1 Topic

WRAPPER feature selection method with SIPINA and R (RWeka package). Comparison with a FILTER approach implemented into TANAGRA.

Feature selection. The feature selection[1] is a crucial aspect of supervised learning process. We must determine the relevant variables for the prediction of the target variable. Indeed, a simpler model is easier to understand and interpret; the deployment will be facilitated, we need less information to collect for prediction; finally, a simpler model is often more robust in generalization i.e. when we want to classify an unseen instance from the population.

Three kinds of approaches are often highlighted into the literature.

The FILTER approaches try to detect the relevant descriptors, independently of the used supervised learning algorithm. The advantage is the quickness and the flexibility. But, there is no guarantee however that the selection based on ad hoc criteria produces the best subset of features whatever the learning method used thereafter. The RANKING methods are certainly the more representative of this family. They calculate an indicator which characterizes the relationship between target attribute and each descriptor (input attribute). We ordered them by the decreasing value of the statistical indicator. Then we choose the best ones. We can also use a statistical hypothesis test in order to select the variables having a significant association with the target variable.

About the EMBEDDED approach, the selection process is integrated into the learning process. The induction of decision trees illustrates perfectly this approach. For instance, with the CART method (Breiman et al., 1984), the Gini index is used to select the segmentation variable on a node, the goal being to obtain the leaves as pure as possible. The algorithm selects automatically the smallest subset of variable in order to obtain them. But consistency does not mean performance. Let us not forget that one aim of the learning process is to produce a classifier with the best generalization capabilities. This is the reason for which CART also used a post-pruning system based on a criterion directly related to performance, different from that used during the tree growing.

The WRAPPER approach uses explicitly a performance criterion during the search of the best subset of descriptors. Most often, this is the error rate. But in reality, any kind of criteria can be used. This may be the cost if we use a misclassification cost matrix. It can be the area under curve (AUC) when we assess the classifier using ROC curves, etc. In this case, the learning method is considered as a black box. We try various subsets of predictors. We will choose the one that optimizes the criterion. In the WRAPPER approach, the result relies heavily on the search algorithm and on the assessment of the performance.

Search strategy. The search strategy is very important in the wrapper approach. We can use

[1] http://jmlr.csail.mit.edu/papers/special/feature03.html

a very simplistic method such as a greedy search, or more sophisticated methods such as genetic algorithm or simulated annealing. But there are two drawbacks to explore a very high number of solutions: we can reach to very specific solution to the used dataset which is not efficient into the population, it is the overfitting problem; the search becomes computationally intractable when we deal with a very large dataset with a large number of descriptors. Thus, hill climbing approaches such as forward search or backward search are in actuality a very good compromise in the most of the situations.

Prediction performance assessment. About the performance assessment, we cannot use the learning set in order to compute the criterion such as the error rate. Indeed, we often favor the complex solution with the largest number of descriptors in this case. Most of the time, to avoid this drawback, we use a validation set (which is not the same as the test set used for the assessment of the final model) or a resampling schema in order to obtain a honest estimation of the error rate during the search process.

Supervised learning algorithm. The learning method is used as a black box in the WRAPPER process. Any method can be implemented. We chose the NAIVE BAYES classifier for several reasons: it is well adapted for categorical predictors; it does not incorporate an internal selection process; and it can be disturbed by irrelevant variables. Thus, the influence of the variable selection on the classifier performance will be particularly discernable.

In this tutorial, we implement the WRAPPER approach with **SIPINA** and **R 2.9.2**. For this last one, we give the source code for a forward search strategy. The readers can easily adapt the program to other dataset. Moreover, a careful reading of the source code for R gives a better understanding about the calculations made internally by SIPINA.

The WRAPPER strategy is a priori the best since it explicitly optimizes the performance criterion. We verify this by comparing the results with those provided by the FILTER approach (FCBF method) available into **TANAGRA**. The conclusions are not as obvious as one can think.

2 Dataset

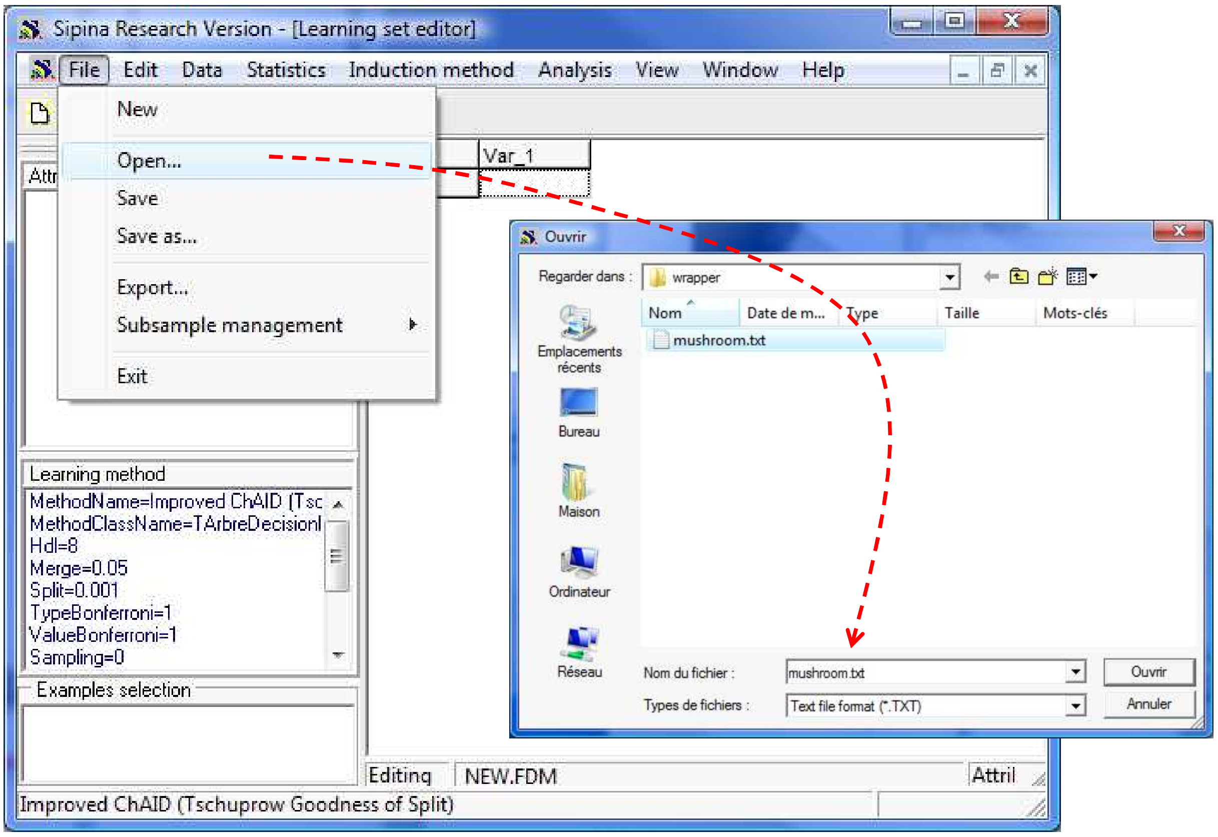
We use the MUSHROOM.TXT dataset in this tutorial (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/mushroom_wrapper.zip). It is well known in the machine learning community. We want to predict if a mushroom is edible or not from their characteristics. We have randomly subdivided the data file into a training sample (2000 instances) and a testing sample (6124 instances). This last one is used only to assess the efficiency of the final classifier based on the selected descriptors. It is not used during the search process. The STATUT column is used to discern the two parts of the data file.

3 Wrapper feature subset selection with SIPINA

The WRAPPET strategy is available into SIPINA (http://eric.univ-lyon2.fr/~ricco/sipina.html). This tool is mainly intended to the decision tree learning, but others supervised learning methods are also available.

3.1 Importing the data file

We launch SIPINA, we click on the FILE / OPEN menu. We select the MUSHROOM.TXT data file.



Learning method

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

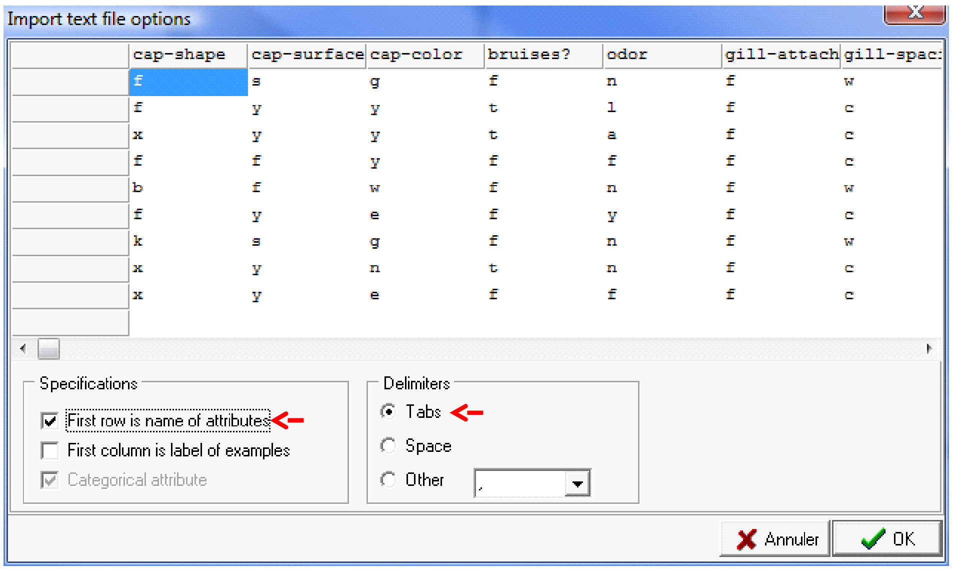
Examples selection

Nom du fichier : mushroom.txt

Types de fichiers : Text file format (*.TXT)

Improved ChAID (Tschuprow Goodness of Split)

A dialog box appears. We specify the configuration of our data file. In our case, the column separator is the tabulation; the first row corresponds to the variable names.

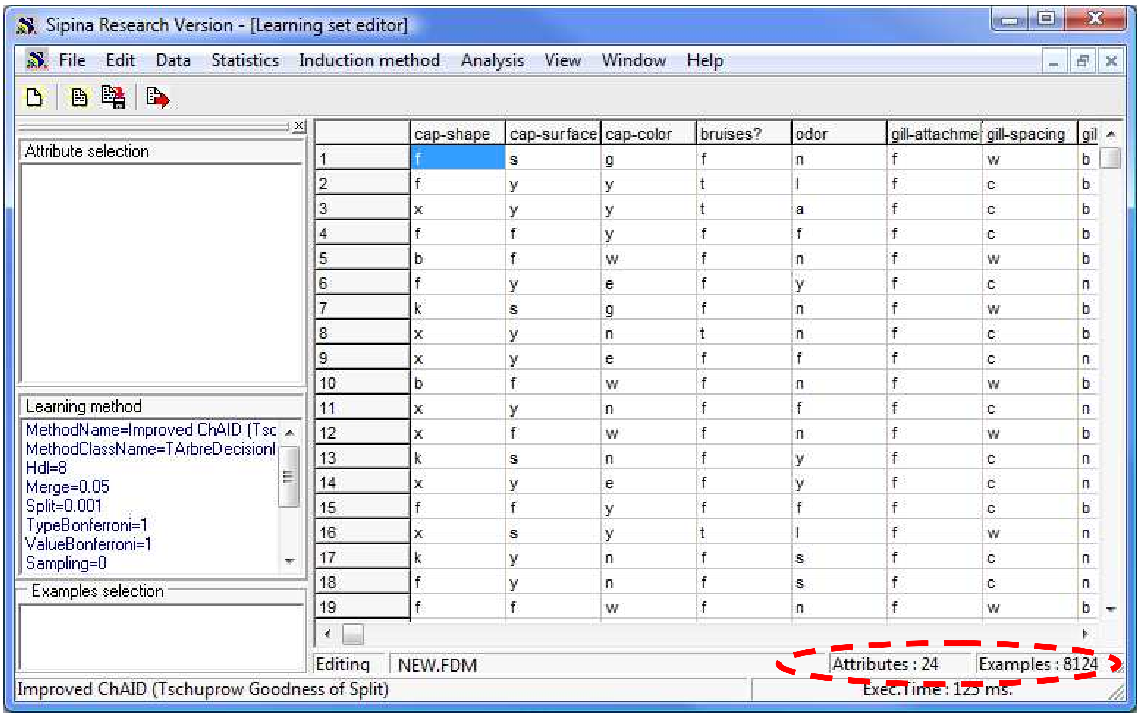


cap-shape	cap-surface	cap-color	bruises?	odor	gill-attach	gill-spac:
f	s	g	f	n	f	w
f	y	y	t	l	f	c
x	y	y	t	a	f	c
f	f	y	f	f	f	c
b	f	w	f	n	f	w
f	y	e	f	y	f	c
k	s	g	f	n	f	w
x	y	n	t	n	f	c
x	y	e	f	f	f	c

Specifications
- [✓] First row is name of attributes
- [ ] First column is label of examples
- [✓] Categorical attribute

Delimiters
- ● Tabs
- ○ Space
- ○ Other ,

The data file is loaded.



