

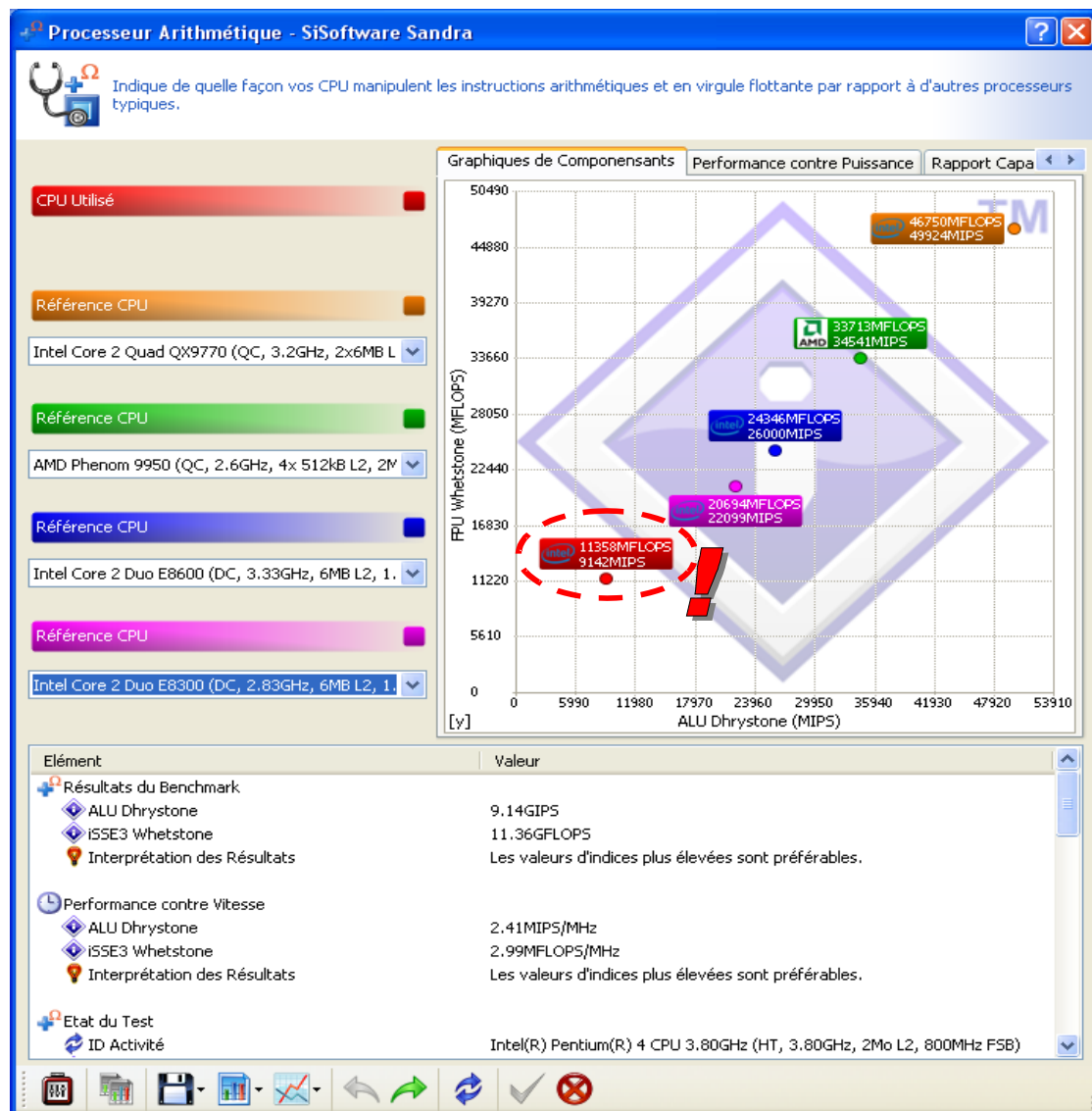
1 Subject

Comparing the quickness and memory occupation of various free implementation of the C4.5 (or assimilated) decision tree algorithm on large dataset.

Dealing with large dataset is one of the most important challenge of the Data Mining. In this context, it is interesting to analyze and to compare the performances of various free implementations of the learning methods, especially the computation time and the memory occupation. Most of the programs download all the dataset into memory. The main bottleneck is the available memory.

In this tutorial, we compare the performance of several implementations of the algorithm C4.5 (Quinlan, 1993) when processing a file containing 500,000 observations and 22 variables.

Our main criteria are memory occupation and computation time. So that everyone can replicate the experience and compare the results, here are the characteristics of our machine: a Pentium 4 3.8 GHz with 2 GB of RAM. We have measured the performances of our computer using the Lite version of the SISoftware SANDRA (<http://www.sisoftware.net/>). We obtain:



We compare the performances of the following programs in this tutorial.

Software	Version	URL
KNIME	1.3.5	http://www.knime.org/index.html
ORANGE	1.0b2	http://www.ailab.si/orange/
R (package rpart)	2.6.0	http://www.r-project.org/
RAPIDMINER (YALE)	Community Edition	http://rapid-i.com/
SIPINA	Research	http://eric.univ-lyon2.fr/~ricco/sipina.html
TANAGRA	1.4.27	http://eric.univ-lyon2.fr/~ricco/tanagra/
WEKA	3.5.6	http://www.cs.waikato.ac.nz/ml/weka/

2 The WAVE dataset

Our data file is WAVE500K.ZIP (<http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/wave500k.zip>). It is the well-known artificial dataset described in the CART book (Breiman et al., 1984). We have generated a dataset with 500.000 observations. The class attribute has 3 values, there are 21 continuous predictors.

We deal primarily with ARFF WEKA file format. The majority of data mining software can handle this format. Otherwise, text file format with tabulation separator will be used.

