**CS-634 Data Mining**

**Midterm Project**

BY-

FNU Amandeep(fa349)

Table of Contents:

1. Introduction ………………………………………………………………… 3
2. Implementation ……………………...…………………………………...… 3
3. Code ………………………………...……………………………………… 4
4. Output Screenshots ………………………………...………………….....… 9
5. Introduction:

The implementation of Apriori Algorithm is made in this report. The algorithm is used for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties.

With such a rapid growth in product consuming market, companies seek to sell maximum in order to make maximum profit and for this patterns of shoppers need to identifies. By giving input of previous transactions to Apriori algorithm, it outputs the percentage of support and confidence for the products set purchased together. E-commerce market use this to suggest relevant products to the consumer and gaint store like Walmart use this to place frequently purchased itemset together. This in returns enhances the customers experience and gains to the store.

1. Implementation:

The code here is implemented in java. The user has to provide the database location to the program.

Here before the program starts the user has to specify the absolute path of the folder containing the databases where the databases are in text format.

As the program runs, it asks for minimum support and minimum confidence values which are in float format. With the database, minimum support and minimum confidence, the algorithm produces the association rules for the respective database. With that the program asks again for minimum support and minimum confidence for the next database and results are produced. It continues for all 5 databases. The program terminates with the output of the last database.

The association rules are in format->

ItemsetsA -> ItemsetsB [ Support , Confidence ]

1. Code:

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.Scanner;

**import** java.lang.Math;

**public** **class** Apriori {

**static** **float** *Min\_Confidence*=0f;

**static** **float** *Min\_Support*=0f;

**public** **static** **int** eliminate\_check(**int** num,**int**[] eliminated,**int** e\_size)

{

**int** is\_existing=0;

**for**(**int** i=0;i<e\_size;i++)

{

**int** num\_temp=num;

**int** temp=eliminated[i];

**int** flag=0;

**while**(temp!=0)

{

//System.out.println("temp "+temp+" num\_temp "+num\_temp);

//System.out.println("temp%10 "+temp%10+" " +" num\_temp "+num\_temp%10);

**if**(temp%10==1 && num\_temp%10==0)

{

flag=1;

**break**;

}

temp=temp/10;

num\_temp=num\_temp/10;

}

**if**(flag==0)

{

is\_existing=1;

**break**;

}

}

**return** is\_existing;

}

**public** **static** **void** name\_me(**int** num)

{

String[] items= {"Apple","Orange","Lemon","Ginger","Water","Milk","Chips","Diapers","iPhone","Case"};

**int** n=0;

**int** temp=num;

**while**(temp!=0)

{

**if**(temp%10==1)

n++;

temp=temp/10;

}

**int**[] set=**new** **int**[n];

temp=num;

**int** count=0;

**int** z=0;

**while**(temp!=0)

{

**if**(temp%10==1)

{

set[z]=count;

z++;

}

count++;

temp=temp/10;

}

**for**(**int** i=0;i<n;i++)

{

System.***out***.print(items[set[i]]+" ");

}

}

**public** **static** **int** merge(**int** itemset1,**int** itemset2)

{

String str1=Integer.*toString*(itemset1);

String str2=Integer.*toString*(itemset2);

//System.out.println(str1+" "+str2);

**int** diff=str2.length()-str1.length();

**if**(diff>0 )

{

**for**(**int** i=0;i<diff;i++)

{

str1="0"+str1;

}

}

**else** **if**(diff<0)

{

**for**(**int** i=diff;i<0;i++)

{

str2="0"+str2;

}

}

String to\_return="";

**for**(**int** i=0;i<str1.length();i++)

{

**if**(str1.charAt(i)=='0' && str2.charAt(i)=='0')

to\_return=to\_return+"0";

**else**

to\_return=to\_return+"1";

}

**int** merged\_number=Integer.*parseInt*(to\_return);

//System.out.println(str1+" "+str2+" "+to\_return);

