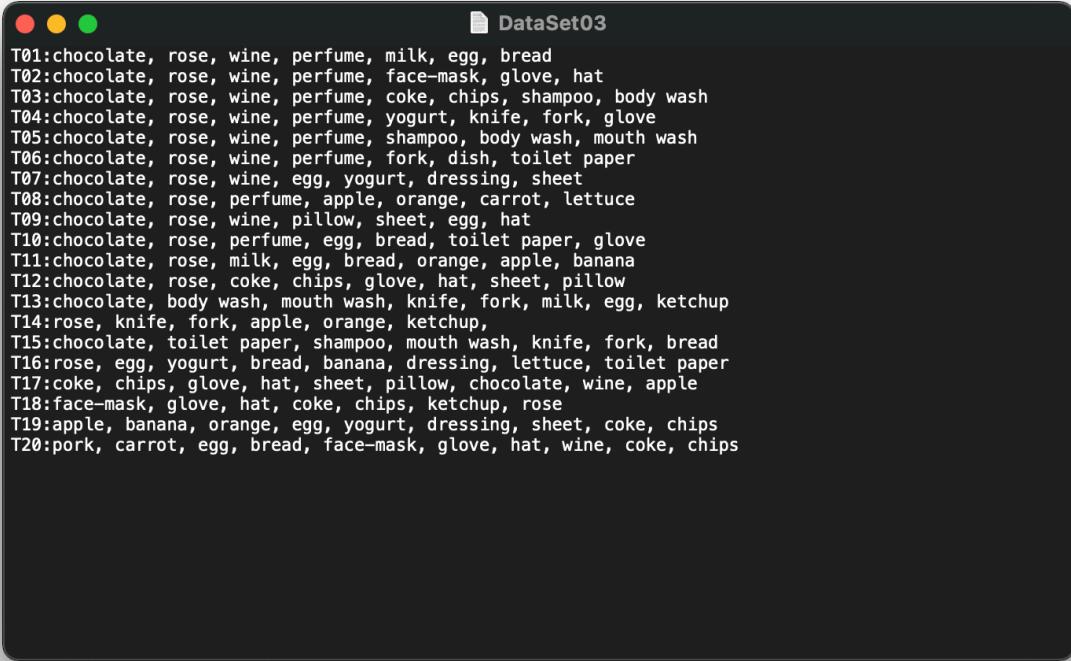
```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 1:06:17 PM)
glove, wine
carrot, glove
perfume, wine
apple, dressing
apple, orange
apple, wine
hat, wine
banana, bread
fork, knife
pillow, wine
rose, wine
3D frequent item-set:
apple, egg, orange
apple, dressing, glove
glove, rose, wine
Can't generate next candidateSet!
can't find more frequent-sets!

Association rule: [perfume]-->[egg] [Support= 0.3, confidence= 0.8571428571428571]
Association rule: [milk]-->[egg] [Support= 0.25, confidence= 1.0]
Association rule: [chocolate]-->[egg] [Support= 0.25, confidence= 0.8333333333333334]
Association rule: [hat]-->[glove] [Support= 0.35, confidence= 0.875]
Association rule: [carrot]-->[glove] [Support= 0.25, confidence= 0.8333333333333334]
Association rule: [fork]-->[knife] [Support= 0.25, confidence= 0.8333333333333334]
Association rule: [pillow]-->[wine] [Support= 0.25, confidence= 1.0]
Association rule: [apple, egg]-->[orange] [Support= 0.25, confidence= 0.8333333333333334]
Association rule: [egg, orange]-->[apple] [Support= 0.25, confidence= 0.8333333333333334]
Association rule: [apple, dressing]-->[glove] [Support= 0.25, confidence= 1.0]
Association rule: [dressing, glove]-->[apple] [Support= 0.25, confidence= 1.0]
Association rule: [glove, rose]-->[wine] [Support= 0.25, confidence= 1.0]
Association rule: [rose, wine]-->[glove] [Support= 0.25, confidence= 0.8333333333333334]
Executing time: 14ms
```

Data set 03:

Scenario: Valentine's day in Walmart

Screenshot of DataSet03.txt:



```
DataSet03
T01:chocolate, rose, wine, perfume, milk, egg, bread
T02:chocolate, rose, wine, perfume, face-mask, glove, hat
T03:chocolate, rose, wine, perfume, coke, chips, shampoo, body wash
T04:chocolate, rose, wine, perfume, yogurt, knife, fork, glove
T05:chocolate, rose, wine, perfume, shampoo, body wash, mouth wash
T06:chocolate, rose, wine, perfume, fork, dish, toilet paper
T07:chocolate, rose, wine, perfume, fork, dressing, sheet
T08:chocolate, rose, wine, egg, yogurt, dressing, sheet
T09:chocolate, rose, wine, pillow, sheet, egg, hat
T10:chocolate, rose, perfume, egg, bread, toilet paper, glove
T11:chocolate, rose, milk, egg, bread, orange, apple, banana
T12:chocolate, rose, coke, chips, glove, hat, sheet, pillow
T13:chocolate, body wash, mouth wash, knife, fork, milk, egg, ketchup
T14:rose, knife, fork, apple, orange, ketchup,
T15:chocolate, toilet paper, shampoo, mouth wash, knife, fork, bread
T16:rose, egg, yogurt, bread, banana, dressing, lettuce, toilet paper
T17:coke, chips, glove, hat, sheet, pillow, chocolate, wine, apple
T18:face-mask, glove, hat, coke, chips, ketchup, rose
T19:apple, banana, orange, egg, yogurt, dressing, sheet, coke, chips
T20:pork, carrot, egg, bread, face-mask, glove, hat, wine, coke, chips
```

The min\_support value is 0.3

The min\_confidence value is 0.6

## Screenshots of output:

```

Console 
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 8:01:01 AM)
please input the min_support value
0.3
the min_support value is 0.3
please input the min_confidence value
0.6
the min_confidence value is 0.6

The input data is following:
bread, chocolate, egg, milk, perfume, rose, wine
chocolate, face-mask, glove, hat, perfume, rose, wine
body wash, chips, chocolate, coke, perfume, rose, shampoo, wine
chocolate, fork, glove, knife, perfume, rose, wine, yogurt
body wash, chocolate, mouth wash, perfume, rose, shampoo, wine
chocolate, dish, fork, perfume, rose, toilet paper , wine
chocolate, dressing, egg, rose, sheet, wine, yogurt
apple, carrot, chocolate, lettuce, orange, perfume, rose
chocolate, egg, hat, pillow, rose, sheet, wine
bread, chocolate, egg, glove, perfume, rose, toilet paper
apple, banana, bread, chocolate, egg, milk, orange, rose
chips, chocolate, coke, glove, hat, pillow, rose, sheet
body wash, chocolate, egg, fork, ketchup, knife, milk, mouth wash
apple, fork, ketchup, knife, orange, rose
bread, chocolate, fork, knife, mouth wash, shampoo, toilet paper
banana, bread, dressing, egg, lettuce, rose, toilet paper, yogurt
apple, chips, chocolate, coke, glove, hat, pillow, sheet, wine
chips, coke, face-mask, glove, hat, ketchup, rose
apple, banana, chips, coke, dressing, egg, orange, sheet, yogurt
bread, carrot, chips, coke, egg, face-mask, glove, hat, pork, wine

2D frequent item-set:
egg, rose
chocolate, egg
chips, coke
perfume, rose
perfume, wine
chocolate, perfume
rose, wine
chocolate, rose
chocolate, wine
3D frequent item-set:
perfume, rose, wine
chocolate, perfume, rose
chocolate, perfume, wine
chocolate, rose, wine

