1 Topic

Understanding the naive bayes classifier for continuous predictors.

The naive bayes classifier is a very popular approach even if it is (apparently) based on an unrealistic assumption: the distributions of the predictors are mutually independent conditionally to the values of the target attribute. The main reason of this popularity is that the method proved to be as accurate as the other well-known approaches such as linear discriminant analysis or logistic regression on the majority of the real dataset (http://en.wikipedia.org/wiki/Naive_Bayes_classifier).

But an obstacle to the utilization of the naive bayes classifier remains when we deal with a real problem. It seems that we cannot provide an explicit model for its deployment. The proposed representation by the PMML standard (http://www.dmg.org/v4-0-1/NaiveBayes.html) for instance is particularly unattractive. The interpretation of the model, especially the detection of the influence of each descriptor on the prediction of the classes is impossible.

This assertion is not entirely true. We have showed in a previous tutorial that we can extract an explicit model from the naive bayes classifier in the case of discrete predictors. We obtain a linear combination of the binarized predictors http://data-mining-tutorials.blogspot.com/2010/07/naive-bayes-classifier-for-discrete.html). In this document, we show that the same mechanism can be implemented for the continuous descriptors. We use the standard Gaussian assumption for the conditional distribution of the descriptors. According to the heteroscedastic assumption or the homoscedastic assumption, we can provide a quadratic model or a linear model. This last one is especially interesting because we obtain a model that we can directly compare to the other linear classifiers (the sign and the values of the coefficients of the linear combination).

This tutorial is organized as follows. In the next section, we describe the approach. In the section 3, we show how to implement the method with **Tanagra 1.4.37** (and later). We compare the results to those of the other linear methods. In the section 4, we compare the results provided by various data mining tools. We note that none of them proposes an explicit model that could be easy to deploy. They give only the estimated parameters of the conditional Gaussian distribution (mean and standard deviation). Last, in the section 5, we show the interest of the naive bayes classifier over the other linear methods when we handle a large dataset (the "mutant" dataset – 16592 instances and 5408 predictors). The computation time and the memory occupancy are clearly advantageous.

2 The naïve bayes classifier

Let $\aleph = (X_1, \ldots, X_J)$ the set of continuous descriptors. Y is the class attribute (with K ≥ 2 values). Into the supervised learning framework, when we want to classify an instance, we use the maximum a posteriori probability (MAP) rule i.e.

$$\hat{y}(\omega) = y_{k^*} \Leftrightarrow y_{k^*} = \arg\max_k P[Y = y_k / \aleph(\omega)]$$

The decision is based on an estimation of the conditional probability P(Y/X). Using Bayes' theorem

$$P[Y = y_k / \aleph(\omega)] = \frac{P(Y = y_k) \times P[\aleph(\omega) / Y = y_k]}{P[\aleph(\omega)]}$$

Because the goal is to detect the maximum according to the class y_k, the denominator of the expression can be removed without a modification of the classification mechanism.

$$\hat{y}(\omega) = y_{k*} \Leftrightarrow y_{k*} = \arg\max_k P(Y = y_k) \times P[\aleph(\omega)/Y = y_k]$$

The probability $P(Y = y_k)$ is easy to estimate from the dataset. We can use the relative frequency; or more sophisticated estimator such as the m-probability estimate in order to smooth the estimation on small dataset. If n_k is the number of instances for the value y_k of the target attribute Y, we have

$$\hat{P}(Y = y_k) = p_k = \frac{n_k + \lambda}{n + \lambda \times K}$$

When λ = 0, we get the standard relative frequency. For $\lambda = 1$, we obtain the Laplace correction.

Finally, the main difficulty is to estimate the probability $P[\aleph(\omega)/Y = y_k]$. Some assumptions are introduced to make it possible.

2.1 Assumption n°1: conditional independence

In the naive bayes classifier context, we assert that **the features are independents conditionally to the classes**. In consequence,

$$P[\aleph(\omega)/Y = y_k] = \prod_{j=1}^{J} P[X_j(\omega)/Y = y_k]$$

The number of parameters to compute from the dataset is dramatically reduced. For each predictor, we must provide now an individual estimation of the probability $P[X_j / Y = y_k]$.

2.2 Assumption n°2: Gaussian distribution

2.2.1 Classification functions

The second usual assumption in the naïve bayes approach for continuous predictors is the Gaussian conditional distribution. For the descriptor Xj, we have

$$P[X_j / Y = y_k] = f_k(X_j) = \frac{1}{\sigma_{k,j}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_j - \mu_{k,j}}{\sigma_{k,j}}\right)^2}$$

$\mu_{k,j}$ is the mean of Xj into the group Y = yk ; $\sigma_{k,j}$ the conditional standard deviation. We consider here that the standard deviations are different according to the classes. This is the heteroscedasticity assumption. These parameters are estimated as follows.

$$\hat{\mu}_{k,j} = \frac{1}{n_k} \sum_{\omega: Y(\omega) = y_k} x_j(\omega)$$

$$\hat{\sigma}_{k,j} = \frac{1}{n_k - 1} \sum_{\omega: Y(\omega) = y_k} [x_j(\omega) - \hat{\mu}_{k,j}]^2$$

Of course, the Gaussian assumption is restrictive. But, rather than to extend the covered distributions (e.g. log-normal, Poisson, gamma, etc.), it is more judicious to try to determine in which contexts the Gaussian assumption remains reasonable.

The Gaussian assumption is efficient while the conditional distribution is symmetric and unimodale (http://en.wikipedia.org/wiki/Unimodality). Besides this framework, the Gaussian assumption is questionable. Especially when we have multimodal distributions which overlap.

In our example (Figure 1)[1], the two distributions overlap. The conditional averages are confounded. We can erroneously think that the positive and the negative instances are not discernible.

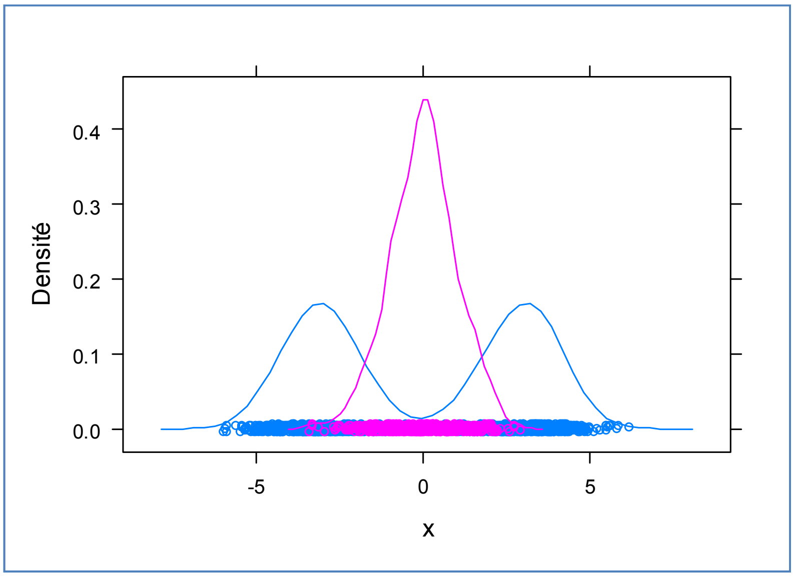


Figure 1 – Bimodal distribution of positive instances (blue), overlapping with the negative distribution

Two solutions are usually highlighted to overcome this drawback. First, we can use kernel density estimation. But it requires computing the estimation for each instance that we want to classify. In addition, we cannot provide an explicit model easy to deploy. Second, we can discretize the descriptors (http://data-mining-tutorials.blogspot.com/2010/05/discretization-of-continuous-features.html) as a pre-processing step. Then, we use the learning strategy for discrete predictors (http://data-mining-tutorials.blogspot.com/2010/07/naive-bayes-classifier-for-discrete.html). This approach is most probably the best one.

About the Gaussian assumption, we can write the classification functions as follows (by applying the logarithm):

$$d(y_k, \aleph) = \ln p_k + \sum_j \left\{ -\left[\frac{1}{2}\ln(2\pi) + \ln(\sigma_{k,j}) \right] - \frac{1}{2}\left(\frac{x_j - \mu_{k,j}}{\sigma_{k,j}} \right)^2 \right\}$$

The assignment rule remains the same, i.e.

[1] We use the following R source code to create this figure.

```
> x <- c(rnorm(1000,-3,1),rnorm(1000,0,1),rnorm(1000,+3,1))
> y <- c(rep(1,1000),rep(2,1000),rep(1,1000))
> library(lattice)
> densityplot(~ x, groups=factor(y))
```

$$\hat{y}(\omega) = y_{k^*} \Leftrightarrow y_{k^*} = \arg\max_k d[y_k, \aleph(\omega)]$$

Here again, all the terms which do not depend on k can be removed. We say "the classification function is proportional to"

$$d(y_k, \aleph) \propto \ln p_k + \sum_j \left\{ -\ln(\sigma_{k,j}) - \frac{1}{2}\left(\frac{x_j - \mu_{k,j}}{\sigma_{k,j}}\right)^2 \right\}$$

$$\propto \ln p_k + \sum_j \left\{ -\ln(\sigma_{k,j}) - \frac{1}{2 \times \sigma_{k,j}^2}\left(x_j^2 - 2 \times x_j \times \mu_{k,j} + \mu_{k,j}^2\right) \right\}$$

Last, we obtain finally

$$d(y_k, \aleph) \propto \ln p_k + \sum_j \left\{ -\frac{1}{2 \times \sigma_{k,j}^2} x_j^2 + \frac{\mu_{k,j}}{\sigma_{k,j}^2} x_j - \left(\frac{\mu_{k,j}^2}{2 \times \sigma_{k,j}^2} + \ln(\sigma_{k,j}) \right) \right\}$$

Equation 1 – Classification functions – Heteroscedasticity assumption

We get a quadratic classification function, but without the interaction terms between the features, according to the conditional independence which underlies the naive bayes classifier. Tanagra provides the coefficients of this function. It is not necessary to manipulate the conditional average and standard deviation when we want to deploy the model.

