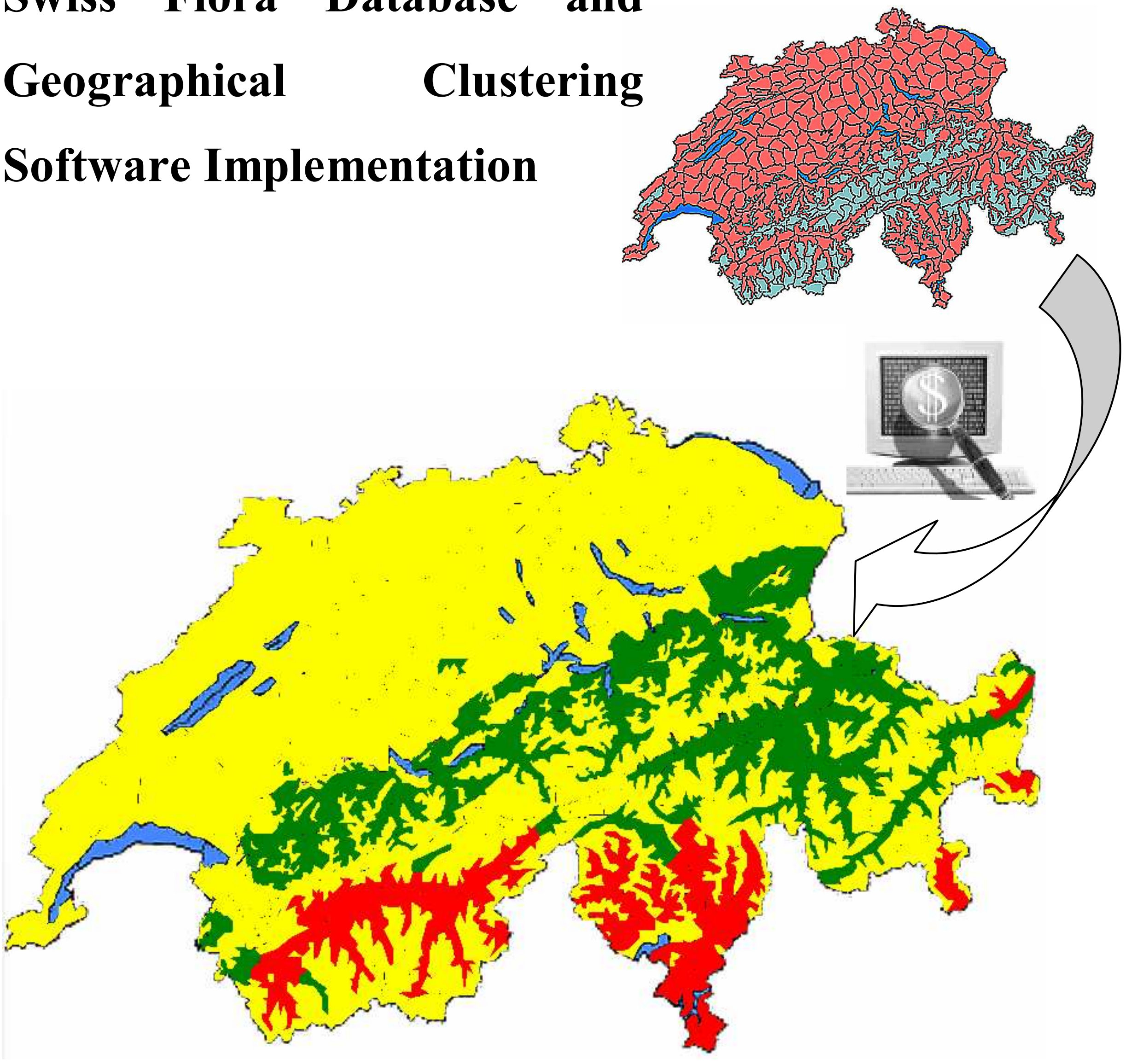
Practical Data Mining on a Swiss Flora Database and Geographical Clustering Software Implementation



AUTHORS:

Óliver Centeno
Javier T. Mediano

SUPERVISOR:

Yifeng Zeng

Faculty of Engineering and Science
Aalborg University

Department of Computer Science

TITLE:
Practical Data Mining on a Swiss Flora Database and Geographical Clustering Software Implementation

SEMESTER PERIOD:
DAT6,
1st of February – 5th of June 2006

PROJECT GROUP MEMBERS:
Óliver Centeno, oliver@cs.aau.dk
Javier T. Mediano, springer@cs.aau.dk

SUPERVISOR:
Yifeng Zeng, yfzeng@cs.aau.dk

NUMBER OF COPIES: 4

PAGES IN REPORT: 9 – 98

PAGES IN APPENDIX: 99 – 152

TOTAL NUMBER OF PAGES: 155

ABSTRACT:

This thesis document deals with the context of data mining. We explain various clustering techniques such as partition-based techniques and probabilistic approaches.

We have implemented the k-means clustering algorithm, the trimmed k-means variation of it and the Naïve Bayes with EM learning for clustering. We have compared the results of these algorithms applied to the given database.

Furthermore, we have implemented a tool where these clustering techniques can be applied, and the results of those techniques can be shown on a map of the geographical area where the data come from.

Preface

This project is written at Department of Computer Science, Aalborg University, by Óliver Centeno and Javier T. Mediano during DAT6 (8th semester). The project was written in the spring semester of 2006 to be evaluated on the 20th of June, 2006.
This project was supervised by Yifeng Zeng, whom we would like to thank for his participation in supervising the development of the project.
A special thanks to Manfred Jaeger for his assistance and participation during the project.

Aalborg University, June 2006

Óliver Centeno

Javier T. Mediano

Contents

1. Introduction

This chapter is dedicated to introduce the motivation behind this project and define the goals we want to achieve with it. A brief description of the solutions to the problem and the software we developed are presented.

We will explain what Data Mining is and how we can use it to get our goals.

1.1. Data Mining

Data mining is the process of extracting information patterns from a large data set. These patterns are implicit, non-trivial, unknown, and potentially useful. In this process we will use computational techniques from statistics and pattern recognition.

Often data mining is confused with deductive processing of databases, expert system, statistical analysis, visualization of data or small programs of learning but data mining is all the described process above.

The typical problem in data mining is classification, the task of classification occurs in a wide range of human activity. At its broadest, the term could cover any context in which some decision or forecast is made on the basis of currently available information, and a classification procedure is then some formal method for repeatedly making such judgements in a new situation.

The construction of a classification procedure from a set of data for which the true classes are known has also been variously termed pattern recognition, discrimination, or supervised learning. Contexts in which a classification task is fundamental include, for example, mechanical procedures for sorting letters on the basis of machine-read postcodes, assigning individuals to credits status on the basis of financial and other personal information, and the preliminary diagnosis of a patient's disease in order to select immediate treatment while awaiting definitive test results. In fact, some of the most urgent problems arising in science, industry and commerce can be regarded as classification or decision problems using complex and often very extensive data.

1.1.1. KDD

In the elaboration of this project, we will use the KDD process, KDD is an acronym meaning Knowledge Discovery from Databases. The KDD is a complete process of extracting knowledge from database.

The KDD started in 1989 to emphasize in the concept that knowledge is the final result of a process of discovery guided by the data. This means all the process you need to do to get the knowledge from a database. The data mining is a phase of the full process.

The KDD process contains:

- The use of the database, pre-process, transformation.
- Algorithms to obtain patterns/models from the data.
- Evaluation of the algorithms' results and select those models that can consider knowledge.

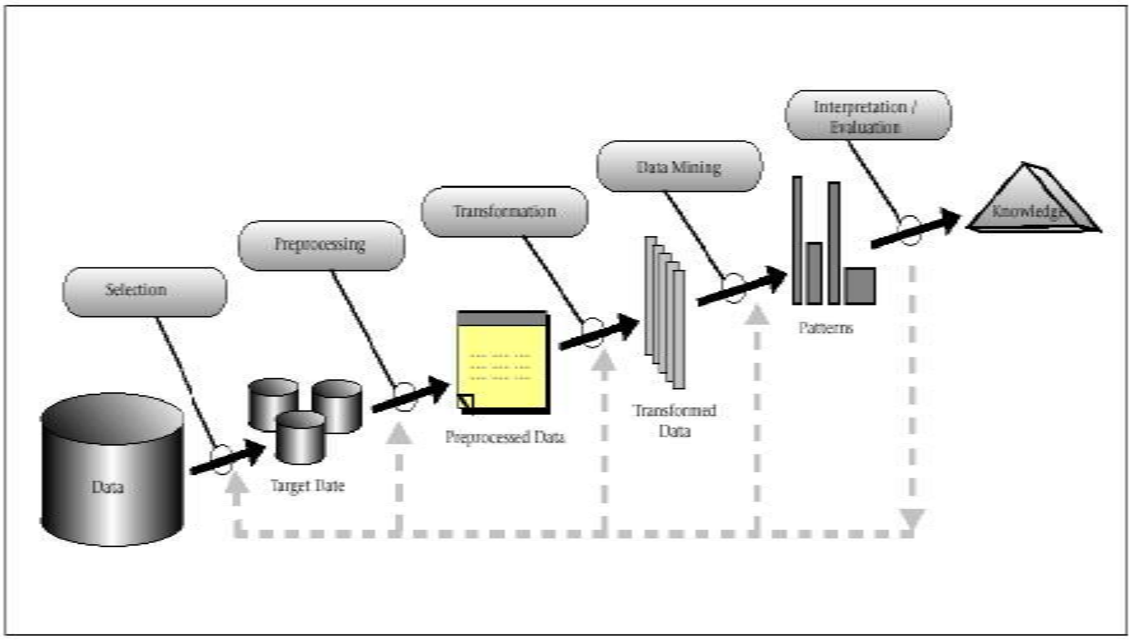


Figure 1.1-1: The KDE process

Phases of the process:

1. **Understanding of the domain of application of the problem, identification of a priori knowledge and creation of the Data Warehouse.**

   The understanding of the domain of application of the problem will allow to obtain a priori knowledge and to reduce the space of possible solutions, which will give more efficiency to the rest of the process. In this phase is important the unification of the information in Data Warehouse from several databases.

2. **Selection of data, data cleaning and pre-processing.**

   - From the result of the previous phase selects a data set using analysis and visualizations of the data.
   - Data cleaning: The missing values fill up, identify and eliminate the outliers, the noise is smoothed and inconsistencies are eliminated
   - Pre-processing: Transformation of the data, variables, values.

3. **Sieve of data: Reduction and projection.**

   - Reduction of cases, rows: This is made by techniques like compression of sampling of the data, or election of representatives (clustering).

- Projection: To select the set of attributes adapted for the specific task to make.

4. **Selection of the data mining technique and application of concrete data mining algorithms.**

   In this moment of the process we have to ask us what kind of knowledge are we searching? What technique is the most adapted? Will we need uncertainty?

5. **Evaluation, interpretation and presentation of the obtained results.**

   In the last phase we can obtain many hypotheses of models, it will be necessary to decide that models are but the valid ones. In order to do that is usual to run independent test set.

6. **Diffusion and use of the new knowledge.**

   The last phase of the process, it is time to make final reports, use the knowledge or/and incorporate it in an existing systems.

1.2. Motivation

The project we are involved in is the plant data mining. For numerous geographic regions extensive data collections exist related to the distribution of plant species in that area. For this collections the region is partitioned into a large number of small units (typically just using a regular grid that is imposed on the region), and for each unit in the partition, and for all plant species, it is recorded whether the species occurs or not in the unit.

Using clustering techniques from Machine Learning, one can use this data, for example, to find natural sub regions exhibiting relatively uniform vegetation (typically corresponding to regions with certain climatic and/or geological characteristics). Furthermore, one can classify the plants according to their distribution patterns.

One of the simplest possible models that can be used in this task is a Naïve Bayes model, represented by a Bayesians Network with one root node for the unobserved region type, and one child for every plant species. After learning parameters for this data from the distribution data, one can cluster the areas according to their most probable regions type, according to the learned model. More sophisticated models can be designed, which also take the spatial relationship between neighbouring regions into account.

1.3. Initial problem

For this project plant distribution data from Switzerland is available. Data from other geographic regions can perhaps also be obtained so it can be interesting to develop a tool that integrates several clustering techniques and able to show the clustered regions on a map of the land.

The main purpose of the project is to analyze this plant distribution applying several Machine Learning techniques making emphasis in the use of Bayesian methods as the Naïve Bayes model.

A second goal is to develop a tool to show the clustering results for this kind of data on the corresponding map of the region where the data come from. This tool can also contain some methods to compare two clustering results in order to identify the equivalences of clusters among both results and to show the instances that differ from one clustering to the other.

1.4. Description of the solution

Once we understood the problem we started to think how we could solve it. We had a huge database and we did not know much about the data contents. We began understanding the data, the different attributes that it has and the different values that each attribute can get. The idea was to get a complete general view of the database meanings.

After that, we decided to perform a first approach to take a look at the kind of information we could extract from that database. For doing that we used Weka, an application in data mining which support different knowledge analysis algorithms[1]. We used the K-Means algorithm[2] included in Weka's API for the first clustering approach making different tests transforming de original database. The changes we made were relative to the abundance values.

When we took this first approach we realized that the transformations of the dataset were useful for further work on the project so we decided to define three dataset for the rest of the experiments. Other conclusion that we obtained from the first approach was that the results that Weka offers are difficult to analyze. It offers the centers of the clusters, the amount of members in each cluster but not a relation of the cluster membership. It seemed to us that this was not enough information because, in our case with the Swiss flora, we found that could be more interesting to show the clustering results in a map of Switzerland.

Because of this we decided to implement an application with possibilities to apply clustering techniques and show the results in the Swiss map. This application is going to be described in Chapter 4 and the user's manual can be found in Appendix D. The main utility of the application is making possible to apply some clustering methods and see the results with the same dataset.

We obtained the map of Switzerland from the Webflora's site at `http://www.wsl.ch/land/products/webflora/`.. In this map the different areas are shown separated by lines so the main problem we had was finding a way to draw in a

[1] See Appendix C for further details about Weka.
[2] See Appendix A for further details about the K-Means algorithm and other clustering techniques.

selected area. To do that, we needed some references of each area to be able to decide where to draw. We decided to get the points of the contour of each area and, for that purpose, we were looking for an application that could do that easily.

As we did not find anyone to do that automatically we tried to extract the points manually. In this way we found an application which stores the points of a polygon that we define by clicks with the mouse. The application we used was Vextractor 3.10 demo version.

We created a polygon for each area, going through the areas in the order imposed by the dataset[3]. This was a long work, because there were many areas and some of them small and mixed with other ones; but this work was made and the points were stored in a text file in which there was one line for each area with the point of the polygon formed for it.

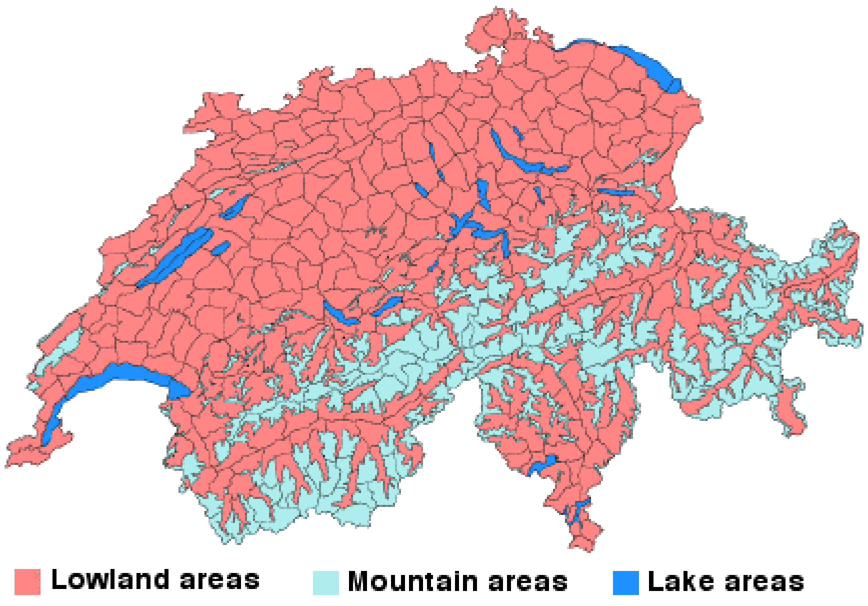


Figure 1.4-1: Original map from the Webflora's site

Figure 1.4-1 shows the map that we found in the Webflora's site. It contains all areas in which the land has been divided and it differences between 3 kinds of regions: lowlands, mountain areas and lake areas. We have to mention that the lake areas are not included in the database, so we are not going to cluster that kind of areas. This means that lake areas will always appear as they do in this map.

To put this map into the application we transformed it to make it clearer. We rescaled the map to do it bigger and sharpened it to show the areas more defined. The adapted map is showed in Figure 1.4-2.

With all these modifications now we could draw in the map the clustering results obtained from some clustering method, so the next steep was to implement those methods that the application was going to support and the functions necessary to show the results in the map. We considered implementing four clustering methods, two partition-based methods and two probabilistic clustering methods.

[3] A description of the order imposed to the areas is shown on Chapter 2.

The two partition-based methods are the K-Means algorithm and a variation of it, the Trimmed K-Means[4]. These methods have been implemented based on the approach found in Weka's API introducing the necessary modifications to adapt the method to our needs. For the K-Means algorithm we mainly introduced the clustering assignments, necessary to know which area is in each cluster. For the Trimmed K-Means we had to insert more modifications.

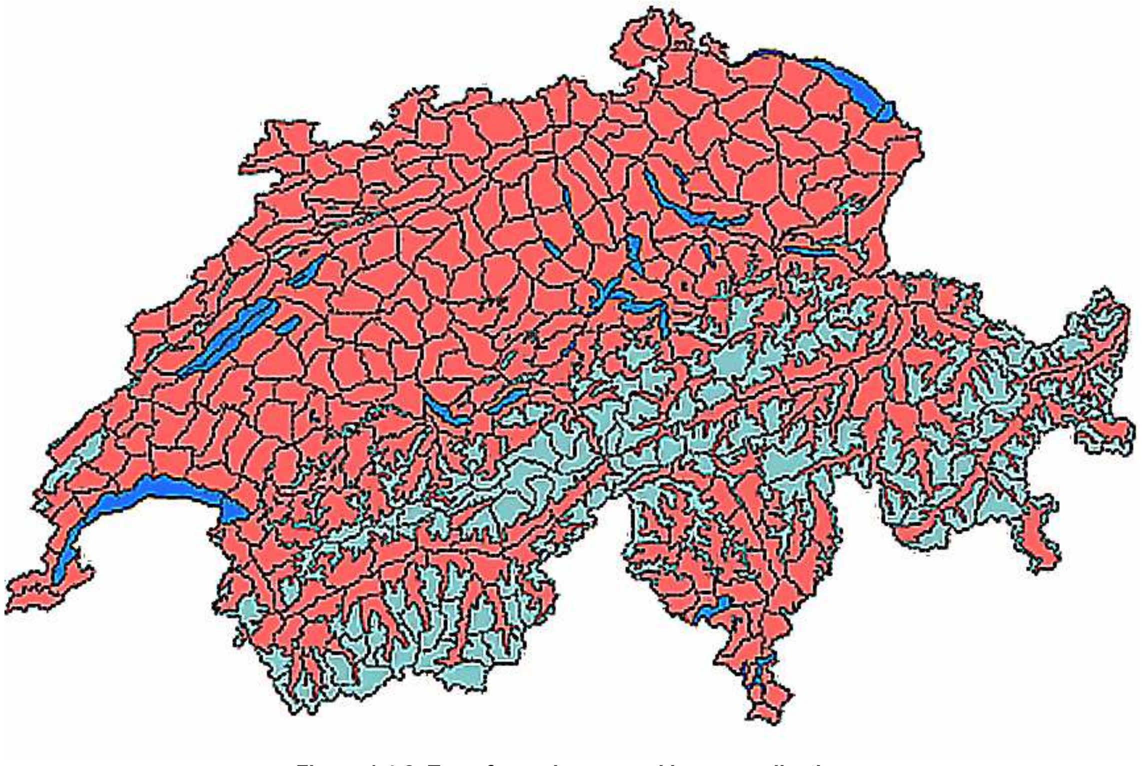


Figure 1.4-2: Transformed map used in our application

The probabilistic methods we decided to implement were a Naïve Bayes approach based on Hugin Expert API, and a version to the EM algorithm[5] based on the BC-EM method proposed by Peña et al. [PLL99] and the ECM approach [Sch97]. As we said above, for the Naïve Bayes approach we used the Hugin API[6] to build a Bayesian Network, and applied the learning algorithm implemented by Hugin that is a version of the EM algorithm named penalized EM. Our approach to the EM algorithm is not a penalized version of the EM implemented in Java. For further information about implementation check Appendix E.

After the implementation of all these clustering methods, the next steep was to integrate all them in our application. With the application built we could run the different techniques with each version of the database. We analyzed the results in detail in Chapters 5 and 6 and a comparison between them in Chapter 7.

Finally we drew some general conclusions after all the experiments, and do some proposals for further work related with the project.

[4] See Appendix A for further details about the K-Means algorithm and other clustering techniques.
[5] See Appendix B for further information about Bayesian Network and EM algorithm.
[6] See Appendix C for further details about Hugin Expert.

1.5. Thesis Organization

This chapter gives a brief overview on our project, the targets of our work, and our proposed approaches. The rest of this thesis is organized in four main sections:

Section I. DESCRIPTIVE APPROACH
- **Chapter 2.** Description and Data Pre-processing
- **Chapter 3.** First Approach Analysis
- **Chapter 4.** Description of the Application

Section II. ANALYTICAL APPROACH
- **Chapter 5.** K-Means Approach Analysis
- **Chapter 6.** Naïve Bayes Approach Analysis
- **Chapter 7.** Comparative Analysis of Different Approaches

Section III. CONCLUSIONS
- **Chapter 8.** General Conclusions and Further Work

Section IV. APPENDIXES
- **Appendix A.** Clustering Techniques
- **Appendix B.** Bayesian Networks
- **Appendix C.** Tools
- **Appendix D.** User Manual.
- **Appendix E.** Implementation and Source Code

In Section I we go through the data describing the original dataset, the transformations realized to work with it in other sections and the results obtained from Weka that describe the dataset in parameters of descriptive statistics as clusters size, mean/mode and standard deviation. We also describe the analysis tool we developed before going through the real analysis.

In Section II we analyze the data after applying some clustering techniques available in our application such as the K-Means algorithm and the Naïve Bayes network. We also analyze the differences of the clusters obtained by each technique and present the option for automatic comparison integrated in our tool.

In Section III we draw general conclusions from the complete project work and suggest several paths to follow the work thru. Some of these paths can be a seed for new projects and others are related to amplify the scope of the application.

Last Section groups the appendixes where further information related to several issues of the project can be found. We explain the theoretical concepts of the Machine Learning techniques we consider in the project, the tools we have worked with and implementation details as well as the source code of the main functions of the application. We also include the user manual of the application where we explain how to use it and the main functionalities of the tool.

2. Description and Data Pre-processing

2.1. Description of the Database

The initial database, which is available in its original format in the file flora_ch.xls, contains the data related to the flora of Switzerland divided in 565 areas that are explained in the Webflora's site at `http://www.wsl.ch/land/products/webflora/`. The original database is divided into twelve Excel sheets because the rows are too long to be kept in a unique sheet.

Each row in the database corresponds with one of the areas of the Swiss map, and for each of these rows different values of abundance for each plant are presented. These values come from discrete variables which can take integer values from 2 to 7. The total amount of areas represented in the database is 565, which means that not all the areas of the Swiss map are represented in the database. In fact, only the lowlands and mountain areas have been considered not taking into account the lake areas.

The 565 areas considered are numbered sequentially on the basis of its belonging to one of the 9 regions in which Switzerland has been divided into. We can see the regions painted on the map in Figure 2.2-1.Those regions are the following:

100's:	Jura Mountains
200's:	Central Plateau, western parts
300's:	Central Plateau, central parts
400's:	Central Plateau, eastern parts
500's:	Northern Pre-Alps, western parts
600's:	Northern Pre-Alps, eastern parts
700's:	Central Alps, Valais
800's:	Southern Alps, Tessin
900's:	Central Alps, the Grisons

2.2. Description of the attributes

In this section, we will describe the differences attributes present in the database and for each of them we will explain the values that it can take and the meaning of the corresponding values. We have the following information contained in the Excel file:

- Species names: A row that has a string containing the name of the plants that are registered in the database.

- Species number: This is a number associated to each specie name. This has no more information than the proper name; it is just a numerical identification of the plants. It is in range 1 to 21934.

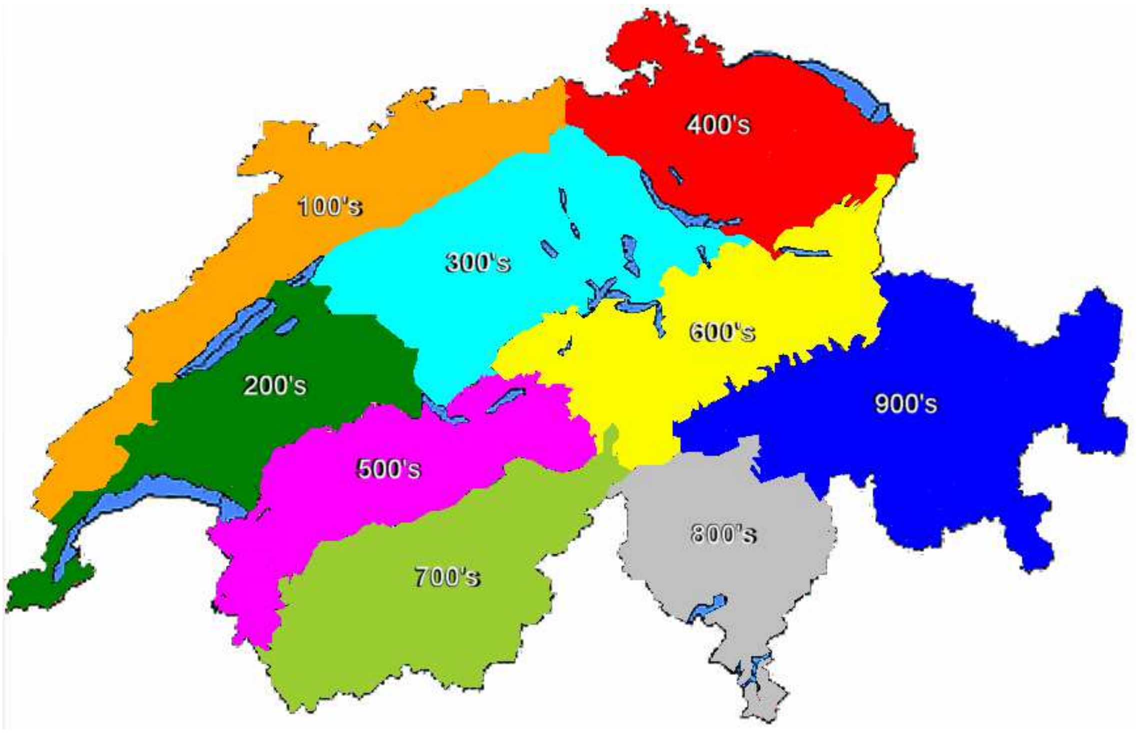


Figure 2.2-1: Map of Switzerland with the 9 regions represented in different colours

- Number ecological group: This is a value associated to each plant and it represents a discrete variable taking integer values in the range 0 to 8. Each value means a different ecological group:

0	No data
1	Forest species
2	Mountain species
3	Pioneers lowlands
4	Water species
5	Wetland species
6	Dry/partially dry grassland species
7	Ruderals and weeds
8	Fertilized meadow species

- Abundance: This is the value that represents the frequency of each species that have been found in an area. It represents a discrete integer variable in the range 2-7 meaning:

2: Rare[7]
3: Frequent (not rare)
4: Registered in 1984
5: Registered in 1985
6: Registered in 1998
7: Registered in 2000

[7] Found at one location, trees planted outside of their natural range.

As is described in the Webflora’s site, the original dataset had another two values, 0 and 1, but it also mentions that because the reduced data quality of these values they are not included here. These values were references to literature data (0) and herbarium specimen data (1).

- Number of the mapping area: This value represents, for each area, the number on the map related to the 9 regions in which Switzerland has been divided into. This value is in range 101-999 but there are only 565 different values.

- Timberline: This is a binary variable that can take two values, T or B, and is associated to each area. It means that the area is a mountain area, so the area is above the timberline (T) or it is a lowland area, which is below the timberline (B). In the original distribution atlas, also 28 lake mapping units were distinguished. Due to partly inconsistent data within the lake mapping units they have been omitted.

All this may be usable to extract different information from the dataset but, as we are interested in clustering the areas, data related to the ecological group and species number is useless.

The main information we are dealing with is the one related to the abundance of each plant into each area and the number of those areas on the map. The information relative to the timberline has been rejected because it only takes into account if the area belongs to a mountain zone or to the lowlands, and the clustering already finds these two main groups of areas.

2.3. Extraction of the data

The original dataset contains data from 2697 plants located in 565 areas from Switzerland tabulated as seen below in Table 2.3-1.

	Species name, Flora Helvetica	2697 names
	Nr ecological group	2697 values; range 0-8
	Species Nr, Flora Helvetica	2697 values; range 1-21934
Timberline	Nr. of the mapping area	abundance
565 values; T: below timberline B: above timberline	565 values; range 101-999	565x2697 values; range 2-7

Table 2.3-1: Tabulation of the data contained in the original Excel file

As seen on section 2.1 this dataset come on an Excel workbook with the dataset spread in 12 sheets. So, the first step consisted on extracting that dataset from the Excel container into a CSV format. To do that, first we saved each sheet in 12 ASCII format files and then replaced all tabs with commas. To do this the easiest way we used the option “replace all” from GVim editor.

The only values of the dataset we considered for the posterior analyses were the abundance values ranged 2-7 and, as we were going to use Weka and it needs a column with the class label, we also needed the 'Nr. of the mapping area' so we also saved that column in another ASCII file.

The next step was to transform the ASCII files into a Weka format and for that purpose we wrote the following Java class. This code generates a file "ddbb.arff" with all the values of the attributes for each area and the code of the area as last column. It supposes that 13 ASCII files, named 1 to 13, exist with each row of the 12 Excel sheets by files and the data separated by commas and, in the last file, just one column with the names of the areas.

```
import java.io.*;

class Procesa{

   public static void main(String args[]){
      String[][] table = new String[568][13];

      try{
         /* READ PHASE */
         for(int j=1; j<14; j++){
            BufferedReader in = new BufferedReader(
                  new FileReader("" + j));
            String line = in.readLine();
            for(int i=0; i<568; i++){
               table[i][j-1] = line;
               line = in.readLine();
            }
            in.close();
         }
         /* WRITE PHASE */
         PrintWriter out = new PrintWriter(
            new FileWriter("ddbb.arff"));
         for(int i=0; i<568; i++){
            for(int j=0; j<12; j++)
                out.print(table[i][j] + ",");
            out.print(table[i][12] + "\n");
         }
         out.close();
      }catch(Exception ex){
         System.out.println("IO Exception: " + ex);
      }
   }
}
```

The headers to complete the Weka's ARFF format are added manually from a table containing the structure show in Table 2.3-2 consisting on a column with the string "@attribute ", other with the name of the plant copied from the original Excel file and transposed, and the last one with the possible values 2 to 7 that the attribute can get. Afterwards, this table is saved into an ASCII file.

"@attribute "	Species name (transposed)	{2,3,4,5,6,7}

Table 2.3-2: Structure of the table used to generate the Weka's headers

For the cases of the class variable we also transposed the column from the original data file and replaced the tabs by commas.

2.4. Data transformations

After the first approach we realized that we can make some transformations of the original data to run different experiments and compare the results. In this chapter we explain the different dataset used in the next chapters, the transformations made and the meaning of the values that we use in the different dataset.

These transformations have been done because the original dataset has inconsistencies; the values 4, 5, 6 and 7, which refer to the presence of plants which have been registered in several years, do not contribute with clear information to the clustering problem. Also, there are a lot of blanks in the database that we must treat and decide which value is adequate to fill them.

2.4.1. General considerations

There are several factors that are going to be the same in all the transformations of the dataset that we are going to use. These are, for instance, the format of the data files, what we have done with the blanks, etc.

First of all, we are going to explain what we did with the blanks. Reading the instructions of the original data base we can see the meaning of each value; the data available have values referred to the abundance of each plant. These numeric values reflect the abundance of each plant in each area; with this values we suppose and assume that the blank mean that there is no plant of that specie in that area, so we use zero to replace all the blanks.

About the format of the dataset we chose, for the first approach, the Weka's format because it was the way to make that approach fast. For the rest of the experiments we chose one more general and easier to adapt to the different programs we were going to use format. We chose a simple format based in the character separated by comas, that is, plain text file in a CSV format. Based on this format we will adapt it to the appropriate files required for each method considered.

Finally, we have to say that we are going to use for the clustering just the abundance values rejecting the values related to the number of ecological group and timberline because the ecological number is something related with plants and we are going to cluster the areas and the timberline information only makes a difference among lowlands and mountain areas.

2.4.2. Original dataset

This dataset is extracted directly from the original database; the only transformation we made was to replace the blanks by zeroes. Other values of the abundance have been kept as in the original dataset.

The meaning of the values is based on the one seen in section 2.2:

Value	Meaning
0	No presence of the plant
2	Rare (found at one location, trees planted outside of their natural range)
3	Frequent (not rare)
4	Registered in 1984
5	Registered in 1985
6	Registered in 1998
7	Registered in 2000

Table 2.4-1: Definition of original dataset

2.4.3. *Transformed dataset*

To generate this dataset we made some changes to rescale the values of the original dataset so they could represent a measure of the abundance of the plants in each area. The objective for doing this is to codify the most proximal concept with near numerical values, in order to permit the clustering methods to measure the distance in a more correct way.

The scale of value chose is 0 to 3 maintaining the same meaning for values 0 and 3, recoding 2 as 1 and creating the value 2 as a new value that groups the values 4 to 7 from the original dataset as "registered data". It seems logical to code the values this way because the codifications 4 to 7 in the original dataset do not mention the abundance and we supposed that these presences can be considered as not rare because the plants have been found.

The definition of these values is presented on Table 2.4-2.

Value	Meaning
0	No presence of the plant
1	Rare (found at one location, trees planted outside of their natural range)
2	Registered in further years
3	Frequent (not rare or registered)

Table 2.4-2: Definition of the transformed dataset

We tried to order these values in a correct way considering first that the rare species are near to those that are not found. Then, we decided that the registered plants are rare but in a different way because a plant found in 1984 do not have to be rare. So we chose to put this value between the rare and the frequent value, which is the highest of the new scale.

2.4.4. *Binary dataset*

The binary dataset was made to be able to consider only the presence or absence of each species in each area. In this case we do not have information about the abundance of the specie and the possible values are only 0 and 1.

The definition of the values is shown in Table 2.4-3.

Value	Meaning
0	Absence of the plant
1	Presence of the plant

Table 2.4-3: Definition of binary dataset

With these two values we unify all the abundances in only one value. It seems interesting to us because we can see how this changes influence in the methods applied. This can reveal how relevant is the information of the abundance or, in the other hand, if it does not give too much information.

2.5. Conclusions

The dataset we are going to work with comes on an Excel file with the information split in several sheets because the number of columns exceeds the maximum permitted by the spreadsheet.

From all the possible attributes we can work with, we chose the abundance of plants by areas because the information we want to cluster is related to the areas in which Switzerland is divided.

We found that this information has a non informative format so we transformed the dataset to use the abundance as a distance measure among the instances. We considered using only the presence or absence information relative to each plant but this gives not much information about the distance between areas, so we recoded the original dataset into the following four cases:

Value	Meaning
0	No presence of the plant
1	Rare
2	Registered in further years
3	Frequent

3. First Approach Analysis

For the first approach we decided to run the K-Means method included in the Weka's API. We first tried to use the original dataset filling the blanks with the missing value considered by Weka; however, with this dataset the procedure returned bad results so we changed the dataset to make a further analysis.

3.1. Weka's K-Means first run

The first change we did to the dataset was to fill the blanks with question marks '?' that is the value that Weka recognizes as missing. This way, the range of attributes was 2-7 as we explained in Chapter 2 meaning the same as the original range.

Unfortunately, the results were bad as most of the instances were assigned to a unique cluster and most of them just contained 1 instance. The clusters generated for k=8 are shown in Table 3.1-1.

Cluster	Center	Size
0	102	482
1	101	1
2	311	1
3	103	1
4	638	1
5	929	1
6	345	77
7	702	1

Table 3.1-1: Weka results for the 8-means using the original dataset with missing values

Next step was to make the data numeric and with this dataset the results were a bit better but the method still generates clusters with just one instance or a few more. This can be seen in Table 3.1-2

Cluster	Center	Size
0	267	54
1	101	3
2	311	1
3	236	144
4	638	1
5	102	19
6	124	51
7	151	113

Table 3.1-2: Weka results for the 8-means using the numerical dataset with missing values

As these results are not accurate enough because the numeral values are not chosen the better way to use them as distance measures, we decided to fill the blanks with a value different than a missing value and still representative of the non existence of that plant. In the next section we go through the analysis using zero as the value we used to fill the blanks.

3.2. Filling the blanks

After the first run described above the next change we did to the dataset was to fill the blanks so we replaced de question mark '?' for ceros. This way, the range of attributes was 0-7 meaning the same as the original range but with the new value '0' that means 'no data'.

The last thing we needed was the attributed definition and our attributed were the plant species and the classes, and to do that we just transpose the species name row and the Nr. of the mapping area column.

For this dataset we applied the k-means procedure to generate 10, 12 and 15 clusters and the results for each run of the method were the ones shown in Table 3.2-1.

Cluster	*Center*	*Size*
0	104	52
1	101	44
2	148	82
3	507	68
4	238	84
5	566	92
6	565	49
7	501	25
8	111	42
9	801	27

(a)

Cluster	*Center*	*Size*
0	104	19
1	101	46
2	148	79
3	507	59
4	238	85
5	566	82
6	565	49
7	501	25
8	111	37
9	801	27
10	236	11
11	343	46

(b)

Cluster	*Center*	*Size*
0	104	19
1	101	40
2	222	69
3	626	49
4	632	2
5	566	76
6	565	51
7	504	16
8	111	45
9	801	27
10	532	1
11	443	51
12	501	20
13	236	21
14	232	78

(c)

Table 3.2-1: Results for k-means clustering with values 0-7 for k=10 (a), k=12 (b) and k=15 (c). The table shows for each cluster which area is the center of the cluster and how many areas are in that cluster

As we can see, making 10 or 12 clusters we have a good distribution of the areas because for the 15 clusters run two of them have few areas. Also we can see that most of the clusters have the same center (101, 104, 501, 801 …) and a similar size so we can conclude that these clusters are important.

3.3. Reducing the range of the data

The next change we did to the dataset was to reduce the range of values because values 2 and 3 are more significant than values 4 to 7 that only mean that there have been found plants in a concrete year after the first study. Now the range of attributes was 0-3 meaning the same as the original range but with the value '0' that means 'no data' and the new value '1' that means 'data registered between 1984 and 2000'.

We also applied the k-means procedure to make 10, 12 and 15 clusters getting the results in Table 3.3-1 for each run of the method.

The results are similar to the previous ones and we can get the same conclusions. Strong clusters are again centerd in areas such as 101, 104 and 801; and again the best choice could be to make 10 or 12 clusters but now the clusters are more unequal.

Cluster	Center	Size
0	104	52
1	101	47
2	148	83
3	507	68
4	238	90
5	566	92
6	565	46
7	504	19
8	111	41
9	801	27

(a)

Cluster	Center	Size
0	104	19
1	101	45
2	148	81
3	507	59
4	238	93
5	566	82
6	565	47
7	504	20
8	111	36
9	801	27
10	236	11
11	343	45

(b)

Cluster	Center	Size
0	104	19
1	101	39
2	148	78
3	626	49
4	632	2
5	566	76
6	565	51
7	504	16
8	111	37
9	801	27
10	532	1
11	443	51
12	501	20
13	236	21
14	232	78

(c)

Table 3.3-1: Results for k-means clustering with values 0-3 for k=10 (a), k=12 (b) and k=15 (c). The table shows for each cluster which area is the center of the cluster and how many areas are in that cluster

Comparing with the previous dataset we can see that one of the clusters change for both the 12 clusters and the 15 clusters runs. In the first case the change is not that significant because the center moves from 501 to 504 and the size of the cluster is just a bit lower, but on the second case we can see that the center changes much from 222 to 148 and making a bigger cluster. This could mean that for the first dataset the values 4-7 have enough relevance to change the size and the center of that cluster. And with the second dataset, as we don't have those values, we can't see those differences and the method just creates a bigger cluster.

3.4. Making the dataset binary

Our last change was to make the dataset binary. We transformed the values 2-7 into a new value '1' just for representing that there have been found that plant in a concrete area. The attributes now got binary (0-1) values meaning 'no data' for the '0' value and 'data registered' for the '1' value.

We also applied the k-means procedure to make 10, 12 and 15 clusters getting the results that are presented on Table 3.4-1 for each run of the method.

The results for the k-means procedure making 10, 12 and 15 clusters are again similar to the previous ones and we can get the same conclusions. Strong clusters are again centered in areas such as 101, 104 and 801 but now the method doesn't identify these centers as quick as with the other datasets.

Now the center and the size of the clusters change a bit more because we only consider the presence-absence criteria. This way the method doesn't take into account the abundance of each plant in each area so it just groups the registered or not registered value.

Cluster	Center	Size
0	104	48
1	101	37
2	148	82
3	443	74
4	238	92
5	566	93
6	565	47
7	504	18
8	103	47
9	801	27

(a)

Cluster	Center	Size
0	236	18
1	101	38
2	148	80
3	567	53
4	238	91
5	566	84
6	565	48
7	504	19
8	111	44
9	801	27
10	104	18
11	443	45

(b)

Cluster	Center	Size
0	343	27
1	101	38
2	148	83
3	626	48
4	346	23
5	566	76
6	565	49
7	508	18
8	111	35
9	801	27
10	104	18
11	506	39
12	614	2
13	236	10
14	232	72

(c)

Table 3.4-1: Results for k-means clustering with values 0-1 for k=10 (a), k=12 (b) and k=15 (c). The table shows for each cluster which area is the center of the cluster and how many areas are in that cluster

As a first approach we could see that the size of the clusters change as we use different datasets and different values of k. But we cannot say many things about the clusters because we cannot see which areas are included in each cluster, and where are they located in the map.

We cannot either see if the areas that compose each cluster are mountain areas or lowland areas, or which areas changes from one cluster to another in each of the runs. That is why we opted for the development of the tool to visualize this information on the Switzerland map and base on this software further analysis.

3.5. Conclusions

For this first approach we decided to use the application Weka that is known by us both and has most of the interesting Machine Intelligence methods available.

After the first run with the original dataset we found that the results are not usable because most of the clusters obtained had just one instance. We run the Simple K-Means method with the dataset composed by discrete values and, after that we considered numerical values. In both cases the blanks in the original dataset were treated as missing values and that's why the results were that bad, so we took the determination of treating those values as absences.

With this absence information we run the k-means method with several values of k and different datasets. These first transformations have nothing to do with the transformations explained in Chapter 2 and do not take into account any distance measure. These results were better than before but now the problem was in the impossibility of extracting more interpretations than the size and cluster center.

That’s why we opted for implementing an application able to represent the clustering results on a map of the land considered. This application is described in Chapter 4.

4. Description of the Application

In the previous chapter we have seen the results obtained with Weka have a difficult interpretation in the field where we are working. For this reason we decided to implement a software application able to run several clustering methods and show the results on a map of the land where the dataset comes from.

In this chapter we describe that program from a Software Engineering point of view including its relations with several Machine Intelligence libraries as the ones provided with Weka and Hugin.

4.1. Class Diagrams

The class diagram is one of the diagrams included in the UML standard and it shows the main classes that are going to interact in the program. We also show the packages where the classes inherit from. Figure 4.1-1 shows the simplified class diagram generated with Rational Rose™.

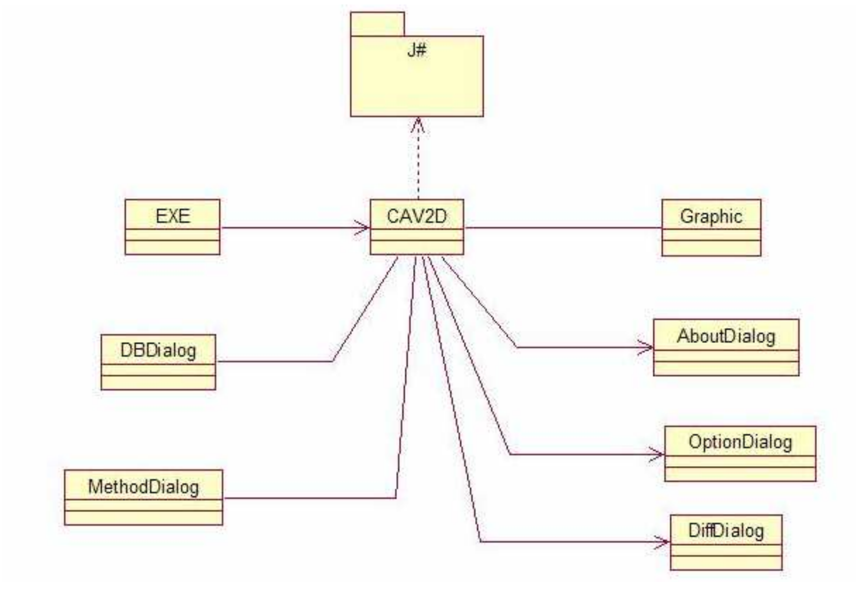


Figure 4.1-1: Simplified class diagram of the application

There is a launcher class, EXE which is responsible of starting the application. It creates an object from CAV2D, the main window, which is created with J# and uses several components of that API as menus, labels, buttons, and etcetera. Some of those components are other windows that act as dialogs. This is the case of DBDialog, used to select the dataset to load an the parameters of that dataset; AboutDialog, where some information about the program is shown; MethodDialog, to choose the clustering method and its parameters; OptionDialog[8], to show a general dialog that requires the

[8] This dialog is required because J# does not provide this kind of control.

user's interaction; and DiffDialog, where the user chooses the parameters for a clustering comparison.

Some of these dialogs are just accessed when needed, and others are fields in CAV2D. That's why the class diagram represents them in a different way. This diagram is simplified because all these dialogs also inherit and use the package J#.

Graphic is the component where the clusters will be drawn. It handles the paint event to draw the polygons that represents each area in a concrete color. For this purpose, it has a structure where the polygons are stored indexed by area name, number in this case, and those polygons come from a file where they are stored in as format described in the user manual[9].