**return** merged\_number;

}

**public** **static** **int** cal\_Support(**int**[][] d,**int** number)

{

**int** count=0;

**for**(**int** i=0;i<20;i++)

{

**int** c=number;

**int** flag=1;

**for**(**int** j=0;j<10;j++)

{

**if**(d[i][j]!=c%10 && c%10==1 )

flag=0;

c=c/10;

}

count=count+flag;

}

**return** count;

}

**public** **static** **int** flip(**int** n)

{

**if**(n==7)

**return** 8;

**else**

**return** 7;

}

**public** **static** **void** headtail(**int** n,**int** num,**int** d[][])

{

**int**[] set=**new** **int**[n];

**int** temp=num;

**int** count=0;

**int** z=0;

**while**(temp!=0)

{

**if**(temp%10==1)

{

set[z]=(**int**) Math.*pow*(10,count);

//System.out.println(set[z]);

z++;

}

count++;

temp=temp/10;

}

**int** c=(**int**) (Math.*pow*(2, n));

**int**[][] a=**new** **int**[c][n];

**for**(**int** j=0;j<n;j++)

{

**int** pq=(**int**)(Math.*pow*(2,j+1));

**int** bit=7;

**int** bc=0;

**for**(**int** i=0;i<c;i++)

{

**if**(bc==c/pq)

{

bit=*flip*(bit);

bc=0;

}

a[i][j]=bit;

bc++;

}

}

**int** left=0;

**int** right=0;

**for**(**int** i=1;i<c-1;i++)

{

left=0;

right=0;

**for**(**int** j=0;j<n;j++)

{

**if**(a[i][j]==7)

{

left=*merge*(left,set[j]);

}

**else**

{

right=*merge*(right,set[j]);

}

}

//System.out.println("num "+ num+" left "+left+" right "+right);

**float** confidence=((**float**)*cal\_Support*(d,num)/(**float**)*cal\_Support*(d,left))\*100;

**float** support=((**float**)*cal\_Support*(d,num) /20 )\*100;

//System.out.println("confidence "+confidence);

**if**(confidence>=*Min\_Confidence*)

{

*name\_me*(left);

System.***out***.print("->");

*name\_me*(right);

System.***out***.print("["+support+","+confidence+"]\n");

}

}

}

**public** **static** **void** associations(**int** num,**int** d[][])

{

**int** nOfOne=0;

**int** temp=num;

**while**(temp!=0)

{

**if**(temp%10==1)

nOfOne++;

temp=temp/10;

}

//System.out.println(nOfOne);

//int size=((int) Math.pow(2, nOfOne))-2;

*headtail*(nOfOne,num,d);

}

**public** **static** **void** Table\_creation(**int** d1[][])

{

//System.out.println("//Table c1");

//Table c1

**int** c1[][]=**new** **int**[2][10];

**for**(**int** i=0;i<10;i++)

{

c1[0][i]=(**int**) Math.*pow*(10,i);

c1[1][i]=*cal\_Support*(d1,c1[0][i]);

}

//Printing c1

// for(int i=0;i<10;i++)

// {

// System.out.println(c1[0][i]+" "+c1[1][i]);

// }

//calculating size for L1

**int** size=0; // size for next array -> L1 array

**for**(**int** i=0;i<10;i++)

{

**if**(((**float**)c1[1][i]/20)\*100>=*Min\_Support*)

{

//System.out.println(c1[0][i]);

size++;

}

}

//System.out.println("//Table l1");

//Table l1

**int** l1[][]=**new** **int**[2][size];

**int** initial=0;

**for**(**int** i=0;i<10;i++)

{

**if**(((**float**)c1[1][i]/20)\*100>=*Min\_Support*)

{

l1[0][initial]=c1[0][i];

l1[1][initial]=c1[1][i];

//System.out.println(l1[0][initial]+" "+l1[1][initial]);

initial++;

}

}

//System.out.println("initial "+initial);

**int** []eliminated=**new** **int**[100000];

**int** e\_size=0;

**int** c2[][];

**while**(size>1)

{

//System.out.println("//Table c2");