```

```

Console 
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 1:08:08 PM)
Chocolate, perfume, rose
chocolate, perfume, wine
chocolate, rose, wine
4D frequent item-set:
chocolate, perfume, rose, wine
Can't generate next candidateSet!
can't find more frequent-sets!

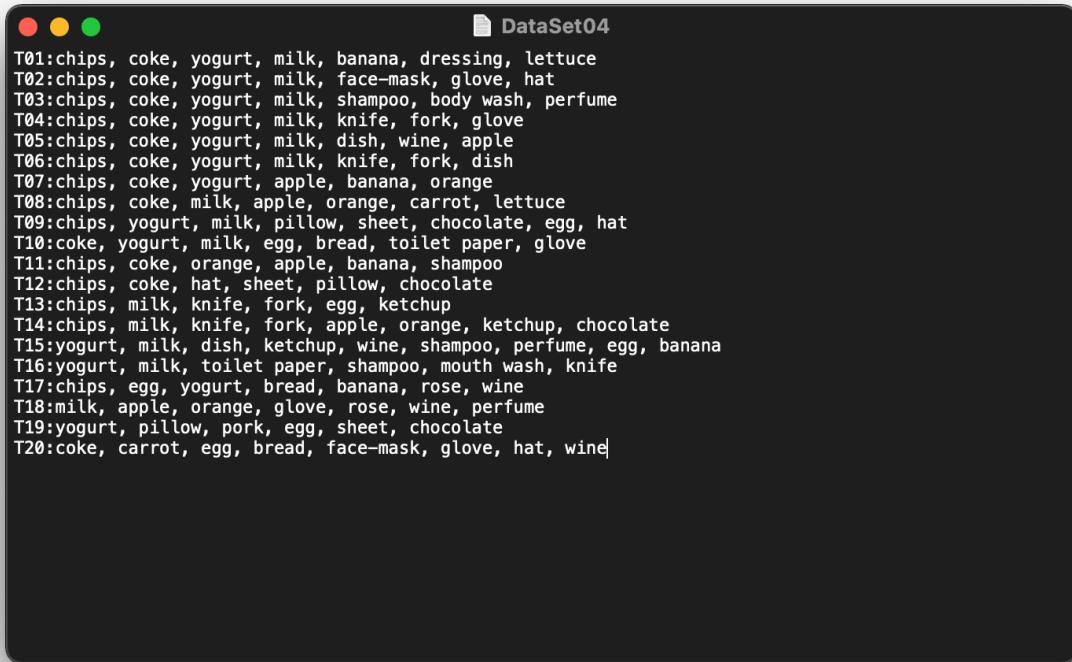
Association rule: [egg]-->[rose] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [egg]-->[chocolate] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [chips]-->[coke] [Support= 0.3, confidence= 1.0]
Association rule: [coke]-->[chips] [Support= 0.3, confidence= 1.0]
Association rule: [perfume]-->[rose] [Support= 0.4, confidence= 1.0]
Association rule: [perfume]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[perfume] [Support= 0.3, confidence= 0.6]
Association rule: [wine]-->[chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [wine]-->[rose] [Support= 0.4, confidence= 0.8]
Association rule: [chocolate]-->[rose] [Support= 0.6, confidence= 0.8]
Association rule: [rose]-->[chocolate] [Support= 0.6, confidence= 0.8]
Association rule: [chocolate]-->[wine] [Support= 0.45, confidence= 0.6]
Association rule: [wine]-->[chocolate] [Support= 0.45, confidence= 0.9]
Association rule: [perfume]-->[rose, wine] [Support= 0.3, confidence= 0.75]
Association rule: [perfume, rose]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[perfume, rose] [Support= 0.3, confidence= 0.6]
Association rule: [perfume, wine]-->[rose] [Support= 0.3, confidence= 1.0]
Association rule: [rose, wine]-->[perfume] [Support= 0.3, confidence= 0.75]
Association rule: [perfume]-->[chocolate, rose] [Support= 0.4, confidence= 1.0]
Association rule: [chocolate, perfume]-->[rose] [Support= 0.4, confidence= 1.0]
Association rule: [chocolate, rose]-->[perfume] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [perfume, rose]-->[chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [perfume, rose]-->[chocolate, wine] [Support= 0.4, confidence= 1.0]
Association rule: [chocolate, perfume]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume]-->[wine] [Support= 0.3, confidence= 0.6]
Association rule: [wine]-->[chocolate, perfume] [Support= 0.3, confidence= 0.6]
Association rule: [chocolate, wine]-->[perfume] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [perfume, wine]-->[chocolate] [Support= 0.3, confidence= 1.0]
Association rule: [chocolate, rose]-->[wine] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [wine]-->[chocolate, rose] [Support= 0.4, confidence= 0.8]
Association rule: [chocolate, wine]-->[rose] [Support= 0.4, confidence= 0.8888888888888888]
Association rule: [rose, wine]-->[chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [perfume]-->[chocolate, rose, wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume, rose]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume, rose]-->[chocolate, wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume, rose]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[chocolate, perfume, rose] [Support= 0.3, confidence= 0.6]
Association rule: [chocolate, wine]-->[perfume, rose] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [perfume, wine]-->[chocolate, rose] [Support= 0.3, confidence= 1.0]
Association rule: [chocolate, perfume, wine]-->[rose] [Support= 0.3, confidence= 1.0]
Association rule: [rose, wine]-->[chocolate, perfume] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, rose, wine]-->[perfume] [Support= 0.3, confidence= 0.75]
Association rule: [perfume, rose, wine]-->[chocolate] [Support= 0.3, confidence= 1.0]
Executing time: 12ms

```

#### Data set 04:

Scenario: 4 items (chips, coke, yogurt, milk) are on sale in Walmart

Screenshot of DataSet04.txt:



```
DataSet04
T01:chips, coke, yogurt, milk, banana, dressing, lettuce
T02:chips, coke, yogurt, milk, face-mask, glove, hat
T03:chips, coke, yogurt, milk, shampoo, body wash, perfume
T04:chips, coke, yogurt, milk, knife, fork, glove
T05:chips, coke, yogurt, milk, dish, wine, apple
T06:chips, coke, yogurt, milk, knife, fork, dish
T07:chips, coke, yogurt, apple, banana, orange
T08:chips, coke, milk, apple, orange, carrot, lettuce
T09:chips, yogurt, milk, pillow, sheet, chocolate, egg, hat
T10:coke, yogurt, milk, egg, bread, toilet paper, glove
T11:chips, coke, orange, apple, banana, shampoo
T12:chips, coke, hat, sheet, pillow, chocolate
T13:chips, milk, knife, fork, egg, ketchup
T14:chips, milk, knife, fork, apple, orange, ketchup, chocolate
T15:yogurt, milk, dish, ketchup, wine, shampoo, perfume, egg, banana
T16:yogurt, milk, toilet paper, shampoo, mouth wash, knife
T17:chips, egg, yogurt, bread, banana, rose, wine
T18:milk, apple, orange, glove, rose, wine, perfume
T19:yogurt, pillow, pork, egg, sheet, chocolate
T20:coke, carrot, egg, bread, face-mask, glove, hat, wine|
```

