1 Theme

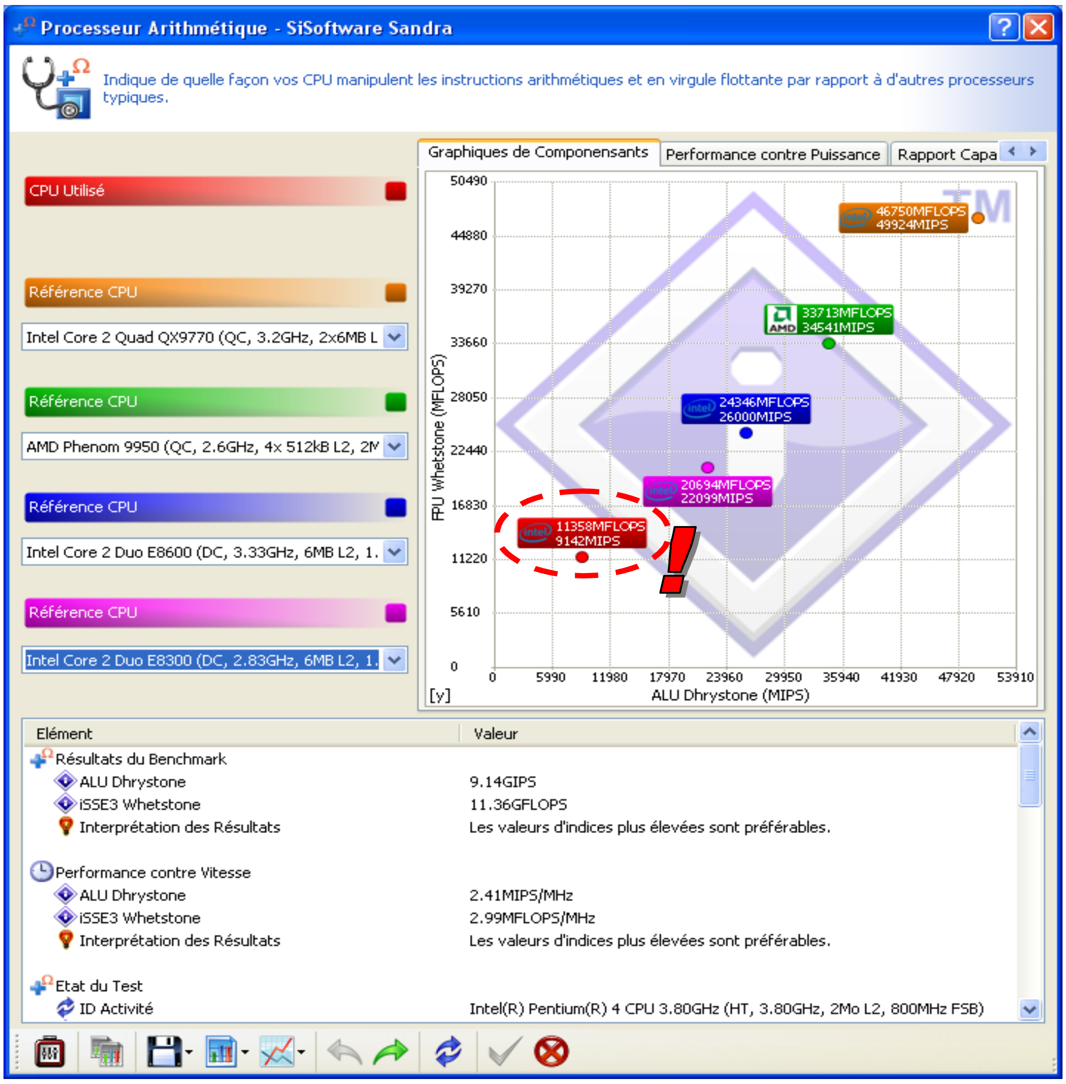
Comparing the quickness and memory occupation of various free implementation of the C4.5 (or assimilated) decision tree algorithm on large dataset.

2011/10/29 - The results are updated from a new computer (Q9400 core 2 quad processor) and a new operating system (Windows 7 - 64 bit) – See section 5.

Dealing with large dataset is one of the most important challenges 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 C4.5 algorithm (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:



Indique de quelle façon vos CPU manipulent les instructions arithmétiques et en virgule flottante par rapport à d'autres processeurs typiques.

CPU Utilisé

Référence CPU: Intel Core 2 Quad QX9770 (QC, 3.2GHz, 2x6MB L

Référence CPU: AMD Phenom 9950 (QC, 2.6GHz, 4x 512kB L2, 2M

Référence CPU: Intel Core 2 Duo E8600 (DC, 3.33GHz, 6MB L2, 1.

Référence CPU: Intel Core 2 Duo E8300 (DC, 2.83GHz, 6MB L2, 1.

Elément	Valeur
Résultats du Benchmark	
ALU Dhrystone	9.14GIPS
iSSE3 Whetstone	11.36GFLOPS
Interprétation des Résultats	Les valeurs d'indices plus élevées sont préférables.
Performance contre Vitesse	
ALU Dhrystone	2.41MIPS/MHz
iSSE3 Whetstone	2.99MFLOPS/MHz
Interprétation des Résultats	Les valeurs d'indices plus élevées sont préférables.
Etat du Test	
ID Activité	Intel(R) Pentium(R) 4 CPU 3.80GHz (HT, 3.80GHz, 2Mo L2, 800MHz FSB)

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

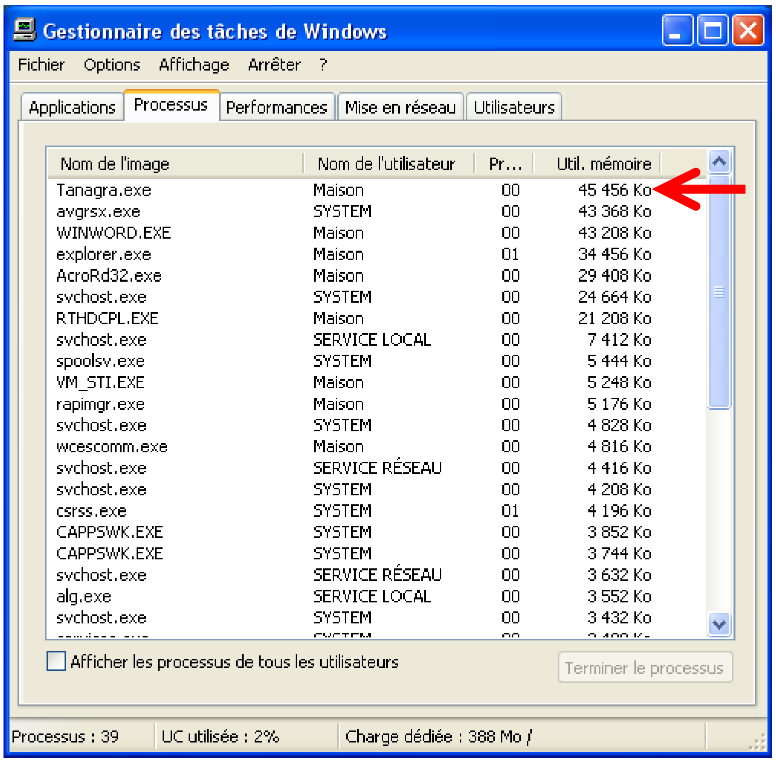
Software	**Version**	**URL**
KNIME	1.3.5	http://www.knime.org/index.html
ORANGE	1.0b2	http://www.ailab.si/orange/
R (package rpart)	2.6.0	http://www.r-project.org/
RAPIDMINER (YALE)	Community Edition	http://rapid-i.com/
SIPINA	Research	http://eric.univ-lyon2.fr/~ricco/sipina.html
TANAGRA	1.4.27	http://eric.univ-lyon2.fr/~ricco/tanagra/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


Gestionnaire des tâches de Windows
Fichier Options Affichage Arrêter ?
Applications Processus Performances Mise en réseau Utilisateurs
Afficher les processus de tous les utilisateurs
Terminer le processus

Nom de l'image	Nom de l'utilisateur	Pr...	Util. mémoire
Tanagra.exe	Maison	00	45 456 Ko
avgrsx.exe	SYSTEM	00	43 368 Ko
WINWORD.EXE	Maison	00	43 208 Ko
explorer.exe	Maison	01	34 456 Ko
AcroRd32.exe	Maison	00	29 408 Ko
svchost.exe	SYSTEM	00	24 664 Ko
RTHDCPL.EXE	Maison	00	21 208 Ko
svchost.exe	SERVICE LOCAL	00	7 412 Ko
spoolsv.exe	SYSTEM	00	5 444 Ko
VM_STI.EXE	Maison	00	5 248 Ko
rapimgr.exe	Maison	00	5 176 Ko
svchost.exe	SYSTEM	00	4 828 Ko
wcescomm.exe	Maison	00	4 816 Ko
svchost.exe	SERVICE RÉSEAU	00	4 416 Ko
svchost.exe	SYSTEM	00	4 208 Ko
csrss.exe	SYSTEM	01	4 196 Ko
CAPPSWK.EXE	SYSTEM	00	3 852 Ko
CAPPSWK.EXE	SYSTEM	00	3 744 Ko
svchost.exe	SERVICE RÉSEAU	00	3 632 Ko
alg.exe	SERVICE LOCAL	00	3 552 Ko
svchost.exe	SYSTEM	00	3 432 Ko

Processus : 39 UC utilisée : 2% Charge dédiée : 388 Mo /

Figure 1 – Measuring the memory occupation under Windows OS

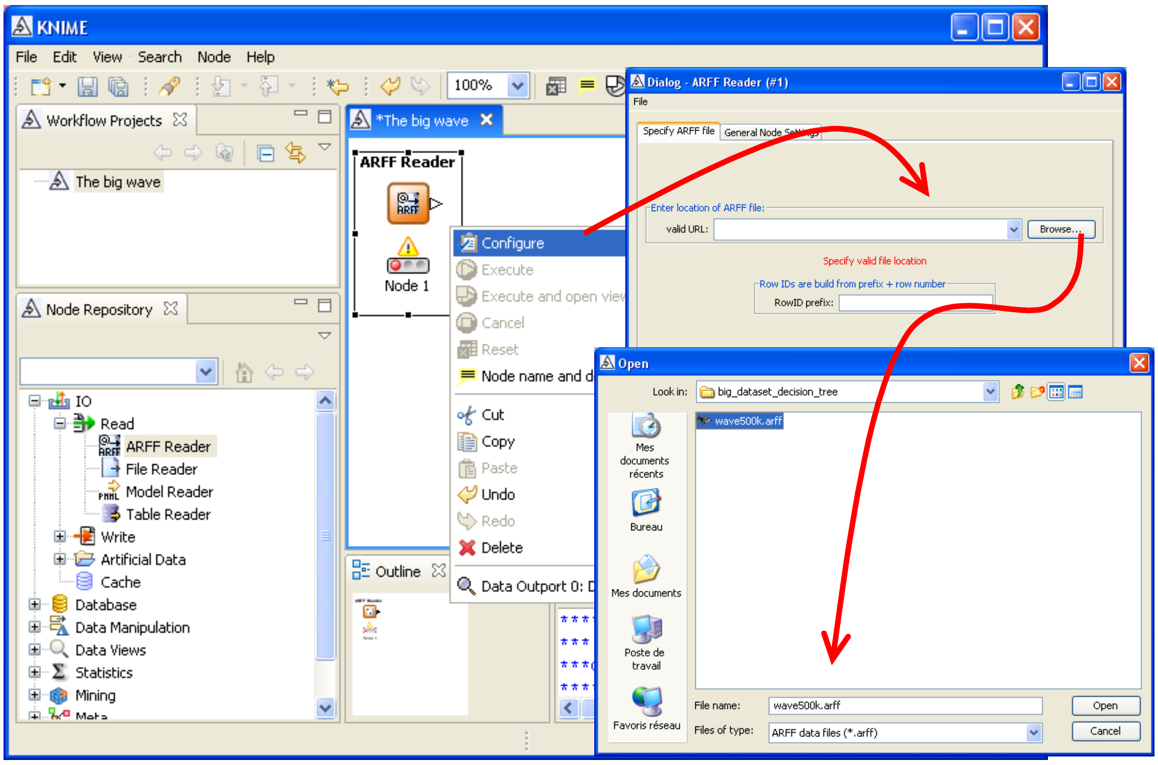
We utilize the indications of Windows OS about the memory occupation (Figure 1). We measure the memory occupation when the program is launched, when the dataset is loaded, during the learning phase, after the learning phase. The maximum occupation during the learning phase is the most important indicator. 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 to obtain only an order of magnitude 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.