3 Comparison of performances

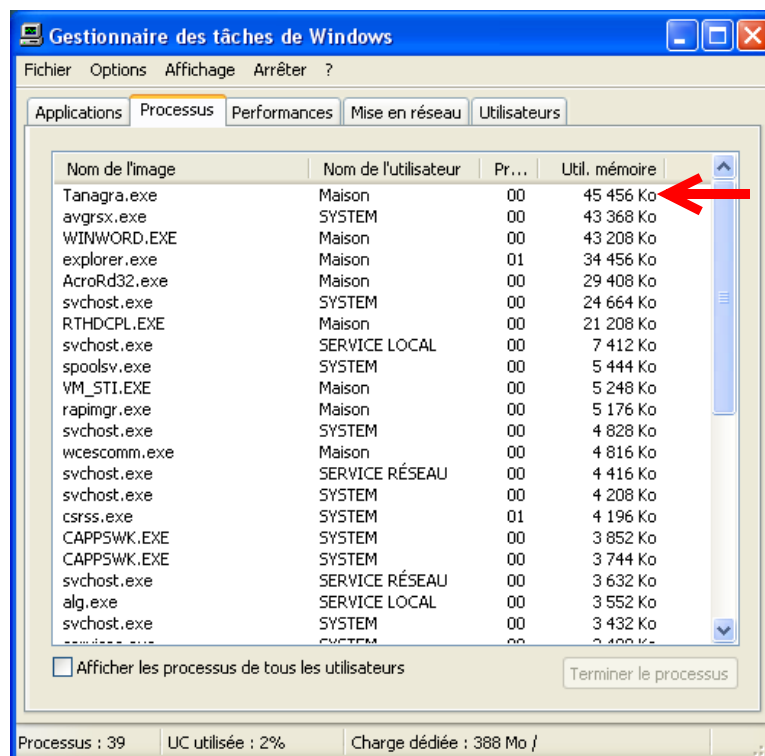


Figure 1 – Measuring the memory occupation under Windows OS

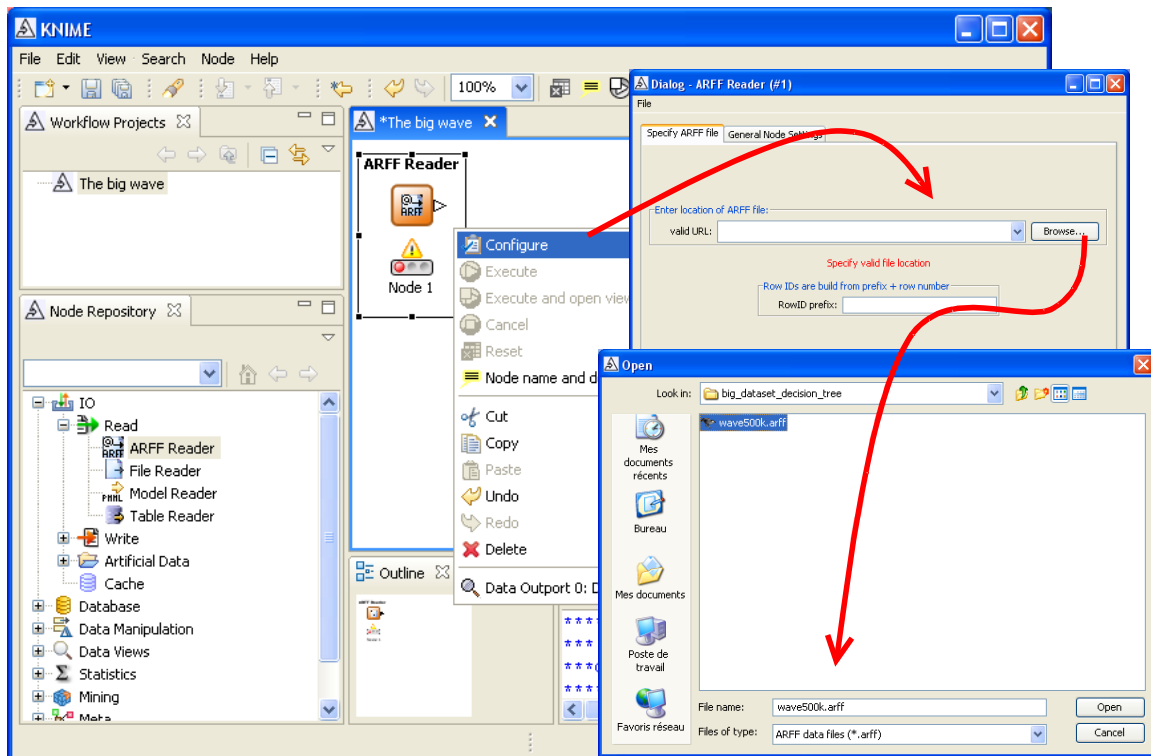
We utilize the indications of Windows OS about the memory occupation (Figure 1). We measure the memory occupation when the program is launched, when the dataset is loaded, during the learning phase, after the learning phase. The maximum occupation during the learning phase is the most important indicator. This is the main bottleneck of the whole process.

About the computation time, some programs provide it. For the other cases, we use a chronograph. Indeed, we try mainly to obtain an approximated value of the computation time. It is not necessary to measure a very precise value.

3.1 KNIME

KNIME (Konstanz Information Miner -- <http://www.knime.org/>) describes the treatments as a succession of operations, represented by a workflow.

KNIME and the JRE. When we launch the software, the JRE (Java Runtime Environment) is started and the global memory occupation is 92.6 MB.



We create a new project (FILE / NEW). We add the ARFF READER component into the workflow and we click on the CONFIGURE menu in order to select the data file. After the loading, the memory occupation becomes 160.4 MB.

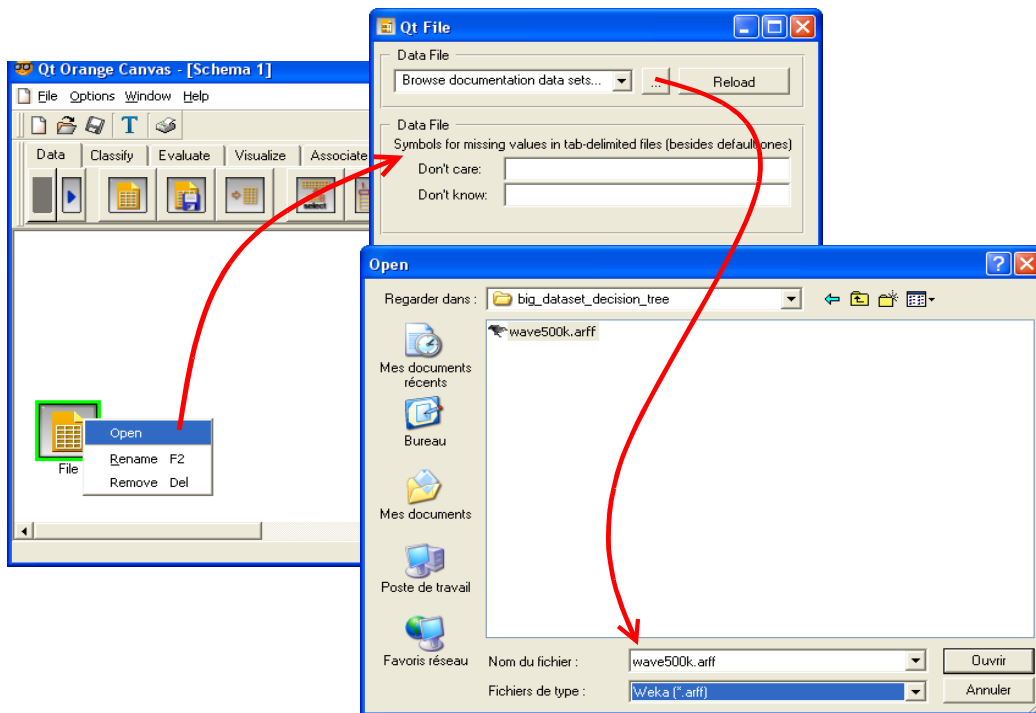
We insert the decision tree LEARNER in the workflow, we see on the panel at the right a description of the method and its parameters. We connect the component ARFF READER, then we click on the menu CONFIGURE in order to define the parameters.

It is not really the C4.5 method, but it is quite similar. We set the minimum number of examples per node at 1000. We set the number of thread used to 1 in order to make the results comparable. We note that Knime is the only one software that can use several threads during the induction of the decision tree.

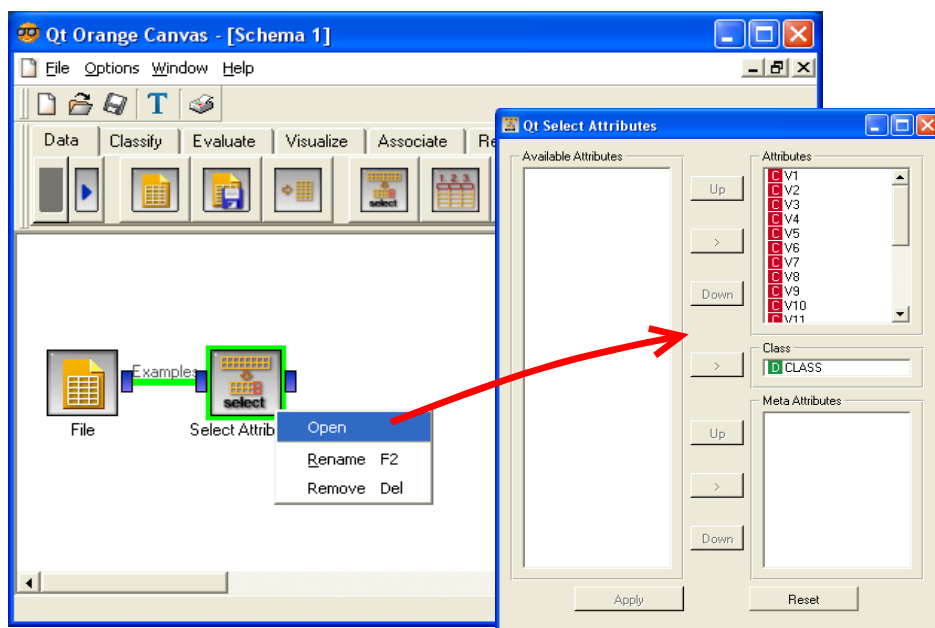
3.2 ORANGE

ORANGE can be used through standard programming language (python) or visual programming. We use this last framework in this tutorial.