The min\_support value is 0.35

The min\_confidence value is 0.6

## Screenshots of output:

```

Console ☒
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 8:11:19 AM)
please input the min_support value
0.35
the min_support value is 0.35
please input the min_confidence value
0.6
the min_confidence value is 0.6

The input data is following:
banana, chips, coke, dressing, lettuce, milk, yogurt
chips, coke, face-mask, glove, hat, milk, yogurt
body wash, chips, coke, milk, perfume, shampoo, yogurt
chips, coke, fork, glove, knife, milk, yogurt
apple, chips, coke, dish, milk, wine, yogurt
chips, coke, dish, fork, knife, milk, yogurt
apple, banana, chips, coke, orange, yogurt
apple, carrot, chips, coke, lettuce, milk, orange
chips, chocolate, egg, hat, milk, pillow, sheet, yogurt
bread, coke, egg, glove, milk, toilet paper, yogurt
apple, banana, chips, coke, orange, shampoo
chips, chocolate, coke, hat, pillow, sheet
chip, egg, fork, ketchup, knife, milk
apple, chips, chocolate, fork, ketchup, knife, milk, orange
banana, dish, egg, glove, ketchup, milk, perfume, shampoo, wine, yogurt
knife, milk, mouth wash, shampoo, toilet paper, yogurt
banana, bread, chips, egg, rose, wine, yogurt
apple, glove, milk, orange, perfume, rose, wine
chocolate, egg, pillow, pork, sheet, yogurt
bread, carrot, coke, egg, face-mask, glove, hat, wine

2D frequent item-set:
chips, yogurt
coke, yogurt
milk, yogurt
chips, coke
chips, milk
coke, milk
3D frequent item-set:
chips, coke, yogurt|
chips, milk, yogurt
coke, milk, yogurt
chips, coke, milk
can't find more frequent-sets!

Association rule: [chips]-->[yogurt] [Support= 0.45, confidence= 0.6428571428571429]
Association rule: [yogurt]-->[chips] [Support= 0.45, confidence= 0.6923076923076923]
Association rule: [coke]-->[yogurt] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [yogurt]-->[coke] [Support= 0.4, confidence= 0.6153846153846154]
Association rule: [milk]-->[yogurt] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [yogurt]-->[milk] [Support= 0.5, confidence= 0.7692307692307693]
Association rule: [chips]-->[coke] [Support= 0.5, confidence= 0.7142857142857143]

```

```

Console ☒
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 1:09:45 PM)
can't find more frequent-sets!

Association rule: [chips]-->[yogurt] [Support= 0.45, confidence= 0.6428571428571429]
Association rule: [yogurt]-->[chips] [Support= 0.45, confidence= 0.6923076923076923]
Association rule: [coke]-->[yogurt] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [yogurt]-->[coke] [Support= 0.4, confidence= 0.6153846153846154]
Association rule: [milk]-->[yogurt] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [yogurt]-->[milk] [Support= 0.5, confidence= 0.7692307692307693]
Association rule: [chips]-->[coke] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [coke]-->[chips] [Support= 0.5, confidence= 0.8333333333333334]
Association rule: [chips]-->[milk] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [milk]-->[chips] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [coke]-->[milk] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [chips, coke]-->[yogurt] [Support= 0.35, confidence= 0.7]
Association rule: [chips, yogurt]-->[coke] [Support= 0.35, confidence= 0.7777777777777778]
Association rule: [coke, yogurt]-->[chips] [Support= 0.35, confidence= 0.875]
Association rule: [chips, milk]-->[yogurt] [Support= 0.35, confidence= 0.7]
Association rule: [chips, yogurt]-->[milk] [Support= 0.35, confidence= 0.7777777777777778]
Association rule: [milk, yogurt]-->[chips] [Support= 0.35, confidence= 0.7]
Association rule: [coke, milk]-->[yogurt] [Support= 0.35, confidence= 0.875]
Association rule: [coke, yogurt]-->[milk] [Support= 0.35, confidence= 0.875]
Association rule: [milk, yogurt]-->[coke] [Support= 0.35, confidence= 0.7]
Association rule: [chips, coke]-->[milk] [Support= 0.35, confidence= 0.7]
Association rule: [chips, milk]-->[coke] [Support= 0.35, confidence= 0.7]
Association rule: [coke, milk]-->[chips] [Support= 0.35, confidence= 0.875]

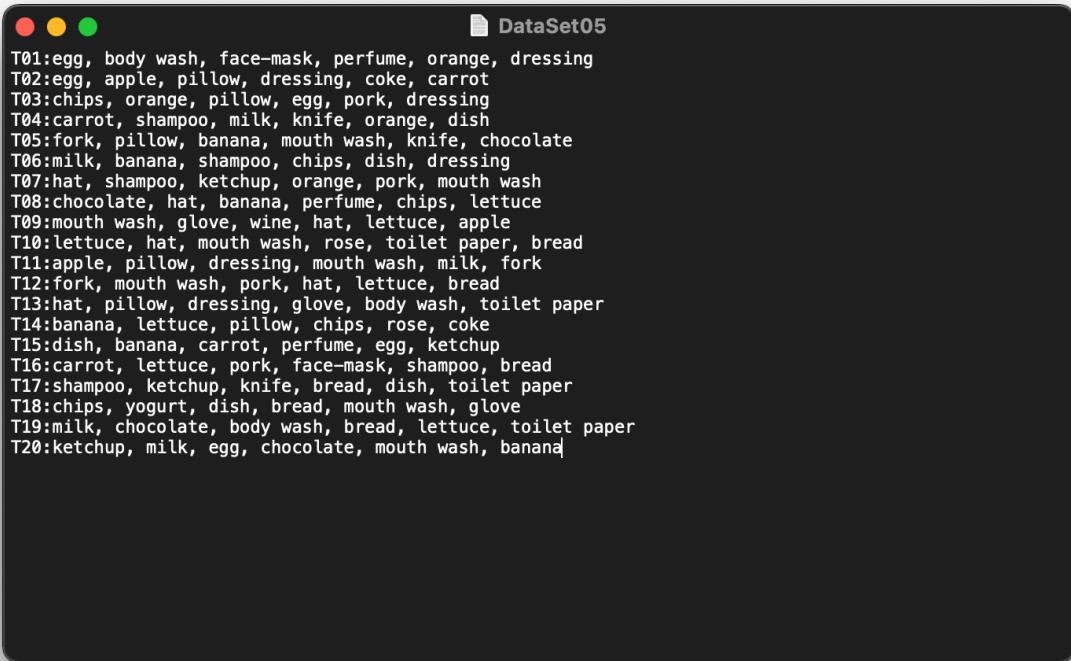
Executing time: 9ms