Sipina Research Version - [Learning set editor]

	cap-shape	cap-surface	cap-color	bruises?	odor	gill-attachme	gill-spacing	gil
1	f	s	g	f	n	f	w	b
2	f	y	y	t	l	f	c	b
3	x	y	y	t	a	f	c	b
4	f	f	y	f	f	f	c	b
5	b	f	w	f	n	f	w	b
6	f	y	e	f	y	f	c	n
7	k	s	g	f	n	f	w	b
8	x	y	n	t	n	f	c	b
9	x	y	e	f	f	f	c	n
10	b	f	w	f	n	f	w	b
11	x	y	n	f	f	f	c	n
12	x	f	w	f	n	f	w	b
13	k	s	n	f	y	f	c	n
14	x	y	e	f	y	f	c	n
15	f	f	y	f	f	f	c	b
16	x	s	y	t	l	f	w	n
17	k	y	n	f	s	f	c	n
18	f	y	n	f	s	f	c	n
19	f	f	w	f	n	f	w	b

Learning method

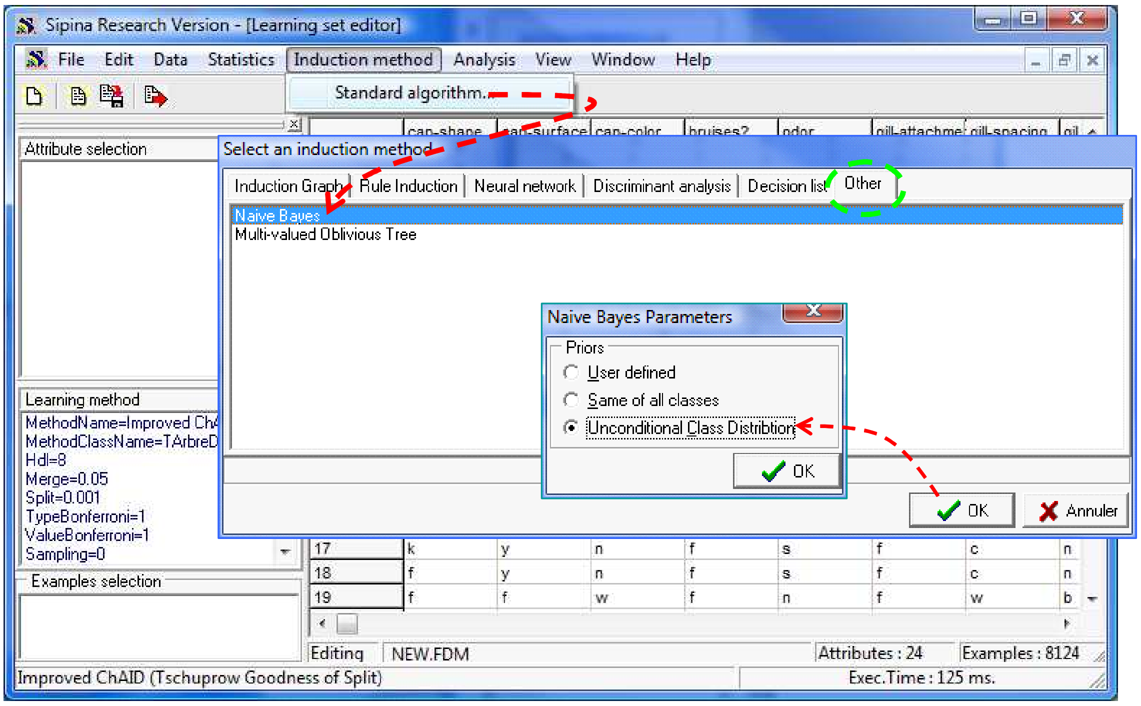
MethodName=Improved ChAID (Tsc
MethodClassName=TArbreDecisionI
Hdl=8
Merge=0.05
Split=0.001
TypeBonferroni=1
ValueBonferroni=1
Sampling=0

Editing NEW.FDM Attributes : 24 Examples : 8124

Improved ChAID (Tschuprow Goodness of Split) Exec.Time : 125 ms.

3.2 Choosing the learning algorithm

We use the naïve bayes classifier (http://en.wikipedia.org/wiki/Naive_Bayes_classifier). In spite it relies on an apparent unrealistic assumption, the naive bayes classifier works well in many practical situations. In order to define the learning method under SIPINA, we click on the INDUCTION METHOD / STANDARD ALGORITHM menu. We select the OTHER tab into the dialog settings. We click on the NAÏVE BAYES approach. In the following dialog box, we can define the parameters of the method. We keep the default parameters.



Sipina Research Version - [Learning set editor]

Select an induction method

Naive Bayes
Multi-valued Oblivious Tree

Naive Bayes Parameters

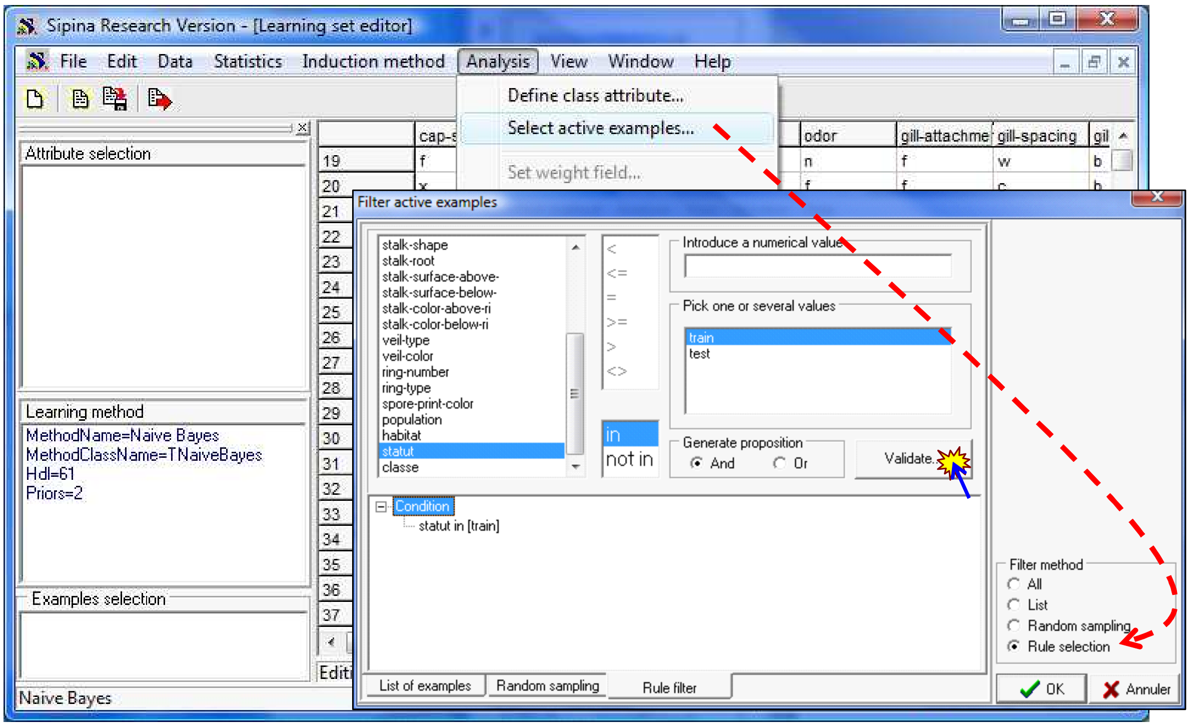
Priors
( ) User defined
( ) Same of all classes
(•) Unconditional Class Distribution

Editing NEW.FDM Attributes : 24 Examples : 8124

Improved ChAID (Tschuprow Goodness of Split) Exec.Time : 125 ms.

3.3 Partitioning the dataset (train and test samples)

We use the STATUT column in order to subdivide the dataset. We click on the ANALYSIS / SELECT ACTIVE EXAMPLES menu. We select the RULE SELECTION option i.e. the subdivision is based on a logical rule. We set the rule **[STATUT] IN [TRAIN]** then we click on the VALIDATE button.

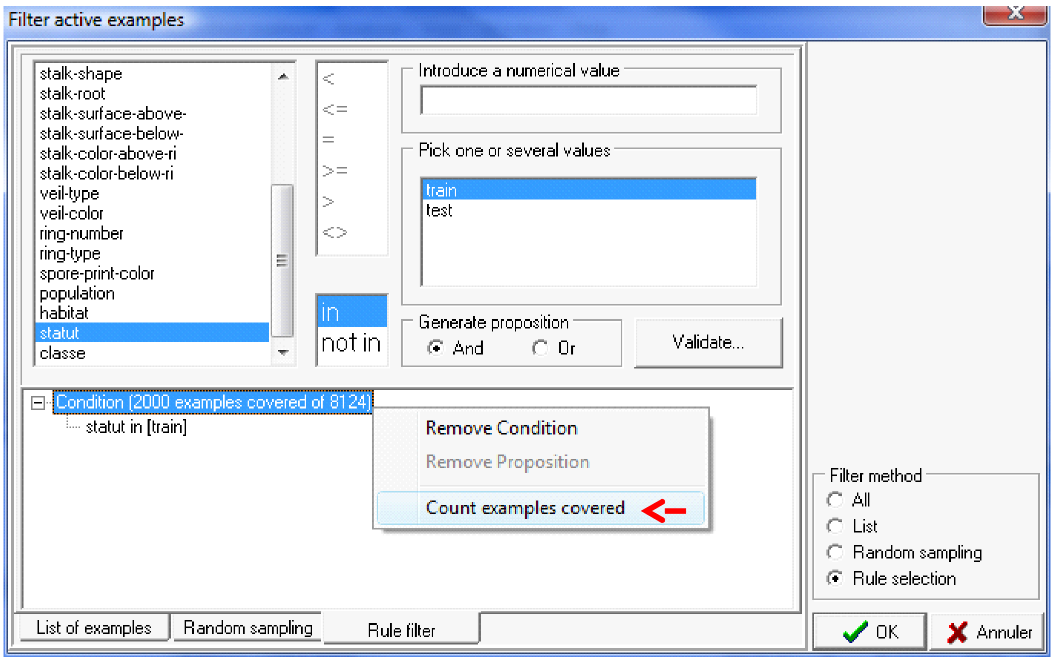


MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Condition
statut in [train]

Filter method: [✓ Rule selection]

