Subject

Handling a large dataset with ID3 decision tree algorithm.

In the data mining domain, the increasing size of the dataset is one of the major challenges in the recent years. Fifteen years ago, a dataset with 5,000 examples and 22 variables, such the famous "wave" (Breiman and al., 1984), was considered as a large file in the machine learning community. Today, the files know a very fast evolution according the areas: an increasing number of observations, such as in the marketing domain; or an increasing number of descriptors, in the bioinformatics domain for instance, in fact in the domains where we deal with unstructured dataset and we generate descriptors automatically.

The ability to handle large data sets is an important criterion to distinguish between research and commercial software. Commercial tools have often a very efficient data management systems, limiting the amount of data loaded into memory at each step of the treatment.

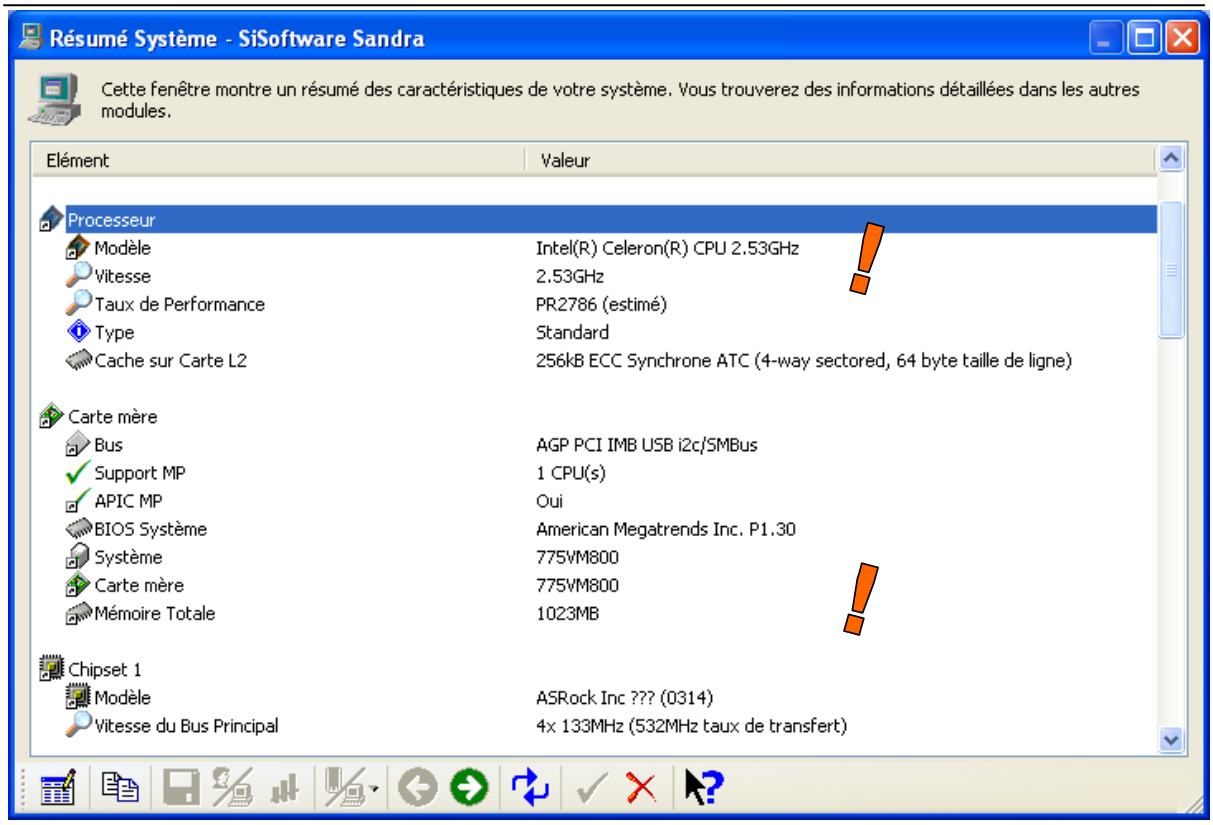
Research tools, at the opposite, load all data in memory. The limits are clearly the memory capacity of the machine in this context. It is certainly a drawback for the treatment of large files. We note however that, nowadays, you can have very powerful personal computers at less cost. This drawback is continuously postponed. With an appropriate encoding strategy, we can fit all the data in memory, even if we handle a large data file[1].

