ASSIGNMENT 1

Generating frequent item sets and interesting association rules using Apriori and FP-growth algorithm.

by

Shreyansh Garg(2017A7PS1730H)

Jain Jai Sandeep(2017A7PS1585H)

Vrutik Halani(2017A7PS1732H)

Apriori:

Dataset used: Car evaluation dataset

Programming language used: Python

Code Description:

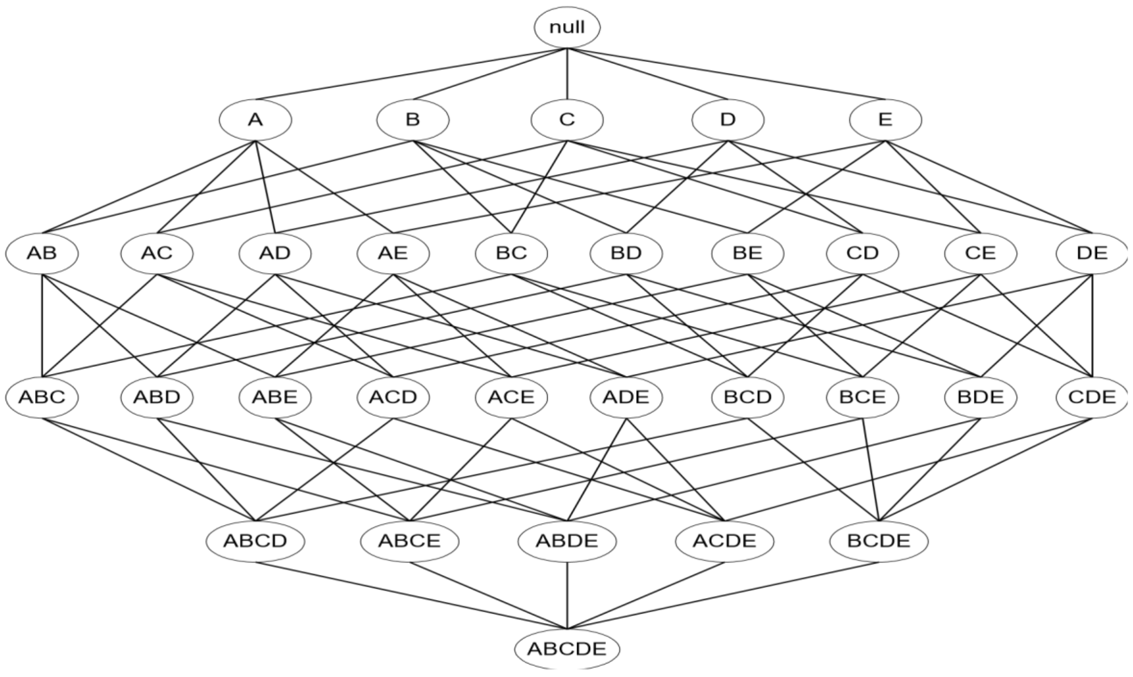
The data in try.txt is read and stored in a variable.

- **Apriori principle: - If an itemset is frequent, then all of its subsets must also be frequent**

  Candidate generation:

  a) Joining: The Apriori algorithm is used to generate frequent item sets. We first generated frequent item sets of size 1 and using it we generated progressively larger item sets.

  b) Pruning: From the generated items, those containing infrequent subsets are pruned from list thus reducing its subsets. This is done in the Pruning step.



- Fig1: showing all possible item sets

  To reduce complexity, pruning is used which reduces number of comparisons. This is shown in the following figure.