The class diagram from Figure 4.1-1 only shows the classes involved in the user interface (UI) and does not reflex the clustering methods. These methods are implemented in Java and the difficulty to integrate them into the application made us take the determination of access to them via the command line. The class diagram for these classes is shown in Figure 4.1-2 and is very simple because they only access to the API of the tool from which they are based on.

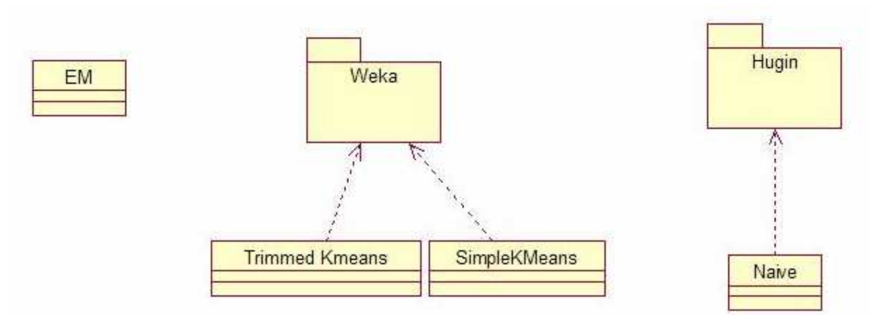


Figure 4.1-2: Class diagram of the clustering methods

The EM class is special because it is implemented completely isolated and, because of that, it does not have any connection but the obvious one to the Java™ API that we have omitted[10].

4.2. Sequence Diagrams

The sequence diagram shows the steps taken in the interaction among the classes and the user of an application. We show the general sequence for applying a generic clustering method and showing its results on the map. As we need to load a dataset, we also show the general load sequence for any of the files that the program works with. A concrete interesting interaction is the change of the map and points that is also described in sequence diagram.

[9] The user manual is located in Appendix D.
[10] For implementation details check Appendix E.

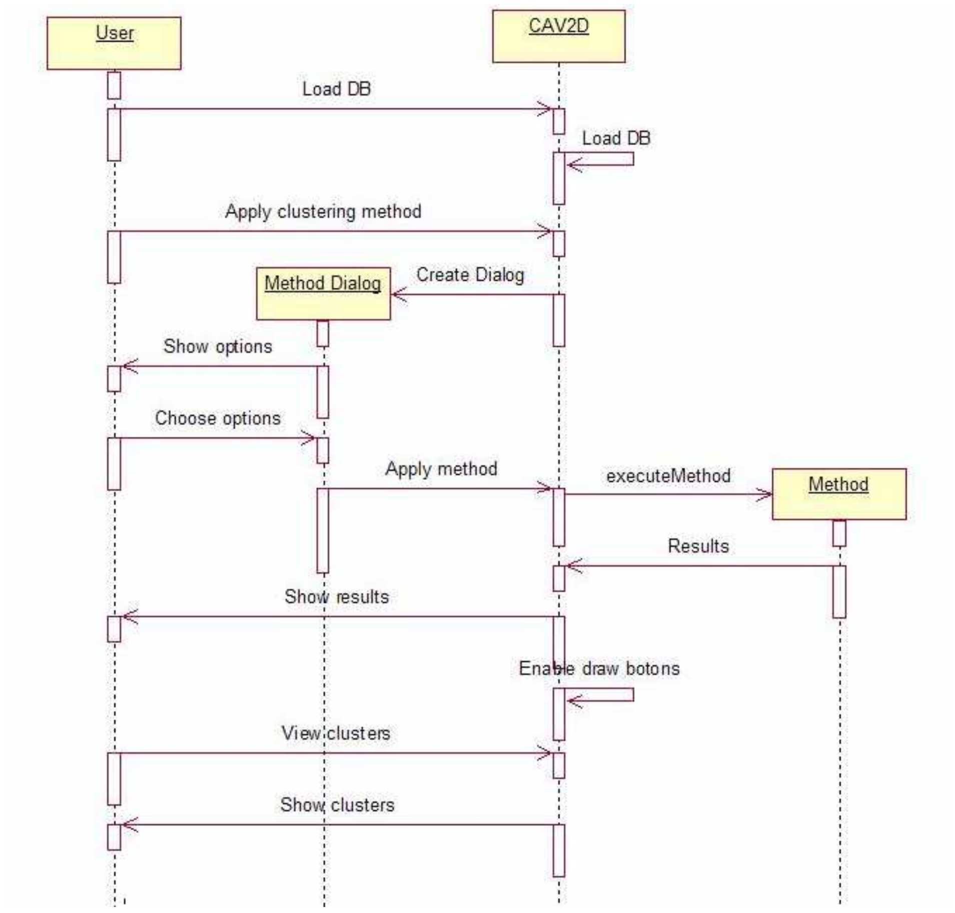


Figure 4.2-1: Sequence diagram for the complete process of clustering

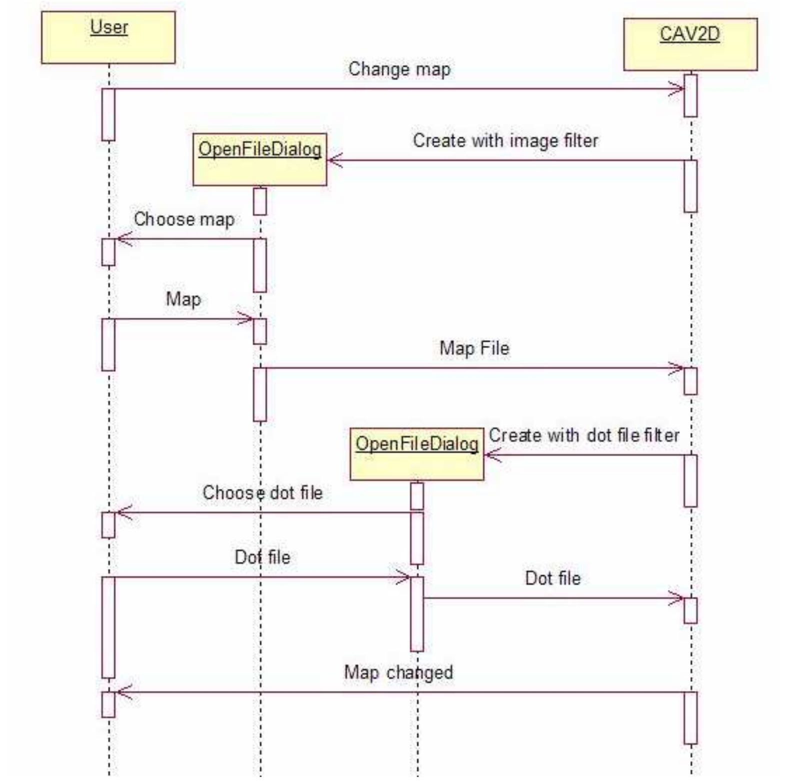


Figure 4.2-2: Sequence diagram for the change map process

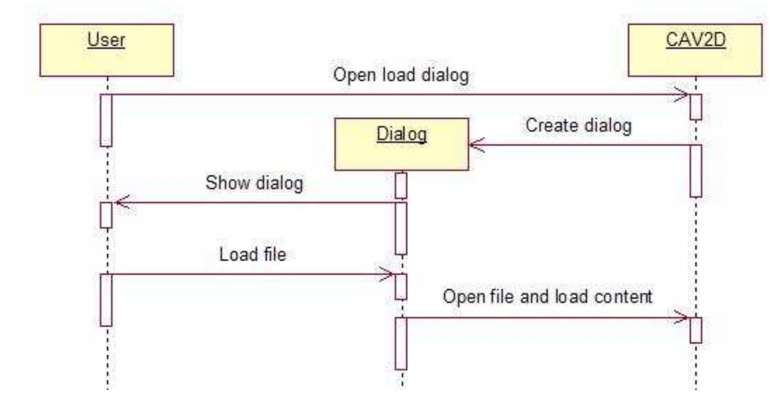


Figure 4.2-3: Sequence diagram for the generic load process

4.3. Files Formats

The application uses different file formats for different purposes. It supports several dataset formats but imposes a concrete one for the results of the clustering.

4.3.1. Dataset File Formats

The dataset can come on several formats because the load dialog permits to specify parameters such as data separator and presence of class variable and attributes names. The following files are supported:

- Data files (*.dat)
- Comma separated values files (*.csv)
- Generic ASCII files (*.txt)
- Tabbed files (*.tab)

Also a generic flat file can be loaded if it fulfils the only rule of having as separator one of the three permitted: comma, tabulator and space.

The file can also contain, and it must be checked, a first line with the names of the attributes and a last value with the class variable useful as areas identification. It was contemplated the possibility of loading an office file (Access or Excel file) because the original dataset comes on an Excel format, but it could not be implemented because of lack of time. This is one of the proposals as further work in chapter 8.

4.3.2. Clustering Results File Formats

The results of the clustering analysis come in two formats. One of them is a text based format that is the one presented as results of the clustering analysis. This information can be saved as rich text in a rich text format file (.rtf). This option is accessible through both, the file menu and the toolbar.

The other results file is automatically generated and it contains the clustering assignments using as identifier of the instances the class variable without mattering if it comes on the dataset or it has been automatically generated[11]. It is a flat ASCII file that contains, in a CSV format, the instances assigned to each of the clusters found. These files are called clustering assignments files (.ca) and their format and extension are required for further uses of these results as reload of the assignments to view them on the map, or compare two clustering assignments automatically.

The CA files generated will be located in the same directory where the dataset is and they will have the same name of the dataset followed by the method run to create them.

4.3.3. Map Formats

The last kinds of files the program considers are the formats related to the map where the clustering is showed on. The change of the map requires two kinds of files, an image with the new map and a file with the points that define each of the polygons representing the areas on the map.

The image file can come on one of the standard image file formats where as the file with the polygons must have a concrete structure explained below. The supported image files are:

- Windows Bitmap (*.bmp)
- Jpeg File (*.jpg; *.jpeg)
- Compuserver (*.gif)
- Portable Network Graphics (*.png)

For the polygons file we use, we talk about the dot file format (.dot) but the program also accepts .csv and .txt files. The format for all them is concretely defined and if the file does not follow it the change of the map will not have success.

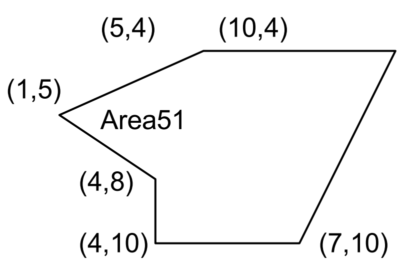
The first line of the dot file must contain the amount of areas considered. This is for the initialization of some structures that we are using to store and draw the areas. The following lines must have the following information knowing that the point (0, 0) of an image is the top left corner:

$$\texttt{<area id>},X_1,Y_1,X_2,Y_2,\ldots,X_k,Y_k$$

The first value always will be the identification of the area and it must be followed, in a CSV format, by the pairs of point that draw the polygon that covers the area. An example of this format can be seen in Figure 4.3-1.

In the example show in Figure 4.3-1 we begin from the western point (1, 5) and go around the area opposite to the clockwise direction to (4, 8) and so on until we reach again (1, 5) that must be included again to close the polygon.

[11] The automatically generated class column contains the string "Inst" followed by the number of instance.



```
"Area51,1,5,4,8,4,10,7,10,10,4,5,4,1,5"
```

Figure 4.3-1: Example of an area transformed into a dot file format

To generate the dot file for the Switzerland map we used the program Vextractor 3.10 demo that permits to "create polylines" on an image that will define a polygon and save them as a vector file in Autocad, ArcView, Windows MetaFile, and other formats. From those other formats is very interesting for us the ASCII xyz file format. This format saves the polylines as plain text with a line for each point and it has the possibility to include an index for the points belonging to the same polyline and a Z-coordinate that for us is useless.

Creating the polylines in the order the areas appear in the Webflora's site, saving them as "map.xyz" and having the identifications loaded in another file "areas.txt" as CVS we only needed to run the following Java code to obtain the dot file:

```
import java.io.*;

class ProcessDots{
   public static void main(String args[]){
      StringBuffer output = new StringBuffer();
      try{
         String areas[] = new BufferedReader(
         new FileReader("areas.txt")).readLine().split(",");
         /* READ PHASE */
         BufferedReader in = new BufferedReader(
         new FileReader("map.xyz"));
    String readed = in.readLine();
    String [] line = readed.split(",");
    //The polyline id, without Z-coordinate, is the 3rd value
    String prev = line[2];
    int i=0;
    //add the number of areas and the first area id
    output.append(areas.length+"\n"+areas[i]);
         while(readed != null){
       line = readed.split(",");
            //if the polyline id is new add break and the next area
       if(!line[2].equals(prev)){
          output.append("\n"+areas[++i]);
          prev = line[2];
       }
            //the values are 200 times smaller and are related to
            //the bottom left corner. 588 is the height of the map
       double X = (Float.parseFloat(line[0])*200);
       double Y = 588.0 - (Float.parseFloat(line[1])*200);
            output.append(","+X+","+Y);
       readed = in.readLine();
         }
         in.close();
```

```
            /* WRITE PHASE */
            PrintWriter out = new PrintWriter(
                      new FileWriter("swiss.dot"));
            out.print(output);
            out.close();
        }catch(IOException ex){ ex.printStackTrace(); }
    }
}
```

4.4. Main Functionalities

The main purpose of this program is to have a tool where some clustering method could be run and where the results could be shown on a map, so these are the main functionalities of it.

The application can load datasets in the formats explained in section 4.3.1 and show them into a data grid with the attributes as columns and one row per instance. This loaded dataset can be saved in a CSV format keeping the class variable if it exist in the original dataset but adding a row with the attributes names it the do not appear on that original dataset.

The application supports 4 clustering methods:

- Simple K-Means Approach[12]
- Trimmed K-Means Approach[12]
- Naïve Bayes Network[13]
- Expectation-Maximization Algorithm

Each of these methods can be applied only once a dataset has been loaded and they will generate automatically a file, or a set of files[14], with the name of the method run. All they also show text based results that are shown once the method finish and that can be saved as .rtf files or printed. To show the graphical clustering results an option is activated when the method finishes rightly.

The application can always load CA files and show the clustering in them but not the text-based results because, as .rtf files, they can be red with any of the multiple text processing applications existing.

Finally, the program allows comparing two clustering assignment files automatically searching for the equivalent clusters on both files. This search can be based on any of the two files and can follows two different comparison criteria. These two criteria are explained in chapter 7.

[12] Theoretical explanations of the k-means methods can be found in Appendix A. Details about the implantation are described in Appendix E.
[13] Theoretical explanations about the Bayesian Networks and the EM algorithm are located in Appendix B. Also, details about implementation are found in Appendix E.
[14] The Naïve Bayes method generates three files when it is run with two root nodes. One file per root node and other with the crossed clustering results.

4.5. Conclusions

This chapter gives a brief description of the application tool we have developed from the viewpoint of the Software Engineering including the main functionalities and the restrictions to the files supported by the program.

Other details about the GUI or how to work with it are explained in the user manual located in Appendix D.

5. K-Means Approach Analysis

In this chapter we analyze the results of applying the Simple K-Means algorithm on the datasets created from the original database of Swiss flora. Also, we analyze and compare the Trimmed K-Means variation with the Simple K-Means approach.

5.1. Introduction

For the analysis of k-means method we used the algorithm implemented in Weka API, the SimpleKMeans class. We modified the code for our necessities in the project basically to show the cluster assignments (cluster membership) that the method generates. The original algorithm only shows the number of items that each cluster has and the centers of the clusters, but knowing the cluster membership was necessary for our project in order to draw the areas of each cluster on the map.

In this chapter we will analyze several experiments running the k-mean algorithm on each of the datasets we created. In the experiments, we change the number of clusters and compare the results. We are going to use the binary dataset and the transformed dataset in the experiments, since these two datasets contain enough information, and the original dataset has many possible values which are difficult to be interpreted.

5.2. Binary dataset

We run several experiments using this dataset with different numbers of clusters. The experiments start with small values of the k and end with bigger values. With this dataset the algorithm has no information about the distance because it only represents the presence/absence data (0/1). We started with k=5, continued with k=8, and ended with k=12.

For k=5 the results are shown in Figure 5.2-1. In that figure we can see that there are four significant clusters. There is a big one, cluster 3 in green, covering the north part of the map and some areas in the south, which can be interpreted as a snowball effect because the areas to the south clustered into this one are far from the northern areas and seem to have a low relation with them. We can see the same phenomena in other clustering results.

Other three important clusters are in the center-south of Swiss. These three clusters reveal vegetation differences along the south of Swiss interlaced each other. This is due to the lowlands and mountain areas located in the south of Swiss that gives the clustering that root shape. This part of the map is conformed by several mountains and valleys or depressions caused by some rivers that flow through the mountains. This configuration makes the flora different from one area to another and similar to the mountain areas and areas along the rivers.

Cluster 4, the light blue one, covers the mountain areas to the south and center of Swiss. The other two clusters, numbers 2 and 5, show differences between the lowland areas to the south and in the center of the land. Cluster 2 also shows the snowball effect containing areas in the north-west.

The last cluster 1, the one in yellow, has a smaller size than the others. As it clusters several zones on the map, it may be representing areas in which there are strange plants that are similar around the map. This can also happen because these are mountain areas and they have similar special conditions. However, it is different enough from the configuration of the southern mountain areas because they are clustered separately in cluster 5.

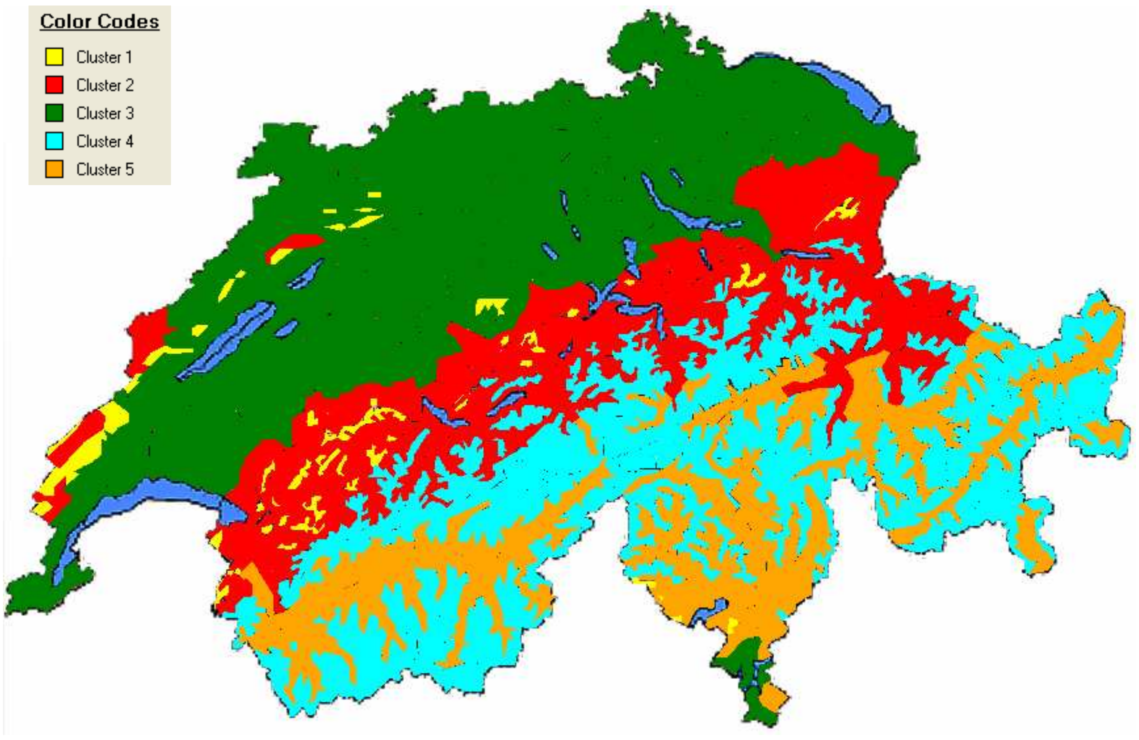


Figure 5.2-1: Clustering of Simple K-Means method with k=5 on the binary dataset

Figure 5.2-2 shows the clustering for k=8. Now we can see how, in the north-west zone, cluster 1 still stands so it must be an important cluster with significative differences respect to the other clusters. Also, the cluster in the middle, now number 5, the orange cluster. The big cluster at the north of the map has been split into another two distinct clusters, numbers 2 and 3. This reveals a lower level clustering of this big area that shows less important differences between the floras of the north of Swiss. This also happens with the light blue cluster, number 4, which now is divided into two sub-clusters that again represents more concrete differences among the flora of the lowland areas of the south of Swiss.

Again we can see a division in cluster 5, the orange one, which now has been divided into two clusters, numbers 7 and 8. The areas in these two clusters are lowland areas surrounded by mountains; however, we cannot infer anything from the position or the other adjacent clusters because both clusters touch with the two mountain areas regions and also both are spread along the south of the country. One possible explanation for this division is the existence of two main depressions, one to the west into cluster 8 and the other to the east into cluster 7. These two depressions can have vegetation differences that create these two clusters, and the other areas just are assigned to the closest cluster.

In a hierarchical approach we can generate the dendrogram shown in Figure 5.2-4 to see graphically the divisions of the clusters of higher level into a lower level clustering and the ones that are coming on the next analysis with k=12. Not all the clusters are represented because some of them are not the result of the division of a previous cluster.

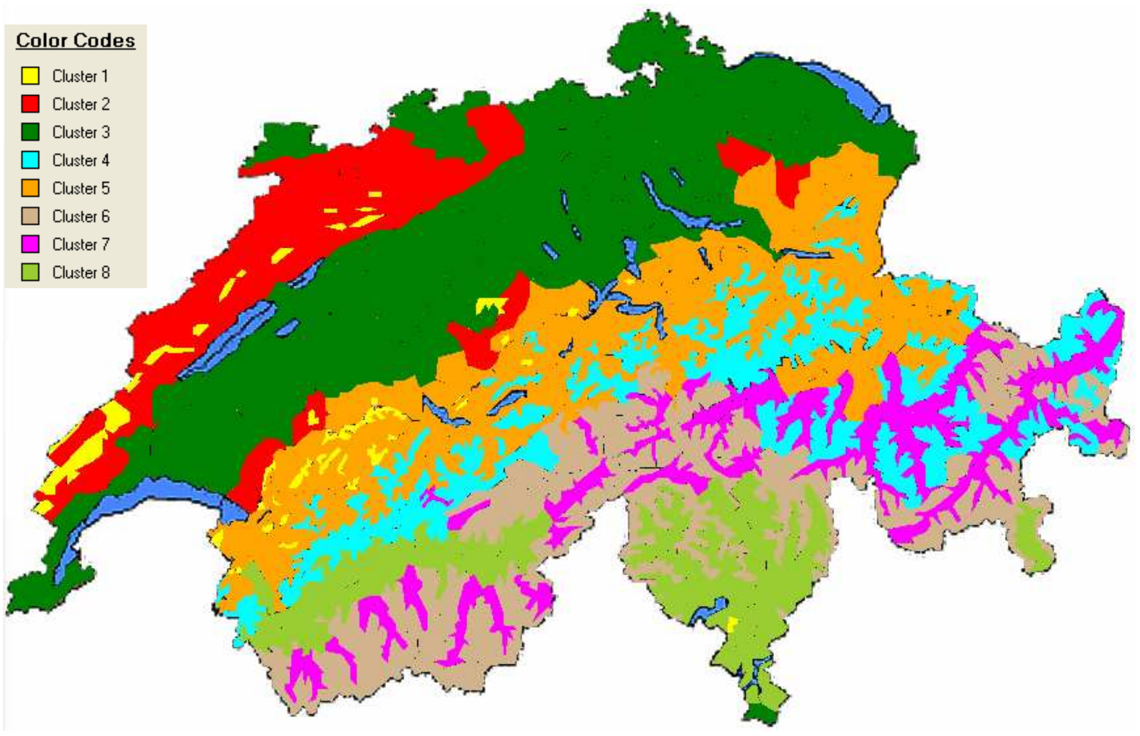


Figure 5.2-2: Clustering of Simple K-Means method with k=8 on the binary dataset

The last experiment for this dataset was done with k=12. The results are shown in Figure 1.2-3. Again we can see how the north zone, cluster 3, is split into two clusters representing different groups of areas. Now, the cluster that previously was in yellow is in dark red but it still represents more or less the same group of areas that have a very specific flora configuration. From the Webflora's site we can infer this is because these areas are mountain areas in the middle of a huge extension of lowlands. It is not strange to think that in these areas there are plants that are endemic from this concrete kind of areas.

In the middle and south of the map are there more changes. Here we can see many clusters mixed. Cluster 4 remains practically unchanged as cluster 6 does. These two clusters are very compact clusters because, even with a high value of k as 12, they remain practically unchanged but they come from a unique cluster when k=5 so this means that these two clusters are very close each other.

Two new clusters have appeared taking areas from several other clusters of higher level. These clusters are numbers 1 and 12, and they appear as small clusters probably just for the need of get 12 different groups. Cluster 1 is clearly this kind of cluster but cluster 12 seems to have more reasons to be. This cluster groups areas along the center of the map and all these areas are mountain areas so maybe it represents another group of mountain areas that are particularly different from the one northern and southern the land. The differences are not that obvious because with a lower k these areas were clustered with the northern or southern mountain areas. This reveals that this group of areas could be some kind of mixture of both edge clusters.

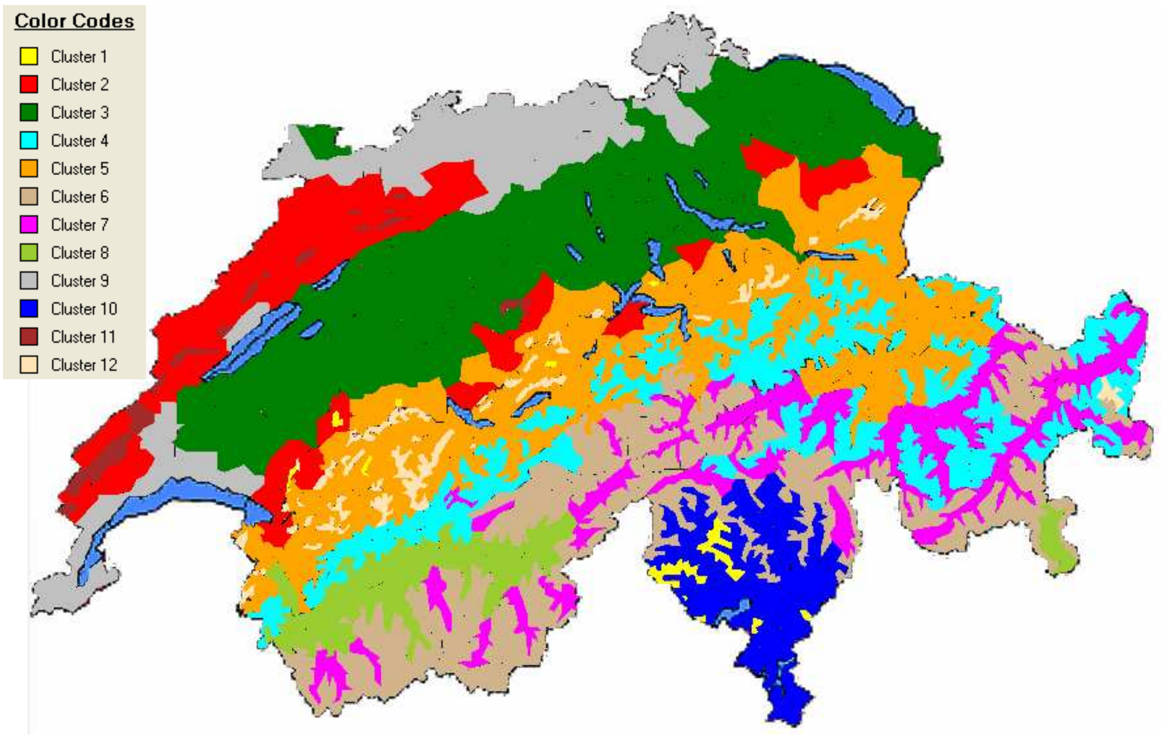


Figure 5.2-3: Clustering of Simple K-Means method with k=12 on the binary dataset

Other areas have been clustered together according to more detailed similarities between their flora configurations. Clusters 2 and 5, for instance, have changed in some areas according to the new centers found. Cluster 7 has also changed a bit referring to the previous clustering for the same reason.

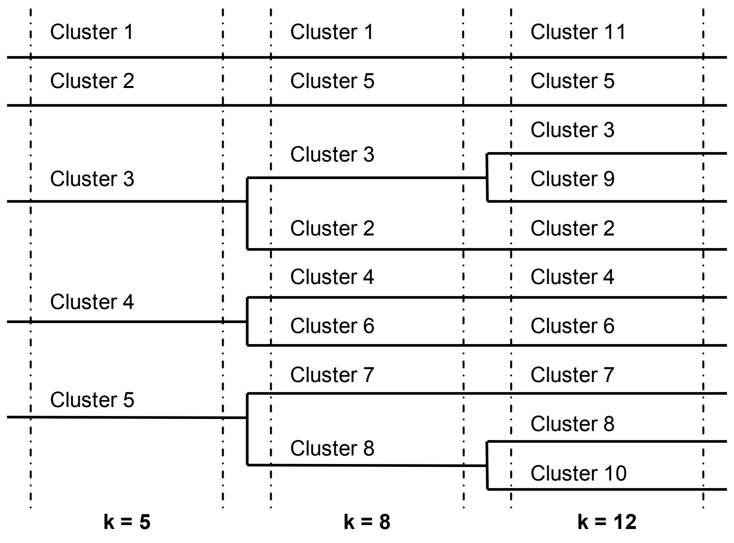


Figure 5.2-4: Dendrogram[15] representation of the Simple K-Means runs with the binary dataset

[15] See Appendix A for further information about the dendrogram.

Cluster 8, the light green cluster, now has been divided into two new clusters, numbers 8 and 10, as it happened before when this cluster appeared with k=8. Taking into account the splitting we have found as we increase the value of k, it is logical to group the results in a hierarchical structure of the clustering.

This hierarchical dendrogram can be seen in Figure 5.2-4. The Figure shows two clusters that are maintained along the different runs changing only the name given on each of these runs; hence, these clusters are very compact. Other two very compact clusters are numbers 4 and 6 that come from cluster 4 when k=5. These clusters are very compact but also they are very close because for k=5 they form a unique cluster.

In the rest of the clusters, there are two compact clusters, numbers 2 and 7, that are grouped with other clusters of less relevance with k=5. We say that other clusters are less relevant because, at a lower level, they present enough difference to make the method group them in separate clusters. We can see a bi-dimensional representation of the dendrogram of Figure 5.2-4 in Figure 5.2-5.

This figure represents the same information as Figure 5.2-4 but in a Venn's diagram format. In this format it is easy to see that there are 5 main clusters, and three of them can be split into two each other. We see how clusters 1 and 2 are never split; cluster 4 is just split once when we look for a greater number of clusters whereas the other clusters are split again to generate the 10 smaller clusters that are represented.

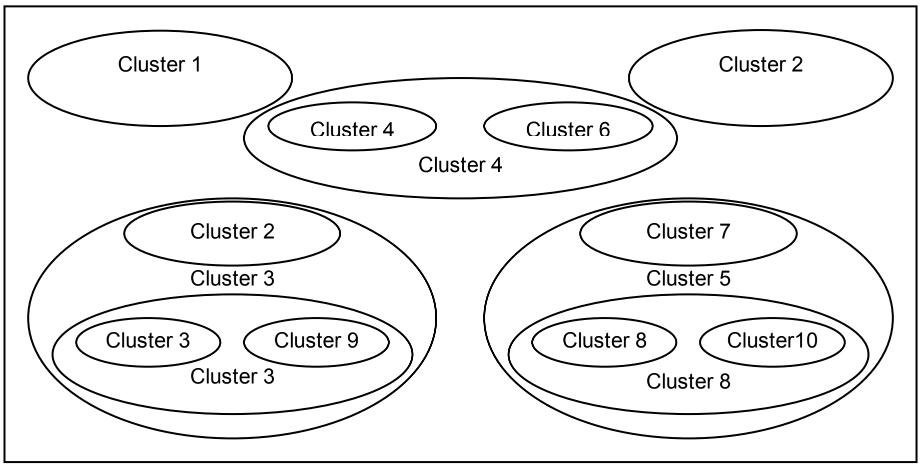


Figure 5.2-5: A 2D graphical view of the hierarchical clustering of the original dataset

As we have seen along these experimental results, the binary dataset is good to find general differences among the data. The k-means method groups the areas taking into account which plants appear in each area and then grouping the areas with similar kinds of plants in it. The method is useful to find big main clusters but it has not enough information with this dataset to go further into these main clusters.

In the data we have found 2 important groups (clusters 1 and 2 in the 5-means experiment), one of them not so big but also important because it represents a set of plants that appears only in these few areas that are assigned to cluster 1. These two main clusters reveal the two main different groups of plants and appear with any k you give to the method.

The next pair of important groups of plants appears with a higher value of k. The plants found into them are similar so they are grouped together with a low k. These clusters are numbers 4 and 6, and they represent the two main differences among the flora of the southern mountain areas so it is logical that these areas are grouped together at a high level of clustering.

Other two important clusters that also appear with k=8 are numbers 2 and 7 which represent respectively a set of plants that only occur in the north west part of Swiss and an special vegetation related to the lowlands surrounded by mountains. The rest of the clusters have less importance because they appear at a low level of clustering, and at this level, the clusters are formed with very concrete differences just for the need of getting the specified number of clusters.

5.3. Transformed dataset

Using the transformed dataset we followed the same experimental structure that the one we used with the previous dataset. Now the k-means algorithm can calculate and use as distance measure the possible values of each attribute. These values have been selected appropriately in order to represent the distances in an appropriate way.

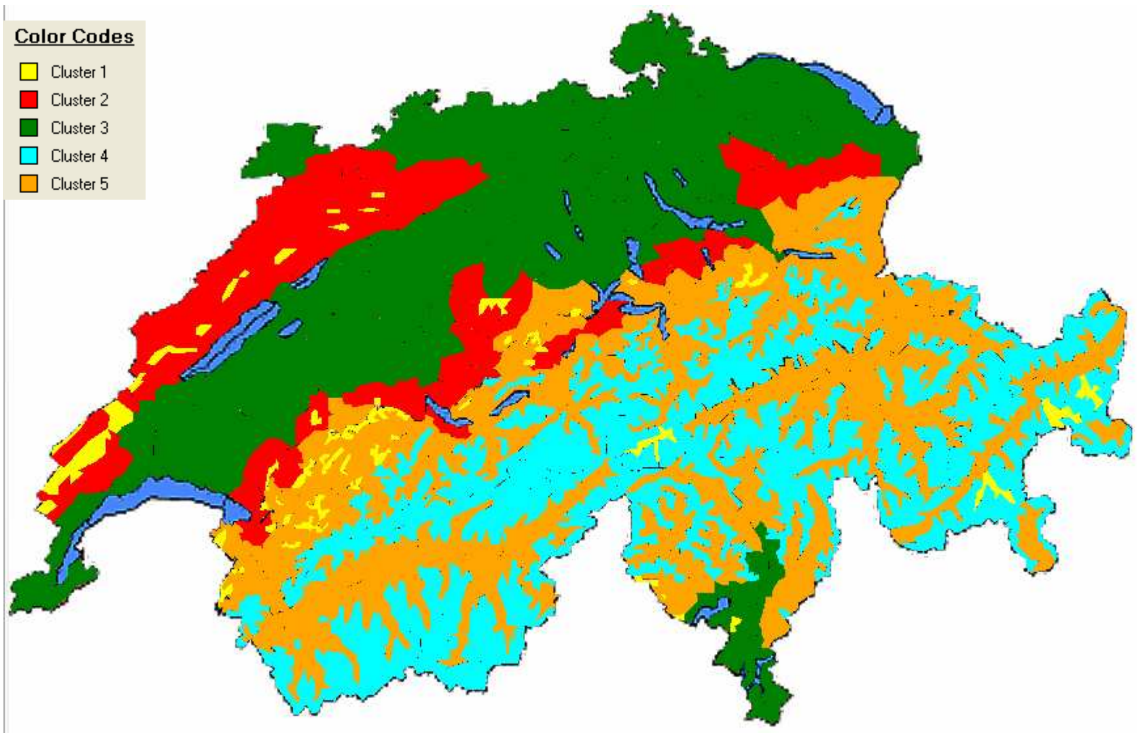


Figure 5.3-1: Clustering of Simple K-Means method with k=5 on the transformed dataset

The 5-means clustering is shown in Figure 5.3-1. These results are quite different from the ones using the binary dataset: the big cluster on the north of the map has been divided into two smaller clusters, the red and green ones, seen as an interesting thing the fact that the red cluster includes areas from both the north and the south of the green cluster. This means that the green cluster has a main flora that in the edges has changed by contact with other species. Also we can see some areas in the south of the map grouped with cluster 3, which supports the theory of a general cluster.

Now, the lowland and mountain areas to the south are distinguished more clearly and both have an important extension. Cluster 5 now represents all lowland areas from the center and south of the land so we can infer that there are enough differences introducing cluster 2. It is no necessary to divide this cluster into two as it happened with the binary dataset.

The mountain areas to the north are still grouped together with several mountain areas across the land. This can be interpreted as an amount of mountain areas different from the main mountain area, cluster 4, that show more similarities with the northern mountain areas group. These areas can have some endemic plants in common that are not in cluster 4.

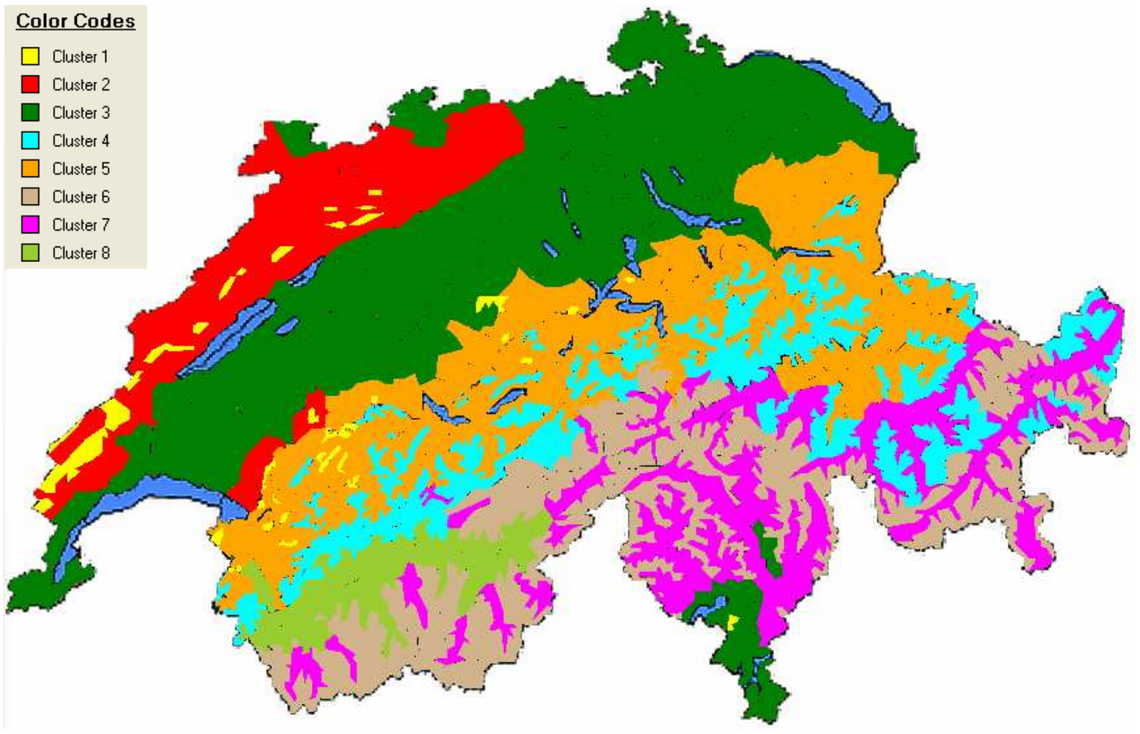


Figure 5.3-2: Clustering of Simple K-Means method with k=8 on the transformed dataset

The clustering results from k=8 run in Figure 5.3-2 have less difference with the binary dataset analysis. With this dataset cluster 5 is almost the same cluster; however, it is a bit bigger because some areas allocated on clusters 2 and 7 have been taken into cluster 5. Compared with the 5-means clustering the big orange cluster has been separated into 3 new clusters, the orange (cluster 5), the purple (cluster 7), and the light green (cluster 8) ones. This division is different from the one we got in the binary dataset because now the southern lowland areas are clustered into cluster 7, and only some areas to the west conforms cluster 8.

Cluster 8 has been reduced and areas that were assigned to that cluster now belong to cluster 7. This happens because, with the binary dataset, this cluster grouped only those lowland areas where particular plants were found not considering the amount of plants. Now cluster 8 represents a concrete region with concrete specific flora and a concrete amount of some plants moreover than the simple presence-absence of each of them. This supports again the theory of a main depression that has a concrete flora. In fact, this cluster now represents the particular flora along the depression caused by the river that flows between the south western mountains and, as we can see in the figure, these particularities are different from the ones that define the other depression to the south west of the land.

Again cluster 4 has been divided into clusters 4 and 6 so the two clusters are still close. As clusters 4 and 6 keep more or less the same configuration than the one they have with the binary dataset, their similarities are just related to the presence or absence of characteristic plants; but currently, the light blue cluster has taken more areas from the center of the map so there are more similarities than the ones we found both with the binary dataset and with k=5 and the current dataset.

The main difference of this map happens on clusters 2 and 3. Here we have found that the areas in the middle of the land that for the 5-means were clustered into cluster 2 are now closer to cluster 5. This means that cluster 2 was taking areas from both clusters 3 and 5, probably because of a chain effect that makes these areas closer to the closest area from cluster 2 than to the closest areas of the other two clusters.

With this clustering we still can see areas from the south of Swiss clustered together with areas from the north. This could mean that they are so different from their neighborhood that they are grouped with a big cluster, the one closer but not that similar, just because of the fact that they must be clustered somewhere. Other interpretation could be that in fact these areas are similar to the ones from cluster 3 but some of these areas in the 5-means run now have been grouped with cluster 7 so this argument has not enough weight.

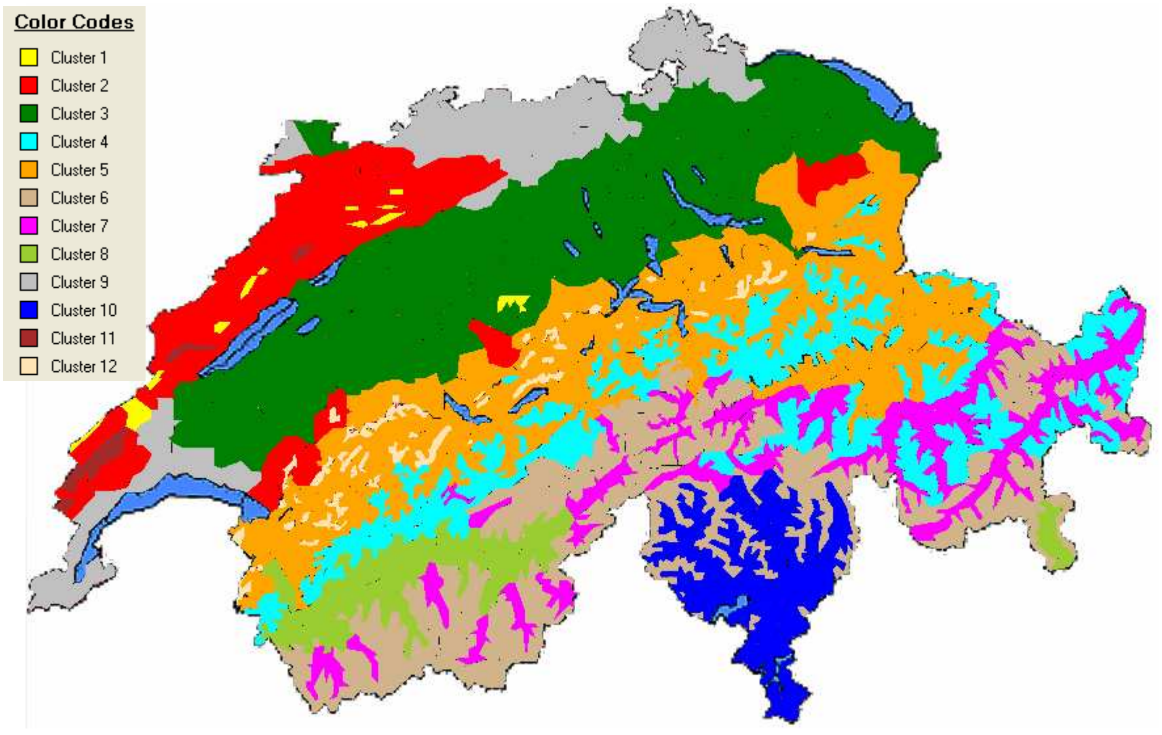


Figure 5.3-3: Clustering of Simple K-Means method with k=12 on the transformed dataset

For the 12-means clustering we obtained the results we can see in Figure 5.3-3. The results are again a bit different respect the clustering made with the binary dataset. One of the differences is at clusters 2, 3 and 9 (red, green and grey clusters). In the clustering made with the binary dataset the areas of some edges of each of these three clusters were assigned in a different way having cluster 2 more areas than now.

This happens because with the binary dataset only the presence or absence of each plant in these areas fulfilled the flora configuration to be clustered as they were; but now, with the amount information, the method also considers the weight of that presence. With this criterion of amount of presence, those areas in the edge of the clusters are closer to the new cluster they are assigned to. This happens because, in these areas, there must be a mixture of plants from both bordering clusters and taking into account only the existing plants the assignment is different than taking into account also the amount of that presence.

Some areas belonging to cluster 9 now have been assigned to cluster 2 and areas in the edge between clusters 3 and 5 that were assigned to cluster 2 now have been re-clustered into one of these bordering clusters. But this is not strange comparing with the 8-means results because the orange cluster is almost the same. Just the north edge of this cluster changes from one cluster to other, which can be interpreted as the existence of a mixture of species that, in some cases is closer to one cluster and in other cases is closer to other cluster.

Now cluster 7 is the one where cluster 10 comes from but this only means that this new cluster will appear at this level of detail. Again, those areas belonging to cluster 3 at the south of the map are clustered into this new region shows the reason why they were assigned to cluster 3 was just because this was the closest one.

Other clusters are almost the same except the yellow one that now has been divided into two, clusters 1 and 11. This means that, looking for differences, these areas are farther than other mountain or lowland areas. Knowing this now we can say that with the transformed dataset we can find more differences in the northern mountain areas and there is no need to search those differences in the southern ones as it happened with the binary dataset. Also we can say that the central mountain areas have an own flora different from the other mountain areas.