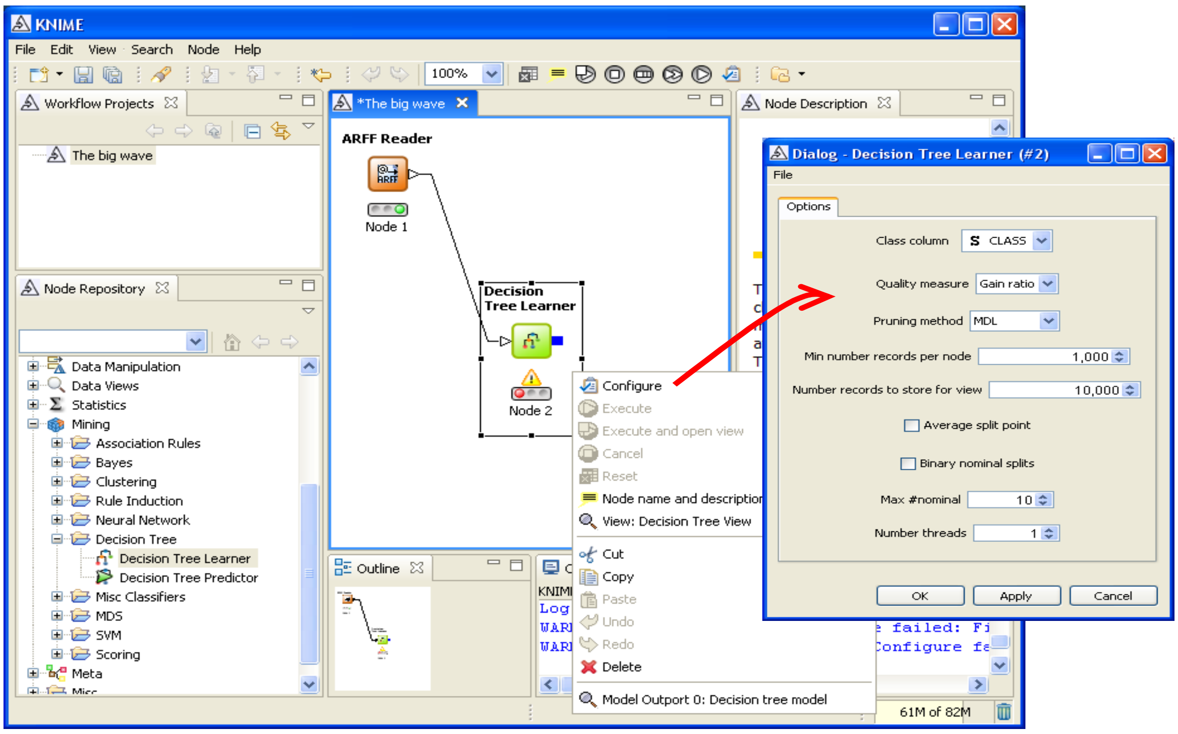
KNIME and the JRE. When we launch the software, the JRE (Java Runtime Environment) is started and the global memory occupation is 92.6 MB.



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.

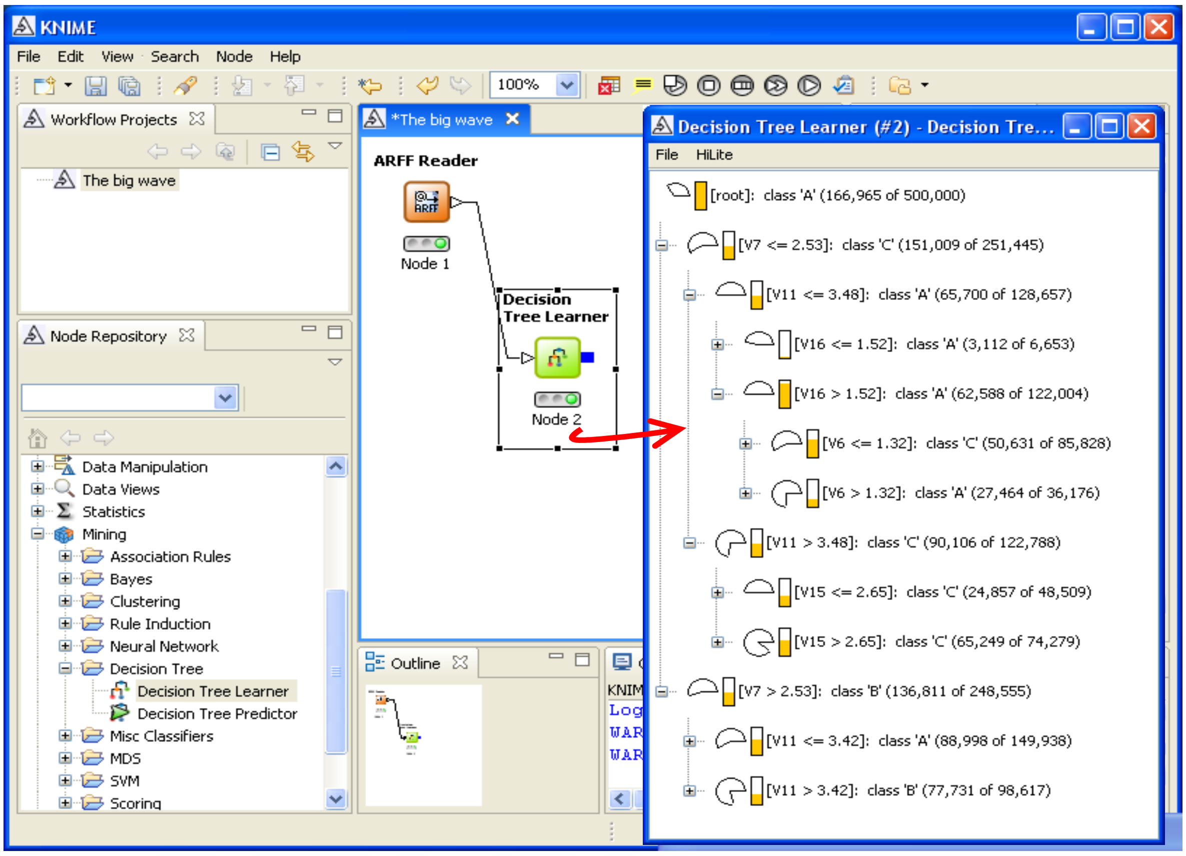
We insert the decision tree LEARNER in the workflow, we see on the panel at the right a description of the method and its parameters. We connect the component ARFF READER, then we click on the menu CONFIGURE in order to define the parameters.

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.



Class column: S CLASS
Quality measure: Gain ratio
Pruning method: MDL
Min number records per node: 1,000
Number records to store for view: 10,000
☐ Average split point
☐ Binary nominal splits
Max #nominal: 10
Number threads: 1

We launch the calculation by clicking on the EXECUTE AND OPEN VIEW menu. Knime does not give indications about the tree structure (number of nodes and leaves). The processing time is 270 seconds; the maximum memory occupation is 245.8 MB during the learning phase.



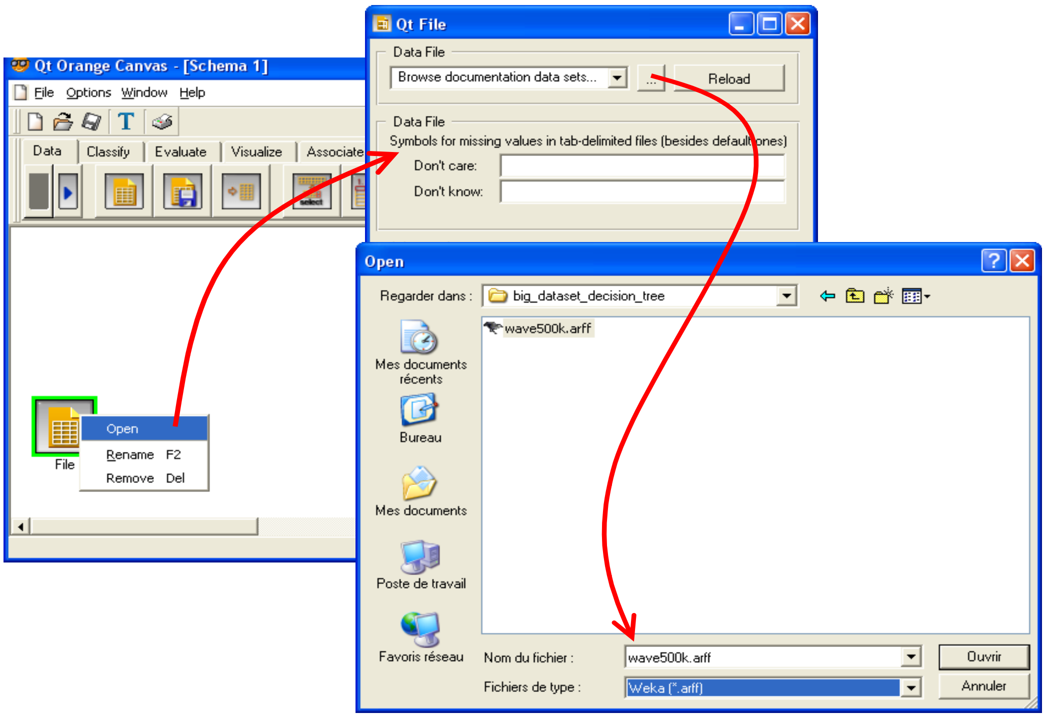
```
[root]: class 'A' (166,965 of 500,000)
  [V7 <= 2.53]: class 'C' (151,009 of 251,445)
    [V11 <= 3.48]: class 'A' (65,700 of 128,657)
      [V16 <= 1.52]: class 'A' (3,112 of 6,653)
      [V16 > 1.52]: class 'A' (62,588 of 122,004)
        [V6 <= 1.32]: class 'C' (50,631 of 85,828)
        [V6 > 1.32]: class 'A' (27,464 of 36,176)
    [V11 > 3.48]: class 'C' (90,106 of 122,788)
      [V15 <= 2.65]: class 'C' (24,857 of 48,509)
      [V15 > 2.65]: class 'C' (65,249 of 74,279)
  [V7 > 2.53]: class 'B' (136,811 of 248,555)
    [V11 <= 3.42]: class 'A' (88,998 of 149,938)
    [V11 > 3.42]: class 'B' (77,731 of 98,617)
```

It is possible to save the workflow. But, it seems that the software tries to store all the available information, including the dataset. The processing time can be very important for our analysis.