```

## Data set 05:

Scenario: a random data set (The reason why I generate a random data set although I know it's not proposed is, I want to find out what's the difference between a random data set and a artificial data set like each of the other 4 data sets)

Screenshot of DataSet05.txt:

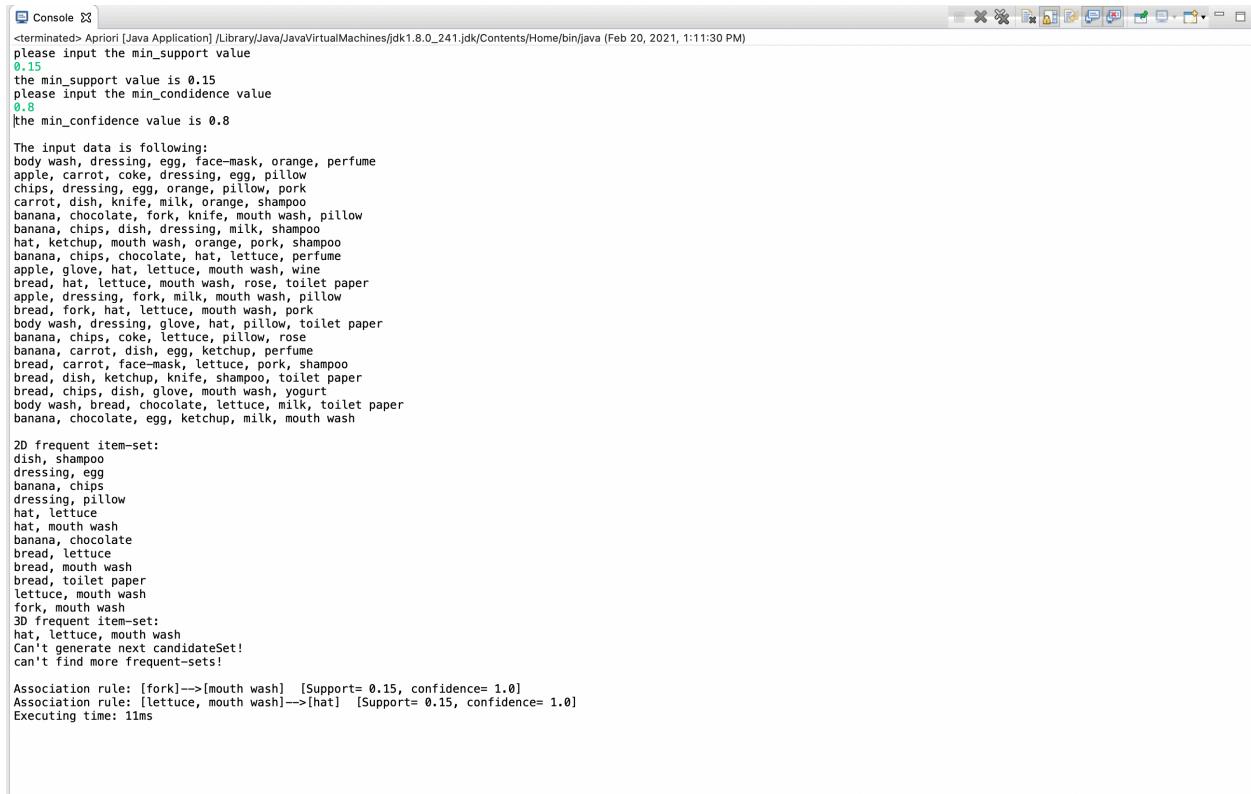


```
DataSet05
T01:egg, body wash, face-mask, perfume, orange, dressing
T02:egg, apple, pillow, dressing, coke, carrot
T03:chips, orange, pillow, egg, pork, dressing
T04:carrot, shampoo, milk, knife, orange, dish
T05:fork, pillow, banana, mouth wash, knife, chocolate
T06:milk, banana, shampoo, chips, dish, dressing
T07:hat, shampoo, ketchup, orange, pork, mouth wash
T08:chocolate, hat, banana, perfume, chips, lettuce
T09:mouth wash, glove, wine, hat, lettuce, apple
T10:lettuce, hat, mouth wash, rose, toilet paper, bread
T11:apple, pillow, dressing, mouth wash, milk, fork
T12:fork, mouth wash, pork, hat, lettuce, bread
T13:hat, pillow, dressing, glove, body wash, toilet paper
T14:banana, lettuce, pillow, chips, rose, coke
T15:dish, banana, carrot, perfume, egg, ketchup
T16:carrot, lettuce, pork, face-mask, shampoo, bread
T17:shampoo, ketchup, knife, bread, dish, toilet paper
T18:chips, yogurt, dish, bread, mouth wash, glove
T19:milk, chocolate, body wash, bread, lettuce, toilet paper
T20:ketchup, milk, egg, chocolate, mouth wash, banana
```

The min\_support value is 0.15

The min\_confidence value is 0.8

## Screenshots of output:



```
<terminated> Apriori [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 1:11:30 PM)
please input the min_support value
0.15
the min_support value is 0.15
please input the min_confidence value
0.8
the min_confidence value is 0.8

The input data is following:
body wash, dressing, egg, face-mask, orange, perfume
apple, carrot, coke, dressing, egg, pillow
chips, dressing, egg, orange, pillow, pork
carrot, dish, knife, milk, orange, shampoo
banana, chocolate, fork, knife, mouth wash, pillow
banana, chips, dish, dressing, milk, shampoo
hat, ketchup, mouth wash, orange, pork, shampoo
banana, chips, chocolate, hat, lettuce, perfume
apple, glove, hat, lettuce, mouth wash, wine
bread, hat, lettuce, mouth wash, rose, toilet paper
apple, dressing, fork, milk, mouth wash, pillow
bread, fork, hat, lettuce, mouth wash, pork
body wash, dressing, glove, hat, pillow, toilet paper
banana, chips, coke, lettuce, pillow, rose
banana, carrot, dish, egg, ketchup, perfume
bread, carrot, face-mask, lettuce, pork, shampoo
bread, dish, ketchup, knife, shampoo, toilet paper
bread, chips, dish, glove, mouth wash, yogurt
body wash, bread, chocolate, lettuce, milk, toilet paper
banana, chocolate, egg, ketchup, milk, mouth wash

2D frequent item-set:
dish, shampoo
dressing, egg
banana, chips
dressing, pillow
hat, lettuce
hat, mouth wash
banana, chocolate
bread, lettuce
bread, mouth wash
bread, toilet paper
lettuce, mouth wash
fork, mouth wash
3D frequent item-set:
hat, lettuce, mouth wash
Can't generate next candidateSet!
can't find more frequent-sets!

Association rule: [fork]-->[mouth wash] [Support= 0.15, confidence= 1.0]
Association rule: [lettuce, mouth wash]-->[hat] [Support= 0.15, confidence= 1.0]
Executing time: 11ms
```

Analysis: It's very obvious that generating association rules from a random dataset is more difficult than generating association rules from an artificial dataset. In order to generating association rules, we need to make either the value of support or the value of confidence lower. In this case, I lower the value of support which is 0.15.