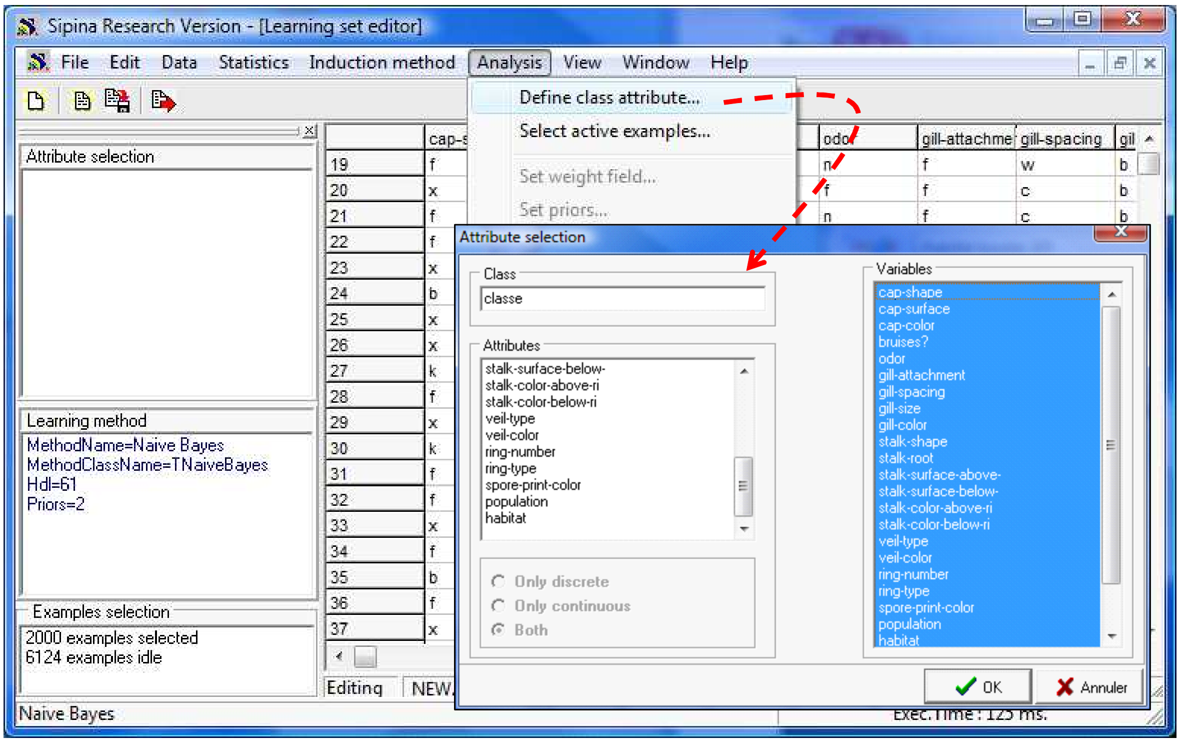
We can count the number of instances included into the learning sample by clicking on the COUNT EXAMPLES COVERED contextual menu: 2000 instances among 8124 are used during the search process.



Condition (2000 examples covered of 8124)
statut in [train]

3.4 Defining the target and the input attributes

We click on the ANALYSIS / DEFINE CLASS ATTRIBUTE menu in order to specify the target and the input attributes. Of course, the STATUT column is not used here.


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

Attribute selection

Class: classe

Attributes: stalk-surface-below-, stalk-color-above-ri, stalk-color-below-ri, veil-type, veil-color, ring-number, ring-type, spore-print-color, population, habitat

Only discrete / Only continuous / [✓ Both]

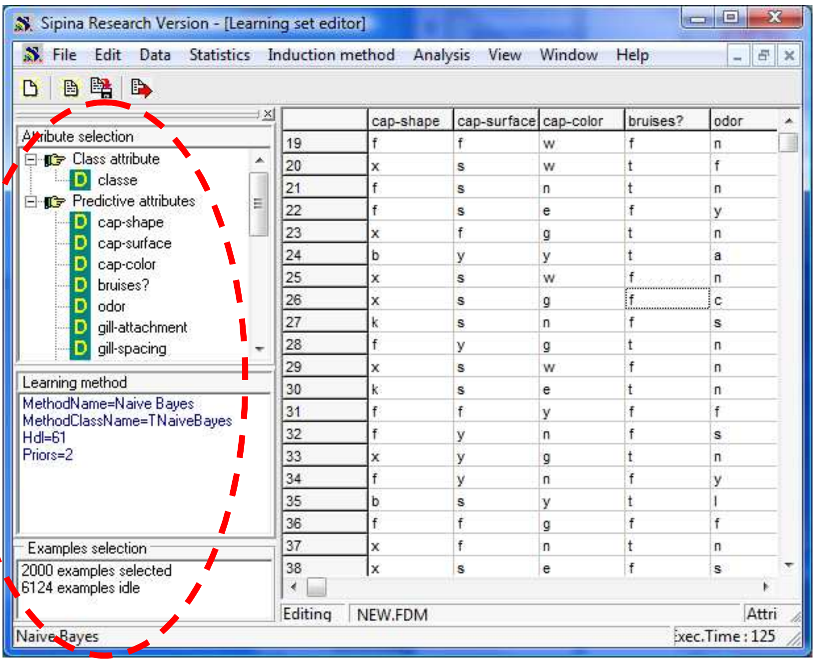
Variables: cap-shape, cap-surface, cap-color, bruises?, odor, gill-attachment, gill-spacing, gill-size, gill-color, stalk-shape, stalk-root, stalk-surface-above-, stalk-surface-below-, stalk-color-above-ri, stalk-color-below-ri, veil-type, veil-color, ring-number, ring-type, spore-print-color, population, habitat

Learning method
MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection
2000 examples selected
6124 examples idle

Naive Bayes

A summary of the configuration is available on the left part of the main window.


Sipina Research Version - [Learning set editor]
File Edit Data Statistics Induction method Analysis View Window Help
Editing
NEW.FDM
Attri
Exec.Time : 125
Naive Bayes

Attribute selection
- Class attribute
  - classe
- Predictive attributes
  - cap-shape
  - cap-surface
  - cap-color
  - bruises?
  - odor
  - gill-attachment
  - gill-spacing

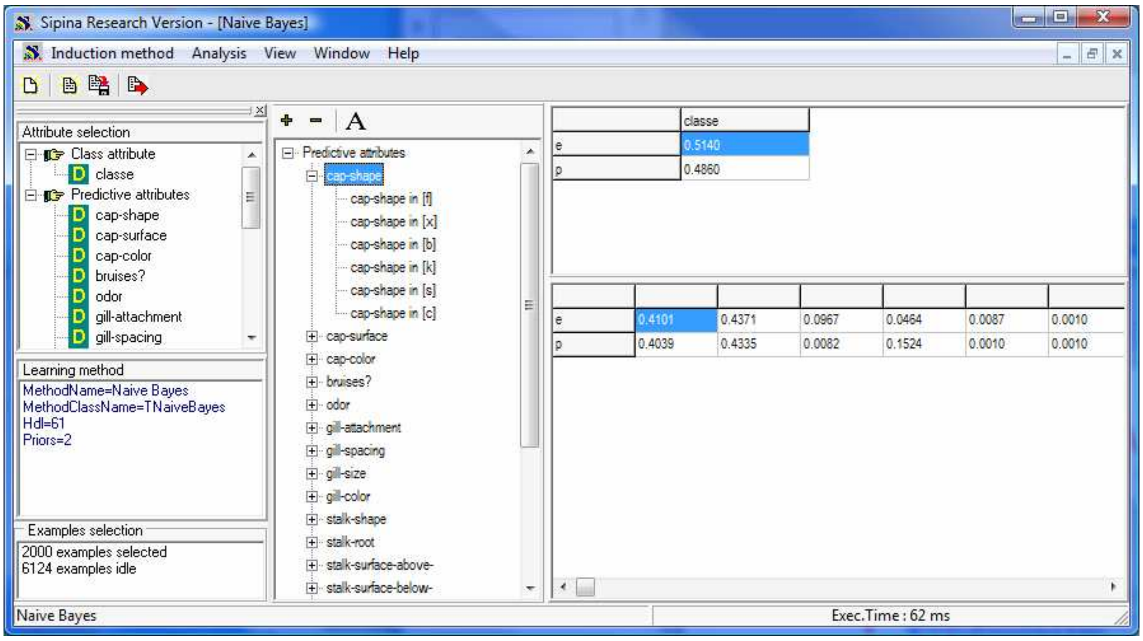
Learning method
MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection
2000 examples selected
6124 examples idle

	cap-shape	cap-surface	cap-color	bruises?	odor
19	f	f	w	f	n
20	x	s	w	t	f
21	f	s	n	t	n
22	f	s	e	f	y
23	x	f	g	t	n
24	b	y	y	t	a
25	x	s	w	f	n
26	x	s	g	f	c
27	k	s	n	f	s
28	f	y	g	t	n
29	x	s	w	f	n
30	k	s	e	t	n
31	f	f	y	f	f
32	f	y	n	f	s
33	x	y	g	t	n
34	f	y	n	f	y
35	b	s	y	t	l
36	f	f	g	f	f
37	x	f	n	t	n
38	x	s	e	f	s

3.5 Accuracy of the full model

The full model including all the input variables is our reference classifier. We learn the model on the learning set by clicking on the ANALYSIS / LEARNING menu. A new window appears. We can view the conditional probabilities for each descriptor. They are used for the classification of the unseen instances.



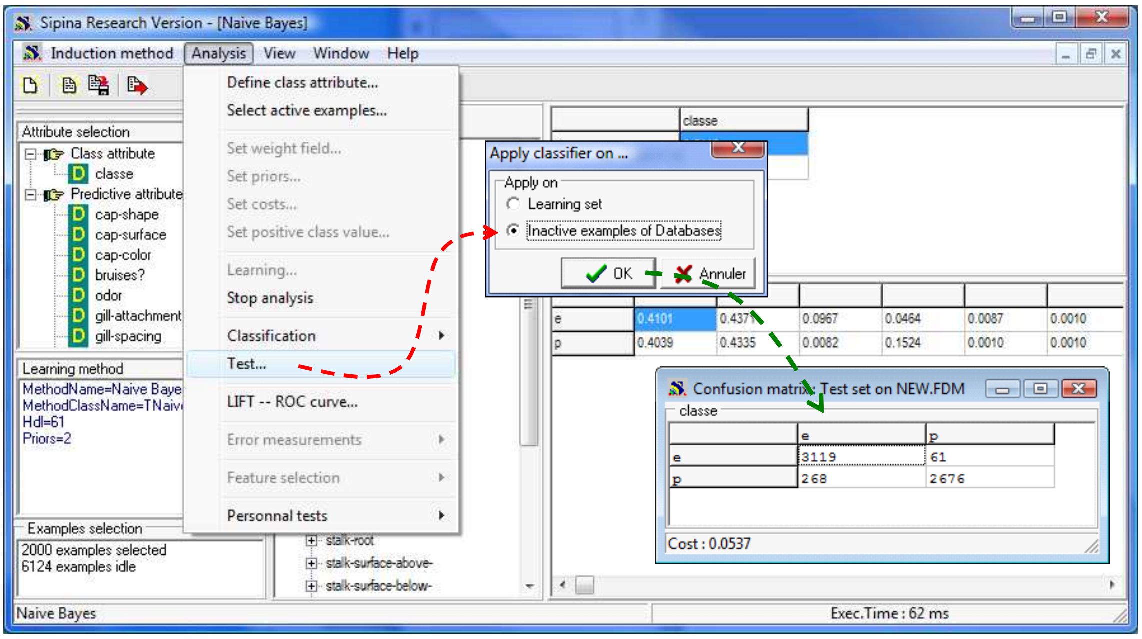
Learning method
MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection
2000 examples selected
6124 examples idle

	classe
e	0.5140
p	0.4860

e	0.4101	0.4371	0.0967	0.0464	0.0087	0.0010
p	0.4039	0.4335	0.0082	0.1524	0.0010	0.0010

We evaluate the performance by clicking on the ANALYSIS / TEST menu. We select the INACTIVE EXAMPLES OF THE DATABASE (the test sample) into the dialog box. We obtain an unbiased estimation of the generalization error rate because it is computed on the test set.



