МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Лабораторна робота №6

з курсу

«Інженерія даних та знань»

*на тему:*

**«Розробка додатку для роботи з асоціативними правилами»**

Виконав:

ст. гр. ITIT-12

Кобак Р.З.

Прийняла:

Рибчак З.Л.

**Львів 2017**

**Мета роботи:**

Вивчення алгоритму формування асоціативних правил та його застосування для

розв’язування задач аналізу даних.

**Індивідуальне завдання:**

**Варіант 3**

**Варіант 1**

Реалізувати алгоритм пошуку асоціативних правил та сформувати множину асоціативних правил на основі таблиці з такими параметрами: minsupport = 70%, minconfidence = 70%.

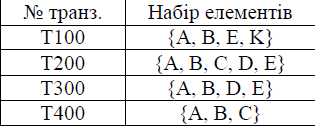


Рис.1 Таблиця з вхідними даними

**Порядок виконання:**

Реалізовано додаток, що обробляє статистичні дані транзакцій в форматі CSV (Comma Separated Value), та генерує асоціативні правила на основі отриманих даних з заданими пороговими значеннями підтримки та достовірності.

**Код програми:**

**public class** AprioriAlgorithm {  
  
 **protected** TransactionSource **source**;  
 **protected** List<Itemset> **transactions**;  
  
 **public** AprioriAlgorithm() {  
 }  
  
 **protected void** log(String s) {  
 System.***out***.println(s);  
 }  
  
 **public void** setSource(TransactionSource transactionSource) {  
 LabelsMap.*getInstance*().clear();  
 SupportMap.*clear*();  
 **source** = transactionSource;  
 }  
  
 **public** List<Itemset> getTransactions() {  
 **transactions** = **source**.generate();  
 log(**"The following transactions have been loaded: "**);  
 **for** (Itemset i : **transactions**) {  
 log(**"\t> "** + i);  
 }  
 **return transactions**;  
 }  
  
 **public** List<Rule> run(**int** support, **int** confidence) {  
  
 log(**"Launching A Priori algorithm with following parameters:"**);  
 log(**"Support (absolute) min: "** + support);  
 log(**"Confidence min: "** + confidence);  
  
 **double** begin = System.*currentTimeMillis*();  
 SupportMap.*setup*(**transactions**);  
 log(**"Computing frequents of size 1..."**);  
 List<Itemset> previousLevel = **new** ArrayList<>();  
 **for** (**int** i : LabelsMap.*getInstance*().getAllProducts()) {  
 Itemset temp = **new** Itemset();  
 temp.add(i);  
  
 **int** itemSupport = temp.getRelativeSupport();  
 log(**"\t> Item: "** + temp + **" (support: "** + itemSupport + **")"**);  
  
 **if** (itemSupport >= support) {  
 log(**"\t\t=> This item is a frequent itemset of size 1."**);  
 previousLevel.add(temp);  
 } **else** {  
 log(**"\t\t=> This item isn't a frequent itemset of size 1."**);  
 }  
 }  
  
 log(**"Computing frequents of higher size..."**);  
 List<Itemset> allFrequents = **new** ArrayList<>(previousLevel);  
 List<Itemset> nextLevel = **null**;  
 **int** level = 1;  
 **while** (previousLevel.size() >= 2) {  
 log(**"\t> Computing frequents of size "** + ++level);  
 nextLevel = **new** ArrayList<>();  
  
 **for** (**int** i = 0, s = previousLevel.size(); i < s; ++i) {  
 Itemset is1 = previousLevel.get(i);  
 **for** (**int** j = i + 1; j < s; ++j) {  
 Itemset is2 = previousLevel.get(j);  
 List<Integer> prefix = **new** ArrayList<>(is2);  
 **int** last = prefix.remove(prefix.size() - 1);  
  
 Itemset generated = **new** Itemset(is1);  
 generated.add(last);  
 **if** (!SupportMap.*alreadyProcessed*(generated)) {  
 **if** (is1.containsAll(prefix)  
 && previousLevel.containsAll(generated  
 .subsetWithoutOneElement())) {  
 **int** generatedSupport = generated  
 .getRelativeSupport();  
 log(**"\t\t>> Generated itemset: "** + generated  
 + **" (support: "** + generatedSupport + **")"**);  
  