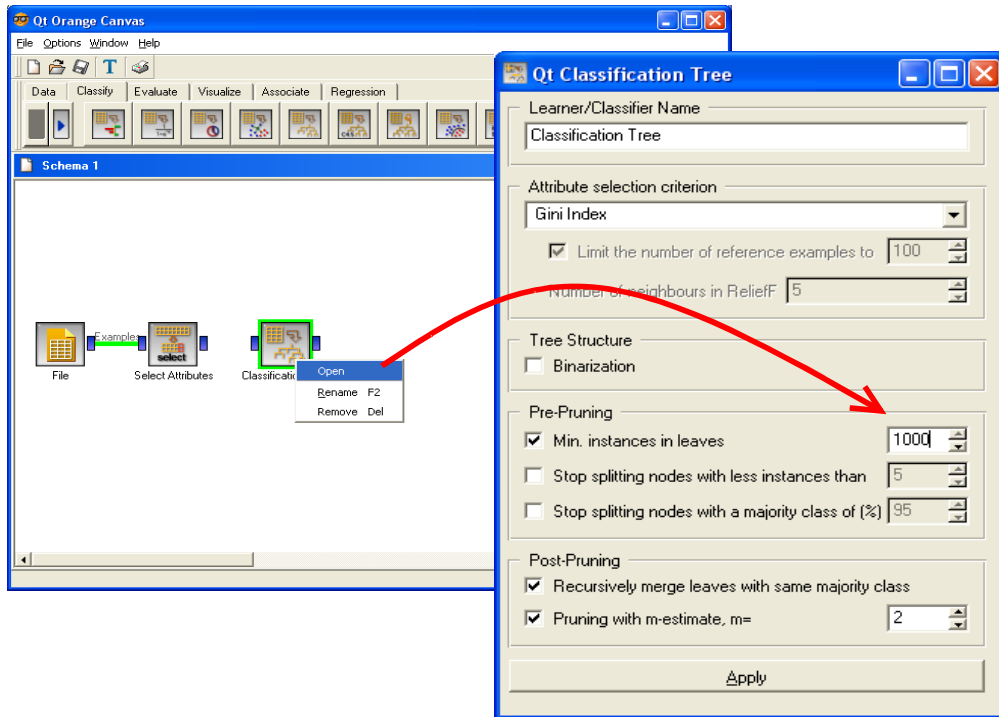
When we launch ORANGE, the **Python Runtime Environment** is started. The global memory occupation is 24.9 MB. We have a familiar interface. The treatments are defined in a schema. The components are available in the tool bar. We add the FILE (DATA tab) into the schema. We click on the OPEN menu and we select the data file. The loading is automatically started; there is no specific menu for launching the operation.



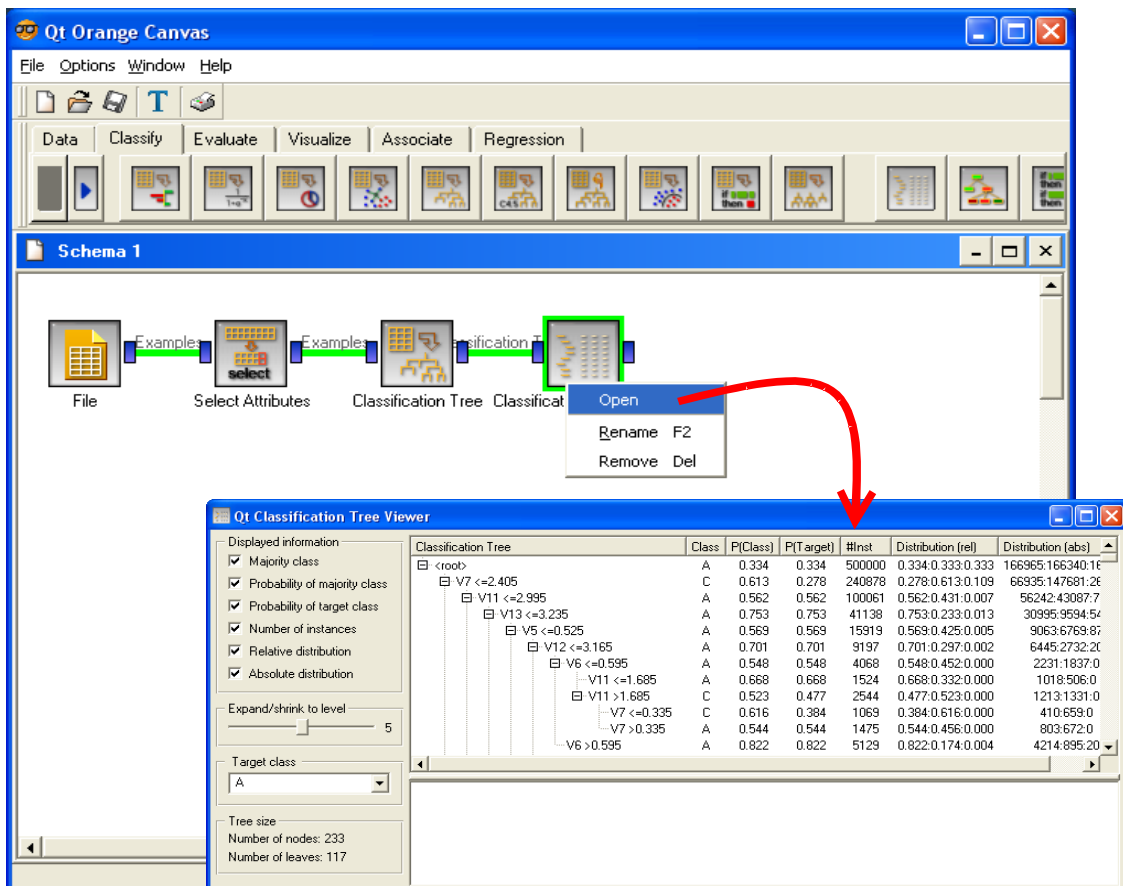
The computation time is 90 seconds. The memory occupation becomes 259.5 MB. We add the SELECT ATTRIBUTE component in order to define the position of the variables.



We can insert now the CLASSIFICATION TREE (CLASSIFY tab) component. We configure it **before** the connection. We click on the OPEN MENU. We set the parameters of the algorithm.



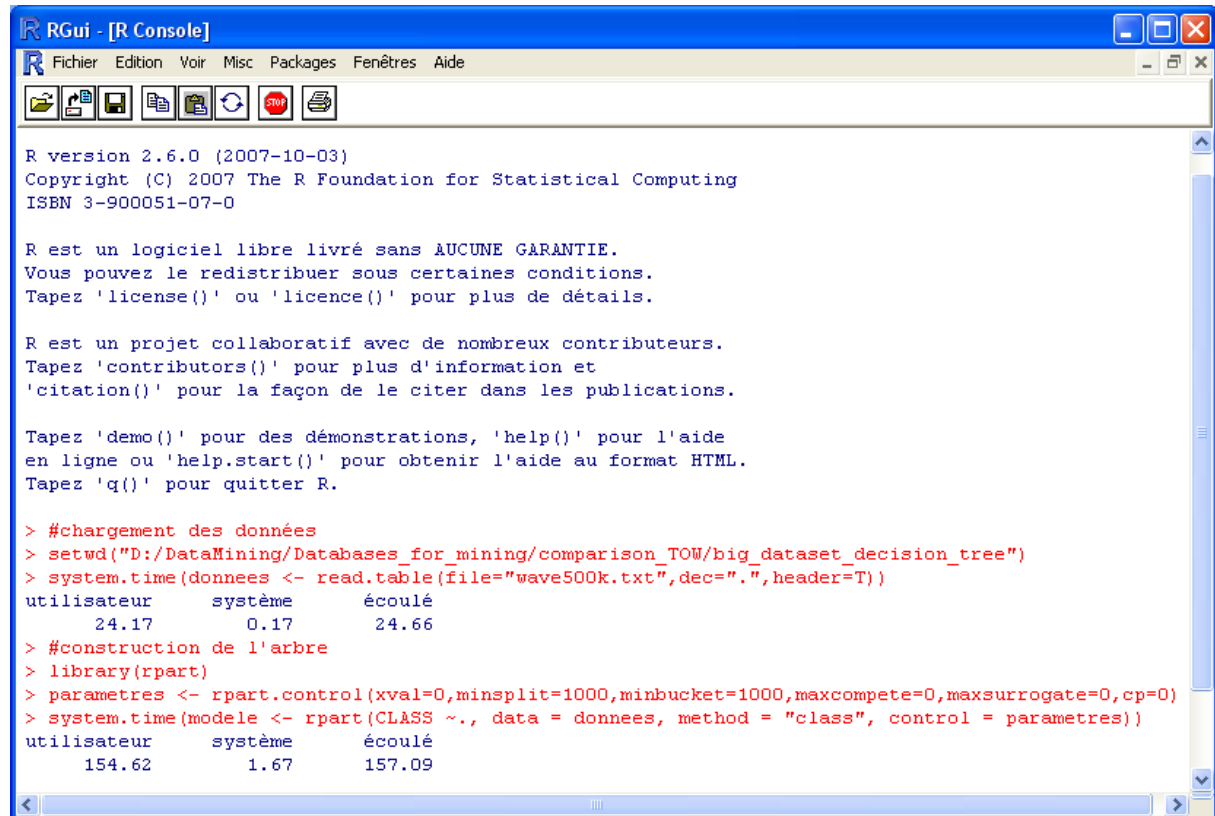
We insert the CLASSIFICATION TREE VIEWER (CLASSIFY tab) into the schema. We make the connection between this viewer and the learner. Finally, we make the connection between SELECT ATTRIBUTES and CLASSIFICATION TREE. The computation is automatically launched.