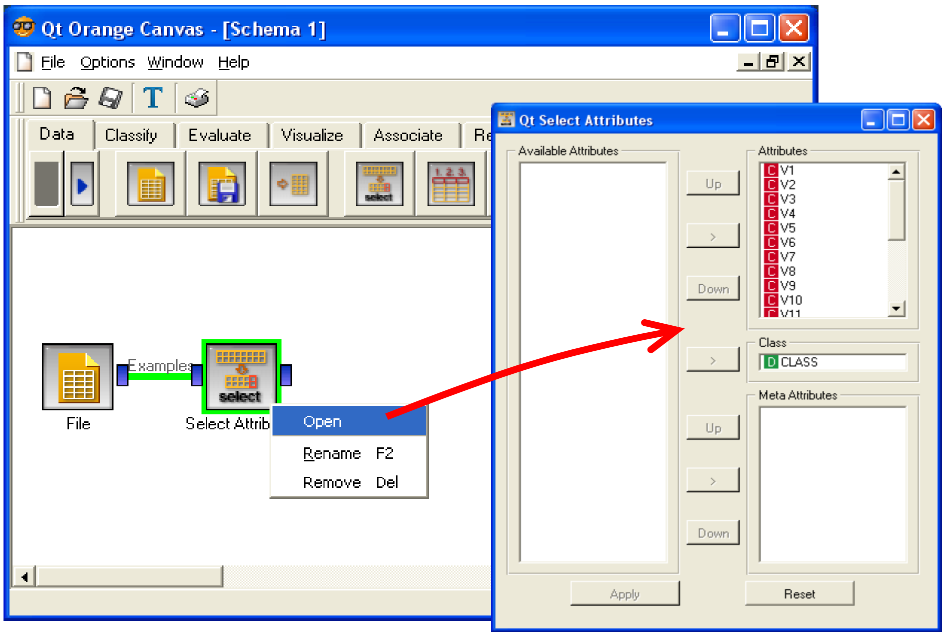
3.2 ORANGE

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

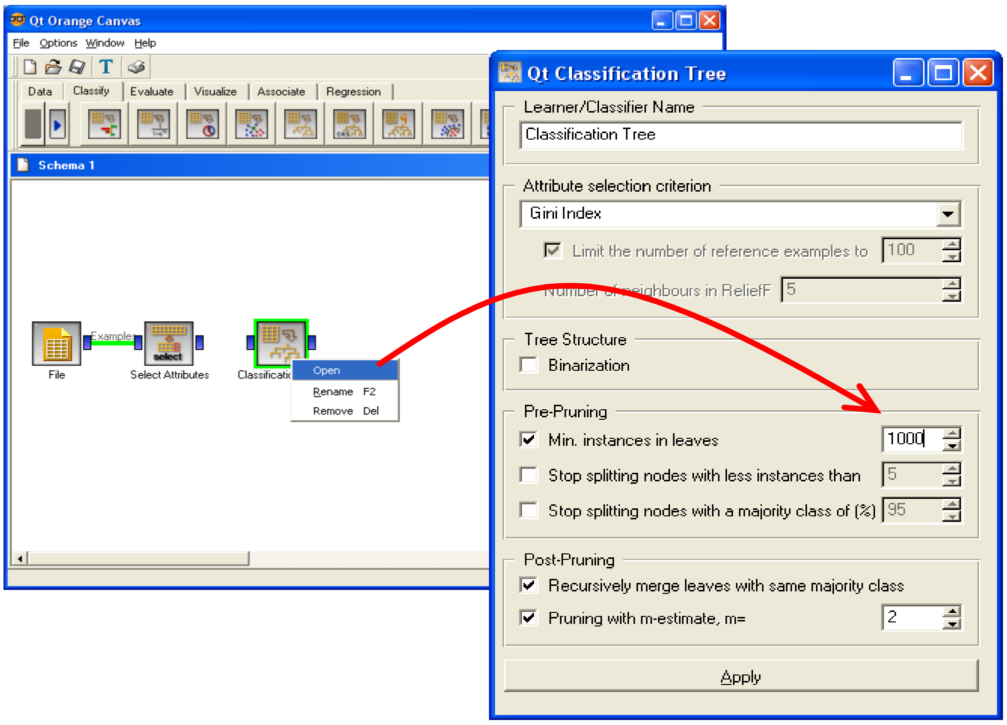
When we launch ORANGE, the **Python Runtime Environment** is started. The global memory occupation is 24.9 MB. We have a familiar interface. The treatments are defined in a schema. The components are available into the tool bar. We add the FILE (DATA tab) into the schema. We click on the OPEN menu and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


Qt Orange Canvas - [Schema 1]
File Options Window Help
Data Classify Evaluate Visualize Associate
Open
Rename F2
Remove Del
File
Qt File
Data File
Browse documentation data sets...
Reload
Symbols for missing values in tab-delimited files (besides default ones)
Don't care:
Don't know:
Open
Regarder dans : big_dataset_decision_tree
wave500k.arff
Mes documents récents
Bureau
Mes documents
Poste de travail
Favoris réseau
Nom du fichier : wave500k.arff
Fichiers de type : Weka (*.arff)
Ouvrir
Annuler

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** we make the connection. We click on the OPEN MENU. We set the parameters of the algorithm.



Qt Classification Tree

Learner/Classifier Name: Classification Tree

Attribute selection criterion: Gini Index

- [✓] Limit the number of reference examples to 100
- Number of neighbours in ReliefF 5

Tree Structure

- [ ] Binarization

Pre-Pruning

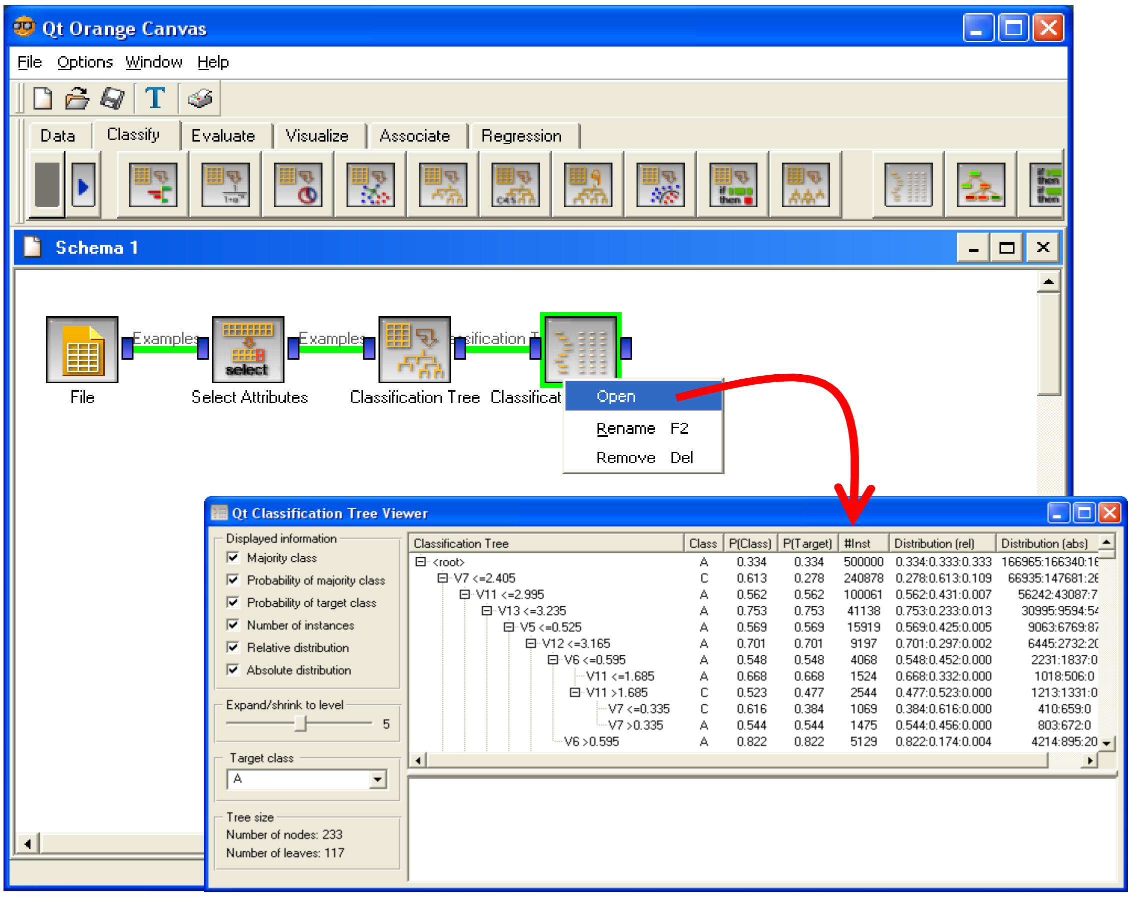
- [✓] Min. instances in leaves 1000
- [ ] Stop splitting nodes with less instances than 5
- [ ] Stop splitting nodes with a majority class of (%) 95

Post-Pruning

- [✓] Recursively merge leaves with same majority class
- [✓] Pruning with m-estimate, m= 2

Apply

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.