2.2.2 Numerical example: IRIS dataset (1)

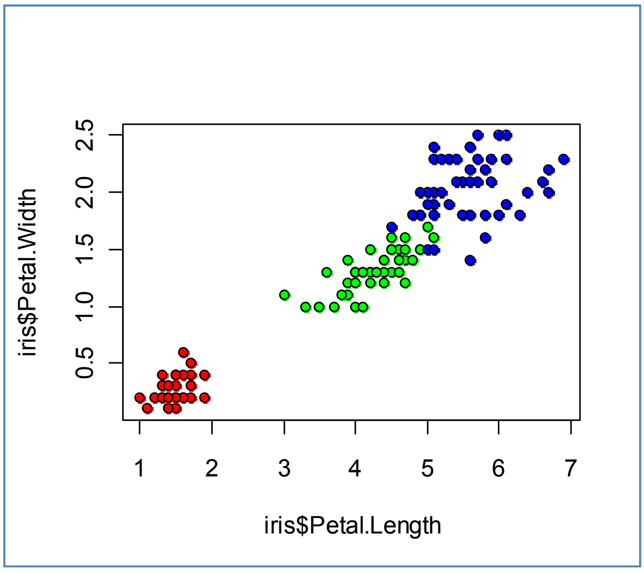


Figure 2 - IRIS de Fisher

We use the famous IRIS dataset (http://archive.ics.uci.edu/ml/datasets/Iris) in this section. We want to predict the IRIS type (setosa: red, versicolor: green, virginica: blue) from the two last descriptors: the length and the width of the petals (Figure 2).

a. Estimating the parameters of the model

We have 150 instances. The prior class probabilities can be estimated as follows (λ = 0),

$$p_k = \frac{50}{150} = 0.333,\ \forall k$$

We compute the conditional average and standard deviation for petal.length

	Données	
type	Moyenne de pet_length	Écartype de pet_length
Iris-setosa	1.4640	0.1735
Iris-versicolor	4.2600	0.4699
Iris-virginica	5.5520	0.5519

For petal.width

	Données	
type	Moyenne de pet_width	Écartype de pet_width
Iris-setosa	0.2440	0.1072
Iris-versicolor	1.3260	0.1978
Iris-virginica	2.0260	0.2747

We can form the coefficients of the classification function for "setosa".

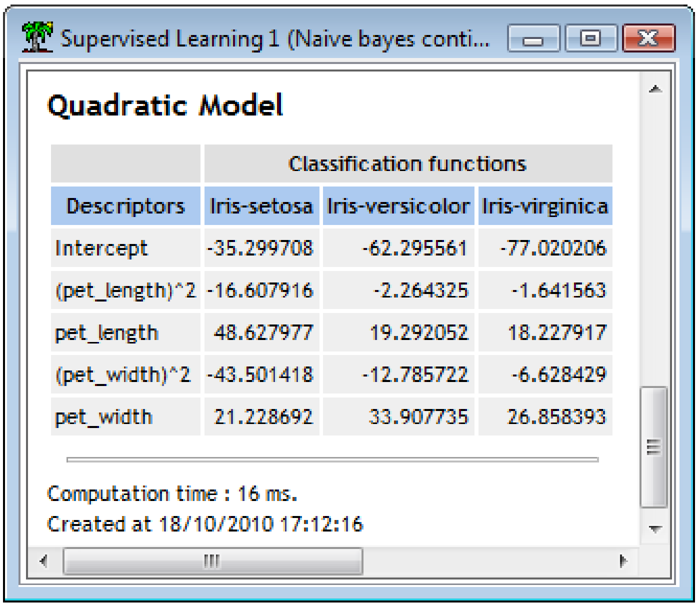
$$\begin{aligned} d(setosa,\aleph) &= \ln p_k + \sum_j \left\{ -\frac{1}{2\times\sigma^2_{k,j}} x_j^2 + \frac{\mu_{k,j}}{\sigma^2_{k,j}} x_j - \left(\frac{\mu^2_{k,j}}{2\times\sigma^2_{k,j}} + \ln(\sigma_{k,j}) \right) \right\} \\ &= -1.099 + \left[\left(-16.608 x_1^2 + 48.628 x_1 - 33.844 \right) + \left(-43.501 x_2^2 + 21.229 x_2 - 0.357 \right) \right] \\ &= -16.608 x_1^2 + 48.628 x_1 - 43.501 x_2^2 + 21.229 x_2 - 35.300 \end{aligned}$$

We perform the same calculations for the other classes.

$$d(versicolor,\aleph) = -2.264 x_1^2 + 19.292 x_1 - 12.786 x_2^2 + 33.908 x_2 - 62.296$$

$$d(virginica,\aleph) = -1.642 x_1^2 + 18.228 x_1 - 6.628 x_2^2 + 26.858 x_2 - 77.020$$

b. Tanagra output



	Classification functions		
Descriptors	Iris-setosa	Iris-versicolor	Iris-virginica
Intercept	-35.299708	-62.295561	-77.020206
(pet_length)^2	-16.607916	-2.264325	-1.641563
pet_length	48.627977	19.292052	18.227917
(pet_width)^2	-43.501418	-12.785722	-6.628429
pet_width	21.228692	33.907735	26.858393

The outputs of Tanagra are consistent to the calculations above. We get the coefficients for each Xj² and Xj. The constants are in the first row of the table.

We will describe below the utilization of Tanagra in the context of naive bayes classifier induction.

c. Classifying a new instance

For classifying an unseen instance, we apply the classification functions. We assign the instance to the class which maximizes d(Y, X).

Into the table below, we want to classify three instances. We note the consistency between the assigned class and the position of the point into the representation space (Figure 2).

N°	Coordinates (X1, X2)	d(setosa)	d(versicolor)	d(virginica)	Prediction
1	(1.5, 0.5)	**0.013**	-24.695	-41.600	setosa
2	(5.0, 1.5)	-273.393	**-0.350**	-1.546	versicolor
3	(5.0, 2.0)	-338.906	-5.771	**0.283**	virginica

Except for the point n°2, the decisions are unambiguous. We understand this phenomenon when we consider the figure. To assign an instance to a class is only difficult in the area where the versicolor and the virginica overlap. For the instance (X1 = 5, X2 = 1.6), we get [d(setosa) = -284.755 ; d(versicolor) = -0.923 ; d(virginica) = -0.915]. The decision "iris = virginica" is not obvious.

2.2.3 The particular case of binary problem

For the binary problem, Y = {+, -}, we get two classification functions. We can deduce a unique decision function as follows

$$\begin{aligned} d(\aleph) &= d(+,\aleph) - d(-,\aleph) \\ &= \delta + \sum_j \left\{ \alpha_j x_j^2 + \beta_j x_j + \gamma_j \right\} \end{aligned}$$

The assignment rule becomes

$$\text{If } d\left[\aleph(\omega)\right] > 0 \text{ Then } \hat{y}(\omega) = + \text{ Else } \hat{y}(\omega) = -$$

2.2.4 Assessing the relevance of predictive attributes

To assess the relevance of one attribute, it seems attractive to use a comparison of conditional distributions. If they are confounded, we can conclude that the attribute is irrelevant. It has no influence on the differentiation of the classification functions.

But the procedure is not easy. We must compare **simultaneously** the conditional average and the conditional standard deviation. More formally, according the classification functions outlined above (Équation 1), the null hypothesis (irrelevance of an attribute) is written as follows:

$$H_0 : \begin{cases} \dfrac{1}{\sigma_{k,j}^2} = \text{constant}, \forall k \\ \dfrac{\mu_{k,j}}{\sigma_{k,j}^2} = \text{constant}, \ \forall k \end{cases} \Leftrightarrow H_0 : \begin{cases} \sigma_{k,j}^2 = \text{constant}, \forall k \\ \mu_{k,j} = \text{constant}, \ \forall k \end{cases}$$

Currently, I confess I do not know which statistical test can answer this question.

2.3 Assumption n°3: Homoscedasticty – Linear classifier

2.3.1 Classification function

We can introduce an assumption in order to simplify again the model. We claim that the conditional standard deviations are the same whatever the classes. This is the homoscedasticity assumption. For the attribute Xj, we have

$$\sigma_{k,j} = \sigma_j \ , \ \forall k$$

We use the following estimation for the common standard deviation over the classes.

$$\hat{\sigma}_j^2 = \frac{1}{n-K}\sum_{k=1}^{K}(n_k - 1)\times\hat{\sigma}_{k,j}^2$$

Consequently, when we introduce this expression into the classification function, we get

$$d(y_k,\aleph) \propto \ln p_k + \sum_j \left\{ -\frac{1}{2\times\sigma_j^2}x_j^2 + \frac{\mu_{k,j}}{\sigma_j^2}x_j - \left(\frac{\mu_{k,j}^2}{2\times\sigma_j^2} + \ln(\sigma_j) \right) \right\}$$

Again, we can remove all the additive terms which do not depend to the class. The simplified classification function becomes

$$d(y_k,\aleph) \propto \ln p_k + \sum_j \left\{ \frac{\mu_{k,j}}{\sigma_j^2}x_j - \frac{\mu_{k,j}^2}{2\times\sigma_j^2} \right\}$$

Équation 2- Classification function – Homoscedasticity assumption

The squared terms have disappeared. We obtain a linear classifier. Except particular circumstances, the classifier is as accurate as the other linear classifiers.

2.3.2 Numerical example: the IRIS dataset (2)

a. Creating the predictive model

We treat again the IRIS dataset in the same context (section 2.3). We must compute now the within-class variance estimation. We have for X1 (petal length) and X2 (petal width):

$$\hat{\sigma}_1 = \sqrt{\frac{1}{150-3}\left[49\times 0.1735^2 + 49\times 0.4699^2 + 49\times 0.5519^2\right]} = 0.430$$

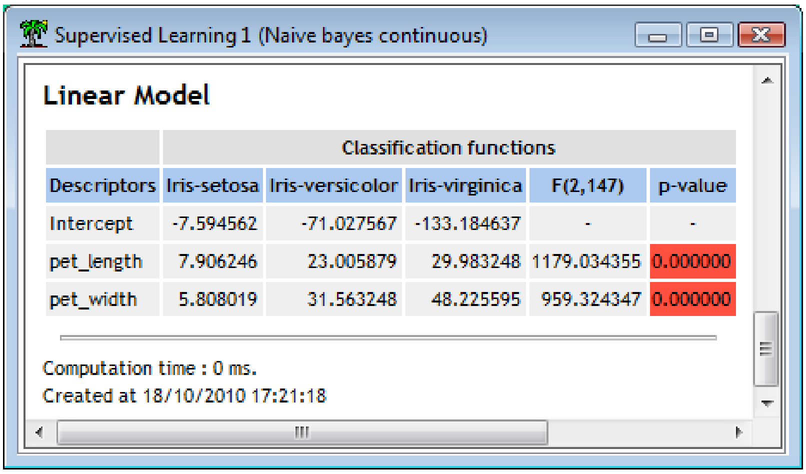
$$\hat{\sigma}_2 = \sqrt{\frac{1}{150-3}\left[49\times 0.1072^2 + 49\times 0.1978^2 + 49\times 0.2747^2\right]} = 0.205$$

The three classification functions are the follows:

$$\begin{aligned}
d(setosa,\aleph) &= \ln p_k + \sum_j \left\{ \frac{\mu_{k,j}}{\sigma_j^2}x_j - \frac{\mu_{k,j}^2}{2\times\sigma_j^2} \right\} \\
&= -1.099 + (7.906x_1 - 5.787) + (5.808x_2 - 0.709) \\
&= 7.906x_1 + 5.808x_2 - 7.595 \\
d(versicolor,\aleph) &= 23.006x_1 + 31.563x_2 - 71.028 \\
d(virginica,\aleph) &= 29.983x_1 + 48.226x_2 - 133.185
\end{aligned}$$

b. Tanagra output

Tanagra provides the coefficients of the classification function (Figure 3). It provides also an indication about the relevance of the predictive attributes (F statistic and the p-value). We describe the details of the calculations below.