The processing time is 130 seconds; the memory occupation is 795.7 MB. The tree includes 117 leaves.

3.3 The R Software with the RPART package

R works with scripts. We use the **system.time(.)** command to obtain the processing time.



```

RGui - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

R version 2.6.0 (2007-10-03)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> #chargement des données
> setwd("D:/DataMining/Databases_for_mining/comparison_TOW/big_dataset_decision_tree")
> system.time(donnees <- read.table(file="wave500k.txt",dec=".",header=T))
utilisateur      système      écoulé
      24.17         0.17        24.66
> #construction de l'arbre
> library(rpart)
> parametres <- rpart.control(xval=0,minsplit=1000,minbucket=1000,maxcompete=0,maxsurrogate=0,cp=0)
> system.time(modele <- rpart(CLASS ~., data = donnees, method = "class", control = parametres))
utilisateur      système      écoulé
      154.62         1.67       157.09

```

At the beginning, the memory occupation is 18.8 MB. We use the text file format with R, we load the WAVE500K.TXT. The computation is 24 seconds; the memory occupation becomes 184.1MB.

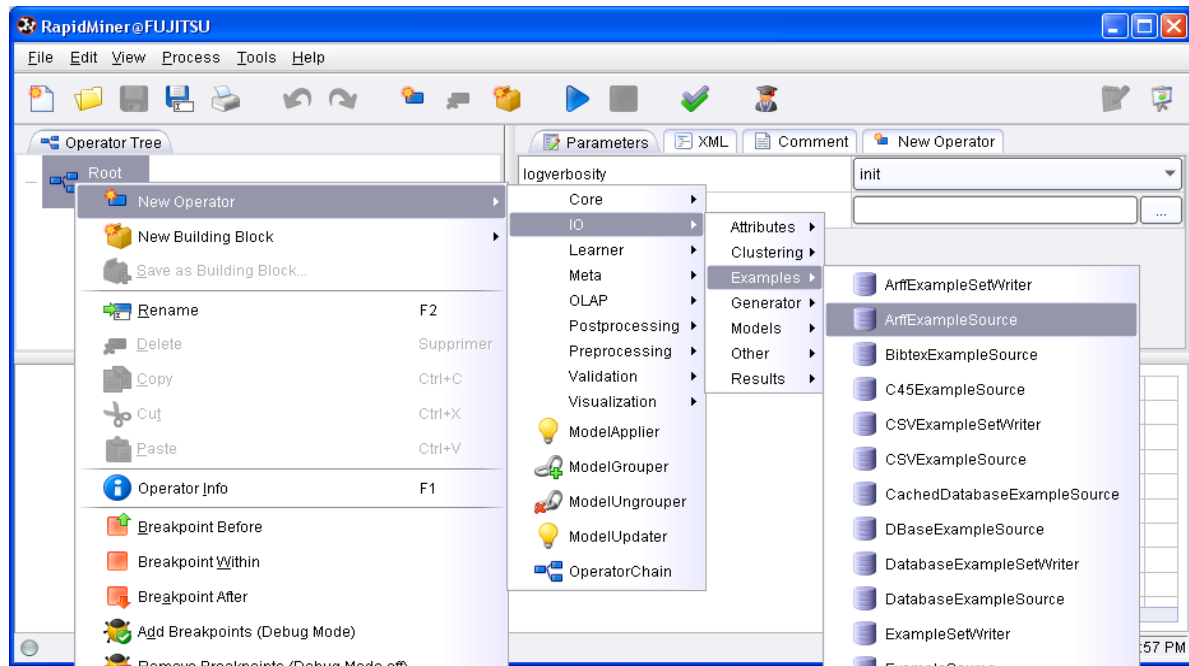
The RPART procedure is rather similar to CART method. But we set the parameters in order to produce a tree similar to the others programs i.e. MINSPLIT = 1000 observations; MINBUCKET = 1000 observations; CP = 0. There is not post-pruning here. But, it does not matter. We know that this part of the processing is fast in the C4.5 approach.

The processing time for the tree construction is 157 seconds. The memory occupation becomes 718.9 MB.

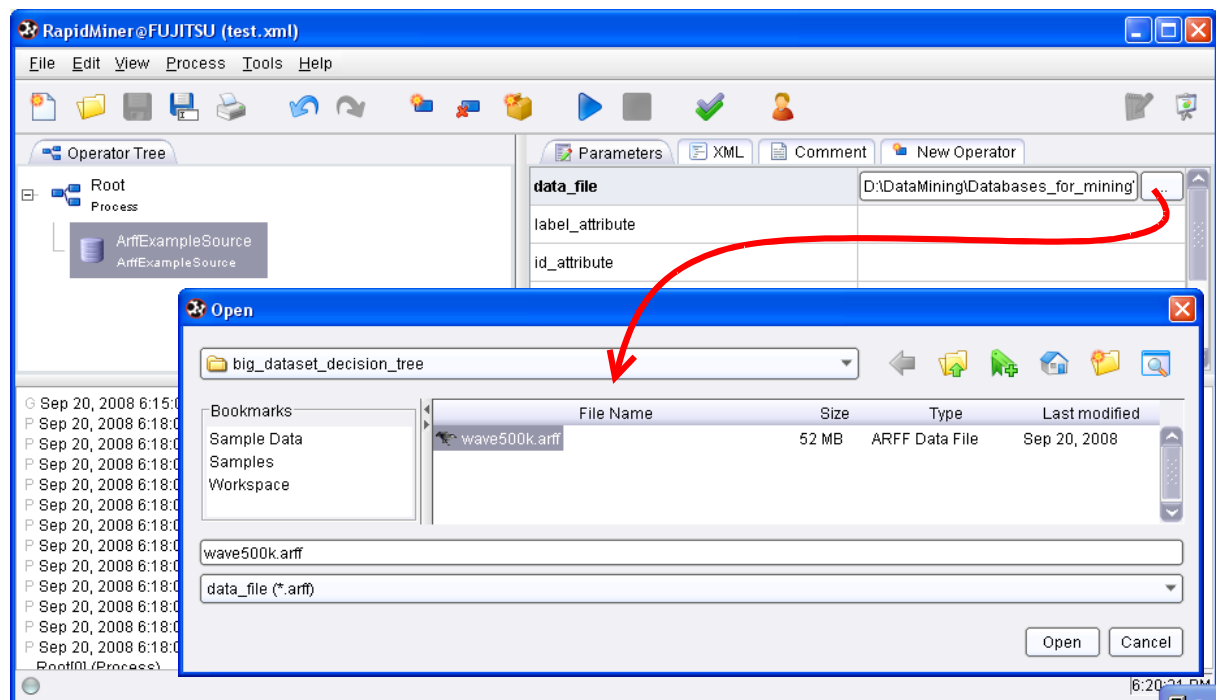
3.4 RAPIDMINER (formerly YALE)

When we launch RAPIDMINER, we create a new project by clicking on FILE / NEW. An empty OPERATOR TREE is available.

