

# 1 Introduction

Supported data file format in SIPINA.

The data access is the first step of the data mining process. It is a crucial step. It is one of the main criteria used when we want to assess the quality of a tool. If we do not able to load a dataset, we cannot perform any kind of analysis. The software is not useable. If the data access is not easy and requires complicated operations, we will devote less time to the other steps of the data exploration.

Two points of view are essential when we want to evaluate the data file format: the flexibility and the performance. The flexibility refers to the ability to manipulate the dataset with other tools than the dedicated software, and thus, the ability to exchange the dataset between various tools. The delimited text file format, says also CSV file format, is most probably the easiest to handle. The great majority of tools can handle this format, especially the spreadsheet programs such as Excel or Open Office Calc, but also the DBMS (database management system) such as SQL Server or Oracle.

The performance refers mainly to the quickness of the input/output operations and, to a lesser extent, to the disk occupation. It is above all important when we treat a large dataset (millions of observations and/or hundreds of variables) or when we must repeatedly load and save the dataset.

When we deal with a moderate size dataset (up to 65,535 rows under Office XP; 1,048,576 under Office 2007), it is clear that the spreadsheet takes a particular position. Polls show that many people use Excel (or more generally spreadsheet) with a specialized data mining tools<sup>1</sup>. It is not surprising. A spreadsheet program is very suitable to handle a data in the attribute-value format. In this context, the putting up a link between a spreadsheet and the specialized data mining tool is a very useful solution. We show below how to implement it with Sipina.

The first goal of this tutorial is to describe the various file formats that are supported in Sipina. Some of the solutions are more deeply described in other tutorials elsewhere; we indicate the appropriate reference in these cases. The second goal is to describe the behavior of these formats when we handle a large dataset with **4,817,099 instances** and **42 variables**.

Last, we learn a decision tree on this dataset in order to evaluate the behavior of Sipina when we process a large data file.

## 2 Dataset

First, we use the WEATHER.TXT dataset which a tab-separated values text file format (Quinlan, 1993 -- <http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/weather.txt>).

---

<sup>1</sup> See <http://www.kdnuggets.com/polls/2008/tools-languages-used-data-cleaning.htm> and <http://www.kdnuggets.com/polls/2008/data-mining-software-tools-used.htm>

Outlook	Temp	Humid	windy	Class
sunny	75	70	yes	Play
sunny	80	90	yes	DontPlay
sunny	85	85	no	DontPlay
sunny	72	95	no	DontPlay
sunny	69	70	no	Play
ovcast	72	90	yes	Play
ovcast	83	78	no	Play
ovcast	64	65	yes	Play
ovcast	81	75	no	Play
rain	71	80	yes	DontPlay
rain	65	70	yes	DontPlay
rain	75	80	no	Play
rain	68	80	no	Play
rain	70	96	no	Play

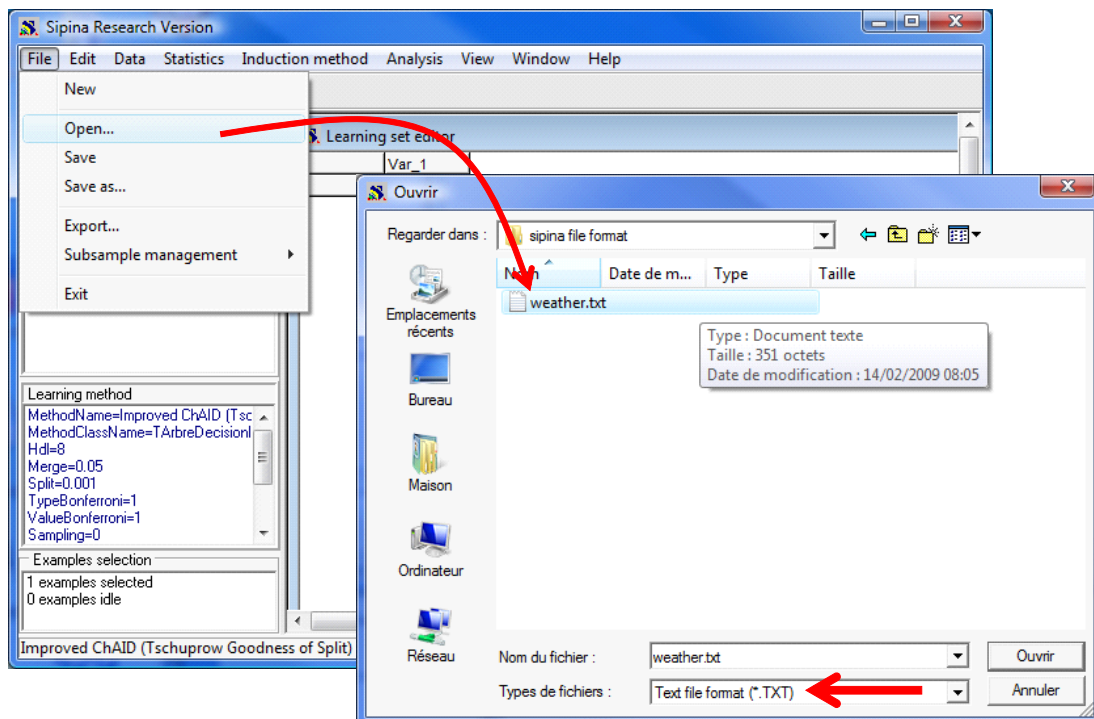
This kind of file format is very used. In most of the cases, we consider that the first row corresponds to the name of the variables.

### 3 Tab-separated values file format

The starting point is the WEATHER.TXT above. In the following, we use the same process: (1) we load the dataset in a specified format; (2) we check if all the values are imported; (3) we export the dataset in a specified format; (4) we check if we can import it thereafter.

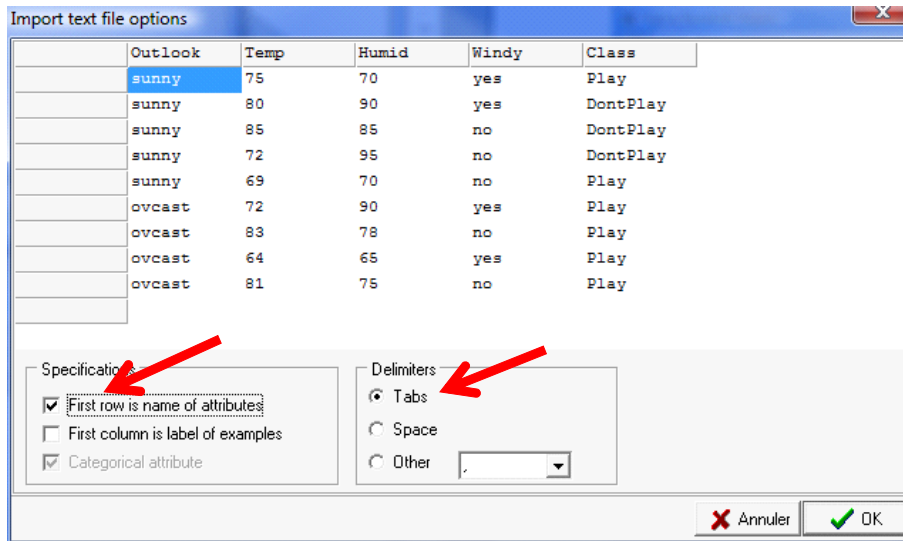
#### 3.1 CSV (\*.TXT) text file format

**Importing the TXT file format.** After we launch SIPINA, we click on the FILE / OPEN menu. A dialog box appears. We select the TEXT FILE FORMAT (\*.TXT). We select the WEATHER.TXT data file.

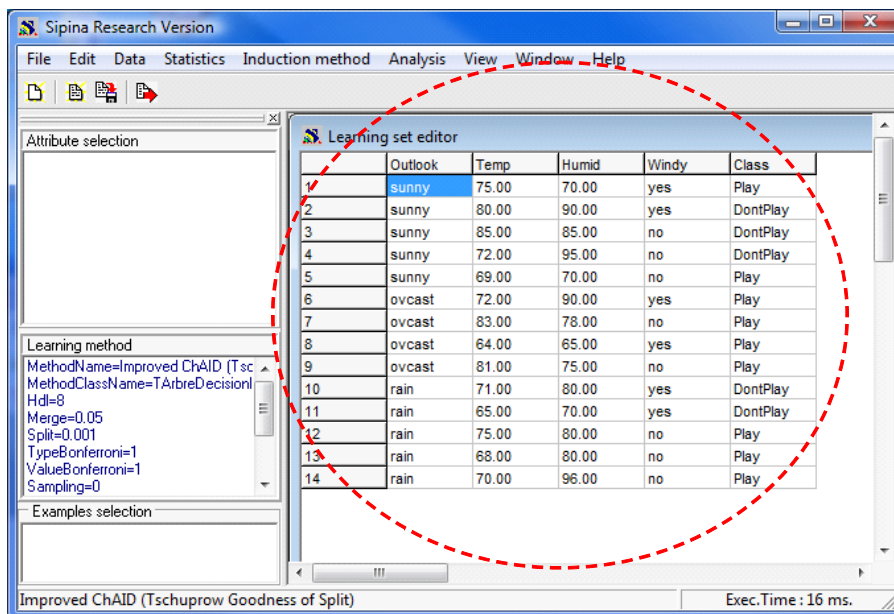


Another dialog box appears, we set the column separator and we indicate that the first row

corresponds to the name of the variables.