Supervised Learning 1 (Naive bayes continuous)

Linear Model

	Classification functions				
Descriptors	Iris-setosa	Iris-versicolor	Iris-virginica	F(2,147)	p-value
Intercept	-7.594562	-71.027567	-133.184637	-	-
pet_length	7.906246	23.005879	29.983248	1179.034355	0.000000
pet_width	5.808019	31.563248	48.225595	959.324347	0.000000

Computation time : 0 ms.
Created at 18/10/2010 17:21:18

Figure 3 – Linear model – IRIS dataset

c. Classifying a new instance

We classify the same instances as above.

N°	Coordinates (X1, X2)	d(setosa)	d(versicolor)	d(virginica)	Prediction
1	(1.5, 0.5)	**7.169**	-20.737	-64.097	setosa
2	(5.0, 1.5)	40.649	**91.347**	89.070	versicolor
3	(5.0, 2.0)	4.553	107.128	**113.183**	virginica

The conclusions are the same. We observe that when we are near to the frontier (e.g. (X1 = 5, X2 = 1.6), the decision can be different: we predict versicolor [d(setosa) = 41.229 ; d(versicolor) = 94.503 ; d(virginica) = 93.893] rather than virginica (for the quadratic model).

2.3.3 Linear classifier for binary problem

When we deal with a binary problem, we can highlight a unique decision function.

$$d(\aleph) = a_0 + a_1 x_1 + a_2 x_2 + \cdots$$

The classification rule becomes

$$\text{If } d\left[\aleph(\omega)\right] > 0 \text{ Then } \hat{y}(\omega) = + \text{ Else } \hat{y}(\omega) = -$$

2.3.4 Assessing the relevance of the predictors

Contrary to the model under the heteroscedasticity assumption, it is possible to produce a simple test to assess the relevance of a variable. Indeed, from the classification function (Équation 2), a variable Xj does not contribute to discrimination if

$$H_0 : \mu_{k,j} = \text{constant}, \forall k$$

This is the null hypothesis of the one-way ANOVA. The test statistic F is written as follows

$$F_j = \frac{\dfrac{\sum_k n_k \left(\hat{\mu}_{k,j} - \hat{\mu}_j\right)^2}{K-1}}{\dfrac{\sum_k (n_k - 1)\hat{\sigma}^2_{k,j}}{n-K}}$$

Under the null hypothesis, it follows a Fisher distribution with (K-1, n-K) degrees of freedom. We reject the null hypothesis at the significance level α (i.e. we decide that the variable is relevant) if $F_j \geq F_{1-\alpha}(K-1, n-K)$; $F_{1-\alpha}$ is the quantile of the Fisher distribution at the (1-α) level. Another way to make the decision is to compare the p-value to the significance level of the test. We conclude that the variable is relevant if the p-value is lower than the chosen significance level.

For the IRIS dataset, we observe that all the descriptors are relevant. It is not surprising if we refer to the relative position of the groups into the representation space (Figure 2). The test statistic F is equal to 1179.03 (resp. 959.32) for the petal length attribute (resp. petal width) (Figure 3). We found the same values if we perform a standard analysis of variance (Figure 4).

Note: This test assesses individually the descriptors. It does not take into account the redundancy between them. So, if we replicate the same variable ten times, all are relevant according to the Fisher test.

One-way ANOVA 1

Results

Attribute_Y	Attribute_X	Value	Examples	Average	Std-dev
pet_length	type	Iris-setosa	50	1.4640	0.1735
		Iris-versicolor	50	4.2600	0.4699
		Iris-virginica	50	5.5520	0.5519
		All	150	3.7587	1.7644
pet_width	type	Iris-setosa	50	0.2440	0.1072
		Iris-versicolor	50	1.3260	0.1978
		Iris-virginica	50	2.0260	0.2747
		All	150	1.1987	0.7632

Statistical test – Variance decomposition

Attribute_Y	Source	Sum of square	d.f.
pet_length	BSS	436.6437	2
	WSS	27.2200	147
	TSS	463.8637	149
pet_width	BSS	80.6041	2
	WSS	6.1756	147
	TSS	86.7797	149

Significance level

Attribute_Y	Statistics	Value	Proba
pet_length	Fisher's F	1179.034355	0.000000
pet_width	Fisher's F	959.324347	0.000000

Figure 4 – Analysis of variance – Iris dataset

3 The naive bayes classifier under Tanagra

3.1 Dataset

To describe the approach, we use the famous "breast-cancer" dataset (breast.txt; http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29). This dataset is particularly interesting because the Gaussian assumption is questionable. Of course, the predictors seem continuous. But it appears above all as an integer indicating the interval belonging of the instance after a discretization process. We have not the original values of the variables. By representing the conditional distribution (Figure 5), we observe that the homoscedasticity assumption is questionable also. We note however that the overlap between the conditional distributions is weak whatever the variable. This suggests that it is possible to obtain a good quality of discrimination. Let us see if the naive bayes classifier is nevertheless able to detect the frontier between the classes.

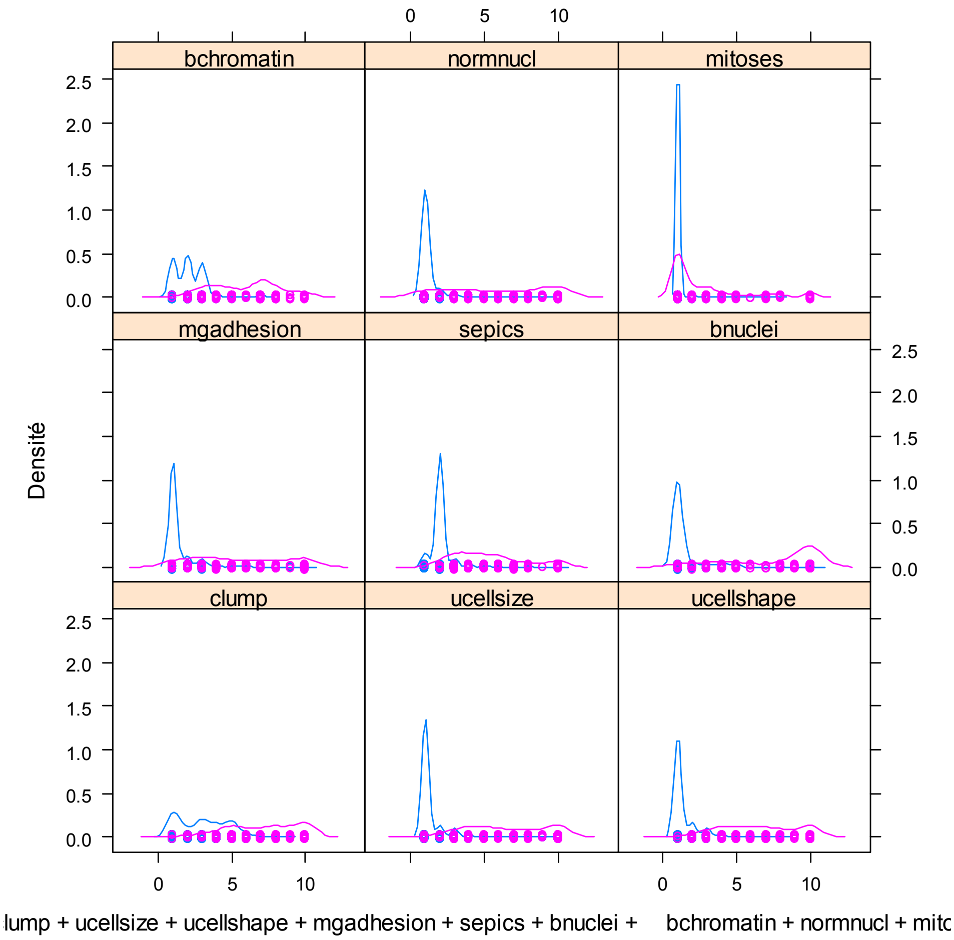
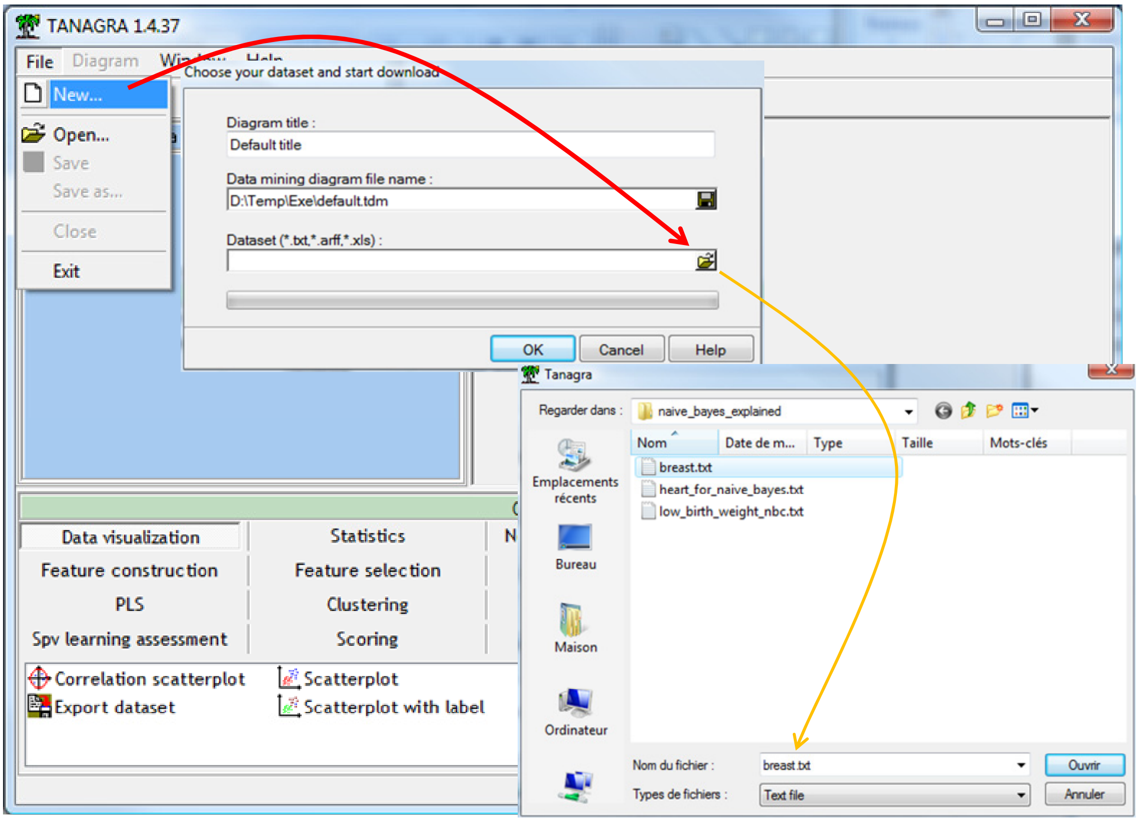