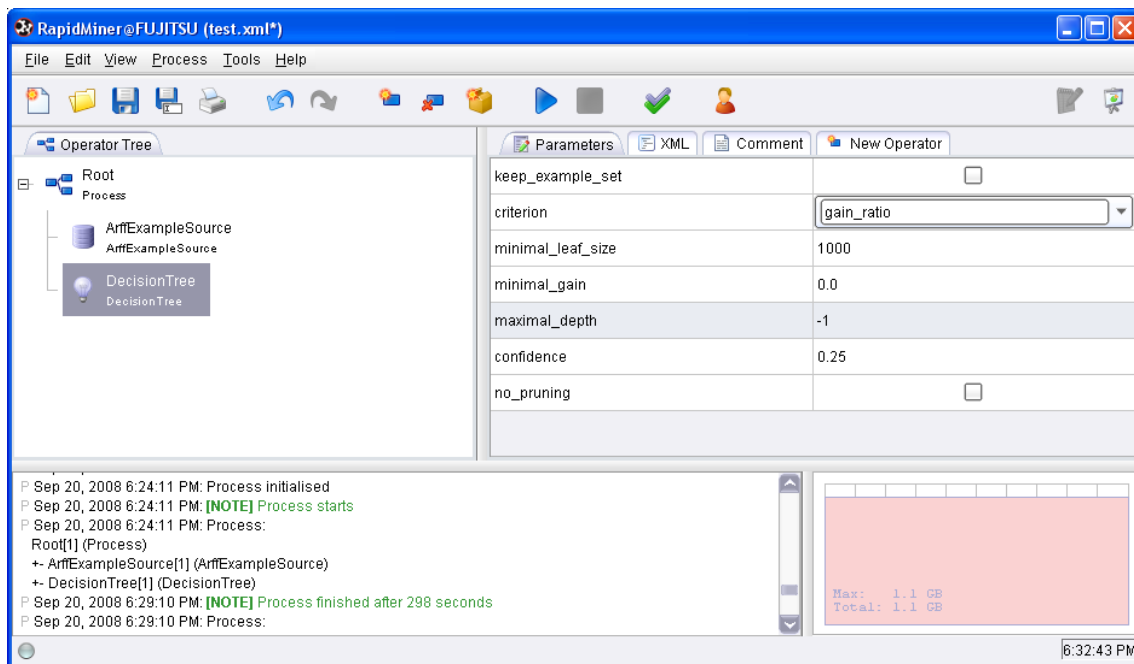
We add an ARFFEXAMPLESOURCE component into the tree, in order to load the dataset.



The component appears into the operator tree, we set the right parameters by selecting our data file. We click on the PLAY button of the tool bar. The computation time for the data importation is 7 seconds. The allocated memory increases from 136.6MB to 228.1MB.



We must now insert the "decision tree" into the diagram. We operate from the menu NEW OPERATOR on the root of the tree. We use the DECISIONTREE operator. We use parameters similar to the other programs.



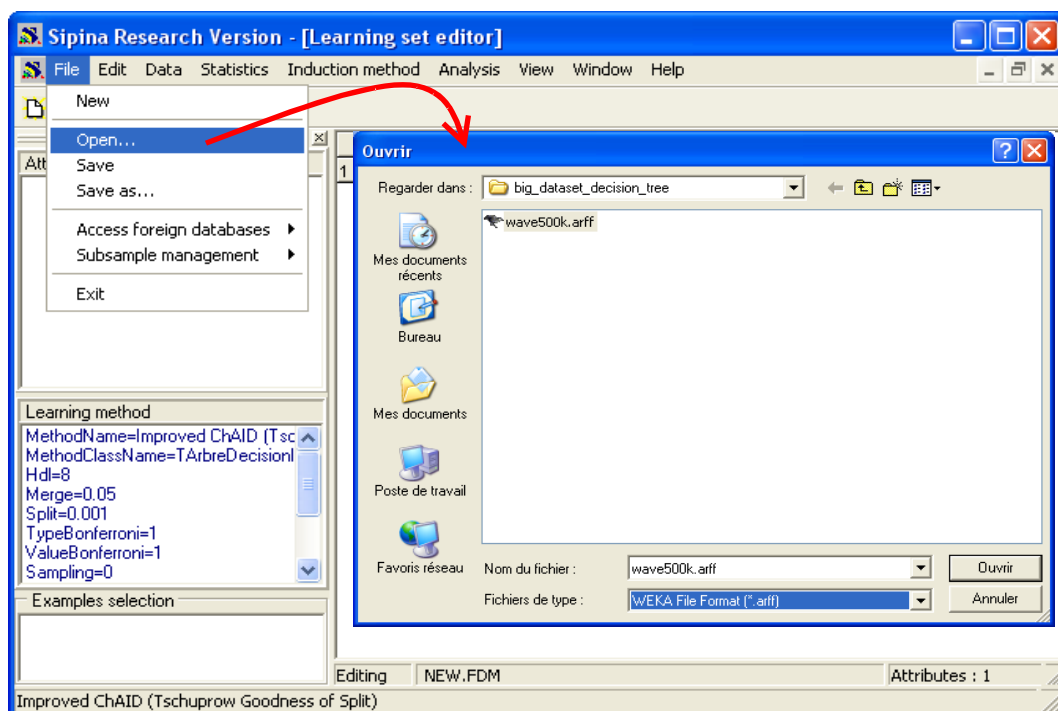
The PLAY button starts again all the computations. We obtain the decision tree after 298 seconds.

Curiously, RAPIDMINER needs much memory: 1274.7MB were necessary during the calculations. It seems we must take with caution this value.

3.5 SIPINA

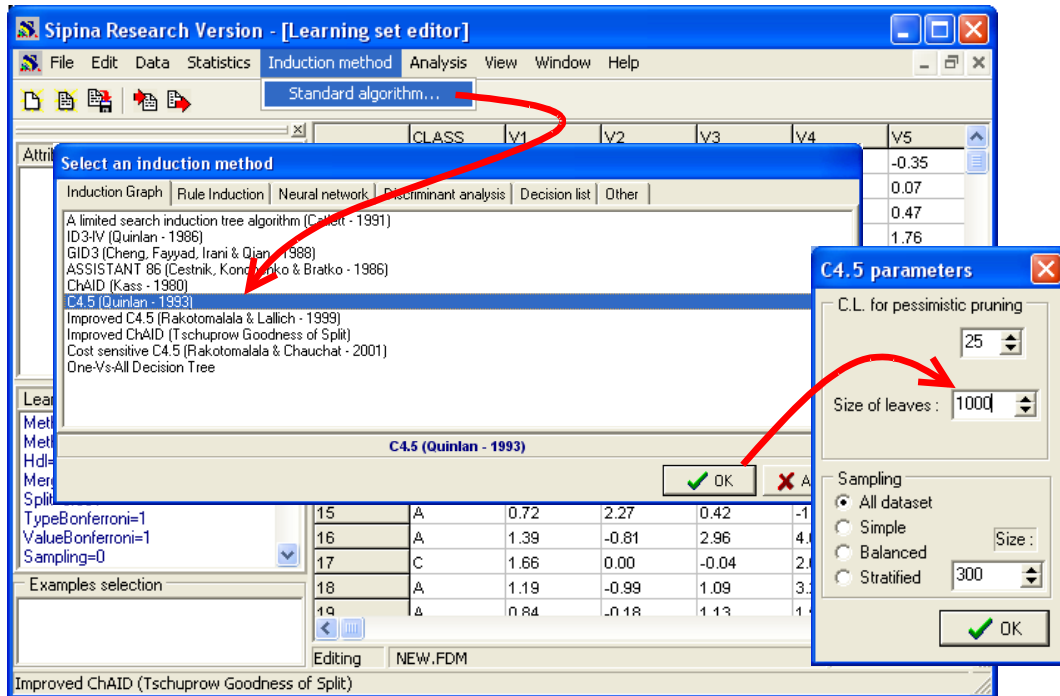
SIPINA is one of my old projects, specialized to decision tree induction. Unlike the other programs, SIPINA proposes a graphical display of the tree, and above all, possesses interactive functionalities.

This advantage is also a drawback because SIPINA must store much information on each node of the tree: the characteristics of split for the other variables, distributions, list of examples, etc. Thus, the memory occupation will be inevitably high compared to the other programs.