Qt Classification Tree Viewer

Displayed information

- [✓] Majority class
- [✓] Probability of majority class
- [✓] Probability of target class
- [✓] Number of instances
- [✓] Relative distribution
- [✓] Absolute distribution

Expand/shrink to level 5

Target class: A

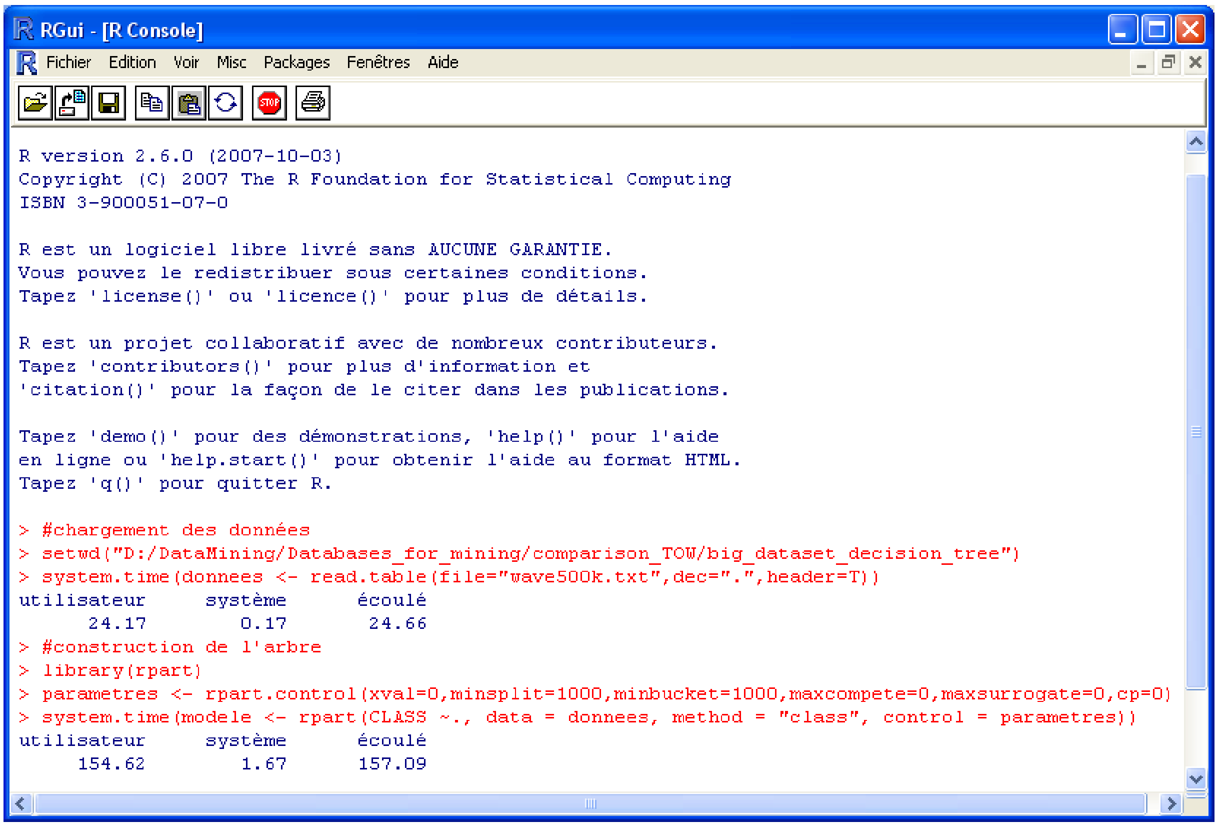
Tree size
Number of nodes: 233
Number of leaves: 117

Classification Tree	Class	P(Class)	P(Target)	#Inst	Distribution (rel)	Distribution (abs)
⊟ <root>	A	0.334	0.334	500000	0.334:0.333:0.333	166965:166340:1
⊟ V7 <=2.405	C	0.613	0.278	240878	0.278:0.613:0.109	66935:147681:2
⊟ V11 <=2.995	A	0.562	0.562	100061	0.562:0.431:0.007	56242:43087:7
⊟ V13 <=3.235	A	0.753	0.753	41138	0.753:0.233:0.013	30995:9594:5
⊟ V5 <=0.525	A	0.569	0.569	15919	0.569:0.425:0.005	9063:6769:87
⊟ V12 <=3.165	A	0.701	0.701	9197	0.701:0.297:0.002	6445:2732:2
⊟ V6 <=0.595	A	0.548	0.548	4068	0.548:0.452:0.000	2231:1837:0
V11 <=1.685	A	0.668	0.668	1524	0.668:0.332:0.000	1018:506:0
⊟ V11 >1.685	C	0.523	0.477	2544	0.477:0.523:0.000	1213:1331:0
V7 <=0.335	C	0.616	0.384	1069	0.384:0.616:0.000	410:659:0
V7 >0.335	A	0.544	0.544	1475	0.544:0.456:0.000	803:672:0
V6 >0.595	A	0.822	0.822	5129	0.822:0.174:0.004	4214:895:20

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.



```
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.

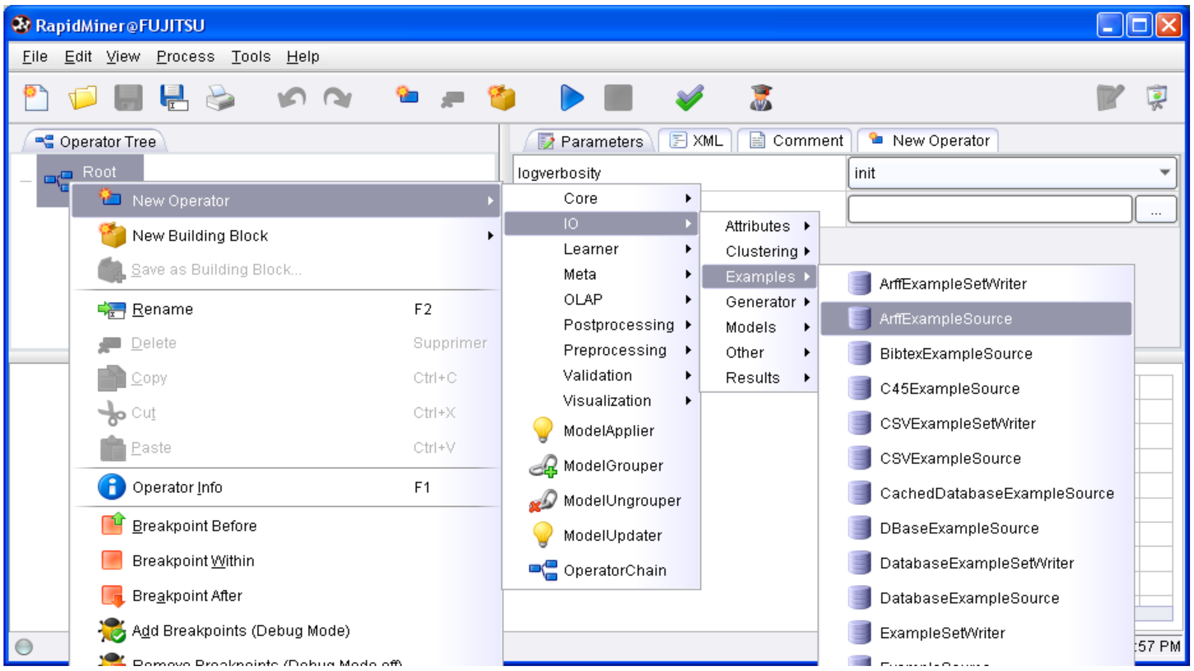
The RPART procedure is rather similar to CART method. But we set the parameters in order to produce a tree similar to the others programs i.e. MINSPLIT = 1000 observations; MINBUCKET = 1000 observations; CP = 0. There is not post-pruning here. But, it does not matter. We know that this part of the processing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

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

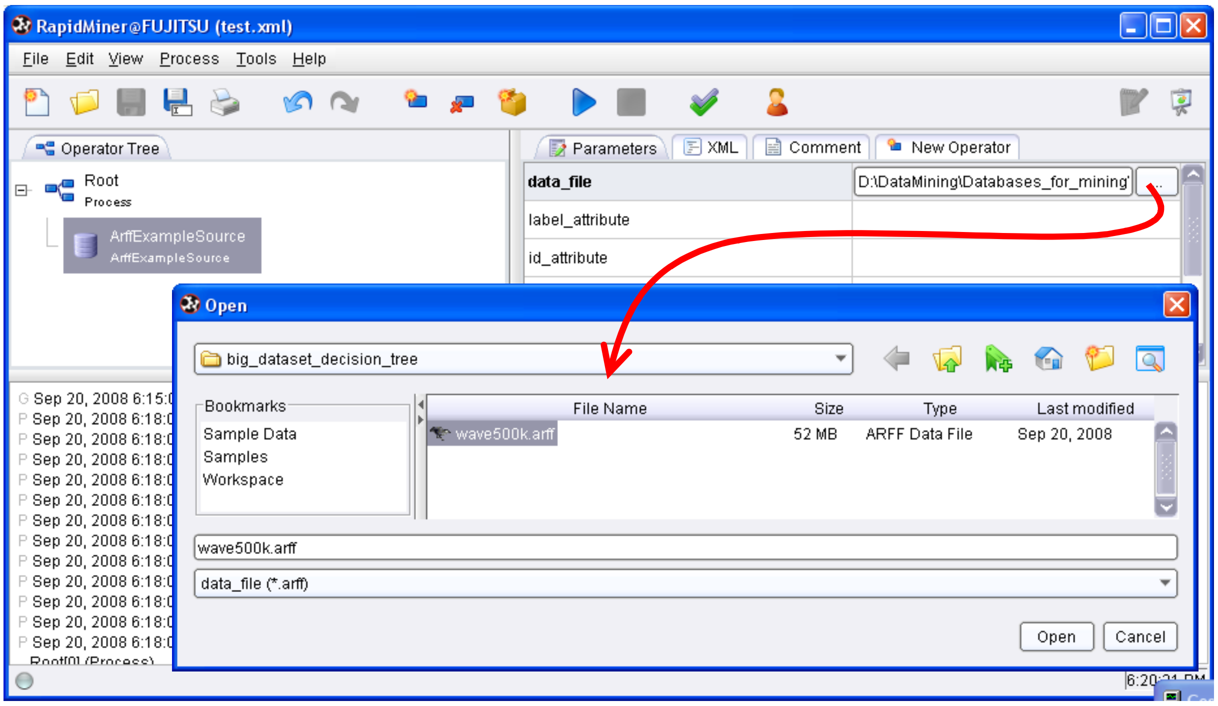
3.4 RAPIDMINER (formerly YALE)

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

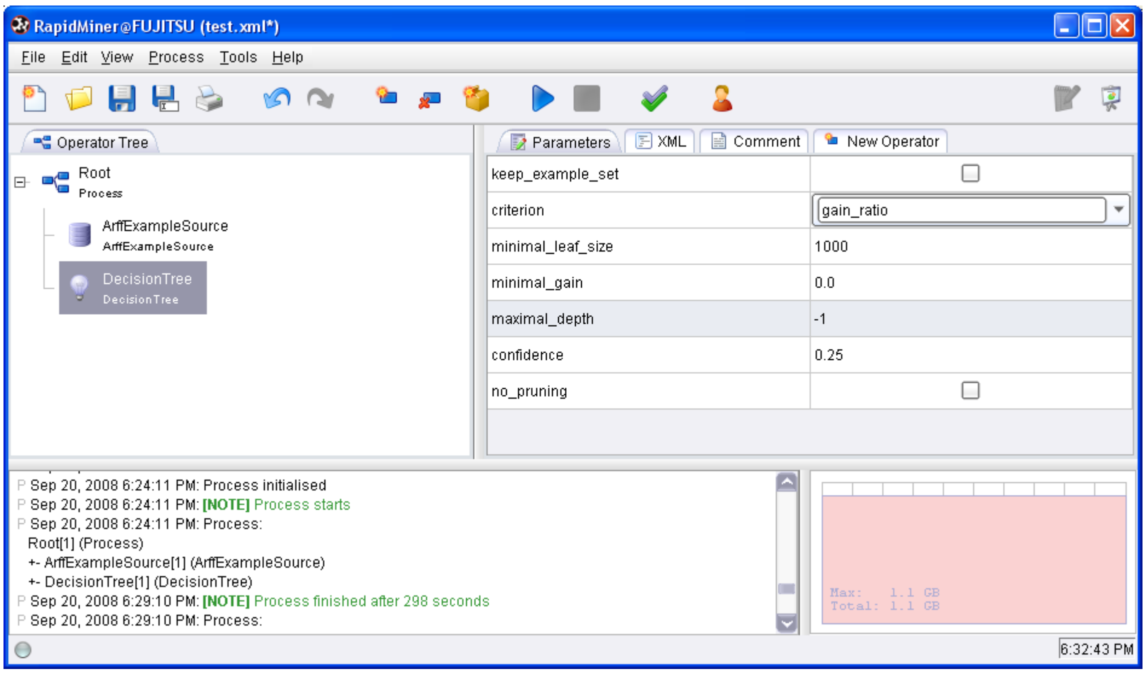
We add an ARFFEXAMPLESOURCE component into the tree, in order to load the dataset.



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.


RapidMiner@FUJITSU (test.xml)
File Edit View Process Tools Help
Operator Tree
Root
Process
ArffExampleSource
ArffExampleSource
Parameters XML Comment New Operator
data_file D:\DataMining\Databases_for_mining\
label_attribute
id_attribute
Open
big_dataset_decision_tree
Bookmarks
Sample Data
Samples
Workspace
File Name Size Type Last modified
wave500k.arff 52 MB ARFF Data File Sep 20, 2008
wave500k.arff
data_file (*.arff)
Open Cancel

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.



keep_example_set	☐
criterion	gain_ratio
minimal_leaf_size	1000
minimal_gain	0.0
maximal_depth	-1
confidence	0.25
no_pruning	☐