## Part2: brute force algorithm

### Source code:

```
package cs634_midterm;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Scanner;
import java.util.Set;

public class Brute_force {
    private static double min_support;
    private static double min_confidence;
    static List<List<String>> record=new
ArrayList<List<String>>();
    static List<Set<String>> frequentIS=new
ArrayList<Set<String>>();
    static List<String> itemset=new ArrayList<String>();
    static boolean endTag=false;
    static int times=1;

    public static void main(String[] args) {
        System.out.println("please input the min_support
value");
        Scanner in=new Scanner(System.in);
        min_support=in.nextDouble();
        System.out.println("the min_support value is
"+min_support);
        System.out.println("please input the min_confidence
value");
        min_confidence=in.nextDouble();
        System.out.println("the min_confidence value is
"+min_confidence);
        System.out.println();
        in.close();
        long start=System.currentTimeMillis();

        record=getrecord("./DataSet01.txt");//scan data-set
    }
}
```

```

//showData(record); for test

for(int i=0;i<record.size();i++) {
    for(int j=0;j<record.get(i).size();j++) {
        String temp=record.get(i).get(j);
        if(!itemset.contains(temp)) {
            itemset.add(temp);
        }
    }
}
Collections.sort(itemset);

while(!endTag) {
    List<Set<String>> temp=new ArrayList<>();
    boolean haveFS=false;
    temp=brute_force(itemset,times);
    for(int i=0;i<temp.size();i++) {
        int count=0;
        for(int j=0;j<record.size();j++) {
            List<String> temp_=new
ArrayList<String>();
            temp_=record.get(j);
            if(temp_.containsAll(temp.get(i))) {
                count++;
            }
            /*if(count>=min_support*record.size()) {
                frequentIS.add(temp.get(i));
                haveFS=true;
            }*/
        }
        if(count>=min_support*record.size()) {
            frequentIS.add(temp.get(i));
            haveFS=true;
        }
    }
    if(!haveFS) {
        endTag=true;
    }
    else times++;
};

System.out.println();

```

*AssociationRulesMining(); //generate all the association rules*

```

        long end=System.currentTimeMillis();
        System.out.println("Executing time: "+(end-start)
+"ms");
    }

    private static List<List<String>> getrecord(String path){
        try {
            File dataset=new File(path);
            if(dataset.isFile()&&dataset.exists()) {
                InputStreamReader read=new
InputStreamReader(new FileInputStream(dataset));
                BufferedReader buffered_reader=new
BufferedReader(read);
                String one_line;
                while((one_line=buffered_reader.readLine())!=
null) {
                    String[] one_record=one_line.split(",",
|:");
                    List<String> lineList=new
ArrayList<String>();
                    for(int i=1;i<one_record.length;i++) {
                        lineList.add(one_record[i]);
                    }
                    Collections.sort(lineList);
                    record.add(lineList);
                }
                read.close();
            }
            else if(dataset.isDirectory()&&dataset.exists())
{
                System.out.println("Path is a directory!");
            }
            else System.out.println("Can't find the file");
        }
        catch(Exception e){
            e.printStackTrace();
        }
        System.out.println("The input data is following:");
        showData(record);
        System.out.println();
        return record;
    }

    private static List<Set<String>> brute_force(List<String>
itemset, int k){

```

```

        List<Set<String>> result=new ArrayList<>();
        getSubsets(itemset, k, 0, new HashSet<String>(),
result);
        return result;
    }

    private static void getSubsets(List<String> superSet, int
k, int idx, Set<String> current,List<Set<String>> solution) {
        //successful stop clause
        if (current.size() == k) {
            solution.add(new HashSet<>(current));
            return;
        }
        //unsuccessful stop clause
        if (idx == superSet.size()) return;
        String x = superSet.get(idx);
        current.add(x);
        //"guess" x is in the subset
        getSubsets(superSet, k, idx+1, current, solution);
        current.remove(x);
        //"guess" x is not in the subset
        getSubsets(superSet, k, idx+1, current, solution);
    }

    private static void AssociationRulesMining() {
        for(int i=0;i<frequentIS.size();i++) {
            List<String> temp=new ArrayList<String>();
            temp.addAll(frequentIS.get(i));
            if(temp.size()>1) {
                List<String> tempClone=new
ArrayList<String>();
                tempClone.addAll(temp);
                List<List<String>>
allSubset=getSubsets(tempClone);

                for(int j=0;j<allSubset.size();j++) {
                    List<String> s1=allSubset.get(j);
                    List<String> s2=getRest(temp,s1);
                    double
confidence=isAssociationRule(s1,s2,temp);
                    if(confidence>=min_confidence) {
                        double support=getCount(temp)*1.0/
record.size();
                        System.out.println("[Support=
"+support+", confidence= "+confidence+"]");
                    }
                }
            }
        }
    }
}

```

```

        }
    }

}

private static List<List<String>> getSubsets(List<String>
list) {
    List<List<String>> result=new
ArrayList<List<String>>();
    int length=list.size();
    int number0fSubsets=length==0?0:1<<(length);

    for(int i=1;i<number0fSubsets-1;i++) //Starting from
1 because we don't need empty set
    List<String> subset=new ArrayList<String>();

    int index=i;
    for(int j=0;j<list.size();j++) {
        if((index&1)==1) {
            subset.add(list.get(j));
        }
        index=index>>1;
    }
    Collections.sort(subset);
    result.add(subset);
}

return result;
}

private static List<String> getRest(List<String> temp,
List<String> s1){
    List<String> result=new ArrayList<String>();

    for(int i=0;i<temp.size();i++) {
        String item=temp.get(i);
        if(!s1.contains(item))
            result.add(item);
    }
    return result;
}

private static double isAssociationRule(List<String>
s1,List<String> s2,List<String> temp) {
    double confidence;

```

```

        int counts1;
        int counttemp;

        if(temp.size()!=0&&temp!=null) {
            counts1=getCount(s1);
            //System.out.println(counts1);
            counttemp=getCount(temp);
            //System.out.println(counttemp);
            confidence=counttemp*1.0/counts1;

            if(confidence>=min_confidence) {
                System.out.print("Association rule:
"+s1.toString()+"-->"+s2.toString()+"  ");
                return confidence;
            }
            else return 0;
        }
        else return 0;
    }

    private static int getCount(List<String> list) {
        int count=0;
        for(int i=0;i<record.size();i++) {
            boolean tag=true;
            for(int j=0;j<list.size();j++) {
                String temp=list.get(j);
                if(!record.get(i).contains(temp)) {
                    tag=false;
                    break;
                }
            }
            if(tag==true) count++;
        }
        return count;
    }

    private static void showData(List<List<String>> list) {
        for(int i=0;i<list.size();i++) {
            int temp=list.get(i).size()-1;
            for(int j=0;j<temp;j++) {
                System.out.print(list.get(i).get(j)+", ");
            }
            System.out.print(list.get(i).get(temp));
            System.out.println();
        }
    }
}