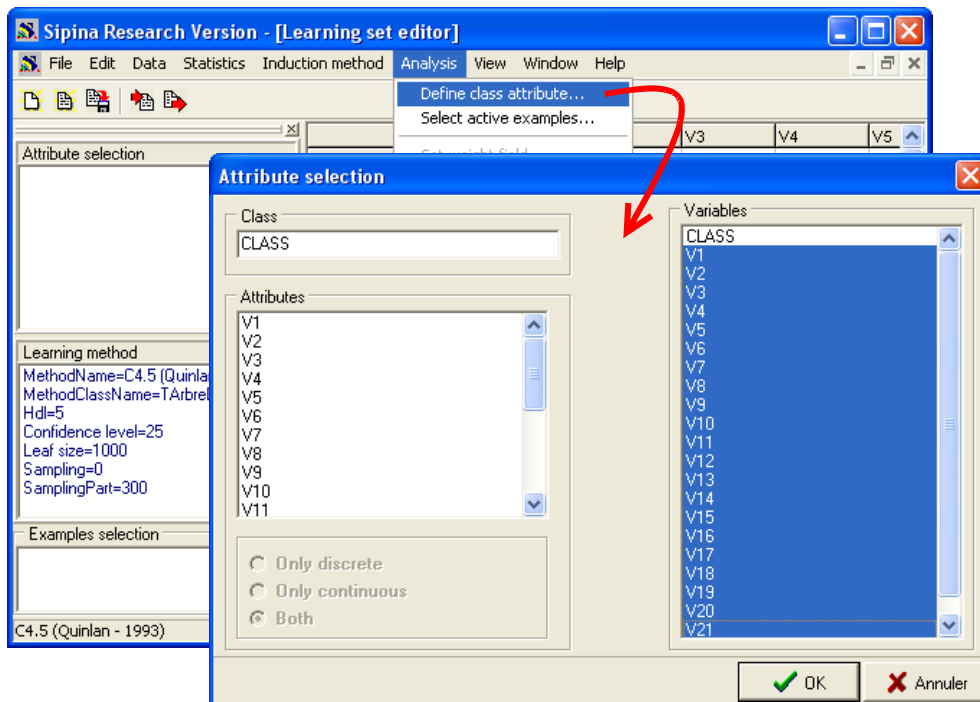


The memory occupation is 7.8 MB when we launch SIPINA. It becomes 67.1 MB after the dataset loading. The computation time is 25 seconds.

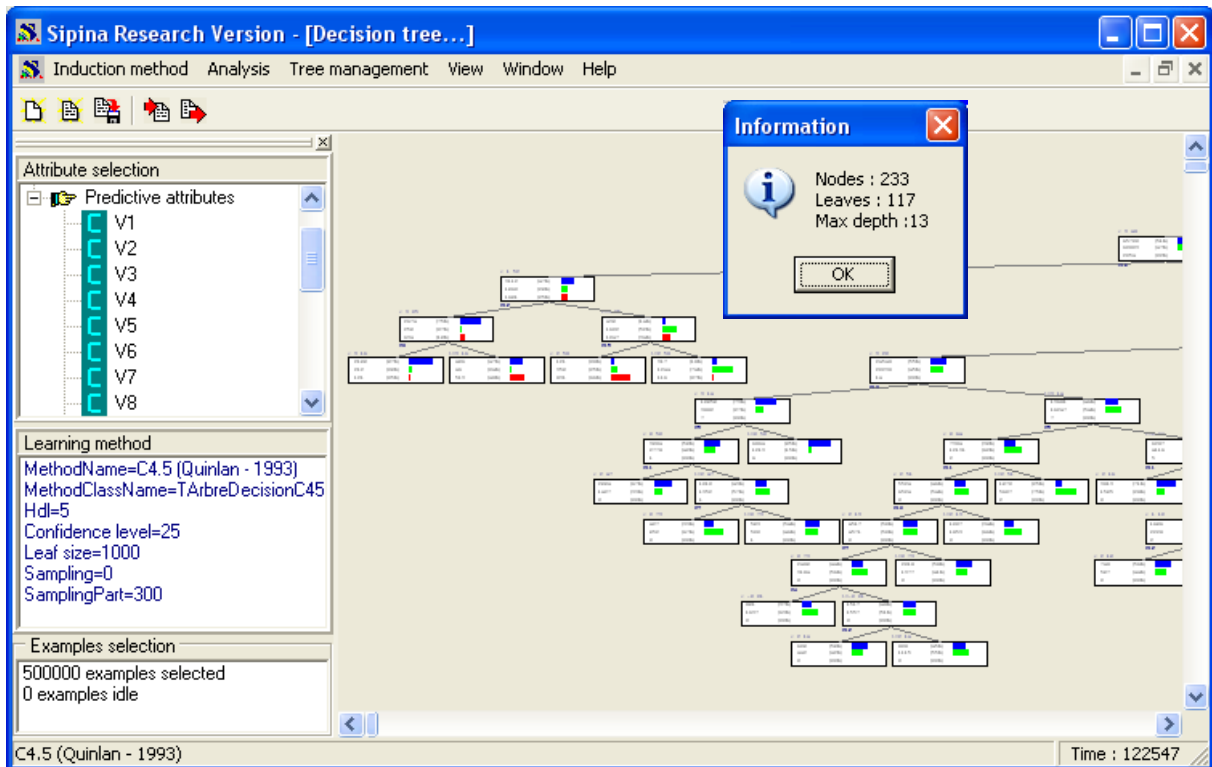
We select the C4.5 algorithm by clicking on the INDUCTION METHOD / STANDARD ALGORITHM menu. We set the minimal size of leaves to 1000.



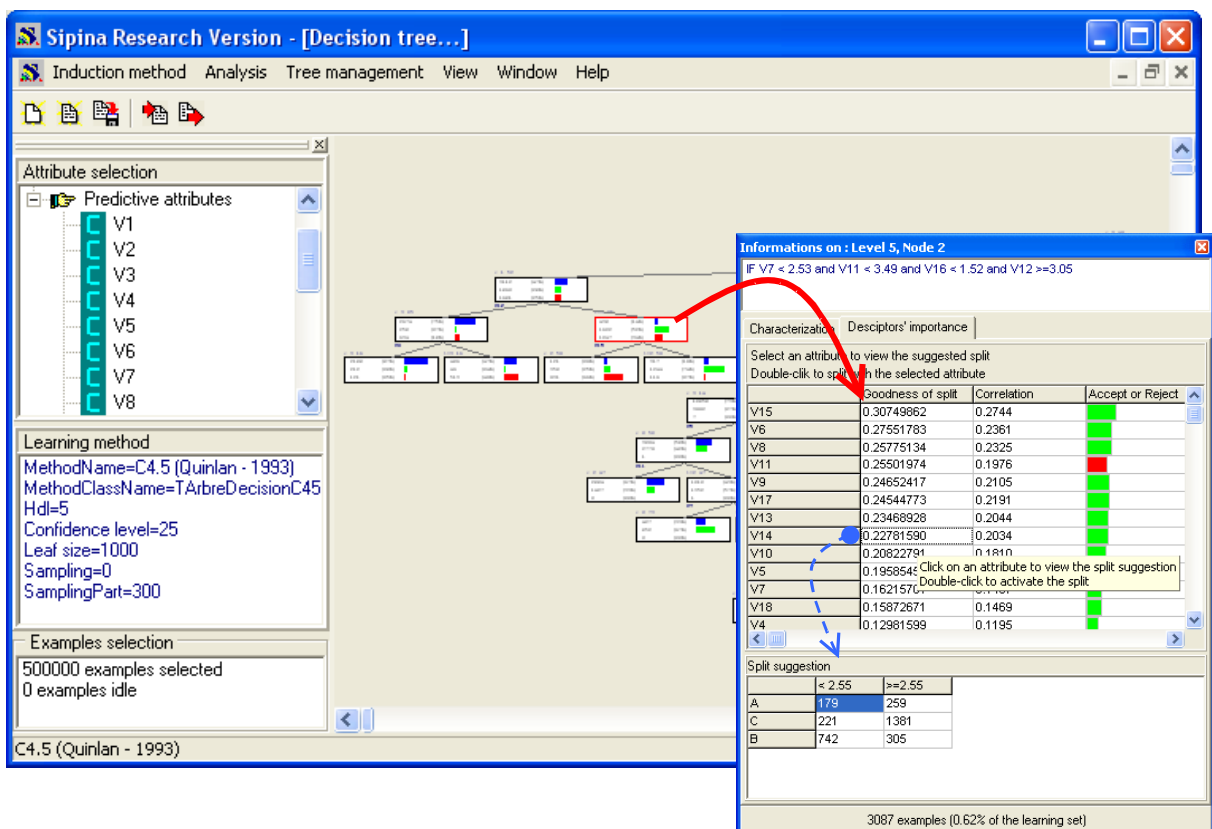
We must specify the status of the variables. We click on the ANALYSIS / DEFINE CLASS ATTRIBUTE menu. We set CLASS as TARGET, and the other variables (V1 ... V21) as INPUT.