```
P Sep 20, 2008 6:24:11 PM: Process initialised
P Sep 20, 2008 6:24:11 PM: [NOTE] Process starts
P Sep 20, 2008 6:24:11 PM: Process:
  Root[1] (Process)
  +- ArffExampleSource[1] (ArffExampleSource)
  +- DecisionTree[1] (DecisionTree)
P Sep 20, 2008 6:29:10 PM: [NOTE] Process finished after 298 seconds
P Sep 20, 2008 6:29:10 PM: Process:
```

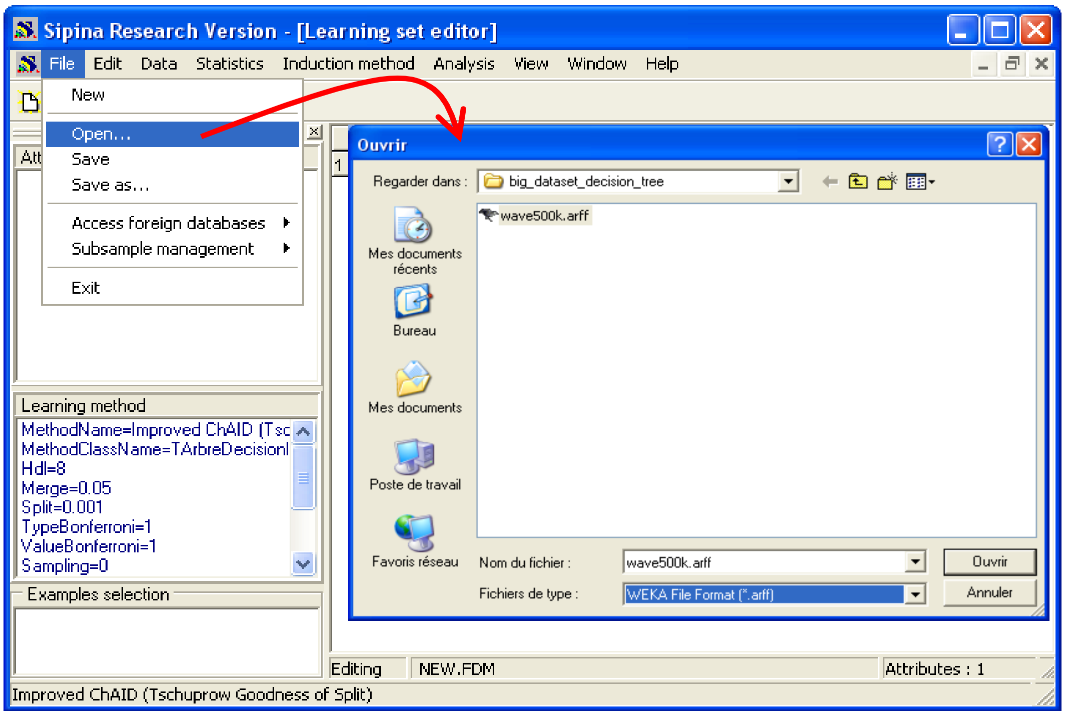
The PLAY button starts again all the computations. We obtain the decision tree after 298 seconds.

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.

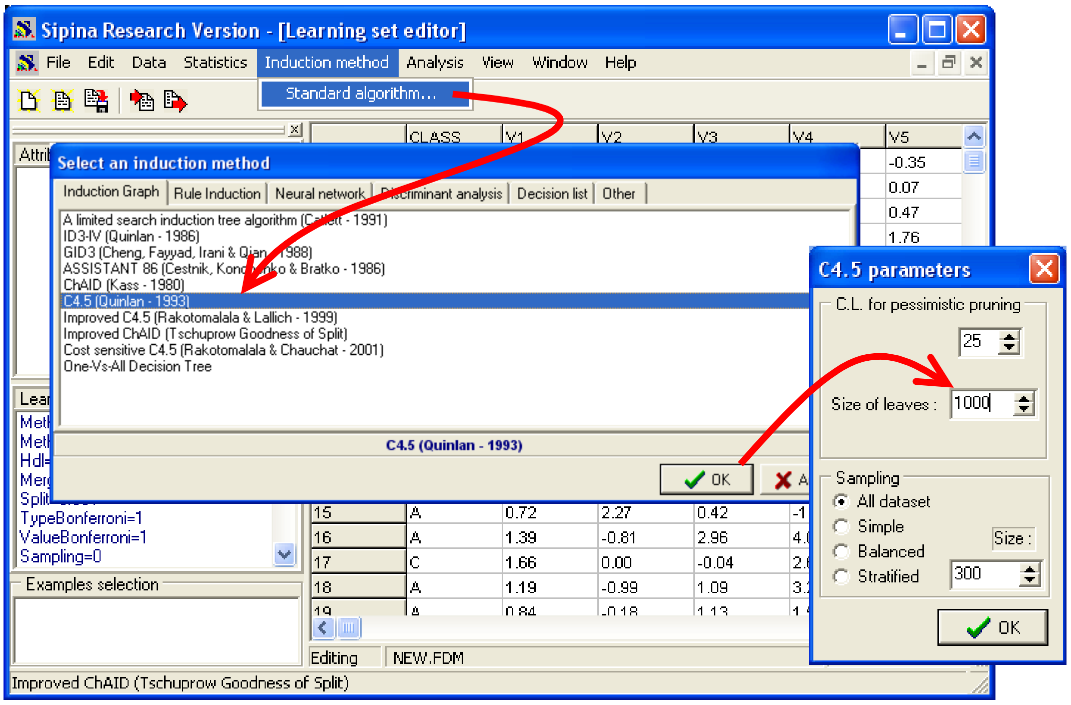