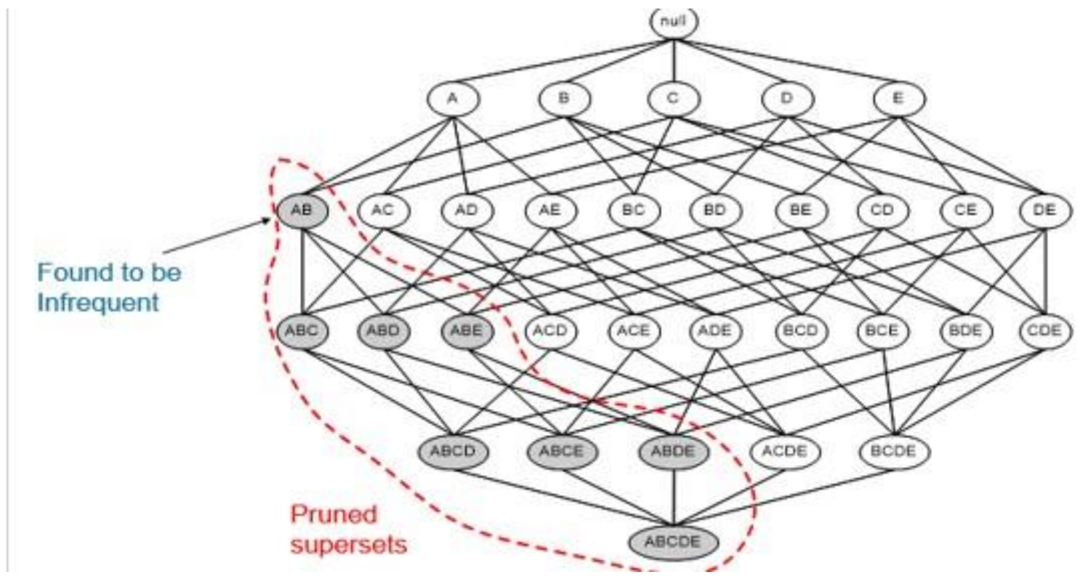


Fig2: Pruning

- The support values of all the frequent item sets are stored in a corresponding list to be used in the next step for calculating support of subsets in generating association rules and finding confidence.
- Then Association rules are generated using the frequent item sets generated in the above step. For each itemset with size greater than 1, all the possible subsets are generated and rules are formed based on the confidence values.

Formulae used:

$$\text{SUPPORT} = \frac{\text{number of transactions containing X and Y}}{\text{total number of transactions}}$$

$$\text{CONFIDENCE} = \frac{\text{number of transactions containing X and Y}}{\text{number of transactions containing X}}$$

Functions used:

Check_support(x,mins):
Checks whether the support of itemset is more than minsup or not. If it is less than minsup it removes that element.

printit(keys):
Prints the frequent item sets along with their support count.

join1(ab,a):
used to create item sets of length 2

ujoin(prev,lp):
ujoin is used to create item sets of length lp+1 by using frequent item sets of length lp

maximal(tree):
used to get all maximal frequent item sets

itemsets_for_rules(tree):
generates rules without confidence

update_support(newkeys,a):
updates the support of item sets and returns the dictionary.

closeitems(tree):
gives all the closed item sets

update_rules(leftr,rightr,confi,mincon)
updates confidence of rules and if less than mincon pops it

makerules(ruleitems,tree):
generates all rules and their confidence

print_rules(l,r,c):
prints all the rule

Number of frequent item sets & association rules:

1. At min_Support=50 and min_Confidence=0.3
   Total number of frequent item sets generated = **521**
   Total number of association rules generated = **683**
   Maximal frequent item sets = **522**
   Closed frequent item sets = **543**

2. At min_Support=100 and min_Confidence=0.6
   Total number of frequent items sets generated = **296**
   Total number of association rules generated = **76**
   Maximal frequent item sets = **296**
   Closed frequent item sets = 317

3. At min_Support=150 and min_Confidence=0.9
   Total number of frequent item sets generated = **65**
   Total number of association rules generated = **10**
   Maximal frequent item sets = 65
   Closed frequent item sets = 86

Rule Mining using FP Trees

Language Chosen

Java (Development Environment - Eclipse)

Pre-processing

- Since the attributes are continuous, the first step in pre-processing was to discretize the continuous data. The following classification principles were used to form subclasses of each of the attributes.
- 0s in attributes apart from No. of pregnancies and the class to identify whether the person was tested positive for diabetes indicate missing values (This is due to the fact that attributes like Blood Pressure clearly cannot be 0 unless the data is missing). The second step of preprocessing involved handling this missing data. The mode of the attribute (i.e.. the class which occurs the most) replaces the missing value.
- Finally, the 9-attribute data was expanded to effectively a 40-attribute itemset (based on the number of sub-classes for

each class). But, at a time since only one subclass can be present in the basket, each transaction is exactly a row of 9 items.

- The support count of each of the attributes was counted (i.e.. one frequent item sets generated).
- The attributes within the transactions were sorted in descending order of their support counts, so that the FP tree is not very broad. For e.g. After this preprocessing, the number of children of the root node is only 4, i.e all transactions begin with only one of those 4 attributes.

Compilation Steps

- javac FP_Growth.java
- Java FP_Growth

The program then runs for the pre-specified Minimum Support and Minimum Confidence threshold values.

Support and confidence value at which interesting rules are generated

Support: 153

Confidence: 0.9

No of Rules Generated: 4

Frequent itemsets : 70

Support: 115

Confidence: 0.9

No of Rules Generated: 13

Frequent itemsets : 123

Support: 130

Confidence: 0.8

No of Rules Generated: 31

Frequent itemsets : 98