We validate by clicking the OK button. The dataset is imported. The values are displayed into the visualization grid.

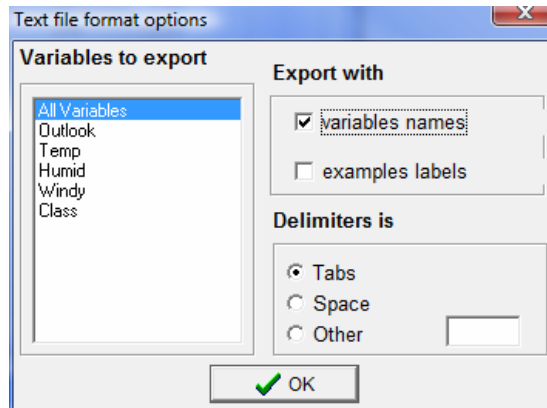


**Note: Specifying the types of the variables.** Sipina refers to the first row of the data (the second row if the first row corresponds to the name of the variables) to determine the type of the variable. If we have a numerical value, Sipina considers that we have a continuous attribute; if we have alphabetical value, it sets the column as a discrete attribute. We can modify these default settings by clicking on the DATA / DEFINE SPECIFICATIONS menu. Sipina can modify the type of variables; it can scan the set of values for each column, etc.

**Caution: decimal point.** For Sipina, the decimal point is always « . » whatever our OS configuration.

**Exporting the dataset into the TXT format.** In order to export a dataset in the TXT file format, we click

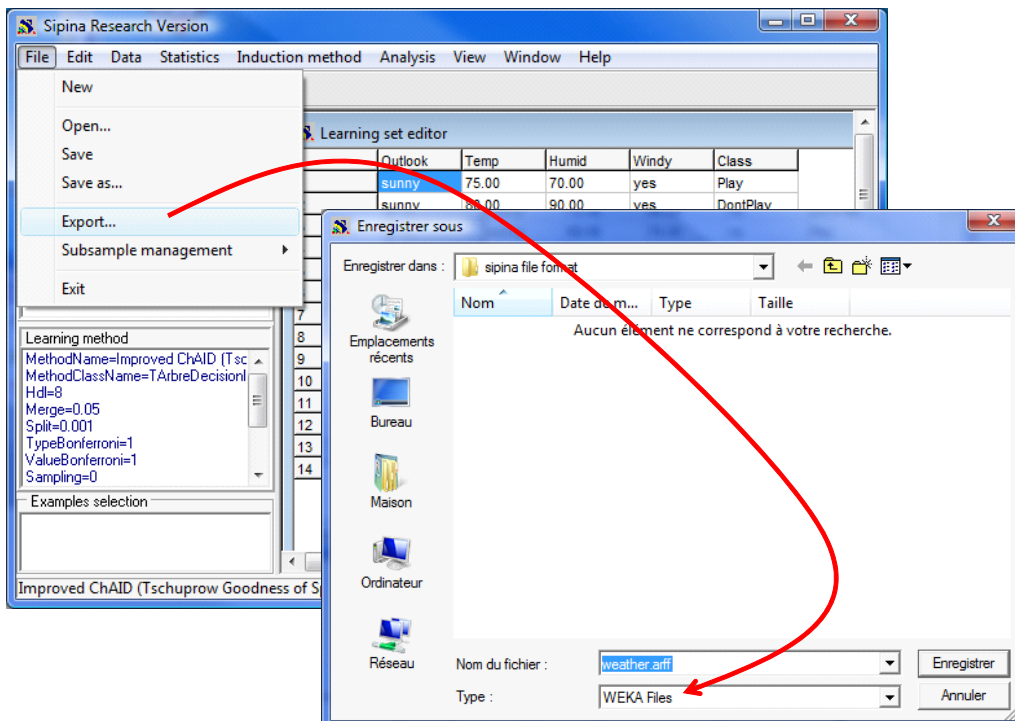
on the FILE / EXPORT menu. We select the TXT format. A dialog box asks us the list of variables to export, and other specifications.



### 3.2 Weka file format (\*.ARFF)

Weka is a very popular data mining tool (<http://www.cs.waikato.ac.nz/ml/weka/>). It has its own format (ARFF) which is a text file format with additional information. The first part is the data dictionary; the second part corresponds to the values.

**Exporting into the ARFF format.** To export the previous dataset into the ARFF file format, we click on the FILE / EXPORT menu. We select the Weka files format. We set WEATHER.ARFF as file name.



We can check the data file on the disk. We see here the contents of the file into the standard Windows notepad.

```

weather.arff - Bloc-notes
Fichier Edition Format Affichage ?
@relation weather.arff
@attribute outlook {sunny,ovcast,rain}
@attribute Temp REAL
@attribute Humid REAL
@attribute windy {yes,no}
@attribute Class {Play,DontPlay}
@data
sunny,75,70,yes,Play
sunny,80,90,yes,DontPlay
sunny,85,85,no,DontPlay
sunny,72,95,no,DontPlay
sunny,69,70,no,Play
ovcast,72,90,yes,Play
ovcast,83,78,no,Play
ovcast,64,65,yes,Play
ovcast,81,75,no,Play
rain,71,80,yes,DontPlay
rain,65,70,yes,DontPlay
rain,75,80,no,Play
rain,68,80,no,Play
rain,70,96,no,Play

```

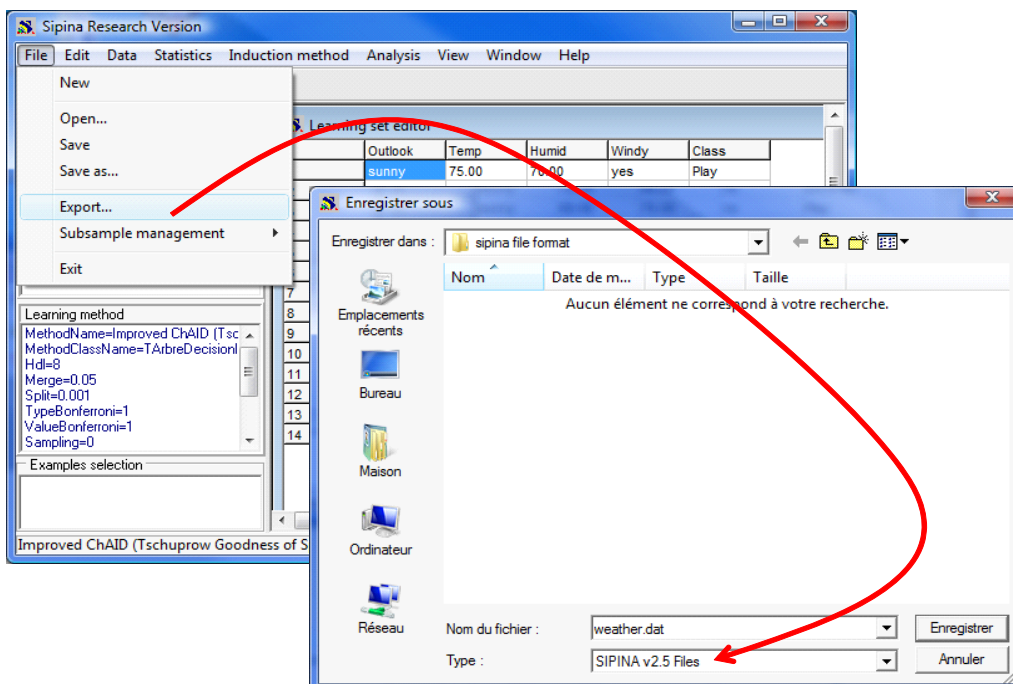
In the data dictionary part, a continuous variable is pointed out by the REAL keyword; the set of values are simply enumerated for a discrete attribute. The “@data” part is similar to the CSV file format.

**Importing the ARFF file format.** Sipina can load a data file in the ARFF format. To do this, we click on the FILE / OPEN and we select the Weka File Format (\*.ARFF). No additional settings are required.