Figure 5 - "Breast" dataset – Conditional distributions

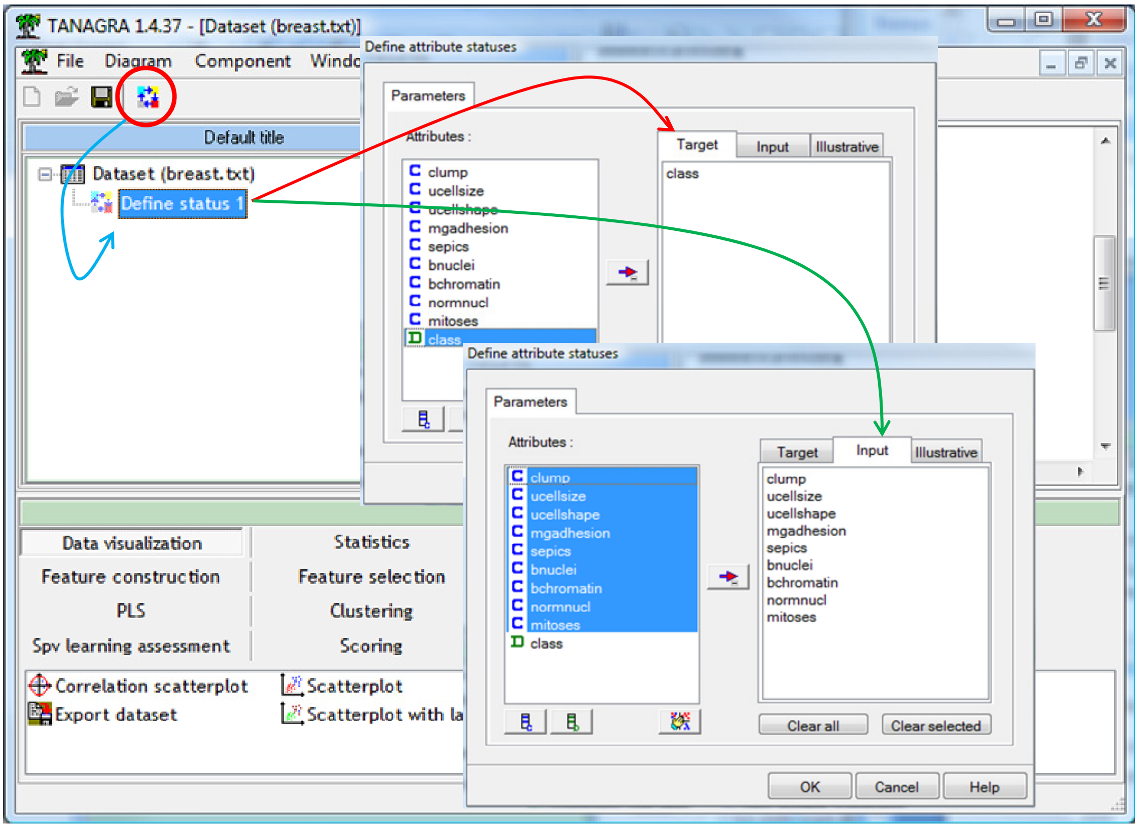
In what follows, we will build the naive bayes classifier on the "breast" dataset. We will put a special emphasis on reading the results. About the model under the homoscedasticity assumption, since it is linear, we compare the coefficients of the hyperplane separator with those provided by other linear techniques. We will see that they are consistent.

3.2 The naïve bayes classifier under Tanagra

We click on the FILE / NEW menu in order to create a new diagram. We select the "breast.txt" data file (tab separator). A new diagram is created and the dataset is automatically loaded. We have 699 instances and 10 variables (1 class attribute and 9 predictors).



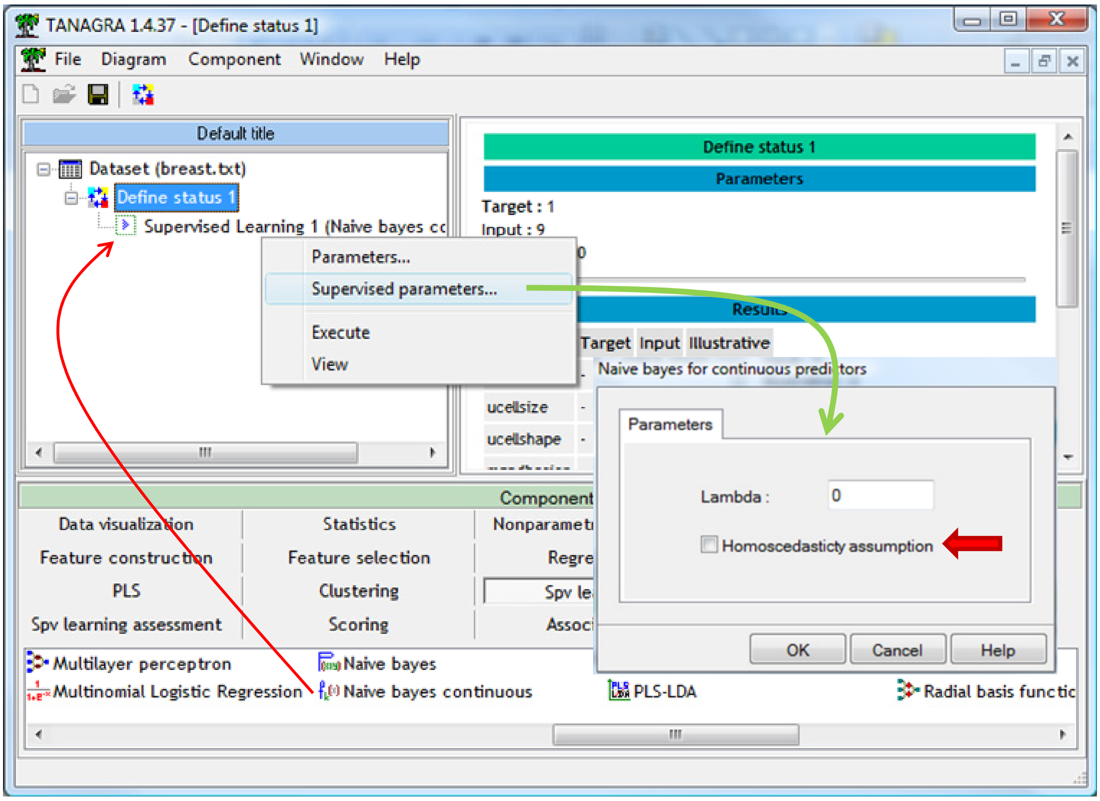
We insert the DEFINE STATUS component to specify the status of the variables: CLASS is the TARGET attribute; the others (CLUMP…MITOSES) are the INPUT ones.



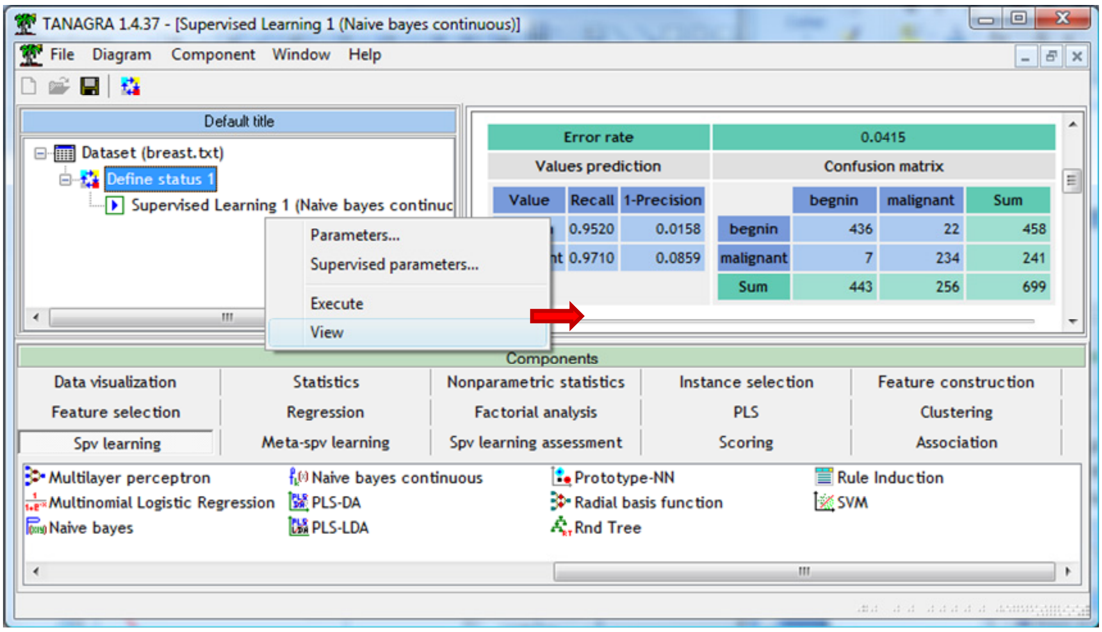
We click on the contextual VIEW menu to validate the operation.