Apply on
( ) Learning set
(•) Inactive examples of Databases

Confusion matrix : Test set on NEW.FDM

classe

	e	p
e	3119	61
p	268	2676

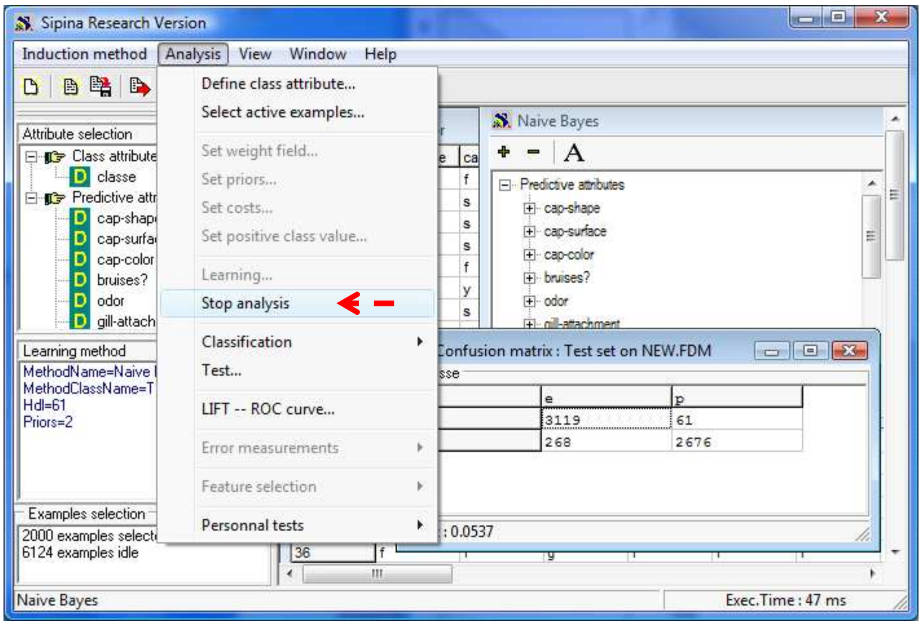
Cost : 0.0537

The confusion matrix is displayed. The test error rate is 5.37%.

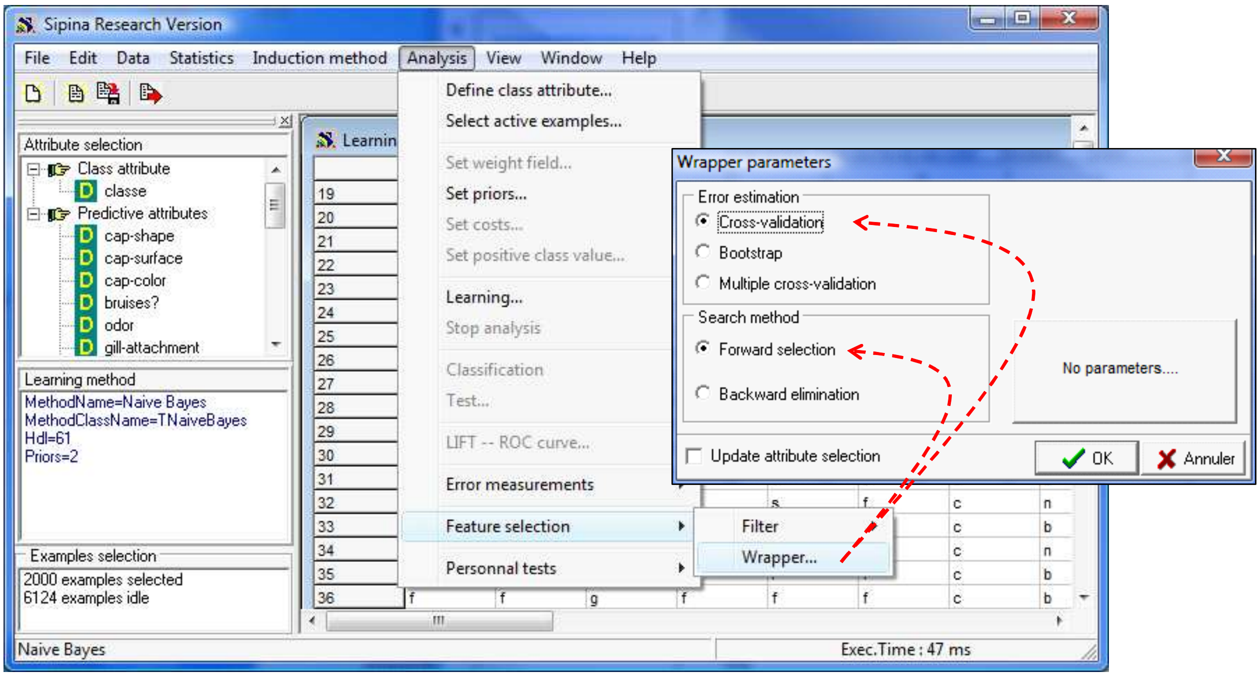
Now, we want to know if we can make better with the WRAPPER feature subset selection process.

3.6 WRAPPER approach with SIPINA

First, we must stop the current analysis session. We click on the ANALYSIS / STOP ANALYSIS menu. The preceding results (windows) are removed.



We activate the ANALYSIS / FEATURE SELECTION / WRAPPER menu for launching the selection process. In the dialog settings, we use the cross validation in order to estimate the error rate of each subset of features. The search strategy is the FORWARD approach i.e. nested subsets of variables are evaluated, from the model with only one descriptor to the model with all the descriptors.



Wrapper parameters

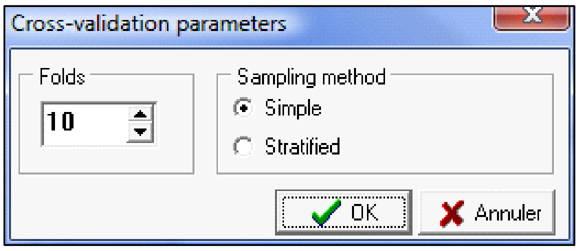
Error estimation: [✓ Cross-validation] Bootstrap / Multiple cross-validation

Search method: [✓ Forward selection] Backward elimination

No parameters....

Update attribute selection

Into the next dialog settings, we set the number of folds for the cross validation.

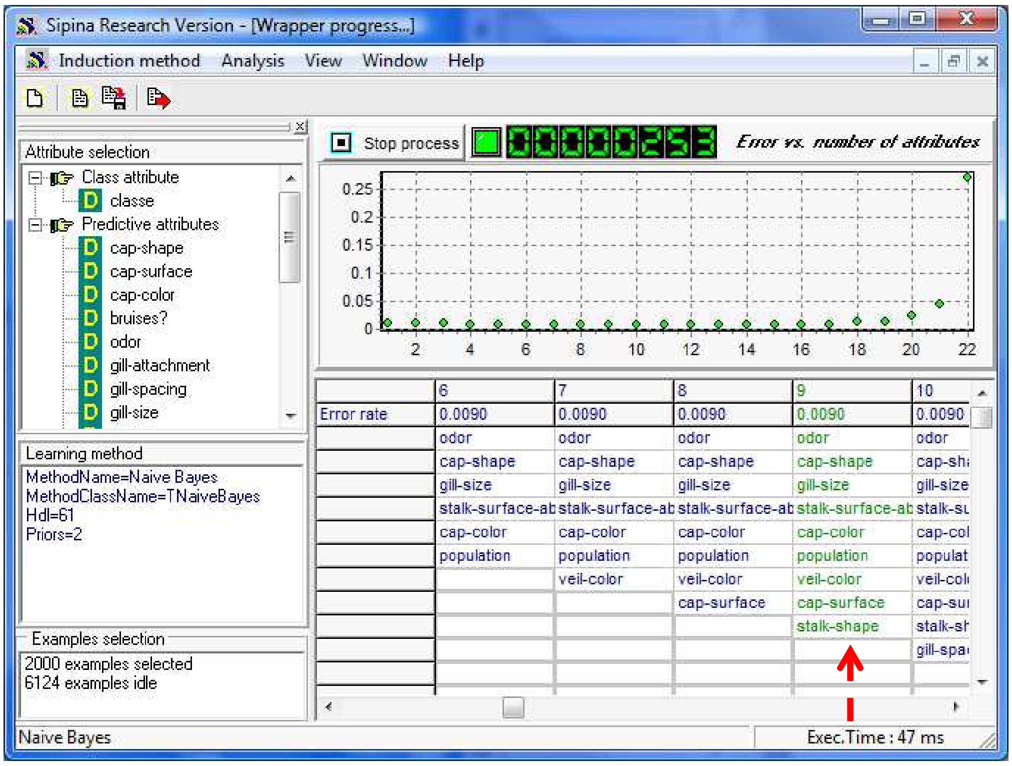


Folds: 10

Sampling method: [✓ Simple] Stratified

The calculation is started. For the selected subset at each step (1 variable, 2 variables, etc.), SIPINA displays the selected attributes and the corresponding error rate. The user can interactively stop the process.

Note: *Because the seed number of the random number generator is not the same according to the computer, you can obtain a slightly different solution on your computer*.


Sipina Research Version - [Wrapper progress...]
Induction method
Analysis
View
Window
Help
Stop process
Error vs. number of attributes

Attribute selection
- Class attribute
  - classe
- Predictive attributes
  - cap-shape
  - cap-surface
  - cap-color
  - bruises?
  - odor
  - gill-attachment
  - gill-spacing
  - gill-size

Learning method

MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection

2000 examples selected
6124 examples idle

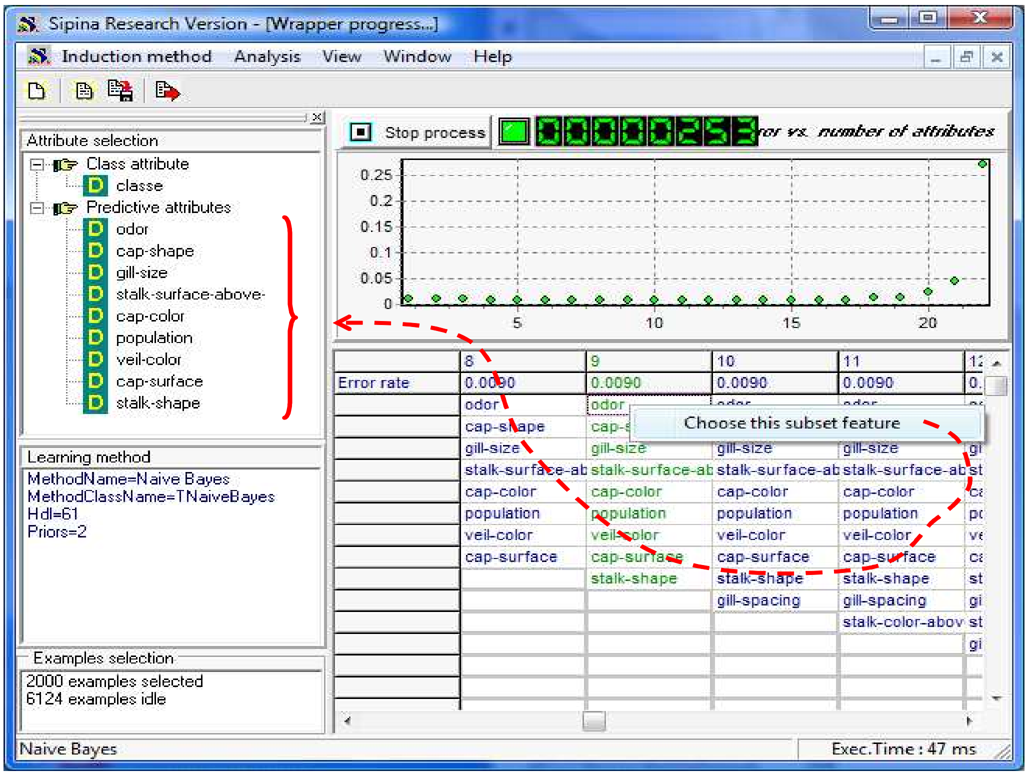
	6	7	8	9	10
Error rate	0.0090	0.0090	0.0090	0.0090	0.0090
	odor	odor	odor	odor	odor
	cap-shape	cap-shape	cap-shape	cap-shape	cap-sha
	gill-size	gill-size	gill-size	gill-size	gill-size
	stalk-surface-ab	stalk-surface-ab	stalk-surface-ab	stalk-surface-ab	stalk-su
	cap-color	cap-color	cap-color	cap-color	cap-col
	population	population	population	population	populat
		veil-color	veil-color	veil-color	veil-col
			cap-surface	cap-surface	cap-sur
				stalk-shape	stalk-sh
					gill-spa

Naive Bayes

Exec.Time : 47 ms

SIPINA highlights a subset with 9 descriptors. The cross validation error rate computed during the search is about 0.9%. But, for obtaining an unbiased estimation of the error rate, we must apply the resulting classifier on the test set.

We want to select the subset with these 9 descriptors (Figure 1). We set a right click on the right column of the grid; we click on the CHOOSE THIS SUBSET FEATURE contextual menu. In the left part of the window, the selected variables for the learning process are modified. ***Note***: *the best solution incorporates 9 descriptors, but we see into the chart above that solutions with fewer variables are very close according to the error rate*.