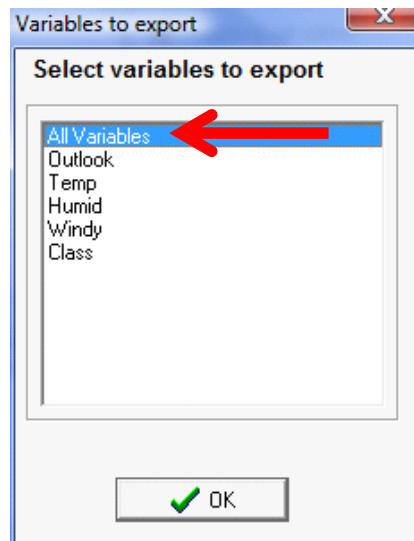
### 3.3 Text file for the old Sipina - Version 2.5

The old version of Sipina (2.5 and older) uses a text format which is very similar to the ARFF format. But, it uses two separate files: the first corresponds to the data dictionary; the second contains the values. The main drawback is that the user had to handle simultaneously two files. It was not really convenient.

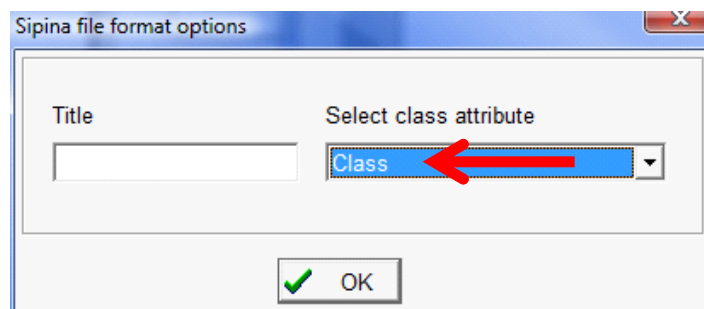
**Exporting into the Sipina version 2.5 file format.** We click on the FILE / EXPORT format. We select the SIPINA V2.5 FILES option. We set WEATHER.DAT as file name.



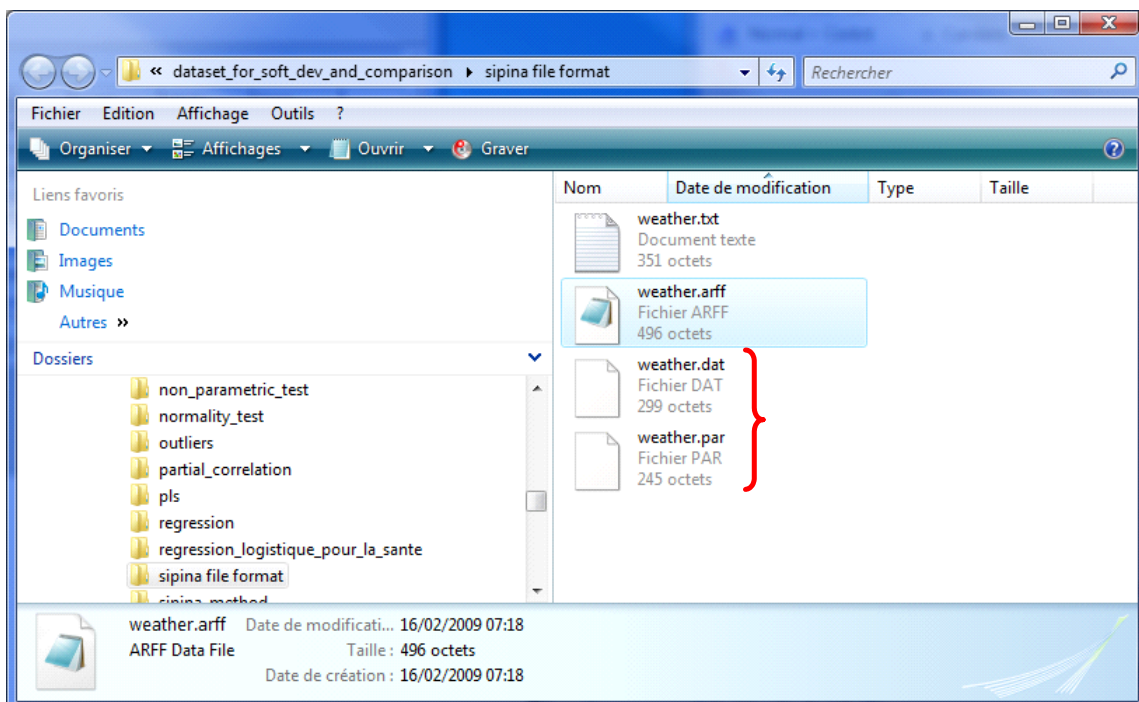
A dialog box appears. We set the list of variables to export, ALL VARIABLES here.



In the following dialog box, we choose the target attribute for a supervised learning scheme.

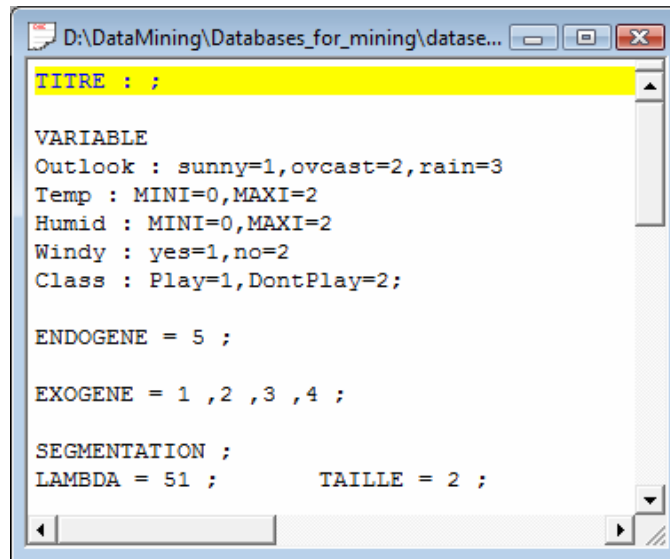


We can see that two new files are created on the disk.





We can open them in the NOTEPAD. For WEATHER.PAR, we have the description of the attributes and their statuses in the learning scheme.



```

D:\DataMining\Databases_for_mining\datase...
TITRE : ;

VARIABLE
Outlook : sunny=1,ovcast=2,rain=3
Temp : MINI=0,MAXI=2
Humid : MINI=0,MAXI=2
Windy : yes=1,no=2
Class : Play=1,DontPlay=2;

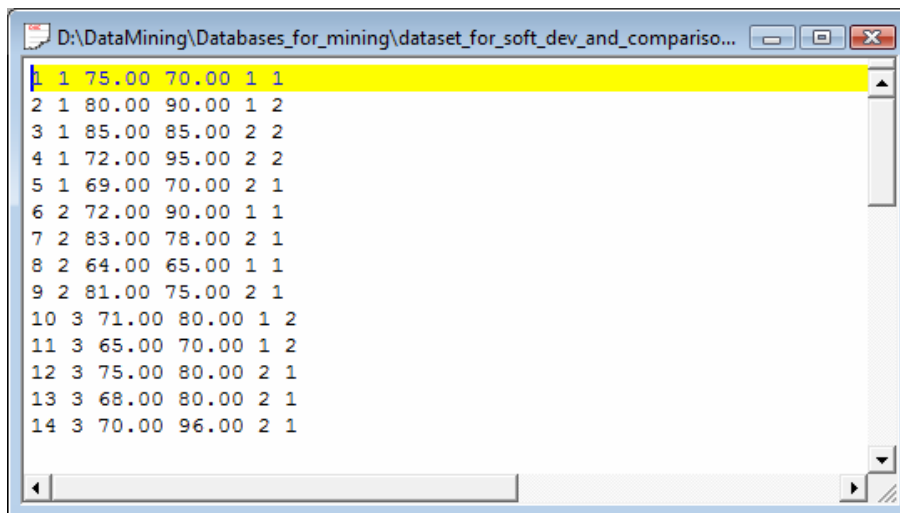
ENDOGENE = 5 ;

EXOGENE = 1 ,2 ,3 ,4 ;

SEGMENTATION ;
LAMBDA = 51 ;      TAILLE = 2 ;