```

}

## Screenshots:

### Data set 01:

```
Console 
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:43:32 PM)
please input the min_support value
0.2
the min_support value is 0.2
please input the min_confidence value
0.5
the min_confidence value is 0.5

The input data is following:
apple, banana, bread, chocolate, dressing, egg, lettuce, milk
chips, coke, face-mask, glove, hat, ketchup, rose
body wash, bread, chips, coke, egg, milk, perfume, shampoo
apple, bread, fork, glove, knife, orange, yogurt
apple, body wash, dish, mouth wash, perfume, shampoo, wine
banana, dish, fork, knife, mouth wash, shampoo, toilet paper, wine
apple, banana, dressing, egg, orange, sheet, yogurt
apple, carrot, dressing, glove, lettuce, orange, rose, wine
chocolate, egg, face-mask, glove, hat, perfume, pillow, rose, sheet, wine
bread, carrot, egg, face-mask, glove, hat, orange, pork, toilet paper
apple, banana, bread, egg, milk, orange, shampoo
apple, chips, chocolate, face, glove, hat, pillow, sheet, wine
body wash, egg, fork, ketchup, knife, milk, mouth wash, shampoo
apple, chocolate, fork, ketchup, knife, orange
banana, dish, egg, ketchup, perfume, shampoo, wine
bread, fork, knife, mouth wash, shampoo, toilet paper
banana, bread, dressing, egg, lettuce, toilet paper, yogurt
apple, carrot, dressing, glove, lettuce, orange, rose, wine
chocolate, egg, perfume, pillow, pork, rose, sheet, wine
bread, carrot, egg, face-mask, glove, hat, pork, wine

Association rule: [dressing]-->[apple] [Support= 0.2, confidence= 0.8]
Association rule: [milk]-->[banana] [Support= 0.2, confidence= 0.5]
```

```
Console 
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:43:32 PM)
bread, carrot, egg, face-mask, glove, hat, pork, wine

Association rule: [dressing]-->[apple] [Support= 0.2, confidence= 0.8]
Association rule: [glove]-->[apple] [Support= 0.2, confidence= 0.5]
Association rule: [orange]-->[apple] [Support= 0.3, confidence= 0.8571428571428571]
Association rule: [apple]-->[orange] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [banana]-->[egg] [Support= 0.25, confidence= 0.8333333333333334]
Association rule: [bread]-->[egg] [Support= 0.3, confidence= 0.75]
Association rule: [egg]-->[bread] [Support= 0.3, confidence= 0.5454545454545454]
Association rule: [glove]-->[carrot] [Support= 0.2, confidence= 0.5]
Association rule: [carrot]-->[glove] [Support= 0.2, confidence= 1.0]
Association rule: [dressing]-->[lettuce] [Support= 0.2, confidence= 0.8]
Association rule: [lettuce]-->[dressing] [Support= 0.2, confidence= 1.0]
Association rule: [milk]-->[egg] [Support= 0.2, confidence= 1.0]
Association rule: [perfume]-->[egg] [Support= 0.2, confidence= 0.8]
Association rule: [shampoo]-->[egg] [Support= 0.2, confidence= 0.5714285714285714]
Association rule: [glove]-->[face-mask] [Support= 0.2, confidence= 0.5]
Association rule: [face-mask]-->[glove] [Support= 0.2, confidence= 1.0]
Association rule: [face-mask]-->[hat] [Support= 0.2, confidence= 1.0]
Association rule: [hat]-->[face-mask] [Support= 0.2, confidence= 0.8]
Association rule: [fork]-->[knife] [Support= 0.25, confidence= 1.0]
Association rule: [knife]-->[fork] [Support= 0.25, confidence= 1.0]
Association rule: [glove]-->[hat] [Support= 0.25, confidence= 0.625]
Association rule: [hat]-->[glove] [Support= 0.25, confidence= 1.0]
Association rule: [orange]-->[glove] [Support= 0.2, confidence= 0.5714285714285714]
Association rule: [glove]-->[orange] [Support= 0.2, confidence= 0.5]
Association rule: [glove]-->[rose] [Support= 0.2, confidence= 0.5]
Association rule: [rose]-->[glove] [Support= 0.2, confidence= 0.8]
Association rule: [glove]-->[wine] [Support= 0.25, confidence= 0.625]
Association rule: [wine]-->[glove] [Support= 0.25, confidence= 0.5555555555555556]
Association rule: [shampoo]-->[mouth wash] [Support= 0.2, confidence= 0.5714285714285714]
Association rule: [mouth wash]-->[shampoo] [Support= 0.2, confidence= 1.0]
Association rule: [perfume]-->[wine] [Support= 0.2, confidence= 0.8]
Association rule: [rose]-->[wine] [Support= 0.2, confidence= 0.8]
Association rule: [glove]-->[face-mask, hat] [Support= 0.2, confidence= 0.5]
Association rule: [face-mask]-->[glove, hat] [Support= 0.2, confidence= 1.0]
Association rule: [face-mask, glove]-->[hat] [Support= 0.2, confidence= 1.0]
Association rule: [hat]-->[glove, face-mask] [Support= 0.2, confidence= 0.8]
Association rule: [glove, hat]-->[face-mask] [Support= 0.2, confidence= 0.8]
Association rule: [face-mask, hat]-->[glove] [Support= 0.2, confidence= 1.0]
Executing time: 89ms
```

## Data set 02:

```
Console > 
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/dk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:50:01 PM)
please input the min_support value
0.25
the min_support value is 0.25
please input the min_confidence value
0.8
the min_confidence value is 0.8

The input data is following:
apple, banana, bread, dish, egg, knife, milk, orange, shampoo, yogurt
apple, chips, chocolate, coke, glove, hat, pillow, sheet, wine, yogurt
body wash, egg, fork, glove, ketchup, knife, milk, mouth wash, shampoo
apple, chocolate, egg, fork, ketchup, knife, lettuce, orange, pillow, wine
banana, dish, dressing, egg, hat, ketchup, perfume, pillow, shampoo, wine
banana, bread, fork, knife, mouth wash, orange, rose, shampoo, toilet paper
banana, bread, dressing, egg, lettuce, milk, rose, toilet paper, yogurt
apple, body wash, carrot, dressing, glove, lettuce, orange, perfume, rose, wine
body wash, chocolate, egg, mouth wash, perfume, pillow, pork, rose, sheet, wine
bread, carrot, egg, face-mask, glove, hat, pork, rose, toilet paper, wine
apple, banana, bread, chocolate, dressing, egg, glove, hat, lettuce, milk
apple, chips, coke, face-mask, fork, glove, hat, ketchup, rose, wine
body wash, bread, carrot, chips, coke, egg, knife, milk, perfume, shampoo
apple, bread, carrot, dressing, fork, glove, hat, knife, orange, yogurt
apple, body wash, dish, egg, mouth wash, orange, perfume, shampoo, wine
banana, dish, fork, glove, knife, mouth wash, shampoo, toilet paper, wine
apple, banana, chocolate, dressing, egg, glove, orange, perfume, sheet, yogurt
apple, carrot, dressing, egg, glove, lettuce, mouth wash, orange, rose, wine
chocolate, egg, face-mask, glove, hat, perfume, pillow, rose, sheet, wine
banana, bread, carrot, egg, face-mask, glove, hat, orange, pork, toilet paper

Association rule: [carrot]-->[glove] [Support= 0.25, confidence= 0.833333333333334]
Association rule: [chocolate]-->[egg] [Support= 0.25, confidence= 0.833333333333334]
Association rule: [milk]-->[egg] [Support= 0.25, confidence= 1.0]
Association rule: [perfume]-->[egg] [Support= 0.3, confidence= 0.8571428571428571]
Association rule: [fork]-->[knife] [Support= 0.25, confidence= 0.833333333333334]
Association rule: [hat]-->[glove] [Support= 0.35, confidence= 0.875]
Association rule: [pillow]-->[wine] [Support= 0.25, confidence= 1.0]
Association rule: [apple, dressing]-->[glove] [Support= 0.25, confidence= 1.0]
Association rule: [dressing, glove]-->[apple] [Support= 0.25, confidence= 1.0]
Association rule: [egg, orange]-->[apple] [Support= 0.25, confidence= 0.833333333333334]
Association rule: [apple, egg]-->[orange] [Support= 0.25, confidence= 0.833333333333334]
Association rule: [glove, rose]-->[wine] [Support= 0.25, confidence= 1.0]
Association rule: [rose, wine]-->[glove] [Support= 0.25, confidence= 0.833333333333334]
Executing time: 103ms
```

