



# CP465 Database II Design Document


***Data Mining Algorithms***

***Instructor: Ilias Kotsireas***

***Students: Chris Ye, Evgheni Naida***

***Student ID: 104160010, 090305930***

2013/11/30



# Design Document

## Table of Contents

1.	Introduction .....	2
1.1	Purpose .....	2
1.2	Scope .....	2
1.3	Overview .....	2
2.	Implementation Description .....	2
2.1	Description of implementation features .....	2
2.2	Description of design choices made .....	3
2.3	Description of Software used .....	4
2.4	Instructions for the code .....	4
2.5	Contributors .....	4

## 1. Introduction

### 1.1 Purpose

The Objective of the project is to implement two Data Mining algorithms which is ID3 and Apriori algorithm, and test these implementations with various reasonably large datasets. The set is to be interpreted as anywhere from a few hundreds to a few thousands instances.

### 1.2 Scope

Data Mining is an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. This implement is to test two algorithms of data mining.

### 1.3 Overview

The Data Mining design document summarizes the following:

- Description of what features work and what features not work.
- Description and justification of the design choices made.
- Description of data structures used.
- Instructions on how to compile and run the code.
- Contributors

## 2. Implementation Description

### 2.1 Description of implementation features

**ID3:** For ID3 algorithm, the purpose of this program is only to demonstrate some concepts of ID3 algorithm, the output of the program is not an actual tree, and it more looks like description of the decision tree. The algorithm will automatically create an output file named 'test\_document.txt'. One of the issues with code is that when the decision tree is created it is copied from console to the text file, in the process of copying every byte an error occurs, causing ' }; ' to be shifted to next line.

**Apriori:** The project name is APR and inside there is 'apriori.java' class which is the starting point for the program and algorithm, and 'AprioriAlgorithm.java' which is the algorithm itself. 'Apriori.java' and 'AprioriAlgorithm.java' are commented using Javadoc and in addition, the authors have added their own comments and remarks. APR consists of two classes: apriori and AprioriAlgorithm. Apriori contains main and essentially launches the algorithm by creating an object of class AprioriAlgorithm and then calls Process method which executes the algorithm. AprioriAlgorithm is flexible, and it allows the user to use multiple transaction files because the user can customize the item separator used in transactions. Also, the user can change what values the algorithm counts as TRUE for attributes/columns. For example, from 'y' to '1' to 'true' to 'x', etc.

## 2.2 Description of design choices made

**ID3:** The dataset file downloaded from the website does not contain the attributes, just tuples. My design of the code will not take the layout of the original file; since the algorithm will output a tree, the choice was made to add attribute names on line 1. For example, user who wants to use the code has to add attribute names to line 1 into the dataset.

**Apriori:** For the design choice, AprioriAlgorithm and its methods are described below:

- **Process()** - starts off the program, loads in the configuration file and proceeds to generate candidates and calculates frequent itemsets
- **getConfig()** - loads in the configuration file, also allows changes to item separator and TRUE values of attributes/columns. In addition, writes some of initial setup information into the header of output file.
- **generateCandidates(int set)** - generates all possible candidates for an itemset, by looking at previous frequent itemsets that met min support requirements. Considers special cases of set1 and set2. Stores new candidates into 'candidates' vector.
- **calculateFrequentItemsets(int n)** - takes the candidates generated by generateCandidates(int set) and calculates frequency, compares with min support and creates an itemset. Writes all valid itemsets into 'output.txt'.
- **getInput()** - gets user input from System.in.

## 2.3 Description of Software used

Both ID3 algorithm and Apriori algorithm are designed in Java language using Eclipse IDE.

## 2.4 Instructions for the code

**ID3:** First, download one reasonably large dataset and save as a text file or copy and paste those tuples from the website and save as a text file. Second, open the 'dataset.names' file online then write attributes one by one on the top of the text file, use ',' as an item separator. Third, run the code in Eclipse then there will be a text file named test\_document.txt automatically created for the decision tree. The dataset used can be found here:

<http://archive.ics.uci.edu/ml/machine-learning-databases/adult/>.

**Apriori:** For the program to run and compile there needs to be at least 2 text files in the root directory of APR called 'config.txt' and 'transactions.txt', after running the algorithm a third file will be generated - 'output.txt'. The description of what needs to be in the files is below.

The program must be able to locate 'config.txt' and 'transactions.txt', otherwise an exception is triggered and program stalls. The original implementation comes with all 3 files in the directory. The dataset used can be found here:

<http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records>

### Files:

- config.txt - contains 3 lines. 1st is number of attributes, 2nd - number of total transactions or tuples, 3rd - minimum support.
- transactions.txt - contain the transactions or the database itself, where attributes are separated and the number of transactions and attributes matches that of config file.
- output.txt - is generated when the algorithm was executed. Will contain the header, outlining the number of transactions, number of attributes, min support and all the TRUE values for columns.

## 2.5 Contributors

This standard was developed by the Chris Ye and Evgheni Naida in Winter 2013 class of Database II (CP 465).

Code for Apriori and development of design document for Apriori part: Evgheni Naida

Code for ID3 and development of design document for ID3 part: Chris Ye