```

Into WEATHER.DAT, we have the description of the values. Sipina uses the encoding values for the discrete attributes.



```

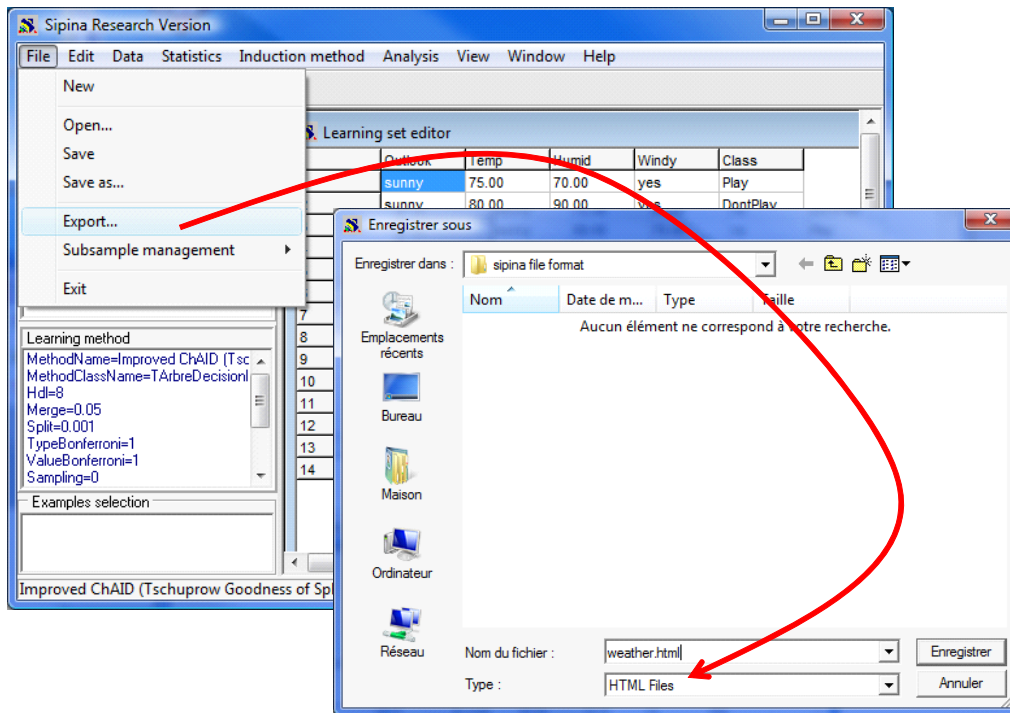
D:\DataMining\Databases_for_mining\dataset_for_soft_dev_and_compariso...
1 1 75.00 70.00 1 1
2 1 80.00 90.00 1 2
3 1 85.00 85.00 2 2
4 1 72.00 95.00 2 2
5 1 69.00 70.00 2 1
6 2 72.00 90.00 1 1
7 2 83.00 78.00 2 1
8 2 64.00 65.00 1 1
9 2 81.00 75.00 2 1
10 3 71.00 80.00 1 2
11 3 65.00 70.00 1 2
12 3 75.00 80.00 2 1
13 3 68.00 80.00 2 1
14 3 70.00 96.00 2 1

```

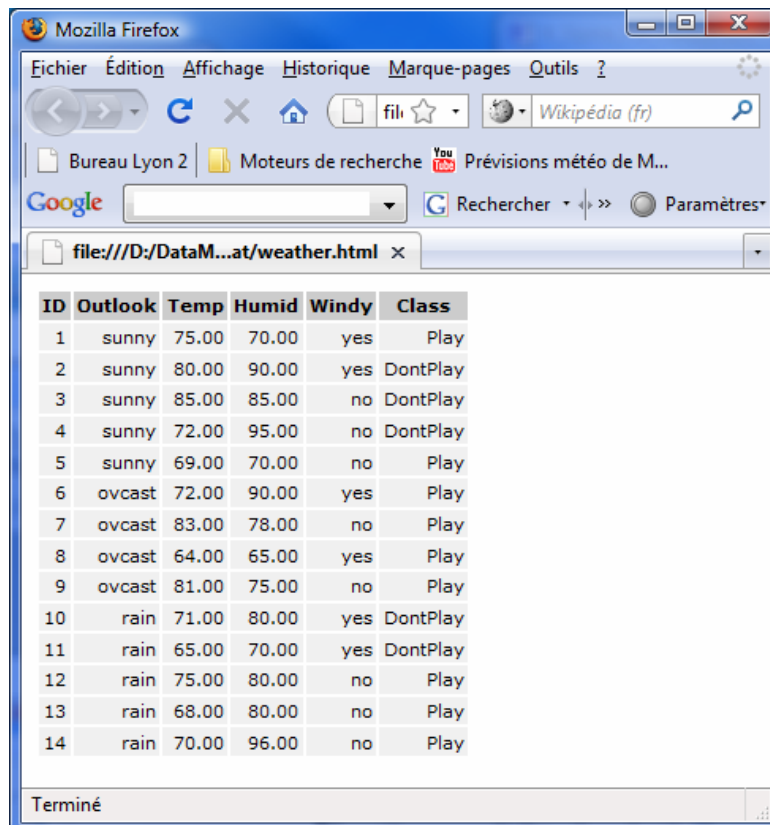
### 3.4 HTML file format

The HTML file format is also a text file format. The dataset can be visualized in a browser. It can be also imported into a spreadsheet (Excel or Open Office Calc). We note however that HTML is a very verbose format. The size of the file can increase rapidly for large datasets.

**Exporting into the HTML file format.** We click on the FILE / EXPORT menu. We select the HTML file format. We set WEATHER.HTML as file name.



The dataset can be displayed into a browser, Firefox here.



The dataset can be also imported into a spreadsheet, Open Office Calc for instance.



	A	B	C	D	E	F	G
1	ID	Outlook	Temp	Humid	Windy	Class	
2	1	sunny	75.00	70.00	yes	Play	
3	2	sunny	80.00	90.00	yes	DontPlay	
4	3	sunny	85.00	85.00	no	DontPlay	
5	4	sunny	72.00	95.00	no	DontPlay	
6	5	sunny	69.00	70.00	no	Play	
7	6	ovcast	72.00	90.00	yes	Play	
8	7	ovcast	83.00	78.00	no	Play	
9	8	ovcast	64.00	65.00	yes	Play	
10	9	ovcast	81.00	75.00	no	Play	
11	10	rain	71.00	80.00	yes	DontPlay	
12	11	rain	65.00	70.00	yes	DontPlay	
13	12	rain	75.00	80.00	no	Play	
14	13	rain	68.00	80.00	no	Play	
15	14	rain	70.00	96.00	no	Play	
16							

## 4 The binary file formats of Sipina

Sipina has a proprietary binary file format. In the simplest version (FDM), the vectors of values are directly dumped into a file. The input/output operations are very fast for this format. It is especially intended to data files containing a high number of instances and a moderate number of variables. It is less adapted for the dataset containing a high number of variables.

About the size of the file, we can easily compute it:

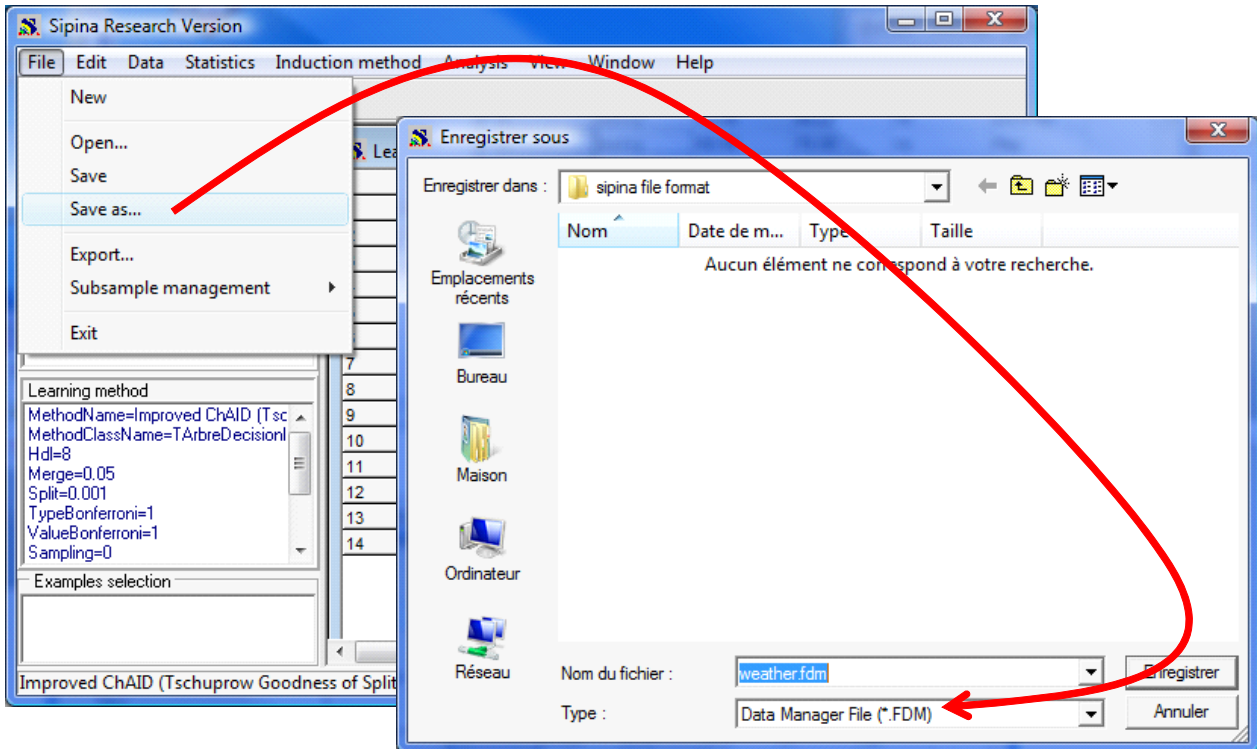
- The disk occupation of each value is 4 bytes.
- To each row, we have a label described on 25 bytes.
- Each name of variable is encoded on 25 bytes.
- We can add eventually the description of the values for a discrete attribute.

Thus, for a dataset with 100,000 instances and 10 continuous attributes, the file size is:

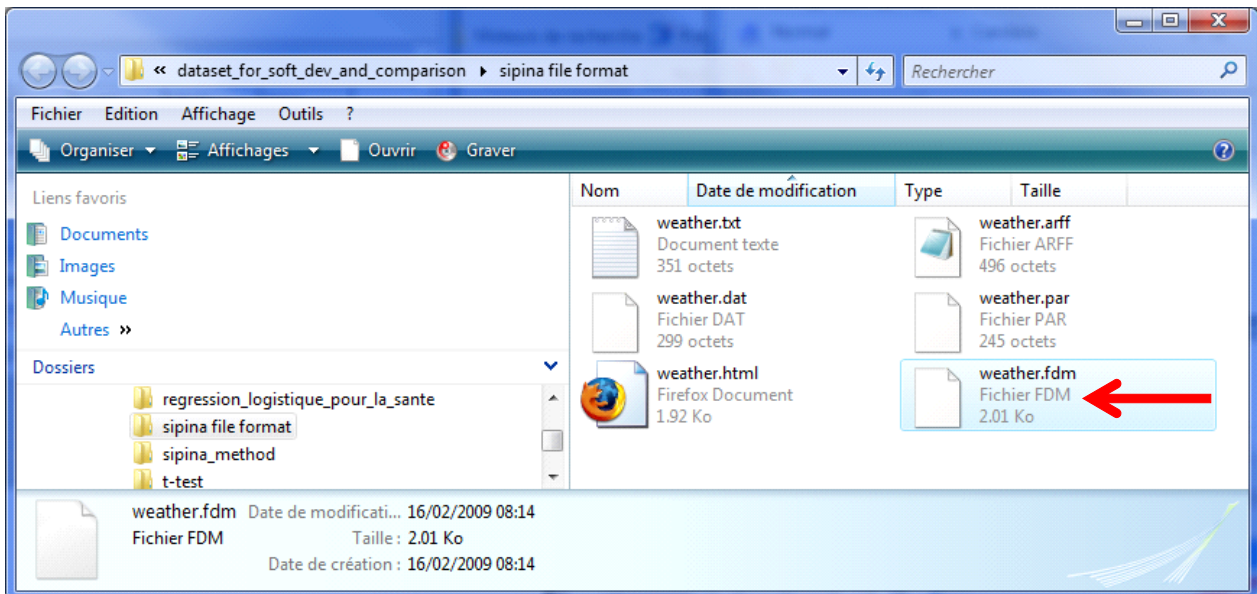
$$100.000 \times 10 \times 4 + 100.000 \times 25 + 10 \times 25 = 6.500.250 \text{ bytes} \# 6.2 \text{ MB}$$

### 4.1 Handling the FDM file format

**Exporting in the FDM file format.** We click on the FILE / SAVE AS menu. The default file format is FDM, we set WEATHER.FDM as file name.



As we see into the directory, the file size (WEATHER.FDM) is higher than the others (TXT, ARFF).



**Reading the FDM file format.** Only Sipina can read the FDM file. We click on the FILE / OPEN menu and we select the appropriate file (WEATHER.FDM).

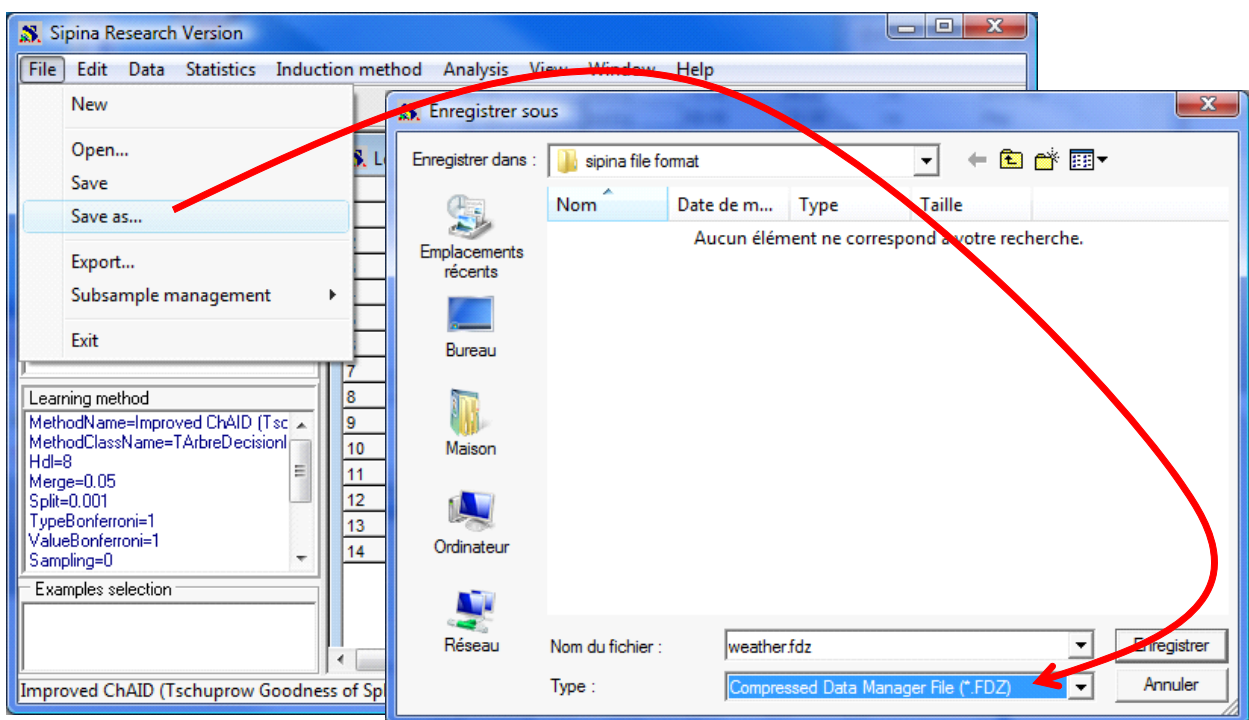
The main argument in favor of FDM is the quickness. The main drawback is the file size. In this context, we had developed a compressed format for the FDM binary file format. There are two compressed format in Sipina.

## 4.2 The FDZ file format

FDZ is also a binary file format. It is the FDM file which is compressed with the LZW algorithm (Lempel Ziv Welch - <http://en.wikipedia.org/wiki/Lempel-Ziv-Welch>). Rather the quality of the compression, we give priority to the quickness of the processing. The resulting file size is thus larger than those obtained with dedicated tools such as 7-Zip.

We note that this strategy creates a temporary file on the disk during the processing. It is of course removed when the final FDZ file is created.

**Creating a file in a FDZ format.** We click on the FILE / SAVE AS menu. We select the FDZ format and we set WEATHER.FDZ as file name.



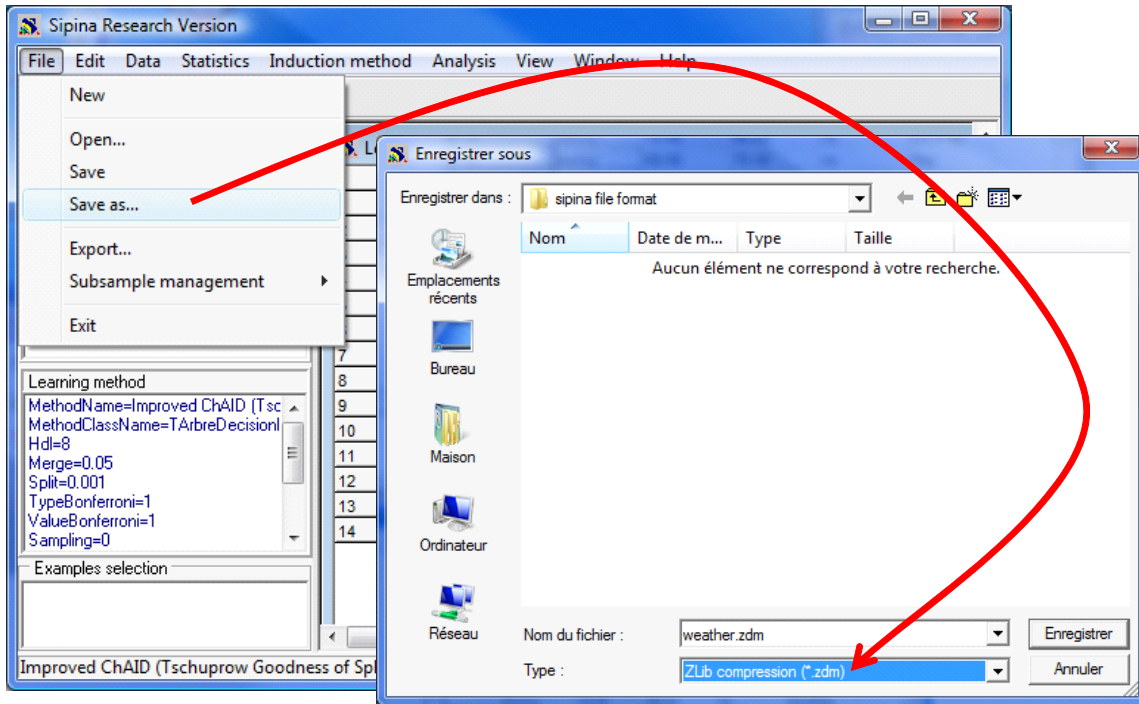
**Reading the FDZ file format.** We click on the FILE / OPEN menu when we want to read a FDZ file.