//Table c2

**int** c2\_size=(size\*(size-1))/2;

//System.out.println("size="+size+" c2\_size "+c2\_size);

c2=**new** **int**[2][c2\_size];

**int** z=0; // to increment value in c2

**for**(**int** i=0;i<size;i++)

{

**for**(**int** j=i+1;j<size;j++)

{

**if**(l1[0][i]==l1[0][j])

**continue**;

c2[0][z]= *merge*(l1[0][i],l1[0][j]);

c2[1][z]=*cal\_Support*(d1,c2[0][z]);

//System.out.println(z+" "+ c2[0][z]+" "+c2[1][z]);

z++;

}

}

//calculating size for L2

**int** sizeL2=0; // size for next array -> L1 array

**for**(**int** i=0;i<z;i++)

{

**if**(((**float**)c2[1][i]/20)\*100>=*Min\_Support* && *eliminate\_check*(c2[0][i],eliminated,e\_size)==0 )

{

**if**(*eliminate\_check*(c2[0][i],c2[0],i)==0)

{

//System.out.println(c2[0][i]+" sizeL2 "+sizeL2);

sizeL2++;

}

}

}

//System.out.println("sizeL2 "+sizeL2);

//System.out.println("//Table lL2");

//Table L2

l1=**new** **int**[2][sizeL2];

**int** initialL2=0;

**for**(**int** i=0;i<z;i++)

{

**if**(((**float**)c2[1][i]/20)\*100>=*Min\_Support* && *eliminate\_check*(c2[0][i],eliminated,e\_size)==0 )

{

**if**(*eliminate\_check*(c2[0][i],c2[0],i)==0)

{

l1[0][initialL2]=c2[0][i];

l1[1][initialL2]=c2[1][i];

//System.out.println(l1[0][initialL2]+" "+l1[1][initialL2]+" "+initialL2);

*associations*(l1[0][initialL2],d1);

initialL2++;

}

}

**else**

{

eliminated[e\_size]=c2[0][i];

e\_size++;

}

}

size=sizeL2;

}

}

**public** **static** **void** printdatabase(**int**[][] d1)

{

**int** t=101;

System.***out***.print("\t");

String[] items= {"Apple","Orange","Lemon","Ginger","Water","Milk","Chips","Diapers","iPhone","Case"};

**for**(**int** i=0;i<10;i++)

{

System.***out***.print(items[i]+"\t");

}

System.***out***.println();

**for**(**int** i=0;i<20;i++)

{

System.***out***.print("T"+t+"\t");

**for**(**int** j=0;j<10;j++)

{

System.***out***.print(d1[i][j]+"\t");

}

System.***out***.println();

t++;

}

}

**public** **static** **void** scanfile(**int** d1[][],**int** db) **throws** FileNotFoundException

{

File file = **new** File("/Users/amandeep/Desktop/database"+db+".txt");

Scanner sc1=**new** Scanner(file);

sc1.nextLine();

System.***out***.println();

**for**(**int** i=0;i<20;i++)

{

sc1.next();

**for**(**int** j=0;j<10;j++)

{

//System.out.println(sc1.next());

d1[i][j]=Integer.*parseInt*(sc1.next());

}

}

sc1.close();

*printdatabase*(d1);

}

**public** **static** **void** main(String[] args) **throws** FileNotFoundException {

Scanner sc=**new** Scanner(System.***in***);

**for**(**int** db=1;db<6;db++)

{

System.***out***.print("Enter the Minimum Support for dataBase"+db+" :");

*Min\_Support*=sc.nextFloat();

System.***out***.print("Enter the Minimum for dataBase"+db+" :");

*Min\_Confidence*=sc.nextFloat();

**int** d1[][] = **new** **int**[20][10];

*scanfile*(d1,db);

System.***out***.println("\nFor DataBase"+db+" with Min Support= "+*Min\_Support*+" MinConfidence="+*Min\_Confidence*);

System.***out***.println("+++++------ASSOCIATION RULES------+++++");

*Table\_creation*(d1);

System.***out***.println("\n");

}

sc.close();

}

}

1. Output Snapshots:











