

1 Topic

Data Mining with Scilab.

I know the name "Scilab" for a long time (<http://www.scilab.org/en>). For me, it is a tool for numerical analysis. It seemed not interesting in the context of the statistical data processing and data mining. Recently a mathematician colleague spoke to me about this tool. He was surprised about the low visibility of Scilab within the data mining community, knowing that it proposes functionalities which are quite similar to those of R software. I confess that I did not know Scilab from this perspective. I decided to study Scilab by setting a basic goal: is it possible to perform simply a predictive analysis process with Scilab? Namely: loading a data file (learning sample), building a predictive model, obtaining a description of its characteristics, loading a test sample, applying the model on this second set of data, building the confusion matrix and calculating the test error rate.

We will see in this tutorial that the whole task has been completed successfully easily. Scilab is perfectly prepared to fulfill statistical treatments. But two small drawbacks appear during the catch in hand of Scilab: the library of statistical functions exists but it is not as comprehensive as that of R; their documentation is not very extensive at this time. However, I am very satisfied of this first experience. I discovered an excellent free tool, flexible and efficient, very easy to take in hand, which turns out a credible alternative to R in the field of data mining.

2 Scilab

2.1 What is Scilab?

Scilab is an open source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization, and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations. Scilab is one of several open source alternatives to MATLAB ([Wikipedia](#), 2014/05/01).

I noticed several interesting features by reading the available documentation:

1. Scilab has a data management mechanism. It proposes among others the tools for manipulating vectors and matrices.
2. It can handle tabular data in plain text form (CSV, comma-separated values)¹. This format is widely used in the data mining context.
3. Scilab is a tool but it is also a high level numerically oriented programming language. It has all of its features: choice (if then else), loops, subroutines, etc.
4. The objects provided by the statistical procedures have properties that we can use in subsequent calculations.
5. Graphics functions enable to create and customize plots and charts.
6. It is possible to enhance the library of functions using external modules (the "toolbox") that we can create and distribute. A repository takes an inventory of these libraries. An automatic

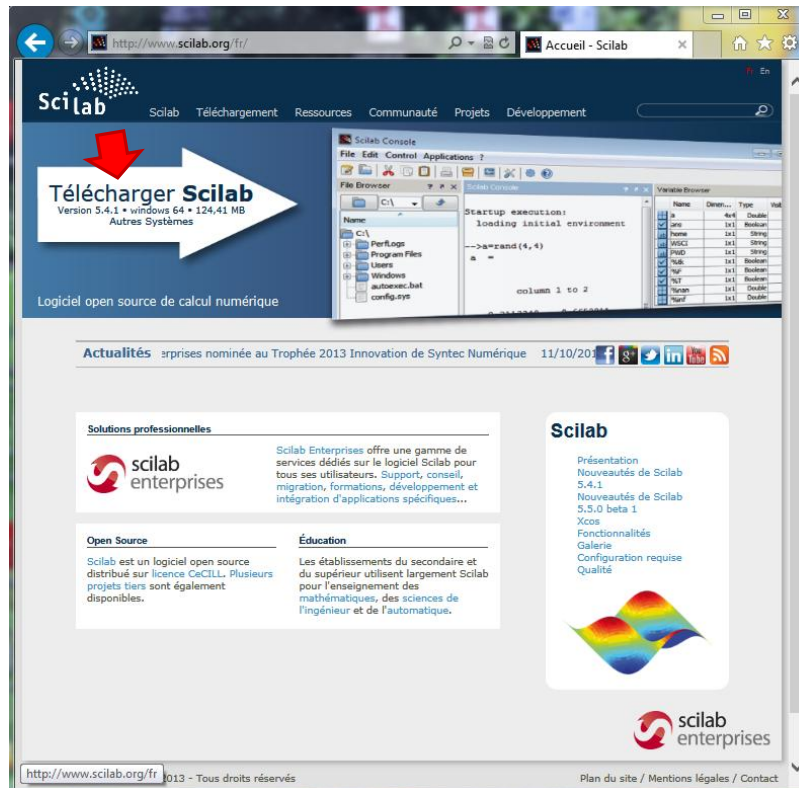
¹ http://help.scilab.org/docs/5.4.1/fr_FR/csvRead.html

importing mechanism makes easier their installation and their use (ATOMS: AuTomatic mODules Management for Scilab - <http://atoms.scilab.org/>).

I feel like I describe the specifications of R! At this stage, I think that apart from the learning of new programming language syntax, the identification and understanding of the commands necessary for my analysis should not be too difficult.

2.2 Downloading and installation

Scilab can be downloaded from the website of the publisher.



For Windows, we have a setup file which performs the whole installation. I downloaded the **5.4.1** version for Windows 64-bit. Other versions for other platforms are available.

Scilab 5.4.1

Sortie le 02/04/2013
[Configuration requise](#) | [Notes de version](#) | [Nouveautés de Scilab 5.4.1](#)

GNU/Linux



- + [Scilab 5.4.1 - Linux 32 bits](#) 163,30 MB
- + [Scilab 5.4.1 - Linux 64 bits](#) 160,13 MB

Windows XP, Vista, 7, 8



- + [Scilab 5.4.1 - Windows 32 bits](#) 117,09 MB
- + [Scilab 5.4.1 - Windows 64 bits](#) 124,41 MB

Si vous ne savez pas quelle version télécharger, choisissez la 32 bits.

Mac OS X (Plateformes Intel uniquement)



- + [Scilab 5.4.1 - Mac OS X](#) 135,24 MB

Sources

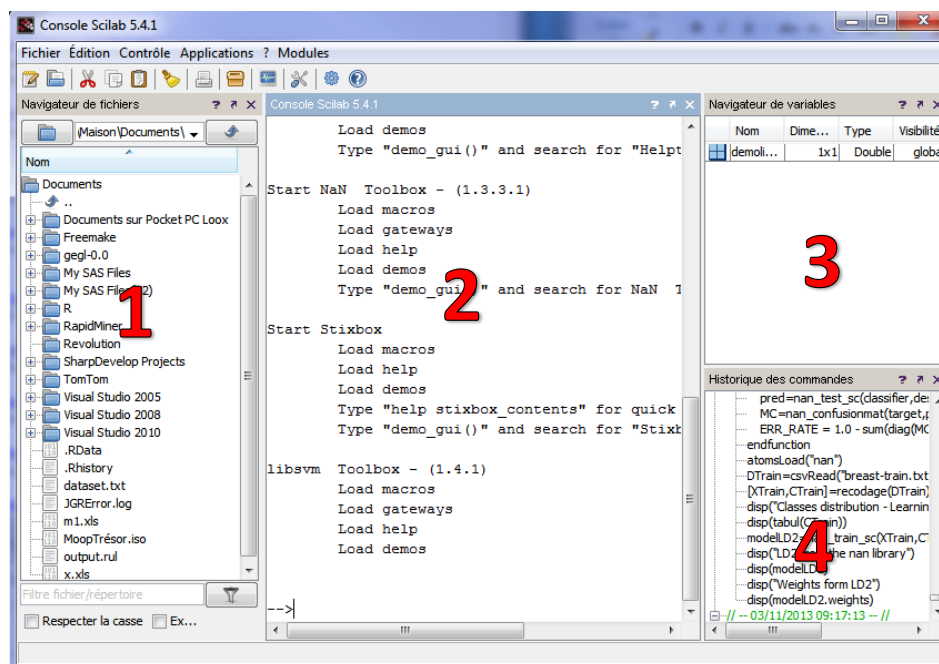


- + [Scilab 5.4.1 source version](#)

Autres versions : [Versions précédentes](#) | [Nightly Builds](#) | [Git Version](#)
 Pour être averti par email des sorties de version, inscrivez-vous sur [notre canal de diffusion](#)

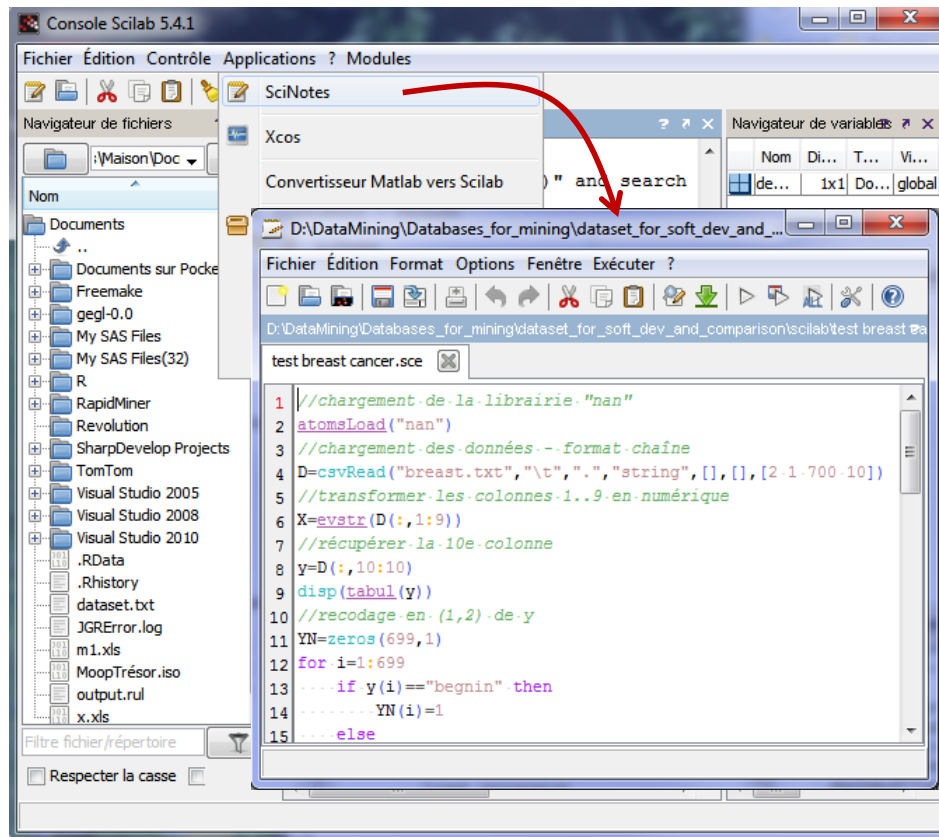
2.3 Launching Scilab

When we launch Scilab, we obtain a workspace subdivided in 4 parts.