## 4.3 The ZDM

The ZDM format is also a compressed version of the FDM file. But, unlike the FDZ format, the compression process is performed "on the fly". None temporary file is created.

We must give priority to this format when we work on a support with poor performances (such as USB memory bar for instance).

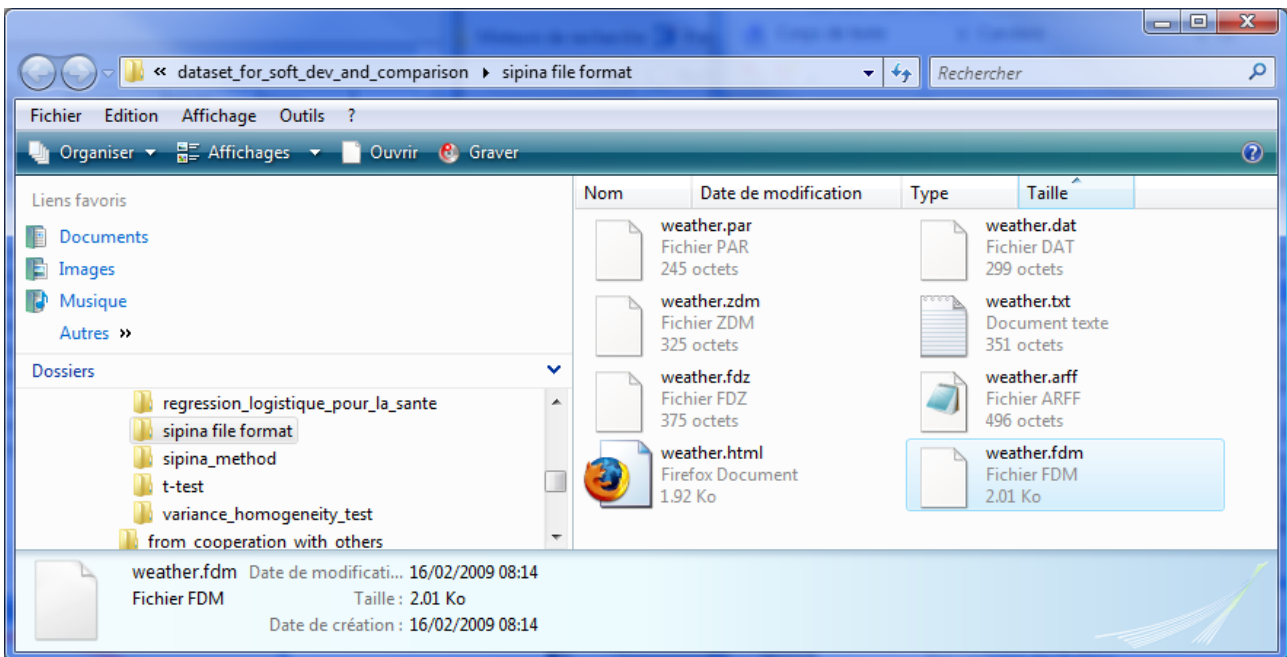
**Creating a ZDM file.** We click on the FILE / SAVE AS menu and we select the ZDM format. We set WEATHER.ZDM as file name.



**Reading a ZDM file.** Such as the other binary formats, we click on the FILE / OPEN menu and we select the appropriate format when we want to read the data file.

## 5 Summary – The size of the various file format

We summarize here the file size according to the selected file format. For the old Sipina 2.5 format, we must add up the size of PAR and DAT files.



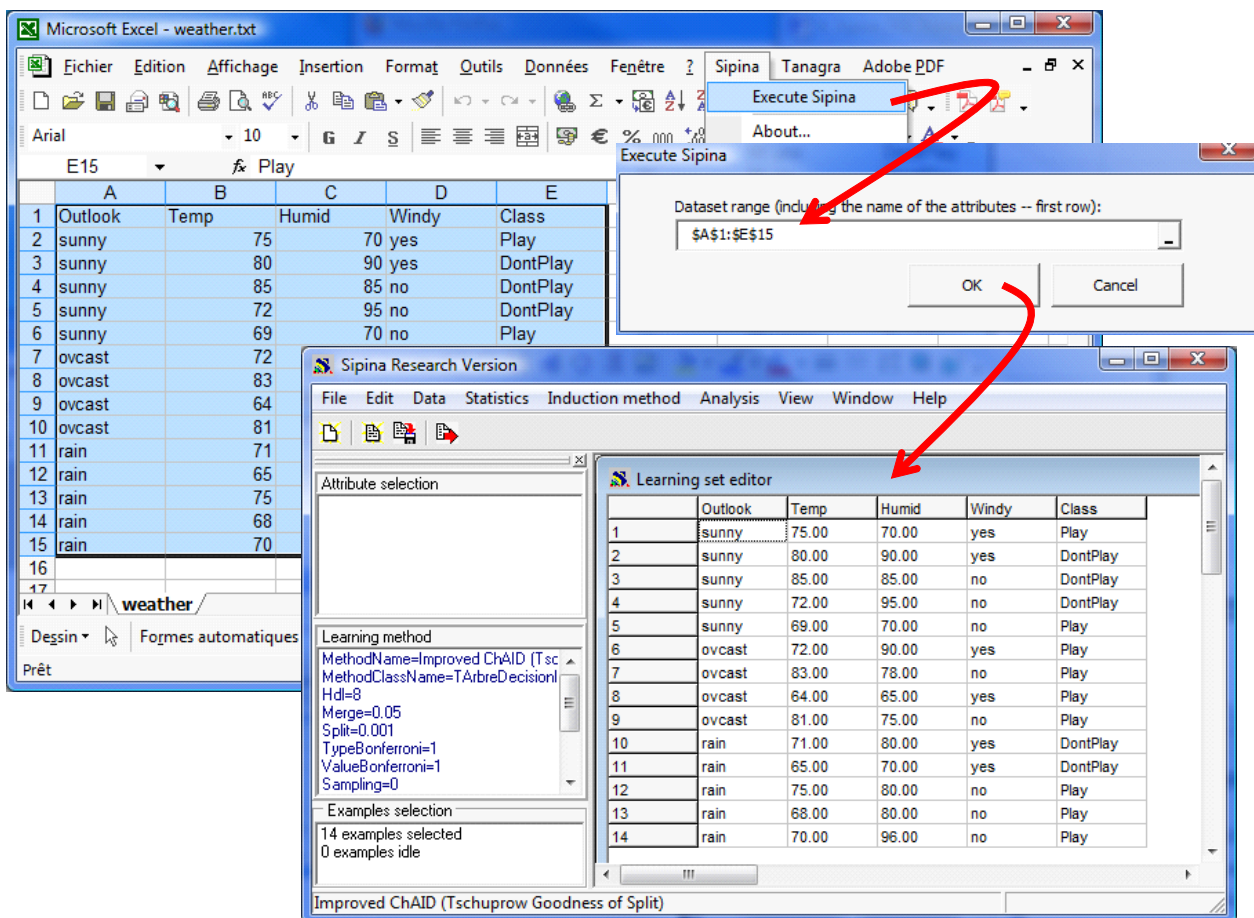
We note that the ZDM format is smallest; at the opposite, the FDM is the largest. This is fairly typical situations that we encounter in practice.

## 6 A link between Excel and Sipina

Many data miner use a spreadsheet during the data preparation phase. Because it is easy to use and it is rather efficient for the moderate sized datasets.

We can send a dataset from Excel to Sipina using the SIPINA.XLA add-in. The installation of the add-in is described here: [http://eric.univ-lyon2.fr/~ricco/doc/sipina\\_xla\\_installation.htm](http://eric.univ-lyon2.fr/~ricco/doc/sipina_xla_installation.htm); its use is described here: [http://eric.univ-lyon2.fr/~ricco/doc/sipina\\_xla\\_processing.htm](http://eric.univ-lyon2.fr/~ricco/doc/sipina_xla_processing.htm).

For our dataset, after the installation of the add-in, we perform the following steps in order to send the dataset towards Sipina. Sipina is automatically launched.