 **if** (generatedSupport >= support) {  
 log(**"\t\t\t>> Frequent itemset."**);  
 nextLevel.add(generated);  
 } **else** {  
 log(**"\t\t\t>> Not frequent."**);  
 }  
 }  
 }  
 }  
 }  
  
 allFrequents.addAll(nextLevel);  
 log(**"\t> All the frequent itemsets of size "** + level  
 + **" are the following: "**);  
 **for** (Itemset itemset : nextLevel) {  
 log(**"\t\t>> "** + itemset);  
 }  
  
 previousLevel = nextLevel;  
 }  
  
 log(**"There are "** + allFrequents.size() + **" frequent itemsets"**);  
 log(**"Generating rules..."**);  
 **int** generatedRules = 0;  
 List<Rule> ruleList = **new** ArrayList<>();  
 **for** (Itemset frequent : allFrequents) {  
 **for** (Rule r : frequent.generateRules()) {  
 **double** ruleConfidence = r.confidence();  
 log(**"\t> Considering the rule "** + r + **"..."**);  
 **if** (ruleConfidence >= confidence) {  
 ++generatedRules;  
 log(**"\t\t=> Rule kept."**);  
 ruleList.add(r);  
 } **else** {  
 log(**"\t\t=> Confidence too low."**);  
 }  
 }  
 }  
 log(generatedRules + **" rules have been generated."**);  
  
 log(**"Total duration: "** + ((System.*currentTimeMillis*() - begin) / 1000.)  
 + **"s"**);  
  
 **return** ruleList;  
 }  
}

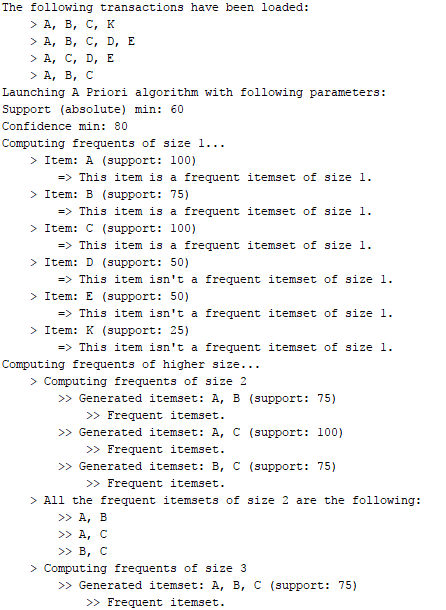
**Вхідні дані:**A,B,C,K  
A,B,C,D,E  
A,C,D,E  
A,B,C

**Результат виконання програми:**

Результат виконання програми представлений на Рис 2. При заданих 60% підтримки і 80% достовірності було згенеровано 7 асоціативних правил. Час виконання операції 0,007 секунди.

**Висновок:**

При виконанні даної лабораторної роботи було реалізовано додаток побудови асоціативних правил на основі статистичних даних з заданням мінімальних значень. Повний код програми знаходиться на репозиторії за даною URL- адресою: https://github.com/amukysh/EngineeringOfDataAndKnowledge.git/



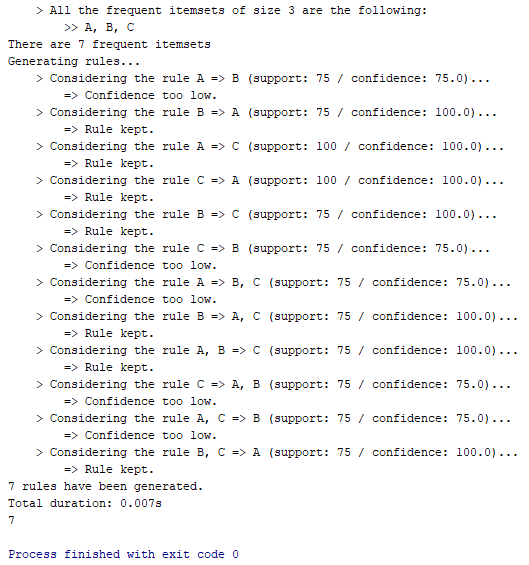


Рис.2 Результат виконання програми