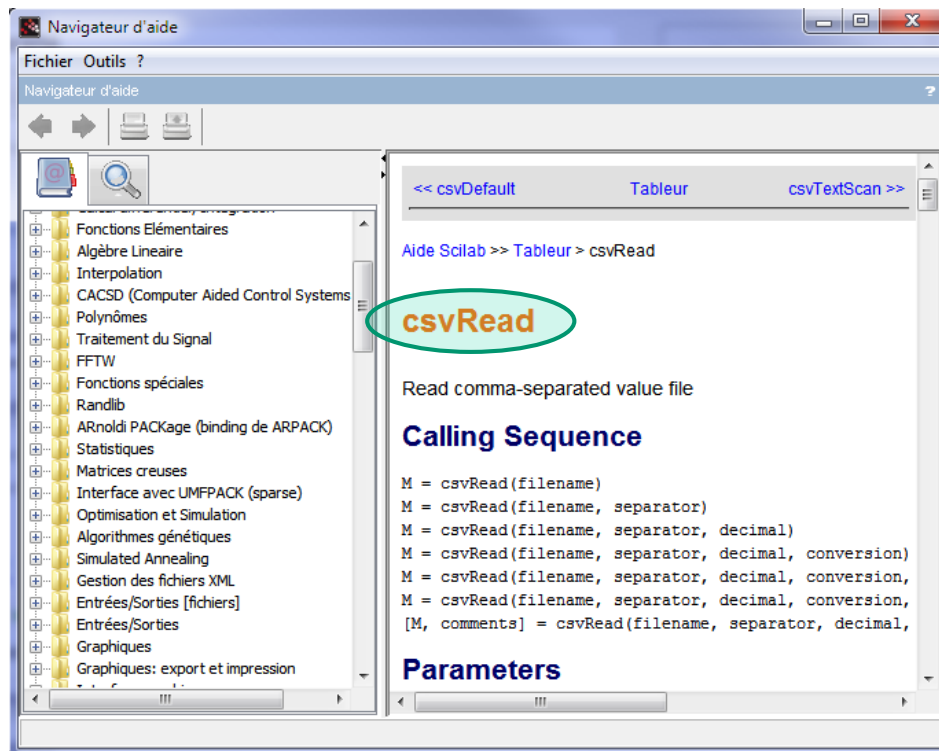


(1) This is a file browser. (2) The command editor enables us to enter the instructions and visualize the results. (3) The objects available into the memory are shown here. (4) The list of last entered commands is available here. From a certain perspective, the organization of the workspace is similar to those of the specialized editors for R such as R-Studio.

For batch processing, it is better to use the code editor: SCINOTES. We click on the APPLICATIONS / SCINOTES menu. A new editor window appears. It enables to handle Scilab program files (*.sce).



We use the “help” command to get help. E.g. `help('csvRead')` shows the description about the corresponding command.



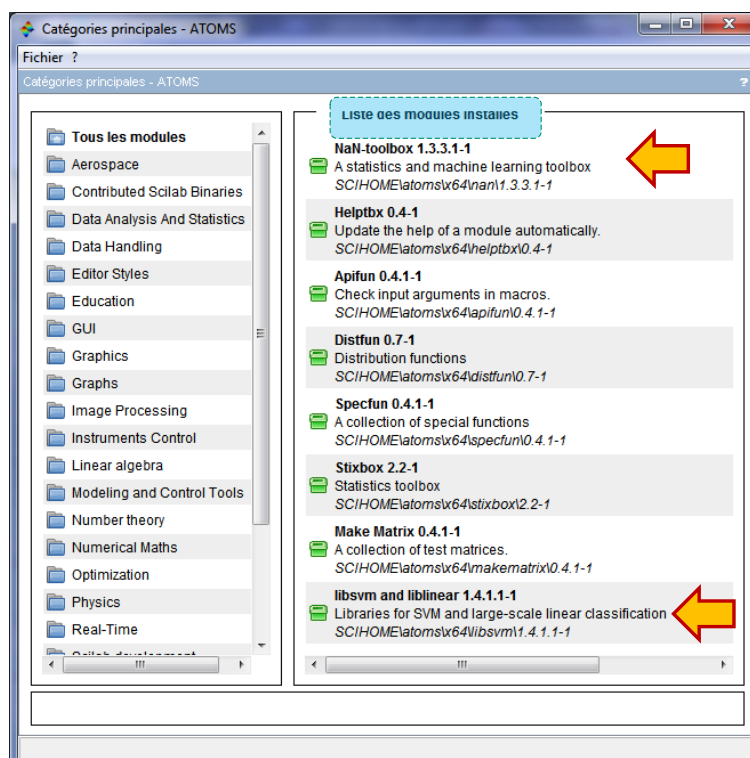
2.4 External libraries: the “toolbox”

Scilab provides a mechanism for managing packages ('Toolbox'). It enables to enrich its library of numerical methods. This kind of feature is, among others, the reason of the great success of R in recent years. Scilab can achieve the same success. It is nevertheless noted the relative scarcity of modules for data mining and statistical computing available to date, and more boring in my opinion, the almost total lack of detailed documentation about them.

The list of available toolboxes can be found on the ATOMS website.



We can obtain also the same list with the `atomsGui()` command.



A window for managing toolboxes appears. We observe for instance that the libraries "nan" and "libsvm" for statistics and machine learning are available. We can use this tool to install a toolbox. We can use also the `atomsInstall('name of the toolbox')` and `atomsLoad('name of the toolbox')` commands. The installed toolboxes are automatically loaded at each starting of Scilab.

3 Classification task with Scilab

3.1 Dataset

We use the BREAST-CANCER-WISCONSIN² dataset in this tutorial. There are 9 continuous descriptors and the class attribute CLASSE (2 values: BEGNIN and MALINGNANT). We have subdivided the dataset into two parts: the train set with 399 instances (breast-train.txt), the test set with 300 instances (breast-test.txt). The columns are separated by the tab stop character. We show below the first rows of the "breast-train.txt" data file.

	clump	ucellsize	ucellshape	mgadhesion	sepics	bnuclei	bchromatin	normnucl	mitoses	classe
1	4	2	2	1	2	1	2	1	1	begnin
2	1	1	1	1	2	1	2	1	1	begnin
3	2	1	1	1	2	1	2	1	1	begnin
4	10	6	6	2	4	10	9	7	1	malignant
5	4	1	1	1	2	1	2	1	1	begnin
6	1	1	1	1	2	1	2	1	1	begnin
7	3	1	1	1	2	1	2	1	1	begnin
8	5	3	3	2	3	1	3	1	1	begnin
9	4	2	2	1	2	1	2	1	1	begnin
10										

3.2 Importing the data file and variables coding

We load the data file into the "DTrain" matrix using the `csvRead()` command. The last parameter defines the data range into the file: from the 2nd row and the 1st column to the 400th row and the 10th column. At this time, we consider that all the values are in the character format.

```
//loading the data file
DTrain=csvRead("breast-train.txt","\t",".", "string", [], [], [2 1 400 10])
```

We show below the 5 first rows of the matrix.

```
//displaying the 5 first rows and all the columns of the matrix
disp(DTrain(1:5,:)) //nous aurions pu écrire disp(DTrain(1:5,1:$))
```

We obtain:

The screenshot shows the Scilab interface with the following components:

- Console Scilab 5.4.1:** Displays the command `-->disp(DTrain(1:5,:))` and its output:


```
!4 2 2 1 2 1 2 1 1 begnin !
!
!1 1 1 1 2 1 2 1 1 begnin !
!
!2 1 1 1 2 1 2 1 1 begnin !
!
!10 6 6 2 4 10 9 7 1 malignant !
!
!4 1 1 1 2 1 2 1 1 begnin !
-->
```
- Navigateur de variables:** Shows a table of variables:

Nom	Dimension	Type	Visibilité
DTrain	399x10	Chaîne de caractères	local
demolistock	1x1	Double	global
- Historique des commandes:** Shows the execution history, including the `csvRead` command used to load the data.

² <http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29>

We cannot use the data in the character format. We must specify to Scilab that the descriptors are numeric; and the class attribute (the 10th column) has 2 distinct values {1, 2}. For that, we write the function **recodage()**. We can use it for the test sample. The values of the descriptors are described in a matrix; the values of the class attribute are stored in a vector.

```
//own user function for data coding
function [descriptors, target]=recodage(D)
    //transforming the values of the descriptors in a numeric type
    descriptors = evstr(D(:,1:9))
    //retrieving the last column (10th column)
    y=D(:,10:10)
    //coding the class attribute with 1 and 2
    n=size(y,1) //n number of instances
    grp=zeros(n,1) //creating a vector with 0
    for i=1:n
        if y(i)=="begnin" then
            grp(i)=1
        else
            grp(i)=2
        end
    end
    target=grp
endfunction
```

We can make some comments about this function:

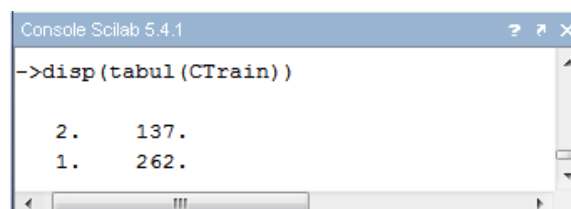
1. It is really easy to write a function under Scilab. The managing of a multiple output is also easy.
2. The **evstr()** function coerces the character values into the numeric type.
3. **size()** returns usually the size of the matrix. As it is used here, it returns the length of the vector i.e. the number of instances.
4. **zeros()** enables to initialize the vector with the value 0. This command enables also to set the size of the vector once for all and avoid its incremental resizing when we assign values in the loop. The successive memory allocation can lead to a very slow execution time.
5. The use of the **"for"** loop is identical to other programming languages.
6. It is the same about the **"if-then-else"** statement.

The program is written in a very academic way. It must be easy to improve it to make it more efficient. We note that the Scilab language is very easy to understand.

We call the function as following in the main program.

```
//calling the own user recodage() function
[XTrain,CTrain]=recodage(DTrain)
disp(tabul(CTrain))
```

XTrain is the matrix of the descriptors; CTrain is a vector corresponding to the class attribute. The **tabul()** function calculates the frequency distribution of the class attribute.



```
Console Scilab 5.4.1
->disp(tabul(CTrain))

 2. 137.
 1. 262.
```

3.3 Discriminant analysis using the « Nan » toolbox

You must install and load the 'Nan' Toolbox to be able to continue. It will be subsequently loaded automatically each time you start Scilab. We want to use the linear discriminant analysis (LD2) in a first step. The library provides several functions for the predictive discriminant analysis. I confess I do not know the differences between them. Unfortunately, the help of Scilab describes only the use of the procedures, not the underlying methods [cf. [help\('nan_train_sc'\)](#)].

We set the following command.

```
//learning phase - LD2
modelLD2=nan_train_sc(XTrain,CTrain,'LD2')
disp(modelLD2)
```

Scilab provides the following output.

```
-->disp(modelLD2)

datatype: "classifier:statistical:ld2"
Labels: [1,2]
MD: hypermat
NN: hypermat
weights: [10x2 constant]
```

modelLD2 is an object provides by the [nan_train_sc\(\)](#) function. Various properties are available. To obtain the weights of the classification functions, we set:

```
disp(modelLD2.weights)
```

The operator « . » enables to access to the properties of an object.

```
-->disp(modelLD2.weights)

    1.4421052  - 1.4421052
- 0.0717352   0.0717352
- 0.0435822   0.0435822
- 0.0258666   0.0258666
- 0.0091593   0.0091593
- 0.0131619   0.0131619
- 0.0841811   0.0841811
- 0.0499893   0.0499893
- 0.0291415   0.0291415
- 0.0153958   0.0153958
```

We do not obtain the same values as the other tools such as SAS or Tanagra. Because the help file does not provide more information about the underlying method, we cannot really put forward more comments here.

3.4 Applying the classifier on the test sample

We must perform several operations to apply the classifier on the test set and evaluate its performance: (1) loading the test sample; (2) coding the values using the same process as for the learning sample; (3) creating the prediction column by applying the classifier on the test set; (4) computing the confusion matrix with the predicted and observed values of the class attribute; (5) calculating the error rate from the confusion matrix.

For the steps (1) and (2), we use the same commands as for the learning sample.

```
//loading the test sample
DTest=csvRead("breast-test.txt","\t",".", "string", [], [], [2 1 301 10])

//coding
[XTest,CTest]=recodage(DTest)
disp(tabul(CTest))
```

The distribution of the target variable is very similar to that observed for the sample of learning. This is, among others, a simple way to verify that the subdivision of the dataset was carried out randomly.

```
->disp(tabul(CTest))

    2.    104.
    1.    196.
```

We define a new function for the steps (3), (4) and (5).

```
//function for the evaluation of a classifier
//input: classifier (object from nan toolbox)
//values for the descriptors (matrix)
//values for the class attribute (vector)
function [MC, ERR_RATE]=test_classifier(classifier, descriptors, target)
    //prediction
    pred=nan_test_sc(classifier,descriptors)
    //confusion matrix
    MC=nan_confusionmat(target,pred.classlabel)
    //error rate
    ERR_RATE = 1.0 - sum(diag(MC))/sum(MC)
endfunction
```

We apply this function on the test sample.

```
//evaluation of the classifier on the test sample
[mc,err_rate] = test_classifier(modelLD2,XTest,CTest)
disp(mc)
disp(err_rate)
```

We obtain the following output:

```
->[mc,err_rate] = test_classifier(modelLD2,XTest,CTest)
err_rate =

    0.0233333
mc =

    193.    3.
     4.    100.
```

7 instances over 300 are misclassified; the test error rate is 2.33%.