In this tutorial, we show how to import a file with 581,012 observations and 55 variables, and then how to build a decision tree with the ID3 method. In relation to other decision tree algorithm such as C4.5[2] or CART, the determination of the right size of the tree is based on a pre-pruning rule. We will see that the computation is fast because of this characteristic.

Our Personal Computer is a quite banal machine of which characteristics were measured with a shareware version of SiSoftware Sandra. It is a Celeron 2.53 GHz with 1 GB RAM running Windows XP SP2. This information is important because it allows the reader to compare the performance reported in this tutorial with the ones you get on your own computer.

[1] See for instance http://data-mining-tutorials.blogspot.com/2009/01/performance-comparison-under-linux.html

[2] See http://data-mining-tutorials.blogspot.com/2008/11/decision-tree-and-large-dataset.html for the performance comparison of various free tools on C4.5 algorithm.



Dataset

The data file COVTYPE contains 581102 examples and 54 discrete (or discretized) descriptors. The class attribute have 7 values. The size of the file, in the text format, is 62 MB[3].

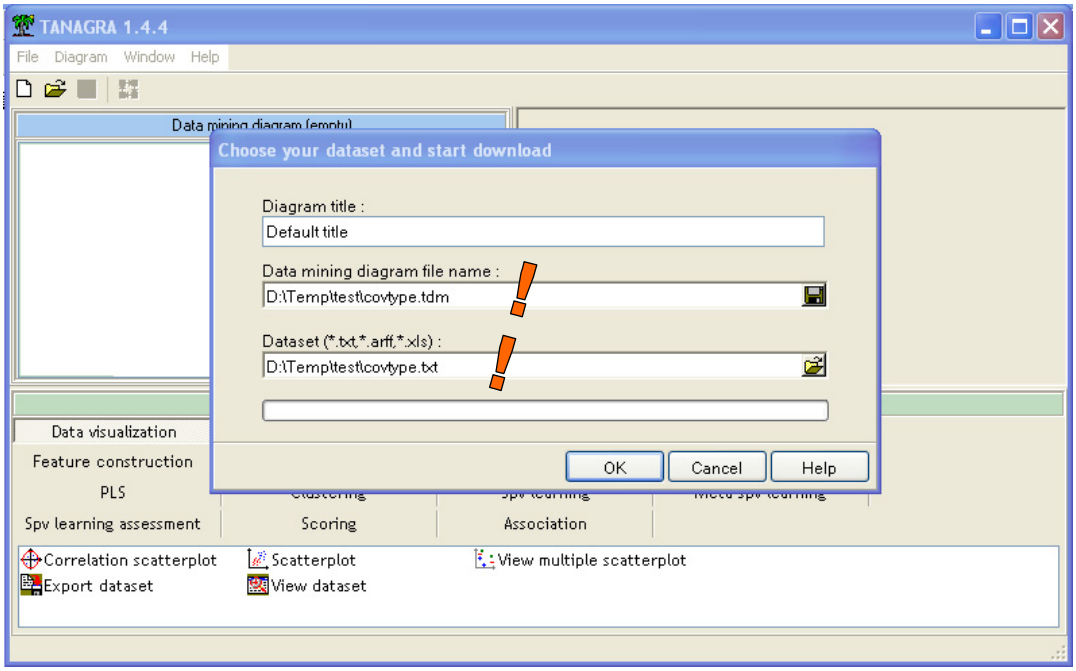
Var01	Var02	Var03	Var04	Var05	Var06	Var07	Var08	Var09	Var10	Var11	Var12	Var13
A	A	A	A	A	A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A	A	A	A
B	B	A	A	A	B	A	A	A	A	A	A	A
B	C	B	A	A	B	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A	A	A	A
A	B	A	B	A	A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A	A	A	A
A	C	A	A	A	A	A	A	B	A	A	A	A
B	C	A	B	A	C	A	A	A	B	A	A	A
B	B	B	A	A	B	A	A	C	A	A	A	A
A	C	A	A	A	A	A	A	B	A	A	A	A
A	C	A	A	A	A	A	A	A	C	A	A	A
A	A	A	A	A	A	A	A	A	C	A	A	A
A	D	A	A	A	A	A	A	B	A	A	A	A