Finally, the changes for the lowland areas are located at the edges of these areas probably because of the mixture of flora configurations.

5.4. Trimmed K-Means analysis

The Trimmed K-Means approach is a specific version of the k-means algorithm that leaves out of the clustering an amount of instances to make clusters with less variance. See Appendix A for further details.

We applied the method using only the transformed dataset because this method needs some kind of distance measure to trim the instances with a trimming level $\alpha=0.05$ that is a standard statistical significance level. This means that the method will leave a

5% of the instances out of the clustering. For our dataset with 565 instances these are 28 instances unclustered.

Figure 5.4-1 shows the clustering results for the trimmed 5-means with α=0.05; this is known as 0.05 trimmed 5-means. As we can see in the figure, the trimmed clustering is almost equal to the one without trimming except the two areas to the north that without trimming are in cluster 3 and now have been assigned to cluster 2, and another one to the west previously in cluster 5 and now also in cluster 2.

The main importance of this technique is that it generates spherical clusters taking into account for the new centers computation only those instances significantly close to one of the centers. With this restriction the method is not affected by outlier instances and it reaches better centers than the simple k-means approach. Also, this method shows those instances that are not assigned to any cluster, called the *trimmed instances* or *outlier instances*. These trimmed areas are not colored in the figures we are going to show and they only have the color of the original map.

Often, the method allows to differentiate between *outlying clusters* and *radial outliers*. The first kind of outliers is constituted by groups of outliers that differ from the main clusters found but without enough “strength” to be considered as one of those main clusters. The radial outliers are isolated instances that do not constitute groups [GGM03].

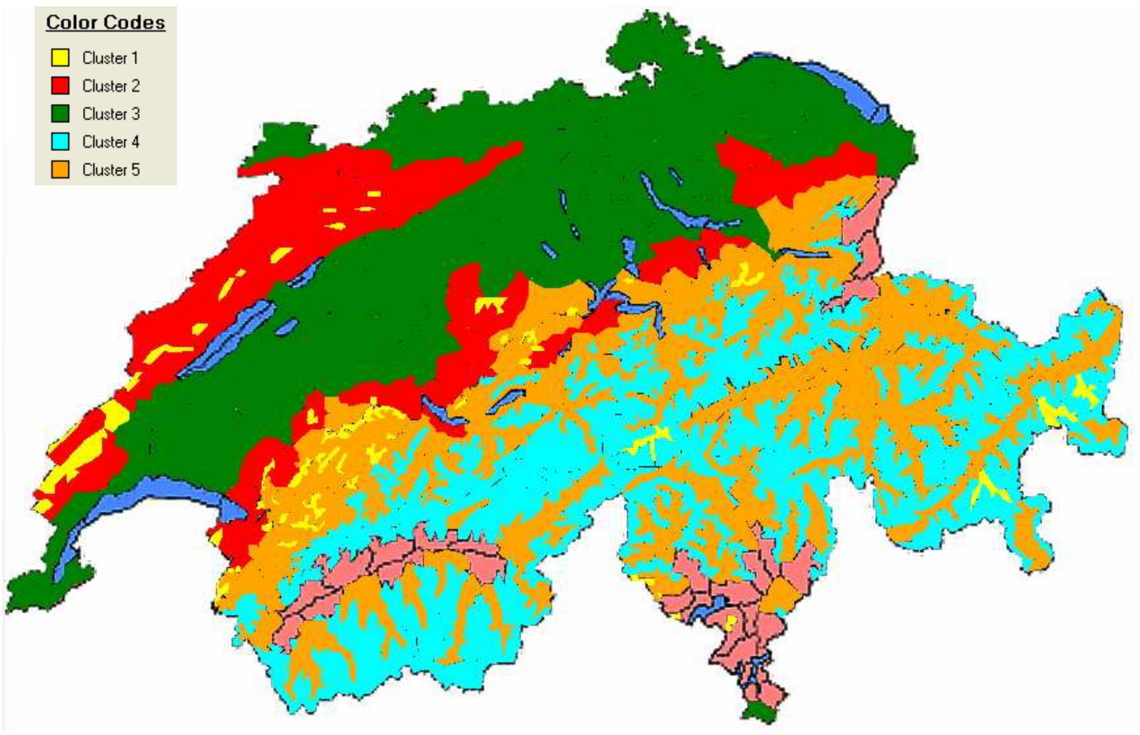


Figure 5.4-1: Clustering of 0.05 Trimmed K-Means method with k=5 on the transformed dataset

For the 0.05 trimmed 5-means those trimmed instances are the three without a color association to the east, 9 to the south west on the depression area we have talked about previously and 16 to the south that are the ones assigned to the big cluster to the north with the simple k-means approach. These two groups of areas to the south are candidates to be outlying clusters. This means that probably they represent two "hidden" clusters that have not been formed because they did not have enough strength against the bigger clusters, and probably they will appear with a higher value of k.

Now we can infer from these results that these trimmed areas are far from all centers. As we suggested before, now we can see that effectively there is an important weigh on the areas along the depression to the west and, as we can say now, it is true that those areas cause a deviation on the k-means centers because they are far from their closest clusters. In this case, that cluster was number 5 and, for this approach, it only caused one misclassification on the clustering. The same can be said about the western trimmed areas and some on the south that also were assigned to cluster 5.

Most of the areas that have been trimmed are from 800's zone because in these areas the flora is very different from all the other flora configurations represented by the clusters formed. These areas are the ones to the center-south of the map and, as many of the most distant are from this zone it seems logical that they cause a higher effect on the center's deviation. For these southern areas assigned to cluster 3 we can see that they were causing a misclassification of two areas because of their effect in the center of this cluster.

These trimmed areas have all a high distance to their closest center and with a higher trimming level we can see that the new trimmed areas are around these ones. We can see the 0.15 trimmed 5-means results in Figure 5.4-2 supporting this argument. This means that, in fact, these areas form one or several outlying clusters and there should be some "hidden" clusters around some of these trimmed areas. And this is not that strange after analyzing the simple k-means results.

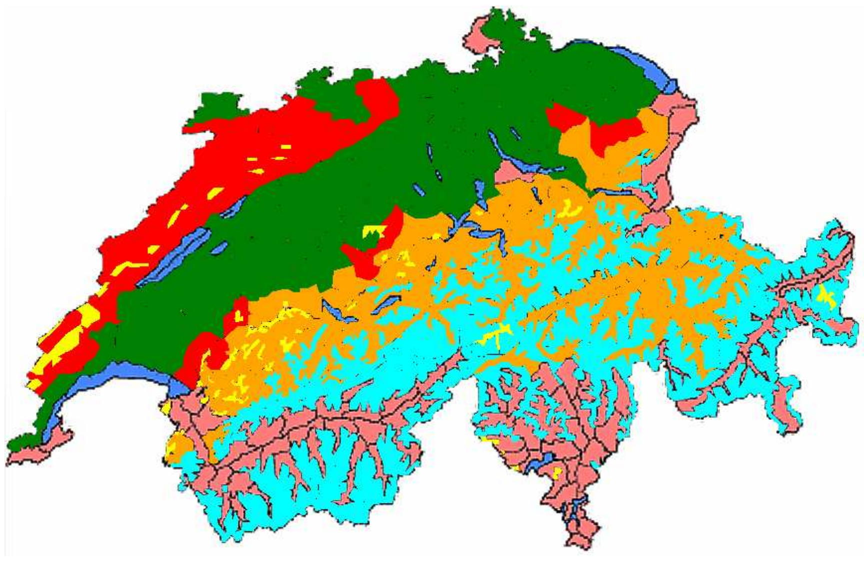


Figure 5.4-2: Clustering of 0.15 Trimmed K-Means method with k=5 on the transformed dataset

This new image, Figure 5.4-2, also supports the argument of an important weigh or variance on the areas along the two depressions, the one to the south west of the land and the other to the south east, because a new group of trimmed areas have appeared along the eastern depression.

We must take a look at the fact that all the trimmed areas are lowland areas. This means that the mountain areas have a similar concrete flora maybe because of the temperature and other climatic conditions related to the mountain. Also we can infer that the lowland areas have a more diverse flora and that is why the clusters vary that much from one clustering to other.

Figure 5.4-3 shows the trimmed clustering for k=8. Comparing with the trimmed 5-means we can see that now the method has found the cluster for the western untrimmed areas, cluster 8, and have grouped them into this cluster. Still there are areas to the south that are outliers and, as they haven't found a cluster to belong close enough to be considered, which means that they have a very specific flora very different from the one found in other areas. Other trimmed areas have less significance because they spread all over the map, and only represent the amount that must be trimmed.

Comparing with the simple k-means results we can see that the clustered areas are the same apart from the ones trimmed. This means there is no important deviation of the centers caused by the trimmed areas because if we go further into the trimming we could see that are many trimmed areas that belonged to cluster 3 but areas clustered into this cluster are the same than with the untrimmed approach.

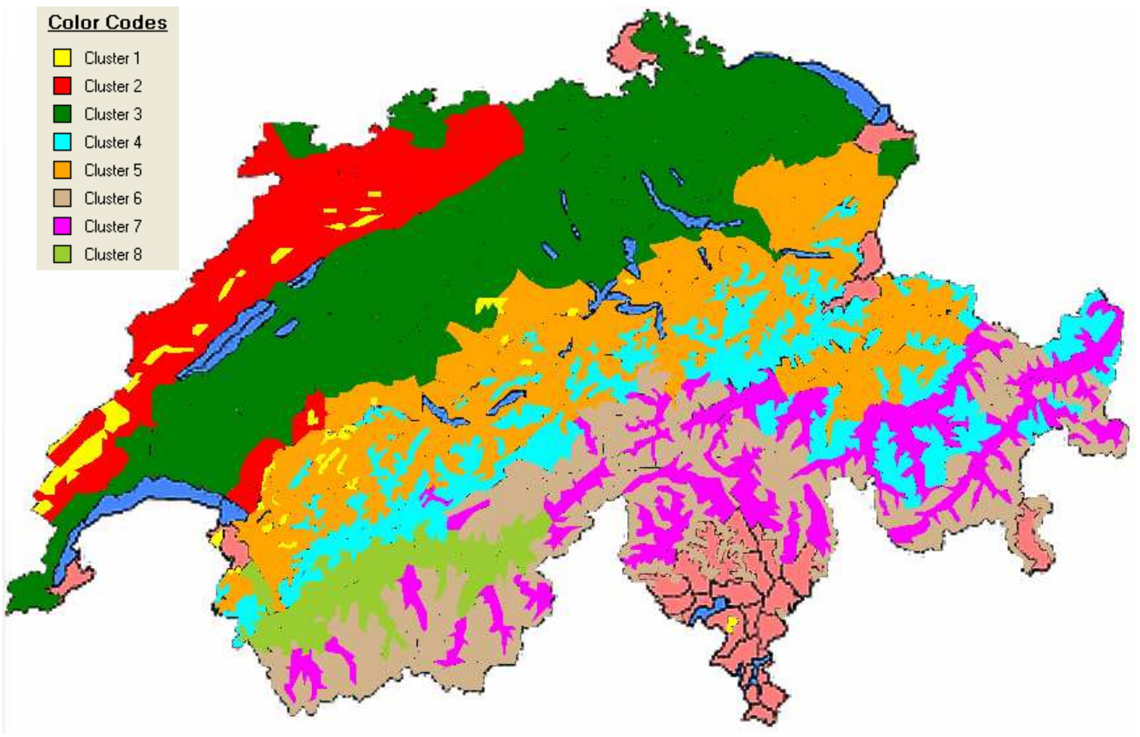


Figure 5.4-3: Clustering of 0.05 Trimmed K-Means method with k=8 on the transformed dataset

Figure 5.4-4 shows the results for the 0.05 trimmed 12-means experiment. As we can see, now the southern trimmed areas have also been clustered together as we predicted before so these areas were, actually, an outlying cluster. Other areas have been trimmed because of their distance to the closest cluster but these new trims seem to be just radial outliers. Looking at the map we can imagine that this happens because the method trims always a concrete number of instances.

Compared with the simple clustering there are more evidences of the centers' deviation as some of the areas assigned to cluster 2 before now have been corrected into clusters 5 and 9. There is an area previously assigned to cluster 3 that now also belongs to cluster 9. This means that cluster 2 has less significance than the one we inferred in the simple k-means analysis because it is affected by the outliers that were assigned to both clusters 5 and 9.

We can also see an estrange effect of the trimming in one area assigned to cluster 8 to the east of the map. This area must be close to both clusters 5 and 8 but, because of the trimming and/or the presence of cluster 8, it has been now assigned to number 8 instead of cluster 5 that seems more logical. This happens because of the need to trim an amount of instances, which is an undesired effect of the method.

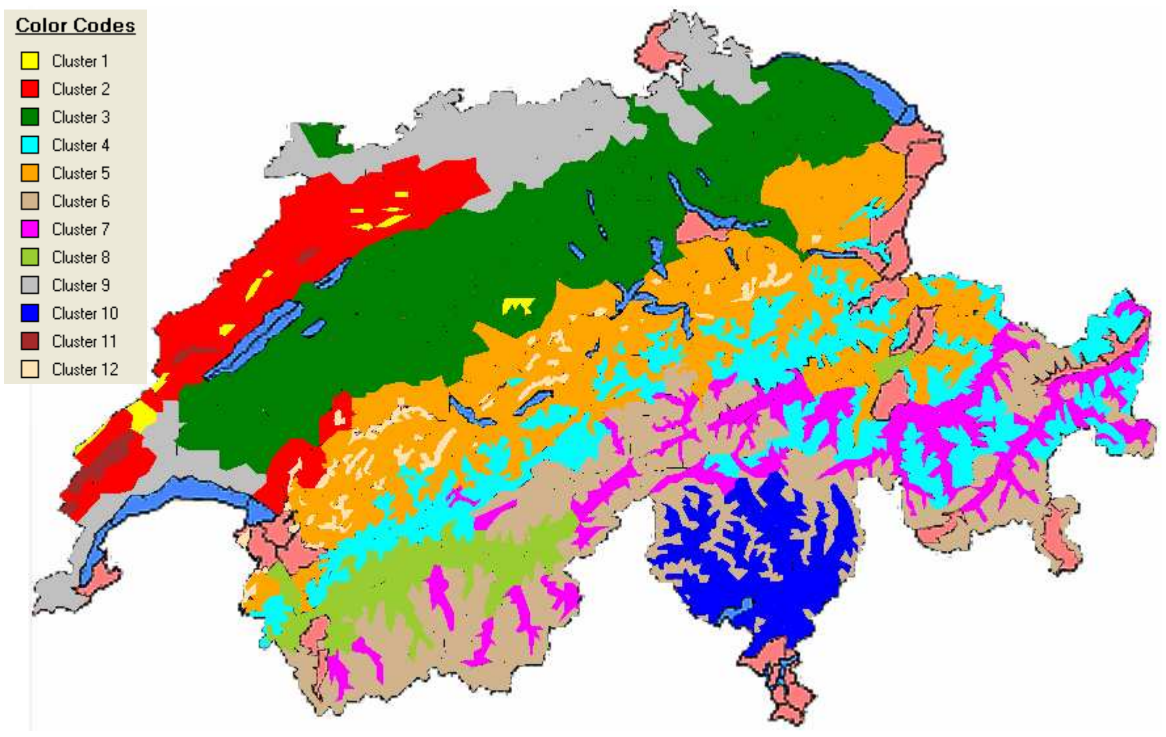


Figure 5.4-4: Clustering of 0.05 Trimmed K-Means method with k=12 on the transformed dataset

To solve the problem of miss-clustering doubt to an excess of trimming and find an optimal value for k and α, García Escudero et al. created the k-variogram to represent the trimmed k-variance functions defined in [GGM03] against α for a concrete value of k. It also uses the second derivate of the trimmed k-variance functions that shows peaks and exponential decay easier to interpret.

When the function shows a smooth decay the value of k is good or high depending on the graphic for the previous k; and when there are peaks on the function they must be interpreted as the presence of a small cluster accounting a percentage of the data given by the value of α at that peak. Unfortunately, by the time this document has been written there is no computational implementation to make an automatic selection of the optimal values for k and α since we did not have time to approach the problem.

Another possible approximation is to calculate the distance between the trimmed instances and the one with the higher separation to its closer cluster. If the instance is closer to this outlier than the half distance to its closer cluster it can also be interpreted as outlier, otherwise the instance can be untrimmed. This also can be said for a concrete trimmed instance *i* as: "*if the ratio of the distance between the farther outlier and i* *divided by the distance from i to its closer cluster is greater than 0.5, i can be considered as non-outlier*".

The algorithm for this alternative can check, for each *i* in the trimmed instances, the following condition:

$$if\left(\frac{d(farthest\{trimmed\}, trimmed_i)}{d(trimmed_i, closerCluster_i)} > \frac{1}{2}\right) \Rightarrow untrim(trimmed_i)$$

5.5. Conclusions

After all these experiments we can see how the k-means algorithm finds clusters splitting others when we increase the value of k. This supports a good way to see the most important clusters that we can find in the data. Also this method lets us find compact clusters that remain almost unchanged as we increase the numbers of centers and the clusters that are close because of the split effect we mentioned above.

Another conclusion we can obtain is that the datasets we used produce similar results, which means the main information in the data is the presence or absence of plants in an area. However, this information is not enough when we make a high or low number of clusters as we could see the results from k=5 and k=12, because there is not enough information to find 12 good clusters, and neither to find important similarities in the data to generate 5 clusters.

We found that a good number for clustering is around 8-10 because with a lower k the method generates big clusters that are the union of two more compact but close each other, and with a higher k it expends the time searching for concrete differences that let it have this number of centers.

As we could see, the trimmed k-means approach helps to find better centers removing the outlier instances, and with this removal we can find sometimes the presence of another cluster for which we will need another center. This method as it is can also cause undesired effects because of the trimming of important instances for a cluster that can make another instance belonging to this cluster be misclassified.

A solution for this problem is to check the distance variation of the outlier instances looking for an important jump that represents the change from outliers to important instances. This can be done manually with the k-variogram to detect a good value for k and α. A computational approach is to check whether the distance between an instance and the farthest one is less than the half of itself; if this happens the instance can be considered an outlier; otherwise, can be considered to calculate the new centers.

6. Naïve Bayes Approach Analysis

This chapter is dedicated to the Naïve Bayes approach for all of the three datasets we created. We used the Hugin Expert 6.4 API to build a Naïve Bayes model, and to apply the learning process that uses the EM algorithm. In this chapter will be commented the results obtained with different Naïve Bayes models and with different initial number of classes to cluster in.

6.1. Introduction

As we said before, we used Hugin 6.4 API for the construction of the model. We implemented a Bayesian Network with a Naïve Bayes structure; it consists of one root node connected to the rest of the nodes acting as leaves. There is no connection between the leaves but through the root node. This was the first model we built; after that, a second root node was added with the proposal of comparing both roots clustering results and to cross them to generate a new partition of the dataset to study. The reason for use two roots nodes is because the EM algorithm is based in the initial probability distribution of the root node, so if we use two root nodes we can compare the results of a running of EM with different initial probabilities and observe if they are similar or not, and in case that they were not similar we can obtain the join clusters of the two root nodes.

For example, for the Swiss flora's dataset the program creates a root node for the cluster assignments initialized with the number of cluster that we want to generate, and one leave node for each one of the plant that we have in our database. For the leave nodes, the parent is the cluster node and the number of states of these nodes depends on the different values that the dataset has. Each node will have as many states as the amount of possible values that the plant can get.

With this structure created, the learning will be made through the EM algorithm implemented in Hugin API 6.4. The EM algorithm requires to have missing values on the cluster label attribute of each instance and nowhere else because the purpose of this analysis is to make the network learn the best clusters for the data.

6.2. Building model

The first thing that we have to do for constructing the model is to define the structure of the network, which is the Naïve Bayes structure. We implemented this structure using the Hugin API, which support a set of classes useful for that purpose.

The steps that we took in this way were:

1) Create a Domain object. This is the first step to create a Bayesian network in Hugin. Every item in a network is associated with one domain that is the

container of the network, and establishes the closed environment where the network is going to work.

2) Create all the nodes. In this step we create the nodes that are necessary for the Naïve Bayes network, that is the two root nodes and all the leaves associated with each plant. For our network we used the class "`NumberedDiscrete-ChanceNode`" that represents a node with discrete values but all of them are numerical. The name of each of these nodes will come from the first row that should contain the name of the attributes.
3) Define the number of state of each node. For the root nodes it is defined by the number of clusters wanted, and for the leave nodes this comes from the different values of each attribute on the dataset.
4) Connect the nodes. In this step are defined the relationships between the nodes. In our case all the leaf nodes associated with the plants have as parents the two root nodes and there is no link between the brothers neither between the roots.
5) Create experience tables. Each node has a conditional probabilities table (CPT) where the node keeps the probabilities for each of its states. It is necessary for the posterior application of the EM algorithm.
6) Init experience tables. For the root nodes we initialized the tables randomly because the default uniform distribution gave bad results in the learning phase. The leave nodes are initialized with the default distribution.
7) Load dataset. Loads the file with the cases to apply the method with.
8) Compile domain. Before applying the learning algorithm the domain must be compiled to 'lock' it and make it ready for the execution.
9) Apply the learning algorithm: Now is when the network learns the structure of the dataset. It updates the conditional probabilities tables using the cases loaded into domain. For this purpose, the API uses an implementation of the EM algorithm.

In Figure 6.2-1 and Figure 6.2-2 are shown the structure of the two models constructed for applying the EM algorithm. In the first one, Figure 6.2-1, we can see all the plants represented as leave nodes and all of these leave nodes as children of the Cluster node. In Figure 6.2-2 we have a similar structure, but now there are two root nodes, Cluster1 and Cluster2. With this structure we will possibly get two different clustering results using the same dataset.

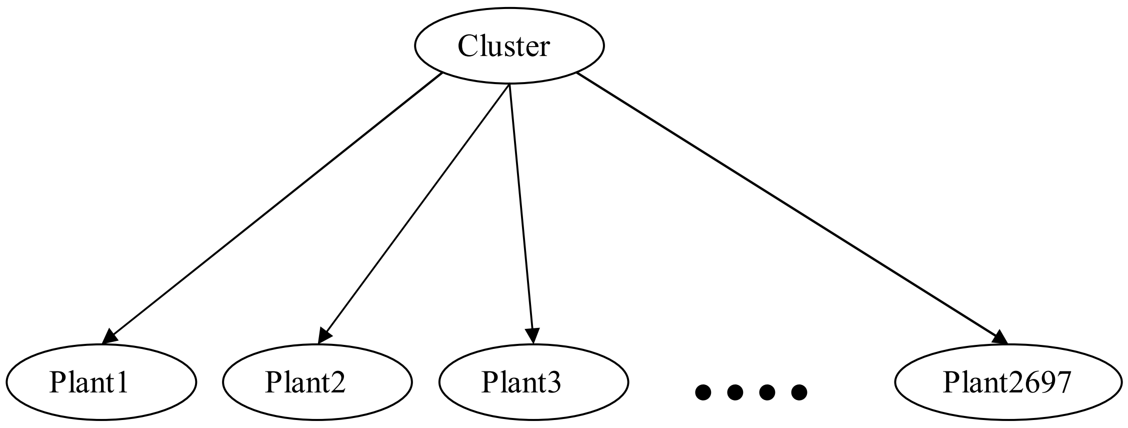


Figure 6.2-1: Naïve Bayes model with one root node

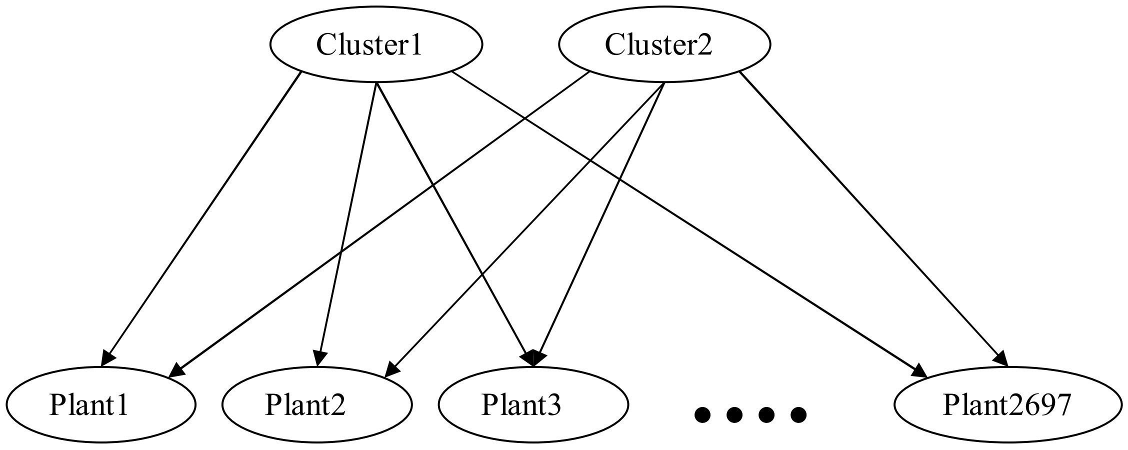


Figure 6.2-2: Naïve Bayes model with two root nodes

Figure 6.2-3 shows a screenshot of the Hugin GUI with the Network created for the case of one root node.

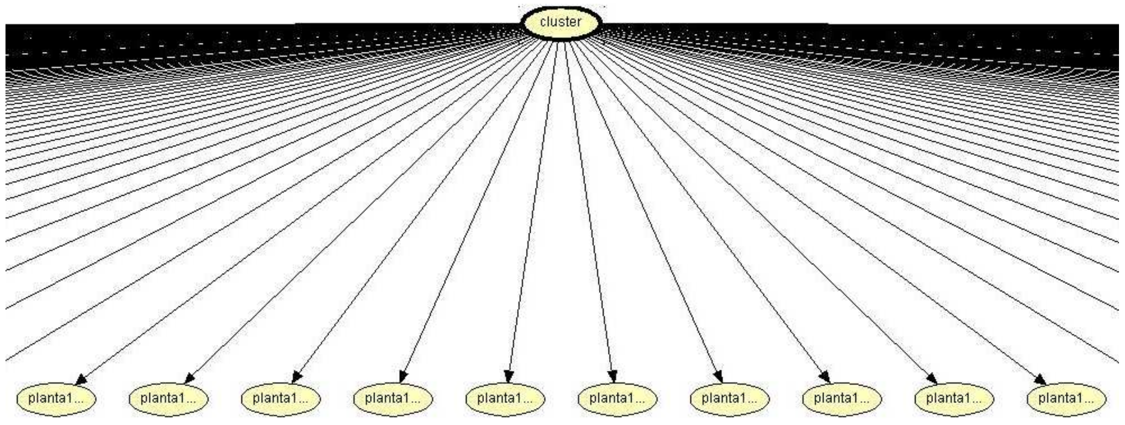


Figure 6.2-3: Screenshot of the Hugin GUI showing the model with one root node

After all these steps, the conditional probabilities tables (CPT) of the nodes have the probabilities belonging to each cluster. An example of this table is shown below. This table was obtained from a leave node; we can see that for an observed value of the attribute of 0 the probability of cluster 0 is 1 and the probabilities of other clusters are shown on the same row. These are conditional probabilities of each state at the root node given each state of a leave node.

Cluster	0	1	2	3	4
0	1.0	0.978261	0.974684	0.966102	0.764706
1	0.0	0.0217391	0.0126582	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0126582	0.0338983	0.235294

Table 6.2-1: Example of CPT of a leave node with 4 states and a root node with 5 states

With all the probabilities tables calculated, the next step is to enter the evidence of each cluster and propagate it to see in which cluster of both root nodes the probability is higher and assign the instance to it. That was made following the next steeps:

1) Enter the evidences of one case in the case files, all the leaves nodes has an activated state.

2) Propagate evidences; this is through a Hugin API function.
3) Search through all states in the root nodes which have the higher belief.

That was the process followed for the different runs that we made. This process is common to all the tests we made and the changes are related to the dataset that we used and the number of root nodes and states of each of them.

6.3. Original dataset

We begin the experiments with the original dataset. We run the method with two root nodes; one is with three states and the other with four states. The results of the clustering using the Naïve Bayes (NB) model are shown in Figure 1.3-1.

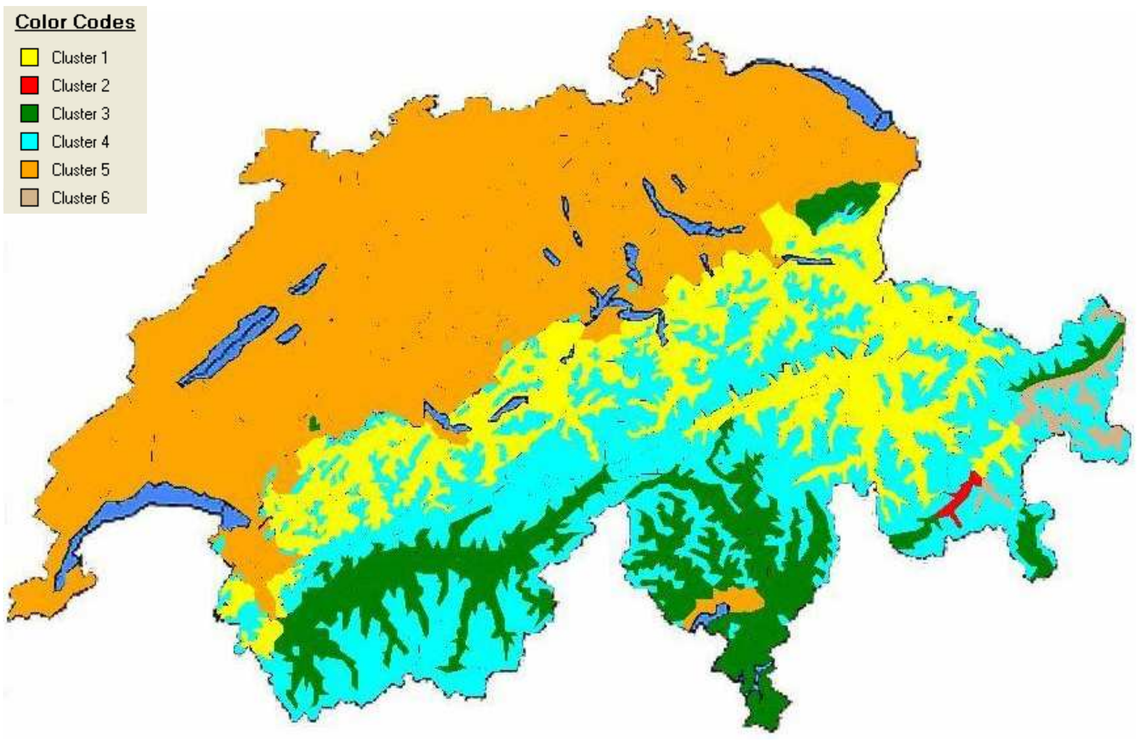


Figure 6.3-1: Clustering of Naïve Bayes method with 2 root nodes (3 & 4 states) on the original dataset

The results show not all the states have found relevant areas. Although we put three and four states respectively in the two root nodes respectively, only six clusters appear instead of twelve as we expect. We can see a big orange cluster, which covers all the north part of the map that comes from the two big clusters. In fact, the two big clusters belong to each of the two root nodes individually. This cluster and what the big two clusters come from represent a vast lowland area that has common plants. Hence they do not have enough differences, which disable the method to divide them.

The method has also recognized the main mountain area to the south of the land and two different lowlands around this one. The yellow cluster represents a zone affected for both, the mountain areas to the south and the big lowland area to the north so it has a different flora as NB shows. The green cluster is again built with the strange lowland areas to the south and west that in trimmed k-means approach were trimmed. Since

Naïve also recognizes these as different areas we have more power to agree that affirmation.

The red area is less significant because it just means that in each root node it was assigned to different clusters. Cluster 6 probably appears for the same reason. Knowing this we can say that both root nodes make a similar but not equal clustering because for concrete areas each node takes into account different probabilities to assign them to the most probable class.

About the complete dataset we considered that the data is not correctly ordered, there is not order relationship between the data which has the value "rare" and the data which has the value "registered 1984". We cannot order this data so we decided not to spend much time making many experiments in this dataset.

6.4. Binary dataset

We tried to apply this model to the binary dataset. The dataset has the value 1 for plants which has some kind of presence in the area, and 0 when there is no data about the presence of the plant in the area.

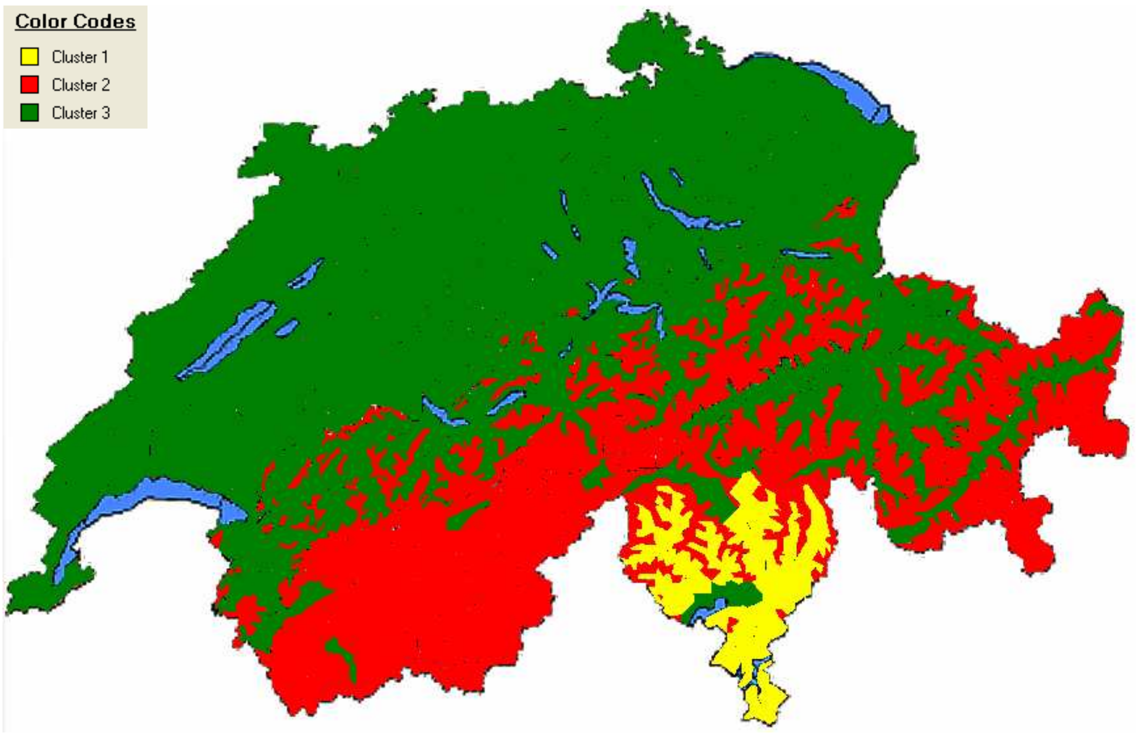


Figure 6.4-1: Clustering of first root node (3 states) of Naïve Bayes method on the binary dataset

The first test was made using two root nodes, one of them with three states and the other with four states. The results of each node are shown in Figure 6.4-1 and Figure 6.4-2. In these results we observe that in the two nodes there is a big cluster that contains most of the areas, and this big cluster is in the north of the map in each time. Furthermore, in middle zone of the map we see that in both clustering we obtain different clusters; it seems that there are two clusters in that zone and each of the roots recognize one of them.

The big green cluster on root 1 represents all the lowland areas that have general similarities. That's why it does not cover other lowland areas to the south that, as we have seen several times, have special vegetation. On the other hand, the big red cluster represents most of the mountain areas and, as we can see, it also takes the lowlands to the west. This means that these areas are more similar to mountain areas than to lowlands, which supports again the conclusion of a very particular flora.

In the second root clustering we can see that the big yellow cluster has grouped together the northern lowlands and the southern mountain areas. The main differences that this root has found are between the lowlands to the south and center of the map. The big yellow cluster must have taken areas from mountain and lowlands in an equal measure and the probability distribution has come in this way.

Finally another cluster that both nodes have found is in the south appears; however, the first root only assigns the southern areas to it whereas the second root also clusters some areas to the east and many to the west of the county. We must take into account that the second root has found only 3 clusters although it was initialized with 4 states. One of the states has been taken to a low probability in all the cases so no areas have been assigned to it. Checking the combination of the results we can see with more detail which clusters have been found.

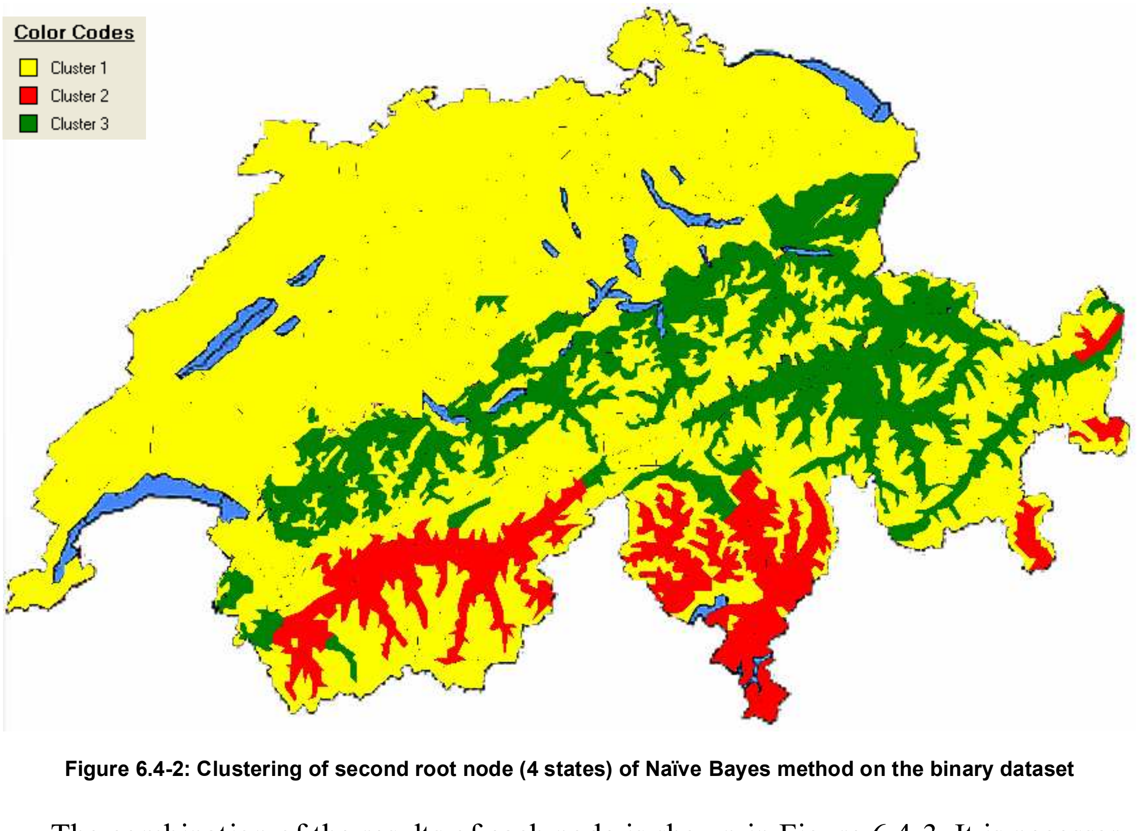


Figure 6.4-2: Clustering of second root node (4 states) of Naïve Bayes method on the binary dataset

The combination of the results of each node is shown in Figure 6.4-3. It is necessary to say that the combination of the two roots with three and four clusters respectively should return twelve clusters; but first, the second node did not find four clusters it should find, it only found three; and second, some of the combinations between clusters have not got any area in them. That is why after the crossing we only obtain 5 clusters instead of the 12 we expect to get.

With these results we can see how there is a big cluster again situated in the north of the map. As we commented when analyzing the original dataset's results, this area comes from the crossing of the two biggest ones in the roots clustering and it groups the common areas that, again, are the lowlands to the north.

In the middle zone there are two different clusters, numbers 2 and 5, which come from the crosses between the big red area from root 1 and the big yellow one from root 2 and both green areas from both roots. These clusters show both differences found on each of the roots between the lowlands and mountain areas to the south and center of the map. We can see another two relevant clusters that represent the two outlier zones we have talked about other times.

We can see that the join clusters of the two root nodes are more accurate because now we have more representative clusters. This happens because now we join the patterns found on the data by the two root nodes. With the two patterns we obtain more information from the dataset.

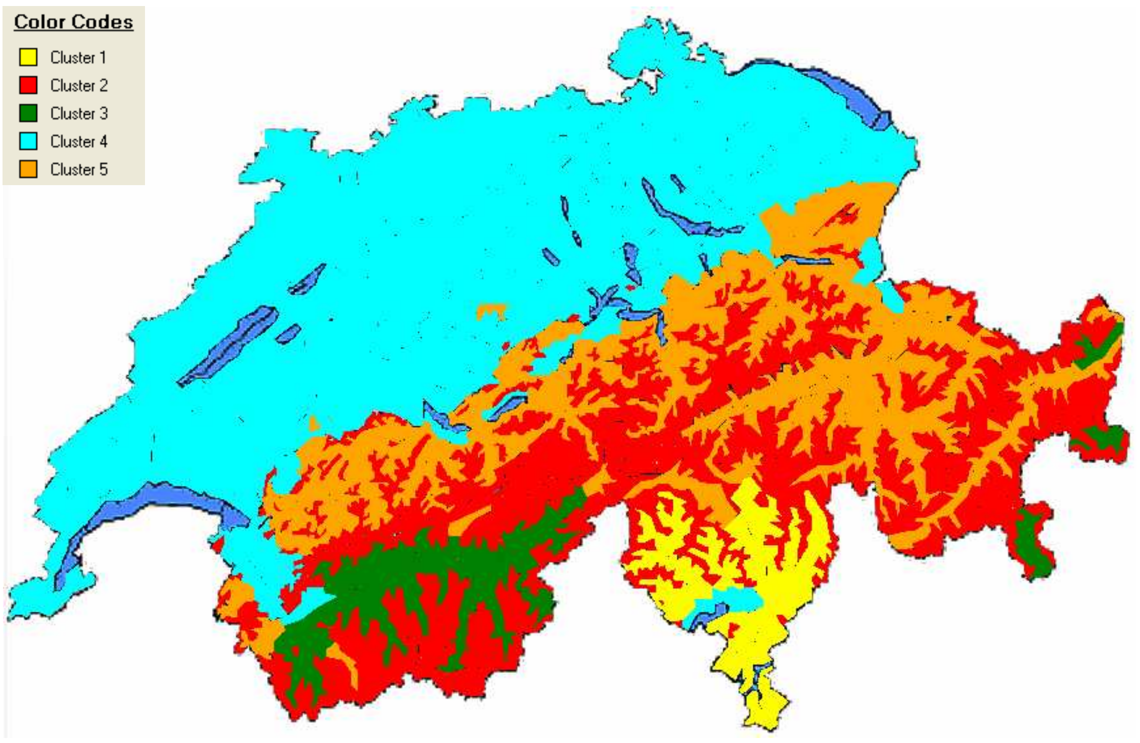


Figure 6.4-3: Clustering of Naïve Bayes method with 2 root nodes (3 & 4 states) on the binary dataset

The next experiment was done with one root node having eight states. The objective of the experiment was to compare the clustering with two root nodes and the clustering with only one. The results are shown in Figure 6.4-4. The results show both clustering assignments are quite similar with the crossed clustering. Both have the big main zone to the north that now is extended with some areas to the south and both sides of the map as a chain that tries to reach the southern areas.

Again the lowlands and mountain areas to the center and south of the country have been detected but now the outlier zones have been clustered together. This is similar to the trimmed 5-means run where these areas were trimmed because they were too far from their closest centers.

One interesting difference is in the small area in the center of the map. This area was clustered with the center lowlands on cluster 5 when we run the method with two root nodes; and now, with the one root node model, the method has found more similarities with the lowland areas on cluster 4. This means that the characteristic found by the one root execution is related to the fewer common plants because it groups this area with the southern ones, and with two roots both of them find the common similarities and that's why both cluster this area into one of the bigger clusters it generates.

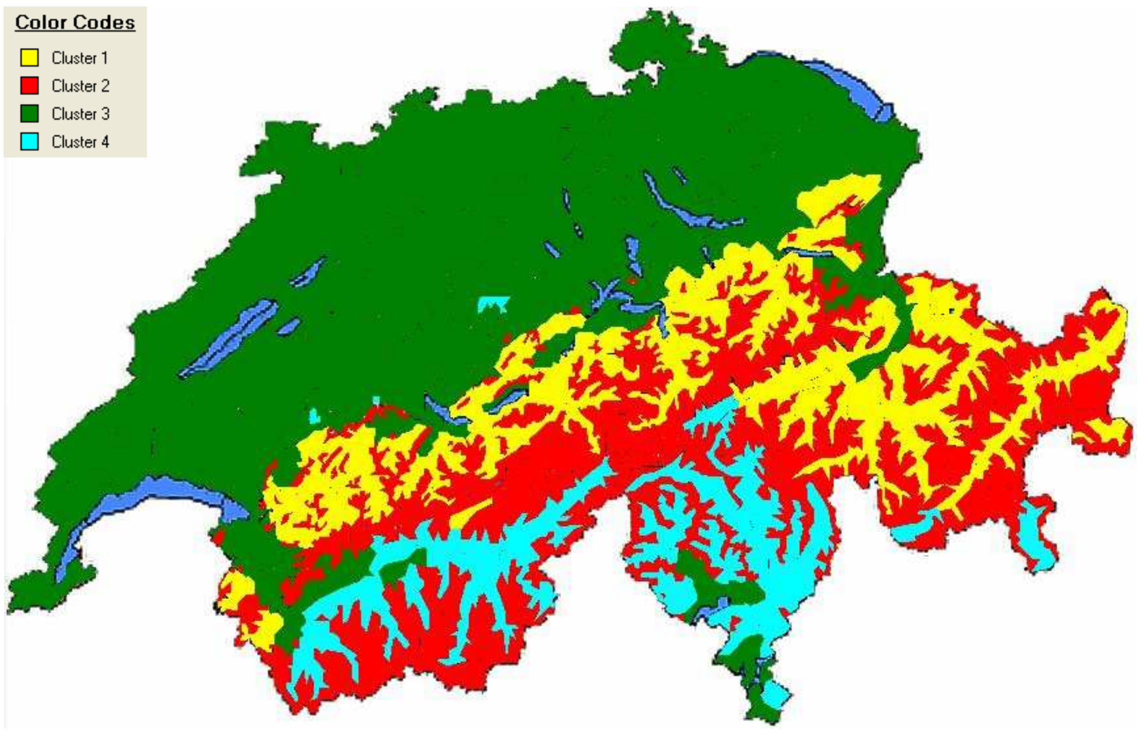


Figure 6.4-4: Clustering of Naïve Bayes method with 1 root node (8 states) on the binary dataset

Another interesting thing is that this small area was assigned to northern mountain areas in the k-means approach. We are not comparing both clustering now; but it is necessary to mention that NB does not consider the distances between instances whereas k-means does. Hence, with a distance criterion, the similarities and differences of the data have a strongpoint on the numerical results. NB does not have this strongpoint but it can find peculiarities in the data that then are used to suggest a possible clustering of that data.