Sipina Research Version - [Wrapper progress...]
Induction method
Analysis
View
Window
Help
Stop process
00000253
Error vs. number of attributes
Choose this subset feature

Attribute selection
- Class attribute
  - classe
- Predictive attributes
  - odor
  - cap-shape
  - gill-size
  - stalk-surface-above-
  - cap-color
  - population
  - veil-color
  - cap-surface
  - stalk-shape

Learning method
MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection
2000 examples selected
6124 examples idle

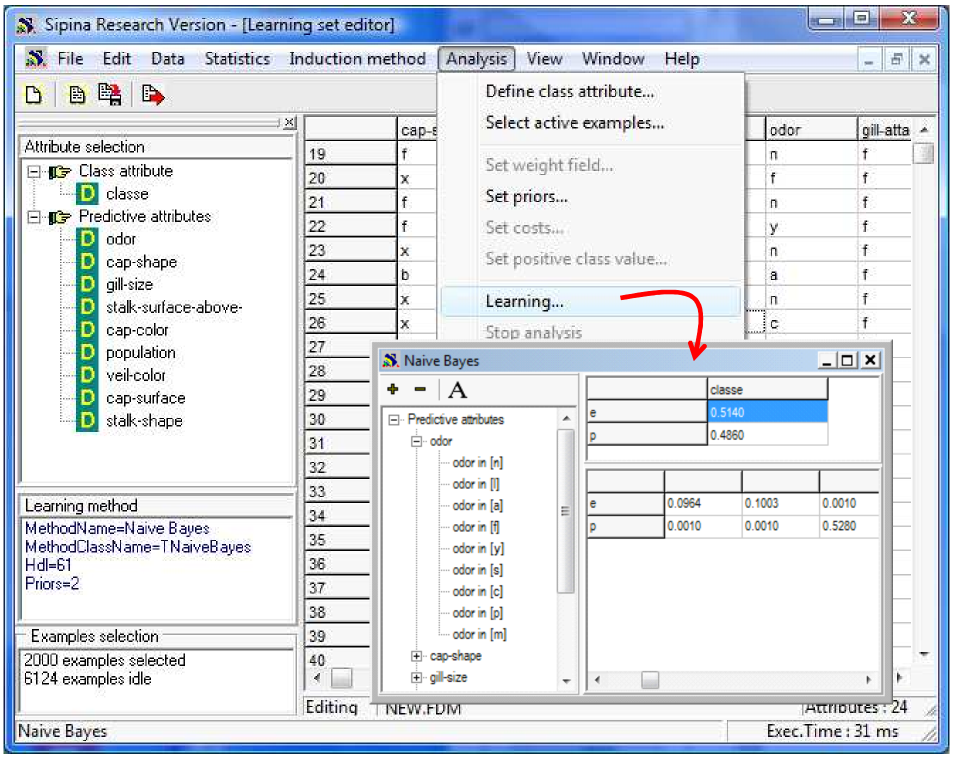
	8	9	10	11
Error rate	0.0090	0.0090	0.0090	0.0090
	odor	odor	odor	odor
	cap-shape	cap-s		
	gill-size	gill-size	gill-size	gill-size
	stalk-surface-ab	stalk-surface-ab	stalk-surface-ab	stalk-surface-ab
	cap-color	cap-color	cap-color	cap-color
	population	population	population	population
	veil-color	veil-color	veil-color	veil-color
	cap-surface	cap-surface	cap-surface	cap-surface
		stalk-shape	stalk-shape	stalk-shape
			gill-spacing	gill-spacing
				stalk-color-abov

Naive Bayes
Exec.Time : 47 ms

Figure 1 – Selecting the "best" subset of features

3.7 Accuracy of the simplified model

We want to assess the classifier with 9 variables on the test set. We click first on the ANALYSIS / LEARNING menu in order to create the classifier on the train set.



	classe
e	0.5140
p	0.4860

e	0.0964	0.1003	0.0010
p	0.0010	0.0010	0.5280

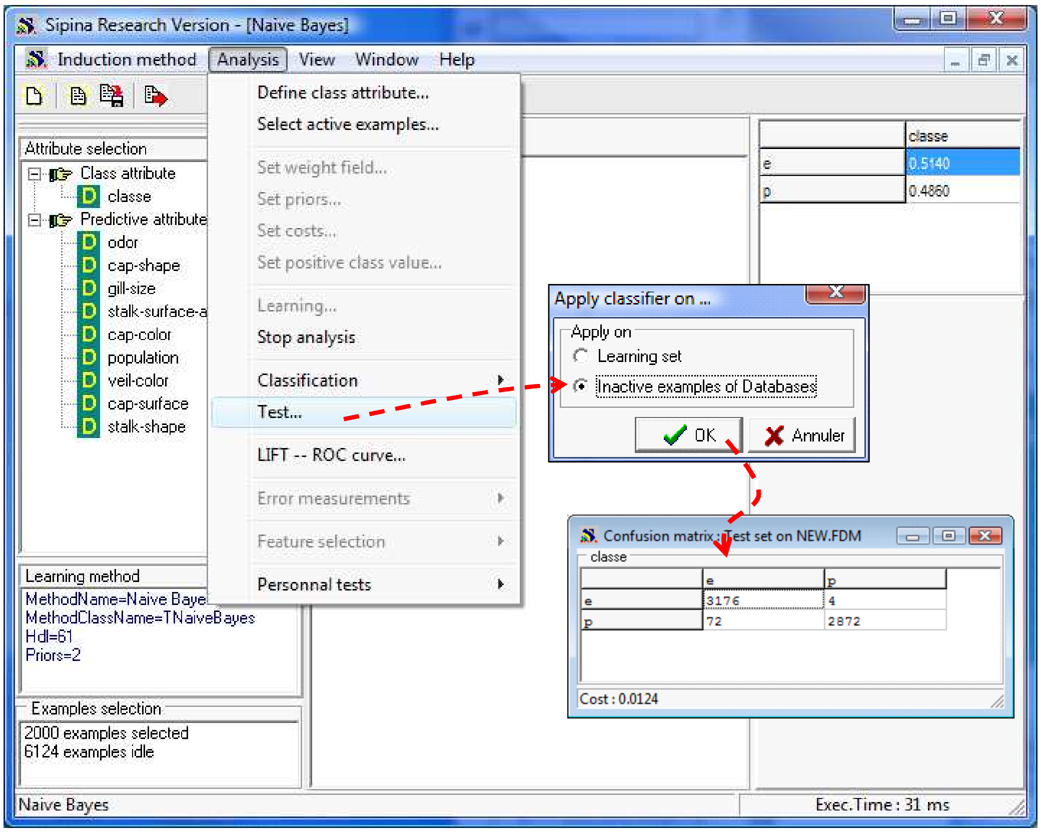
Attribute selection
- Class attribute
  - classe
- Predictive attributes
  - odor
  - cap-shape
  - gill-size
  - stalk-surface-above-
  - cap-color
  - population
  - veil-color
  - cap-surface
  - stalk-shape

Learning method
MethodName=Naive Bayes
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection
2000 examples selected
6124 examples idle

Editing | NEW.FDM | Attributes : 24
Naive Bayes | Exec.Time : 31 ms

Then we click on the ANALYSIS / TEST menu and we set the INACTIVE EXAMPLES OF DATABASE option in order to compute the confusion matrix on the test set.



	classe
e	0.5140
p	0.4860

Learning method
MethodName=Naive Baye
MethodClassName=TNaiveBayes
Hdl=61
Priors=2

Examples selection
2000 examples selected
6124 examples idle

Apply on: ( ) Learning set [✓ Inactive examples of Databases]

classe

	e	p
e	3176	4
p	72	2872

Cost : 0.0124

Exec.Time : 31 ms

The "true" test error rate is 1.24%. The simplified model is much better than the full model including all of the descriptors.

4 WRAPPER approach with R

In this section, we give the details of the source code for the forward selection according the wrapper approach under the R software (http://www.r-project.org/). The main advantage is that we can describe each step of the process.

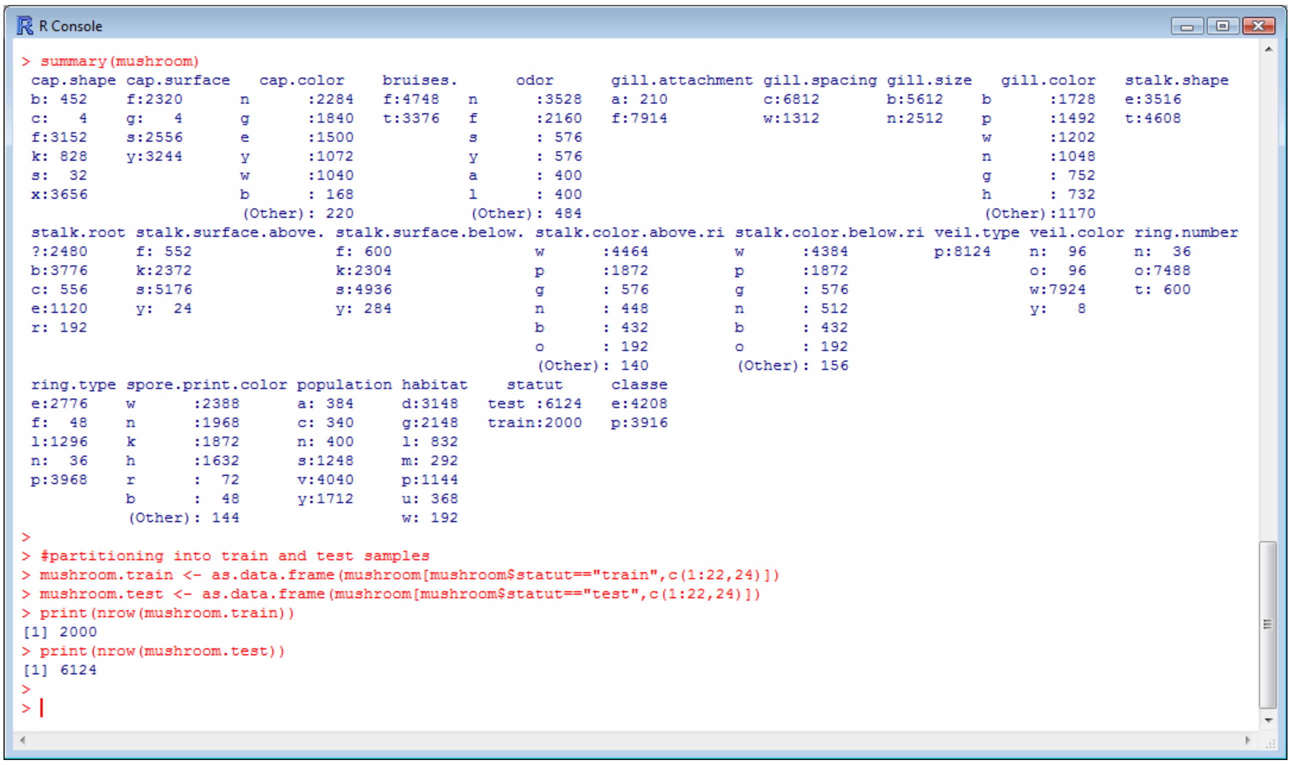
4.1 Importing and partitioning the dataset

First, we load the dataset that we subdivide into train and test samples.

```
#load the data file
mushroom <- read.table(file="mushroom.txt",header=TRUE,sep="\t")
summary(mushroom)

#partitioning into train and test samples
mushroom.train <- as.data.frame(mushroom[mushroom$statut=="train",c(1:22,24)])
mushroom.test <- as.data.frame(mushroom[mushroom$statut=="test",c(1:22,24)])
print(nrow(mushroom.train))
print(nrow(mushroom.test))
```