[3] http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/covtype.zip

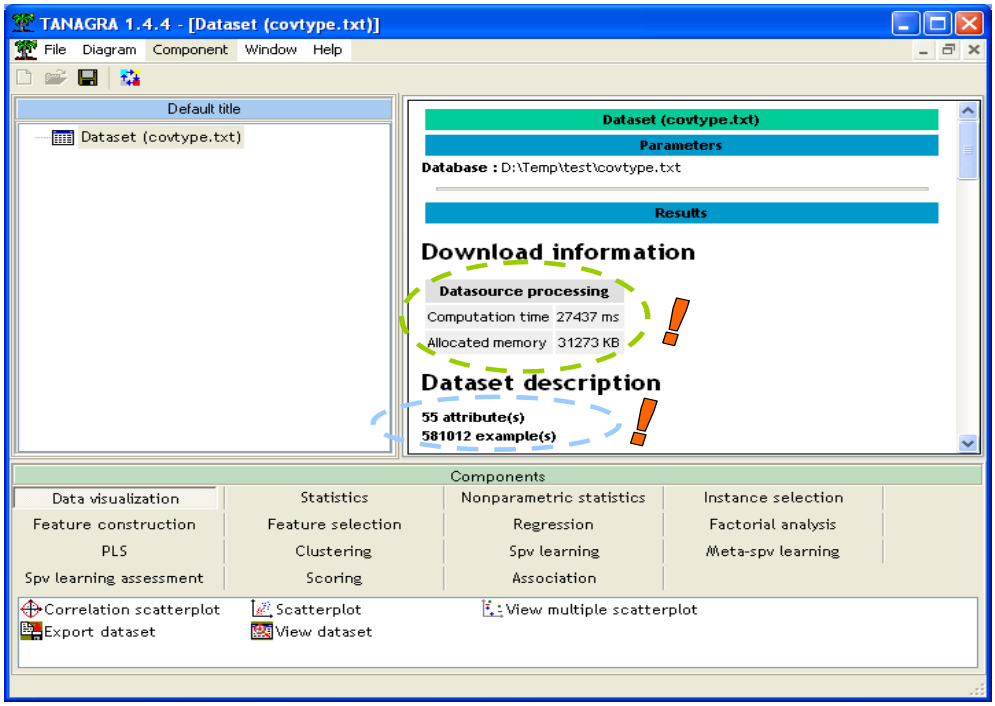
ID3 on a large dataset

Importing the data

First, we create a new diagram (FILE / NEW) and we select the data file (COVTYPE.TXT).



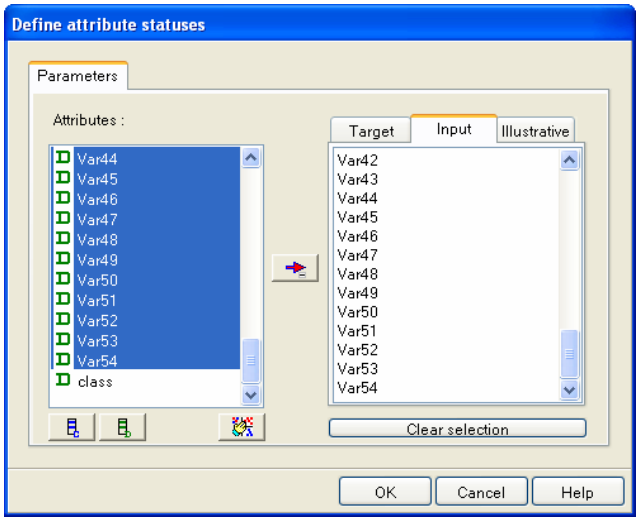
We validate by clicking on OK. A bar shows the progression. We note that we can interrupt the operation with the CANCEL button if we want.



The calculation time is 27 seconds. The memory occupation of the dataset is about 31 MB.