3.5 Predictive analytics using the « Naive Bayes Classifier »

We have implemented all the elements which allow us to apply the same process with another learning method. In this section, we build and evaluate the naive bayes classifier model 'NBC' provided by the "NaN" toolbox.

```
//learning phase - Naive Bayes Classifier
modelNBC=nan_train_sc(XTrain,CTrain, 'NBC')
```

```
//evaluation
[mc,err_rate] = test_classifier(modelNBC,XTest,CTest)
```

The test error rate is 2.66%.

```
->[mc,err_rate] = test_classifier(modelNBC,XTest,CTest)
err_rate =

    0.0266667
mc =

    190.    6.
     2.   102.
```

3.6 The « libsvm » toolbox

The famous LIBSVM³ library is also available for Scilab. We perform the train-test schema using a linear SVM classifier in this section. Of course, we must install and load the toolbox in a first step.

```
//'libsvm' toolbox - learning phase
//SVM - linear kerne '-t 0'
modellLINSVM=libsvm_svmtrain(CTrain,XTrain,'-t 0')
//prediction on the test sample
[predLINSVM]=libsvm_svmpredict(CTest,XTest,modellLINSVM)
//confusion matrix
mc=libsvm_confmat(CTest,predLINSVM)
```

As output of the learning function [[libsvm_svmtrain](#)], Scilab provides the list of the support vectors (**Svs**) and the associated weights (**sv_coef**).

```
-->modelSVM=libsvm_svmtrain(CTrain,XTrain)
*
optimization finished, #iter = 302
nu = 0.168887
obj = -40.457819, rho = 0.742082
nSV = 180, nBSV = 27
Total nSV = 180
modelSVM =

  Parameters: [5x1 constant]
  nr_class: 2
  totalSV: 180
  rho: 0.7420821
  Label: [2x1 constant]
  ProbA: [0x0 constant]
  ProbB: [0x0 constant]
  nSV: [2x1 constant]
  sv_coef: [180x1 constant]
  SVs: [180x9 constant]
```

From the prediction column provided by [[libsvm_svmpredict](#)], we compute the confusion matrix using the predefined [[libsvm_confmat](#)] function. Here is the confusion matrix.

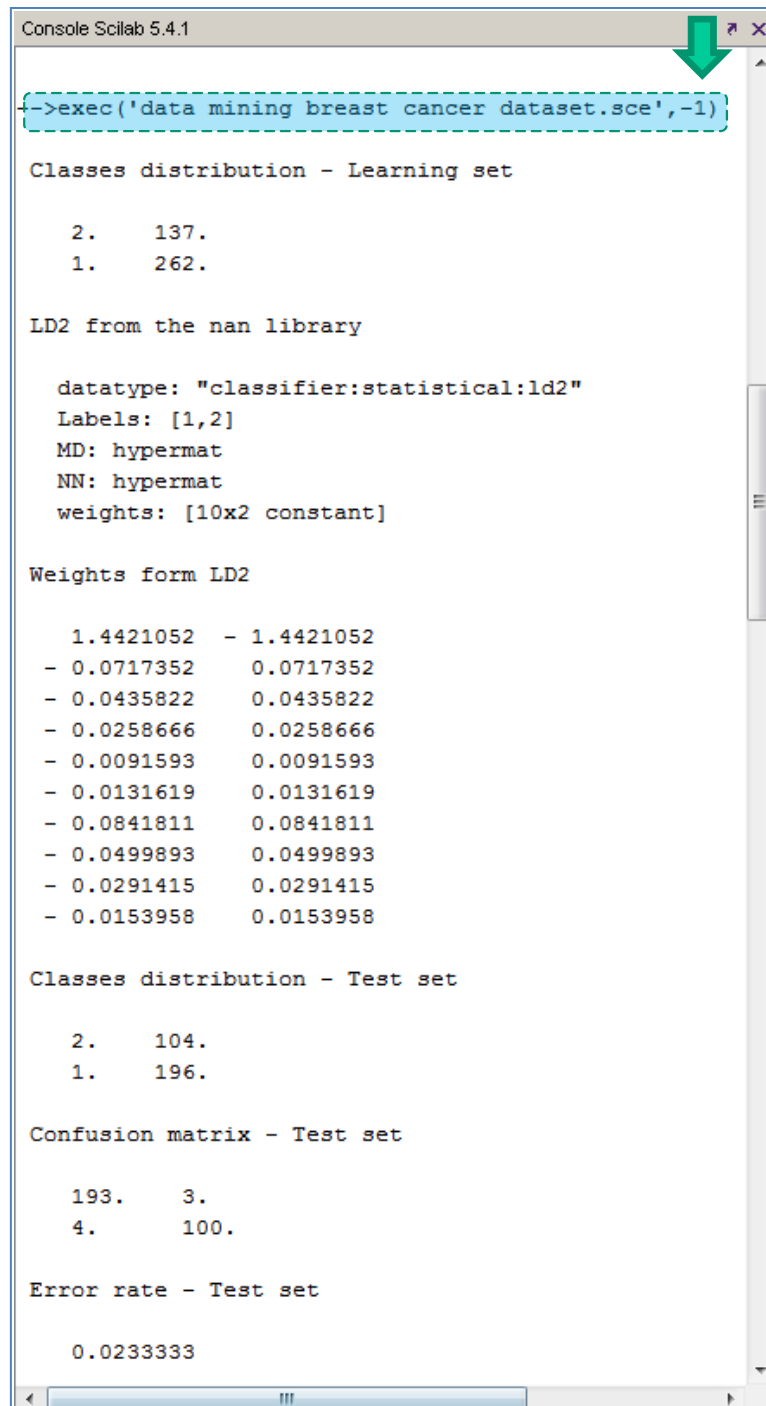
```
-->mc=libsvm_confmat(CTest,predLINSVM)
mc =

    193.    3.
     3.   101.
```