3.2.1 Learning the quadratic model

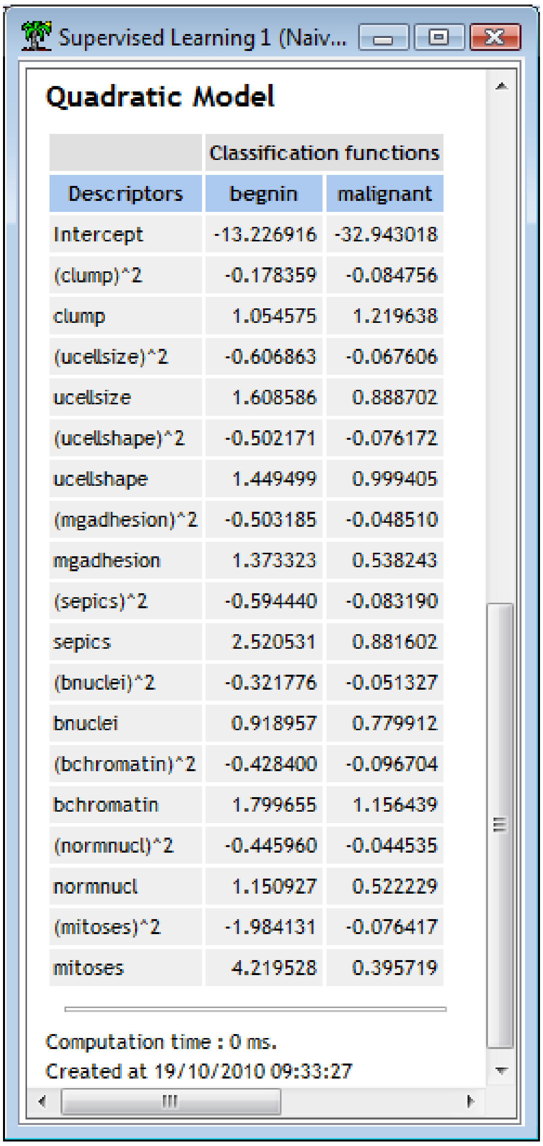
We add the NAIVE BAYES CONTINUOUS component (SPV LEARNING tab) into the diagram. We click on the SUPERVISED PARAMETERS contextual menu to set the parameters. In a first step, we want to learn a quadratic model i.e. under the heteroscedasticity assumption.



We activate the VIEW menu. We obtain the confusion matrix and the resubstitution error rate (4.15%). We know that this value is often optimistic because we use the same dataset for the training and the testing of the model.



Below, we have the coefficients of the classification functions.



Quadratic Model

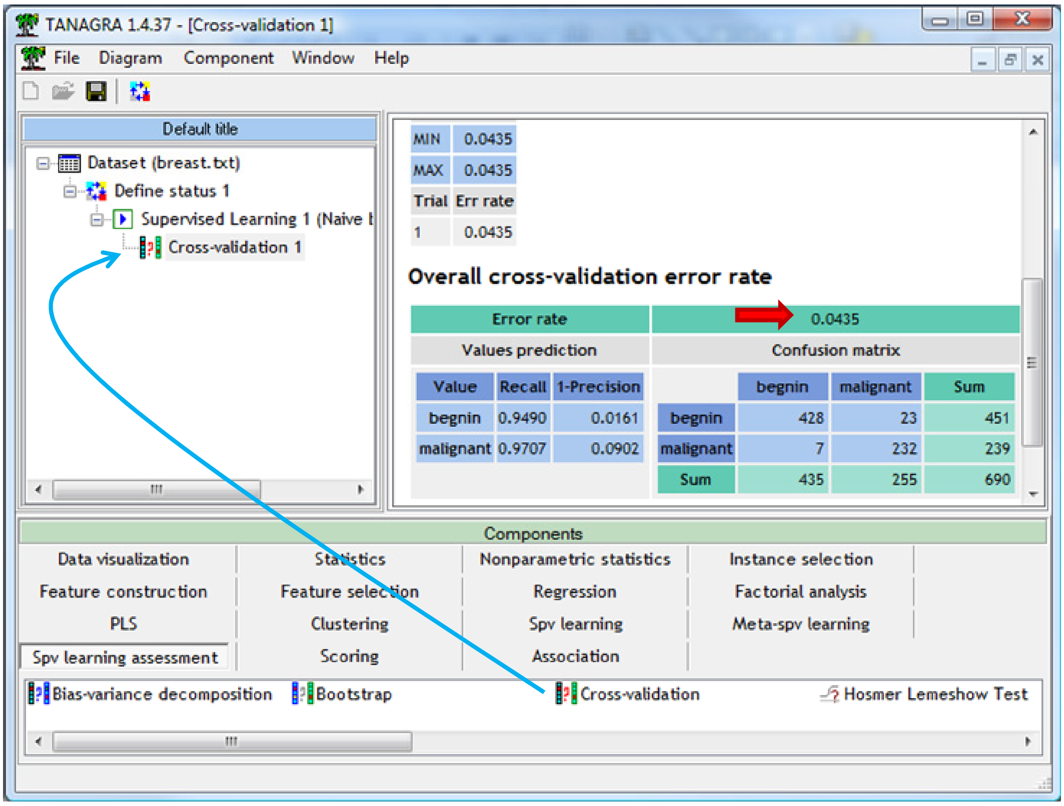
	Classification functions	
Descriptors	begnin	malignant
Intercept	-13.226916	-32.943018
(clump)^2	-0.178359	-0.084756
clump	1.054575	1.219638
(ucellsize)^2	-0.606863	-0.067606
ucellsize	1.608586	0.888702
(ucellshape)^2	-0.502171	-0.076172
ucellshape	1.449499	0.999405
(mgadhesion)^2	-0.503185	-0.048510
mgadhesion	1.373323	0.538243
(sepics)^2	-0.594440	-0.083190
sepics	2.520531	0.881602
(bnuclei)^2	-0.321776	-0.051327
bnuclei	0.918957	0.779912
(bchromatin)^2	-0.428400	-0.096704
bchromatin	1.799655	1.156439
(normnucl)^2	-0.445960	-0.044535
normnucl	1.150927	0.522229
(mitoses)^2	-1.984131	-0.076417
mitoses	4.219528	0.395719

Computation time : 0 ms.
Created at 19/10/2010 09:33:27

For the « begnin » class for instance, we have:

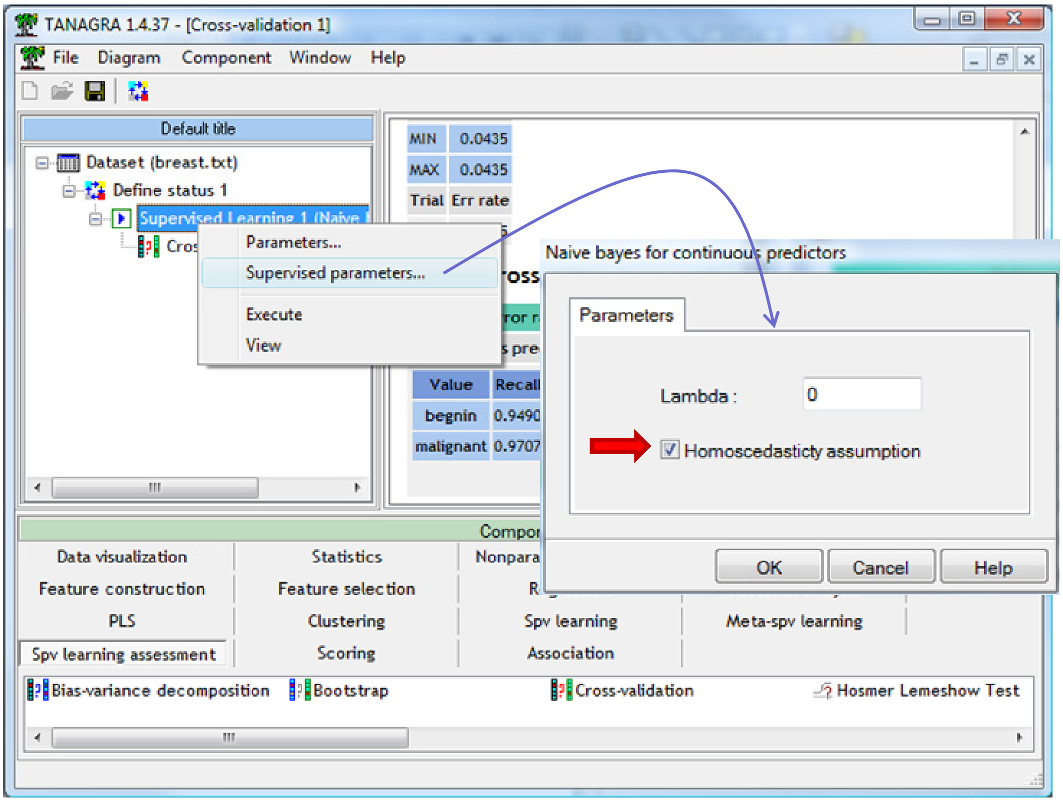
$$d(begnin,\aleph) = -13.226916 - 0.178359 \times (clump)^2 + 1.054575 \times clump + \cdots$$

To obtain a less biased estimation of the generalization performance, we use the cross-validation (CROSS-VALIDATION, SPV LEARNING ASSESSMENT tab). We click on VIEW. Tanagra shows that the "true" generalization error rate would be 4.35%.



3.2.2 Linear naïve bayes classifier – Homoscedasticity assumption

We want to implement the linear classifier now. We click on the menu SUPERVISED PARAMETERS of the NAÏVE BAYES CLASSIFIER component. We select the "Homoscedasticity assumption" option.



We click on the VIEW menu, we obtain the following model.