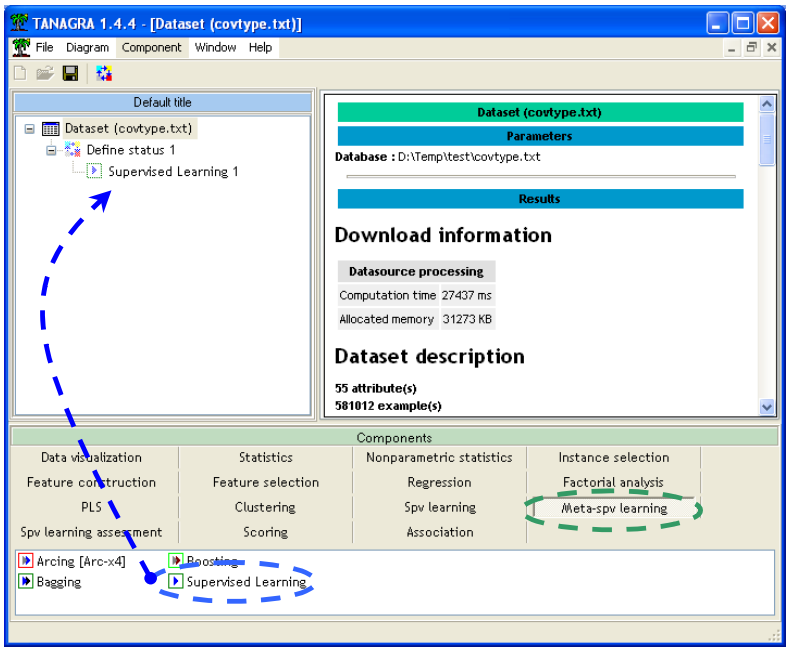
Specifying the type of the variables

We insert the DEFINE STATUS component into the diagram in order to define the type of the variables. We set CLASS as TARGET, all the others as INPUT.



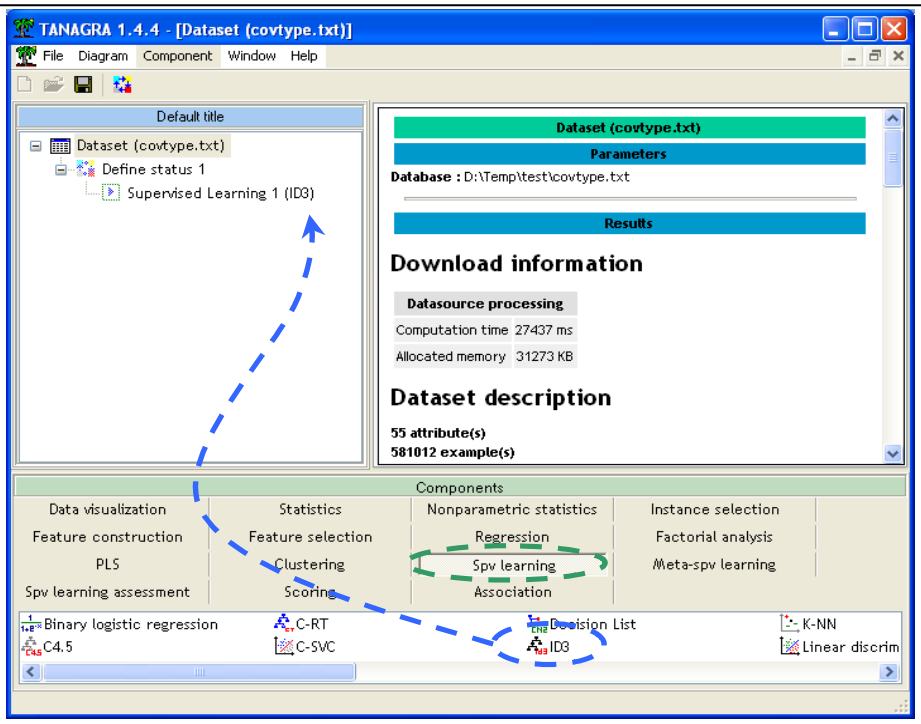
ID3

We must define the supervised learning algorithm. We make the operation by two steps: (1) we insert the SPV LEARNING component (META SPV LEARNING tab), this component instantiates the learning method;