```
MethodName=Improved ChAID (Tsc
MethodClassName=TArbreDecisionI
Hdl=8
Merge=0.05
Split=0.001
TypeBonferroni=1
ValueBonferroni=1
Sampling=0
```

Improved ChAID (Tschuprow Goodness of Split)

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.


Sipina Research Version - [Learning set editor]
File Edit Data Statistics Induction method Analysis View Window Help
Standard algorithm...
Select an induction method
Induction Graph | Rule Induction | Neural network | Discriminant analysis | Decision list | Other
C4.5 parameters
OK
Editing NEW.FDM

A limited search induction tree algorithm (Catlett - 1991)
ID3-IV (Quinlan - 1986)
GID3 (Cheng, Fayyad, Irani & Qian - 1988)
ASSISTANT 86 (Cestnik, Kononenko & Bratko - 1986)
ChAID (Kass - 1980)
C4.5 (Quinlan - 1993)
Improved C4.5 (Rakotomalala & Lallich - 1999)
Improved ChAID (Tschuprow Goodness of Split)
Cost sensitive C4.5 (Rakotomalala & Chauchat - 2001)
One-Vs-All Decision Tree

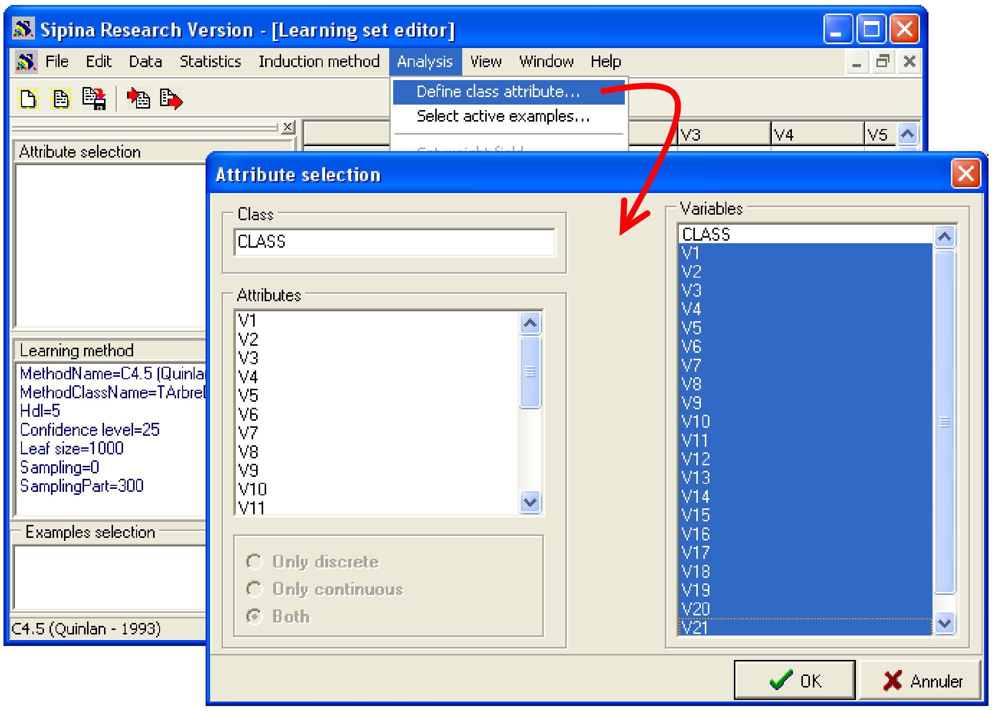
C4.5 (Quinlan - 1993)

C.L. for pessimistic pruning: 25
Size of leaves : 1000
Sampling: [✓ All dataset] Simple Balanced Stratified
Size : 300

TypeBonferroni=1
ValueBonferroni=1
Sampling=0

Improved ChAID (Tschuprow Goodness of Split)

We must specify the role 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.


Sipina Research Version - [Learning set editor]
File Edit Data Statistics Induction method Analysis View Window Help
Define class attribute...
Select active examples...
Attribute selection
Learning method
Examples selection
OK
Annuler

MethodName=C4.5 (Quinla
MethodClassName=TArbre
Hdl=5
Confidence level=25
Leaf size=1000
Sampling=0
SamplingPart=300

C4.5 (Quinlan - 1993)

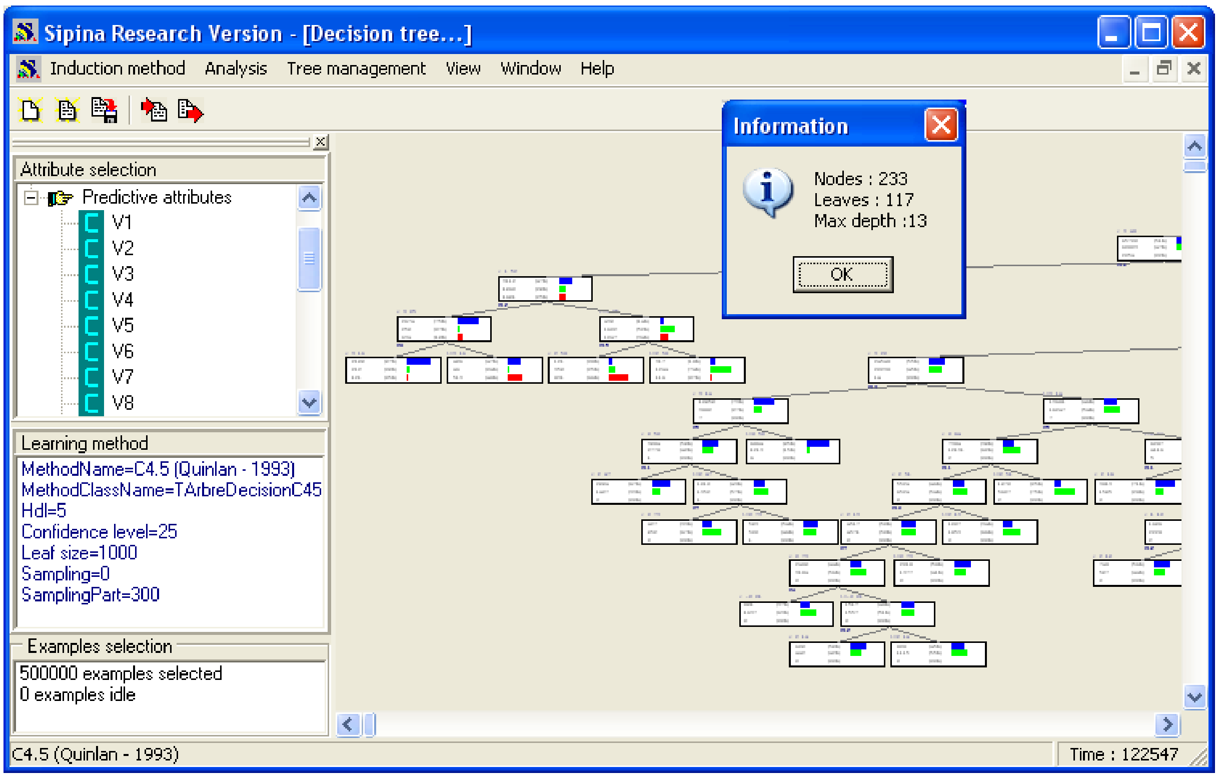
Class: CLASS

Attributes: V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11

Only discrete / Only continuous / [✓ Both]

Variables: CLASS V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21

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.



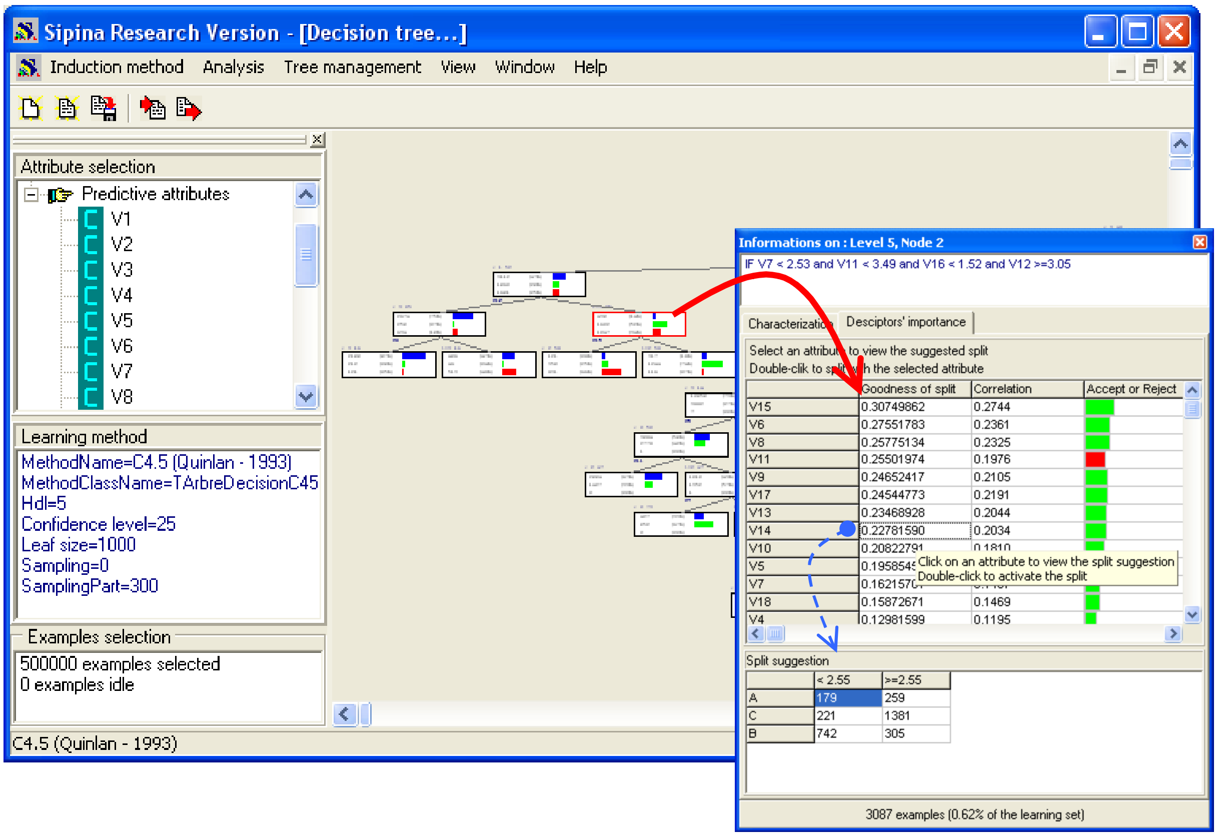
Nodes : 233
Leaves : 117
Max depth :13

MethodName=C4.5 (Quinlan - 1993)
MethodClassName=TArbreDecisionC45
Hdl=5
Confidence level=25
Leaf size=1000
Sampling=0
SamplingPart=300

500000 examples selected
0 examples idle

C4.5 (Quinlan - 1993)
Time : 122547

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



MethodName=C4.5 (Quinlan - 1993)
MethodClassName=TArbreDecisionC45
Hdl=5
Confidence level=25
Leaf size=1000
Sampling=0
SamplingPart=300

500000 examples selected
0 examples idle

C4.5 (Quinlan - 1993)

IF V7 < 2.53 and V11 < 3.49 and V16 < 1.52 and V12 >=3.05

Select an attribute to view the suggested split
Double-clik to split with the selected attribute

	Goodness of split	Correlation	Accept or Reject
V15	0.30749862	0.2744	🟩
V6	0.27551783	0.2361	🟩
V8	0.25775134	0.2325	🟩
V11	0.25501974	0.1976	🟥
V9	0.24652417	0.2105	🟩
V17	0.24544773	0.2191	🟩
V13	0.23468928	0.2044	🟩
V14	0.22781590	0.2034	🟩
V10	0.20822791	0.1810	🟩
V5	0.1958545		
V7	0.1621570		🟩
V18	0.15872671	0.1469	🟩
V4	0.12981599	0.1195	🟩

Click on an attribute to view the split suggestion
Double-click to activate the split

	< 2.55	>=2.55
A	179	259
C	221	1381
B	742	305

3087 examples (0.62% of the learning set)

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.

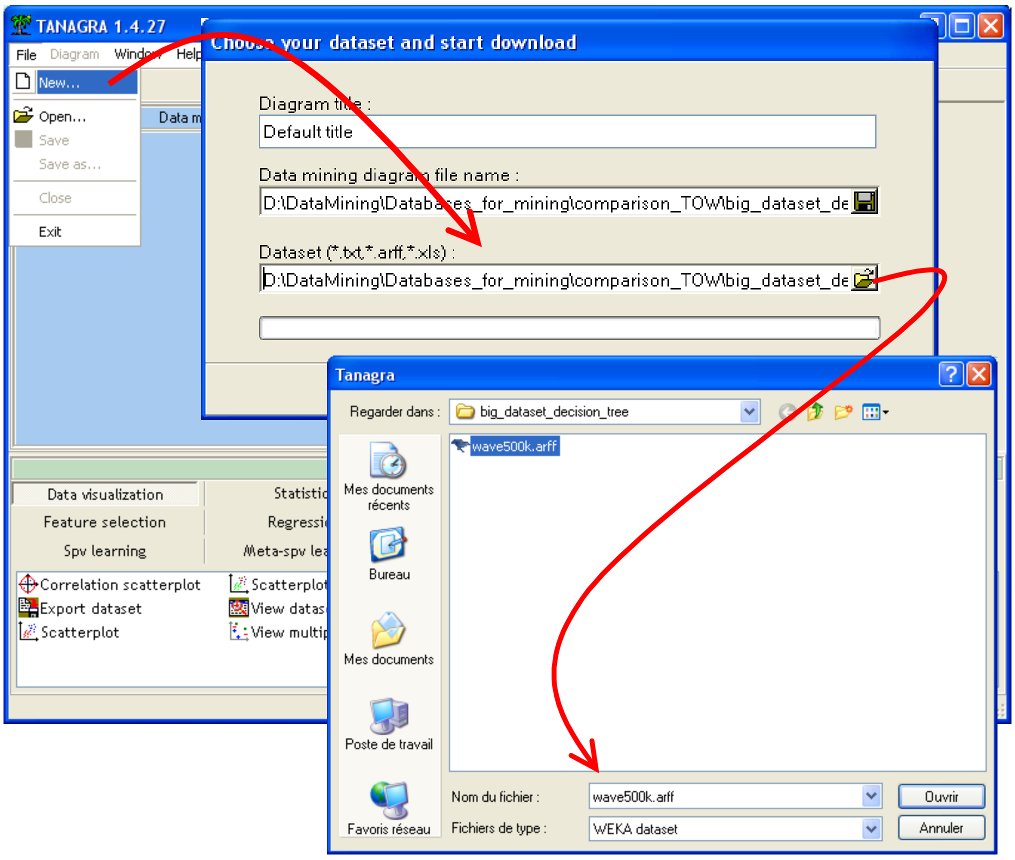


Diagram title : Default title

Data mining diagram file name : D:\DataMining\Databases_for_mining\comparison_TOW\big_dataset_de

Dataset (*.txt,*.arff,*.xls) : D:\DataMining\Databases_for_mining\comparison_TOW\big_dataset_de

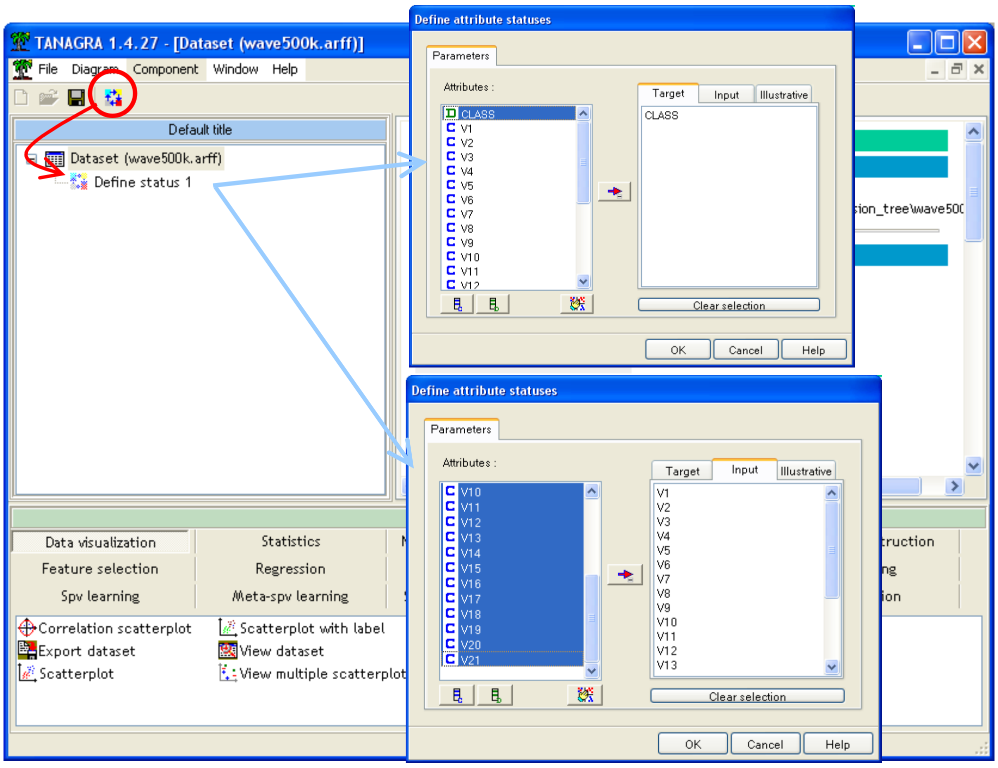
Regarder dans : big_dataset_decision_tree

Nom du fichier : wave500k.arff

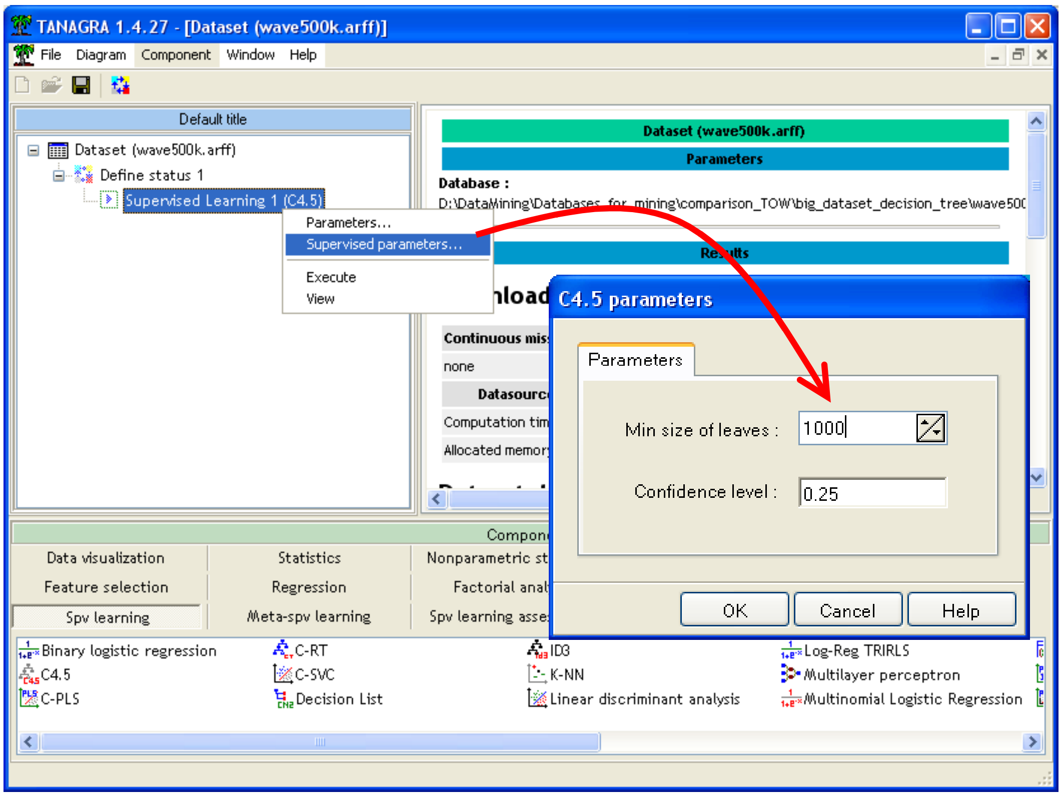
Fichiers de type : WEKA dataset

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).



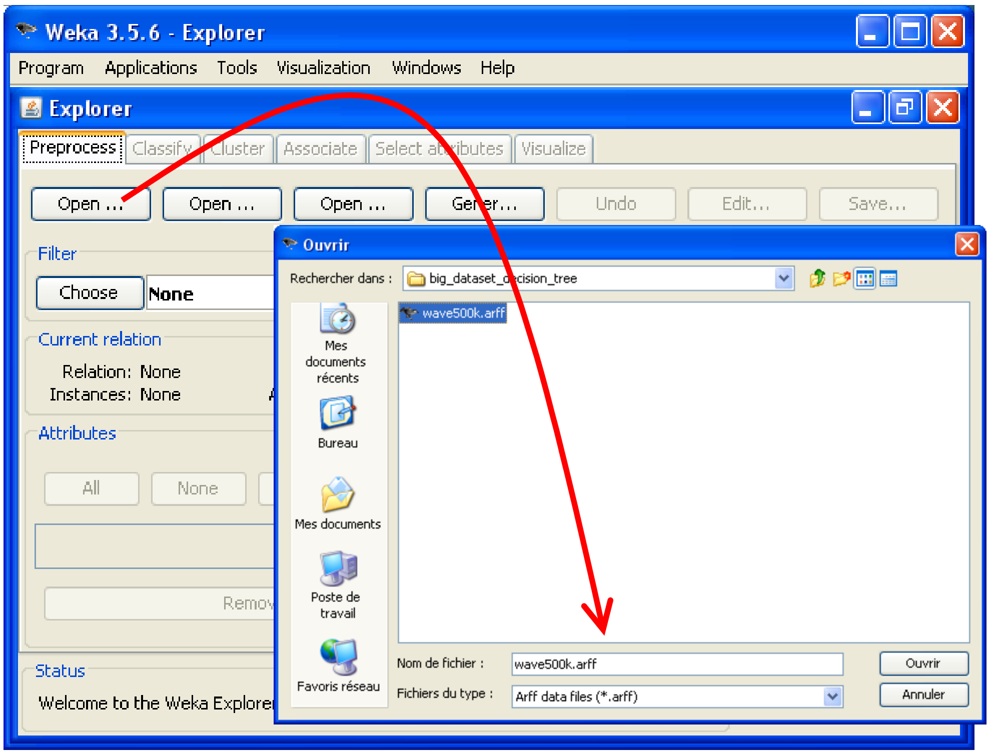
We can now insert the C4.5 component and set the adequate parameters.



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.



Relation: None
Instances: None

Rechercher dans : big_dataset_decision_tree

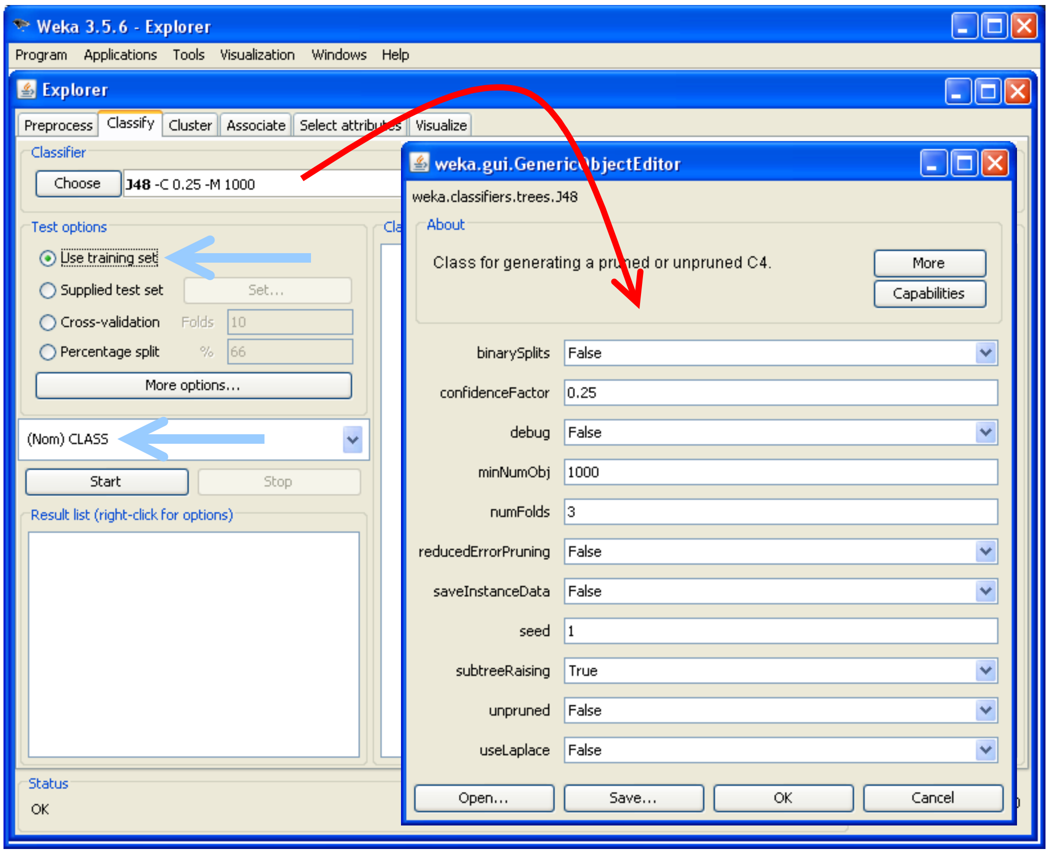
wave500k.arff

Nom de fichier : wave500k.arff

Fichiers du type : Arff data files (*.arff)

Welcome to the Weka Explorer

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


Weka 3.5.6 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
Percentage split
%
More options...
Start Stop
Result list (right-click for options)
Status
weka.gui.GenericObjectEditor