We found out these clustering results obtained with one root node are depending on the number of initial clusters given to the node but it does not conditions the resulting number of clusters. As we can see in Figure 6.4-1 and Figure 6.4-2, even asking for a low number of clusters, the method does not guarantee to find all them even when they exist, as we can see in Figure 6.4-4.

The NB method finds a random amount of patterns inside the dataset having as maximum the a priori number of clusters given to the method. This is why the experiment with 8 states finds 4 clusters and the experiment with 4 states does not find the fourth cluster.

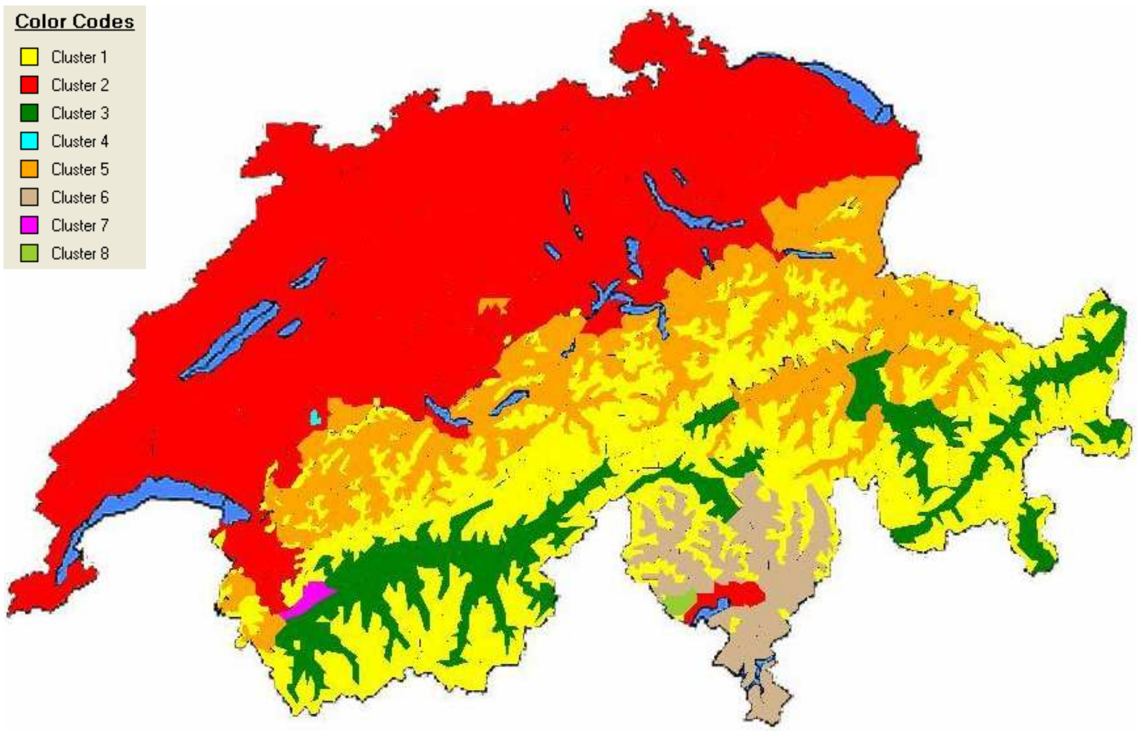


Figure 6.4-5: Clustering of Naïve Bayes method with 2 root nodes (4 states each) on the binary dataset

The last experiment we run with this dataset was with two root nodes having four states individually. The reason for this is trying to obtain more clusters from the same basis we applied when we run the method with 2 roots of 3 and 4 states respectively. The results can be seen in Figure 6.4-5.

Nevertheless the results reveal only a few more clusters because of few differences in the roots clustering. We obtained eight clusters but three of them have only one member so they are not relevant. The clusters found are quite similar but not the same because although the big cluster to the north and the one to the south now in yellow are almost the same, they have been changed on the configuration of cluster 5 that now have fewer areas since some of them have been assigned to cluster 3.

With this fact we can say that now the NB has found the particularities that makes cluster 3 different from the others and, as we saw on the small map for the 0.15 trimmed 5-means, outlier and cause perturbations to a distance based method as k-means is. Nothing to mention about cluster 6 that again is almost the same cluster except for one of those three areas without relevance.

As we commented before, we have noticed that in all of the experiments executed with this dataset appears one area in the middle of the map, inside the big cluster, that belongs to another cluster; it reveals that it must contain one species or several species which are not common in the neighbor areas.

6.5. Transformed dataset

With the transformed dataset we rescaled the data; this has been done to try to have consecutives values representing a scale on the amount of plants information. With this dataset we tried to make the same experiments that with binary dataset.

Figure 6.5-1 shows the experiment with only one root node and eight states. In this case there are again five representative clusters because cluster 3 is only composed of three areas and these areas are spread along the country. The big cluster is again on the north of the map but this time it does not covers that many areas to the south and sides of the land. cluster 5, in the middle zone, is bigger than before on one hand because of the areas that cluster 2 has not been taken and on the other hand because the importance of cluster 4 has decreased with this dataset and it has lost some areas close to cluster 5.

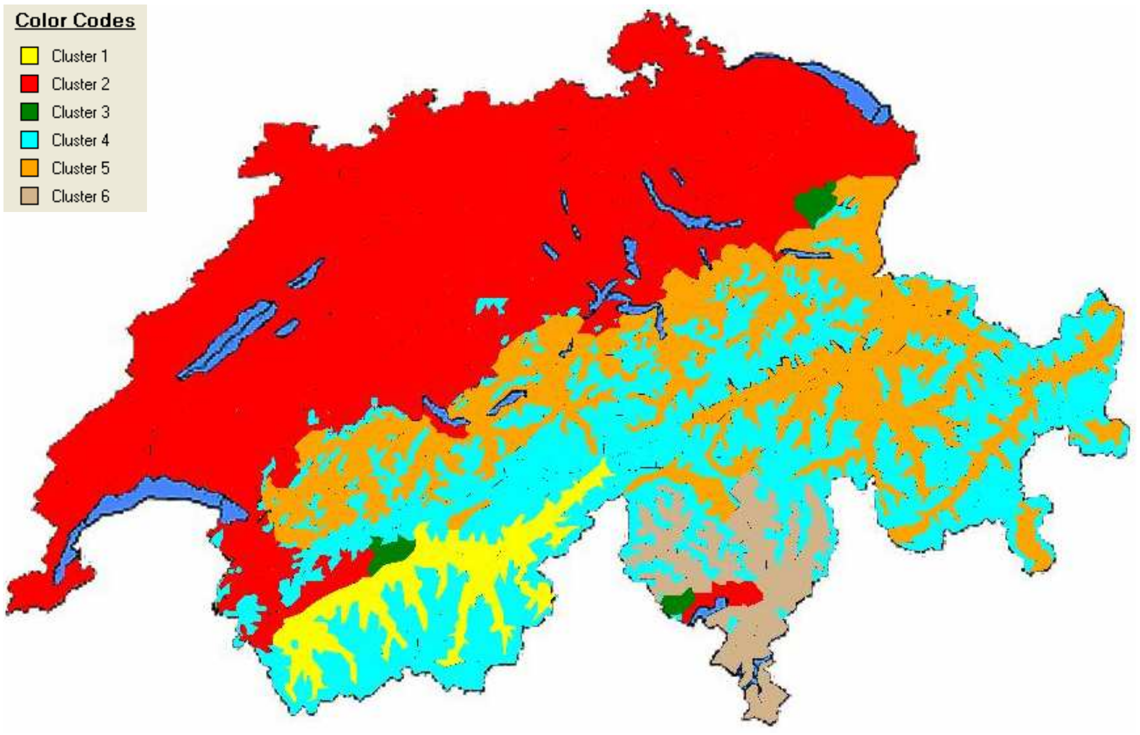


Figure 6.5-1: Clustering of Naïve Bayes method with 1 root node (8 states) on the transformed dataset

The method has found the cluster on the depression to the west and it have distinguished it from the southern mountain areas, so we can again support the conclusion that the number of initial states is relevant to the method to search only main similarities or also particularities on the data. Related to the dataset we can say that now it has information to classify a better way because with the presence-absence dataset the method assigned areas to the big cluster due to its size. This happened because with just the presence-absence information the decisions are taken as a binary function when the real information has a multinomial distribution.

Figure 6.5-2 shows the second experiment we made was with two root nodes with three and four states respectively. Now they were more different as we can see in the graphics. The big area to the north has again been reduced and confined to that part of the map. Two of the cluster, numbers 4 and 6, appeared, which shows discrepancies between the two root nodes when clustering some areas.

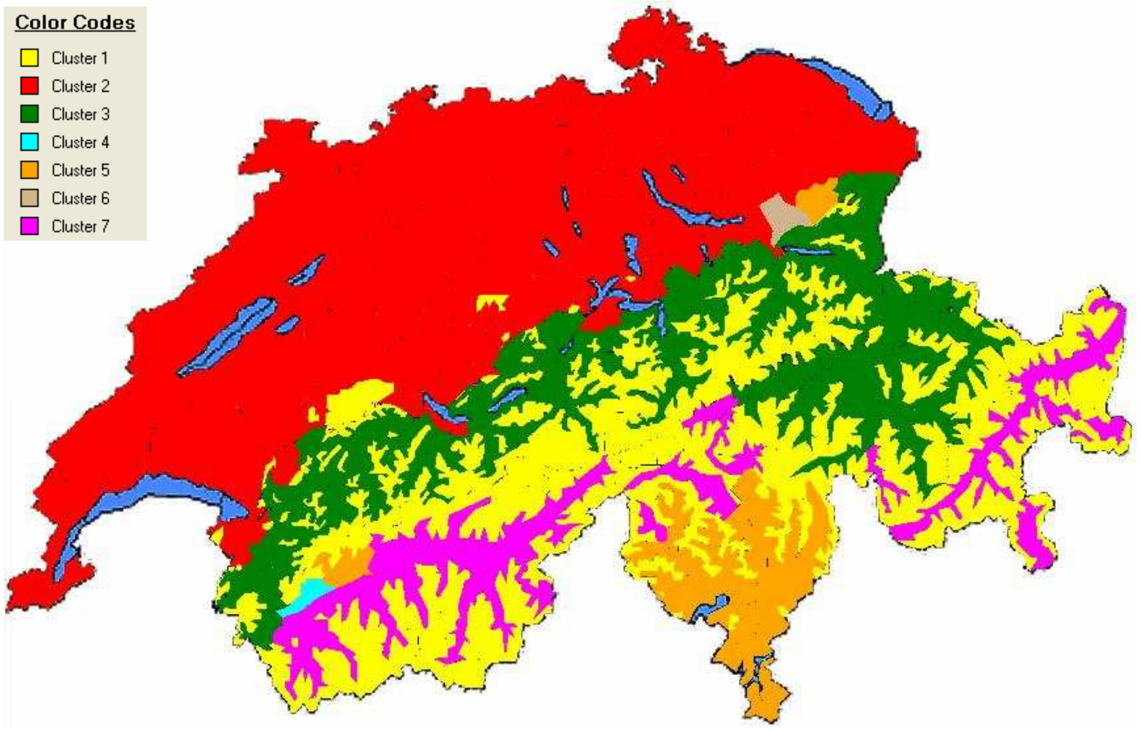


Figure 6.5-2: Clustering of Naïve Bayes method with 2 root node (3 & 4 states) on the transformed dataset

The big mountain area to the south has been kept almost the same as before; but now it has taken a set of areas from the big red cluster to the north and the small area in the middle of the land. Let us remember that this small area was grouped with the center lowlands using only the binary dataset but, as this cluster has also been reduced and many of its areas have been clustered into cluster 7, the particularities of the small land made it been assigned to this mountain areas cluster.

Cluster 5 is almost the same as with the binary dataset and, in fact, it has grown because now the power of the big cluster is less and the probability distribution is more accurate. And back to cluster 7 it has been revealed as an important cluster that supports the theory of similar vegetation along both depressions to the east and to the west. These particularities found in the dataset are the ones that we were supposing in the k-means approach and now we can say that with enough information, not only the presence-absence criterion, the mapping of the land can be done by lowland and mountain areas and differencing the north, south and center of the country.

As we noticed and expected, the results obtained with the two datasets have the differences we explained above. Unlike the k-means approach, the NB method does not take into account any distance criterion among the instances; but there still appear difference between the results because with the binary dataset the method can only treat the presence/absence of the plants while with the transformed dataset it has more information in the abundance of the plants.

This information is used to update the probabilities, and for this purpose, the method has more freedom if the information used follows a Multinomial Probability Distribution instead of a Binomial Distribution that is what happens when the binary dataset is used.

6.6. Conclusions

The main conclusion we can obtain is that EM algorithm applied in a Naïve Bayes structure cannot find many clusters from the data that we have. In all the cases we obtained fewer than five relevant clusters so we can conclude that the method is good as the first approximation to the clustering. About the dataset we can conclude that there are five well defined clusters that the algorithm could find; however, the algorithm cannot reach further into the data.

We have found that the results of the Naïve Bayes method depends on how many clusters the method is looking for, because when it looks for a few number of clusters it finds the main similarities in the data; while, if it is searching for more clusters it pays attention on the dissimilarities and peculiarities of each instance to update the probabilities tables.

We can conclude it is interesting to use two root nodes in the case of the binary dataset. When we applied the method with one root and 3 or 4 states we saw that, in both cases, the results obtained were not good clusters; and when we combined them we obtained more interesting and accurate clusters. In the transformed dataset the differences are less and the results with two root nodes and one root node are quite similar.

This happens because the method searches for a pattern within the data that can change in each run because of the random initialization of the prior distributions. This is why it has advantages to run the method with two root nodes because each of them can find different patterns and after that, these patterns are combined; and using just one node, even with a big amount of states, it will find clusters following just one of these patterns within the data.

Finally we can conclude again with the importance of the dataset moreover for this algorithm because the calculus of probabilities is more accurate using a multinomial distribution function to model the data in the nodes, and because using a binary decision variable generates a clustering that only shows the main similarities in the data and leaves the particularities hidden.

7. Comparative Analysis of Different Approaches

7.1. Introduction

In this chapter we are going to analyze the differences between the clustering approaches taken on the project. We are going to comment the differences found as well as to explain the causes of the different results obtained. To do that we will use the result obtained in the previous chapters and the utility "Clustering Comparison" integrated in the application that we have developed. This utility is destined to compare two different clustering results given in the concrete file format we explain below.

7.2. Clustering comparison

This utility was implemented by us in the CAV-2D application. It is accessed by the tools menu, and it permits load two files of clustering results and compare them. It compares using two different criteria, and it shows the results in text format and a graphical representation on the map. The user can also base the analysis on any of the two results files obtaining a different explanation of what happen.

To compare two clustering results, these results must come in our ".CA" file format. This format is a plain ASCII with one line per cluster and the areas belonging to it separated by commas. The user can create its own .CA file but we recommend using the ones that are automatically generated by the application. After that, you have to choose which of the clustering results will be the base file for the comparison and the comparison criterion that you want to apply.

We have developed two comparison criteria and called them the *Maximum concordance criterion* and the *Minimum discordance criterion* because the first one looks for the pairs with more areas in common and the second one tries not to leave clusters without pair taking into account the percentage that a pair represents in the two clusters that become a pair, not the size.

7.2.1. Maximum concordance criterion

The maximum concordance criterion (max criterion) is the simplest criterion available in the application. This criterion treats as similar clusters of the two results the one with the maximum number of areas in common. It just counts the number of areas that are in the two clusters and choose firstly the pair which has more areas in both clusters.

In this criterion the significance is related to the size if the cross related to the whole table, this is, the value of each cell related to the sum of all the values in the table. Said in other words, this method looks for the higher values in the table and sort them descending to select the pairs.

To explain the criterion we are going to show the Example 7.2-1:

Example 7.2-1: In this example we compare two clustering files, one with five clusters and the other one with two clusters. We are basing the analysis on the five clusters file, so these five clusters are named with the letters A-E. In the same way, the surrogated clustering is named with numbers 0-1.

The Table 7.2-1 is the equivalence table that shows the number of instances that both clustering results have in common. For instance, the number 228 in A0 means that between the cluster A from the base file and the cluster 0 from the surrogated file there are 228 areas in common.

	0	**1**
A	228	117
B	195	1
C	20	0
D	0	3
E	0	1

Table 7.2-1: Clustering equivalence table

With this equivalence table, the next step is to obtain the most significant pairs applying the criterion, in this case the maximum concordance. The results of the criterion are shown in Table 7.2-2.

Pairs	Significance
A-0	40.35%
B-0	34.51%
A-1	20.70%
C-0	3.539%
D-1	0.530%
B-1	0.176%
E-1	0.176%

Table 7.2-2: Maximum criterion concordance results

The significance obtained for each pair is calculated dividing each cell in the equivalence table by the sum of all cells[16]. This is showed as a percentage to make it easier to interpret. It represents the size of the cross in the table and we can sort the pairs by this value.

The method now selects the pairs with a higher significance identifying them as equivalent clusters. For the max criterion the first pair will be A-0 because it has the highest significance. This means that the highest value in the table is in cell A0 and that crossed-cluster has a 40% of the areas in it. As this criterion only looks for the highest crossed-clusters this is the first the method will identify as a pair.

[16] This is the number of areas that must be the same for both clustering results.

The next pair with a high value would be B-0 but cluster 0 is already assigned as equivalent to cluster A so this pair is not considered. We go down to the next maximum significance until we get a pair where none of the clusters has been assigned as equivalent to other and we found that the only one available is the pair D-1 that is the last that we can make because the surrogated clustering only has 2 clusters. So with this criterion we obtain that cluster A corresponds to cluster 0, and cluster D corresponds to cluster 1.

7.2.2. Minimum discordance criterion

The minimum discordance criterion (min criterion) is a more complex criterion than the max criterion. The main idea of this criterion is to give more importance to the fraction that each cell represents on the two clusters that are crossed. This means to identify similar clusters using the number of areas that both have in common with respect to the amount of areas of each cluster. This is done because for the maximum concordance criterion the biggest clusters have more significance because they have more areas in common. As that amount of areas in common can be less representative than the areas in common for other pair we thought about another significance criterion.

Example 7.2-2: To make this criterion easier to be understood we are going to apply it to the equivalence table from Example 7.2-1. First we must build the "Relative Frequency Tables" (RFT) where we compute the proportion that each cell represents in the two clusters crossed on it. We can see that tables for on Table 7.2-3.

	0	1
A	228/443	117/122
B	195/443	1/122
C	20/443	0/122
D	0/443	3/122
E	0/443	1/122
	443	**122**

	0	1	
A	228/345	117/345	**345**
B	195/196	1/196	**196**
C	20/20	0/20	**20**
D	0/3	3/3	**3**
E	0/1	1/1	**1**

Table 7.2-3: Relative Frequency Tables

The next step is to multiply cell by cell the relative frequency tables to get the significance of each crossed-cluster. The resulting table can be seen in Table 7.2-4.

	0	1
A	34.01%	32.52%
B	43.79%	0.00%
C	4.51%	0.00%
D	0.00%	2.46%
E	0.00%	0.82%

Table 7.2-4: Minimum discordance significance table

And sorting these percentages we obtain the most significant pairs. Table 7.2-5 shows these significance results of applying the minimum discordance criterion to the equivalences table.

Pairs	Significance
B-0	43.79%
A-0	34.01%
A-1	32.52%
C-0	4.514%
D-1	2.459%
E-1	0.819%
B-1	0.004%

Table 7.2-5: Minimum discordance criterion results

> The significance obtained in this case is calculated as the product of the ratio of areas in common divided by each sum of areas for each cluster in the pair. Once we have the significance calculated, the sequence to obtain the comparison between is the same than in the maximum concordance criterion. With this criterion the results obtained are cluster B corresponds to cluster 0 and cluster A corresponds to cluster 1. This means that pair B0 has a high frequency in both clusters more than to have only a high number of areas in common.

As we can see in the example the clusters correspondences are different depending on the criterion used. This happens because the size of a pair does not mean the pair is representative. The representative of a pair is given by the frequency of the pair in the two clusters that form the pair.

Comparing the two results obtained in the examples above we can see that max criterion gives more importance to A0 because it is the biggest pair but it does not have as high representative ness as pair B0 that, for cluster B contains almost 100% of the areas.

Max Criterion	
Pairs	Significance
A-0	40.35%
B-0	34.51%
A-1	20.70%
C-0	3.539%
D-1	0.530%
B-1	0.176%
E-1	0.176%

Min Criterion	
Pairs	Significance
B-0	43.79%
A-0	34.01%
A-1	32.52%
C-0	4.514%
D-1	2.459%
E-1	0.819%
B-1	0.004%

Table 7.2-6: Comparison of both criteria results

As max criterion takes the pair A0, the next in significance, B0, is not considered as it happens with A1 and C0 and the next pair chosen is D1 that only has a 0.5% of the areas. On the other hand, the min criterion takes B0 with a medium significance (about 50%) but the second pair chosen is A1 that also has a medium-low (about 30%) significance.

7.2.3. Map Results Notation

The color codes for the resulting maps applying the Clustering Comparison tool of the program are a bit different from the codes used for the results of a clustering technique.

Now, the codes are related to the clustering equivalences and have concrete operators we are going to describe.

- Operator “:” (A:1). This operator indicates that the cluster A is equivalent to the cluster 1 and the areas painted with this color are the ones that appear in both clusters.
- Operator “\” (A\1). This operator refers the areas that are in cluster A but not in cluster 1. This operator appears when the comparison table has more columns than rows or the same number of columns and rows.
- Operator “∩” (A∩1). This operator indicates the areas that area both in cluster A and cluster 1. It appears when the number of rows is greater than the number of columns because not all the basic clusters can find an equivalent surrogated cluster.

7.3. Comparative Analysis

We begin now the comparison of the main clustering results obtained in the previous chapters. We are comparing the resulting maps and also applying the automatic comparison tool integrated in the application we developed with both comparison criteria to find the equivalences among results.

7.3.1. K-means Approaches Comparative

Here we are going to compare the results for both k-means approaches, the Simple K-Means and the 0.05-Trimmed K-Means. This comparison is based on the Simple K-Means results because it is the one where all the areas are clustered.

The first comparison is for k=5 with the binary dataset. The results of these two clustering methods are shown in Figure 7.3-1. As we can see, the trimmed procedure is not affected by the lowland areas to the south, because they are trimmed, and because of that the clusters in the middle of the land have almost disappeared.

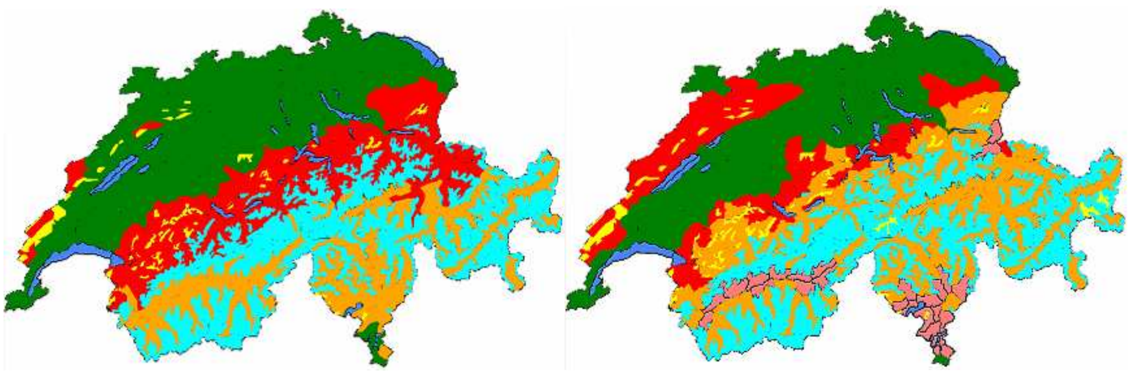


Figure 7.3-1: Results for simple 5-means [left] and 0.05-trimmed 5-means [right] using the binary dataset

With the trimming, those areas are not displacing the cluster center, and the big cluster in the middle of the map of the simple 5-means disappears while the cluster to the north of the trimmed map is revealed.

The results for comparison of these two clustering assignments applying the max criterion are in Table 7.3-1 and Figure 7.3-2. From the table we can infer the following equivalences:

- Cluster A corresponds with Cluster 0
- Cluster B corresponds with Cluster 2
- Cluster C corresponds with Cluster 1
- Cluster E corresponds with Cluster 4
- Cluster D is not crossed

Equivalences for surrogated clustering

	0	1	2	3	4
A	155	0	0	0	5
B	0	0	114	34	4
C	0	68	0	36	2
D	0	66	0	0	22
E	0	0	0	0	59

Pairs Significance

A-0	27.43%
B-2	20.17%
C-1	12.03%
D-1	11.68%
E-4	10.44%
C-3	6.371%
B-3	6.017%

Table 7.3-1: Results for simple 5-means and 0.05-trimmed 5-means comparison using the binary dataset

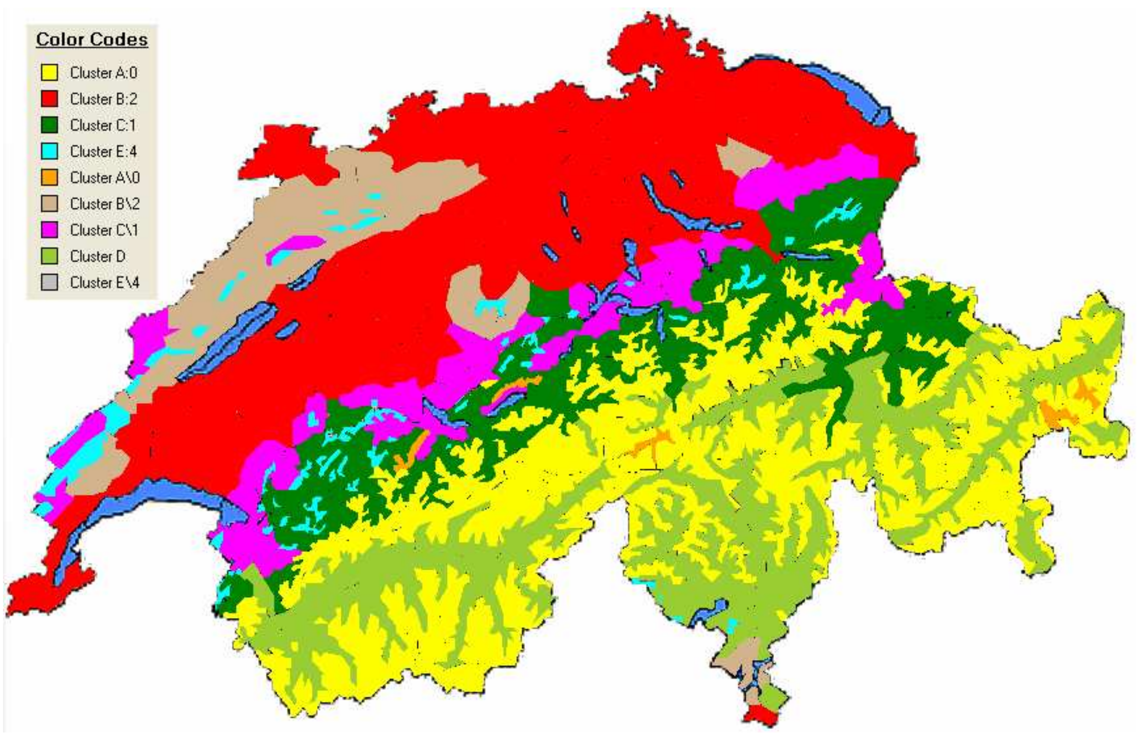


Figure 7.3-2: Map for simple 5-means and 0.05-trimmed 5-means max comparison using the binary dataset

From the map, we can see that the cluster that has more areas in common (in the original clustering results) is the one that represents the southern mountain areas and this means that this cluster is a very compact cluster because both methods find it with almost the same areas. We can see that the areas not in common for these two clusters (A\0) are few and spread through the center and east of the land.

The next important equivalence is B:2. This cluster represents the northern lowlands that have a common flora not affected by outlier areas. This happens because the trimmed clustering identifies as a different cluster areas to the north-west after identifying those outlier areas.

The next cross in importance for this criterion is C:1 but, as we can see, there is more importance in the fact that E\4 is void. This means that for both clustering the areas belonging to the cross E:4 are the same, so this is a very compact cluster although it is a small and spread cluster. As cluster D is not crossed because there is no cell with value different from zero where the surrogated cluster has not been assigned, it is shown as its own.

The max criterion gives results useful for finding the biggest crosses not considering the importance of those crosses on each of the clustering. This is why we are going to analyze the results of the min criterion. These results are showed in Table 7.3-2 and Figure 7.3-3.

Equivalences for surrogated clustering

	0	1	2	3	4
A	155	0	0	0	5
B	0	0	114	34	4
C	0	68	0	36	2
D	0	66	0	0	22
E	0	0	0	0	59

Pairs Significance

A-0	96.87%
B-2	75.00%
E-4	64.13%
D-1	36.94%
C-1	32.55%
C-3	17.46%
B-3	10.86%

Table 7.3-2: Results for simple 5-means and 0.05-trimmed 5-means comparison using the binary dataset

From the table now we can see that all the basic clusters have a surrogated equivalent because the criterion takes into account how much does each cross represents on both clustering. The correspondences now are:

- Cluster A corresponds with Cluster 0
- Cluster B corresponds with Cluster 2
- Cluster E corresponds with Cluster 4
- Cluster D corresponds with Cluster 1
- Cluster C corresponds with Cluster 3

We can see in the map that the two biggest crosses are the same as before but now the third in significance is the one we talked about above, the cross E:4. This cross now has 100% of instances from cluster E and the highest number of instances from cluster 4. This is why it has been relegated to the third place in significance but it still has been identified as an equivalent pair.

Now the method has found an equivalent cluster for cluster D that represents the lowlands surrounded by mountains to the south. As the trimmed procedure has found some of these areas as outliers now we can see them in D\1.

At this point, the interest to analyze C:3 is lower than the interest to analyze C\3 because C\3 represents areas that were miss-clustered on the simple 5-means because of the outliers and C:3 are areas that have been assigned to the same cluster for both methods. This also happens with B\2 that represents another set of miss-clustered areas to the north again because of the outliers.

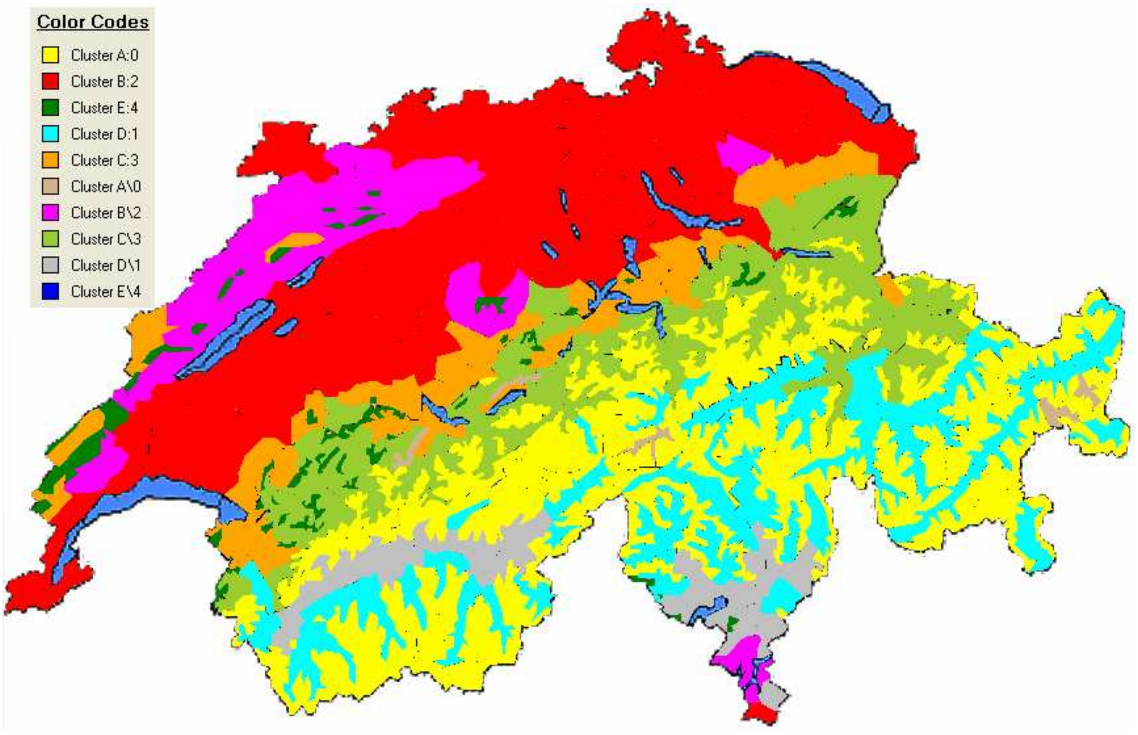


Figure 7.3-3: Map for simple 5-means and 0.05-trimmed 5-means min comparison using the binary dataset

The second comparison is for k=8 also with the binary dataset. The results of these two clustering methods are shown in Figure 7.3-4 and the results for the comparison are on Table 7.3-3 and Figure 7.3-1.

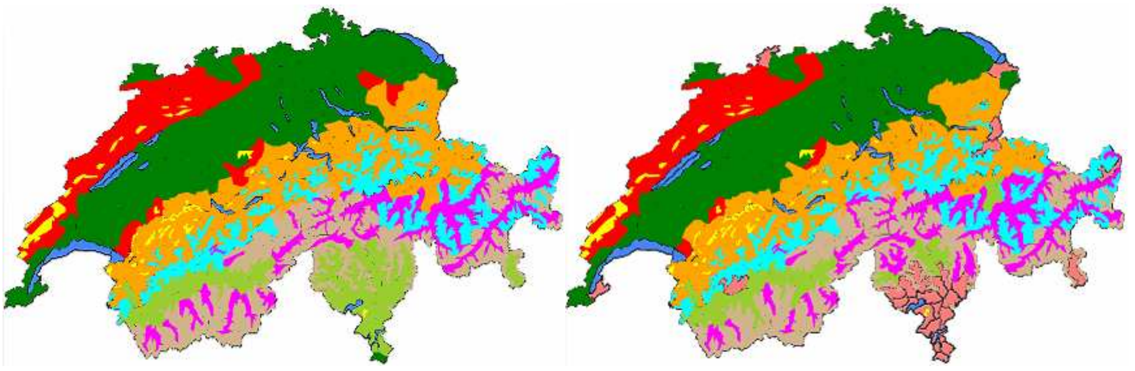


Figure 7.3-4: Results for simple 8-means [left] and 0.05-trimmed 8-means [right] using the binary dataset

As we can see in Figure 7.3-4, the results now are very similar so we can conclude that, for k=8, the outliers does not have an important impact on the binary dataset. In fact, as we can see in Table 7.3-3, the crosses are almost relegated to the diagonal which will make that the equivalences will be clearly determined.

The only difference between the max criterion and min criterion now is the order of the equivalences. That is why we only show one of the maps, concretely the max criterion map.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6	7
A	105	0	0	0	0	0	0	4
B	0	90	0	0	2	0	0	4
C	0	0	91	0	0	0	0	0
D	0	0	0	75	0	0	0	0
E	0	0	0	0	47	0	0	3
F	0	0	0	0	0	50	0	0
G	1	3	0	0	0	0	45	0
H	0	0	0	0	4	0	0	41

Max Criterion		Min Criterion	
A-0	18.58%	C-2	100.0%
C-2	16.10%	D-3	99.99%
B-1	15.92%	F-5	99.99%
D-3	13.27%	A-0	95.42%
F-5	8.849%	G-6	91.83%
E-4	8.318%	B-1	90.72%
G-6	7.964%	E-4	83.35%
H-7	7.256%	H-7	71.83%
A-7	0.707%	H-4	0.670%

Table 7.3-3: Results for simple 8-means and 0.05-trimmed 8-means comparison using the binary dataset

As we can see in Table 7.3-3, the bigger cross is A:0 because is the one that, in both cases, covers most of the northern lowlands of Swiss but the most accurate cross is C:2 that represents the southern mountain areas and in both clustering results it has the same areas. This means that this cluster is the most compact in this analysis, but also D:3 and F:5 are very compact because they also have all the areas in common but are smaller than C:2.

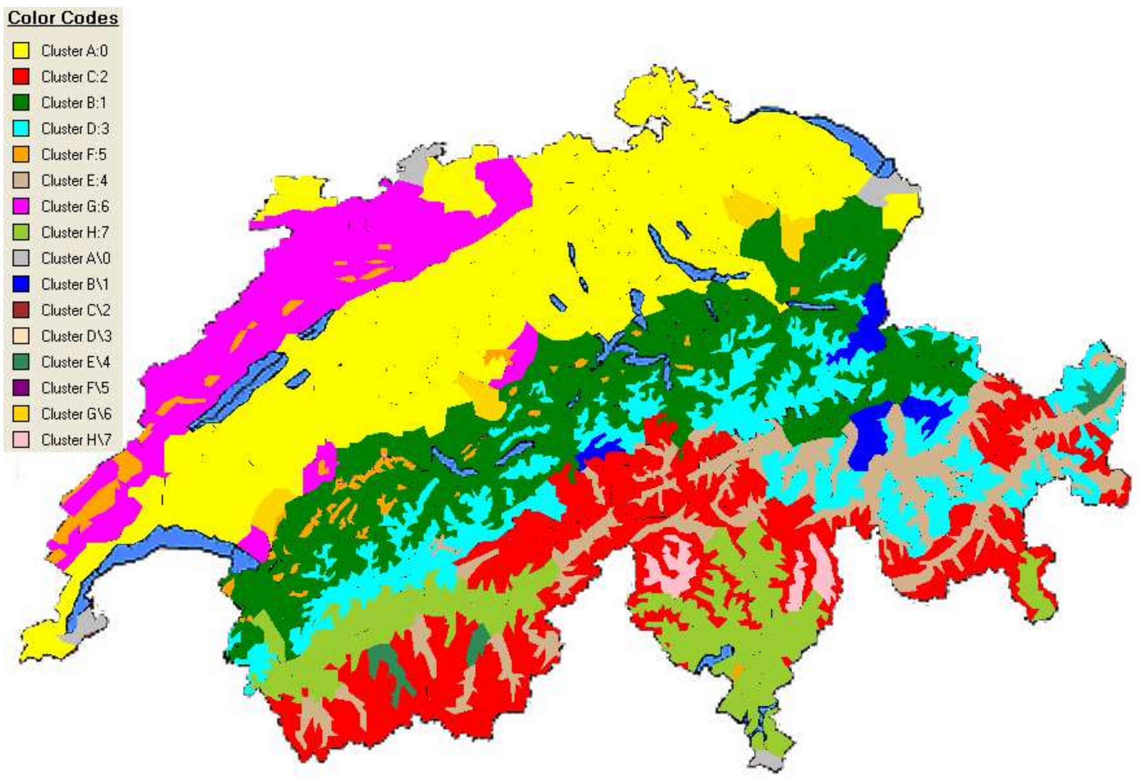


Figure 7.3-5: Map for simple 8-means and 0.05-trimmed 8-means max comparison using the binary dataset

Checking these crosses in the map we can see that these three clusters are the three composed of mountain areas so we can conclude that the mountain flora is very special and specific, and that there are three different specializations across the land, one to the north and center of Swiss and two to the south. As we do not have information about the height or other climatic conditions we cannot say if these clusters answer to any of those climatic factors.

About the lowland areas, there also exist three main clusters to the north, A:0, G:6 and B:1, that change in some areas when the trimmed 8-means is applied but this effect is not very accused.

Now we are going to consider the transformed dataset. With it we are going to compare again the simple 5-means with the 0.05-trimmed 5-means and the simple 8-means with the 0.05-trimmed 8-means.

The maps for k=5 are showed in Figure 7.3-6 while the results for the comparison are contemplated in Table 7.3-4 and Figure 7.3-7. In those maps we can see that the differences between the simple and trimmed results are not very different even without the outliers. This means that, with the distance information contained in the transformed dataset the influence of the presence of outliers now is not as high as before. This is because the amount takes an important role on the clusters construction and the outliers influence.

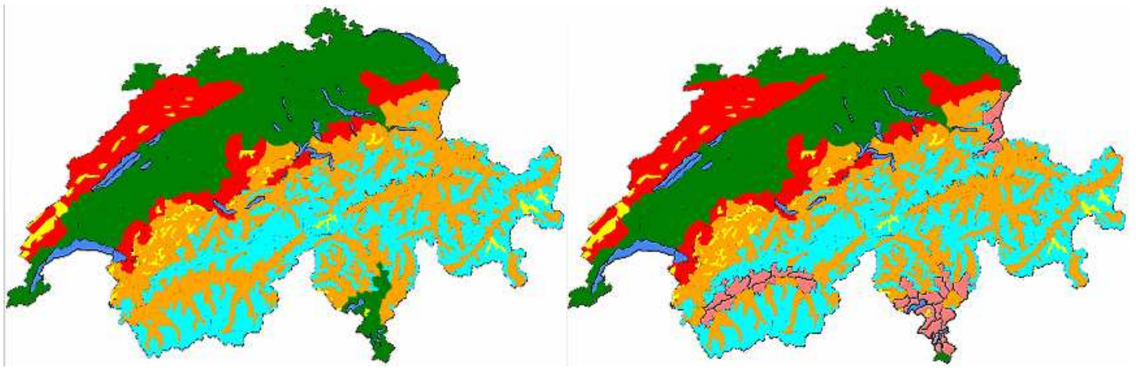


Figure 7.3-6: Results for simple 5-means [left] and 0.05-trimmed 5-means [right] using the transformed dataset

As it happened with the previous analysis, the differences between the max and min criteria are just related to the importance given to the crosses but in both cases is clear which clusters are going to be equivalent seen just the equivalences table.

Equivalences for surrogated clustering

	0	1	2	3	4
A	0	143	0	1	16
B	157	0	0	0	0
C	0	0	111	2	12
D	0	0	0	62	0
E	0	0	0	0	61

Max Criterion		Min Criterion	
B-0	27.78%	B-0	100.0%
A-1	25.30%	D-3	95.38%
C-2	19.64%	A-1	89.37%
D-3	10.97%	C-2	88.80%
E-4	10.79%	E-4	68.53%
A-4	2.831%	A-4	1.797%
C-4	2.123%	C-4	1.294%

Table 7.3-4: Results for simple 5-means and 0.05-trimmed 5-means comparison using the transformed dataset

We can see in the map that again the most compact cluster is the one representing the southern mountain areas because it is not affected by the outliers. Other important clusters are D:3, A:1 and C:2 that represent the three main lowland clusters and seen them with this significance means that now the abundance information gives them a more accurate clustering criterion.

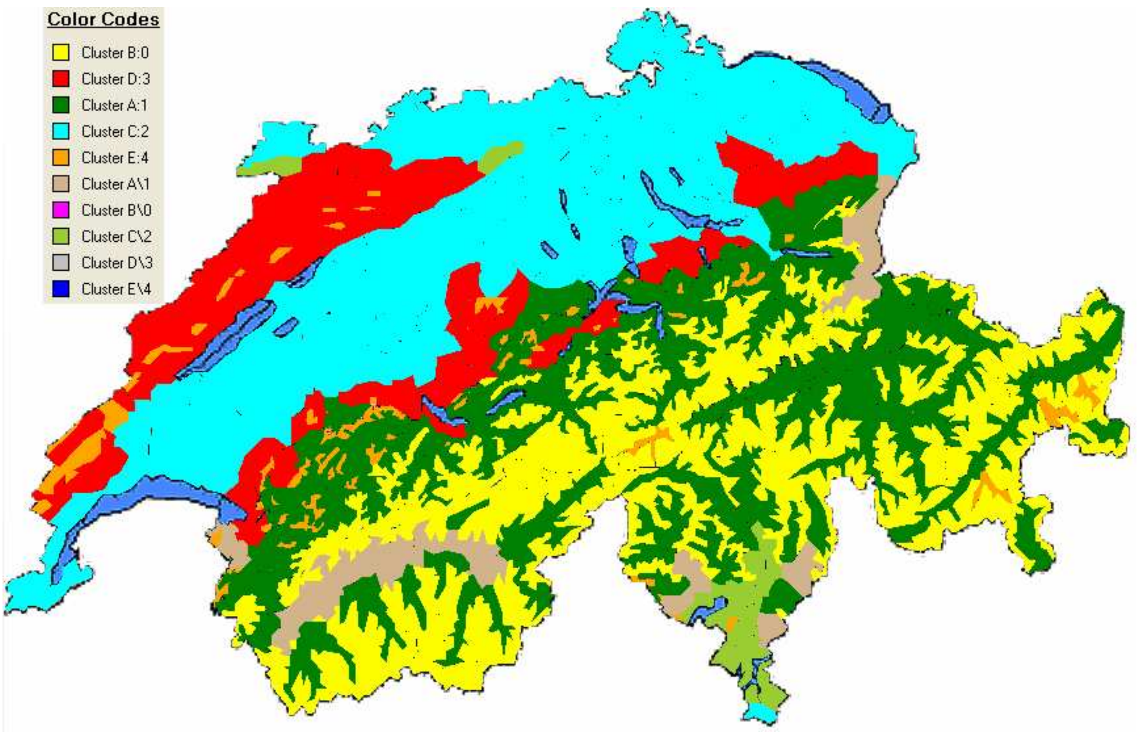


Figure 7.3-7: Map for simple 5-means and 0.05-trimmed 5-means min comparison using the transformed dataset

We can also see the outliers on A\1 and C\2 and now we can see that the northern mountain areas have less significance what means again that this clustering is better with this dataset because there appear more clusters independent or almost independent of the outliers.

The last k-means approaches comparison is among the 8-means results for the transformed dataset because for the previous comparisons the analysis did not report much information. Figure 7.3-8 shows the two clustering results and Figure 7.3-9 and Table 7.3-5 has the results of the comparison procedure.