## 7 Performance comparison

We have various ways to handle a dataset. What is the file format that we must choose?

**For a moderate sized dataset, the connexion between Excel and Sipina must be used.** It is a good compromise between flexibility and quickness. In addition, we have benefit of all the data processing capabilities of a spreadsheet.

**For a large sized dataset, we must choose between the flexibility of the text file format and the quickness of the binary file.** It depends on the needs of data exchange between users or tools



(flexibility) and the need to read and save repeatedly the dataset (quickness).

In order to compare the behavior of the various file formats, we use another data file with **4,817,099** instances and **42** variables (<http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/kdd-cup-discretized-descriptors.txt.zip>). It is a cleansed version of the « KDD Cup 1999 Data<sup>2</sup> ». We use two criteria in order to compare the performances of the file formats: the disk occupation; and the processing time during the reading and the writing. Our PC is a Quad Core Q9400 at 2.66 GHz, running under Windows Vista.

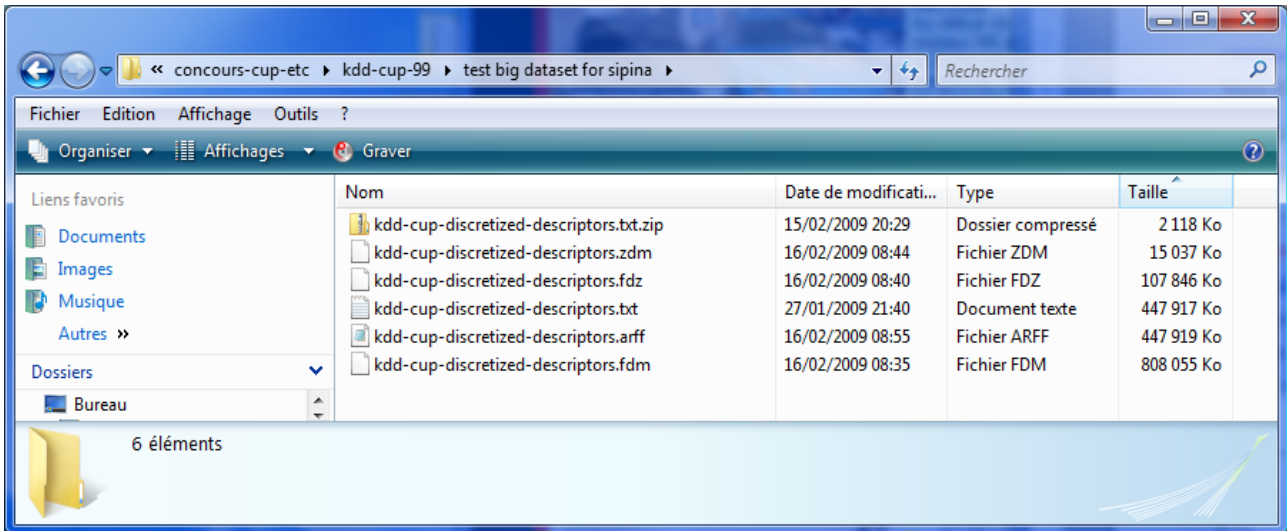
	Text file (TXT)	ARFF (Weka)	Binary FDM	Binary FDZ (compressed)	Binary ZDM (compressed)
File size on the disk (KB)	447.917	447.919	808.055	107.846	<b>15.037</b>
File reading (seconds)	107	105	8	57	<b>7</b>
File writing (seconds)	188	120	<b>8</b>	15	15

We note several results:

- It is obvious that the memory occupation of the dataset is the same when it is loaded, whatever the file format used.
- About the file size, TXT and ARFF are similar. It is not surprising.
- The FDM file is voluminous. The compressed format enables to dramatically reduce the FDM file size, especially for the ZDM format.
- About the processing time, as expected the reading and the writing of the text file (TXT and ARFF) are not fast. Notably because alphabetical value must be encoded during the reading process. The writing seems slower than the reading because it was not really optimized into Sipina.
- On the other hand, the binary file format is very fast, especially for the FDM format. None intermediate operation comes to slow down the input/output of the vectors of values.
- As expected, FDZ and ZDM are a bit slower than FDM. With the exception of ZDM during the reading. I think it is the consequence of the reduced size of the file.

We show here the disk occupation of the various formats of the dataset. We note that when we use an external program in order to compress the text files, we obtain a smaller file size (TXT.ZIP). We use the ULTRA option of 7-ZIP. Perhaps it is another way to improve the disk occupation of the data files. But in this case, the processing is very slow (about 120 seconds on my computer).