R shows the following results.



```
> summary(mushroom)
 cap.shape cap.surface   cap.color    bruises.      odor      gill.attachment gill.spacing gill.size   gill.color   stalk.shape
 b: 452    f:2320      n      :2284   f:4748   n      :3528   a: 210          c:6812       b:5612    b      :1728   e:3516
 c:   4    g:   4      g      :1840   t:3376   f      :2160   f:7914          w:1312       n:2512    p      :1492   t:4608
 f:3152    s:2556      e      :1500            s      : 576                                          w      :1202
 k: 828    y:3244      y      :1072            y      : 576                                          n      :1048
 s:  32                w      :1040            a      : 400                                          g      : 752
 x:3656                b      : 168            l      : 400                                          h      : 732
                       (Other): 220            (Other): 484                                          (Other):1170
 stalk.root stalk.surface.above. stalk.surface.below. stalk.color.above.ri stalk.color.below.ri veil.type veil.color ring.number
 ?:2480     f: 552               f: 600               w      :4464          w      :4384          p:8124    n:  96     n:  36
 b:3776     k:2372               k:2304               p      :1872          p      :1872                    o:  96     o:7488
 c: 556     s:5176               s:4936               g      : 576          g      : 576                    w:7924     t: 600
 e:1120     y:  24               y: 284               n      : 448          n      : 512                    y:   8
 r: 192                                               b      : 432          b      : 432
                                                      o      : 192          o      : 192
                                                      (Other): 140          (Other): 156
 ring.type spore.print.color population habitat    statut      classe
 e:2776    w      :2388      a: 384     d:3148   test :6124   e:4208
 f:  48    n      :1968      c: 340     g:2148   train:2000   p:3916
 l:1296    k      :1872      n: 400     l: 832
 n:  36    h      :1632      s:1248     m: 292
 p:3968    r      :  72      v:4040     p:1144
           b      :  48      y:1712     u: 368
           (Other): 144                 w: 192
>
> #partitioning into train and test samples
> mushroom.train <- as.data.frame(mushroom[mushroom$statut=="train",c(1:22,24)])
> mushroom.test <- as.data.frame(mushroom[mushroom$statut=="test",c(1:22,24)])
> print(nrow(mushroom.train))
[1] 2000
> print(nrow(mushroom.test))
[1] 6124
>
> |
```

The training set contains 2000 instances, the test set contains 6124 instances.

4.2 The full model – The naïve bayes classifier with the RWeka package

We use the RWeka package (http://cran.r-project.org/web/packages/RWeka/index.html) based on the famous Weka free software (http://www.cs.waikato.ac.nz/ml/weka/).

The **library(.)** command loads the package. Then we load the naïve bayes method with the **make_Weka_classifier(.)** instruction. We call **NB(.)** the method in our source code.

We launch the learning process and we evaluate the classifier on the test set with the **evaluate_Weka_classifier(.)** function.

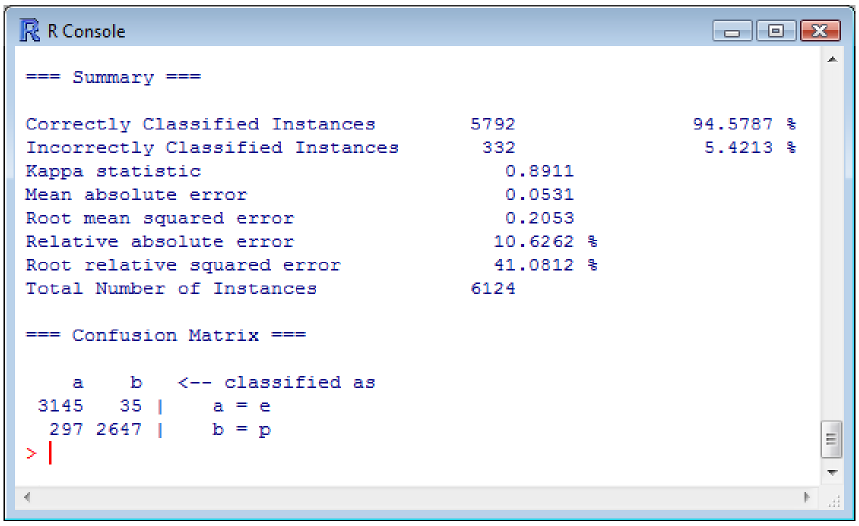
```
#load the RWeka package
library(RWeka)

#make naive bayes classifier function
NB <- make_Weka_classifier("weka/classifiers/bayes/NaiveBayes")

#training phase
full.model <- NB(classe ~ ., data = mushroom.train)

#evaluation on the test set
test.evaluation <- evaluate_Weka_classifier(full.model,newdata=mushroom.test)
print(test.evaluation)
```

We obtain the following results.



```
=== Summary ===

Correctly Classified Instances        5792               94.5787 %
Incorrectly Classified Instances       332                5.4213 %
Kappa statistic                          0.8911
Mean absolute error                      0.0531
Root mean squared error                  0.2053
Relative absolute error                 10.6262 %
Root relative squared error             41.0812 %
Total Number of Instances             6124

=== Confusion Matrix ===

    a    b   <-- classified as
 3145   35 |    a = e
  297 2647 |    b = p
> 
```

The test error rate is 5.24% when we use all the descriptors.

4.3 Wrapper process under R

First, we write a callback function **search_wrapper(.)** in order to evaluate each subset. We apply this function for each subset with the same cardinal. Then, we obtain the best solution at each step of the forward search i.e. the best subset with one descriptor, the best subset with 2 descriptors, etc., with the corresponding error rate obtained by cross validation.

```
#callback function
#searching the best predictive attribute
#for one step of the wrapper process
search_wrapper <- function(x,cur.dataset,K){
  #current dataset
  working <- cbind(cur.dataset,x)
  #learning the model
  cur.model <- NB(as.formula(paste(colnames(working)[1],"~ .")),data = working)
  #evaluating the model
  cur.eval <- evaluate_Weka_classifier(cur.model,numFolds = K,seed=100)
  #getting the error rate
  return (cur.eval$details["pctIncorrect"])
}
```

We note that the **evaluate_Weka_classifier(.)** function can implement a cross validation. **K** is the number of folds.

Last we can write the function which incorporates the callback function. We adopt a very simplistic code. I think we can make much better (faster) but I prefer a code that the readers can inspect easily.

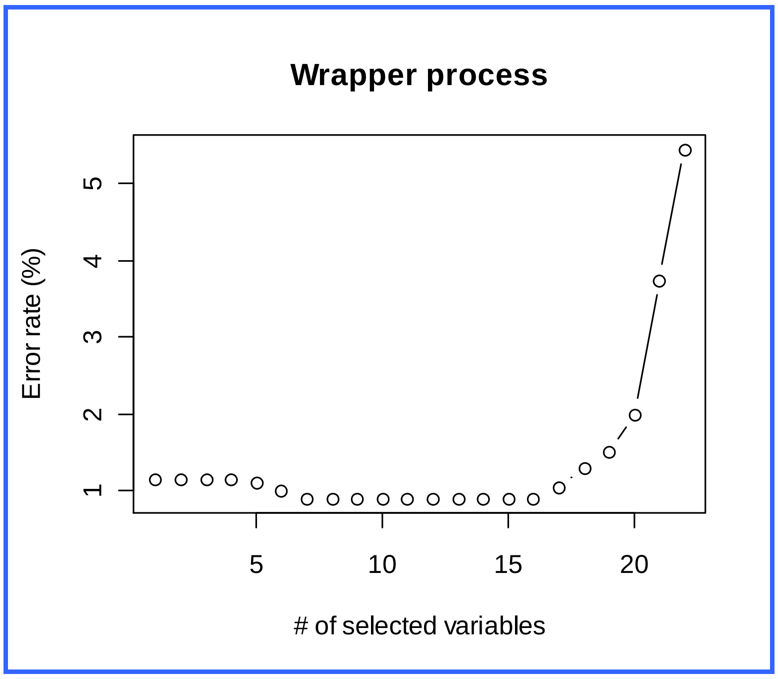
```
#programming the whole wrapper framework for feature subset selection
#forward selection process
#dataset is the available data (train set only, we do not use the test set
here)
#we assume that the last column is the class attribute
#K is the number of fold in the cross-validation
wrapper <- function(dataset,K=10){
 #number of predictive attribute
 P <- ncol(dataset)-1
 #predictive attributes
 predictives <- dataset[1:(ncol(dataset)-1)]
 #classe attribute
 cur.dataset <- dataset[ncol(dataset)]
 #current formula
 cur.formula <- paste(colnames(dataset)[P+1],"~")
 #ordered index of the selected variable
 output <- c()
 #error rate at each step
 error <- c()
 #the whole process
 for (p in 1:P){
  #error rate for this step
  cur.error <- sapply(predictives,search_wrapper,cur.dataset,K)
  #getting the best error rate
  id.min <- which.min(cur.error)
  selected.name <- colnames(predictives)[id.min]
  #adding the id. of the column into the selection
  output[p] <- which.min(match(colnames(dataset),selected.name))
  #add the attribute to the current dataset
  cur.dataset <- cbind(cur.dataset,predictives[id.min])
  #removing the attribute to the predictive attributes
  predictives <- predictives[-id.min]
  #filling out the vector of errors
  error[p] <- min(cur.error)
 }
 #returning the error rate and the index of variables
 return (list(error=error,output=output))
}
```

We launch this function on our dataset with the following program.

```
#launching the wrapper process
mushroom.wrapper <- wrapper(mushroom.train)

#plotting the error rate according the number of selected variables
plot(mushroom.wrapper$error,type="b",main="Wrapper   process",ylab="Error   rate
(%)",xlab="# of selected variables")
```

We get the error rate curve according the cardinal of the selected subset at each step of the process. It is very similar to the curve computed with SIPINA. It is not really surprising.



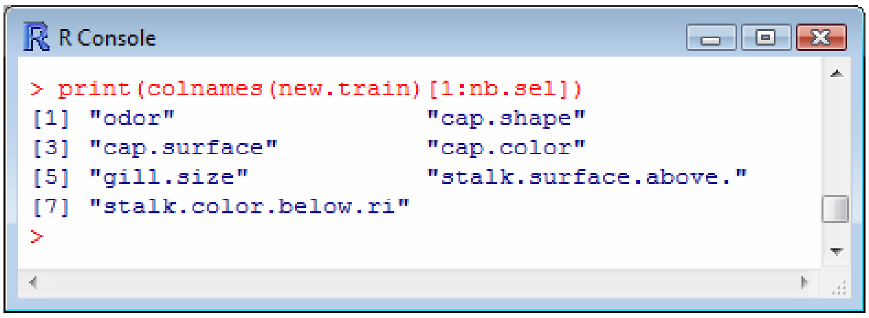
We can now select the subset related to the "optimal" solution.

```
#getting the number of selected variable
nb.sel <- which.min(mushroom.wrapper$error)

#new dataset -- train and test
new.train <-
mushroom.train[,c(mushroom.wrapper$output[1:nb.sel],ncol(mushroom.train))]
new.test <-
mushroom.test[,c(mushroom.wrapper$output[1:nb.sel],ncol(mushroom.test))]

#printing the name of the selected variables
print("Selected variables : ")
print(colnames(new.train)[1:nb.sel])
```

The subset with 7 descriptors seems the best.