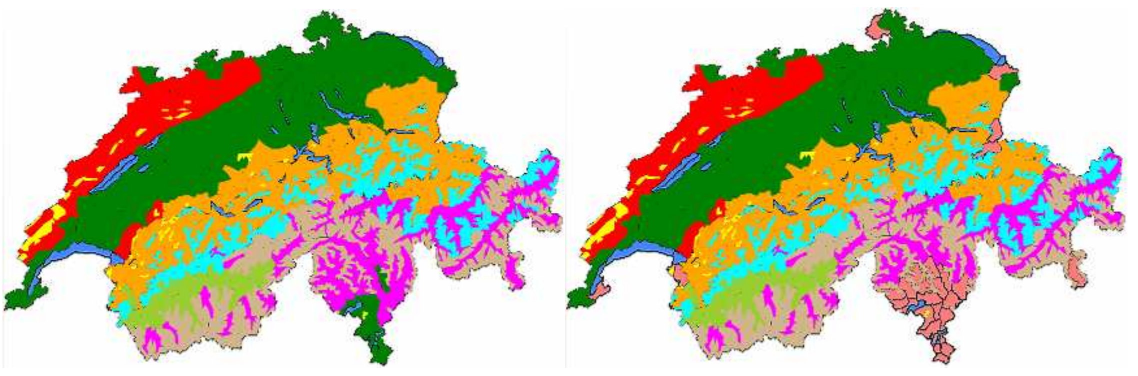


Figure 7.3-8: Results for simple 8-means [left] and 0.05-trimmed 8-means [right] using the transformed dataset

As we can see in Figure 7.3-8, the results are again very similar apart of the outliers to the south of the map. The impact of the outliers is almost null when the transformed dataset is used and the value of k is 8. In fact, as we can see in Table 7.3-5, the crosses are again almost relegated to the diagonal but now the cross H:7 has less power than before because it only has 18 areas in common from the 46 in cluster 7.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6	7
A	104	0	0	0	0	0	0	15
B	0	98	0	0	0	0	0	3
C	0	0	93	0	0	0	0	0
D	0	0	0	77	0	0	0	0
E	0	0	0	0	55	0	0	10
F	0	0	0	0	0	47	0	0
G	0	0	0	0	0	0	45	0
H	0	0	0	0	0	0	0	18

Max Criterion		Min Criterion	
A-0	18.40%	C-2	100.0%
B-1	17.34%	D-3	99.99%
C-2	16.46%	F-5	99.98%
D-3	13.62%	G-6	99.98%
E-4	9.734%	B-1	97.02%
F-5	8.318%	A-0	87.39%
G-6	7.964%	E-4	84.61%
H-7	3.185%	H-7	39.13%
A-7	2.654%	A-7	4.110%

Table 7.3-5: Results for simple 8-means and 0.05-trimmed 8-means comparison using the binary dataset

The interpretation is not very different from the one taken from the binary dataset comparison. The most important crosses are again the three mountain areas that are the ones that have the same areas in both clustering results. And again the next more important clusters are the three lowland clusters to the north that again differ in some areas from one clustering to the other.

Now cluster G:6, the north-western lowland cluster is not affected by outliers and the areas in it are the same for both results. A\0 is restricted to some areas to the south that are trimmed as it happens with E\4 but E:4 is less important cluster because it represents a concrete difference between the lowlands to the south.

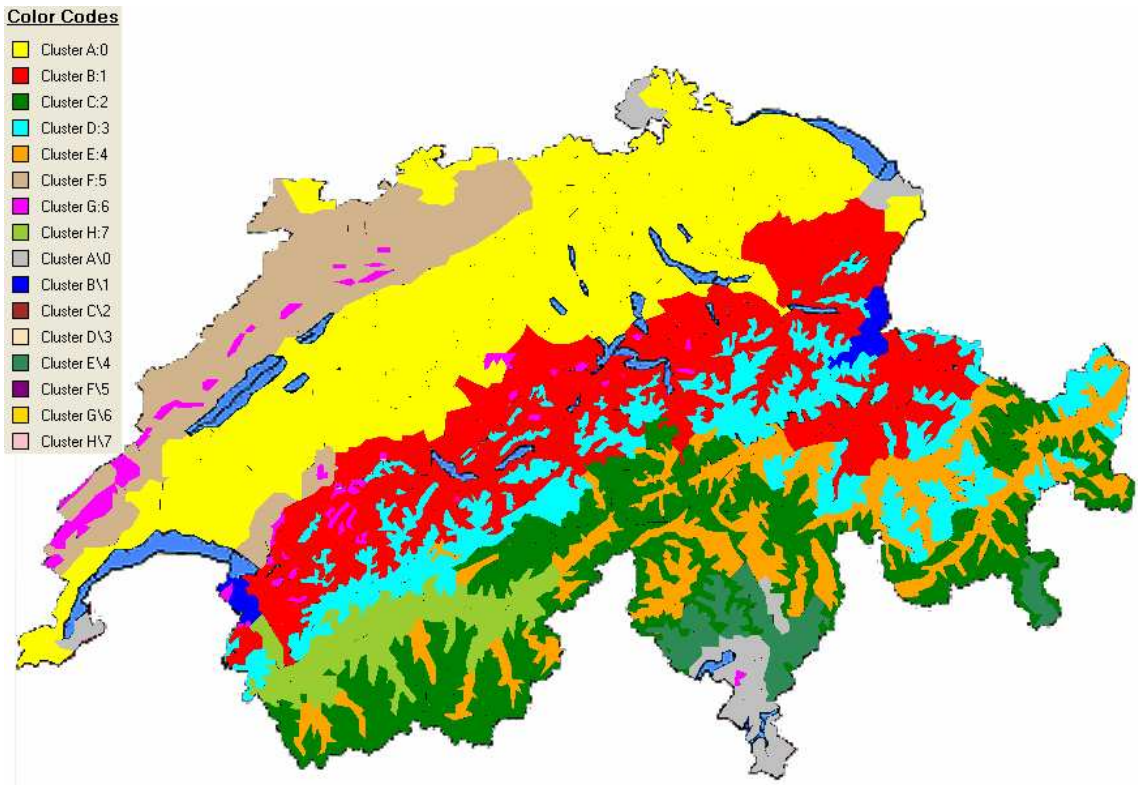


Figure 7.3-9: Map for simple 8-means and 0.05-trimmed 8-means max comparison on the transformed dataset

As we could see, the trimmed k-means is useful to detect the outliers and their influence on the clustering results. In chapter 5 we supposed this behavior but now we have seen when the outliers have more impact and when they are trimmed but they have no influence on the clustering.

When the outliers have a low influence the results of the simple k-means procedure are almost the same but, as we cannot identify those outliers before the run of the method it is always useful to check the trimmed k-means results.

7.3.2. Naïve Bayes Approaches Comparative

Now we are going to compare the results of two Naïve Bayes approaches, the approach with one root node and the run with two root nodes. For the one root node we initialized the number of states of the node to 8, and for the two root nodes run we initialized one root with 3 and the other with 4 states.

For both approaches we compare both, the binary and the transformed datasets, and we use the two comparison criteria basing the analysis on the network with one root node.

We begin with the comparison for the binary dataset. For this dataset we can see in Figure 7.3-10 the two maps resulting from applying the two Naïve Bayes approaches. In this figure we can see that the northern lowland areas cluster and the southern mountain areas cluster are very similar in both results. The yellow cluster to the south equals or almost equals in both maps.

The main difference lies in the central lowlands. These areas have been clustered mostly together[17] for the two root nodes; however, they have been split into the central and the south lowlands in the one root node run. This means that the initialization contributes on the performance of the method but does not impose a number of final clusters as it happens with the k-means method.

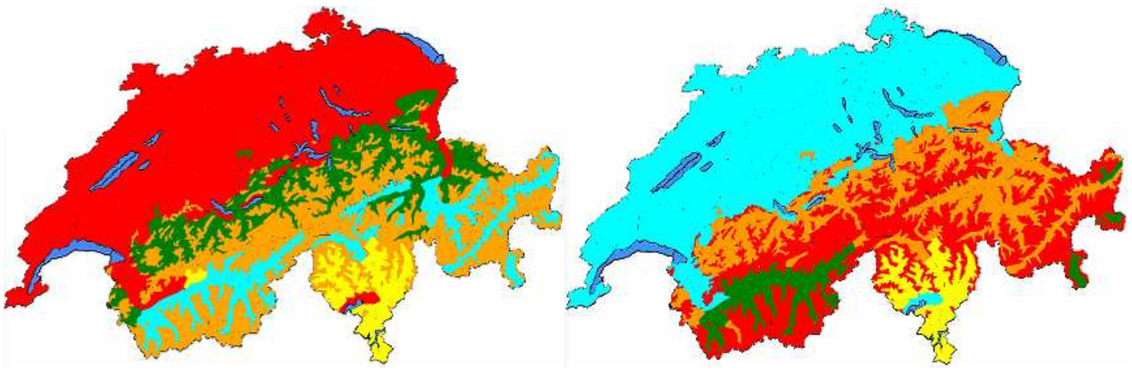


Figure 7.3-10: Results for Naïve Bayes 1 root [left] and Naïve Bayes 2 roots [right] using the binary dataset

Table 7.3-6 shows the results for the comparison of both clustering assignments based on the results obtained with one root node and the significance of each pair using the max and min criterion. Figure 7.3-11 shows the graphical results for this comparison.

We can see that the equivalences table has many zeroes and the bigger values in the diagonal. Because of that the pairs found by the comparison criteria will be the same differing only in the significance given to them. In both cases A:0 and B:1 are the most

[17] Apart from the south-western area clustered apart.

significant in size and in frequency captured. The next pair found by the max criterion is C:2 and the E:4 is the third in significance for the min criterion. These four pairs represent the four biggest clusters that are found by both runs of the Naïve Bayes method.

Equivalences for surrogated clustering

	0	1	2	3	4
A	205	0	5	1	0
B	1	199	2	0	0
C	0	0	73	0	0
D	0	0	28	24	0
E	0	0	0	1	26

Max Criterion		Min Criterion	
A-0	36.28%	B-1	98.51%
B-1	35.22%	A-0	96.68%
C-2	12.92%	E-4	96.29%
D-2	4.955%	C-2	67.59%
E-4	4.601%	D-3	42.60%
D-3	4.247%	D-2	13.96%
A-2	0.884%	E-3	0.142%
B-2	0.353%	A-2	0.109%

Table 7.3-6: Results for Naïve Bayes 1 root against Naïve Bayes 2 roots comparison using the binary dataset

As we can see in the map, both runs find a big cluster to the north and another smaller to the south that together captures more 50% of the areas. The next cluster is important because it only has one area not in common for both runs. This pair is E:4 and it could be dismissed taking into account only its size because it groups less that the 5% of the areas. But, as we have explained, this is an important cluster to consider. In fact, this cluster represents a group of areas that are particular they belong to the south of the land.

The central lowlands represented by C:2 are part of the third biggest pair but there exist no despicable amount of areas that are not belonging to this pair. These are lowland areas that are recognized as a different cluster by the 8 states root so they must have a special configuration of plants that the Naïve model with two roots does not recognize.

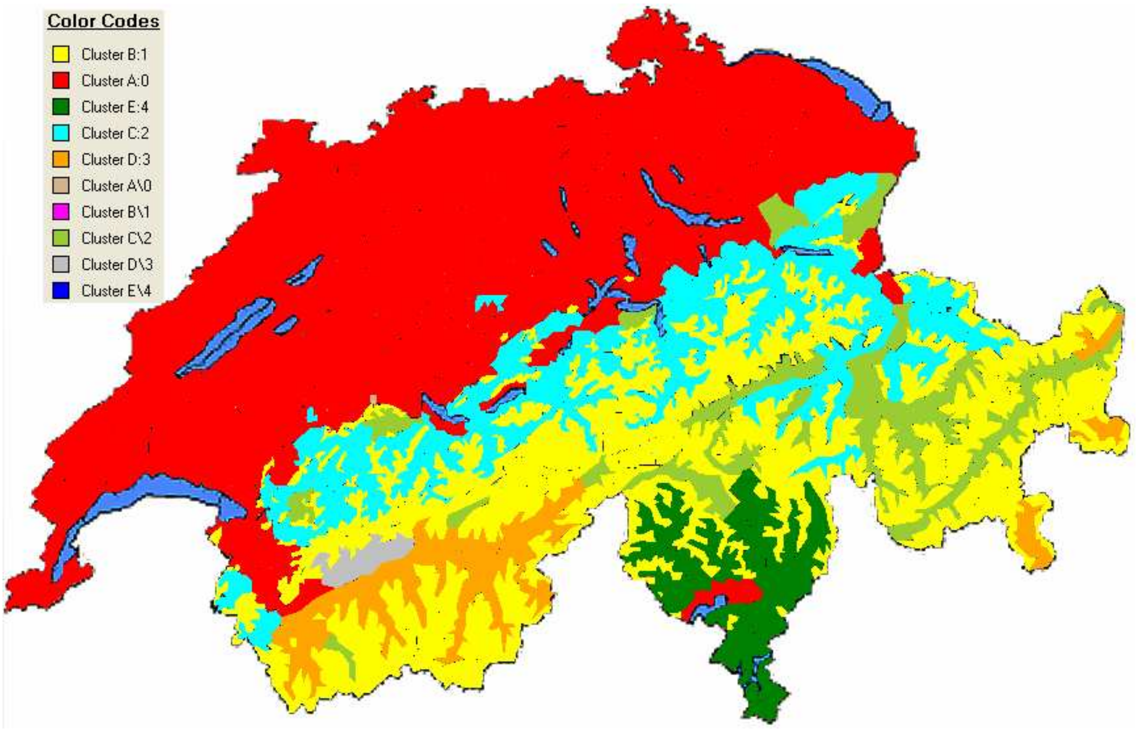


Figure 7.3-11: Map for for Naïve Bayes 1 root against Naïve Bayes 2 roots comparison using the binary dataset

Now we are going to compare the results with the transformed dataset. Again we are going to compare the two results based on the Naïve with one root node and applying the two comparison criteria.

The maps for both results are shown in Figure 7.3-12 while the results for the comparison are contemplated in Table 7.3-7 and Table 7.3-8 for the numerical results and in Figure 7.3-13 and Figure 7.3-14 for the graphical maps.

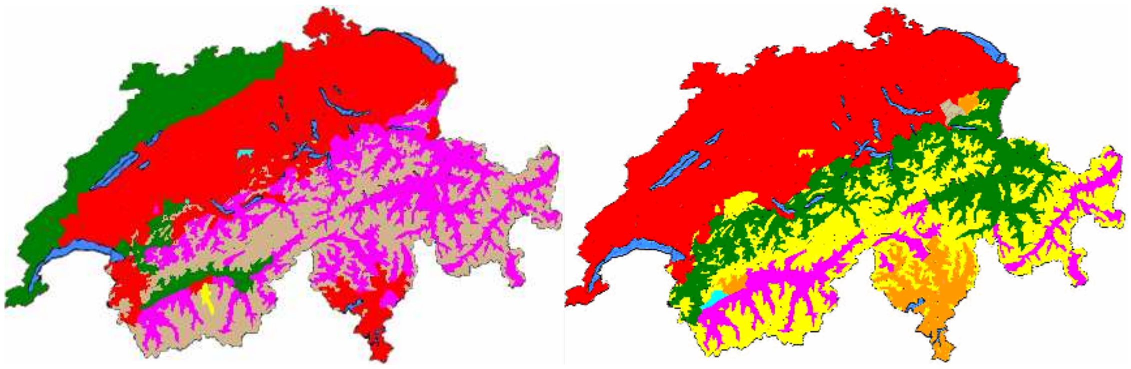


Figure 7.3-12: Results for Naïve Bayes 1 root [left] and Naïve Bayes 2 roots [right] on the transformed dataset

For this run of the Naïve Bayes with one root we find the results are a bit different from other runs. Now the method has found an extended cluster in the north of the land and the other three typical big clusters. One interesting thing is that now the method has found the small area in the middle of the land as a different cluster. This means that it has a very specific flora that makes this area special.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6
A	0	124	0	75	1	0	0
B	194	0	2	0	1	1	0
C	0	16	90	5	0	0	0
D	0	15	10	0	0	0	0
E	0	1	17	6	0	0	1
F	0	0	3	0	0	0	0
G	0	1	0	1	0	0	0
H	0	1	0	0	0	0	0

Max Criterion		Min Criterion	
B-0	34.33%	B-0	97.97%
A-1	21.94%	C-2	59.81%
C-2	15.92%	A-1	48.65%
A-3	13.27%	A-3	32.32%
E-2	3.008%	E-2	9.475%
C-1	2.831%	D-1	5.696%
D-1	2.654%	E-6	4.000%
D-2	1.769%	D-2	3.278%
E-3	1.061%	F-2	2.459%
C-3	0.884%	E-3	1.655%

Table 7.3-7: Results for Naïve Bayes 1 root against Naïve Bayes 2 roots on the transformed dataset

In Table 7.3-7 we can see that both criteria identify the most significant pairs A:1, B:0 and C:2 but in a different order. This means that these pairs have many areas in common and, at the same time, the amount of areas in common represent an important percentage of the areas contained in both of the two clusters crossed in each pair. The other pair is E:3 but it has low significance in both criteria so it can be dismissed.

As the table has more rows than columns there will be some basic clusters without pairing and all the crosses of the two clustering will be shown in Figure 7.3-13. We can see the three main pairs represent three zones clearly defined for both methods but now, as cluster 3 covers a large part of the north, the cluster A∩3 is now and interesting cluster to be taken into account.

With the min criterion we can say that pair B:0 has a high frequency in both clusters but for the next pair in significance that frequency goes down giving a medium significance (about 50%). As we can see in the table, even when the value of the cell is high, other cells in the same column and row have values high enough to make the frequency lower. This is even more accurate in pair A:1 where the value in cell is even higher than the one for C:2 but the presence of that many areas in A∩3 makes this pair to be less significant for the min criterion.

We see other crosses have less interest because they are small clusters and also the pair E:3 has a low interpretation based on the fact that this clusters represents one of the outlying clusters also identified by the trimmed k-means.

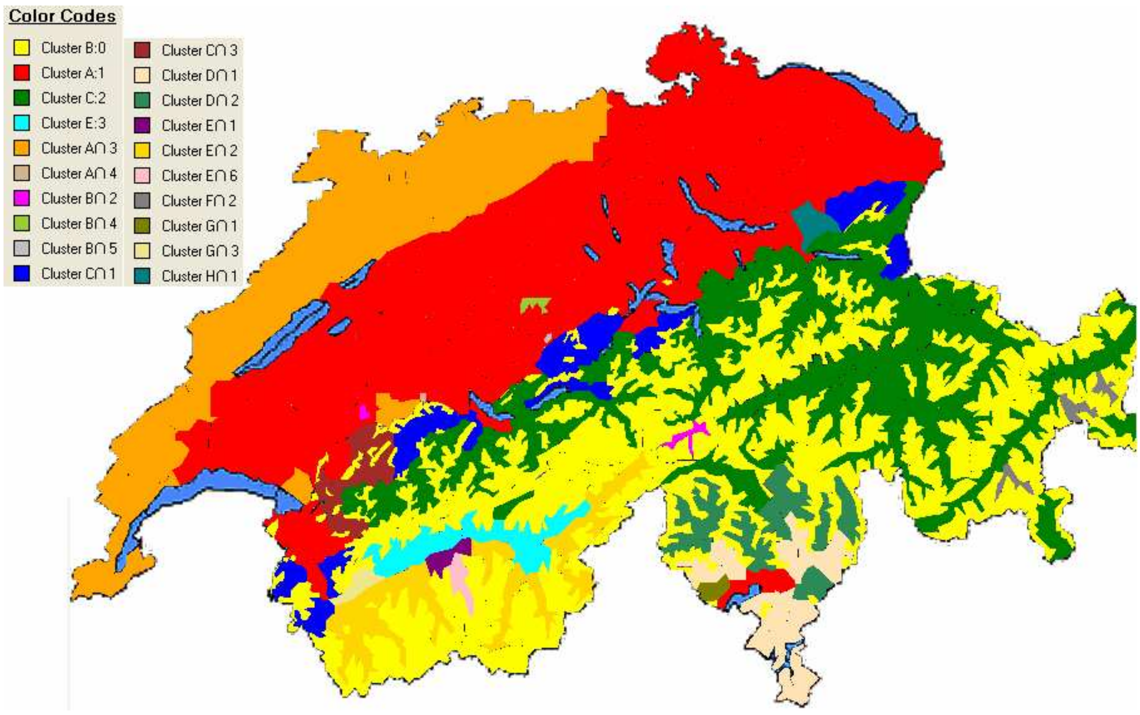


Figure 7.3-13: Map for Naïve Bayes 1 root against Naïve Bayes 2 roots on the transformed dataset

Basing the comparison on the 2 root nodes run we have enough surrogated clusters to assign one of them to every basic cluster. The pairs formed are the same as before because the significance for both criteria does not change. But now their names are different and the analysis is taken from another point of view.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6	7
A	0	194	0	0	0	0	0	0
B	124	0	16	15	1	0	1	1
C	0	2	90	10	17	3	0	0
D	75	0	5	0	6	0	1	0
E	1	1	0	0	0	0	0	0
F	0	1	0	0	0	0	0	0
G	0	0	0	0	1	0	0	0

Max Criterion		Min Criterion	
A-1	34.33%	A-1	97.97%
B-0	21.94%	C-2	59.81%
C-2	15.92%	B-0	48.65%
D-0	13.27%	D-0	32.32%
C-4	3.008%	C-4	9.475%
B-2	2.831%	B-3	5.696%
B-3	2.654%	G-4	4.000%
C-3	1.769%	C-3	3.278%
D-4	1.061%	C-5	2.459%

Table 7.3-8: Results for Naïve Bayes 2 roots against Naïve Bayes 1 root on the transformed dataset

Cluster A:1 now is the previous B:0, C:2 is still the same pair, and A:1 now is B:0. This is obvious because now the equivalences table is transposed. The analysis is from the point of view of the 2 roots Naïve results and the pairs formed have a cluster associated with the areas that are in the basic cluster but not in the surrogated. Now also appear three clusters that do not have a pair, these are clusters E, F and G. This happens because they are very small clusters from the crossed clustering and there are few differences among the results of the two root nodes.

The important clusters are again the mountain cluster to the south and the two lowlands to the north and center of the land. However, as it was seen in the previous analysis, there is a cluster to the north that acquires importance, D\4. This cluster also represents other areas from the cluster to the northern part of the map that are not in the big cluster to the north in the 2 roots results.

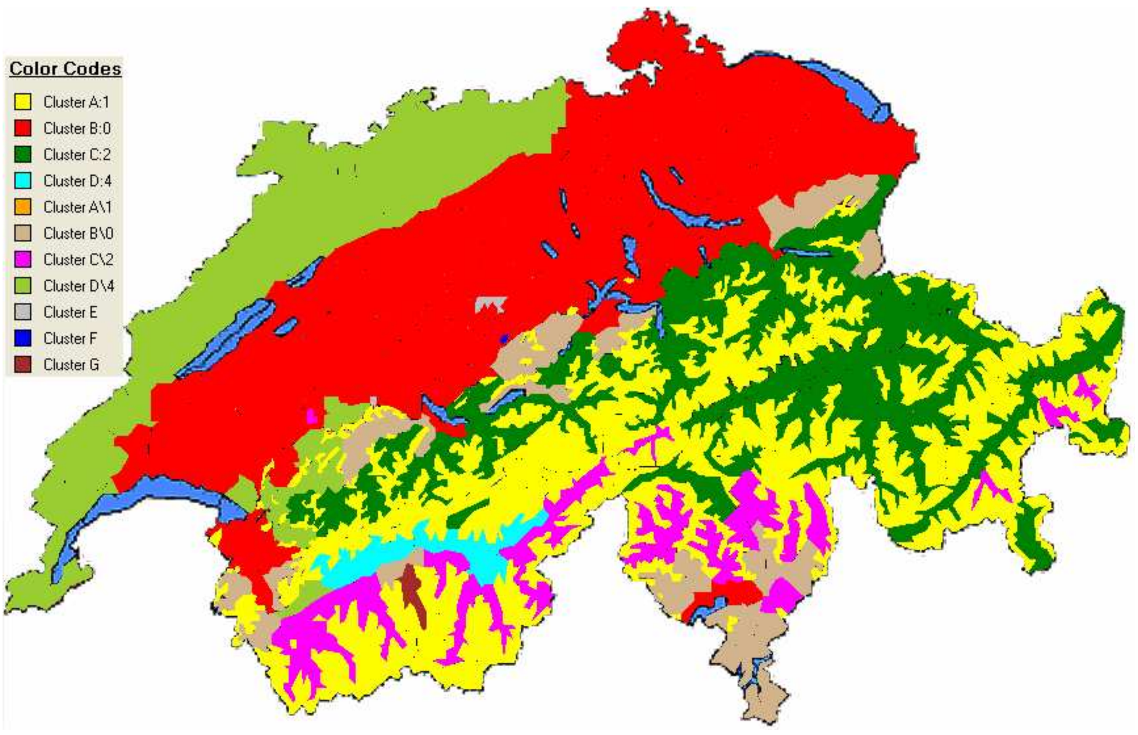


Figure 7.3-14: Map for Naïve Bayes 2 roots against Naïve Bayes 1 root on the transformed dataset

The other small clusters that appear are less significant because they only represent some areas of discrepancy among the two runs.

7.3.3. Naïve Bayes and K-means Comparative

Here we are going to compare the results for the Simple K-Means approach and the Naïve Bayes approach. We compare the Naïve Bayes results with the 5-means clustering firstly with the binary and then with the transformed one. We do only this comparison because the Naïve Bayes approach does not find more than 5 relevant clusters even when it is run with two roots having 4 states in each root.

We compare the clustering results with the two comparison criteria and the analysis is based on both files because different pairs and different resulting maps could be generated.

We begin with the comparison of the binary dataset. For this dataset we can see in Figure 7.3-15 the two maps resulting from applying the k-means and the Naïve approaches. In this figure we can see the southern mountain areas cluster is very similar in both results, and the northern lowlands cluster is equivalent in the two clustering. The main differences are in the lowlands to the south and center where Naïve identifies 3 different clusters whereas k-means only finds 2, and the mountain areas to the north and center that only k-means identifies as a cluster.

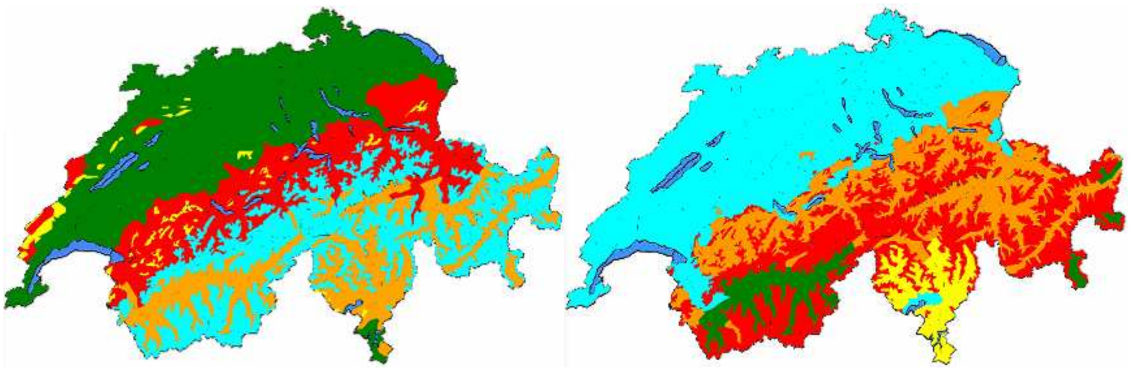


Figure 7.3-15: Results for 5-means [left] and Naïve Bayes 3-4 [right] using the binary dataset

We can see that the k-means method is better for finding small but clearly different clusters, as the northern mountain areas that are clearly different from the lowlands, although it is affected by outliers. Moreover, Naïve is not sensible to very small clusters but it can identify the outlier instances because it is not affected by them because it does not use any distance measure to find the clusters.

Table 7.3-9 shows the results for the comparison of both clustering assignments based on the 5-means results and the significance of each pair using the max and min criterion. Figure 7.3-1 and Figure 7.3-17 show each other the results for the comparison basing the analysis on the k-means and on the Naïve Bayes respectively. We show the significances for the comparison based on Naïve Bayes in Table 7.3-10.

Equivalences for surrogated clustering

	0	1	2	3	4
A	0	160	0	0	0
B	147	0	0	0	5
C	37	0	69	0	0
D	4	0	37	26	21
E	18	39	2	0	0

Max Criterion		Min Criterion	
A-1	28.31%	A-1	80.40%
B-0	26.01%	B-0	69.01%
C-2	12.21%	C-2	41.58%
E-1	6.902%	D-3	29.54%
C-0	6.548%	D-4	19.27%
D-2	6.547%	D-2	14.40%
D-3	4.601%	E-1	12.95%
D-4	3.716%	C-0	6.269%

Table 7.3-9: Results for 5-means against Naïve Bayes 3-4 comparison using the binary dataset

From the table we can see that A:1, B:0 and C:2 are clear equivalences for both criteria. This means in both clustering these three clusters are very stable. This is not strange because the areas that are in A\1 is zero and in B\0 is just 5 so, basing the analysis on the k-means we find that, as the biggest clusters from Naïve Bayes come from clusters with a high probability on both root nodes, they will have a larger size and contain most of the areas from the big k-means clusters. In the opposite way, small clusters from Naïve Bayes, which come from clusters with lower probability, will have even a lower probability and they will be almost contained in k-means clusters.

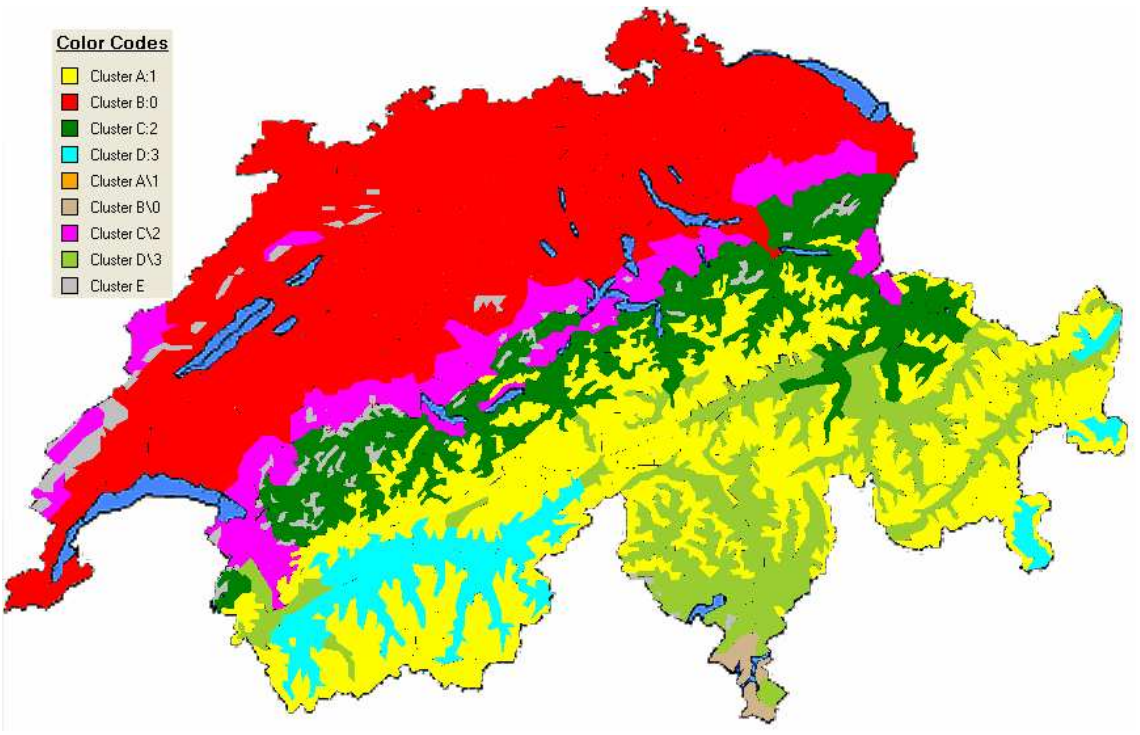


Figure 7.3-16: Map for 5-means (base clustering) and Naïve Bayes 3-4 comparison on the binary dataset

As we can see, there is a cluster from k-means that has not found a pair, cluster E, and it is the one representing the northern mountain areas that Naïve does not find. We can also see one of the outlying clusters that Naïve has found has been paired with cluster D but this is only useful to identify this cluster as a different cluster with a different probability distribution.

Basing the comparison on the Naïve Bayes results we can see that now, big pairs have many areas that are not in the pair as happens with B\0 and A\1. We must point that now the names of the clusters are different because we are basing it on the Naïve.

Equivalences for surrogated clustering

	0	1	2	3	4
A	0	147	37	4	18
B	160	0	0	0	39
C	0	0	69	37	2
D	0	0	0	26	0
E	0	5	0	21	0

Max Criterion		Min Criterion	
B-0	28.31%	B-0	80.40%
A-1	26.01%	A-1	69.01%
C-2	12.21%	C-2	41.58%
B-4	6.902%	D-3	29.54%
A-2	6.548%	E-3	19.27%
C-3	6.547%	C-3	14.40%
D-3	4.601%	B-4	12.95%
E-3	3.716%	A-2	6.269%

Table 7.3-10: Results for Naïve Bayes 3-4 against 5-means comparison using the binary dataset

We can see that B\0 represents mountain areas to the center for the land and that A\1 are several areas, lowlands and mountain areas, which were on the biggest cluster from

Naïve Bayes because of its high probability but in the k-means were identified as another cluster or part of other clusters.

In Figure 7.3-17 we see there are many areas Naïve does not detect as a cluster and also the outlying clusters that are not detected by k-means (D:3 and E). We can see again that the southern mountain area is found without problems by both methods so this is the most compact cluster in the land.

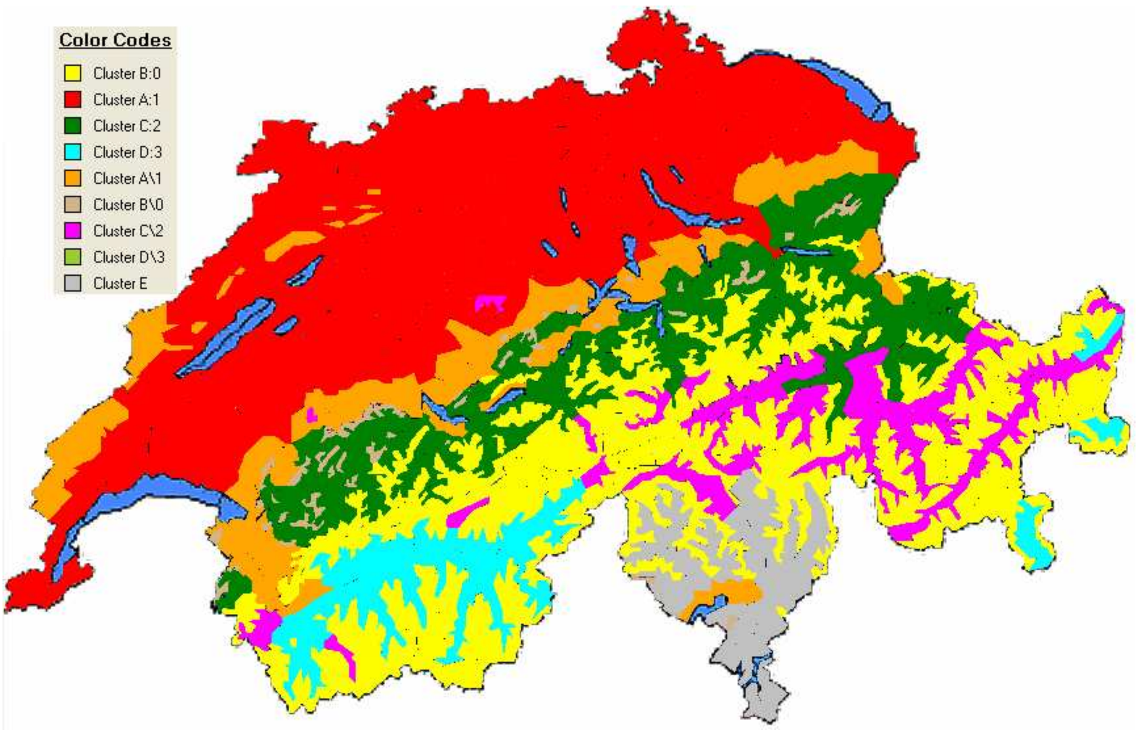


Figure 7.3-17: Map for Naïve Bayes 3-4 (base clustering) and 5-means comparison on the binary dataset

Now we are going to consider the transformed dataset. With it we are going to compare again the simple 5-means with Naïve Bayes results basing the comparison on both methods and applying the two comparison criteria.

The maps for both results are shown in Figure 7.3-18 while the numerical results for the comparison are contemplated in Table 7.3-11 and Table 7.3-12, and graphical results are shown in Figure 7.3-19 and Figure 7.3-20. In Figure 7.3-18 we can see that the differences between the k-means and the Naïve results are very large since now we are using the transformed dataset. K-means finds three clusters to the north that naïve does not because it only discovers the very big one. This happens again because the very big cluster comes from the two big ones with a high probability on the two root nodes from the Naïve model.

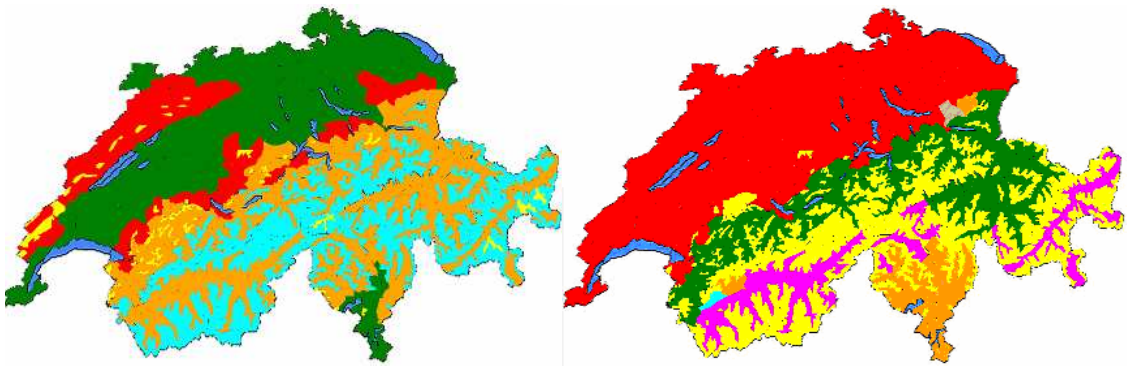


Figure 7.3-18: Results for 5-means [left] and Naïve Bayes 3-4 [right] using the transformed dataset

Again, k-means does the same with the small clusters that Naïve finds to the south. These are three lowlands clusters that in k-means are not identified and they form a unique cluster. Both methods find again the southern mountain areas as an important cluster.

Equivalences for surrogated clustering

	0	1	2	3	4
A	8	0	114	61	17
B	0	157	0	0	41
C	110	0	0	1	0
D	14	0	11	0	0
E	25	0	0	0	0
F	0	0	0	0	3
G	2	0	0	0	0
H	1	0	0	0	0

Max Criterion		Min Criterion	
B-1	27.78%	B-1	79.29%
A-2	20.17%	C-0	68.13%
C-0	19.46%	A-2	51.98%
A-3	10.79%	A-3	30.00%
B-4	7.256%	E-0	15.62%
E-0	4.424%	B-4	13.91%
A-4	3.008%	F-4	4.918%
D-0	2.477%	D-0	4.900%
D-2	1.946%	D-2	3.872%
A-0	1.415%	A-4	2.368%

Table 7.3-11: Results for Naïve Bayes 3-4 against 5-means comparison using the transformed dataset

In Table 7.3-11 we can see that both criteria identify as most significant pairs A:2, B:1 and C:0 but in a different order. This means these pairs have many areas in common and, at the same time, the amount of areas in common represent an important percentage of the areas contained in both of the two clusters crossed in each pair. The other pair, F:4, has low significance in both criteria so it can be dismissed.

As the table has more rows than columns for the comparison based on the Naïve results there will be some basic clusters without pair and all the crosses of the two clustering will be shown on the map.

We can see that the three main pairs represent three zones clearly defined for both methods. These are two lowland zones one to the north and the other in the middle of the land and the huge mountain cluster to the south that always is identified.

In Figure 7.3-19 we can also see the three different areas that are in cluster A but not in cluster 2 which are found by k-means because the abundance of the plants makes them different. We also see in B∩4 mountain areas to the center that Naïve groups with the southern mountain areas and k-means groups apart from this cluster. This means that Naïve finds more general characteristics within the data and k-means looks for concrete differences among the instances.

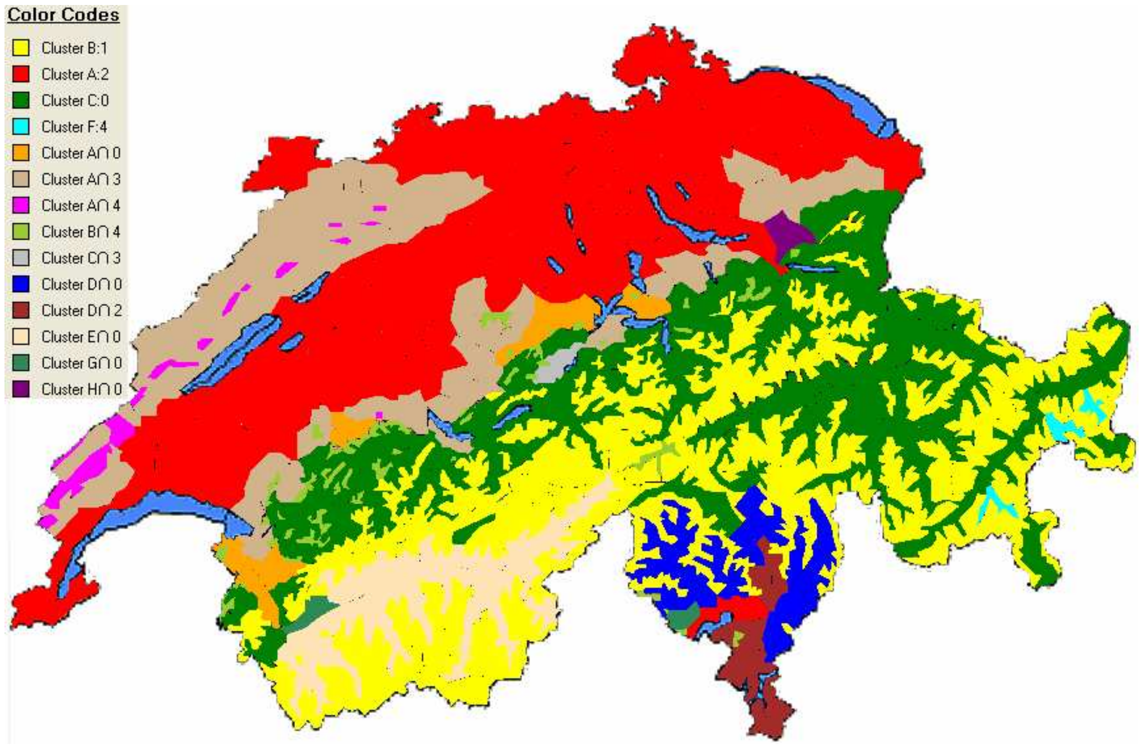


Figure 7.3-19: Map for Naïve Bayes 3-4 against 5-means comparison using the transformed dataset

Other pairs are less important except E and D which are the ones in the Naïve results that identify the southern lowland areas as two different clusters, but that in k-means are not interpreted as that and they are clustered together with the rest of lowlands to the center-south of Switzerland.

As we found about that these areas are outliers when the trimmed k-means is applied we can interpret this as an important difference between the k-means and the Naïve; this is that Naïve finds dissimilarities within the data that k-means does not see because it only takes into account the distances among the instances. This is why k-means has problems to find outlying clusters with a wrong k because they use to be close to other existing clusters; whereas Naïve identifies those outlying clusters because they have particularities that make them different.

Basing the comparison on k-means clustering we have enough surrogated clusters to assign to every basic cluster. The pairs formed are the same as before because the significance for both criteria does not change but now the analysis is taken from another point of view.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6	7
A	8	0	110	14	25	0	2	1
B	0	157	0	0	0	0	0	0
C	114	0	0	11	0	0	0	0
D	61	0	1	0	0	0	0	0
E	17	41	0	0	0	3	0	0

Max Criterion		Min Criterion	
B-1	27.78%	B-1	79.29%
C-0	20.17%	A-2	68.13%
A-2	19.46%	C-0	51.98%
D-0	10.79%	D-0	30.00%
E-1	7.256%	A-4	15.62%
A-4	4.424%	E-1	13.91%
E-0	3.008%	E-5	4.918%
A-3	2.477%	A-3	4.899%

Table 7.3-12: Results for 5-means against Naïve Bayes 3-4 comparison using the transformed dataset

Cluster A represents all the southern lowlands that k-means recognizes as a cluster and, as Naïve does interpret these areas as a group of several clusters A has crosses with many Naïve clusters. These areas that are in A but not in 2 are mainly lowlands to the southern part of Swiss that several times have been identified as outlying clusters. Cluster B again includes the mountain areas to the south, and as it happened with the binary dataset there is no area contained in B outside its pair, cluster 1. Cluster C is the big lowland cluster to the north that always appears as one of the three most significant clusters.

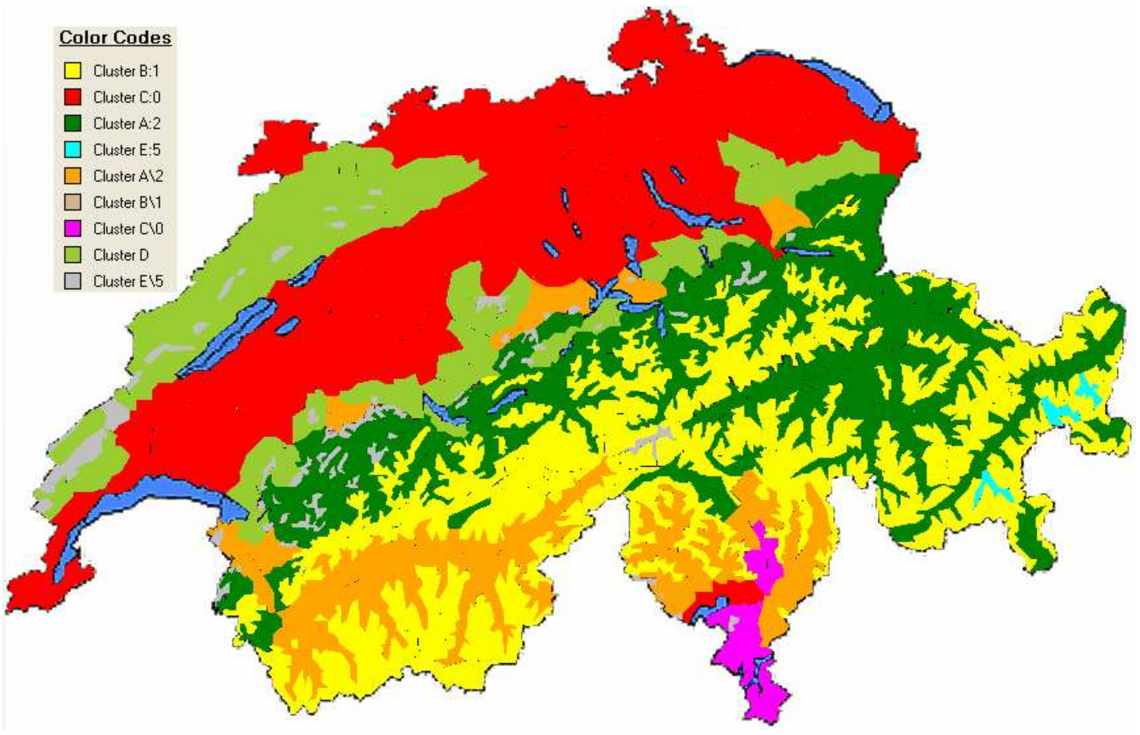


Figure 7.3-20: Map for 5-means against Naïve Bayes 3-4 comparison using the transformed dataset

In the results for this comparison cluster D, containing lowland areas to the north and center of the land, has no pair because it is almost included into cluster 1 and that cluster is assigned to cluster 0, which is bigger. And pair E:5 is less interest because it only represents 3 areas to the eastern part of the map whereas E\5 groups the mountain areas to the north and center that have been identified by the k-means.