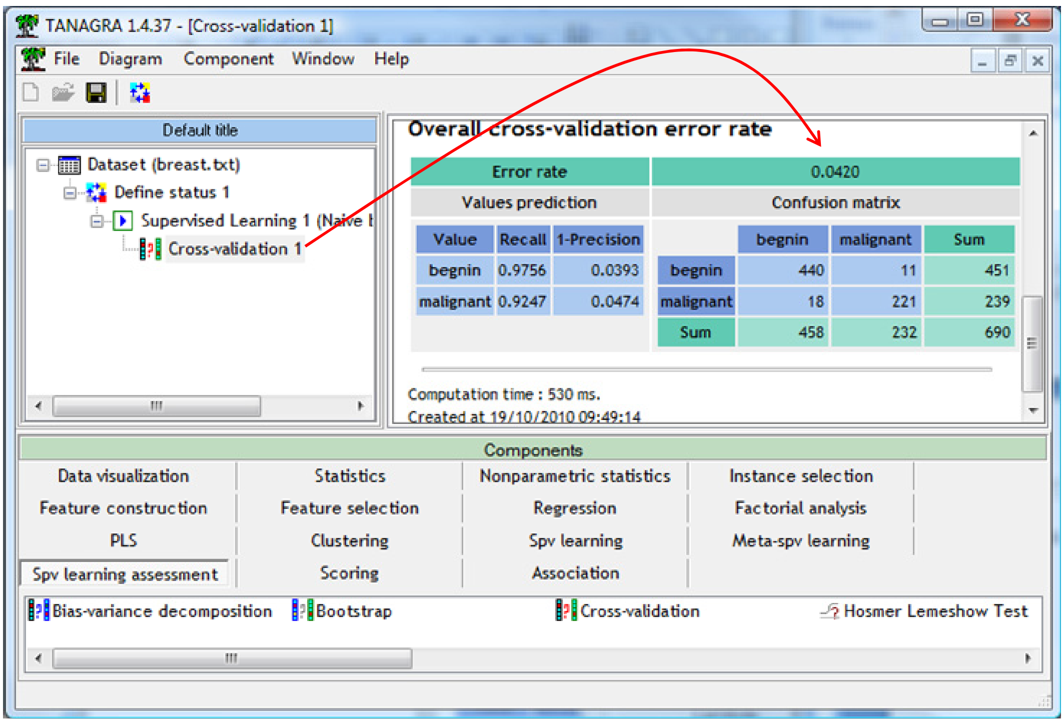
Linear Model

	Classification functions			
Descriptors	begnin	malignant	F(1,697)	p-value
Intercept	-4.787695	-49.901570	-	-
clump	0.764032	1.859474	733.206978	0.000000
ucellsize	0.429352	2.129259	1408.527213	0.000000
ucellshape	0.495435	2.251986	1419.305530	0.000000
mgadhesion	0.324866	1.320701	657.793700	0.000000
sepics	0.808864	2.021601	608.719555	0.000000
bnuclei	0.326527	1.737311	1374.423037	0.000000
bchromatin	0.825128	2.348867	933.287297	0.000000
normnucl	0.280463	1.274319	717.628041	0.000000
mitoses	0.439712	1.070712	152.040239	0.000000

Computation time : 0 ms.
Created at 19/10/2010 09:41:45

The main difference in relation to the quadratic model is that the coefficients for the squared variable are removed, and we have an indication about the relevance of the attribute. As we have seen above, it is based on the F-test of the ANOVA. All the predictors seem relevant here. It is not really surprising if we refer to the conditional distribution described previously (Figure 5).

We perform again the cross validation by clicking on the VIEW menu of the component into the diagram. The estimated generalization error rate is 4.2%. It is very similar to the quadratic model.



3.2.3 Deducing the decision function for a binary problem

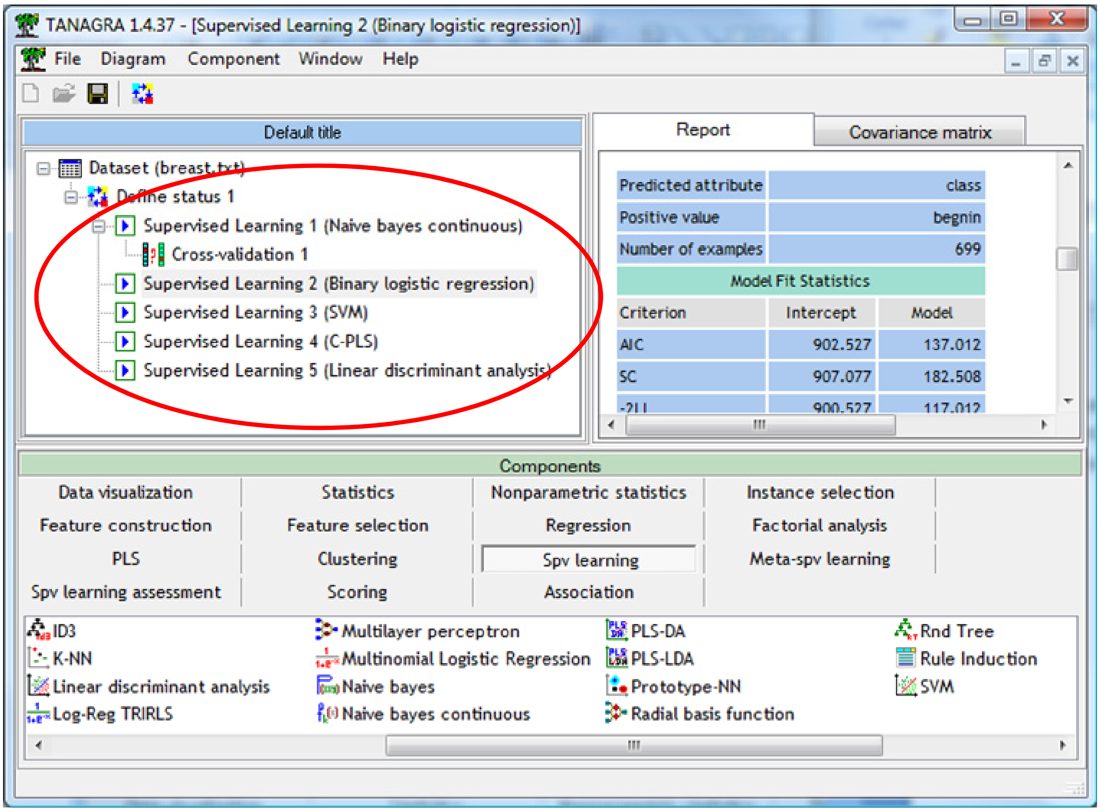
Because we deal with a binary problem, we can deduce a unique decision function from the two classification functions. We obtain the following coefficients:

Descriptors	d(X)
Intercept	45.1139
clump	-1.0954
ucellsize	-1.6999
ucellshape	-1.7566
mgadhesion	-0.9958
sepics	-1.2127
bnuclei	-1.4108
bchromatin	-1.5237
normnucl	-0.9939
mitoses	-0.6310

3.3 Comparison with the other linear approaches

3.3.1 Comparison of the coefficients of the decision function

Since other approaches can induce linear classifiers, let us see how they position themselves in relation to the naive bayes classifier (NBC). We have tried: a logistic regression (BINARY LOGISTIC REGRESSION), a linear support vector machine (SVM), the PLS (partial least squares) discriminant analysis (C-PLS) and the linear discriminant analysis (LDA). We defined the following diagram under Tanagra.



We obtain the following decision functions.

Descriptors	NBC (linear)	Logistic.Reg	SVM	C-PLS	LDA
Intercept	45.1139	9.6710	3.6357	0.6053	20.2485
clump	-1.0954	-0.5312	-0.1831	-0.0350	-0.8867
ucellsize	-1.6999	-0.0058	-0.0290	-0.0195	-0.6081
ucellshape	-1.7566	-0.3326	-0.1275	-0.0219	-0.4381
mgadhesion	-0.9958	-0.2403	-0.0590	-0.0133	-0.1749
sepics	-1.2127	-0.0694	-0.0777	-0.0103	-0.2153
bnuclei	-1.4108	-0.4001	-0.1675	-0.0363	-1.2181
bchromatin	-1.5237	-0.4107	-0.1610	-0.0263	-0.5589
normnucl	-0.9939	-0.1447	-0.0650	-0.0151	-0.4619
mitoses	-0.6310	-0.5507	-0.1437	0.0087	-0.0773

We observe that the results are really consistent, at least concerning the sign of the coefficients. Only one coefficient (mitoses for C-PLS) is inconsistent in relation to the others.

3.3.2 Generalization performance

The "breast" dataset is easy to learn. Not surprisingly, the cross-validation error rates are very similar whatever the method.

Method	Error rate (%)
NBC (quadratic)	4.35
NBC (linear)	4.20
Logistic regression	3.77
SVM (linear)	3.04
C-PLS	3.33
LDA	4.20

4 The naïve bayes classifier under the other data mining tools

4.1 Dataset

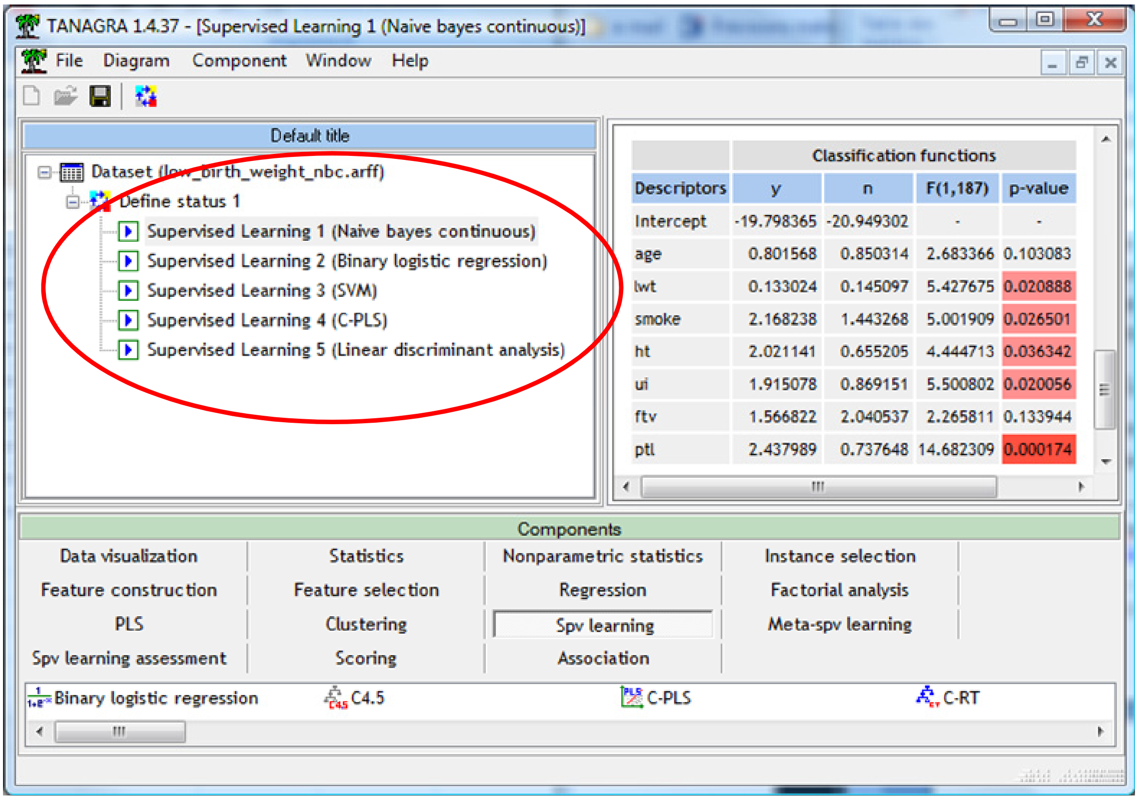
We use the « low_birth_weight_nbc.arff » data file in this section[2]. The aim is to explain the low birth weight of babies from the characteristics (weight, etc.) and behavior (smoking, etc.) of their mother.

Here, more than previously, the assumptions of the naive bayes approach are clearly wrong. Most variables are binary[3]. The Gaussian assumption is false. Yet, we observe that the technique is robust. It gives results comparable to other linear approaches.