About
More
Capabilities
Open... Save... OK Cancel

J48 -C 0.25 -M 1000

Folds 10 · % 66

(Nom) CLASS

OK

weka.classifiers.trees.J48

Class for generating a pruned or unpruned C4.

binarySplits	False
confidenceFactor	0.25
debug	False
minNumObj	1000
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False

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.

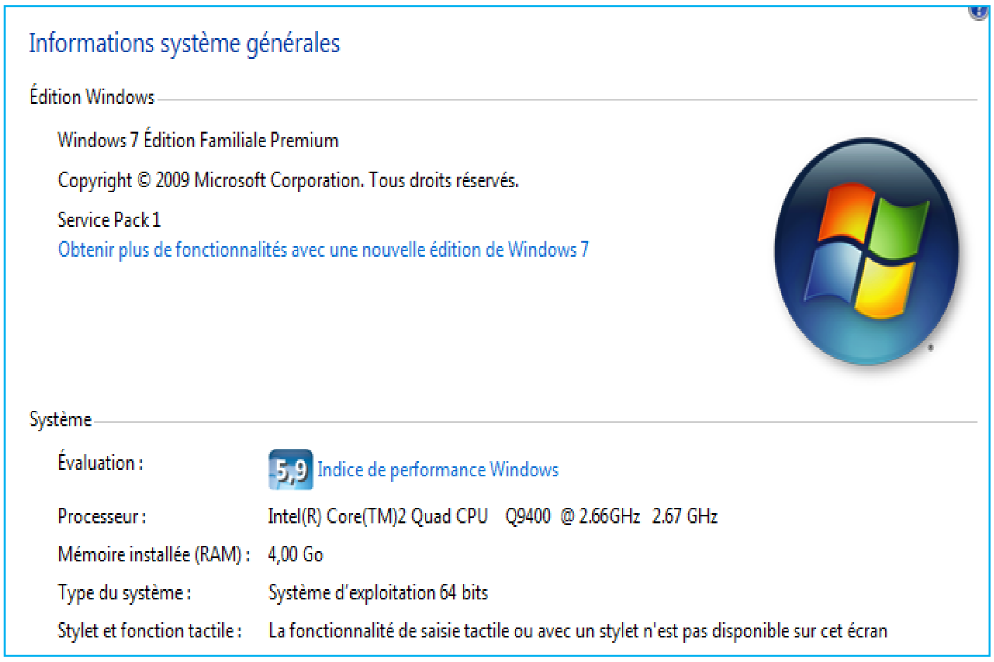
4 Conclusion

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.

5 Update – October 29, 2011

One of the exciting aspects of computing is that things are changing very quickly. The machines are ever more efficient, the operating systems are improved, the software also. Since writing this document, I have a new computer and I use a 64 bit OS (Windows 7). Some of the tools studied propose a 64 bit version (Knime, RapidMiner, R). I wonder how behave the various tools in this new context. To do that, I renewed the same experiment. We note that a more efficient computer allows to improve the computation time (about 20%). The specific gain for a 64 bit version is relatively low, but it is real (about 10%). And some tools are clearly improved their programming of the decision tree induction (Knime, RapidMiner). On the other hand, we observe that the memory occupation remains stable for the most of the tools in the new context.

So that everyone can compare the results on their own machine, here are the characteristics of my computer:



Informations système générales

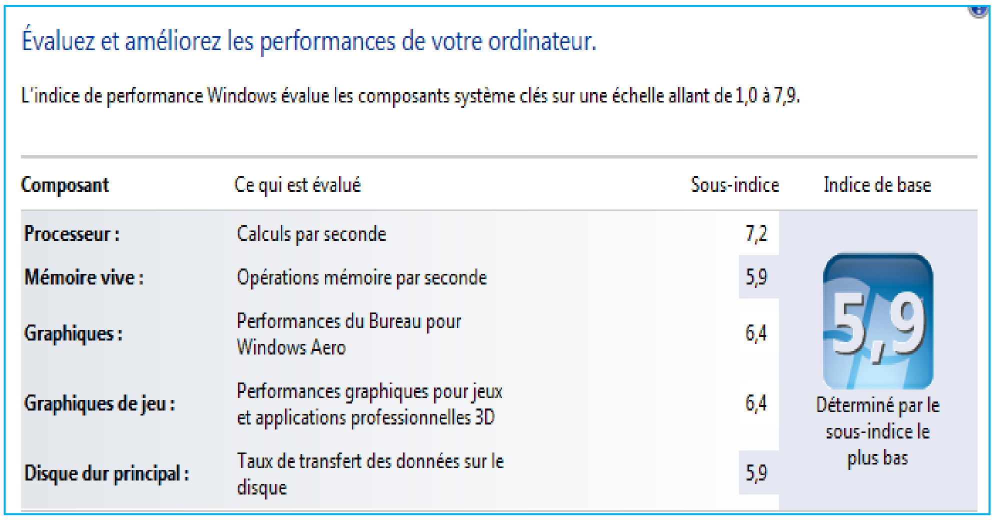
Édition Windows

Windows 7 Édition Familiale Premium
Copyright © 2009 Microsoft Corporation. Tous droits réservés.
Service Pack 1
Obtenir plus de fonctionnalités avec une nouvelle édition de Windows 7

Système

Évaluation :	5,9 Indice de performance Windows
Processeur :	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz 2.67 GHz
Mémoire installée (RAM) :	4,00 Go
Type du système :	Système d'exploitation 64 bits
Stylet et fonction tactile :	La fonctionnalité de saisie tactile ou avec un stylet n'est pas disponible sur cet écran

With the following performance index provided by Windows:



Évaluez et améliorez les performances de votre ordinateur.

L'indice de performance Windows évalue les composants système clés sur une échelle allant de 1,0 à 7,9.

Composant	Ce qui est évalué	Sous-indice	Indice de base
Processeur :	Calculs par seconde	7,2	5,9 Déterminé par le sous-indice le plus bas
Mémoire vive :	Opérations mémoire par seconde	5,9	
Graphiques :	Performances du Bureau pour Windows Aero	6,4	
Graphiques de jeu :	Performances graphiques pour jeux et applications professionnelles 3D	6,4	
Disque dur principal :	Taux de transfert des données sur le disque	5,9	

The direct comparison is hard here because various factors influence the results. (1) We use a more powerful computer (Pentium 4 at 3.8 Ghz with 2 GB RAM vs. Core 2 Quad at 2.66 Ghz with 4 GB de RAM). (2) We have a new OS (Windows XP 32 bit vs. Windows 7 64 bit). (3) Perhaps some tools are improved their programming of the decision tree induction. So, we must remain cautious for the comments of the results.

To facilitate the reading of the results, we take the table related to our previous experiment (September 2008 – Windows XP), we put in parallel the new results (October 2011, Windows 7 - 64 bit).

	Tree learning processing time			Memory occupation (MB)		
Logiciel	Seconds	Rank		Launching	Data loaded	Tree Learning
Knime 1.3.5	270	5		93	160	246
Orange 1.0b2	130	3		25	260	796
R 2.6.0	157	4		19	184	719
RapidMiner 4	298	6		136	228	1274
Sipina 3.0 - C4.5	122	2		8	67	540
Tanagra 1.4.27	33	1		7	53	122
Weka 3.5.6	338	7		52	253	700

Pentium IV 3,8 Ghz - Windows XP

	Tree learning processing time			Memory occupation (MB)		
Logiciel	Seconds	Rank	Improvement (%)	Launching	Data loaded	Tree Learning
Knime (2.4.2 - 64 bits) - 4 threads	39	2	86%	131	328	393
Orange (2.0b - Build 15/10/2011)	84	6	35%	76	405	1008
R 2.13.2 (64 bits)	79	5	50%	28	192	719
RapidMiner 5.1.011 (64 bits)	77	4	74%	369	771	926
Sipina 3.7 - C4.5	52	3	57%	3	63	76
Tanagra 1.4.41	22	1	33%	3	50	119
Weka 3.7.4 (64 bits)	195	7	42%	116	420	805

Q9400 2,66 Ghz - Windows 7 64 bits

Calculation time. The calculation is improved whatever the tool used. We observe mainly the influence of the better efficiency of the computer. The result of Tanagra is interesting because it operates under 32 bit fashion, and I know that the source code is not improvement since the previous studied version (14.27 at September 2008, I used 1.4.41 in this new experiment). Thus, the reduction of the calculation time that we can attribute to the computer is about 30%. The result for Orange is very similar.