The last comparison we are going to treat is the comparison among the Naïve Bayes with one root node and 8 states and the Simple 8-Means with the transformed dataset. We are basing the analysis on the k-means procedure because it is the one that always has 8 clusters.

The maps for both results are shown in Figure 7.3-21 while the numerical results for the comparison are contemplated in Table 7.3-13 and Table 7.3-14, and graphical results are shown in Figure 7.3-22. In Figure 7.3-21 we can see the differences between the k-means and the Naïve results still exist but not so different as other times. K-means finds three clusters to the north whereas Naïve only finds two. Now, Naïve Bayes has found a pattern where the northern lowlands are clustered in two groups as k-means does. However, because of this, the results now do not split the southern lowlands as those with the binary dataset in section 7.3.2.

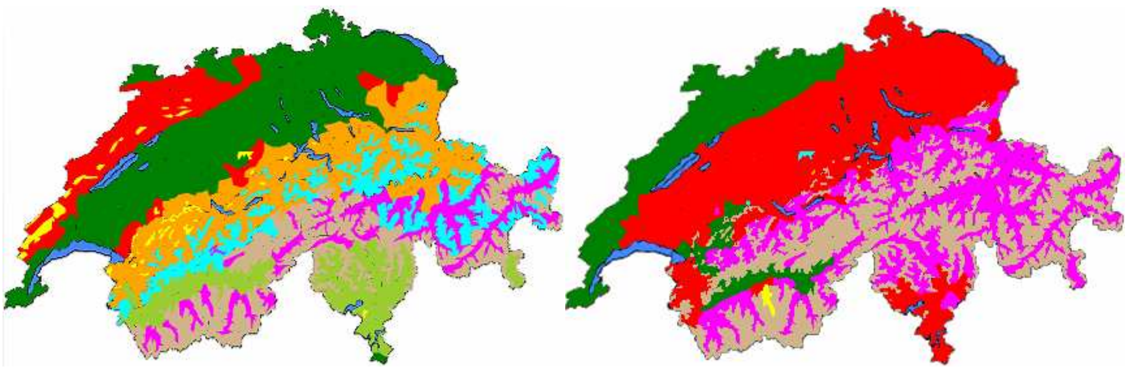


Figure 7.3-21: Results for 8-means [left] and Naïve Bayes 8 [right] using the transformed dataset

Now we can see that k-means has split the mountain areas into clusters to the south where as Naïve only have found two small areas with concrete differences, one in the center of the map and another one in the south-west. Also, Naïve has found as one cluster the lowlands to the south whereas k-means has split that zone into three clusters. One of these three clusters has also been found, in some way, by Naïve because it has not grouped two zones to the south and south-west with the southern lowlands. These areas are again the ones that trimmed k-means identifies as outlying cluster so we can consider them as other two clusters found by Naïve.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6	7
A	0	0	93	77	0	0	24	0
B	107	40	0	0	6	2	0	3
C	0	55	0	0	58	0	1	8
D	12	6	0	0	0	45	17	7
E	0	0	0	0	0	0	2	0
F	0	0	0	0	0	0	1	0
G	0	0	0	0	1	0	0	0

Max Criterion		Min Criterion	
B-0	18.93%	B-0	60.89%
A-2	16.46%	D-5	49.52%
A-3	13.62%	A-2	47.93%
C-4	10.26%	C-4	42.42%
C-1	9.734%	A-3	39.69%
D-5	7.964%	C-1	24.54%
B-1	7.079%	B-1	10.02%
A-6	4.247%	D-6	7.381%
D-6	3.008%	A-6	6.597%

Table 7.3-13: Results for Naïve Bayes 8 against 8-means comparison using the transformed dataset

In Table 7.3-13 we can see the two criteria identify the most significant pairs in a very different way. For the max criterion these pairs are: B:0, A:2, C:4, D:5 and with lower significance E:6. On the other hand, the min criterion identifies B:0, D:5, A:2 and C:4 as the sequence of most significant pairs. From the results of both criteria we can think the significant pairs are going to be very different but, as we have seen, the pairs are again the same.

Basing the comparison on k-means clustering there are no enough surrogated clusters to assign to every basic cluster. The pairs formed are the same as before because the significance for both criteria does not change but now the analysis is taken from a more interesting point of view and because of that we are showing the map with the results.

As the table for this comparison has more rows than columns there will be some basic clusters without pairs and all the crosses of the two clustering will be shown on the map in Figure 7.3-22.

Equivalences for surrogated clustering

	0	1	2	3	4	5	6
A	0	107	0	12	0	0	0
B	0	40	55	6	0	0	0
C	93	0	0	0	0	0	0
D	77	0	0	0	0	0	0
E	0	6	58	0	0	0	1
F	0	2	0	45	0	0	0
G	24	0	1	17	2	1	0
H	0	3	8	7	0	0	0

Max Criterion		Min Criterion	
A-1	18.93%	A-1	60.89%
C-0	16.46%	F-3	49.52%
D-0	13.62%	C-0	47.93%
E-2	10.26%	E-2	42.42%
B-2	9.734%	D-0	39.69%
F-3	7.964%	B-2	24.54%
B-1	7.079%	B-1	10.02%
G-0	4.247%	G-3	7.381%
G-3	3.008%	G-0	6.597%
A-3	2.123%	G-4	4.444%
H-2	1.415%	H-3	3.128%

Table 7.3-14: Results for 8-means against Naïve Bayes 8 comparison using the transformed dataset

Clusters A:1 and F:3 represent the northern lowlands that k-means and Naïve recognizes as clusters. This means that there is a clear separation among the areas in that zone although both methods have discrepancies about the border areas represented by A∩3 and F∩1.

The next significant pair is C:0 but we must also take into account the existence of D∩0 because these two clusters represent the southern mountain areas that the two methods clusters in a different way. This happens because Naïve has found a pattern that conduces it to find the northern split instead of the southern one. As we can see cluster G is crossed with several clusters from the Naïve results. For instance, G∩0 represents the different clustering of the center mountain areas that for k-means are clustered together with the northern ones and for Naïve, as it does not find these northern mountains, they are assigned to the southern cluster.

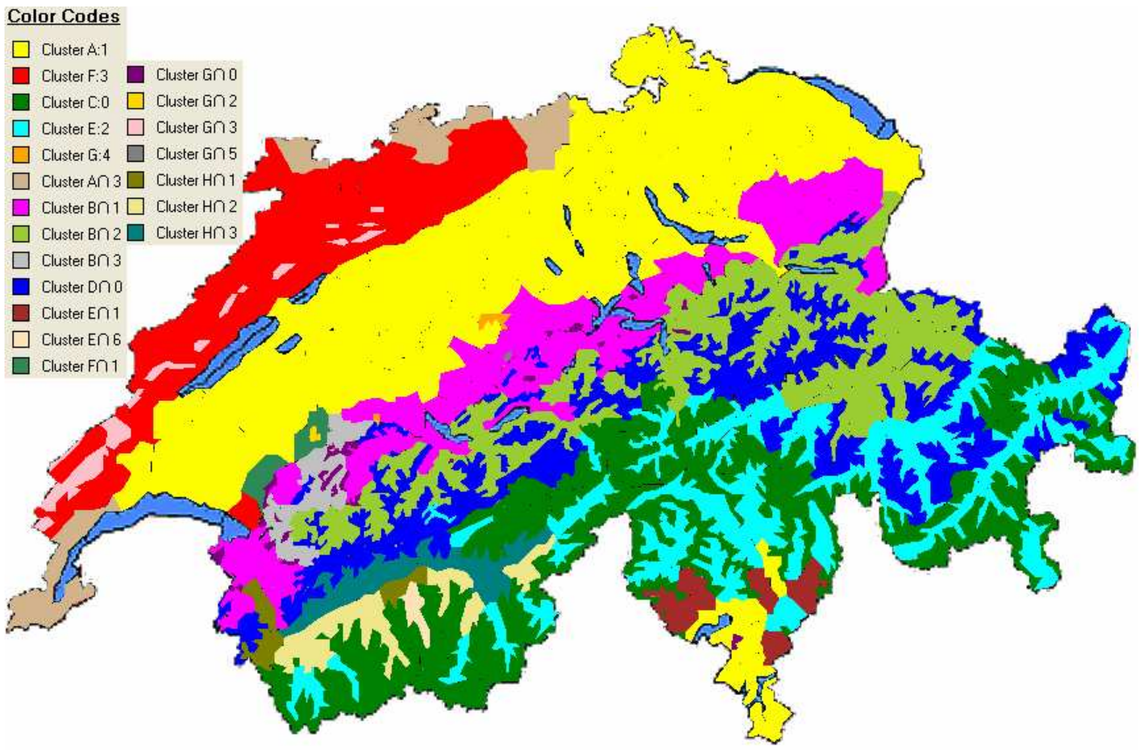


Figure 7.3-22: Map for 5-means against Naïve Bayes 3-4 comparison using the transformed dataset

E:2, B∩2 and B∩1 are clusters that represent the two different lowlands to the center of the land found by k-means that Naïve has classified into the two big clusters one to the north and the other to the south. This means that k-means have more criteria to find small differences than Naïve which only finds the main patterns hidden within the data.

Cluster G is interesting itself because it groups the northern and center mountain areas that Naïve does not recognize except the small area in the middle of the map that this run of the Naïve procedure has identified as a cluster. This is the pair returned by the comparison method and it can be interpreted as an endemic area with plant similar to the mountain areas but with species that does not appear in any other area. This can be a particularly high mountain in the center of the country or something about because it is always clustered as a mountain area by the k-means procedures. The other crosses of cluster G indicates the mountain areas that Naïve has not recognized.

Other pairs are less important except H which is the one in k-means that finds the outlying cluster to the west that trimmed k-means found. The presence of several crosses indicates that Naïve finds some of the areas from this cluster as two clusters different from the others it finds.

After this comparison we can say the k-means approach is better to find the main clusters because it takes into account the distances among instances, which helps to find the different centers that constitute the more different clusters. But the method is affected by the presence of outliers when the value of k is not well chosen.

On the other hand, the Naïve Bayes method is good to find the most different clusters within the data and with its clustering criterion it finds clusters that, in terms the distance measure used by the k-means procedures, are far each other; even outlying clusters, but not concrete small ones as k-means does.

7.4. Conclusions

We have seen the two comparison criteria proposed have a very different behavior but in some cases they return the same equivalent pairs. The max criterion is easier to be understood but it only takes into account the size of the pair. The min criterion computes the frequency of each cell on the two clusters that form the pair and then uses that information to compute a level of significance of each cross.

With the min criterion the pairs with a low relative frequency in one of the clustering, what means that the cross contains few of the total of instances grouped in the cluster, loses significance even if the number of areas in the pair is high. Also, with this criterion we have a significance that can be interpreted in terms of the pairs' size because when the frequency of two pairs is the same the significance is lower for the pair with a small number of areas.

About the k-means approaches, for low values of k the Simple k-Means method can be affected by the presence of outliers; this occurs when the dataset it works with is not much informative as it happens with the binary dataset. In those cases it is a good idea to run once the Trimmed k-Means procedure to identify the outlying clusters and to check if they are causing a miss clustering.

For a right values of k both methods return almost the same results but find a right k is a milestone in these partition-based methods.

For the two Naïve Bayes approaches we must say that the using of two root nodes makes the method faster because with a high number of states in the root node the procedure needs to compute more probabilities. Nevertheless, the method with one root and more states finds different clusters than the two roots network. This happens because the two roots network looks for a few clusters for each root and then crosses the results. If each of the roots finds a different pattern within the data the cross will show both patterns, which does not always happen.

The using of a root node with many states does not guarantee to find all the clusters suggested but the method will look for several clusters following the pattern or patterns found.

The comparison of the two clustering methods, k-means and Naïve Bates, shows each of them has good and bad properties. K-Means is good to find the main clusters because it finds the different centers that constitute the more different clusters. But the method is affected by the presence of outliers when the value of k is not well chosen.

Naïve Bayes is not good for finding those centers but it is good for finding the most different clusters within the data clustering strange areas and even outlying clusters but not concrete small clusters that k-means finds.

With the comparison, we conclude that running only one of the clustering methods available is never a good idea because they all have their pros and cons. So, as far as possible, always it is better to apply several different clustering techniques.

8. Conclusions and Further Work

This chapter concludes our contribution in this project, summarizes our observations on the Swiss plant data, and points out some future work.

8.1. Contribution

The purpose of this project is to analyze the flora dataset containing the abundance information about the plants present in Switzerland. This is an interesting data mining project because, for many geographic regions, extensive data collections exist on the distribution of plant species in the area, and it is interesting to cluster those regions grouping by plant species and amount of each species.

8.1.1. Data Processing

The original dataset contained data from 2697 plants located in 565 areas from Switzerland tabulated as seen below in Table 8.1-1.

	Species name, Flora Helvetica	2697 names
	Nr ecological group	2697 values; range 0-8
	Species Nr, Flora Helvetica	2697 values; range 1-21934
Timberline	Nr. of the mapping area	abundance
565 values; T: below timberline B: above timberline	565 values; range 101-999	565x2697 values; range 2-7

Table 8.1-1: Tabulation of the data contained in the original Excel file.

We transformed the original dataset into a binary dataset (presence-absence) and a rescaled dataset. This transformation is based on some preliminary experiments on the original dataset and have been done because the original dataset has inconsistencies as the values which refer to the presence of plants registered in several years, do not contribute with clear information to the clustering problem.

These two resulting datasets have been the ones used for the different experiments we have carried out.

Finally, the clustering results show the good quality of our data transformation.

8.1.2. System Development

We developed a system to run all the clustering method and do the comparisons of the different clustering results. This is a milestone in this project since we have not found any suitable tool we could use in this project.

We found that the software available was not suitable to make an extensive analysis because the results provided did not allow taking an interpretation based on the position of the areas in the Swiss map. This is why we decided to implement a tool which permits to apply some clustering techniques and show the results on a map to help the user make an easier interpretation.

The application developed was called, CAV-2D (Clustering Analysis and 2D Visualization tool). It has been built under the Visual Studio™ platform and it is compiled for Windows™ systems. The main utility of the program is geographical clustering of the instances of a dataset previously loaded into the application, and represent the resulting clusters in a map. The map used can be changed, but as this project is related to the Swiss flora, the initial map loaded when the program starts is the map of Switzerland.

CAV-2D offers the possibility of applying several clustering methods, and shows the results both in a text format which can be saved in an rtf file, and on a loaded map. The methods supported in the application are Simple K-means, Trimmed K-means, Naïve Bayes and the EM algorithm.

CAV-2D saves the clustering results in an own file format able to be loaded directly to view when necessary or to compare two different clustering results obtained for the same geographical region. The comparison can be made using two different criteria, and the results are also shown in text format and on the map.

8.1.3. Application and Comparison of Clustering Methods

We applied various clustering techniques such as partition-based techniques and probabilistic methods in this project.

One of the partition-based methods applied is the k-means algorithm. This is a moving centers method that finds the best centers for k clusters changing their positions as the method goes. It follows the following steps:

1. Choose the first k centers g_1^0, g_2^0 ... g_k^0.
2. Create the k clusters C_1^{i+1}, C_2^{i+1} ... C_k^{i+1} assigning each point to the class whose center is closer to the point.
3. For each cluster C_j^{i+1} calculate the new centers g_1^{i+1}, g_2^{i+1} ... g_k^{i+1}.
4. Back to step 2 using the new centers to calculate the new clusters.

The method runs until it reaches a fixed number of iterations or when the position of the centers doesn't change significantly. Also, another termination criterion for computational implementations is when the cluster assignments don't change in a concrete iteration.

Another partition-based technique applied was a modification of the k-means algorithm called the trimmed k-means. Given a dataset $\{x_1^0, x_2^0 ... x_n^0\}$ and a trimming level α that represents the percentage of instances trimmed, the algorithm goes as follows:

1. Choose k starting points g_1^0, g_2^0 ... g_k^0 that will serve as seed centers.
2. K-mean step:
   a. Compute the distances of each observation to its nearest center and keep the $\lceil n(1-\alpha) \rceil$ lowest observations.
   b. Create the k clusters C_1^{i+1}, C_2^{i+1} ... C_k^{i+1} assigning each of the kept points to the class whose center is closer to the point.
   c. For each cluster calculate the new centers g_1^{i+1}, g_2^{i+1} ... g_k^{i+1}.
3. Repeat step 2 until a stop condition is fulfilled.

This technique is the basis for a functional analysis made through a graphic called the *k-variogram*. Interpreting this graphic you can find the optimal value for the number of clusters, k, and for the trimming level α.

About the probabilistic methods we applied a simple Bayesian Network called the Naïve Bayes Network that uses a learning procedure called the Expectation Maximization (EM) algorithm. The EM algorithm repeats the following two steeps:

1. Estimation Step (E-step): In this first steep the algorithm calculates an expected value for each missing value using the log-likelihood of the posterior probability of the variable as Bayesian estimator.

2. Maximization Step (M-step): In this steep, the method calculates the new maximum likelihood probability distribution using the values computed on the E-step. Then, the probability distribution is replaced by the new one and iterates.

We also have developed two clustering comparison criteria and called them the *Maximum concordance criterion* (max criterion) and the *Minimum discordance criterion* (min criterion) because the first one looks for the pairs with more areas in common and the second one tries not to leave clusters without pair taking into account the percentage that a pair represents in the two clusters that become a pair, not the size.

The max criterion helps to find the pairs with higher coincidences as equivalent clusters whereas the min criterion finds the pairs that contain a high proportion of the areas contained in the two clusters compared.

With these two criteria, we observed many useful and interesting things from the data. For instance, when the two criteria find the same pair with a high significance we can infer that the pair-cluster is big and stable for both runs. The pair is big in the sense of its size, if the max significance is high, and in the sense of the frequency captured, if the min significance is high.

When the number of clusters of the two results to compare is different the two analyses available return different results because when it is based on the clustering with fewer clusters all them can be paired but when it is based on the clustering with more clusters is impossible to find a pair for all them. This is why for the first case the results are related to the areas in common and not in common for two clusters independently from the cross where the not in common are. The second case cannot be treated this way so all the crosses are shown.

For some results is interesting to analyze the areas that two clusters do not have in common and for others it is interesting to see all the pairs formed.

8.1.4. Other Contributions

During this project we studied the possibility of implementing an own approach to the EM algorithm. Checking several approaches we found two interesting ones that tried to make the algorithm faster, the BC-EM approach [PLL99] and the ECM approach [Sch97].

The BC-EM approach [PLL99] is a method that alternates a Bound and Collapse (BC) step and an Expectation Maximization (EM) step. The BC method [RS98] is as deterministic method to estimate conditional probabilities from incomplete databases. It bounds the set of possible estimates determining a probability interval that is then is collapsed into a unique value.

The ECM algorithm [Sch97] is an extension of the EM where the M-step is replaced by a sequence of N simpler conditional or constrained maximizations known as a CM-step.

Combining these two approaches we have the BC-ECM algorithm but at the moment of writing this thesis the method has not been implemented.

8.2. Conclusions

We based all our conclusions, except the first approach, on the results obtained from the program implemented.

After analyzing several experiments with different methods and different configurations of those methods applied to the available datasets, we obtained a general view of the Swiss flora and the pros and cons of the different clustering techniques.

About the Swiss flora we have found that there is a clear difference among the mountain and lowland areas mainly with the southern mountain areas. Also we observed that the lowland areas are separated into northern and southern lowlands. These are three regions that are recognized by every clustering method so we can infer that they are clearly differentiable because of their flora.

In a further analysis we can see that there are two special regions to the south where the flora configuration has enough differences to be identified by a probabilistic method and only detected as outlying cluster by the trimmed approach of the k-means algorithm. Also we can see that the mountains areas to the south can be split into two different regions, one to the south and the other to the center of the map. This also happens with the northern lowlands region.

We found two depressions, one to south-west of the land and the other to the south-east where there must be special riverside flora because they are identified as separated regions in several experiments.

About the clustering techniques the main conclusion we obtained is that there is no method being the best because all them have pros and cons. Partition based methods are good to find the main groups of instances but they are affected by the initialization and the presence of outliers. On the other hand, probabilistic methods are not influenced by those outliers but they do not have the accuracy of the partition based methods. These probabilistic methods are good to find the main patterns within the data but not concrete differences among instances.

About the k-Means algorithm it is very dependant of the number of clusters (k), and a wrong value of k makes the method return small and low significant clusters, when k is too high or big and maybe wrong clusters if there exists outlying clusters. For this reason is interesting to apply, at least once, the trimmed k-means algorithm. Also with this algorithm we find the miss-clustered instances comparing the results with the simple k-means which gives us a basis to build the k-variogram, a graphic where we can see if the value of k is right and the amount of outliers.

Adding a second root node to the Naïve Bayes network helps the method find different patterns within the data, because in many cases each root gets a different pattern and this is useful when the results are going to be combined. Furthermore, this configuration permits to use a lower number of states for each root node making the learning faster. Moreover, the use of a high number of states makes that the method could find more accurate patterns having enough time to compute.

Related to the datasets we have used, we can say that for partition based methods there is a difference between using the binary or the transformed dataset, because these methods base the clustering on a distance criterion, and in the transformed dataset we have a distance measure that the binary dataset does not have. This difference is not that big in the probabilistic methods because they do not use any distance measure.

As the probabilistic methods learn a probability distribution in the model, the presence of several possible values for each attribute makes decisions with a more solid base; but in our case, the results of using binary or transformed dataset are not very different. This means within the data the patterns are mostly related to the presence and absence of the species in each area.

8.3. Further work

After finishing this project we found several things that deserve for further work or even for another project.

- An open work is to allow datasets coming in Access or Excel format to be loaded. Even it can be contemplated the possibility of allowing the user to introduce manually the data.

- Another interesting work will be to add more clustering techniques and more comparison criteria, including the possibility of a user defined comparator.

- Something very useful will be the possibility of showing the k-variogram graphics to allow the user choosing a right value of k and α, the number of clusters and the trimming level respectively, for the k-means procedures.

- And a very interesting basis for another project is a tool able to recognize the polygons forming each area on a map and save those polygons in our dot file format. Integrating the two applications the scope of the program will cover any geographical clustering problem.

Appendix A
Clustering Techniques

The aim of any Clustering Techniques is to group similar objects using their characteristics. These techniques are used to:

- Get a general descriptive idea of the data.
- Find natural groups.
- Find patterns in the dataset and build laws or models that can explain them.
- Find classification schemes for plants, animals, diseases, …

There are many classification methods because of the many different objectives of the different fields of the science. Good (1977) classified the existing clustering methods based on 45 yes-no criteria. That makes 2^{45} clustering methods.

1. Hierarchic Methods [Gar05]

The hierarchic methods groups or splits the clusters based on a distance measure between the clusters. The process is repeated making different levels of a tree named dendrogram.

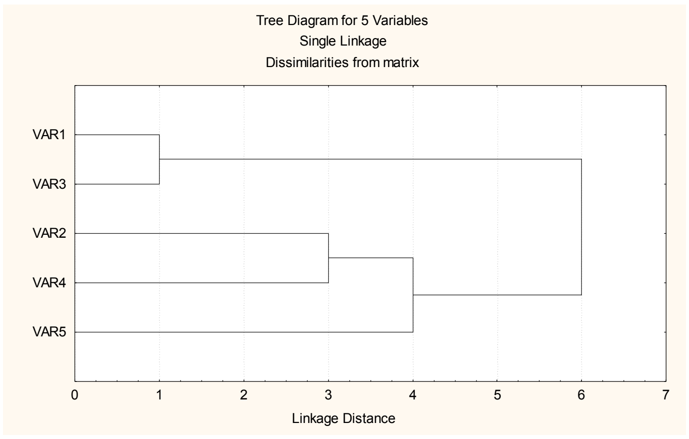


Figure A1-1: Example of a dendrogram for 5 variables

A dendrogram is a tree-form graph in which each vertex represents the union of the two closer classes and the height of that node represents the distance between those classes. Cutting the tree at a determinate height gives the classes within each partition.

There are 2 ways of building a dendrogram that leads to 2 kinds of hierarchic methods: *Agglomerative Methods* and *Divisive Methods*.

Agglomerative Methods (AM) uses the bottom-up approach to build the dendrogram fusing each time the 2 clusters that are closer given a distance measure. A formal definition will be: "Given a set of objects E = {1, 2, …, n} Agglomerative

Methods builds a succession of fitted[18] partitions $\Pi_1, \Pi_2, \ldots, \Pi_n$ in which each set within a partition is as much compact as possible and so each partition has less classes than the previous one".

In AM the first partition is formed by as much classes as the number of objects to classify and the last partition is always a unique class with all the objects in the set E.

On the other hand, Divisive Methods (DM) uses the top-down approach to build the dendrogram splitting each time the cluster with more distance within its objects given a distance measure. The formal definition could be: "Given a set of objects E = {1, 2, …, n} Divisive Methods builds a succession of fitted partitions $\Pi_1, \Pi_2, \ldots, \Pi_n$ in which each set within a partition is as much compact as possible and so each partition has more classes than the previous one". The first partition will always have a unique class, the set E, and the last partition will have n classes, the n object in the set E.

DM begins with all the information available and the process can be stopped anytime we get the number of clusters we need. Processing times for DM are in order of number of attributes squared and for AM are in order of number of instances. But the first step of DM can be taken in $2^{n-1}-1$ different ways and that is a huge requirement of memory when n is big.

1.1. Distance Measures

To measure the distance between the groups we use an *Aggregation Index*. This index defines a measure of the dissimilarity between objects that is used on the dendrogram as height of the corresponding vertex. Some usual Aggregation indexes are:

- **Single linkage**: Use the inferior of all the distances between the members of one cluster and the members of other cluster. This index has 2 undesired effects, as a cluster grows its distance to other objects is lower so it will ate all the point quickly. On the other hand, there could be a chain of close points that make the method add farther points to the cluster instead others that could be closer.

- **Complete linkage**: Use the superior of all the distances between the members of one cluster and the members of other cluster. So this method joins two clusters only if all the points are close each other.

- **Average linkage**: This method uses the average of the distances between the members of one cluster and the members of other clusters.

- **Gravity center**: The gravity center of a cluster is the mean taking all the points in that cluster. This method uses that gravity center to measure the distances between clusters. The new center will be a weighted mean of the previous gravity centers.

[18] A partition π is fitted into another partition π' if each class of π' is obtained fusing 2 or more classes of π.

- **Median method**: This method also uses the gravity center but now the new center will be the mean of the previous ones.

- **Ward criteria**: Ward criteria use the variability between classes and within classes because lower variability between classes' means better is the partition and higher within classes variability also means better clustering. The way to calculate these two measures is using the squared distances from each point to the gravity center or between the gravity centers of each cluster.

2. Partition-Based Methods

The objective of non-hierarchic methods is to build a partition of the space E into k classes. As in hierarchic methods we need the classes to be compact and different each other.

Non-hierarchic methods need to have a fixed number of classes, k, and a comparison criterion for the quality of the different partitions as the size of each class or the separation between classes.

An inoperative approach will be an Exhaustive Algorithm that consist on searching all the partition possible and then select the best of them. The cost of this kind algorithms is in the order of $k^n/k!$ that for n=100 and k=5 it is and order of 10^{67}. As this approach is unviable it is better to use other algorithms which find a good solution in a reasonable time although the solution won't be the optimal solution.

2.1. K-Means Method [Gar05]

K-Means approach is a moving centers method that finds the best centers for k clusters changing their positions as the method goes. K-Means method is based on the Forgy's algorithm that is described below.

1. Choose the first k centers g_1^0, g_2^0 ... g_k^0 randomly, by hand or using another criterion.
2. Create the k clusters C_1^1, C_2^1 ... C_k^1 assigning each point to the class whose center is closer to the point. The method assigns the point x to the class C_r^1 if x is closer to g_r^0 than to any other center g_i^0.
3. For each cluster C_1^1, C_2^1 ... C_k^1 calculate the new centers g_1^1, g_2^1 ... g_k^1.
4. Back to step 2 using the new centers to calculate the new clusters.

The method continues until it reaches a fixed number of iterations or when the position of the centers doesn't change significantly. Also, another termination criterion for computational implementations is when the cluster assignments don't change in a concrete iteration.

There is no theoretic explanation that ensures that the algorithm will converge to the optimal solution but, in practice we can see that the solution approximates to the optimal and it stabilizes in a few iterations. Also, the solution does not depends on the original centers $g_1^0, g_2^0 \ldots g_k^0$.

Here we have an example of how the algorithm works. It shows the i^th^ step with k=3 and p=2. The centers on the previous step were g_1^{i-1}, g_2^{i-1}, g_3^{i-1} and with those centers there are created the clusters C_1^i, C_2^i, C_3^i. The new centers for those clusters are g_1^i, g_2^i, g_3^i.

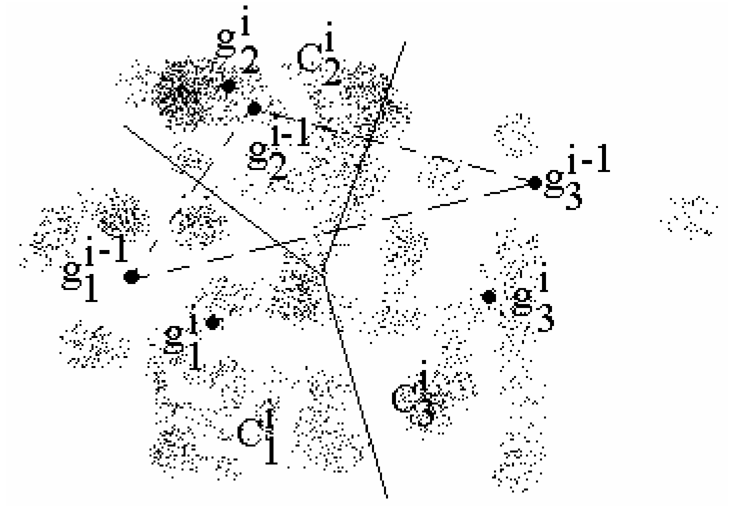


Figure A2-1: Step *i* of the K-Means method with k=3 and p=2.

2.2. Trimmed K-Means Method

Cuesta-Albertos et al. [CGM97] introduced a robust clustering criterion, the trimmed k-means, consisting on the k-means method applied on the observations remaining after removing a fixed proportion of outlying observations. Trimmed k-means constitute the natural extension of the idea of the "impartial trimming" introduced in [Gor91], to the clustering framework [GGM99].

[CGM97] established the existence and a characterization, without moment conditions, of trimmed k-means and proved their consistency for absolutely continuous multivariate distributions having a unique trimmed k-mean. Afterwards, [GGM99] obtained a central limit theorem (CLT) for trimmed k-means under very general conditions: a multivariate framework, a general penalty function, Φ, and every $k \geq 1$.

Trimmed K-Means is a variation of the k-means algorithm that uses a trimmed set (at a given level α) with the lowest possible variation (penalized by the function Φ). That's why in some papers this method is referred to as "Impartially α-Trimmed K-Φ-Means" but the shorter "Trimmed K-Means" will be used.

The method obtains a k-set, if exists, with the minimum k-variation, first into the classes and then between classes. It deletes a number of points in the original dataset to create spherical clusters with minimum variation. For more details of the method see [CGM97] page 554.

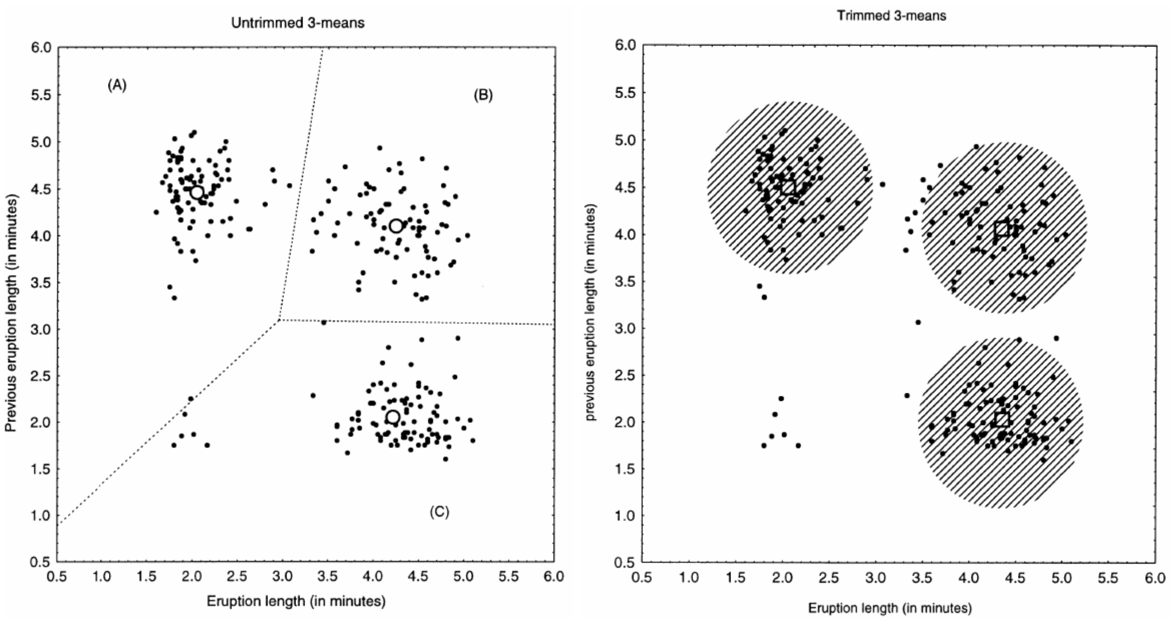


Figure A2-2[19]: [Left] untrimmed 3-mean (circles) and the corresponding clustering allocation. [Right] 0.08 trimmed 3-mean (squares) with the 95% tolerance regions of attaining a 90% coverage.

Given a dataset $\{x_1^0, x_2^0 \ldots x_n^0\}$ and a trimming level α, the trimmed k-means algorithm is as follows:

1. Choose k starting points $g_1^0, g_2^0 \ldots g_k^0$ that will serve as seed centers.
2. K-mean step:
   a. Compute the distances of each observation to its nearest center and keep the $\lceil n(1-\alpha) \rceil$ lowest observations where $\lceil n(1-\alpha) \rceil$ denotes the smallest integer rounding $n(1-\alpha)$ by excess.
   b. Create the k clusters $C_1^{i+1}, C_2^{i+1} \ldots C_k^{i+1}$ assigning each point to the class whose center is closer to the point but only taking into account the points we have kept.
   c. For each cluster calculate the new centers $g_1^{i+1}, g_2^{i+1} \ldots g_k^{i+1}$.
3. Repeat step 2 until a stop condition is fulfilled.
4. Compute the final evaluation function

$$\frac{1}{\lceil n(1-\alpha) \rceil} \sum_{j=1}^{k} \sum_{x_i \in C_j} \left\| x_i - g_j \right\|^2$$

This value is called the *trimmed k-variance* and is the basis for a functional analysis which purpose is to discover the number of clusters and the trimming level using a graphical representation named *k-variogram.*

2.3. Dynamic Clouds Method

Dynamic Clouds method is a generalization of K-Means method proposed by Diday [Did71]. Instead of choosing a point as center of each class and assign the others using the proximity to that center, this method selects a set of h points for each class to be the

[19] Taken from [GCM99]

kernel of the cluster. Then, the method assigns each instance to a class using the proximity to the kernel criteria. Several criteria are listed below:

- Minimum distance to the points of the kernel.
- Distance to the center of the kernel.
- Average distance to the points of the kernel.
- …

Forgy's algorithm for dynamic clouds is described below and an illustration using the minimum distance to the points' criteria with k=2 and h=4 is shown in Figure A2-3.

1. Choose the first k kernels with h points each.
2. Create the k clusters assigning each point using a proximity to the kernel criteria.
3. For each cluster calculate the new kernels inside the class.
4. Back to step 2 using the new kernels to calculate the new clusters.

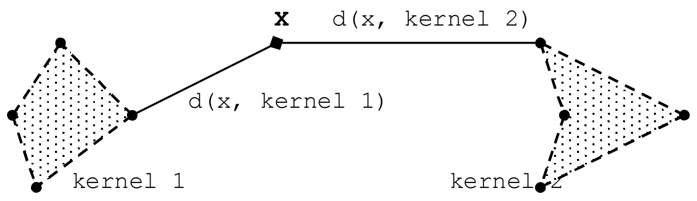


Figure A2-3: Dynamic Clouds method using minimum distance to kernel criteria for k=2 and h=4.

For the dynamic clouds clustering shown in Figure A2-3, the point X will be assigned to the closest point of the ones that compose each of the clusters kernels. Visually, it seems closer to kernel 1 so it will be assigned to Cluster 1.

3. Density-Based Methods

Density-based approaches [Kri00] apply a local cluster criterion. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). These regions may have an arbitrary shape and the points inside a region may be arbitrarily distributed.

The main idea of density-based methods for clustering instances is to identify contiguous regions of high density. It classifies the points in three groups taking into account how many other points are close to the one considered. The method uses a maximum distance ε to the point and a minimum number of neighbors, *k*.

With these two parameters the main algorithm for density-based methods is:

1. Classification step:
    a) Label as *core points*: points with at least k other points within distance ε.
    b) Label as *border points*: points within distance ε of a core point.
    c) Label as *isolated points*: all remaining points.

2. Connection step:
   a) Label as *directly connected points* those within ε distance of each other. Each border point is *directly connected* to one randomly chosen core point within distance ε.
   b) Return as many clusters as *directly connected* relations with at least one core point.

There are several density-based methods with different performances. In the following sections we are describing two of them with a variation for each of them, so four density-based methods are presented.

3.1. DBSCAN

The algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise), based on the formal notion of density-reachability for k-dimensional points, is designed to discover clusters of arbitrary shape.

A generalization of the DBSCAN algorithm is GDBSCAN. GDBSCAN generalizes the notion of point density and therefore it can be applied to objects of arbitrary data type, e.g. 2-dimensional polygons.

3.2. OPTICS

While DBSCAN computes a single level clustering user defined density, the algorithm OPTICS (Ordering Points To Identify the Clustering Structure) represents the intrinsic, hierarchical structure of the data by a (one-dimensional) ordering of the points. The resulting graph (called reachability plot) visualizes clusters of different densities as well as hierarchical clusters.

Based on the same theoretical foundation as OPTICS, LOF (Local Outlier Factors) computes the local outliers of a dataset, i.e. objects that are outliers relative to their surrounding space, by assigning an outlier factor to each object. This outlier factor can be used to rank the objects regarding their outlier-ness. The outlier factors can be computed very efficiently if OPTICS is used to analyze the clustering structure.

Appendix B
Bayesian Networks

1. Introduction

The Bayesian Networks appear as the application of graphical models to specify probability distributions. Basically we can see that a Bayesian Network is an acyclic directed graph, in which the nodes represent the variables of a problem that we want to solve. With this structure we can show the knowledge about the problem from two points of views:

- Qualitative: This point expresses the relationships of dependency or independency between de variables of the problems. It corresponds to the graphical part of the representation. These dependences are represented by the connections in the graph.

- Quantitative: It expresses the belief that we have respect to the relationship expressed in the model. This is represented in the probabilities' tables.

1.1. Definition of Bayesian Networks

A Bayesian Network consists on the following [Jen96]:

- A set of variables and a set of directed edges between variables.
- Each variable has a finite number of mutually exclusive states.
- The variables together with the directed edges form a directed acyclic graph (DAG[20]).
- To each variable A with parents B_1, B_2…, B_n there is attached a conditional probability table with $P(A| B_1, B_2…, B_n)$.

2. Conditional probabilities

As said in the text above, a Bayesian Network is composed by a graph and the conditional probabilities' tables (CPT). On these tables the conditional probabilities of the attributes given a set of evidences are given. To explain it, we are going to begin explaining the concept of probability.

[20] A directed graph is acyclic if there is no directed path $A_1 \rightarrow \dots \rightarrow A_n$ such that $A_1 = A_n$.

2.1. Probability

The probability of an event A, P(A), is a number in the unit interval [0,1] such that it gives the possibility of that event to happen. Probabilities obey the following basic axioms [Jen96].

a) P(A) = 1 if and only if A is certain.
b) If A and B are mutually exclusive, this means independent, then

$$\mathrm{P}(\mathrm{A} \cap B) = P(\mathrm{A}) \cdot P(\mathrm{B})$$

2.2. Conditional probability

The conditional probability is the basis of the Bayesian Network. The concept represents the probability of an event given that the probability of other is X. It can be summarized in one sentence: "*Given the event B, the probability of the event A is x*". This probability is represented as P(A | B) = x.

The conditional probability comes from the update of the probabilities given an event. It is defined from the probability of the intersection of the events.

$$\mathrm{P}(\mathrm{A} \mid \mathrm{B}) = \frac{\mathrm{P}(\mathrm{A} \cap \mathrm{B})}{\mathrm{P}(\mathrm{B})}$$

We can infer now the probability of the joint event as $\mathrm{P}(\mathrm{A} \cap \mathrm{B}) = \mathrm{P}(\mathrm{A} \mid \mathrm{B}) \cdot \mathrm{P}(\mathrm{B})$ and extend the concept to several events:

$$\mathrm{P}(\mathrm{A},\mathrm{B},\mathrm{C}) = \mathrm{P}(\mathrm{A} \cap \mathrm{B} \cap \mathrm{C}) = \mathrm{P}(\mathrm{A} \mid \mathrm{B},\mathrm{C}) \cdot \mathrm{P}(\mathrm{B} \mid \mathrm{C}) \cdot \mathrm{P}(\mathrm{C})$$

Now, from axiom b) we can conclude that, when A and B are mutually exclusive then $\mathrm{P}(\mathrm{A} \mid \mathrm{B}) = \mathrm{P}(\mathrm{A})$. This means that the occurrence of B gives no information about the occurrence of A.

The joint probability has the commutative property, so $\mathrm{P}(\mathrm{A} \cap \mathrm{B}) = \mathrm{P}(\mathrm{B} \cap \mathrm{A})$ and from this we can substitute the probability of the joint event by the conditional probabilities:

$$\mathrm{P}(\mathrm{A} \mid \mathrm{B}) \cdot \mathrm{P}(\mathrm{B}) = \mathrm{P}(\mathrm{B} \mid \mathrm{A}) \cdot \mathrm{P}(\mathrm{A})$$

And from the formula above we can conclude the Bayes' rule:

$$\mathrm{P}(\mathrm{B} \mid \mathrm{A}) = \frac{P(A \mid B)P(B)}{P(A)}$$

Remembering that probabilities should always be conditioned by a context C, the Bayes' rule conditioned on C is:

$$P(B|A,C) = \frac{P(A|B,C)P(B|C)}{P(A|C)}$$

3. Dependency relations

In a Bayesian Network appear relations between the variables represented in the graph. As we said before, these relations are represented by the links between the nodes, which represent the variables. There are three situations in which evidence may be transmitted through variables.

3.1. Serial connection

In this case the relation is A → B → C. This implies that if we have evidence in B, then A and C are independent. In the other hand, if we do not know the state of B, the evidence on A will have influence on the certainty of B. The same occurs in the opposite direction if we have and evidence on C, this will have influence on A through B. This can be represented in a graph on Figure B3-1:

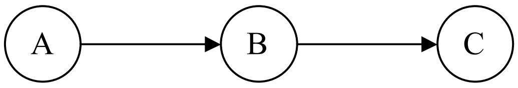


Figure B3-1: A serial connection

We can deduce from this graph that there is independence between A and C through B, $I(A|B|C)$, and there is no independence between A and C if B is unknown, $\neg I(A|\varnothing|C)$.

3.2. Diverging connections

This situation is represented with A ← B → C. For this case we can deduce that if evidence is registered on B, then A and C will be independent but if there is not evidence on B, the evidence on A will have influence in C and the same is in the opposite direction. The graph of the example is in Figure B3-2.

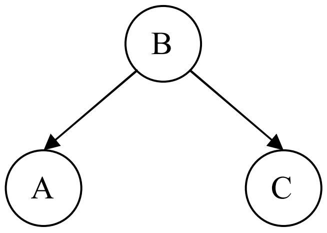


Figure B3-2: A diverging connection

Now the independences deduced are the same as above: $\neg I(A \mid \varnothing \mid C)$ and $I(A \mid B \mid C)$. This can be generalized to any number of children on B node.

3.3. Converging connections

The converging connection occurs when the links are directed to the opposite direction of the previous case, from the leaves to the root node. In this case we have to be more carefully. Now we have the case A → B ← C. For this case if we have evidence on B, the other two variables will be independent, but now if B has evidence, A and C will be dependent through B.

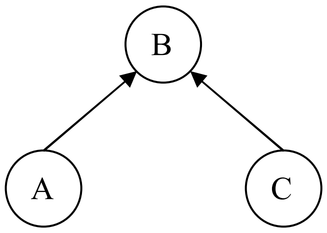


Figure B3-3: A converging connection

That means, in terms of independence, that now we have $I(A \mid \varnothing \mid C)$ and $\neg I(A \mid B \mid C)$.

4. D-Separation

Two variables A and B in a Bayesian Network are d-separated [Jen96] if for all paths between A and B there is an intermediate variable V such that either:

- the connection is serial or diverging and the state of V is known

or

- the connection is converging and neither V nor any of V's descendants have received evidence

Otherwise, we say that A and B are d-connected.

If A and B are d-separated, then changes in the certainty of A have no impact on the certainty on B.

5. Naïve Bayes Network

Naïve Bayes is a practical Bayesian learning method. It consists in a Bayesian Network with a concrete structure. It is oriented to classification. The idea is to suppose that all the attributes are independent known the class variable.