4.2 Comparison of the linear methods

We conducted a study similar to the previous one. We launched each learning method on the same dataset. Here is the Tanagra diagram.

[2] Hosmer and Lemeshow (2000) Applied Logistic Regression: Second Edition.
[3] http://www.statlab.uni-heidelberg.de/data/linmod/birthweight.html



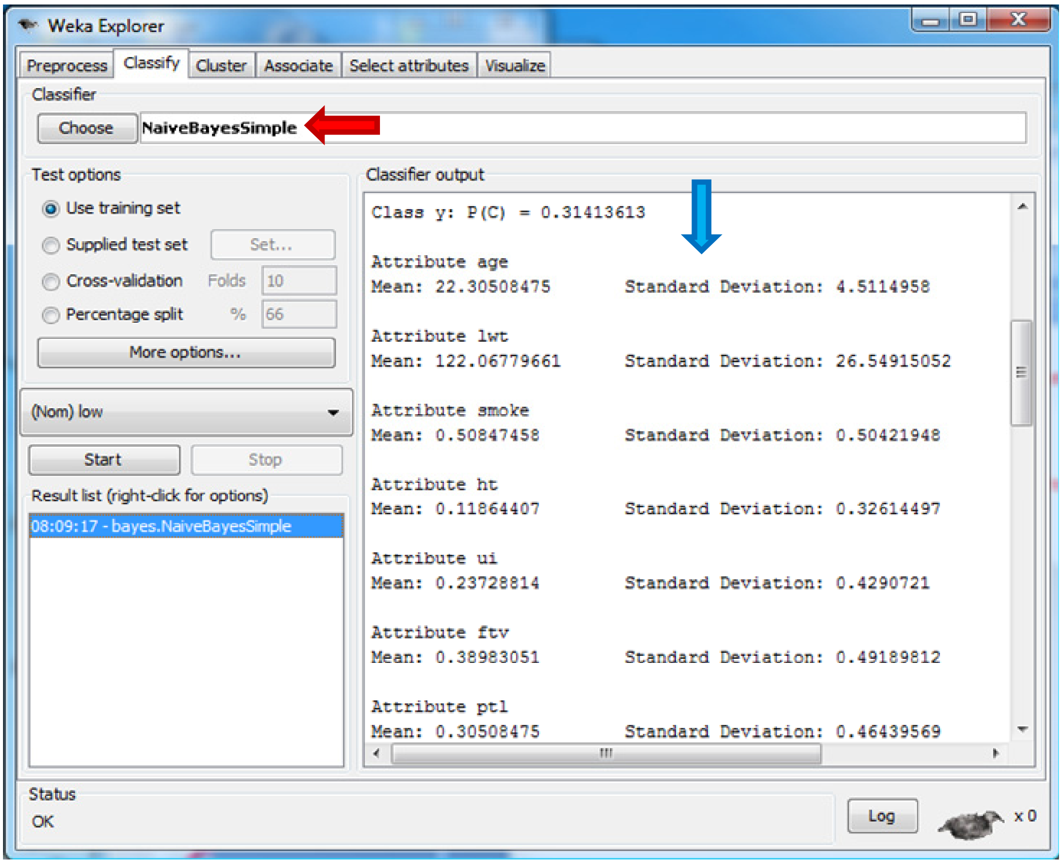
We obtain the following decision functions.

Descriptors	NBC (linear)	Logistic.Reg	SVM	C-PLS	LDA
Intercept	1.1509	1.4915	-0.9994	0.3284	1.0666
age	-0.0487	-0.0467	0.0000	-0.0073	-0.0415
lwt	-0.0121	-0.0140	0.0000	-0.0020	-0.0125
smoke	0.7250	0.4460	0.0012	0.0952	0.4921
ht	1.3659	1.8330	0.0043	0.3622	2.0134
ui	1.0459	0.6680	0.0014	0.1447	0.7927
ftv	-0.4737	-0.2551	-0.0006	-0.0529	-0.2684
ptl	1.7003	1.3368	1.9973	0.2893	1.5611

The results are again consistent (except SVM). In addition, the values of the coefficients are very similar for the naive bayes, the logistic regression and the linear discriminant analysis. It means that the interpretation of the influence of the predictors in the classification process is the same.

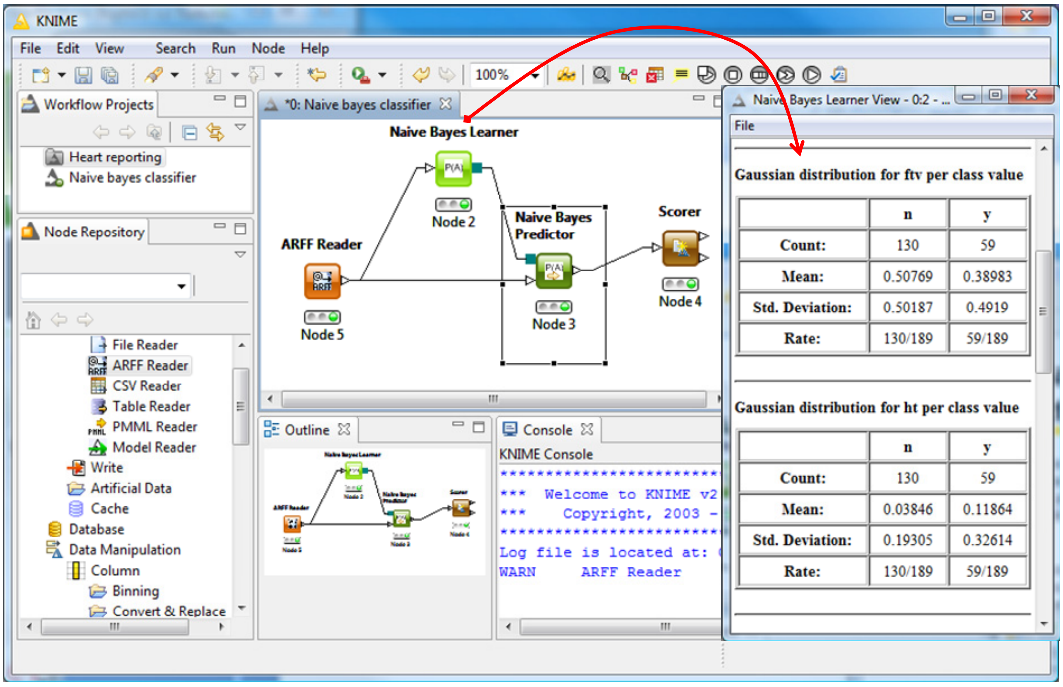
4.3 NBC under Weka

We use Weka under the EXPORER mode. After loading the dataset, we select the CLASSIFY tab. Among the large number of available methods, we choose NAIVE BAYES SIMPLE which corresponds to the naive bayes classifier with the Gaussian heteroscedastic assumption presented in this document. Weka provides no explicit model that we can easily deploy. It simply displays the conditional averages and standard deviations for each predictor. The confusion matrix computed on the learning set is identical to that proposed in Tanagra.



4.4 NBC under Knime

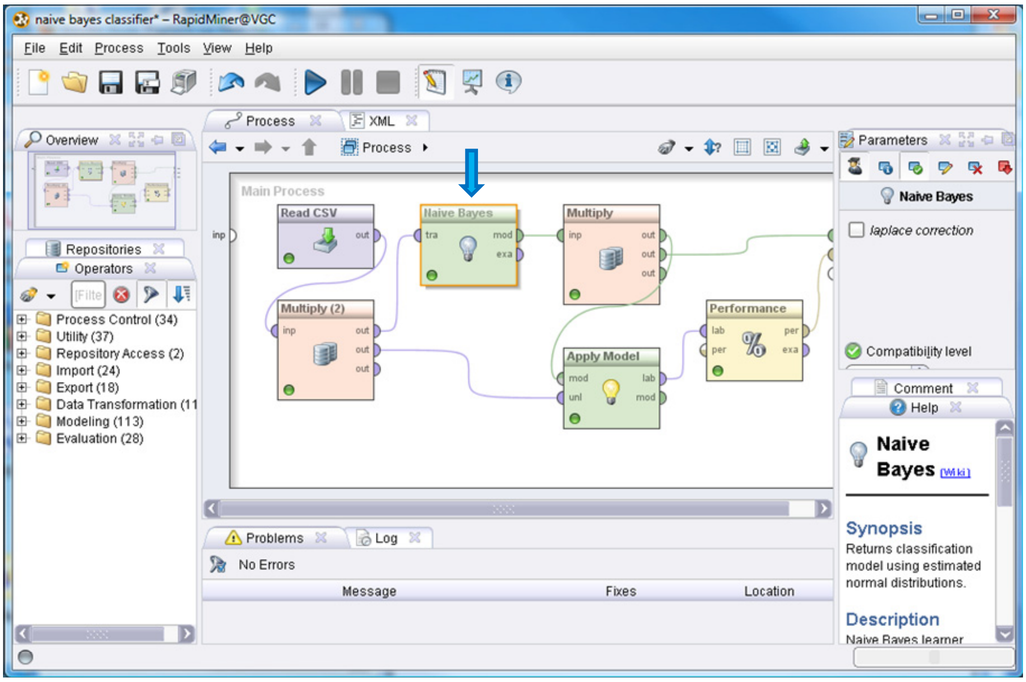
We create the following diagram under Knime 2.2.2.