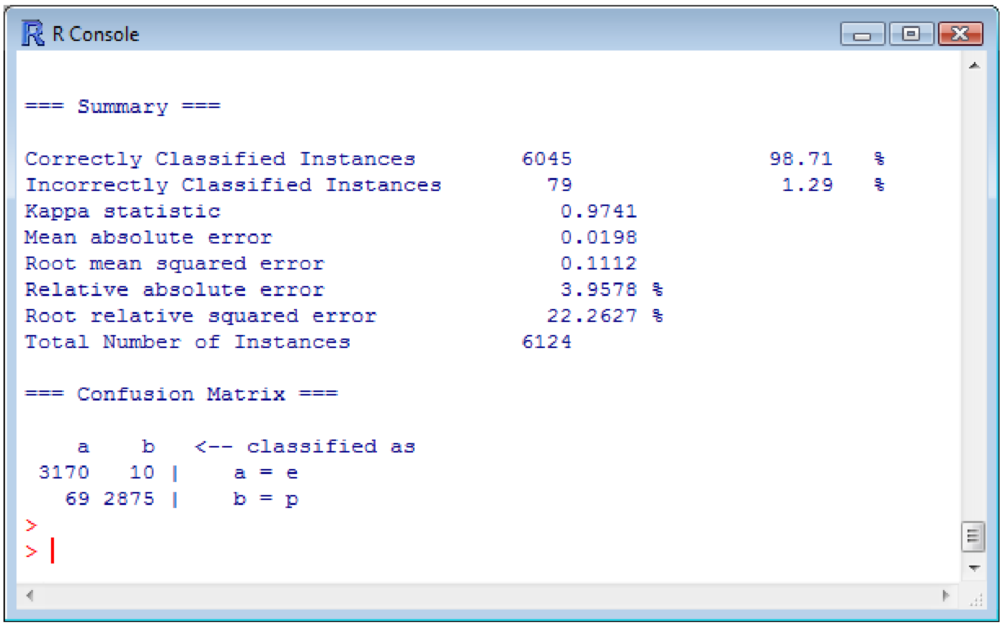
```
> print(colnames(new.train)[1:nb.sel])
[1] "odor"                 "cap.shape"
[3] "cap.surface"          "cap.color"
[5] "gill.size"            "stalk.surface.above."
[7] "stalk.color.below.ri"
>
```

4.4 Assessment of the selected subset of descriptors

Last, we compute the classifier on the learning set and we evaluate its performance on the test set. We use the samples with the selected descriptors now (the 7 descriptors detailed above).

```
#training and test on the selected variables
new.model <- NB(classe ~., data = new.train)
new.eval <- evaluate_Weka_classifier(new.model,newdata=new.test)
print(new.eval)
```

The test error rate is 1.29%. Again, the simplified model is definitely better than the full model including all the descriptors.



```
=== Summary ===

Correctly Classified Instances        6045               98.71   %
Incorrectly Classified Instances        79                1.29   %
Kappa statistic                          0.9741
Mean absolute error                      0.0198
Root mean squared error                  0.1112
Relative absolute error                  3.9578 %
Root relative squared error             22.2627 %
Total Number of Instances             6124

=== Confusion Matrix ===

    a    b   <-- classified as
 3170   10 |    a = e
   69 2875 |    b = p
>
> |
```

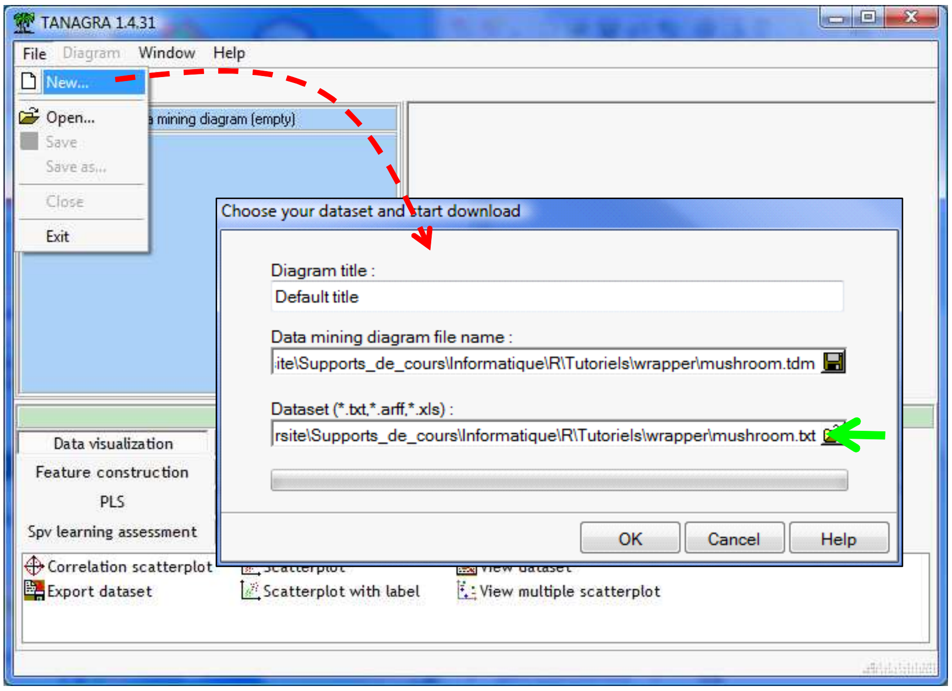
5 FILTER strategy with TANAGRA

The WRAPPER approach is not available in TANAGRA. But, it provides several FILTER methods for the feature selection process. We use the FCBF method here. It is really efficient and can handle a very large dataset (number of descriptors) quickly (Yu and Liu, 2003; http://www.public.asu.edu/~huanliu/FCBF/FCBFsoftware.html).

It operates before the learning process. There is no assurance that the selected variables are the most relevant for the naive bayes classifier. The interest of this tutorial is precisely to try to assess the efficiency of this approach in relation to the WRAPPER method which optimizes explicitly a performance criterion.

5.1 Importing the dataset

After we launch TANAGRA, we create a new diagram. We click on the FILE / NEW menu. We select the MUSHROOM.TXT data file: 24 columns and 8124 instances are loaded.



Choose your dataset and start download

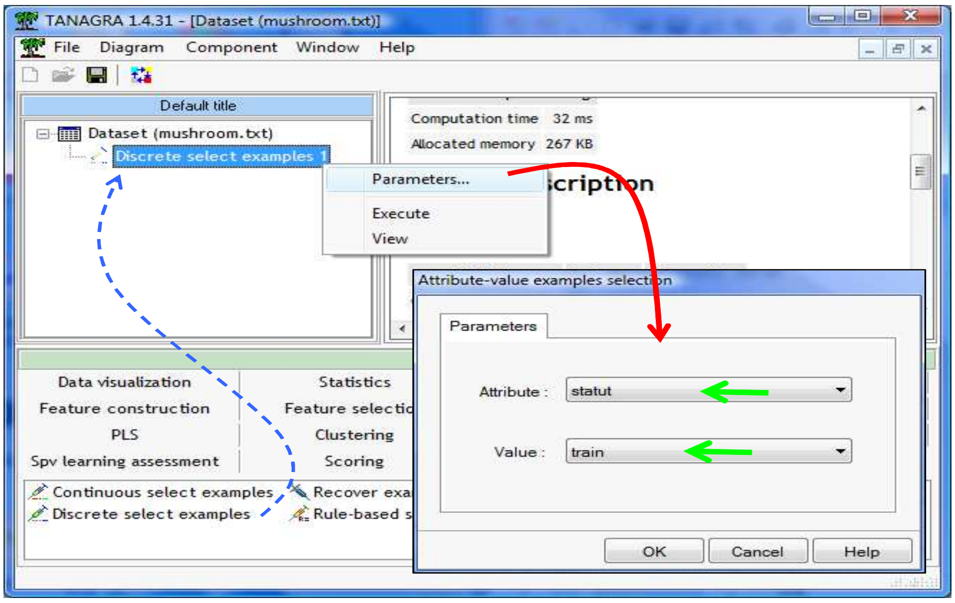
Diagram title : Default title

Data mining diagram file name : ite\Supports_de_cours\Informatique\R\Tutoriels\wrapper\mushroom.tdm

Dataset (*.txt,*.arff,*.xls) : rsite\Supports_de_cours\Informatique\R\Tutoriels\wrapper\mushroom.txt

5.2 Partitioning the dataset

We use the STATUT column in order to subdivide the dataset into a train and test samples. We add the DISCRETE SELECT EXAMPLES component (INSTANCE SELECTION) into the diagram. We click on the PARAMETERS contextual menu and we set the following parameters.



Computation time 32 ms
Allocated memory 267 KB

Attribute-value examples selection

Parameters

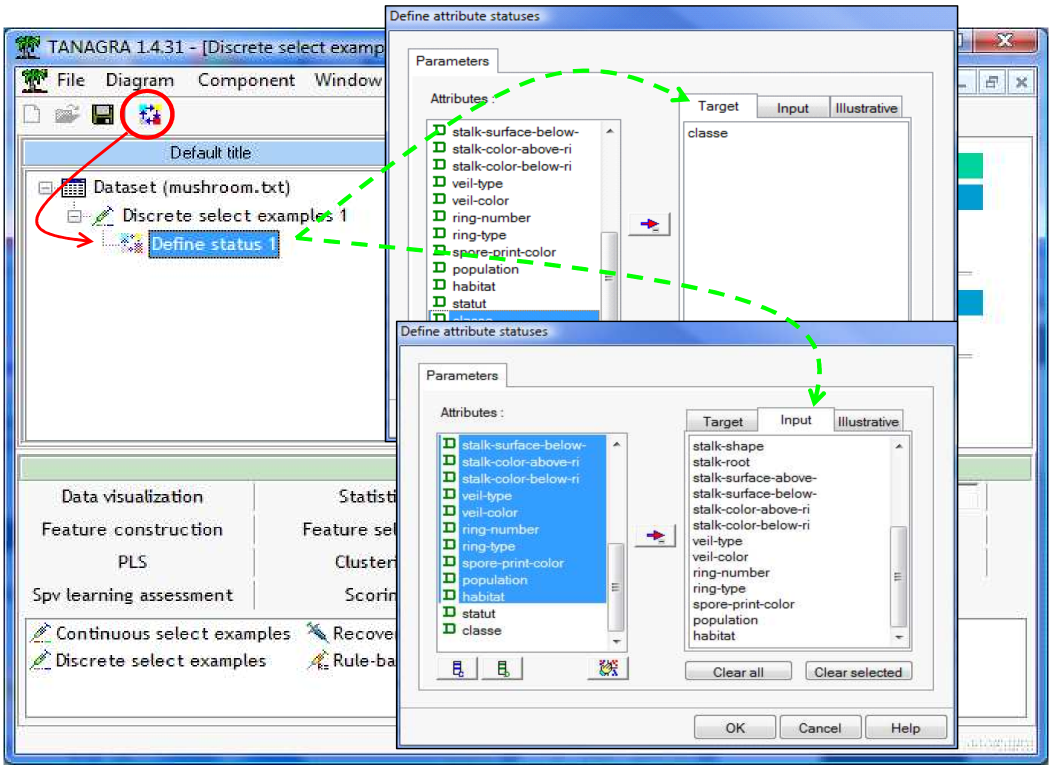
Attribute : statut

Value : train

Then we click on the VIEW menu. There are 2000 instances into the learning set.

5.3 Variable selection with the FCBF method

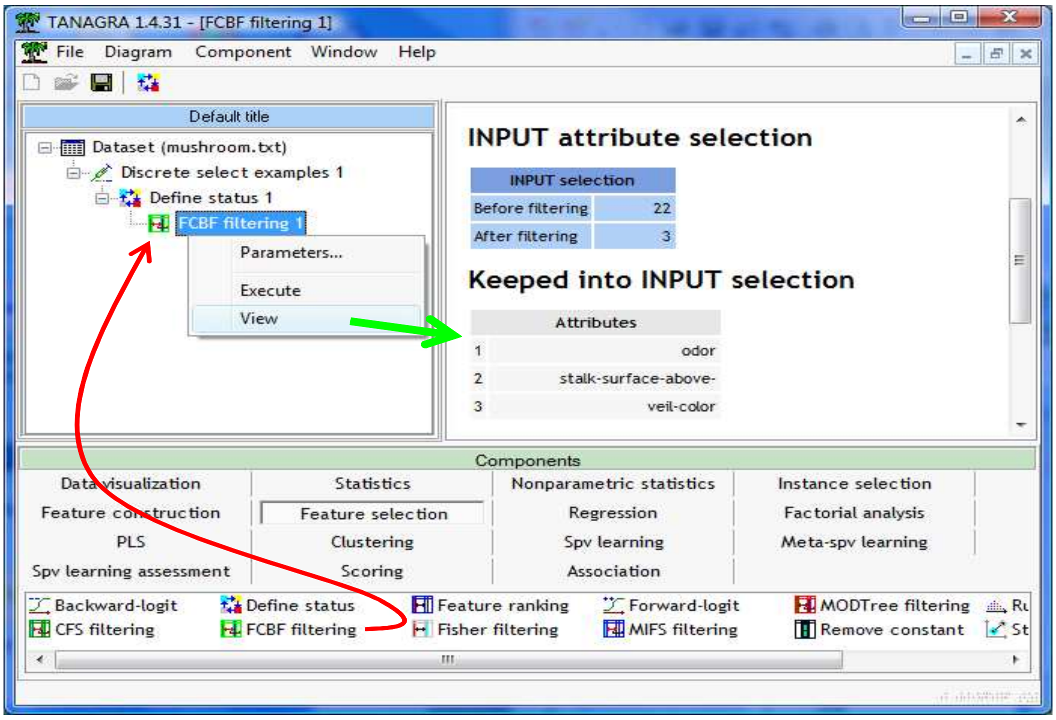
First, we must set the target attribute and the candidate input variables. We add the DEFINE STATUS component using the shortcut into the toolbar. The STATUT column is not used here.