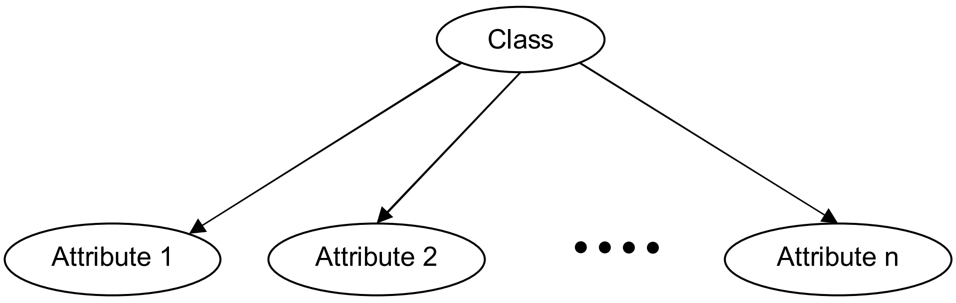


Figure B5-1: The model of a Bayesian Network

This network uses the Bayes' rule for classification:

$$P(C \mid A_1, \ldots, A_n) = \frac{P(A_1, \ldots, A_n \mid C)P(C)}{P(A_1, \ldots, A_n)}$$

This is used to compute posterior probabilities of the class node given evidences on the attribute nodes starting from a random probability distribution for the root node. Most of the times, for this the Uniform Distribution is taken.

5.1. MAP Hypothesis

The MAP hypothesis says that if we want to classify a new case ($a_1 \ldots a_n$) and the class variable has k possible categories $\Omega_c = \{c_1 \ldots c_k\}$, we are interesting in identifying the most probable and return it as classification criterion.

$$\begin{aligned} \mathrm{C}_{\mathrm{MAP}} &= \arg\max_{c \in \Omega_c} P(c \mid a_1, \ldots, a_n) \\ &= \arg\max_{c \in \Omega_c} \frac{P(a_1, \ldots, a_n \mid c)P(c)}{P(a_1, \ldots, a_n)} \\ &= \arg\max_{c \in \Omega_c} P(a_1, \ldots, a_n \mid c)P(c) \end{aligned}$$

The main problem of the MAP hypothesis is that you have to work with join probabilities and that makes this method hard to compute.

The idea of Naïve Bayes simplifies the MAP hypothesis, making it easy to compute, with the assumption that the MAP hypothesis is:

$$\mathrm{C}_{\mathrm{MAP}} = \arg\max_{c \in \Omega_c} \{P(c \mid a_1, \ldots, a_n)\} = \arg\max_{c \in \Omega_c} \left\{ P(c) \prod_{i=1}^{n} P(a_i \mid c) \right\}$$

This is not a very realistic assumption, but the Naïve Bayes algorithm is considered as a standard and its results are competitive against most of the other classifiers.

6. Naïve Bayes' Learning

To estimate the probability La Place's rule is used. This rule assigns as probability of an event the fraction of the times that the event has been observed (n_c) over the total of times the experiment has been repeated (n).

$$P_c = \frac{n_c}{n}$$

This works but it provides poor estimates when n_c is very small. For this reason we can use the Bayesian approach for estimating the probability, using the m-estimate:

$$\frac{n_c + mp}{n + m}$$

Where p is our prior estimate of the probability we wish to determine, m is a constant called the equivalent sample size and n, n_c as we defined before.

6.1. The EM learning

In some cases we do not have a complete dataset because of the presence of missing values or because of perturbations that corrupt the data. For those cases the EM algorithm can be a good solution for the Bayesian Network learning. It can estimate those missing values and check the likelihood of the others assuming that a probability distribution is behind the data. The EM algorithm can be used on variables which have some missing values and on variables which have never been observed.

The EM algorithm repeats the following two steeps:

- Estimation Step (E-step): In this first steep the algorithm calculates an expected value for each missing value using the log-likelihood of the posterior probability of the variable as Bayesian estimator.

- Maximization Step (M-step): In this steep, the method calculates the new maximum likelihood probability distribution using the values computed on the E-step. Then, the probability distribution is replaced by the new one and iterates.

A formal definition of the algorithm is [JN05]:

"Assume that we have a model structure B over the variables U = $\{X_1 \ldots X_n\}$, and let θ_{ijk} denote the parameter corresponding to the conditional probability $P(X_i = k \mid pa(X_i) = j)$, i.e., the conditional probability X_i being in its k^th state given the j^th configuration of the parents of X_i. Using this notation we can find a maximum likelihood estimate $\widehat{\theta}_{ijk}$, for the parameter θ_{ijk} given a data set $D = \{\vec{d}_1, \ldots \vec{d}_m\}$ with m cases as follows":

1. Choose an ε > 0 to regulate the stopping criteria.
2. Let $\theta^0 = \{\theta_{ijk}\}$ be some initial estimate of the parameters (chosen arbitrarily), where:

   $1 \le i \le n$,

   $1 \le k \le |sp(X_i)| - 1$, and

   $1 \le j \le |sp(sp(X_i))|$
3. Set $t = 1$.
4. Repeat:

   - E-steep: For each $1 \le i \le n$ calculate the table of expected counts:

     $$E_{\vec{\theta}^t}[N(X_i, pa(X_i) \mid D] = \sum_{\vec{d} \in D} P(X_i, pa(X_i) \mid \vec{d}, \vec{\theta}^t)$$

   - M-steep: Use the expected counts as if they were actual counts to calculate a new maximum likelihood estimate for all θ_{ijk}:

     $$\widehat{\theta}_{ijk} = \frac{E_{\vec{\theta}^i}[N(X_i = k, pa(X_i) = j \mid D]}{\sum_{k=1}^{|sp(X_i)|} E_{\vec{\theta}^i}[N(X_i = k, pa(X_i) = j \mid D]}$$

   - Set $\vec{\theta}^{t+1} = \widehat{\vec{\theta}}$ and $t = t + 1$

5. Until there is no $\theta_{ijk}{}^t \in \vec{\theta}^t$ s.t. $\left|\theta_{ijk}{}^t - \theta_{ijk}{}^{t-1}\right| > \varepsilon$

Appendix C
Tools

This Appendix is destined to offer general information about the tools used in the realization of the project. These tools are mainly the following:

- `Weka 3.4.7` – The Waikato University's GNU environment that includes several Machine Learning tools.
- `HUGIN Expert 6.4` – Proprietary software to create Bayesian Networks.
- `J2SDK 1.4.2` – The Sun Microsystems' development kit that includes the compiler javac, the documentation generator javadoc, the Java runtime JRE 1.4.2 and other tools used for several purposes.
- `J#` – Programming language similar to Java included in the Visual Studio .NET development environment that allows creating applications in a visual approach.

1. Weka API 3.4.7

Weka is the acronym of **W**aikato **E**nvironment for **K**nowledge **A**nalysis, and it is a collection of state-of-the-art machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. It is implemented in Java released under de GPL (General Public License).

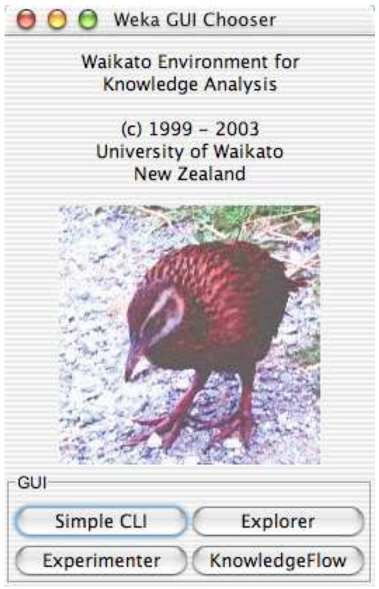


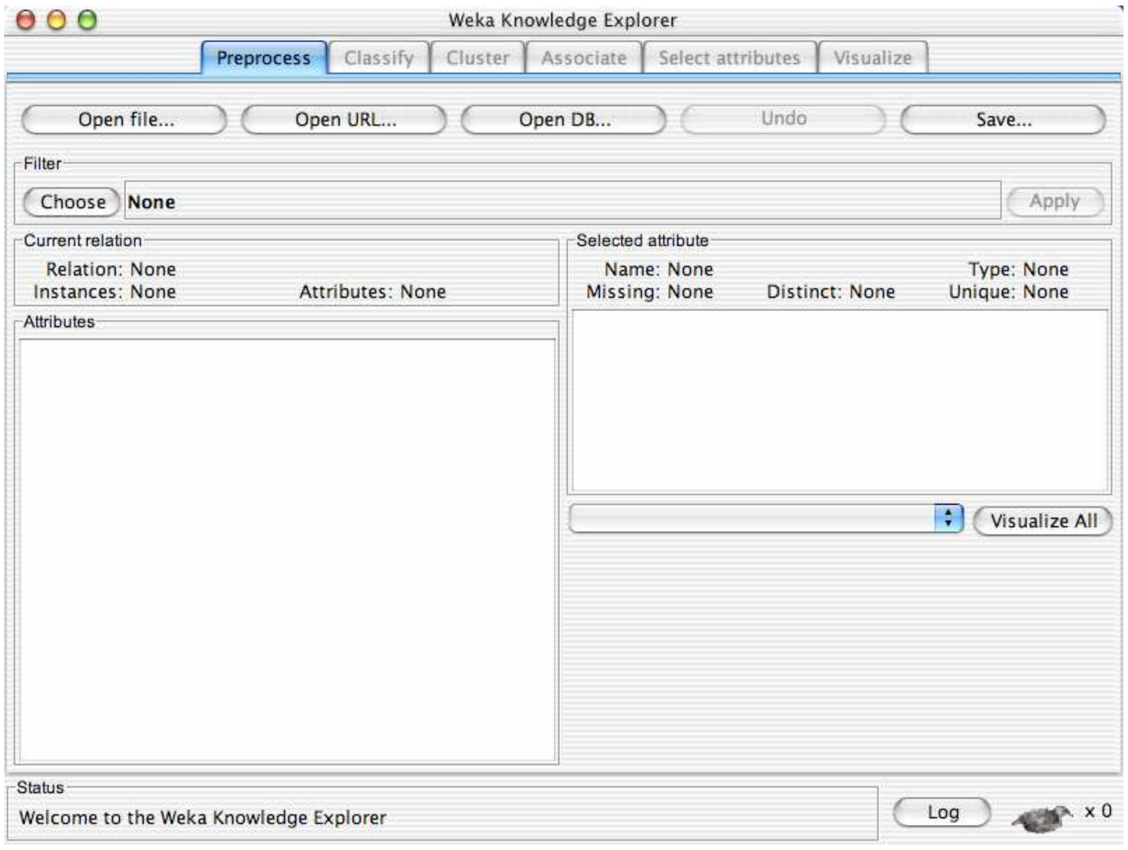
Figure C1-1: Weka GUI Chooser

Weka is designed to support for the whole process of experimental data mining, the preparation of the input data, the statistical evaluation of learning schemes and the visualization of the input data and the result of learning.

The main features are:

- 49 data pre-processing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 15 attribute/subset evaluators + 10 search algorithms for feature selection
- 3 algorithms for finding association rules
- 3 graphical user interfaces
    - "The Explorer" (exploratory data analysis)
    - "The Experimenter" (experimental environment)
    - "The Knowledge Flow" (new process model inspired interface)

You can see in Figure C1-1 and Figure C1-2 the main Weka interface, the GUI chooser, and the Weka explorer where the Machine Learning algorithms are accessible to be applied on a loaded dataset. As we can see, the explorer groups the tools in the 6 features mentioned above.



None

Relation: None
Instances: None
Attributes: None

Name: None
Type: None
Missing: None
Distinct: None
Unique: None

Welcome to the Weka Knowledge Explorer

Figure C1-2: Weka Explorer

2. Hugin Expert API 6.4

The HUGIN API contains a high performance inference engine that can be used as the core of knowledge based systems built using Bayesian belief networks or influence diagrams. A knowledge engineer can build knowledge bases that model the application domain, using probabilistic descriptions of causal relationships in the domain. Given this description, the HUGIN inference engine can perform fast and accurate reasoning.

Licensed to:
KDE3
CS-AAU

Version 6.6

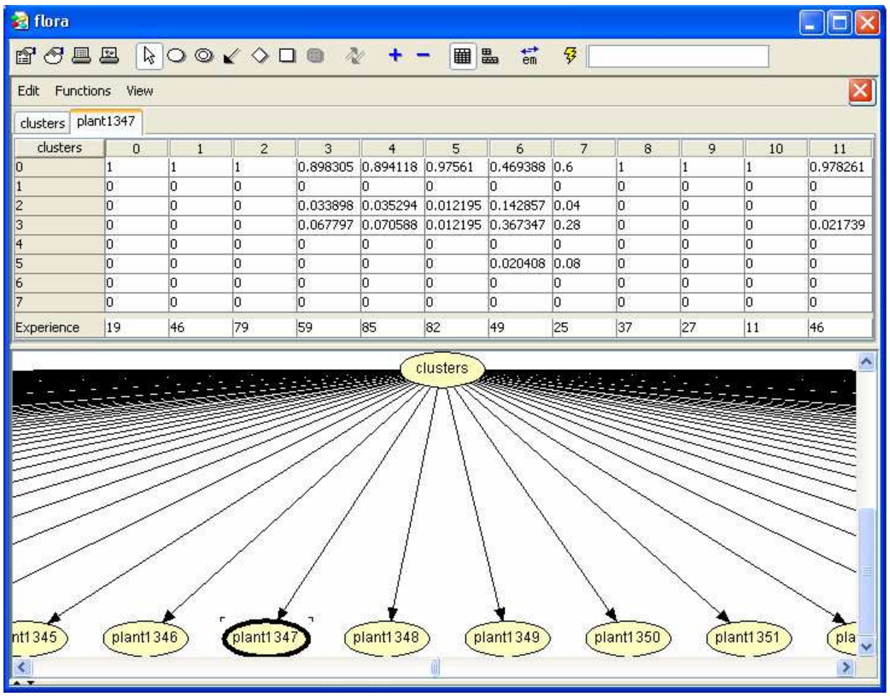
Initializing workspace
Please wait

Copyright © 1995-2005 Hugin Expert A/S

Figure C2-1: The HUGIN Expert 6.6

The HUGIN API is provided in the form of a library that can be linked into applications written using the C, C++, or Java programming languages. The C version provides a traditional function-oriented interface, while the C++ and Java versions provide an object-oriented interface. The present manual describes the C interface. The C++ and Java interfaces are described in online documentation supplied with the respective libraries.

The HUGIN API is used just like any other library. It does not require any special programming techniques or program structures. The HUGIN API does not control your application. Rather, your application controls the HUGIN API by telling it which operations to perform. The HUGIN inference engine sits passive until you engage it.



clusters	0	1	2	3	4	5	6	7	8	9	10	11
0	1	1	1	0.898305	0.894118	0.97561	0.469388	0.6	1	1	1	0.978261
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0.033898	0.035294	0.012195	0.142857	0.04	0	0	0	0
3	0	0	0	0.067797	0.070588	0.012195	0.367347	0.28	0	0	0	0.021739
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0.020408	0.08	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
Experience	19	46	79	59	85	82	49	25	37	27	11	46

Figure C2-2: An example of a Hugin Expert network

3. Java SDK 1.4.2

Java is an object-oriented, platform independent programming language developed in 1991 by a team of Sun Microsystems, Inc. engineers lead by Patrick Naughton and James Gosling. Java was initially intended for use on consumer devices, such as cable TV boxes. Before Java caught a major market share of programming assignments, it had to prove its worthiness.

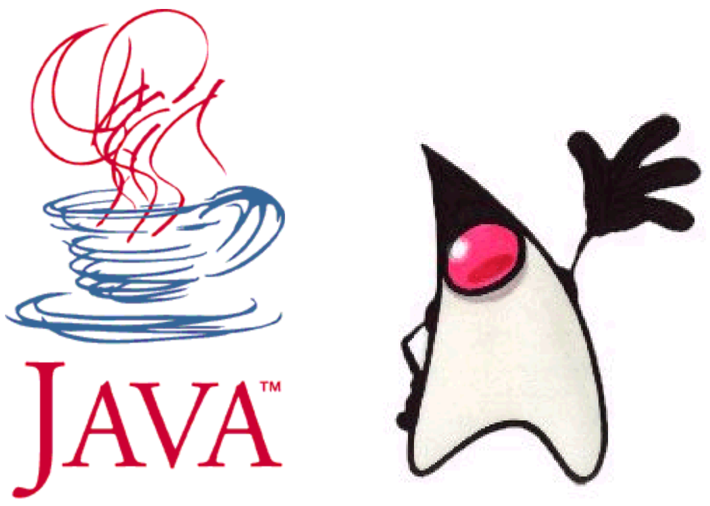


Figure C3-1: The Java logo and its mascot Duke

Java's roots may be found in the "Green Project", which was initially launched by Sun to create smart consumer electronics and from which came the "*7" (a combination PDA/remote control) in 1992. However, the project never got off the ground. Oak, possibly named for the tree outside Gosling's window, was the language used for the "*7". It was discovered that another language went by the Oak moniker, and the creators subsequently changed the name to Java. When the market for smart consumer electronics evaporated, the creators found a new outlet for their "write once, run anywhere" language.

The main features of Java are:

- Platform Independence
- Object Oriented
- Compiler/Interpreter Combo
- Robust
- Several dangerous features of C & C++ eliminated
- Automatic Memory Management
- Security
- Dynamic Binding
- Good Performance
- Threading
- Built-in Networking

4. Microsoft Visual J#

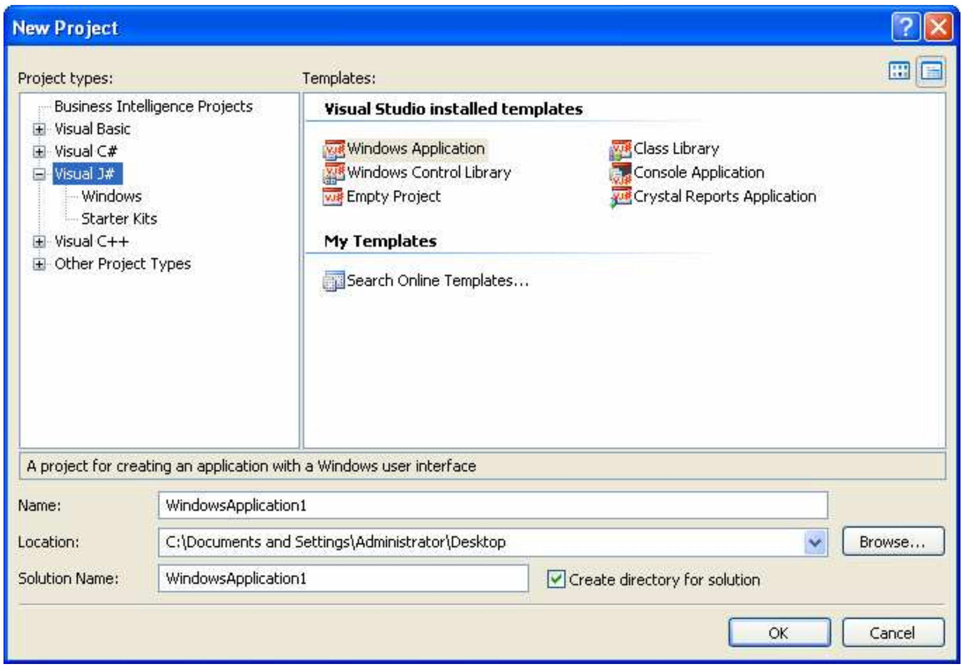
Visual J# is a tool that Java-language programmers can use to build applications and services to run on the .NET Framework.

Visual J# targets the common language runtime (CLR) and can be used to develop .NET Framework applications, including XML Web services and Web applications, making full use of the .NET Framework. Visual J# applications benefit from the following:

- Cross-language integration.
- Enhanced security.
- Versioning and deployment support.
- Debugging and profiling services.

Microsoft Visual J# is a development tool that developers who are familiar with the Java-language syntax can use to build applications and services on the .NET Framework. It integrates the Java-language syntax into the Visual Studio .NET shell. Visual J# also supports the functionality found in Visual J++ 6.0 including Microsoft extensions.

Visual J# is not used for developing applications intended to run on a Java Virtual Machine. Applications and services built with Visual J# will run only on the .NET Framework.



A project for creating an application with a Windows user interface

Name: WindowsApplication1

Location: C:\Documents and Settings\Administrator\Desktop

Solution Name: WindowsApplication1

[✓] Create directory for solution

Figure C4-1: Visual Studio Project creator showing Visual J# templates

Microsoft Visual J# .NET Redistributable Package is the redistributable package for Microsoft Visual J#. The Redistributable Package will only run applications and services developed with Visual J#; Java-language applications written with other Java-language development tools will not run with the Microsoft Visual J# .NET Redistributable Package.

Visual J# and the Microsoft Visual J# .NET Redistributable Package have been independently developed by Microsoft, and are not endorsed or approved by Sun Microsystems, Inc.

Appendix D
User Manual

This appendix is destined to explain how to use the application developed.

1. Introduction

The application developed was called, CAV-2D (Clustering Analysis and 2D Visualization tool). It has been built under the Visual Studio™ platform and it is compiled for Windows™ systems. The main utility of the program is geographical clustering of the instances of a dataset previously loaded into the application, and represent the resulting clusters in a map. The map used can be loaded, but as this application is made to solve the problem with the Swiss flora, the initial map loaded when the program starts is the map of Switzerland.

CAV-2D offers the possibility of applying several clustering methods, and shows the results both in a text format which can be saved in a rtf file, and on a loaded map. The methods supported in the application are two partition-based methods Simple K-means and Trimmed K-means, and two probabilistic methods Naïve Bayes oriented to clustering and EM algorithm. CAV-2D also saves the clustering results in a file and loads them again to view when necessary.

It is possible too, using CAV-2D, to compare two different clustering results obtained for the geographical region. The comparison can be made using two different criteria, and is also shown in the map.

2. Description of the Application

In this section we are going to describe the different interfaces that you can find working with CAV-2D.

2.1. Main Window

When the application is launched appears the main window shown in Figure D2-1. This is a standard user interface with a menu bar at the top followed by a buttons bar, the main panel and the status panel at the bottom of the window. On the left of the main panel there is a zone destined to the color codes of the clusters. In the main panel which we have three tabs where we can choose between the dataset, the results or map. These different tabs are shown in Figure D2-2 and Figure D2-3.

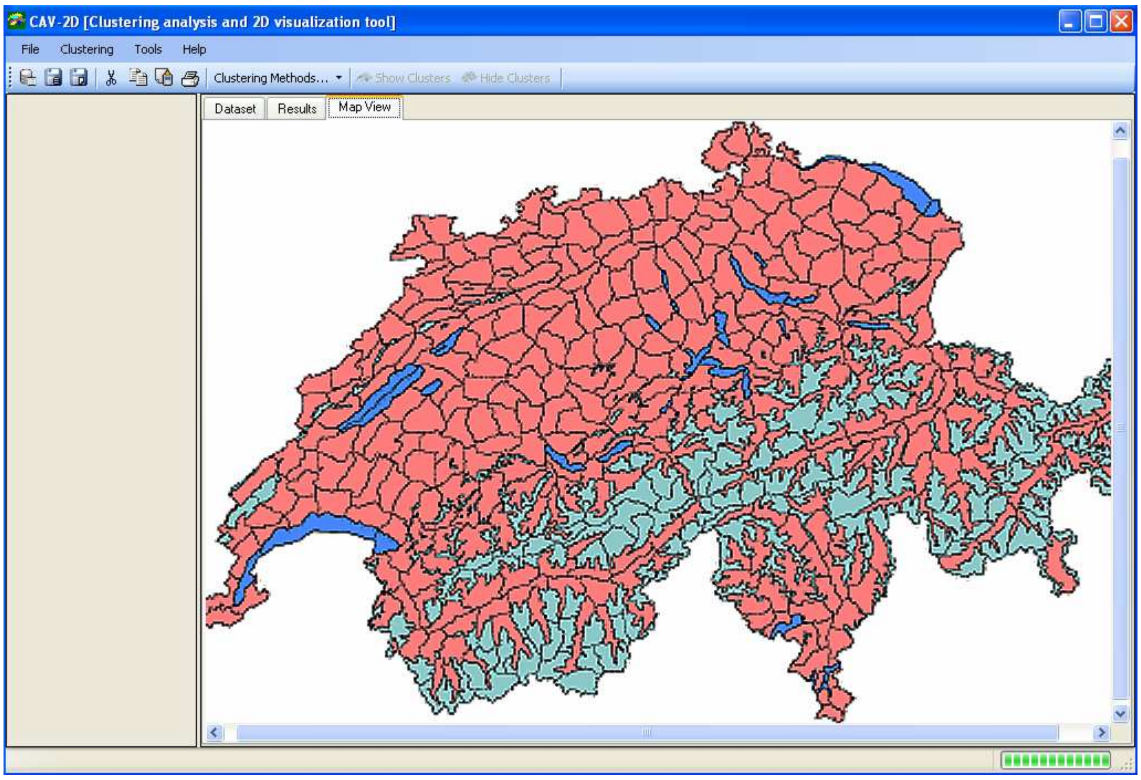
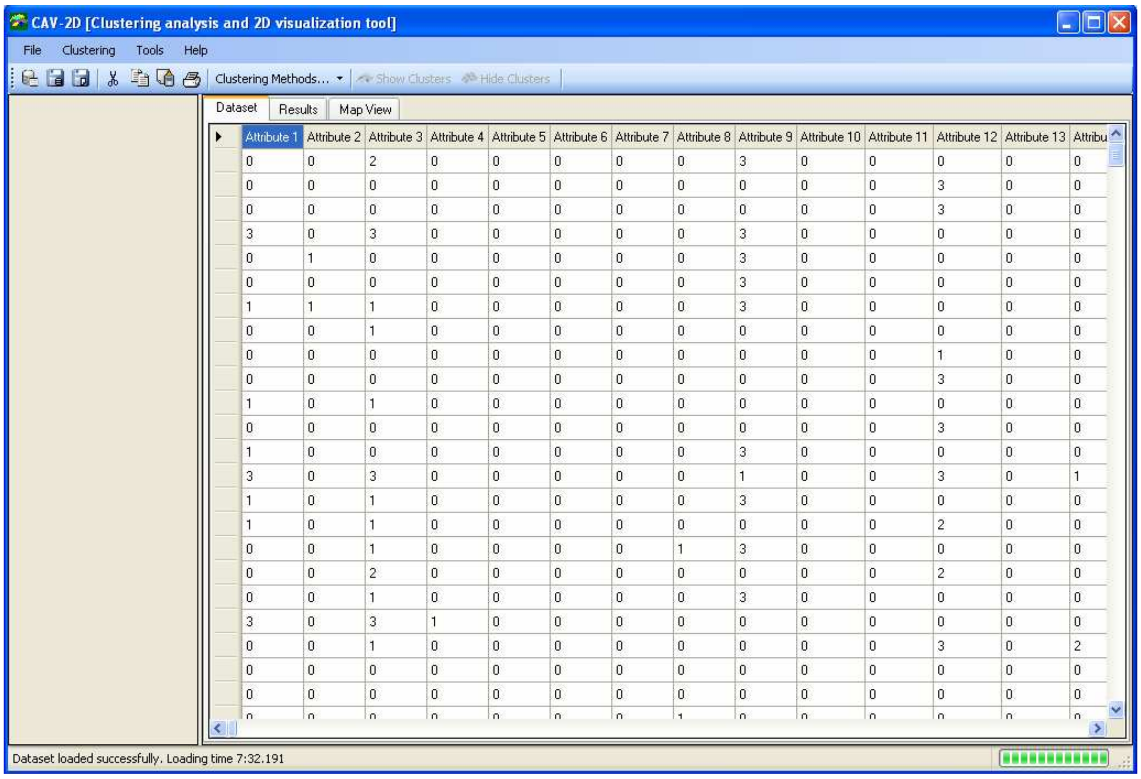


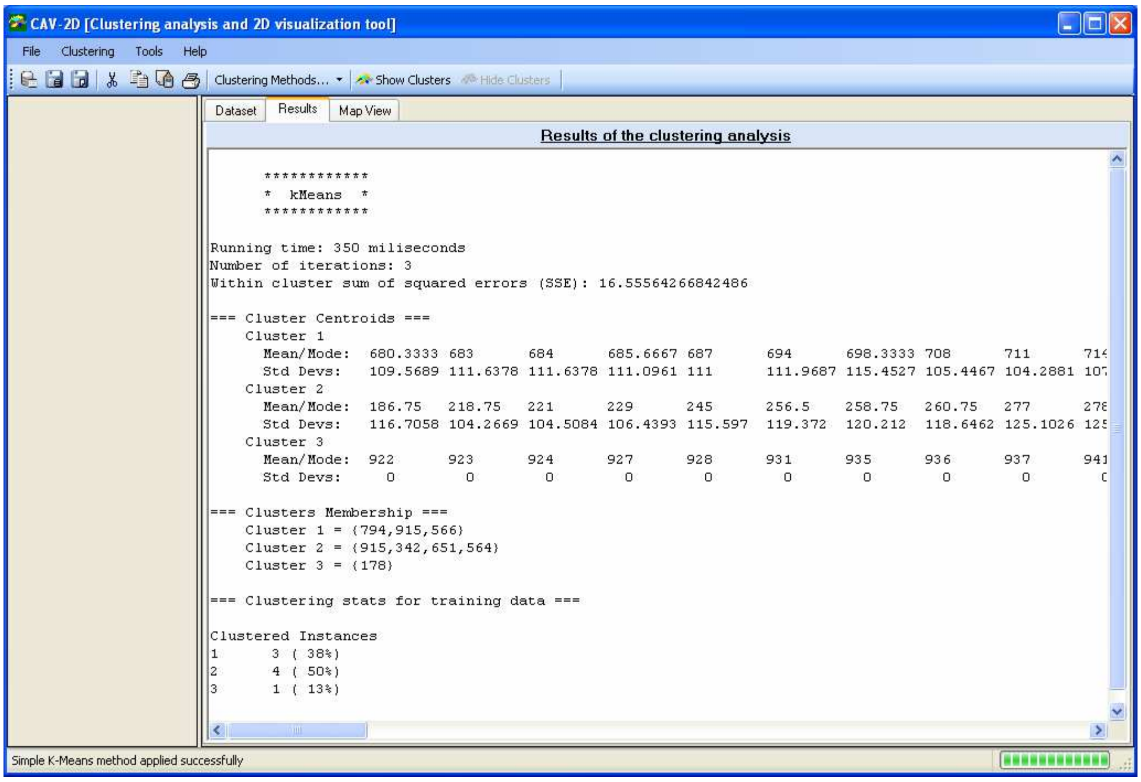
Figure D2-1: CAV-2D's main window. Map tab



Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	Attribute 8	Attribute 9	Attribute 10	Attribute 11	Attribute 12	Attribute 13	Attribu
0	0	2	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0
3	0	3	0	0	0	0	0	3	0	0	0	0	0
0	1	0	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	3	0	0	0	0	0
1	1	1	0	0	0	0	0	3	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0
1	0	0	0	0	0	0	0	3	0	0	0	0	0
3	0	3	0	0	0	0	0	1	0	0	3	0	1
1	0	1	0	0	0	0	0	3	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	2	0	0
0	0	1	0	0	0	0	1	3	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0	0	2	0	0
0	0	1	0	0	0	0	0	3	0	0	0	0	0
3	0	3	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	3	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Dataset loaded successfully. Loading time 7:32.191

Figure D2-2: Dataset Tab



```
      ************
      *  kMeans  *
      ************

Running time: 350 miliseconds
Number of iterations: 3
Within cluster sum of squared errors (SSE): 16.55564266842486

=== Cluster Centroids ===
    Cluster 1
      Mean/Mode:  680.3333 683      684      685.6667 687      694      698.3333 708      711      71
      Std Devs:   109.5689 111.6378 111.6378 111.0961 111      111.9687 115.4527 105.4467 104.2881 10
    Cluster 2
      Mean/Mode:  186.75   218.75   221      229      245      256.5    258.75   260.75   277      27
      Std Devs:   116.7058 104.2669 104.5084 106.4393 115.597  119.372  120.212  118.6462 125.1026 12
    Cluster 3
      Mean/Mode:  922      923      924      927      928      931      935      936      937      941
      Std Devs:     0        0        0        0        0        0        0        0        0

=== Clusters Membership ===
    Cluster 1 = {794,915,566}
    Cluster 2 = {915,342,651,564}
    Cluster 3 = {178}

=== Clustering stats for training data ===

Clustered Instances
1       3 ( 38%)
2       4 ( 50%)
3       1 ( 13%)
```

Simple K-Means method applied successfully

Figure D2-3: Results Tab

2.2. Menu Bar

The menu bar contains the main functionalities of the program separated in 4 different menus: file menu, clustering, tools, and help. These four menus are described in the following sections.

2.2.1. File menu

In Figure D2-4 is shown the file menu. It contains the options to load and save files as well as the printing and the exit option. We are describing each option below.

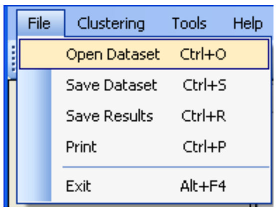


Figure D2-4: File Menu

- **Open dataset**

  With this option the open dataset dialog is shown. The presence of a dataset loaded is necessary to use the different algorithms that have been included in the application. The open dataset dialog is shown in Figure D2-5. In this dialog you can choose the dataset file indicating the application the options of the file you want to load. You can indicate, activating the checkboxes, if the file has the attribute names in the first rows or if the file has a class variable in the last column.

You must indicate too, the separator used in the file for the data. The supported separators appear in the dialog, the application can load files separated by comma, tabulator and space.

Finally you click in open and the dataset will be loaded into the application.

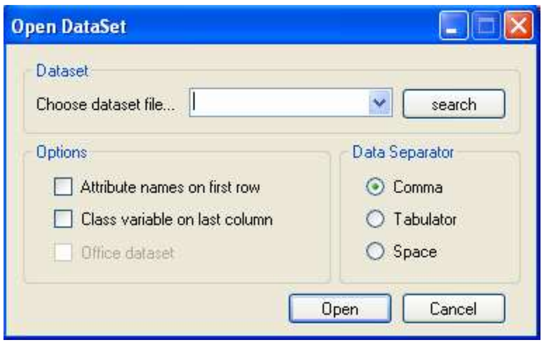


Figure D2-5: Open Dataset Dialog

- **Save dataset**

  Through this option you can save a previously loaded dataset into a file. The dialog that appears is a standard Windows save dialog. The format of the saved file is a CSV (comma separated values) file.

- **Save results**

  The option "Save results" is useful when you have applied a clustering method because it is the one that permits to save the text results into an rtf file. The dialog that appears is the typical save file dialog from Windows.

- **Print**

  Selecting this option appears the typically standard dialog to print the textual results of the results tab.

- **Exit**

  Terminates the program's execution.

2.2.2. Clustering menu

In the clustering menu appears the different clustering methods that can be applied to the dataset loaded before. Figure D2-6 shows the methods that appear in the clustering menu that are also available from a drop down menu in the tool bar.

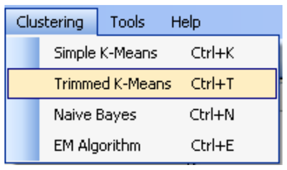


Figure D2-6: Clustering menu

The dialog that appears after select one method is the same for all the methods, but for each one only the parameters appropriated are enabled to be changed.

- **Simple K-Means**

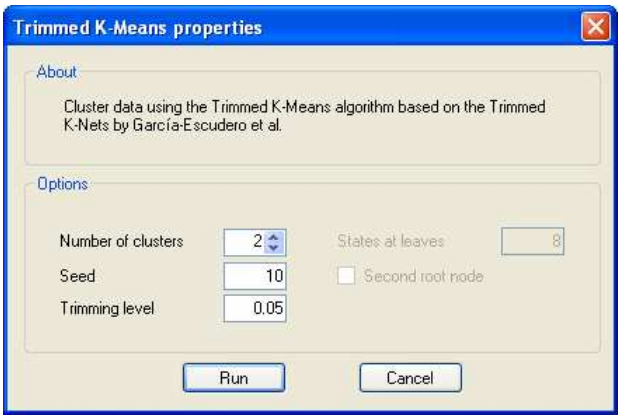
  Selecting this menu appears the dialog to configure the options of the simple k-means method. The algorithm origins from Weka API; however, it is adapted to obtain the cluster memberships. The parameters enabled to change are:

  1. Numbers of clusters: This indicates the number of cluster you want to find into the dataset.
  2. Seed: This parameter is used to generate the random values for the initial centers of the clusters.

- **Trimmed K-Means**

  For this algorithm the parameters to configure are almost the same as the simple k-means. The window dialog which appears is shown in Figure D2-7. In it appears one more parameter enabled:

  1. Trimming level: The trimming level indicates the fraction of the total of instances that you want to trim.



Trimmed K-Means properties

About

Cluster data using the Trimmed K-Means algorithm based on the Trimmed K-Nets by García-Escudero et al.

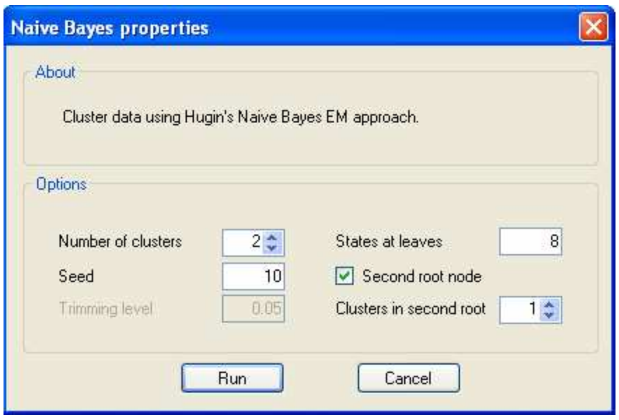
Options

Number of clusters: 2
Seed: 10
Trimming level: 0.05
States at leaves: 8
Second root node

Figure D2-7: Trimmed K-Means properties dialog

- **Naïve Bayes**

  With this option the Hugin's Naïve Bayes EM approach is applied to the data loaded. This method allows choosing to build a Naïve Bayes model with one root node or with two root nodes. The results shown in the map, in the case of the two root nodes, will be the ones obtained for the join of the results of the two root nodes. The results of the separated nodes can be founded in two files generated; these files are indicated in the textual results of the method. The Figure D2-8 shows the option dialog that appears.



Naive Bayes properties

About

Cluster data using Hugin's Naive Bayes EM approach.

Options

Number of clusters: 2
Seed: 10
Trimming level: 0.05
States at leaves: 8
Second root node: [✓]
Clusters in second root: 1

Figure D2-8: Naive Bayes properties dialog

The parameters to configure in this case:

1. Number of clusters: In this case it indicates the number of states for initialize the first root node. That is the number of cluster you are searching for.
2. States at leaves: It represents the number of states that the leaves nodes have. It is related with the different values that the attributes can acquire. It implies that all the attributes must have the same number of states.
3. Second root node: Enabling this checkbox the model generated will have two root nodes and the number of states for the second root will appear.
4. Cluster in second root: It has the same meaning as that in the number of cluster but now it represents the number of states of the second root node.

2.2.3. Tools menu

In this menu there are different interesting tools related with the application. Figure D2-9 shows the contents of the menu.

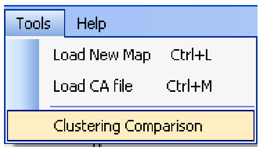


Figure D2-9: Tools menu

- **Load New Map**

This menu permits to load a new map into the application. To load successfully the map it is necessary to have a map image file and a dot file[21] with the points which represent each area of the map. These points are the ones that represent the polygon which forms the areas of the map.

[21] We explain the dot file in section 3.3.

The process to load a new map is: first load the image file with the map; second load the dot file with the point of each area. If one of these steps fails, the change of the map will not have success.

- **Load CA file**

  Loads a CA (Cluster Assignments) file showing in the map the clustering results stored in the file.

- **Clustering Comparison**

  This is an interesting tool that compares two clustering assignments files. As we said in the introduction of the manual it permits to do the comparison of the two clustering using two different criteria. When you select this tool a window dialog as the one shown in Figure D2-10 appears.

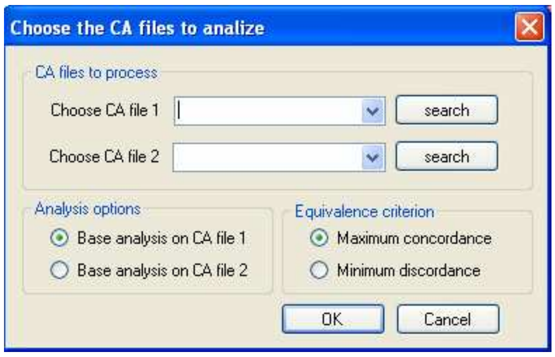


Figure D2-10: Comparison window dialog

In the window dialog you have to choose the two clustering files, the clustering which to base the analysis on and the criterion of comparison. You can choose in which file you want to base the analysis, this is the clustering that the comparison method uses to search for correspondence pairs in the surrogated file.

The equivalence criterion is how the method searches the correspondences between the clusters. There are two different criteria implemented in the application, the *maximum concordance criterion* and the *minimum discordance criterion*. We are explaining both criteria in details in section 4.

2.2.4. Help menu

Finally we have the help menu; it is shown in Figure D2-11.

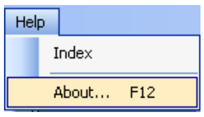
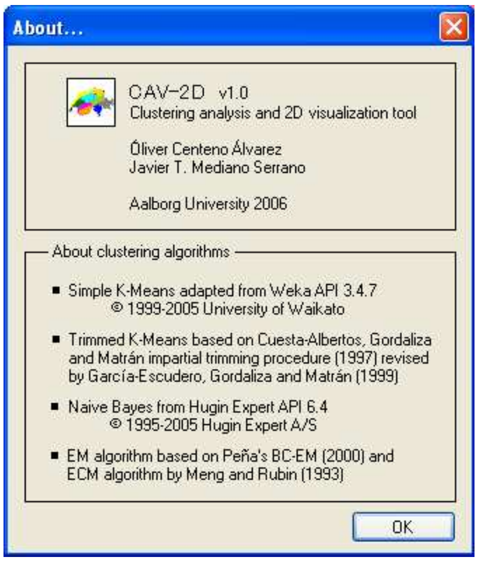


Figure D2-11: Help menu

It only contains two options to choose, one is the index of the help and the other is the information about the program. The about dialog is shown in Figure D2-12.



About...

CAV-2D v1.0
Clustering analysis and 2D visualization tool

Óliver Centeno Álvarez
Javier T. Mediano Serrano

Aalborg University 2006

About clustering algorithms

- Simple K-Means adapted from Weka API 3.4.7
  © 1999-2005 University of Waikato
- Trimmed K-Means based on Cuesta-Albertos, Gordaliza and Matrán impartial trimming procedure (1997) revised by García-Escudero, Gordaliza and Matrán (1999)
- Naive Bayes from Hugin Expert API 6.4
  © 1995-2005 Hugin Expert A/S
- EM algorithm based on Peña's BC-EM (2000) and ECM algorithm by Meng and Rubin (1993)

OK

Figure D2-12: About dialog

2.3. *Tool bar*

The tool bar contains quick access buttons to the main tools of the application. The bar is showed in Figure D2-13.

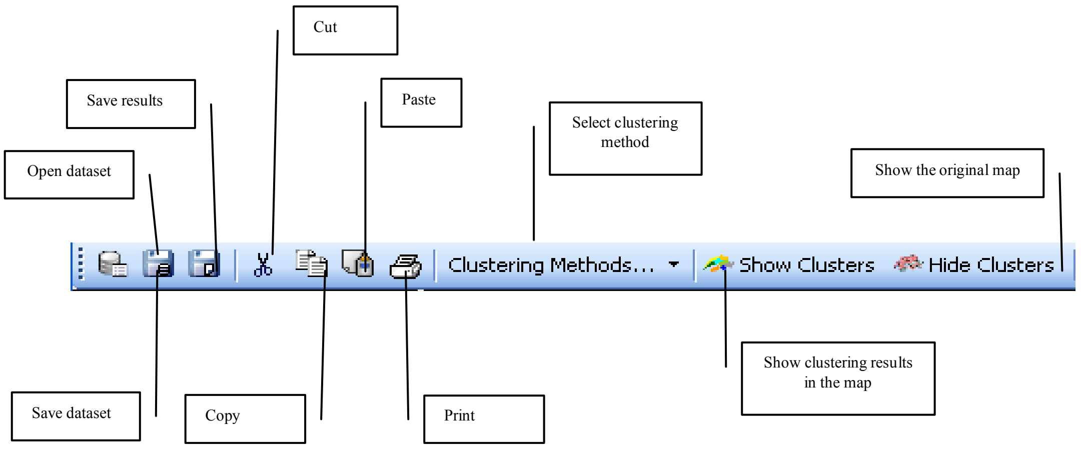


Figure D2-13: Tool bar

3. Files formats supported

The application uses different file formats with different purposes. It supports several dataset formats but imposes a concrete one for the clustering results.

3.1. Dataset File Formats

The dataset can come on several formats because the load dialog permits to specify parameters such as data separator and presence of class variable and attributes names. The following files are supported:

- Data files (*.dat)
- Comma separated values files (*.csv)
- Generic ASCII files (*.txt)
- Tabbed files (*.tab)

Also a generic flat file can be loaded if it fulfils the only rule of having as separator one of the three permitted: comma, tabulator and space.

The file can also contain, and it must be checked, a first row with the names of the attributes and a last column with the class variable useful as areas identification. It was contemplate the possibility of load an office file (Access or Excel file) because the original dataset comes on an Excel format, but it could not be implemented because of the lack of time. This is one of the proposals as further work in chapter 8.

3.2. Clustering Results File Formats

The results of the clustering analysis are shown two formats. One of them is a text based format that is the one presented as results of the clustering analysis. This information can be saved as rich text in a rich text format file (.rtf). This option is accessible through both the file menu and the toolbar.

The other results file is automatically generated and contains the clustering assignments using as identifier of the instances the class variable without mattering if it comes on the dataset or it has been automatically generated[22]. It is a flat ASCII file that contains, in a CSV format, the instances assigned to each of the clusters found. These files are called clustering assignments files (.ca) and their format and extension are required for further uses of these results as reload of the assignments to view them on the map, or compare two clustering assignments automatically.

The CA files generated will be located in the same directory where the dataset is and they will have the same name of the dataset followed by the method run to create them.

3.3. Map Formats

The last kinds of files the program considers are the formats related to the map where the clustering is showed on. The change of the map requires two kinds of files, an image with the new map and a file with the points that define each of the polygons representing the areas on the map.

[22] The automatically generated class column contains the string “Inst” followed by the number of instance.