We can launch the learning phase. The calculation time is 122 seconds. The memory occupation becomes 539.9 MB for a tree with 117 leaves.



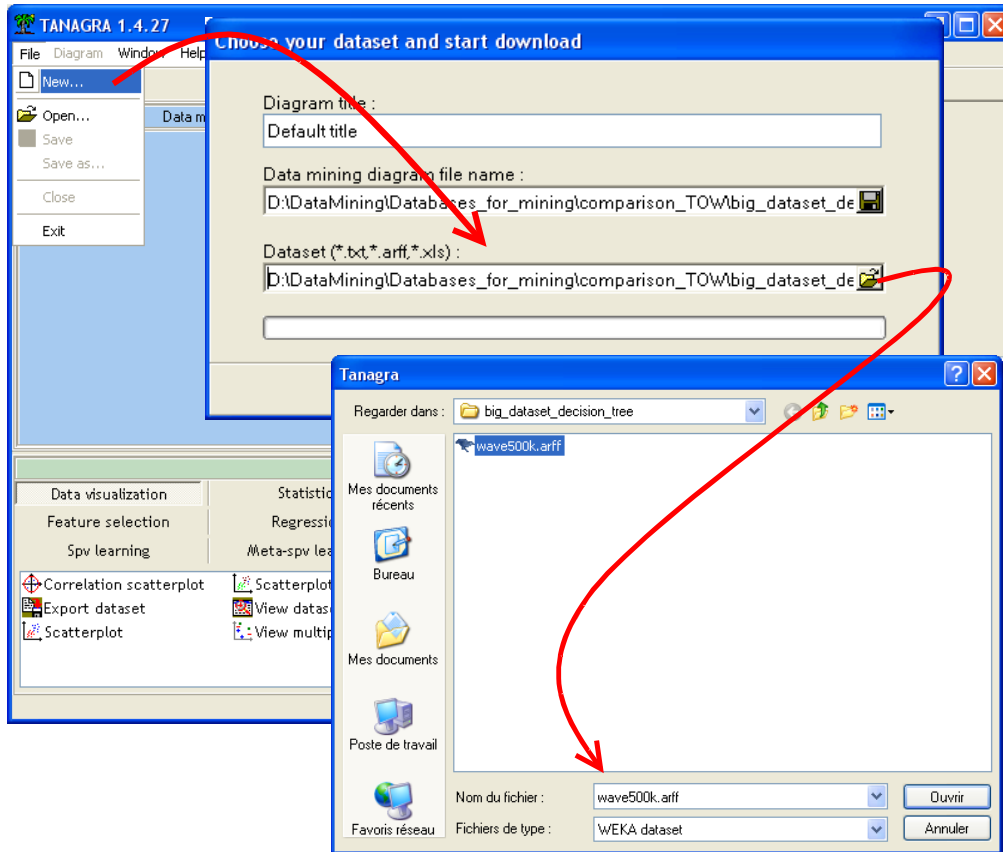
The counterpart of this large memory occupation is that we can explore deeply each node. In the screen shot below, we observe the splitting performance for each variable and, when we select a variable, we see the class distribution on the subsequent leaves.



3.6 TANAGRA

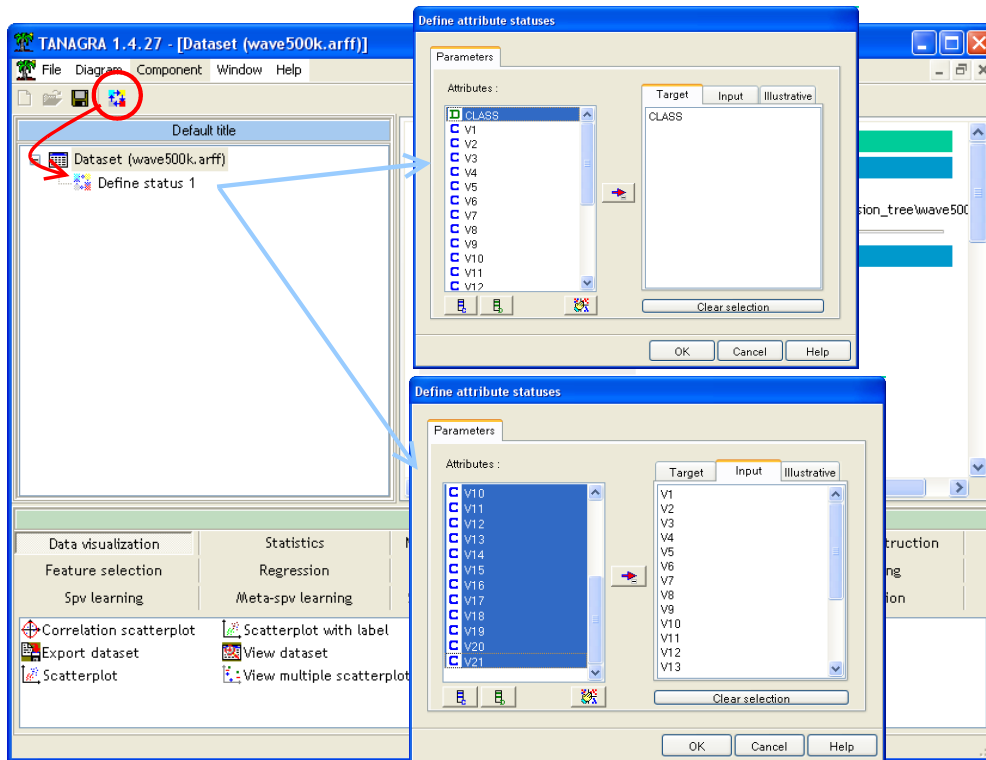
TANAGRA is my current project. Unlike SIPINA, the implementation of the tree is widely modified. The interactive functionalities are anymore available, but the computation time and the memory occupation are optimized.

When we launch TANAGRA, the memory occupation is 7 MB. We click on the FILE / NEW menu in order to create a new diagram. We import the WAVE500K.ARFF data file.

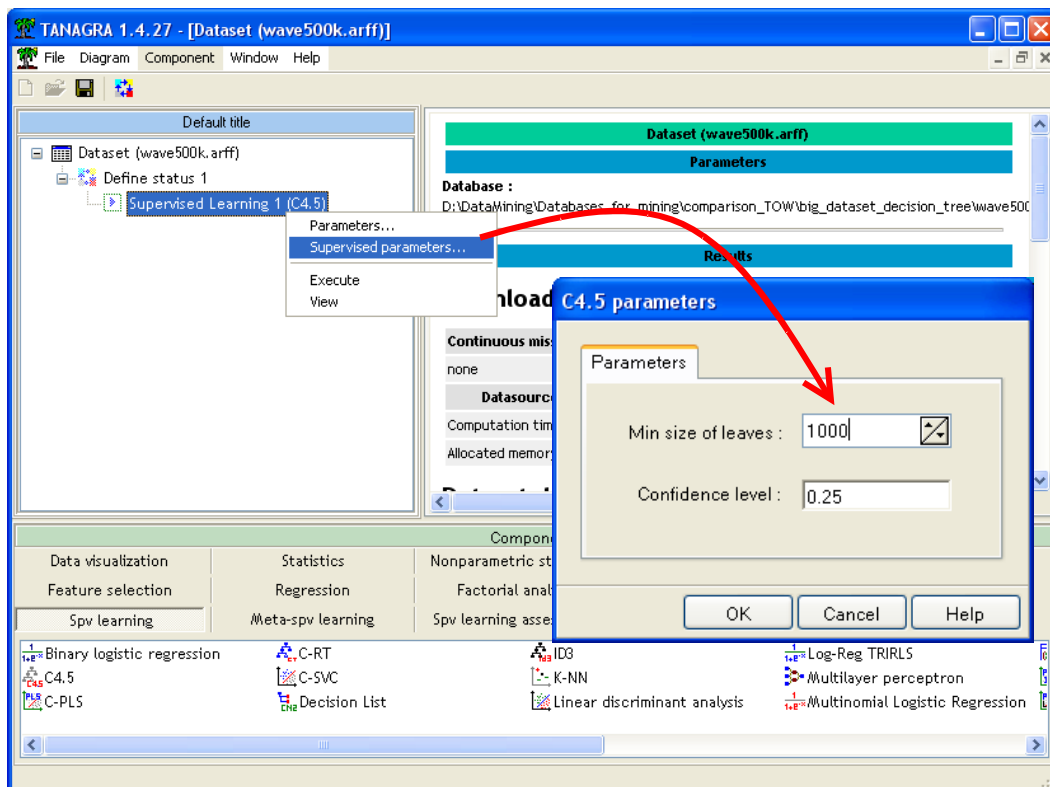


The processing time is 11 seconds; the memory occupation becomes 53.1 MB.