## Data set 03:

```

Console >
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:51:12 PM)
please input the min_support value
0.3
the min_support value is 0.3
please input the min_confidence value
0.6
the min_confidence value is 0.6

The input data is following:
bread, chocolate, egg, milk, perfume, rose, wine
chocolate, face-mask, glove, hat, perfume, rose, wine
body wash, chips, chocolate, coke, perfume, rose, shampoo, wine
chocolate, fork, glove, knife, perfume, rose, wine, yogurt
body wash, chocolate, mouth wash, perfume, rose, shampoo, wine
chocolate, dish, fork, perfume, rose, toilet paper , wine
chocolate, dressing, egg, rose, sheet, wine, yogurt
apple, carrot, chocolate, lettuce, orange, perfume, rose
chocolate, egg, hat, pillow, rose, sheet, wine
bread, chocolate, egg, glove, perfume, rose, toilet paper
apple, banana, bread, chocolate, egg, milk, orange, rose
chips, chocolate, coke, glove, hat, pillow, rose, sheet
body wash, chocolate, egg, fork, ketchup, knife, milk, mouth wash
apple, fork, ketchup, knife, orange, rose
bread, chocolate, fork, knife, mouth wash, shampoo, toilet paper
banana, bread, dressing, egg, lettuce, rose, toilet paper, yogurt
apple, chips, chocolate, coke, glove, hat, pillow, sheet, wine
chips, coke, face-mask, glove, hat, ketchup, rose
apple, banana, chips, coke, dressing, egg, orange, sheet, yogurt
bread, carrot, chips, coke, egg, face-mask, glove, hat, pork, wine

```

```

Console >
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:51:12 PM)
Association rule: [chips]-->[coke] [Support= 0.3, confidence= 1.0]
Association rule: [coke]-->[chips] [Support= 0.3, confidence= 1.0]
Association rule: [egg]-->[chocolate] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [perfume]-->[chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [rose]-->[chocolate] [Support= 0.6, confidence= 0.8]
Association rule: [chocolate]-->[rose] [Support= 0.6, confidence= 0.8]
Association rule: [wine]-->[chocolate] [Support= 0.45, confidence= 0.9]
Association rule: [chocolate]-->[wine] [Support= 0.45, confidence= 0.6]
Association rule: [egg]-->[rose] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [perfume]-->[rose] [Support= 0.4, confidence= 1.0]
Association rule: [perfume]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[perfume] [Support= 0.3, confidence= 0.6]
Association rule: [wine]-->[rose] [Support= 0.4, confidence= 0.8]
Association rule: [perfume]-->[rose, chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [perfume, rose]-->[chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [chocolate, rose]-->[perfume] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [chocolate, perfume]-->[rose] [Support= 0.4, confidence= 1.0]
Association rule: [perfume]-->[wine, chocolate] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[perfume, chocolate] [Support= 0.3, confidence= 0.6]
Association rule: [perfume, wine]-->[chocolate] [Support= 0.3, confidence= 1.0]
Association rule: [chocolate, perfume]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume]-->[perfume] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [wine]-->[rose, chocolate] [Support= 0.4, confidence= 0.8]
Association rule: [rose, wine]-->[chocolate] [Support= 0.4, confidence= 1.0]
Association rule: [chocolate, rose]-->[wine] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [chocolate, wine]-->[rose] [Support= 0.4, confidence= 0.8888888888888888]
Association rule: [perfume]-->[rose, wine] [Support= 0.3, confidence= 0.75]
Association rule: [perfume, rose]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[rose, perfume] [Support= 0.3, confidence= 0.6]
Association rule: [rose, wine]-->[perfume] [Support= 0.3, confidence= 0.75]
Association rule: [perfume, wine]-->[rose] [Support= 0.3, confidence= 1.0]
Association rule: [perfume]-->[rose, wine, chocolate] [Support= 0.3, confidence= 0.75]
Association rule: [perfume, rose]-->[wine, chocolate] [Support= 0.3, confidence= 0.75]
Association rule: [wine]-->[rose, perfume, chocolate] [Support= 0.3, confidence= 0.6]
Association rule: [rose, wine]-->[perfume, chocolate] [Support= 0.3, confidence= 0.75]
Association rule: [perfume, wine]-->[rose, chocolate] [Support= 0.3, confidence= 1.0]
Association rule: [perfume, rose, wine]-->[chocolate] [Support= 0.3, confidence= 1.0]
Association rule: [chocolate, perfume]-->[rose, wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume, rose]-->[wine] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, wine]-->[rose, perfume] [Support= 0.3, confidence= 0.6666666666666666]
Association rule: [chocolate, rose, wine]-->[perfume] [Support= 0.3, confidence= 0.75]
Association rule: [chocolate, perfume, wine]-->[rose] [Support= 0.3, confidence= 1.0]

Executing time: 252ms