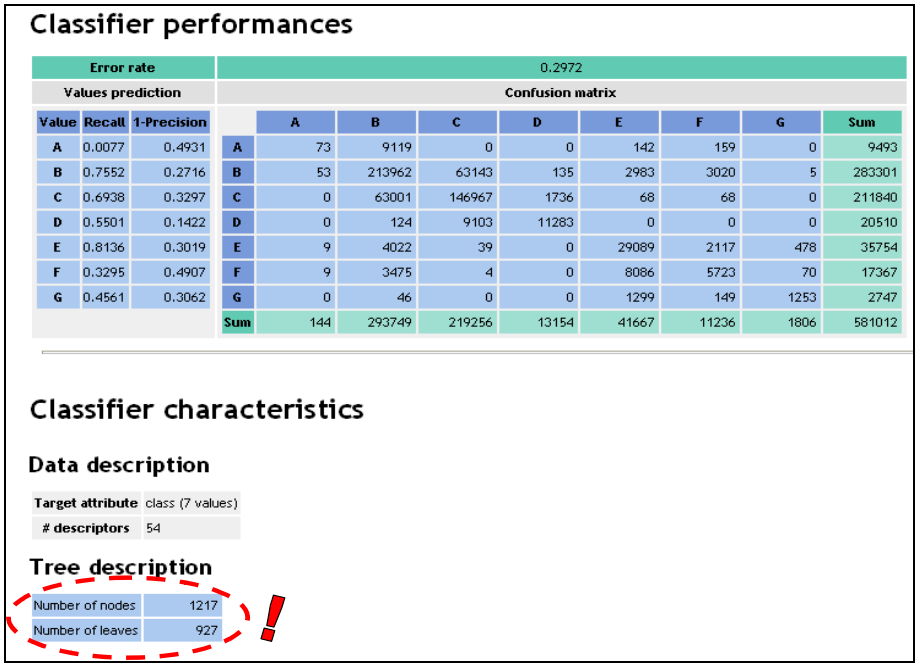


(2) Then, we insert the learning algorithm i.e. ID3 (SPV LEARNING tab)[4].

[4] For the simple instantiation, we can directly insert the learning method (e.g. ID3); the meta supervised learning component is automatically added. This simplified way is not possible for more complex meta-algorithm such as boosting or bagging. In this situation, we must explicitly observe these two steps.



We obtain the results by clicking on the contextual VIEW menu.



Classifier performances

Error rate	0.2972

Values prediction

Value	Recall	1-Precision
A	0.0077	0.4931
B	0.7552	0.2716
C	0.6938	0.3297
D	0.5501	0.1422
E	0.8136	0.3019
F	0.3295	0.4907
G	0.4561	0.3062

Confusion matrix

	A	B	C	D	E	F	G	Sum
A	73	9119	0	0	142	159	0	9493
B	53	213962	63143	135	2983	3020	5	283301
C	0	63001	146967	1736	68	68	0	211840
D	0	124	9103	11283	0	0	0	20510
E	9	4022	39	0	29089	2117	478	35754
F	9	3475	4	0	8086	5723	70	17367
G	0	46	0	0	1299	149	1253	2747
Sum	144	293749	219256	13154	41667	11236	1806	581012

Classifier characteristics

Data description

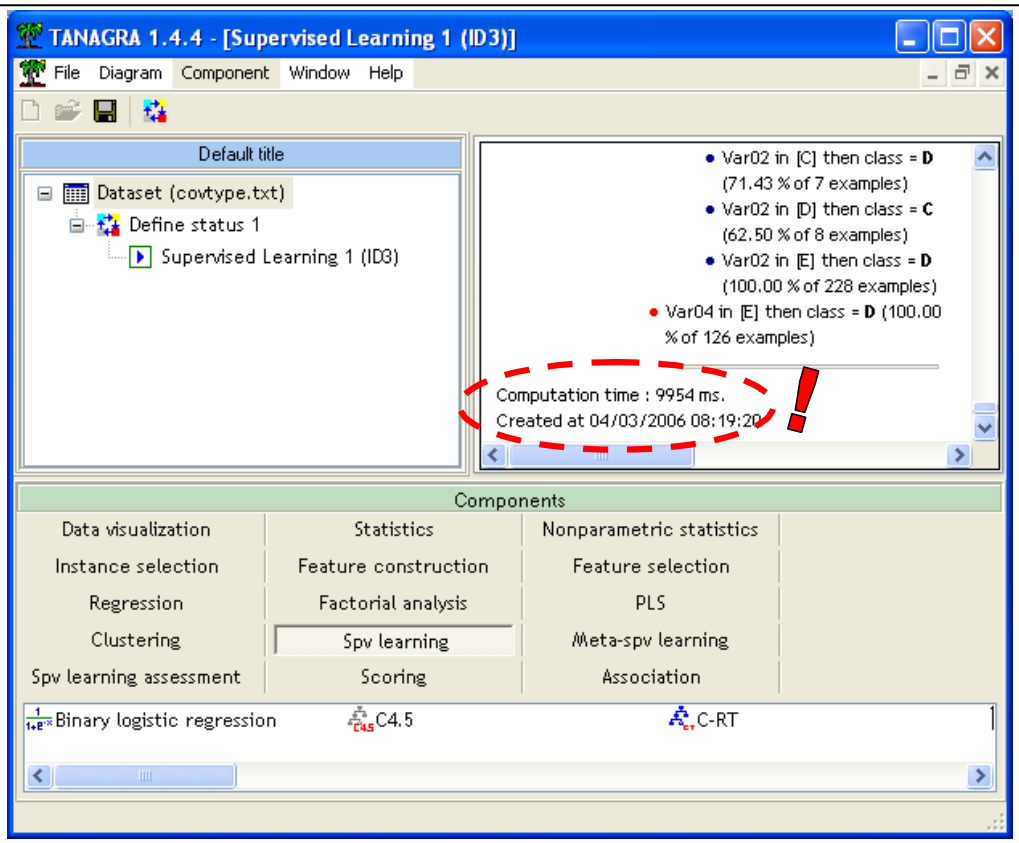
Target attribute	class (7 values)
# descriptors	54