Before the learning phase, we must specify the role of each variable. We use the DEFINE STATUS component available into the tool bar. We set CLASS as TARGET, the other columns as INPUT (V1...V21).



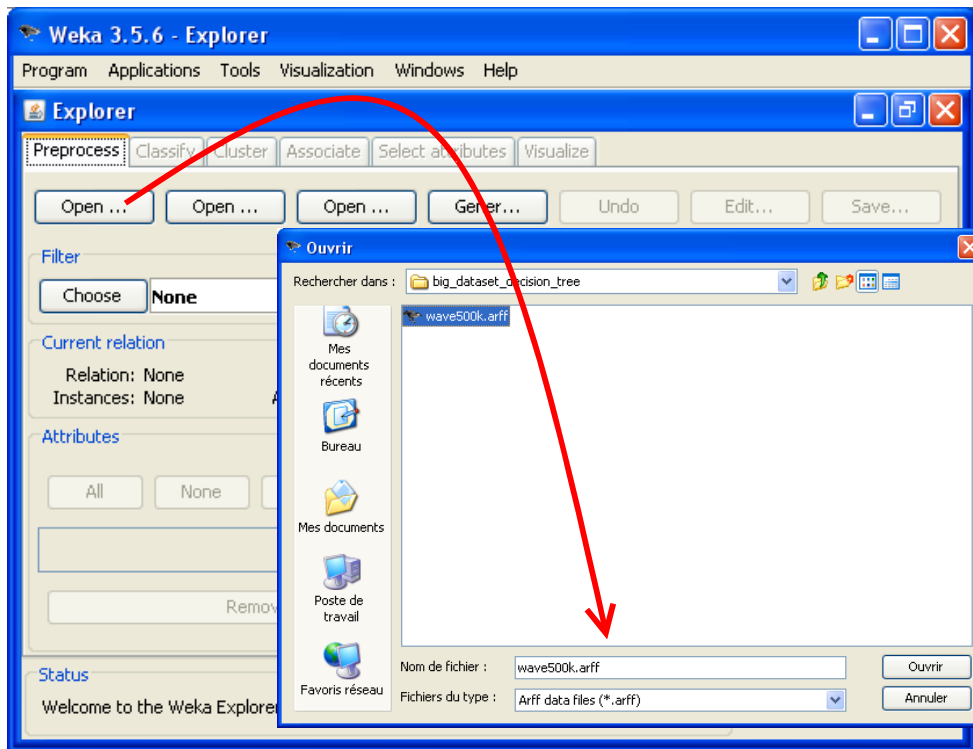
We can now insert the C4.5 component and set the adequate parameters.



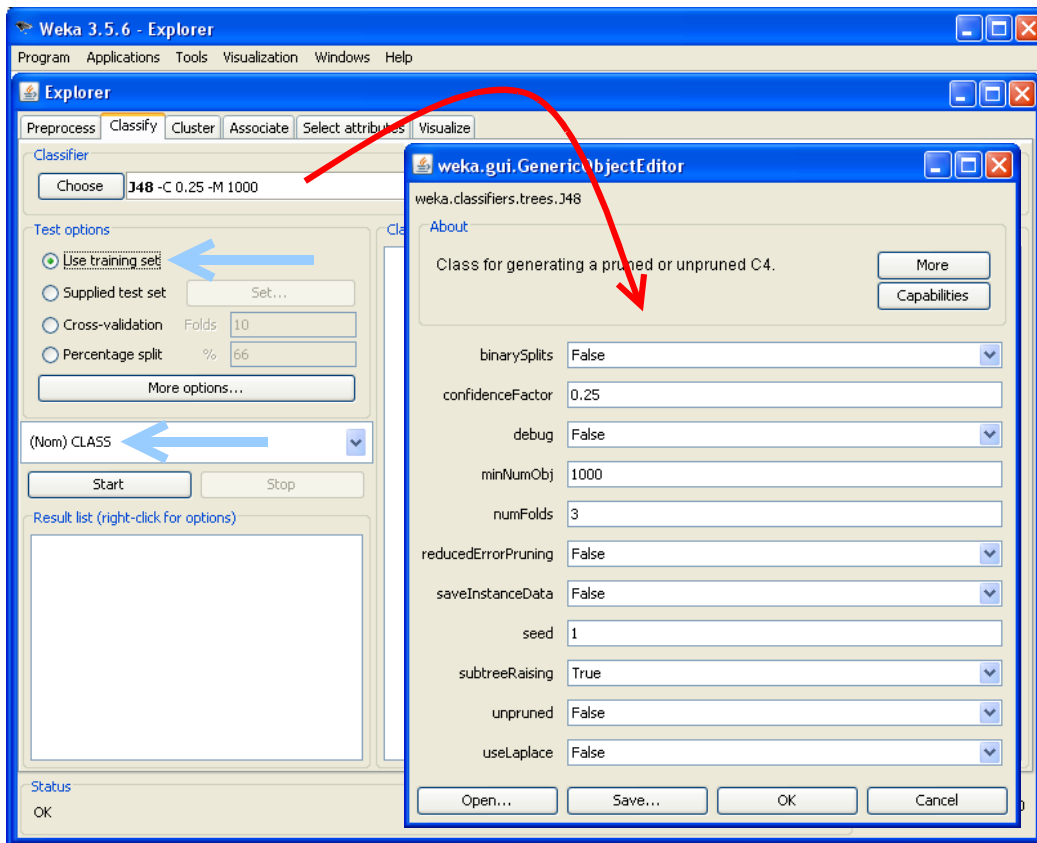
The contextual menu VIEW launches the calculations. The processing time is 33 seconds. The maximal memory occupation during the computation is 121.6 MB, it is 73.5MB when the tree is finished. The decision tree includes 117 leaves.

3.7 WEKA

We use the EXPLORER module for WEKA. We click on the OPEN button of the PREPROCESS tab. We select the WAVE500K.ARFF data file.



The processing time is 10 seconds; the memory occupation becomes 253.2 MB.



When the dataset is loaded, we select the CLASSIFY tab. We choose the J48 decision tree algorithm, we set the adequate parameters. We can now click on the START button.

The processing time is 338 seconds. The memory occupation becomes 699.6 MB. The tree includes 123 leaves.

3.8 Summary

The results are summarized into the following table:

Program	Computation time (seconds)		Memory occupation (MB)			
	Data Importation	Tree induction	After launch	After importation	Max during treatment	After induction
KNIME	47	270	92.6	160.4	245.8	245.8
ORANGE	90	130	24.9	259.5	795.7	795.7
R (package rpart)	24	157	18.8	184.1	718.9	718.9
RAPIDMINER	7	298	136.3	228.1	1274.4	1274.4
SIPINA	25	122	7.8	67.1	539.9	539.9
TANAGRA	11	33	7.0	53.1	121.6	73.5
WEKA	10	338	52.3	253.2	699.6	699.6

Tanagra is much faster than other programs in the decision tree induction context. This is not surprising when we know the internal data structure of Tanagra. The data are organized in columns in memory. It is very performing when we treated individually variable, when searching the discretization cut point for the split operation during the tree induction for instance.

On the other hand, in an upcoming tutorial, we see that this advantage becomes a drawback when we need to handle simultaneously all the variables during the induction process, when we perform a scalar product during the SVM (Support Vector Machine) induction for instance. In this case, Tanagra is clearly outperformed by other programs such as RAPIDMINER or WEKA.

4 Conclusion

This tutorial is one of the possible point views of the decision tree induction with free software in the context of large dataset.

Of course, others point of views are possible: more predictors, a mixture of continuous and discrete predictors, more examples, etc. Because the programs are freely available, everyone can define and evaluate the configuration related to their preoccupation, and use their own dataset. Finally, this is what matters.