The image file can come on one of the standard image file formats where as the file with the polygons must have a concrete structure explained below. The supported image files are:

- Windows Bitmap (*.bmp)
- Jpeg File (*.jpg; *.jpeg)
- Compuserver (*.gif)
- Portable Network Graphics (*.png)

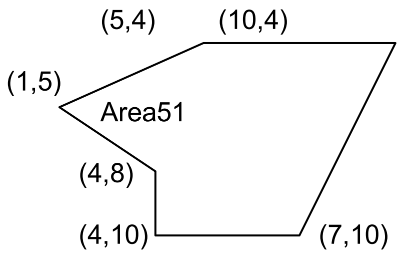
For the polygons file we use talk about the dot file format (.dot) but the program also accepts .csv and .txt files. The format for all them is concretely defined and if the file does not follow it the change of the map will not have success.

The first line of the dot file must contain the amount of areas considered. This is for the initialization of some structures that we are using to store and draw the areas. The following files must have the following information knowing that the point (0, 0) of an image is the top left corner:

<area id>,X_1,Y_1,X_2,Y_2,...,X_k,Y_k

The first value always will be the identification of the area and it must be followed, in a CSV format, by the pairs of point that draw the polygon that covers the area. An example of this format can be seen in Figure 4.3-1.

In the example shown in Figure 4.3-1 we begin with the western point (1, 5) and go around the area opposite to the clockwise direction to (4, 8) and so on until we reach again (1, 5) that must be included again to close the polygon.



"Area51,1,5,4,8,4,10,7,10,10,4,5,4,1,5"

Figure D3-1: Example of an area transformed into a dot file format

4. Clustering Comparison Criteria

4.1. Maximum concordance criterion

The maximum concordance criterion (max criterion) is the simplest criterion available in the application. This criterion treats as similar clusters of the two results the one with the maximum number of areas in common. It just counts the number of areas that are in the two clusters and choose firstly the pair which has more areas in both clusters.

In this criterion the significance is related to the size if the cross related to the whole table, this is, the value of each cell related to the sum of all the values in the table. Said with another words, this method looks for the higher values in the table and sort them descending to select the pairs.

To explain the criterion we are going to show the Example 7.2-1:

Example 4-1. In this example we compare two clustering files, one with five clusters and the other one with two clusters. We are basing the analysis on the five clusters file, so these five clusters are named with the letters A-E. In the same way, the surrogated clustering is named with numbers 0-1.

The Table 7.2-1 is the equivalence table that shows the number of instances that both clustering results have in common. For instance, the number 228 in A0 means that between the cluster A from the base file and the cluster 0 from the surrogated file there are 228 areas in common.

	0	**1**
A	228	117
B	195	1
C	20	0
D	0	3
E	0	1

Table D4-1: Clustering equivalence table

With this equivalence table, the next step is to obtain the most significant pairs applying the criterion, in this case the maximum concordance. The results of the criterion are shown in Table 7.2-2.

Pairs	**Significance**
A-0	40.35%
B-0	34.51%
A-1	20.70%
C-0	3.539%
D-1	0.530%
B-1	0.176%
E-1	0.176%

Table D4-2: Maximum criterion concordance results

The significance obtained for each pair is calculated dividing each cell in the equivalence table by the sum of all cells[23]. This is showed as a percentage to make it easier to interpret. It represents the size of the cross in the table and we can sort the pairs by this value.

The method now selects the pairs with a higher significance identifying them as equivalent clusters. For the max criterion the first pair will be A-0 because it has the highest significance. This means that the highest value in the table is

[23] This is the number of areas that must be the same for both clustering results.

in cell A0 and that crossed-cluster has a 40% of the areas in it. As this criterion only looks for the highest crossed-clusters this is the first the method will identify as a pair.

The next pair with a high value would be B-0 but cluster 0 is already assigned as equivalent to cluster A so this pair is not considered. We go down to the next maximum significance until we get a pair where none of the clusters has been assigned as equivalent to other and we found that the only one available is the pair D-1 that is the last that we can make because the surrogated clustering only has 2 clusters. So with this criterion we obtain that cluster A corresponds to cluster 0, and cluster D corresponds to cluster 1.

4.2. Minimum discordance criterion

The minimum discordance criterion (min criterion) is a more complex criterion than the maximum. The main idea of this criterion is to give more importance to the fraction that each cell represents on the two clusters that are crossed. This means to identify similar clusters using the number of areas that both have in common with respect to the amount of areas of each cluster. This is done because for the maximum concordance criterion the biggest clusters have more significance because they have more areas in common. As that amount of areas in common can be less representative than the areas in common for other pair we thought about another significance criterion.

Example 4-2. To make this criterion easier to be understood we are going to apply it to the equivalence table from Example 7.2-1. First we must build the "Relative Frequency Tables" (RFT) where we compute the proportion that each cell represents in the two clusters crossed on it. We can see that tables for on Table 7.2-3.

	0	1
A	228/443	117/122
B	195/443	1/122
C	20/443	0/122
D	0/443	3/122
E	0/443	1/122
	443	**122**

	0	1	
A	228/345	117/345	**345**
B	195/196	1/196	**196**
C	20/20	0/20	**20**
D	0/3	3/3	**3**
E	0/1	1/1	**1**

Table D4-3: Relative Frequency Tables

The next step is to multiply cell by cell the relative frequency tables to get the significance of each crossed-cluster. The resulting table can be seen in Table 7.2-4.

	0	1
A	34.01%	32.52%
B	43.79%	0.00%
C	4.51%	0.00%
D	0.00%	2.46%
E	0.00%	0.82%

Table D4-4: Minimum discordance significance table

And sorting these percentages we obtain the most significant pairs. Table 7.2-5 shows these significance results of applying the minimum discordance criterion to the equivalences table.

Pairs	Significance
B-0	43.79%
A-0	34.01%
A-1	32.52%
C-0	4.514%
D-1	2.459%
E-1	0.819%
B-1	0.004%

Table D4-5: Minimum discordance criterion results

The significance obtained in this case is calculated as the product of the ratio of areas in common divided by each sum of areas for each cluster in the pair. Once we have the significance calculated, the sequence to obtain the comparison between is the same than in the maximum concordance criterion. With this criterion the results obtained are cluster B corresponds to cluster 0 and cluster A corresponds to cluster 1. This means that pair B0 has a high frequency in both clusters more than to have only a high number of areas in common.

As we can see in the example the clusters correspondences are different depending on the criterion used. This happens because the size of a pair does not mean the pair is representative. The representative of a pair is given by the frequency of the pair in the two clusters that form the pair.

Comparing the two results obtained in the examples above we can see that max criterion gives more importance to A0 because it is the biggest pair but it does not have as high representative ness as pair B0 that, for cluster B contains almost 100% of the areas.

Max Criterion	
Pairs	Significance
A-0	40.35%
B-0	34.51%
A-1	20.70%
C-0	3.539%
D-1	0.530%
B-1	0.176%
E-1	0.176%

Min Criterion	
Pairs	Significance
B-0	43.79%
A-0	34.01%
A-1	32.52%
C-0	4.514%
D-1	2.459%
E-1	0.819%
B-1	0.004%

Table D4-6: Comparison of both criteria results

As max criterion takes the pair A0, the next in significance, B0, is not considered as it happens with A1 and C0 and the next pair chosen is D1 that only has a 0.5% of the areas. On the other hand, the min criterion takes B0 with a medium significance (about 50%) but the second pair chosen is A1 that also has a medium-low (about 30%) significance.

5. Note

The application CAV-2D serves the clustering analysis in Swiss plant data, and provides the foundation of the project involved in this thesis. Nevertheless, it is possible to use it with other kind of data and from other countries for any kind of geographical clustering. The user manual facilitates the using of this application, and will be enriched with more and more follow-up work in this project.

Appendix E
Implementation and Source Code

This appendix is dedicated to show the code of the most interesting functions and classes implemented in the development of the project.

1. Naïve Bayes

The code of the Naïve Bayes network model was implemented in Java, using the Hugin API. The main steeps in the construction of the model are:

1. Create de domain object: This is made calling to the constructor of the Domain class included in the Hugin API.
2. Initialize the domain: This steep is made in the *init* function of the Naïve class. This function uses the *createDCNode* to create all the nodes using the *NumberedDCNode* class constructor of Hugin API, the roots and the leaves. Create the links among them with the *setParent* function of the nodes created and initialize the nodes created with the number of states using the *setNumberOfStates* function for each node created.
3. Compile and load the cases file: This is made with the function of the domain class *compile* and *parseCases*.
4. Initialize the random probabilities of the root nodes; it is made inserting a table of random probabilities for each root node.
5. Apply EM algorithm: The EM algorithm is applied in the *learnTables* function of the domain created.
6. Set evidences: This is made in *setEvidences* function. In this function we read the cases file and for each value of the each case we select the state in the corresponding node using *selectState* function of each node. Once we have all the nodes with one state activated propagate the evidence using *evidenceToPropagate* function of the domain.
7. Search the belief higher: This is made en *setEvidences* function too. It consists in look for all the states of each root node for seeing what state has the belief higher. For do this use the function *getBelief* in the root nodes.
8. Join the result: For do that it called the *join* function, this use the results of the two root nodes and combine them.

The complete code is the following.

```
import COM.hugin.HAPI.*;
import java.util.ListIterator;
import java.awt.geom.Point2D;
import java.io.*;
```

```
public class Naive {

    public Domain domain; //the domain for the network
    private NumberedDCNode nodes[]; //the array of leave nodes
    private String [] names; //the names of the leaves nodes
    private NumberedDCNode root; //the first root node
    private NumberedDCNode root2; //the second root node
    private java.util.Vector asignments, asignments2; //CA results
    private String path, file;
    private int nroot=1; //number of root nodes
    private long ms; //stores the run time
    private String[] key = new String[]{"101"…"999"}; //default areas

    /*
     * Network constructor
     */
    public Naive(String[] args){
      try {
         if(args.length==5){ //there is a file with the key areas
            BufferedReader nk = new BufferedReader(
            new FileReader(args[4]));
            key= nk.readLine().split(",");
            nk.close();
         }
         ms = System.currentTimeMillis(); //get the actual time
         file=args[0];
         path= args[0].substring(0,args[0].lastIndexOf(".")+1);

      // 1. // Create the domain
         domain=new COM.hugin.HAPI.Class(
            new ClassCollection()).createDomain();
         domain.setNodeSize(new Point2D.Double (70, 30));
         asignments = new java.util.Vector();
         asignments2 = new java.util.Vector();

         //Read data file
         BufferedReader in = new BufferedReader(
            newFileReader(args[0]));
         names = in.readLine().split(" ");

      // 2. // Initialize the model
         int l = Integer.parseInt(args[1]);
         int c = Integer.parseInt(args[2]);
         int c2 = Integer.parseInt(args[3]);
         this.init(names.length-1, l, c, c2);

      // 3. // Compile the model
         this.domain.compile();

      // 4. // Randomize the initial probabilities of the root nodes
         root.getTable().setData(rand(root.getTable().getData(), 1));
         root2.getTable().setData(rand(root2.getTable().getData(), 1));
         this.load(args[0]);

      // 5. // apply EM algorithm
         this.domain.setMaxNumberOfEMIterations(100);
         this.domain.learnTables();

      // 6-7. // Set the evidences
         this.setEvidences(in, c, c2);
```

```
   // 8. // Save and join the clustering results
      this.save(1);
      this.save(2);
      this.join();
      this.toString();
   }catch(ExceptionHugin e){
      System.out.println("\tException occured!!\n");
      e.printStackTrace();
   }catch (IOException e){
      System.out.println("Exception reading file");
   }
}

/*
 * 2. Init the naive bayes.
 */
public void init(int num, int states, int clss, int clss2)
      throws ExceptionHugin{

   nodes = new NumberedDCNode[num];
   root=constructNDC(names[num], names[num], clss);
   root.getExperienceTable();
   root.setPosition(new Point2D.Double((num*100)/2, 500));
   root2=constructNDC(names[num] + "2", names[num] + "2", clss2);
   root2.getExperienceTable();
   root2.setPosition(new Point2D.Double(((num*100)/2)+200, 500));

   for(int i=0; i<num; i++){
      nodes[i] = constructNDC(names[i], names[i], states);
      nodes[i].setPosition(new Point2D.Double(i*100, 150));
      nodes[i].addParent(root);
      nodes[i].addParent(root2);
      nodes[i].getExperienceTable();
   }
}

/*
 * Load a data files into the domain
 */
public void load(String file) throws ExceptionHugin{
   domain.parseCases(file, new DefaultClassParseListener());
}

/*
 * Create a Numbered Node
 */
protected NumberedDCNode constructNDC(String label,
   String name, int n){
   try {
      NumberedDCNode node = new NumberedDCNode(domain);
      node.setNumberOfStates(n);
      for (int i = 0; i < n; i++) node.setStateValue(i, i);
      for (int i = 0; i < n; i++) node.setStateLabel(i, ""+i);
      node.setLabel(label);
      node.setName(name);
      return node;
   }catch (ExceptionHugin e) {
      e.printStackTrace();
   }
   return null;
}
```

```
/*
 * 6. introduces the evidences into the network
 */
public void setEvidences(BufferedReader in, int c, int c2)
  throws ExceptionHugin, IOException{

 String readed=in.readLine();
 String [] caso;
 String exceptions="";
 int j=0,k=0;

 // read all the instances
 while(readed != null){
    caso = readed.split(" ");
    // set the evidence to the actual instance
    for(int i=0; i<caso.length-1; i++)
       if(!caso[i].equals("?"))
       nodes[i].selectState(Integer.parseInt(caso[i]));
    // propagate the evidence
    this.domain.adapt();
    if(this.domain.evidenceToPropagate())
    try{this.domain.propagate(Domain.H_EQUILIBRIUM_SUM,
       Domain.H_EVIDENCE_MODE_NORMAL);
    }catch(ExceptionHugin ehug){
       exceptions += (j+1)+",";
    }

    System.gc();
  // 7. // search the states with more belief in the roots
    double maxLH1=0.0, maxLH2=0.0;
    int cluster=0, cluster2=0;
    for(int i=0; i<c; i++)
       if(maxLH1 < root.getBelief(i)){
        maxLH1 = root.getBelief(i);
        cluster = i;
    }
    for(int i=0; i<c2; i++)
       if(maxLH2 < root2.getBelief(i)){
     maxLH2 = root2.getBelief(i);
     cluster2 = i;
      }

    //assing the instance to the cluster given by the state
    asignments.add(j++, new Integer(cluster));
    asignments2.add(k++, new Integer(cluster2));
    this.domain.retractFindings();
    readed=in.readLine();
 }
 if(exceptions.length() != 0)
    System.out.println("\n\tPropagation exceptions @ "
  +exceptions+"\b");
}

/*
 * 8. Obtain the join results of the two root nodes
 */
public void join() throws  ExceptionHugin,IOException{

   String path = file.substring(0,file.lastIndexOf(".")+1);
   PrintWriter out = new PrintWriter(path + "Naive Bayes.ca");
```

```
      String[] output = new String[root.getNumberOfStates()*
      root2.getNumberOfStates()];

      for(int i=0; i< output.length; i++) output[i] = "";
      for(int i=0;i<key.length;i++){
         output[((Integer)asignments.get(i)*
            root2.getNumberOfStates())
            +(Integer)asignments2.get(i)] += key[i]+",";
      }
      for(int i=0; i< output.length; i++)
         if(output[i].length()!=0)
            out.println(output[i].substring(0,
               output[i].length()-1));
      out.close();
   }

    /*
     * Saves the assignments in a txt file
     */
    public void save(int n) throws IOException, ExceptionHugin{

       String[] output= null;
       String path = file.substring(0,file.lastIndexOf(".")+1);
       PrintWriter out = new PrintWriter(path
         + "Naive Bayes Root " + n + ".ca");
       if (n==1) output = new String[root.getNumberOfStates()];
       if (n==2) output = new String[root2.getNumberOfStates()];

       for(int i=0; i< output.length; i++) output[i] = "";
       for(int i=0; i< key.length; i++){
          if(n==1) output[(Integer)asignments.get(i)] += key[i]+",";
          if(n==2) output[(Integer)asignments2.get(i)] += key[i]+",";
       }
       for(int i=0; i< output.length; i++)
         if(output[i].length()!=0)
            out.println(output[i].substring(0,output[i].length()-1));
       out.close();
    }

    /*
     * Generates a random prior distribution
     */
    public double[] rand(double[] rand, int s){
       java.util.Random rnd = new java.util.Random(rand.length*s);
       for(int i=0; i<s; i++){
          double sum = 1.0;
          for(int j=i; j< rand.length-s; j+=s){
             double r=Math.random();
             if((r+ "").length()>=7)
               r=Double.parseDouble((r+ "").substring(0,7));
             if(r>0.5) r /=2;
             while(sum-r < 0.0 || r<0.00001 || r>0.99999) {
               r=Math.random();
               if((r+ "").length()>=7)
                  r=Double.parseDouble((r+ "").substring(0,7));
             }
             rand[j] = r;
             sum -= r;
          }

         double r=Math.random();
      if((r+ "").length()>=7)
         r=Double.parseDouble((r+ "").substring(0,7));
      rand[j] = r;
   }
   return rand;
}

/*
 * Returns the results as text
 */
public String toString()  {

   System.out.println("\t****************");
   System.out.println("\t*  Naive Bayes  *");
   System.out.println("\t****************\n");
   String [] names;
   BufferedReader in=null;
   int n=0;

   try{
     System.out.println("Running time: " +
        getTime(System.currentTimeMillis() - ms));
     System.out.println("Log Likehood after EM "+
        this.domain.getLogLikelihood() + "\n");
   }catch(ExceptionHugin e){
     System.out.println("\tException occured!!\n");
     e.printStackTrace();
   }
   System.out.println("=== Cluster memberships ===\n");
   try{
      in = new BufferedReader(new FileReader(path +
        "Naive Bayes.ca"));
      String reader=in.readLine();
      while(reader!=null){
        System.out.print("Cluster " + n + ": {");
        names = reader.split(" ");
        for(int j=0;j<names.length-1;j++)
           System.out.print(names[j] + ",");
        System.out.print(names[names.length-1] + "}\n\n");
        n++;
        reader=in.readLine();
     }
     System.out.println("Generated files: ");
     System.out.println("\t\"Naive Bayes Root 1.ca\"");
     try{
        if(root.getNumberOfStates()>1)
           System.out.println("\t\"Naive Bayes Root 2.ca\"");
     }catch(ExceptionHugin e){
        System.out.println("\tException occured!!\n");
        e.printStackTrace();
     }
     System.out.println("corresponding to the cluster in each
        root.\nTo open the files use Tool > Load CA file");
     in.close();
  }catch (IOException e){
     System.out.println("Exception reading file");
  }

  return "";
}
```

```
    /*
     * Transforms a long time into a hh:mm:ss.nnnn format
     */
    public String getTime(long ms){
      String time = "";
      if (ms > 3600000)
          time += (ms / 3600000) + " hours, ";
      if (ms > 60000){
          ms = ms % 3600000;
          time += (ms / 60000) + " minutes, ";
      }
      if (ms > 60000){
          ms = ms % 60000;
          time += (ms / 1000) + " seconds and ";
      }
      time += (ms % 1000) + " miliseconds";

      return time;
    }

    /*
     * Create the naive bayes for the swiss flora
     * Use: java Naive <file name> <leaves states> <root1 states>
     *           <root2 states> [areas keys]
     */
    public static void main(String args[]) {
       if(args.length != 4 && args.length != 5)
          System.out.println("Use: java Naive <file name> <leaves
            states> <root1 states> <root2 states> [areas keys]");
       else
          new Naive(args);
   }

}
```

2. Simple K-means

As we said in the report, the simple k-means code is obtained from the Weka API. This is implemented in Java. We modified the code to our necessities. The modifications we made offer the possibility to show the memberships of each cluster and save it as a .ca file. The main algorithm of the simple k-means is the same as in Weka API and the code is showed in this section. We are going to explain the interesting fields used in this fragment of the code.

- `converged`: A Boolean variable used to know when the algorithm has converged. This happens when there are not changes in the clusters assignments in iteration.
- `m_Iteration`: An integer variable which counts the number of iterations taken by the algorithm.
- `clusterAssignments`: This is an array of integer values which have one position for each instance. In it is stored the number of cluster which the instance belongs to.
- `m_ClusterCentroids`: This is a weka.core.Instance object which holds the cluster centers.

- `m_ClusterStdDevs`: Another weka.core.Instance object which holds the standard deviations of each cluster.

Now we can understand the k-means algorithm shown below:

```
  /* K-means algorithm */
//Iterate until the method converges
while (!converged) {
  emptyClusterCount = 0;
  m_Iterations++;   //count the numbers of iterations
  converged = true; //admit the convergence happens
  //go throw all the instances
  for (i = 0; i < instances.numInstances(); i++) {
     Instance toCluster = instances.instance(i); //get the ith
     //Assign the instance to one cluster
     int newC = clusterProcessedInstance(toCluster, true);
     if (newC != clusterAssignments[i])
        //if an instance changes the clustering doesn't converge
        converged = false;
     clusterAssignments[i] = newC; //update the asignments array
  }

  // update centroids
  m_ClusterCentroids = new Instances(instances, m_NumClusters);
  for (i = 0; i < m_NumClusters; i++)
     tempI[i] = new Instances(instances, 0);
  for (i = 0; i < instances.numInstances(); i++)
     tempI[clusterAssignments[i]].add(instances.instance(i));
  for (i = 0; i < m_NumClusters; i++) {
     double [] vals = new double[instances.numAttributes()];
     if (tempI[i].numInstances() == 0)
        emptyClusterCount++; // empty cluster
     else {
        for (int j = 0; j < instances.numAttributes(); j++) {
           vals[j] = tempI[i].meanOrMode(j);
           m_ClusterNominalCounts[i][j] =
              tempI[i].attributeStats(j).nominalCounts;
        }
        m_ClusterCentroids.add(new Instance(1.0, vals));
     }
  }

  //it there are empty clusters it doesn't count them
  if (emptyClusterCount > 0) {
     m_NumClusters -= emptyClusterCount;
     tempI = new Instances[m_NumClusters];
  }
  //if it hasn't converged reset the counts
  if (!converged) {
     m_squaredErrors = new double [m_NumClusters];
     m_ClusterNominalCounts = new int
        [m_NumClusters][instances.numAttributes()][0];
  }
}

/* Calculate the deviations */
m_ClusterStdDevs = new Instances(instances, m_NumClusters);
m_ClusterSizes = new int [m_NumClusters];
for (i = 0; i < m_NumClusters; i++) {
  double [] vals2 = new double[instances.numAttributes()];
```

```
      for (int j = 0; j < instances.numAttributes(); j++) {
         if (instances.attribute(j).isNumeric())
            vals2[j] = Math.sqrt(tempI[i].variance(j));
         else
            vals2[j] = Instance.missingValue();
      }
      m_ClusterStdDevs.add(new Instance(1.0, vals2));
      m_ClusterSizes[i] = tempI[i].numInstances();
    }
  }
```

As we could see, the algorithm uses a method to process an instance that returns the cluster which it is assigned to. This method is used in the algorithm above; it chooses a cluster for each instance. To assign a cluster to one instance it calculates the distance with respect to the centroids of the each cluster.

```
private int clusterProcessedInstance(Instance instance,
boolean updateErrors) {
    double minDist = Integer.MAX_VALUE;
    int bestCluster = 0;
    for (int i = 0; i < m_NumClusters; i++) {
      double dist = distance(instance,
            m_ClusterCentroids.instance(i));
      if (dist < minDist) {
         minDist = dist;
         bestCluster = i;
      }
    }
    if (updateErrors) {
      m_squaredErrors[bestCluster] += minDist;
    }
    return bestCluster;
  }
```

The following code is used to show the clusters membership in the output of the algorithm. This code was added to the toString() method of the SimpleKMeans class.

```
   /* Memberships of each cluster */
    int c = instances.instance(0).numValues();
    temp.append("=== Clusters Membership ===");
    for(int i=0; i< m_NumClusters; i++){
        temp.append("\n    Cluster "+(i+1)+" = {");
      for(int j=0; j<clusterAssignments.length; j++){
         if(clusterAssignments[j]==i)
            temp.append(instances.instance(j).stringValue(c-1)+",");
      }
      temp.deleteCharAt(temp.toString().length()-1);
      temp.append("}");
    }
    return temp.toString();
```

Finally, we use a function to store the results of the method, the clustering assignments, in a file to load them into our application and be able to draw on the map the results.

```
/*
 * Returns the members of each cluster in array
 */
public String[][] getMembers(String output) throws IOException {
   String members[][]=new String[m_NumClusters][];
   String aux=new String();
   String cluster[];
   int c = instances.instance(0).numValues();
   for(int i=0; i< m_NumClusters; i++){
      int k=0;
      aux="";
      for(int j=0; j<clusterAssignments.length; j++){
         if(clusterAssignments[j]==i)
            aux+=(instances.instance(j).stringValue(c-1)+",");
      }
      members[i]=aux.split(",");
   }
   PrintWriter out = new PrintWriter(output);
   for(int i=0; i<m_NumClusters; i++){
      for(int j=0; j<members[i].length-1; j++)
        out.print(members[i][j]+",");
      out.print(members[i][members[i].length-1]+"\n");
   }
   out.close();
   return members;
}
```

3. Trimmed K-means

The code of the trimmed k-means is a modification of the simple k-means. We added the sentences necessary to apply the trims to the instances and we changed also the function which processes the instances. We created a new method, `procesar()`, that now returns the distances to the center to the cluster assigned.

```
/* trimmed k-means algorithm */
//Itereate until method converge
while (!converged) {

   emptyClusterCount = 0;
   m_Iterations++;   //count the iterations
   converged = true; //assume that there si convergence
   //go throws all the instances
   for (i = 0; i < instances.numInstances(); i++) {
      Instance toCluster = instances.instance(i); //take the  i-th
      //Assign the instance i to some cluster
      double newD = procesar(toCluster, i);
      if (newD != distances[i])
      //if one distance changes the method doesn't converge
         converged = false;
      distances[i] = newD; //update the distances array
   }

   //Sort the distances to each centroid
   double[] sorted = distances.clone();
   Arrays.sort(sorted);
   double maxVar = sorted[untrimmed -1];
   int count = 0;
```

```
    // update centroids
    m_ClusterCentroids = new Instances(instances, m_NumClusters);
    for (i = 0; i < m_NumClusters; i++)
       tempI[i] = new Instances(instances, 0)

    // assign only the instances closer than naxVar to the center
    for (i = 0; i < instances.numInstances(); i++){
       if (distances[i] <= maxVar & count < untrimmed){
          tempI[clusterAssignments[i]].add(
          instances.instance(i));
          count++;
       }else clusterAssignments[i] = -1;
    }

    // Calculate the centroids of the clusters
    for (i = 0; i < m_NumClusters; i++) {
       double [] vals = new double[instances.numAttributes()];
       if (tempI[i].numInstances() == 0)
          emptyClusterCount++; // empty cluster
       else {
          for (int j = 0; j < instances.numAttributes(); j++) {
             vals[j] = tempI[i].meanOrMode(j);
             m_ClusterNominalCounts[i][j] =
                tempI[i].attributeStats(j).nominalCounts;
          }
          m_ClusterCentroids.add(new Instance(1.0, vals));
       }
    }

    //If there is empty cluster it doesn't count them
    if (emptyClusterCount > 0) {
       m_NumClusters -= emptyClusterCount;
       tempI = new Instances[m_NumClusters];
    }
    //If the method hasn´t converged reset the values
    if (!converged) {
       m_squaredErrors = new double [m_NumClusters];
       m_ClusterNominalCounts = new int
          [m_NumClusters][instances.numAttributes()][0];
    }
 }

 //Sort the distances to each centroid
 double[] sorted = distances.clone();
 Arrays.sort(sorted);
 double radius = sorted[untrimmed - 1];
 int c = instances.instance(0).numValues();
 trimmed = new java.util.HashMap();
 for (i = 0; i < instances.numInstances(); i++){
    if (distances[i] > radius/* && d<0.5*/){
       trimmed.put(new Double(distances[i]), new String(
                instances.instance(i).stringValue(c - 1)));
       if(clusterAssignments[i] != -1){
          tempI[clusterAssignments[i]].remove(
                instances.instance(i));
          clusterAssignments[i] = -1;
       }
    }
 }

 /* Calcuate the desviations */
```

```
m_ClusterStdDevs = new Instances(instances, m_NumClusters);
m_ClusterSizes = new int [m_NumClusters];
for (i = 0; i < m_NumClusters; i++) {
   double [] vals2 = new double[instances.numAttributes()];
   for (int j = 0; j < instances.numAttributes(); j++) {
      if (instances.attribute(j).isNumeric())
         vals2[j] = Math.sqrt(tempI[i].variance(j));
      else
         vals2[j] = Instance.missingValue();
   }
   m_ClusterStdDevs.add(new Instance(1.0, vals2));
   m_ClusterSizes[i] = tempI[i].numInstances();
}
```

This is the new method designed to process the instances and search the closest cluster for each of them. The difference with respect to the original method that makes this in the SimpleKMeans class is that it returns the distance to the center of the cluster assignment instead of the number of the cluster.

```
private double procesar(Instance instance, int index) {
   double minDist = Integer.MAX_VALUE;
   int bestCluster = 0;
   for (int i = 0; i < m_NumClusters; i++){
      double dist = distance(instance,
                       m_ClusterCentroids.instance(i));
      if (dist < minDist){
         minDist = dist;
         bestCluster = i;
      }
   }
   m_squaredErrors[bestCluster] += minDist;
   clusterAssignments[index] = bestCluster;
   return minDist;
}
```

4. Graphics

The Graphics object contains a function that handles the paint event to draw the polygons that represent the areas on the map. This object is implemented as a J# component and here is shown the code for painting. The following fields are used:

- `see`: A Boolean variable used to know when the graphic must be shown or not.
- `c`: A Color array with 25 different colors to paint 25 different clusters.
- `clusters`: This is a two dimensions array that contains, in each row, the areas belonging to a cluster. Because of this, the array is not a matrix because some clusters will contain more areas than others.
- `areas`: A HashMap that contains, for each area, the array of points that define the polygon associated to the area on the map.

```
private void paint(System.Object o,
      System.Windows.Forms.PaintEventArgs e){

   System.Drawing.Graphics g = e.get_Graphics();

   if (clusters != null & this.see){
         //draw the areas
         for (int i=0; i<clusters.length & i<c.length; i++){
            for (int j = 0; j < clusters[i].length; j++)
               if (clusters[i][j] != null &&
                  areas.get(clusters[i][j]) != null)
                  //fill the polygon with a brush with the color
                  g.FillPolygon(new SolidBrush(c[i]),
                  (PointF[])areas.get(clusters[i][j]));
         }
      }
   }
```

5. Methods Execution

The clustering methods are executed via command line using the following J# functions:

```
/***********************************
 * EXECUTE A COMMAND LINE PROCESS *
 ***********************************/
private String execute(String command, String args){
   String output = "", error = "";
   draw.set_Enabled(false);
   resSaved = true;
   try{
      //create the info of the process that is going to start
      ProcessStartInfo psi = new ProcessStartInfo(command, args);
      psi.set_CreateNoWindow(true);  //force background mode
      psi.set_UseShellExecute(false);//force streams redirection
      psi.set_RedirectStandardOutput(true);
      psi.set_RedirectStandardError(true);
      psi.set_WorkingDirectory(directory); //actual directory
      //run the process and get the output
      Process p = Process.Start(psi);
      output = p.get_StandardOutput().ReadToEnd();
      error = p.get_StandardError().ReadToEnd();
      p.WaitForExit(5000); //wait 5 seconds for the process
      System.gc();
   }
   catch (Win32Exception exc) {
      status.set_Text(exc.get_Message());
   }
   //if there is no error return the output
   if (error.equals("")){
      resSaved = false;
      draw.set_Enabled(true);
      return output;
   }
   return error;
}

/*************************************************
 * METHOD TO APPLY A DETERMINED CLUSTERING METHOD *
 *************************************************/
private void cluster(Object sender, System.EventArgs e){

   if (openDB.dataSets.get_SelectedItem() == null) {
      status.set_Text("Error!! No dataset file loaded");
      new OptionDialog("Information", "You must load a dataset
            file before applying a clustering method.").ShowDialog();
            return;
        }
        if (!resSaved && !results.get_Text().equals(""))
            askSaveRes();
        methodProps.set(sender.toString());

        if (methodProps.ShowDialog() == DialogResult.OK){
            //Get the parameters from the Dialog
            String params = methodProps.method;
            String path = openDB.dataSets.get_SelectedItem().
                ToString().substring(0, openDB.dataSets.
                get_SelectedItem().ToString().lastIndexOf(".")+1);
            numClust = Integer.parseInt(
                methodProps.nClusters.get_Text());

            //Choose the method to apply
            if (sender.toString().Contains("K-Means")){
                saveToArff(instances, path.substring(0,
                    path.lastIndexOf("\\")+1), path.substring(
                    path.lastIndexOf("\\")+1)+sender.toString());
                //Set the params to the method
                params += "-t \""+path+sender.toString()+".arff\" -s "
                    + methodProps.seed.get_Text() + " -N "
                    + methodProps.nClusters.get_Text();
            }
            if (sender.toString().equals("Naive Bayes")){
                numClust *= Integer.parseInt(
                    methodProps.nSecondRoot.get_Text());
                //Set the params to the method
                params += "\"" + openDB.dataSets.get_SelectedItem().
                ToString() + "-tmp\" " + methodProps.nStates.get_Text()
                + " " + methodProps.nClusters.get_Text() + " "
                + methodProps.nSecondRoot.get_Text();
            }

            //Apply the method invoking it via command line
            results.set_Text(execute("java", params));
            tabs.set_SelectedTab(clusters);
            setColors();
            //Load the cluster assigments if everything is OK
            if (draw.get_Enabled()){
                graphic.cluster(readCA(numClust, "" + path
                    + sender.toString() + ".ca"));
                status.set_Text(sender+" method applied successfully");
                resSaved = false;
            }else{
                status.set_Text("Error applying "+sender+" method!!!");
                resSaved = true;
            }
        }
    }
```

6. Clustering Comparison

The clustering comparison is made in the method comparison that is called after the load of the two .ca files considered. The method needs to order a two dimensional array that represents the crossed table of the clustering results, and then apply the concrete comparison criterion. The two criteria are implemented on the methods `maxCriterion()` and `minCriterion()`. The code is shown below.

```
/* CREATES THE CLUSTERING THAT COMPARES TWO CLUSTERINGS */
public void compare(String[][] ca1, String[][] ca2,
   boolean one2two, String crit){
   //Sort the clusters
   status.set_Text("Sorting data");
   ca1 = sort(ca1);
   ca2 = sort(ca2);
   System.gc();

   //Create the map of the clustering
   java.util.AbstractMap map;
   String[][] cruces;

   if (one2two) {
      results.set_Text(results.get_Text()
      + "\n=== Assignments for base clustering 1 ===\n");
      cruces = getMapFrom(ca1, ca2);
   }else{
      results.set_Text(results.get_Text()
      + "\n=== Assignments for base clustering 2 ===\n");
      cruces = getMapFrom(ca2, ca1);
   }

   results.set_Text(results.get_Text() + "\n\n"
      + crit + " criterion results:\n");
   results.set_Text(results.get_Text()
      + "\n\tPairs\tSignificance\n");
   //Apply the selected criterion for cluster equivalence
   if (crit.Contains("Maximum")) map = maxCriterion(cruces);
   else map = minCriterion(cruces);
   System.gc();

   //Get the assignments indexes
   Point[] p = new Point[Math.min(cruces.length,
      cruces[0].length)];
   boolean [][]equiv = new boolean[cruces.length+1]
            [cruces[0].length+1];
   int j = 0;
   for (int i = 0; i < cruces.length || j<p.length; i++){
      if (map.get(new Integer(i)) == null) {
         Point[] q= new Point[j];
         System.Array.ConstrainedCopy(p, 0, q, 0, j);
         p = q; System.gc(); break;
      }
      int X = ((Point)map.get(new Integer(i))).get_X();
      int Y = ((Point)map.get(new Integer(i))).get_Y();
      if (!equiv[X][cruces[0].length] &&
            !equiv[cruces.length][Y]){
         p[j++] = (Point)map.get(new Integer(i));
         equiv[X][Y] = true;
         equiv[X][cruces[0].length] = true;
         equiv[cruces.length][Y] = true;
      }
   }

   results.set_Text(results.get_Text() + "\n");
   //Write the results into a CA file
   numClust = p.length;
```

```
try{
   PrintWriter out = new PrintWriter(
      new FileWriter("Crossed Clustering.ca"));

   //print the cluster similarities
   for(int i=0; i< p.length; i++){
      String line = cruces[p[i].get_X()][p[i].get_Y()].substring(
         0, cruces[p[i].get_X()][p[i].get_Y()].length()-1);
      out.println(line);
      results.set_Text(results.get_Text()+"Cluster "
         +leter[p[i].get_X()]+" corresponds with Cluster "
         +p[i].get_Y()+": {"+line+"}\n");
      colorCodes[i + c.length].set_Text("Cluster " +
         leter[p[i].get_X()] + ":" + p[i].get_Y());
   }

   //print the cluster disimilarities
   results.set_Text(results.get_Text()+"\n");

   //more rows that columns
   if(cruces.length > cruces[0].length){
for (int i = 0; i < cruces.length; i++)
         for (j = 0; j < cruces[0].length; j++)
            if(!equiv[i][j] && cruces[i][j] != null){
            String line = cruces[i][j].substring(0,
               cruces[i][j].length()-1);
            results.set_Text(results.get_Text()+"Instances in "
            +leter[i]+" crossed with "+j+": {"+line+"}\n");
            colorCodes[(numClust++)+c.length].set_Text(
               "Cluster " + leter[i] + "∩" + j);
            out.println(line);
         }
   }else{ //more columns that rows
      for (int i = 0; i < cruces.length; i++){
         String line = ""; int Y = 0;
         for (j = 0; j < cruces[0].length; j++){
            if (equiv[i][j]) Y = j;
            else if (cruces[i][j] != null)
               line += cruces[i][j];
         }
         if (line.length() == 0) line = " ";
         if (equiv[i][Y]){
            results.set_Text(results.get_Text()+"Instances in "
            + leter[i] + " but not in " + Y + ": {"
            + line.substring(0, line.length() - 1) + "}\n");
            colorCodes[(numClust++) + c.length].set_Text(
               "Cluster " + leter[i] + "\\" + Y);
         }else{
            results.set_Text(results.get_Text()+"Instances in "
            + leter[i] + " not crossed: {"
            + line.substring(0, line.length() - 1) + "}\n");
            colorCodes[(numClust++)+c.length].set_Text(
               "Cluster " + leter[i]);
         }
         out.println(line.substring(0, line.length() - 1));
      }
   }
   out.close();

   //Put the clustering on the map
   graphic.cluster(readCA(numClust, "Crossed Clustering.ca"));
```

```
    }catch (Exception excep) {
       status.set_Text("IOException writint the CA file");
    }
 }

 /* MULTIPLY THE MATRIX BY THE PROPORTION OF BOTH FREQUENCIES     *
  * REPRESENTED BY EACH CELL AND APPLY THESE RESULTS AS CRITERION */
 public java.util.AbstractMap minCriterion(String[][] cruces) {

    java.util.AbstractMap map = new java.util.HashMap();
    double[] percents = new double[cruces.length*cruces[0].length];
    int[] cumulativeY = new int[cruces[0].length];
    for (int i = 0; i < cruces.length; i++){
       int cumulativeX = 0;
       for (int j = 0; j < cruces[i].length; j++){
          int next = 0;
          if (cruces[i][j] != null)
             next = split(cruces[i][j].substring(0,
                cruces[i][j].length() - 1), ",").length;
          percents[i * cruces[i].length + j] = next;
          cumulativeX += next;
          cumulativeY[j] += next;
       }
       //calculate the proportion of data represented by rows
       for (int j = 0; j < cruces[i].length; j++)
          percents[i * cruces[i].length + j] /=
             cumulativeX / percents[i * cruces[i].length + j];
    }

    //calculate the proportion of data represented by columns
    for (int i = 0; i < cruces.length; i++)
       for (int j = 0; j < cruces[i].length; j++){
          percents[i * cruces[i].length + j] /= cumulativeY[j];
          while (map.get(new Double(
          percents[i * cruces[i].length + j])) != null)
             percents[i * cruces[i].length + j] =
             (percents[i*cruces[i].length + j]
             * cumulativeY[j] - 0.005) / cumulativeY[j];
          map.put(new Double(percents[i * cruces[i].length + j]),
                new Point(i, j));
       }

    //Sort the array descending and update the map
    System.Array.Sort(percents);
    System.Array.Reverse(percents);
    for (int i=0; i<percents.length && percents[i]!=0.0; i++){
       int X = ((Point)map.get(new Double(percents[i]))).get_X();
       int Y = ((Point)map.get(new Double(percents[i]))).get_Y();
       map.put(new Integer(i), map.remove(
          new Double(percents[i])));
       String perc = ((percents[i]*100)+"00000").substring(0,5);
       results.set_Text(results.get_Text() + "\t "+leter[X]
          +"-"+Y+"\t   "+perc+"%\n");
    }
    return map;
 }

 /* GETS THE BIGGER CROSSES AS CRITERION */
 public java.util.AbstractMap maxCriterion(String[][] cruces) {

    java.util.AbstractMap map = new java.util.HashMap();
    double[] percents=new double[cruces.length*cruces[0].length];
    int cumulative = 0;
    for (int i = 0; i < cruces.length; i++){
       for (int j = 0; j < cruces[0].length; j++){
          int next = 0;
          if (cruces[i][j] != null)
             next = split(cruces[i][j].substring(0,
                cruces[i][j].length() - 1), ",").length;
          percents[i * cruces[0].length + j] = next;
          cumulative += next;
       }
    }

    //calculate the proportion of data represented by columns
    for (int i = 0; i < cruces.length; i++)
       for (int j = 0; j < cruces[0].length; j++){
          percents[i * cruces[0].length + j] /= cumulative;
          while(map.get(
          new Double(percents[i*cruces[0].length+j]))!=null)
             percents[i * cruces[0].length + j] =
             (percents[i * cruces[0].length + j] * cumulative
             - 0.005) / cumulative;
          map.put(new Double(percents[i * cruces[0].length + j]),
             new Point(i, j));
       }

    //Sort the array descending and update the map
    System.Array.Sort(percents);
    System.Array.Reverse(percents);
    for (int i=0; i<percents.length && percents[i]!=0.0; i++){
       int X = ((Point)map.get(new Double(percents[i]))).get_X();
       int Y = ((Point)map.get(new Double(percents[i]))).get_Y();
       map.put(new Integer(i), map.remove(new Double(percents[i])));
       String perc = (""+(percents[i] * 100)).substring(0,5);
       results.set_Text(results.get_Text() + "\t "+leter[X]
          +"-"+Y+"\t   "+perc+"%\n");
    }
    return map;
 }
```

Bibliography

[ACW05] Au, W.H., Chan, K.C.C.; Wong, A.K.C.; Wang, Y. "Attribute Clustering for Grouping, Selection, and Classification of Gene Expresion Data". *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 2, No. 2, April-June 2005.

[CGM97] Cuesta-Albertos, J.A.; Gordaliza, A.; Matrán, C. "Trimmed k-means: an attempt to robustify quantizers". *Ann. Statist.* 25 553–576. 1997.

[Did71] Diday, E. "Une nouvelle methode en classification automatique et reconnaissance des formes : la methode des nuees dynamiques". *In Rev. Statist. Appl.*, volume 19, pages 19-33. 1971.

[DLR77] Dempster, A.P; Laird, N.M.; Rubin, D.B. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of Royal Statistical Society. Series B (Methodological)*, Vol. 39, Number 1 (1997) 1-38. 1977.

[Gam05] Gámez Martin, J.A. Notes from the lecture "Minería de datos". Universidad de Castilla-La Mancha. 2005.

[Gam96] Gammerman, A. "Computational Learning and Probabilistic Reasoning". John Wiley and Sons Ltd. 1996.

[Gar05] Garibay, V. Glz. Notes from the lecture "Análisis de Datos". Universidad de Valladolid. 2005.
`http://www.eio.uva.es/~valentin/ad3d/transp.html`

[GGM03] García-Escudero, L.A.; Gordaliza, A.; Matrán, C. "Trimming Tools in Exploratory Data Analysis". *Journal of Computational & Graphical Statistics*, Volume 12, Number 2, 1 June 2003, pp. 434-449(16). 2003.

[GGM99] García-Escudero, L.A.; Gordaliza, A.; Matrán, C. "A central limit theorem for multivariate generalized trimmed *k*-means". *Ann. Statist.* 27 1061–1079. 1999.

[Gor91] Gordaliza, A. "Best approximations to random variables based on trimming procedures". *J. Approx. Theory* 64 162–180. 1991.

[Hal05] Hall, M. "A presentation about Weka's history, current state and future plans". Weka's homepage. 2005.
`http://www.cs.waikato.ac.nz/ml/weka/`

[Hug05] The Hugin Expert A/S. HUGIN API 6.4 Reference Manual. 2005.

[Jen96] Jensen, F. V. "An Introduction to Bayesian Network". *UCL Press Limited*. 1996.

[JN05] Jensen, F.V.; Nielsen, T. D. “Bayesian Networks and Decision Graphs II”. *Preprint*. 2005.

[Kri00] Kriegel, H.P. “Density-Based Cluster and Outlier Analysis”. 2000. http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering

[Mar01] Martin, C. “Java Knowledge Base ITToolbox Java Overview”. 2001. http://java.ittoolbox.com/pub/java_overview.htm

[Mic05] Microsoft MSDN library. http://msdn.microsoft.com/library

[Mit97] Mitchell, T. “Machine Learning”. *McGraw-Hill International*. 1997.

[MST94] Michie, D.; Spiegelhater, D.J.; Taylor, C.C. “Machine Learning, neural and statistical classification”. Ellis Horwood Limited. 1994.

[PLL99] Peña, J.M.; Lozano, J.A.; Larrañaga, P. “Learning Bayesian networks for clustering by means of constructive induction”. Preprint. 1999.

[RS97] Ramoni, M.; Sebastiani, P. “Bayesian Inference with Missing Data Using Bound and Collapse”. *Knowledge Media Institute*. November 1997.

[RS98] Ramoni, M.; Sebastiani, P. “Parameter Estimation in Bayesian Networks from Incomplete Databases”. *Intelligent Data Analysis 2*. 1998.

[Sch97] Schafer J.L. “Analysis of Incomplete Multivariate Data”. *Chapman & Hall/CRC Press LLC*. 1997.

[Sin05] Singhi, S. ”Weka Frequently Asked Questions (FAQ)”. 2005. http://www.public.asu.edu/~sksinghi/weka-faq.html

[Sun04] Sun Microsystems, Inc. “Java™ 2 Platform Standard Edition 5.0 API Specification”. 2004. http://java.sun.com/j2se/1.5.0/docs/api

[Sun05] Sun Developer Network (SDN). http://java.sun.com/