³ <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

4 Batch processing under Scilab

We have gathered the program for this tutorial in the "**data mining breast cancer dataset.sce**" file by adding some commands (intermediate outputs, SVM with RBF kernel, etc.). We can run all processes using the `exec()`⁴ function. We visualize only the information that we have explicitly requested with the `disp()` function, this facilitates the reading of the results.



```
Console Scilab 5.4.1
-->exec('data mining breast cancer dataset.sce',-1)

Classes distribution - Learning set

    2.    137.
    1.    262.

LD2 from the nan library

    datatype: "classifier:statistical:ld2"
    Labels: [1,2]
    MD: hypermat
    NN: hypermat
    weights: [10x2 constant]

Weights form LD2

    1.4421052  - 1.4421052
- 0.0717352   0.0717352
- 0.0435822   0.0435822
- 0.0258666   0.0258666
- 0.0091593   0.0091593
- 0.0131619   0.0131619
- 0.0841811   0.0841811
- 0.0499893   0.0499893
- 0.0291415   0.0291415
- 0.0153958   0.0153958

Classes distribution - Test set

    2.    104.
    1.    196.

Confusion matrix - Test set

    193.    3.
     4.    100.

Error rate - Test set

    0.0233333
```

⁴ We must modify the default directory before performing the command in the screenshot.

5 Conclusion

Scilab is fully suited to the data mining process. It provides all the tools to handling a dataset and achieving a statistical analysis. In addition, if we are accustomed to R (or similar programming language), learning the syntax of Scilab is easy.

Nevertheless, there are some weaknesses that we highlight below:

1. The toolboxes for statistical processing and data mining remain few in number, particularly compared with the packages for R ([2013/11/04], **4986** “official” packages are available for R - <http://cran.univ-lyon1.fr/web/packages/index.html>).
2. The toolboxes are not well documented. I have primarily used the source code of the macros to learn how to use the functions described in this tutorial (e.g. [nan_train_sc.sci](#)).
3. The output of the functions (from "NaN" and "libsvm" at least) are not really well detailed. It is a real obstacle on their use.
4. The data management must be improved. For instance, a special data type for the handling of categorical variables (such as the factor type under R) is needed.

There is a nothing insurmountable. The number of libraries will evolve positively. The existing toolboxes will be enhanced over the years. About the documentation, we can contribute to their improvement.