Define attribute statuses — Target: classe

Define attribute statuses — Input: stalk-shape, stalk-root, stalk-surface-above-, stalk-surface-below-, stalk-color-above-ri, stalk-color-below-ri, veil-type, veil-color, ring-number, ring-type, spore-print-color, population, habitat

We can now add the FBCF component (FEATURE SELECTION). It aims to detect the most relevant and not redundant input variables for the prediction of the target attribute (CLASSE).



INPUT attribute selection

INPUT selection	
Before filtering	22
After filtering	3

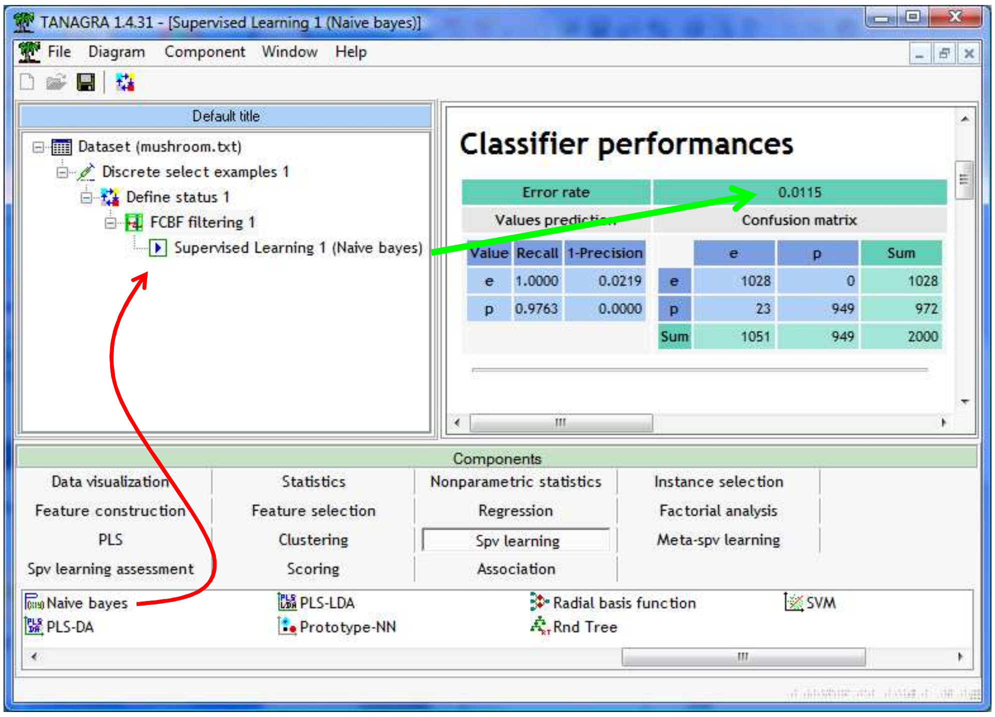
Keeped into INPUT selection

	Attributes
1	odor
2	stalk-surface-above-
3	veil-color

Only 3 descriptors are selected: ODOR, STALK SURFACE ABOVE and VEIL COLOR. These variables are automatically set as input variables at the output of the FBCF component. CLASSE remains the target attribute. We can insert any supervised learning method after the FBCF filter component.

5.4 Learning on the selected variables

We add the NAÏVE BAYES component (SPV LEARNING). We click on the VIEW menu. We obtain, among others, the resubstitution error rate.



Dataset (mushroom.txt)
Discrete select examples 1
Define status 1
FCBF filtering 1
Supervised Learning 1 (Naive bayes)

Classifier performances

Error rate	0.0115

Values prediction

Value	Recall	1-Precision
e	1.0000	0.0219
p	0.9763	0.0000

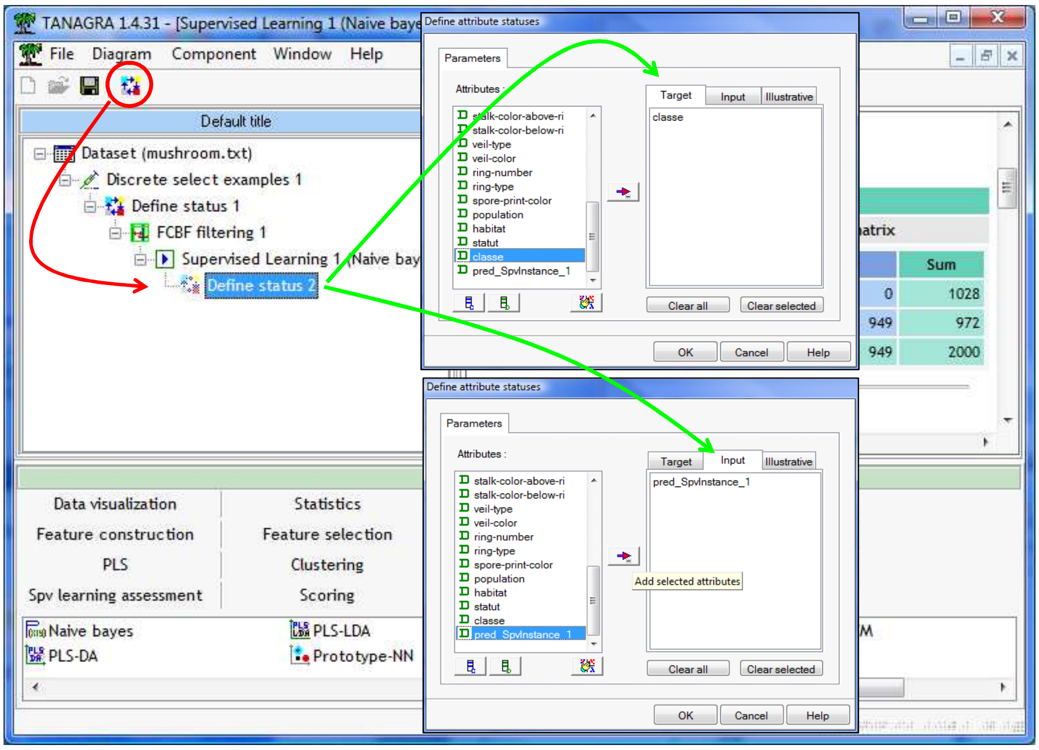
Confusion matrix

	e	p	Sum
e	1028	0	1028
p	23	949	972
Sum	1051	949	2000

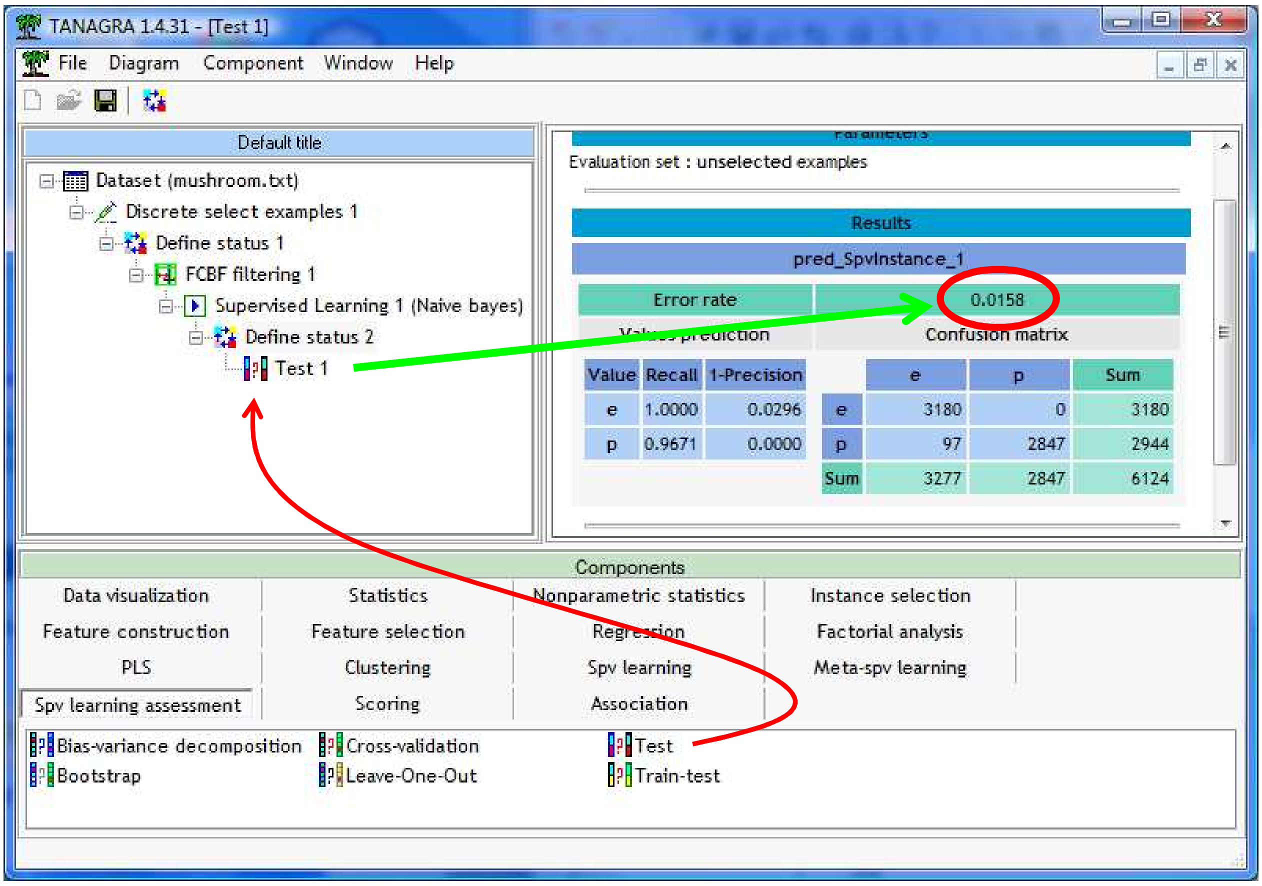
5.5 Assessing on the test set

In order to compute an honest estimation of the generalization error rate, we use the test set. To do that, we must specify the target attribute (CLASSE = TARGET) and the prediction of the classifier (PRED_SPV_INSTANCE_1 = INPUT) using the DEFINE STATUS component.

Note that the prediction is computed both on the learning set (the selected instances using the DISCRETE SELECT EXAMPLES component; see section 5.2) and the test set (the unselected instances).



Last, we add the TEST component (SPV LEARNING ASSESSMENT) which creates the confusion matrix and computes the error rate. By default, it operates on the unselected instances i.e. on the test set. This is that we wanted.



Parameters

Evaluation set : unselected examples

Results

pred_SpvInstance_1

Error rate	0.0158

Values prediction

Value	Recall	1-Precision
e	1.0000	0.0296
p	0.9671	0.0000

Confusion matrix

	e	p	Sum
e	3180	0	3180
p	97	2847	2944
Sum	3277	2847	6124

The test error rate is 1.58%. We can compare this to the 1.29% obtained with the WRAPPER approach with R, and the 1.24% obtained with SIPINA. The error rates are directly comparable here because we use the same test set for all the tools. We note that, according the generalization error rate, the deviations between the various approaches (FILTER vs. WRAPPER) are not really significant on our dataset.

6 Conclusion

The first goal of the tutorial is to show how to implement the WRAPPER feature selection approach under SIPINA and R. WRAPPER is supposed to be the best selection strategy because it uses explicitly a performance criterion to determine the "optimal" subset of variables for the prediction.

The second goal of this tutorial is to compare the efficiency of the WRAPPER approach with a FILTER strategy for selecting the relevant descriptors in the context of supervised learning. We use the FBCF method which is very fast and well suited to the datasets with a large number of descriptors. On our dataset, we note that, in spite that FBCF does not take into account the characteristics of the subsequent learning method, it produces a classifier which is very competitive in relation to the one computed after a WRAPPER feature subset selection.