```

## Data set 04:

```

Console [x]
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:53:45 PM)
please input the min_support value
0.35
the min_support value is 0.35
please input the min_confidence value
0.6
the min_confidence value is 0.6

The input data is following:
banana, chips, coke, dressing, lettuce, milk, yogurt
chips, coke, face-mask, glove, hat, milk, yogurt
body wash, chips, coke, milk, perfume, shampoo, yogurt
chips, coke, fork, glove, knife, milk, yogurt
apple, chips, coke, dish, milk, wine, yogurt
chips, coke, dish, fork, knife, milk, yogurt
apple, banana, chips, coke, orange, yogurt
apple, carrot, chips, coke, lettuce, milk, orange
chips, chocolate, egg, hat, milk, pillow, sheet, yogurt
bread, coke, egg, glove, milk, toilet paper, yogurt
apple, banana, chips, coke, orange, shampoo
chip, chocolate, coke, hat, pillow, sheet
chips, egg, fork, ketchup, knife, milk
apple, chips, chocolate, fork, ketchup, knife, milk, orange
banana, dish, egg, ketchup, milk, perfume, shampoo, wine, yogurt
knife, milk, mouth wash, shampoo, toilet paper, yogurt
banana, bread, chips, egg, rose, wine, yogurt
apple, glove, milk, orange, perfume, rose, wine
chocolate, egg, pillow, pork, sheet, yogurt
bread, carrot, coke, egg, face-mask, glove, hat, wine

Association rule: [chips]-->[coke] [Support= 0.5, confidence= 0.7142857142857143]

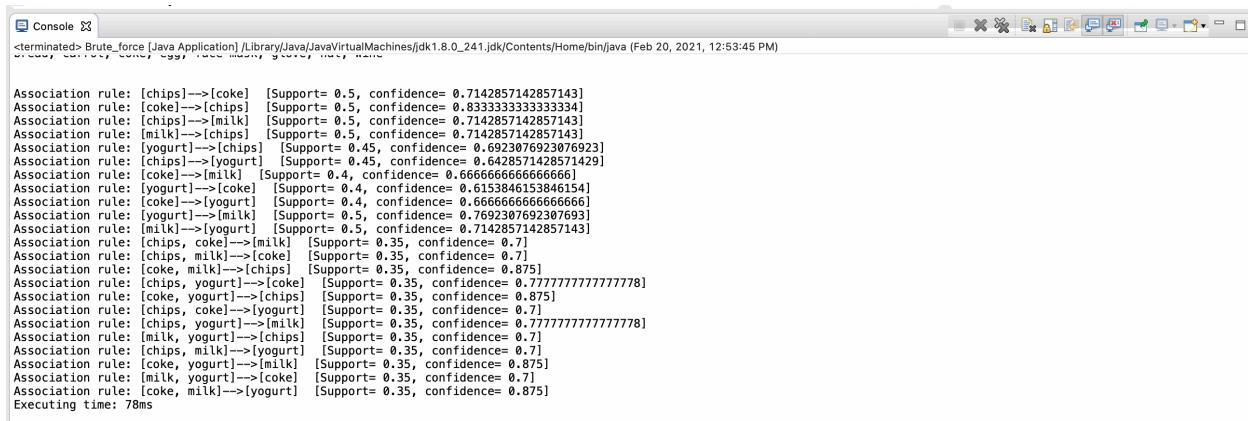
Console [x]
<terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:53:45 PM)
please input the min_support value
0.35
the min_support value is 0.35
please input the min_confidence value
0.6
the min_confidence value is 0.6

Association rule: [chips]-->[coke] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [coke]-->[chips] [Support= 0.5, confidence= 0.8333333333333334]
Association rule: [chips]-->[milk] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [milk]-->[chips] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [yogurt]-->[chips] [Support= 0.45, confidence= 0.6923076923076923]
Association rule: [chips]-->[yogurt] [Support= 0.45, confidence= 0.6428571428571429]
Association rule: [coke]-->[milk] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [yogurt]-->[coke] [Support= 0.4, confidence= 0.6153846153846154]
Association rule: [coke]-->[yogurt] [Support= 0.4, confidence= 0.6666666666666666]
Association rule: [yogurt]-->[milk] [Support= 0.5, confidence= 0.7692307692307693]
Association rule: [milk]-->[yogurt] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [chips, coke]-->[milk] [Support= 0.35, confidence= 0.7]
Association rule: [chips, milk]-->[coke] [Support= 0.35, confidence= 0.7]
Association rule: [coke, milk]-->[chips] [Support= 0.35, confidence= 0.875]
Association rule: [chips, yogurt]-->[coke] [Support= 0.35, confidence= 0.7777777777777778]
Association rule: [coke, yogurt]-->[chips] [Support= 0.35, confidence= 0.875]
Association rule: [chips, coke]-->[yogurt] [Support= 0.35, confidence= 0.7]
Association rule: [chips, yogurt]-->[milk] [Support= 0.35, confidence= 0.7777777777777778]
Association rule: [milk, yogurt]-->[chips] [Support= 0.35, confidence= 0.7]
Association rule: [chips, milk]-->[yogurt] [Support= 0.35, confidence= 0.7]
Association rule: [coke, yogurt]-->[milk] [Support= 0.35, confidence= 0.875]
Association rule: [milk, yogurt]-->[coke] [Support= 0.35, confidence= 0.7]
Association rule: [coke, milk]-->[yogurt] [Support= 0.35, confidence= 0.875]

Executing time: 78ms

```

## Data set 05:



```
Console <terminated> Brute_force [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home/bin/java (Feb 20, 2021, 12:53:45 PM)

Association rule: [chips]-->[coke] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [coke]-->[chips] [Support= 0.5, confidence= 0.8333333333333334]
Association rule: [chips]-->[milk] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [milk]-->[chips] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [yogurt]-->[chips] [Support= 0.45, confidences 0.6923076923076923]
Association rule: [chips]-->[yogurt] [Support= 0.45, confidences 0.6428571428571429]
Association rule: [coke]-->[milk] [Support= 0.4, confidences 0.6666666666666666]
Association rule: [yogurt]-->[coke] [Support= 0.4, confidence= 0.6153846153846154]
Association rule: [coke]-->[yogurt] [Support= 0.4, confidences 0.6666666666666666]
Association rule: [yogurt]-->[milk] [Support= 0.5, confidences 0.7692307692307693]
Association rule: [milk]-->[yogurt] [Support= 0.5, confidence= 0.7142857142857143]
Association rule: [chips, coke]-->[milk] [Support= 0.35, confidence= 0.7]
Association rule: [chips, milk]-->[coke] [Support= 0.35, confidence= 0.875]
Association rule: [chips, yogurt]-->[coke] [Support= 0.25, confidence= 0.7777777777777778]
Association rule: [coke, yogurt]-->[chips] [Support= 0.35, confidence= 0.875]
Association rule: [chips, coke]-->[yogurt] [Support= 0.35, confidence= 0.7]
Association rule: [chips, yogurt]-->[milk] [Support= 0.35, confidence= 0.7777777777777778]
Association rule: [milk, yogurt]-->[chips] [Support= 0.35, confidence= 0.7]
Association rule: [chips, milk]-->[yogurt] [Support= 0.35, confidence= 0.7]
Association rule: [coke, yogurt]-->[milk] [Support= 0.35, confidence= 0.875]
Association rule: [milk, yogurt]-->[coke] [Support= 0.35, confidence= 0.7]
Association rule: [coke, milk]-->[yogurt] [Support= 0.35, confidence= 0.875]
Executing time: 78ms
```

## Comparison of executing time:

Data set 01:

T(Apriori)= 15ms, T(Brute-Force)=89ms

Data set 02:

T(Apriori)= 14ms, T(Brute-Force)=103ms

Data set 03:

T(Apriori)= 12ms, T(Brute-Force)=252ms

Data set 04:

T(Apriori)= 9ms, T(Brute-Force)=78ms

Data set 05:

T(Apriori)= 11ms, T(Brute-Force)=78ms

It's obvious Apriori algorithm is faster than Brute-Force algorithm on each of the five dataset.