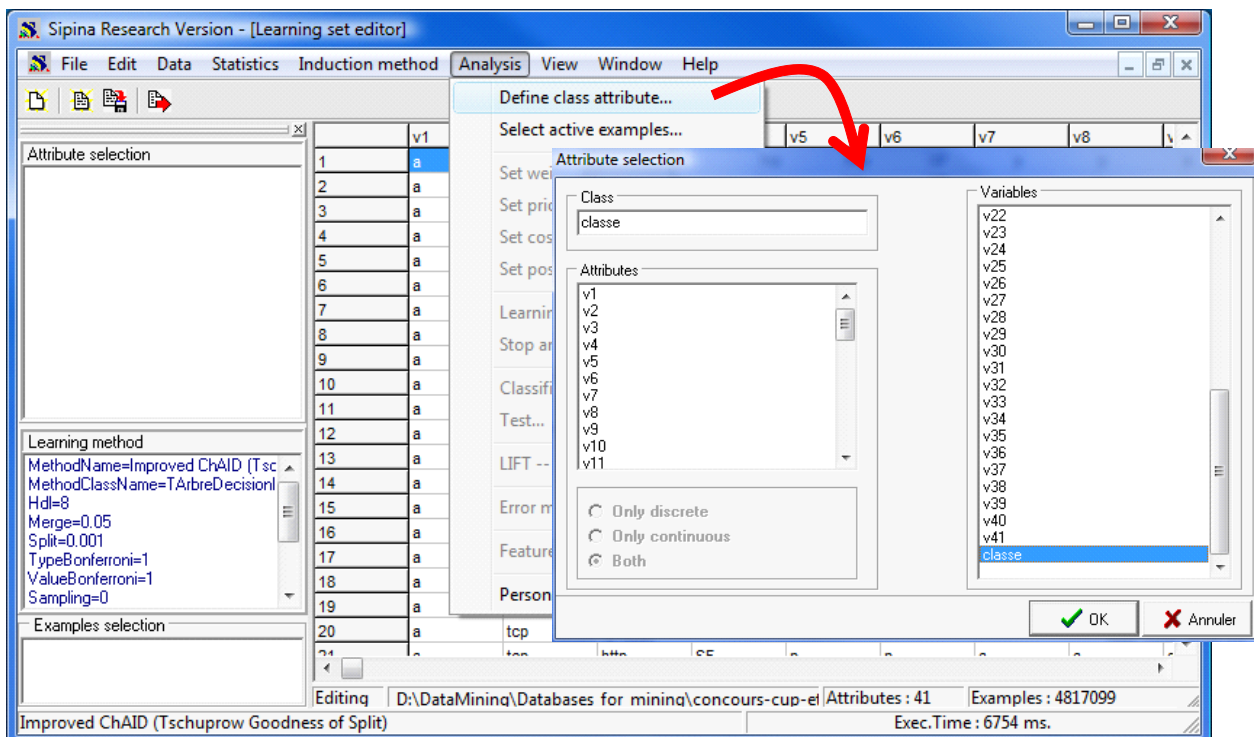




## 8 Decision tree learning on KDD-CUP 1999 dataset

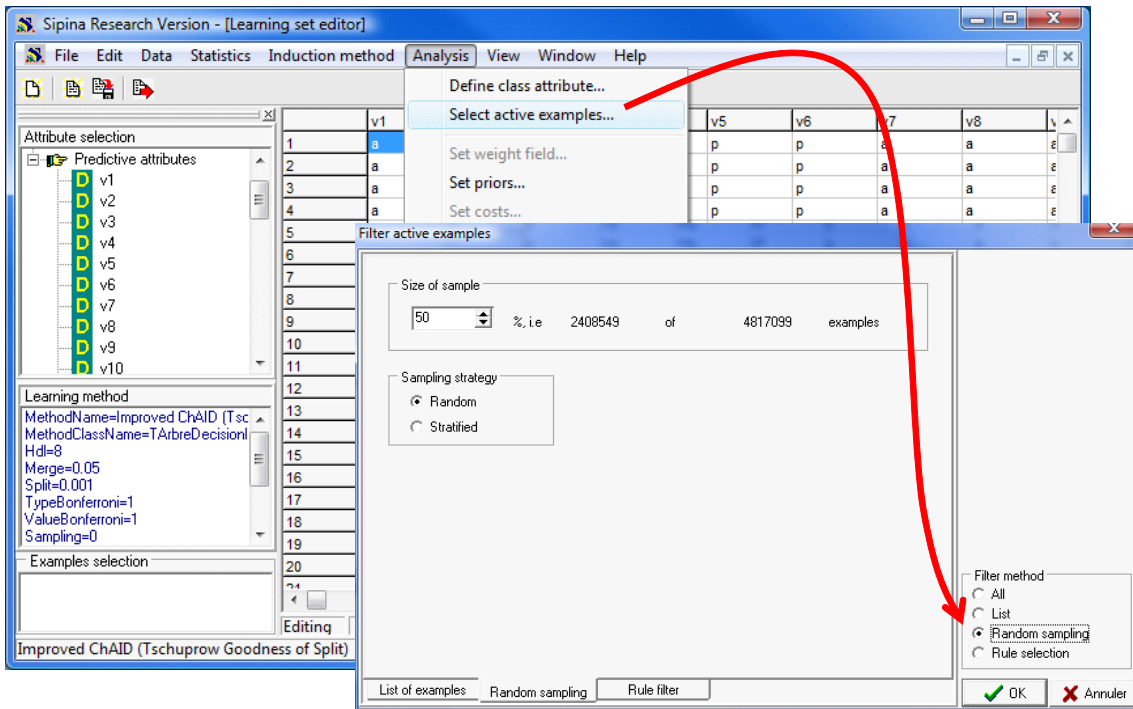
Even if it is not the main subject of this tutorial, we want to observe the behavior of Sipina when we learn a decision tree from our large dataset. With the dataset loaded, and before any learning process, the memory occupation is 882 MB.

We specify the TARGET attribute and the INPUT ones by clicking on the ANALYSIS / DEFINE CLASS ATTRIBUTE menu.

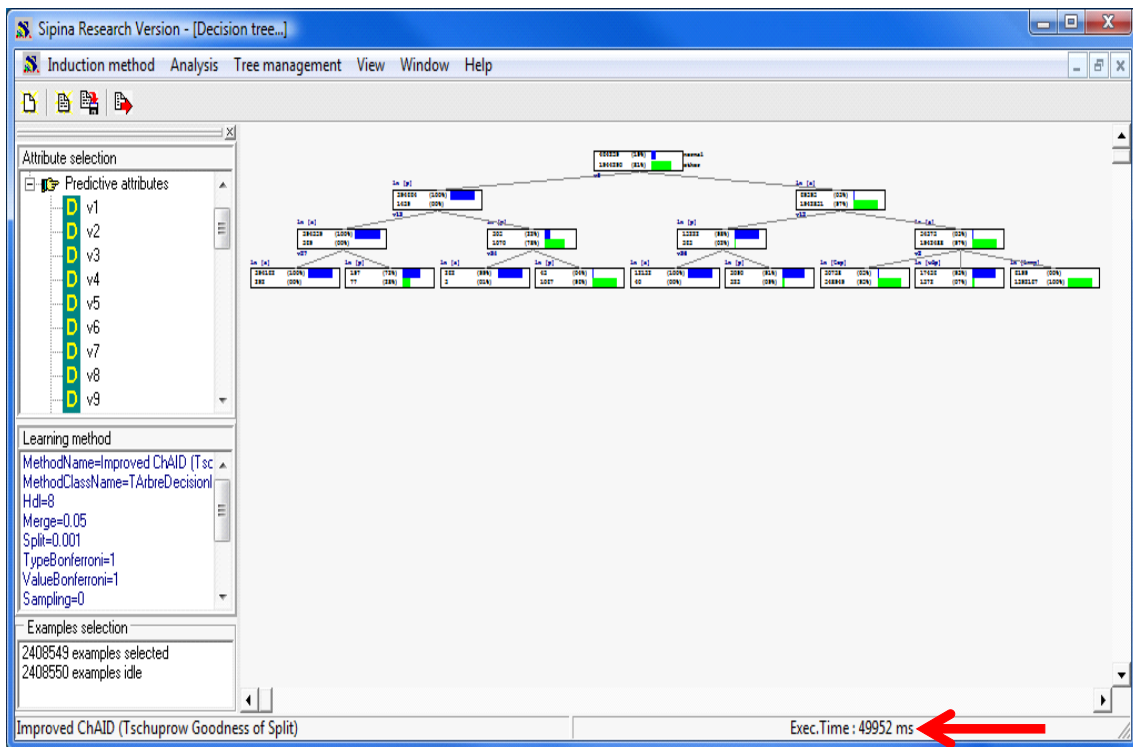


<sup>2</sup> <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

We want to use the half of the dataset for the learning phase, the other half for the evaluation of the classifier. We click on the ANALYSIS / SELECT ACTIVE EXAMPLES menu; then we choose the RANDOM SAMPLING option. The learning sample size is 2,408,549 cases (50% of the dataset).

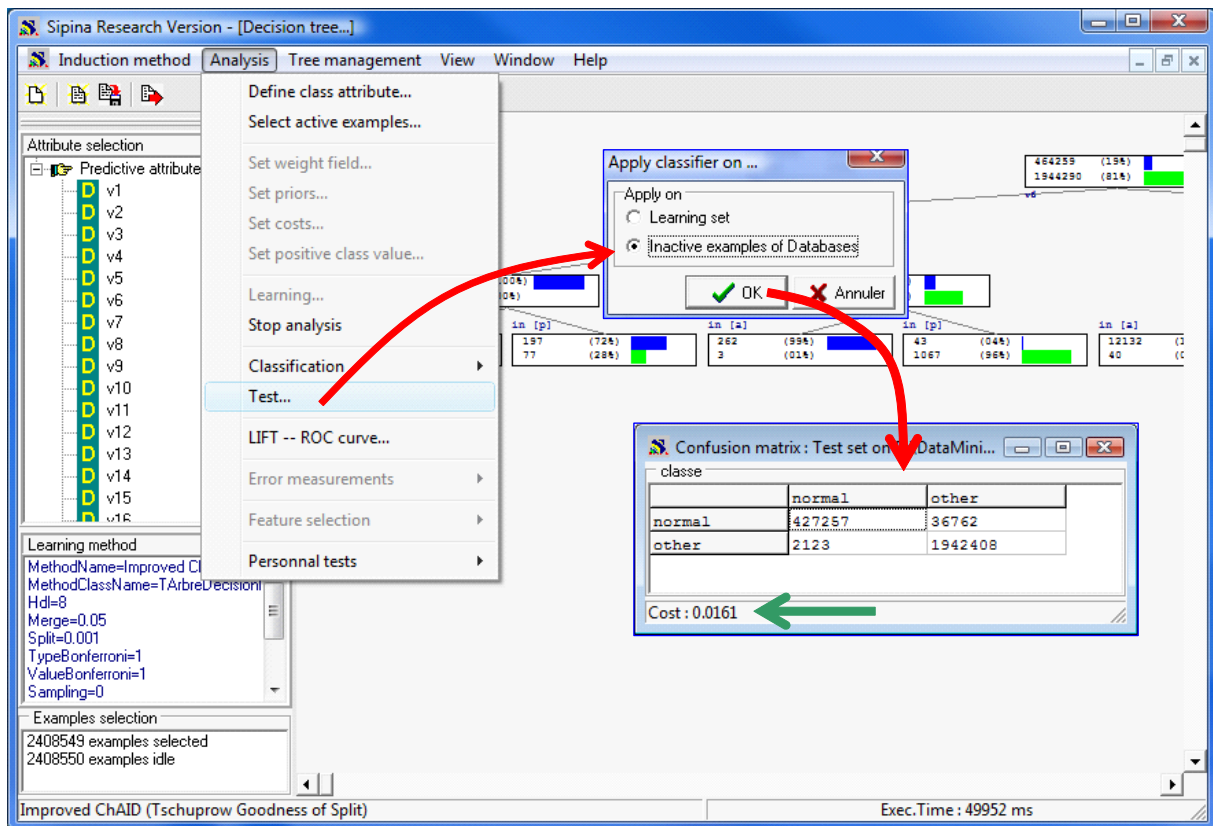


We can launch the learning process. We click on the ANALYSIS / LEARNING menu. The tree is obtained after 50 seconds.



The memory occupation of Sipina is 920 MB at this step.

We want to assess the tree on the test set. We click on the ANALYSIS / TEST menu. Into the dialog settings, we select the INACTIVE EXAMPLES OF DATABASE option. Sipina displays the confusion matrix and the test error rate which is 1.61%.



## 9 Conclusion

Our idea in this tutorial was to give a quick overview of file formats that could be processed with Sipina. They can be classified into two categories: text format, with the main asset flexibility; and binary formats, crucial for processing times and disk occupation when compressed.

Of course, there is no format which could be uniformly better than the others. It depends, on the one hand, to the characteristics of the dataset; on the other hand, to the objective and the constraints of our analysis. The flexibility and the performances (disk occupancy and processing quickness) seem to be good criteria to compare the advantages and the drawbacks of the various formats.