Tree description

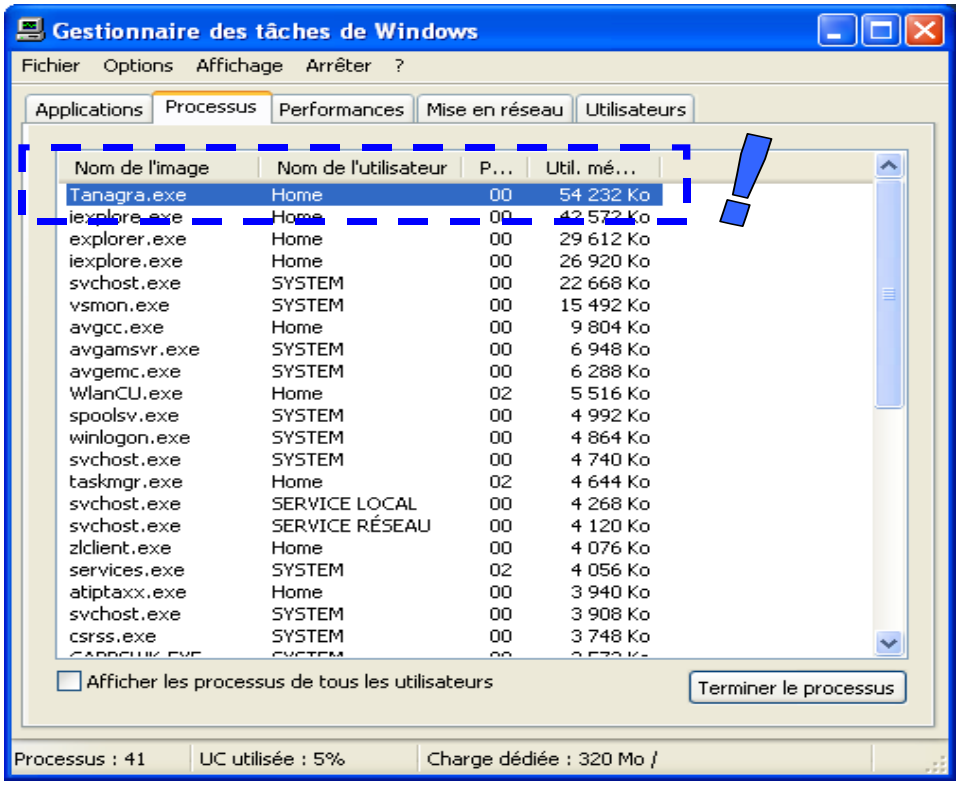
Number of nodes	1217
Number of leaves	927

There are 927 leaves into the tree. In this case, the interpretation of the tree is not really possible. Most interesting in this tutorial is the calculation time. We get the low part of the results. We observe the following[5]:

[5] TANAGRA automatically generates a log file DEBUGFILE.TXT where all operations during the processing and the associated execution times are kept. It is possible to consult this file.



10 seconds are enough to get the tree. In addition, the real memory occupation into the OS is moderate in regard of the size of the dataset (about 54 MB).



Conclusion

Loading the entire dataset into the memory for a data mining process seems to be an unsophisticated approach, especially when we deal with a very large dataset. Yet, the majority of free tools use this strategy. Can we believe that these tools are not usable in a real project?

The answer is more subtle. Because of the increasing performance of the PC (fifteen years ago, a PC with 32 MB memory was considered as a performing tool) and a judicious data structure, the capacity of the tools which handle the whole dataset in memory remains important. In this tutorial, the computation is feasible on a data file with 581102 examples and 54 predictive attributes; we can create a decision tree in a very fast time.

NOTE

As a confirmation of this conclusion, our PC has changed since this version of tutorial was written (March 2006). Three years later (April 2009), I use a PC Quad Core Q9400 at 2.66 GHz under Windows Vista. The internal structure is slightly improved, especially by using a better memory management structure (since Tanagra version 1.4.30). In this context, the data importation time becomes 6.7 seconds (versus 27 seconds) and the decision tree construction time becomes 3.6 seconds (versus 10 seconds). The memory occupation remains the same.