I think also that the source code of Weka's J48 is not really improved. Thus, we can subdivide the gain (42%) in about 30% for the computer and about 10% for the 64 bit operating system. Indeed, this new version of Weka (3.7.4) operates natively under 64 bit fashion.

To confirm this last intuition, I use the R 2.13.2 software. Two versions - 32 bit and 64 bit - are installed on the computer. From this last one, the calculation time is 79 seconds (reported into the table above). With the 32 bit version, it becomes 90 seconds. We dispose of a direct comparison here. We can conclude that the R version which operates natively in a 64 bit fashion under a 64 bit operating system improves the calculation time in 12% in relation to the 32 bit version.

Why then such dramatic improvements for SIPINA, and especially for RAPIDMINER and KNIME? SIPINA works under 32 bit mode. Apart the gain attributed to the computer, the improvement rests on modifications of the source code I made when I introduced a multithreaded version of the decision tree induction (see http://data-mining-tutorials.blogspot.com/2010/11/multithreading-for-decision-tree.html). I think we have the same mechanism for RapidMiner (74%) and Knime (86%). Even if they run under 64 bit mode, the improvements are really spectacular. They rest probably on a great work on the source code. We can evaluated them to about 34% [74 - (30 + 10)] for RapidMiner and 46% [86 - (30 + 10)] for Knime[1].

As regards Knime, a parameter can disturb the comparison. We use a multithreaded version in our new experiment. I set "Number threads" to 4. But, in practice, I observed that only 2 cores are really used during

[1] These values are approximate. They give mainly an order of magnitude on the influence of the various factors on the calculation time.

the induction process. When I set the parameter to "Number thread = 1", the calculation time becomes 48 seconds. The gain is 82%.

About the ranking of the tools, we observe that Tanagra remains the better. I think that this is because the internal data structure is especially well adapted to the decision tree induction. This situation is not true for another learning algorithm (e.g. SVM - http://data-mining-tutorials.blogspot.com/2009/07/implementing-svm-on-large-dataset.html). We note however that the other tools – Knime especially - are significantly improved, they are got really closer.

Memory occupation. We put aside SIPINA for which a great work on memory leak is made for the recent version. The fear concerns mainly the 64 bit version. The memory occupation is automatically inflated for some data structures such as pointers (see http://cran.univ-lyon1.fr/bin/windows/base/rw-FAQ.html#Should-I-run-32_002dbit-or-64_002dbit-R_003f). We observe in this new experiment that the memory occupation has not been deteriorated. On the other hand, we can theoretically handle larger datasets into main memory. For instance, for the tools which rests on the Java technology, we can set a larger value for the "-Xmx" (maximum heap size) option. The capabilities of the tools are dramatically improved.

Conclusion. The main conclusion of this update is that, on a 64 bit system, for the same computer configuration, a native 64 bit application is about 10% faster in a decision tree induction process compared to the same program compiled in a 32 bit mode[2]. The advantage of the 64 bit application is all the more interesting that it can theoretically handle large datasets into main memory. And for the same dataset, the memory occupation is marginally increased. The second conclusion is that some tools, mainly Knime and RapidMiner, have considerably improved their program for the decision tree induction over the 3 last years. But, to date, and this is the last conclusion, Tanagra nevertheless remains very fast for the induction of decision tree.

[2] We note however that this conclusion is not definitive. Our dataset has some particular characteristics: all the descriptors are continuous, there are a few number of descriptors and a large number of instances. We should extend the experiment to other kind of dataset to obtain a larger scope results.