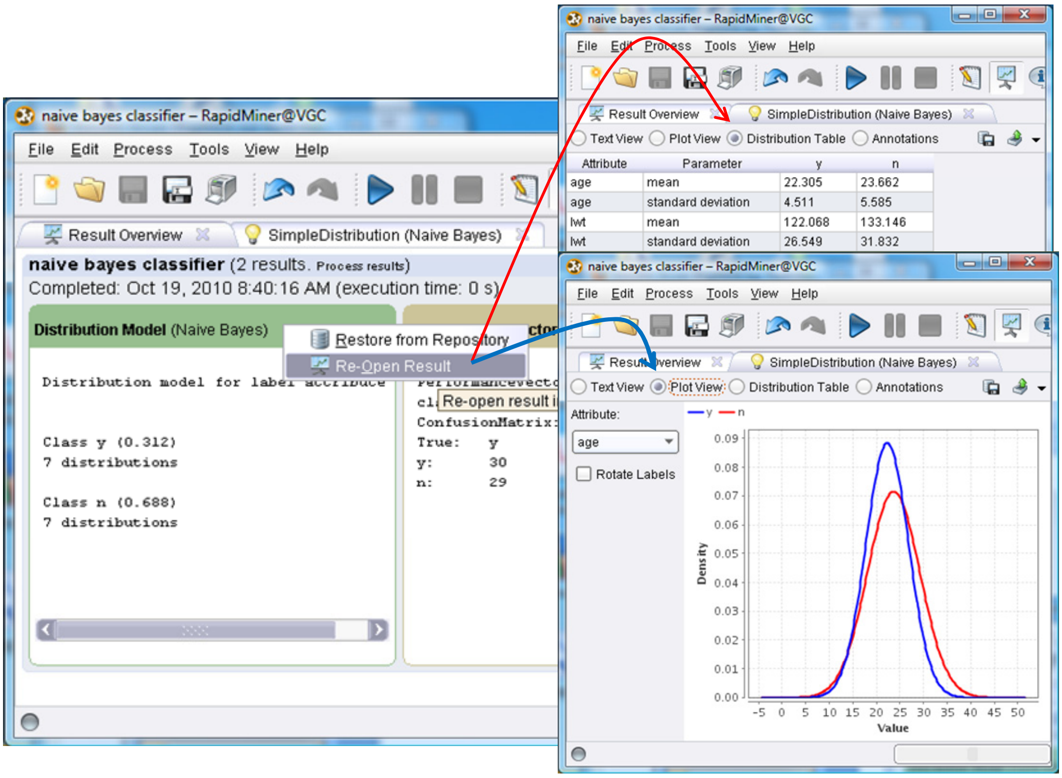
Like Weka, Knime displays the estimated parameters of the conditional Gaussian distributions. We have the same parameters. But curiously we do not have the same confusion matrix when we apply the model on the learning set. Consequently, we do not obtain the same error rate. I confess I do not understand the origin of this discrepancy.

4.5 NBC under RapidMiner

The diagram under RapidMiner 5.0 is very similar to the one created into Knime. We apply the model on the learning set to obtain the resubstitution error rate.



The results are available in a new tab of the main window.



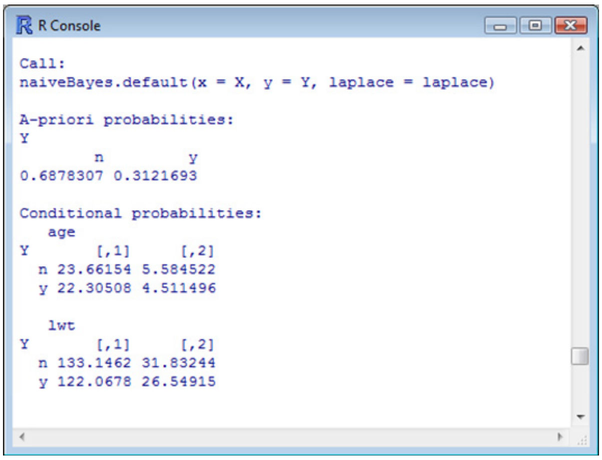
The results (estimated parameters) is the same that those of Weka. The confusion matrix is also consistent. No explicit model is provided.

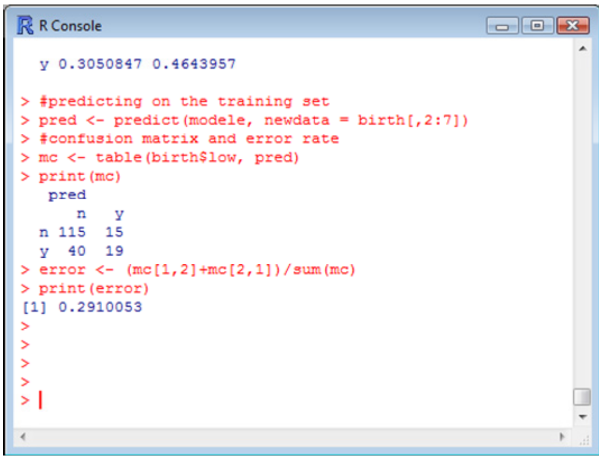
4.6 NBC under R

We use the **e1071** package for R. With the following source code: (1) we learn the model from the dataset; (2) we visualize it; (3) we apply the model on the training set; (4) we compute the confusion matrix and the error rate.

```
#loading the dataset
birth <- read.table(file="low_birth_weight_nbc.txt",header=T,sep="\t")
summary(birth)
#loading the package
library(e1071)
#learning process
modele <- naiveBayes(low ~ ., data = birth)
print(modele)
#predicting on the training set
pred <- predict(modele, newdata = birth[,2:7])
#confusion matrix and error rate
mc <- table(birth$low, pred)
print(mc)
error <- (mc[1,2]+mc[2,1])/sum(mc)
print(error)
```

The **naiveBayes(.)** procedure of the e1071 package provides the conditional average and standard deviation. The computed parameters are consistent with those estimated with Weka, RapidMiner or Tanagra (heteroscedastic assumption).





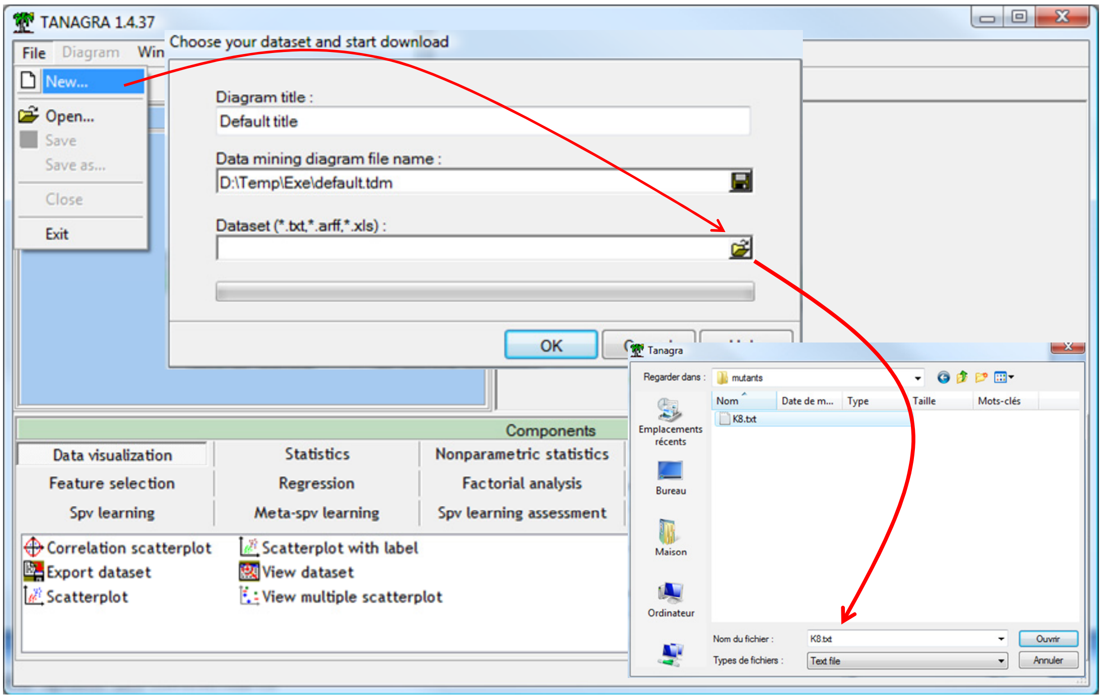
5 Handling a large dataset with the NBC

The main strength of the naive bayes approach is its ability to deal with very large dataset (both the number of instances and the number of predictors). The memory occupancy remains reasonable. And the computation process is very fast in comparison with the other techniques. A single pass over the data file is necessary for the computation of all the parameters of the classifier.

In this section, we treat a data file with 16,592 instances and 5,408 predictive attributes. When we load the dataset, the memory occupation is already high. For all the other techniques, the feasibility of the learning process is questionable. We note for that matter that none of the methods discussed in this tutorial was able to complete the construction of the classifier, whatever the data mining tool used (including Tanagra). The naive bayes approach, on the other hand, has easily built the model in a very short time.

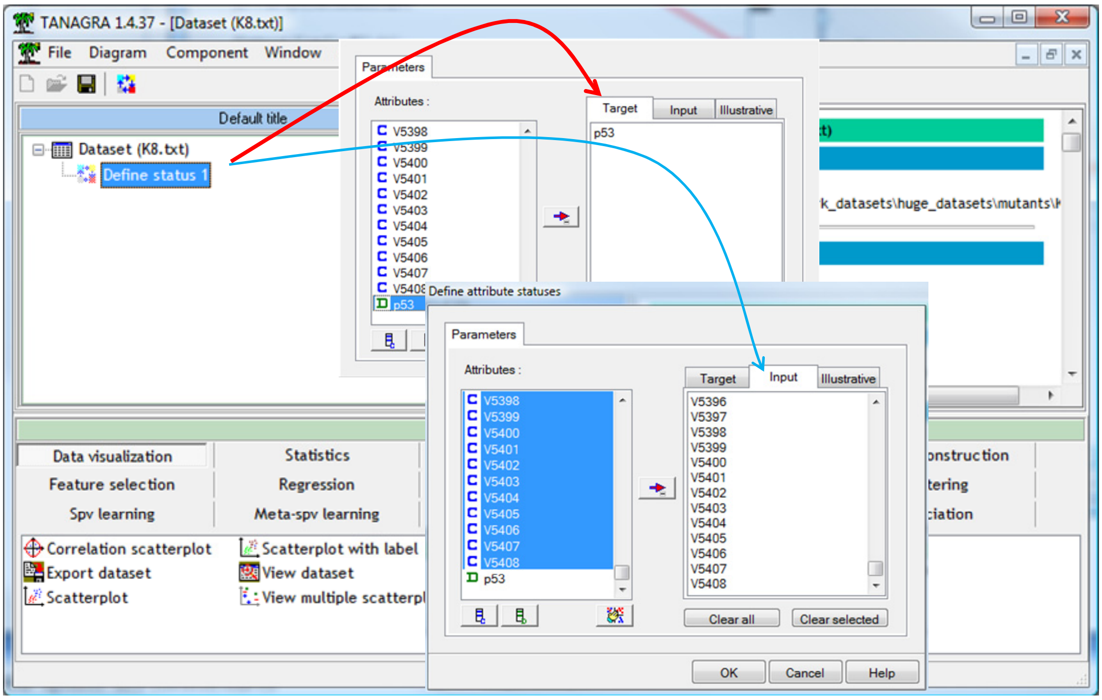
5.1 Importing the dataset

We use the "mutants" dataset (http://archive.ics.uci.edu/ml/datasets/p53+Mutants). The target attribute is « p53 » (active: positive vs. inactive: negative). We launch Tanagra. We click on the FILE / NEW menu to create a diagram and to import the dataset. We select the "K8.txt" data file.

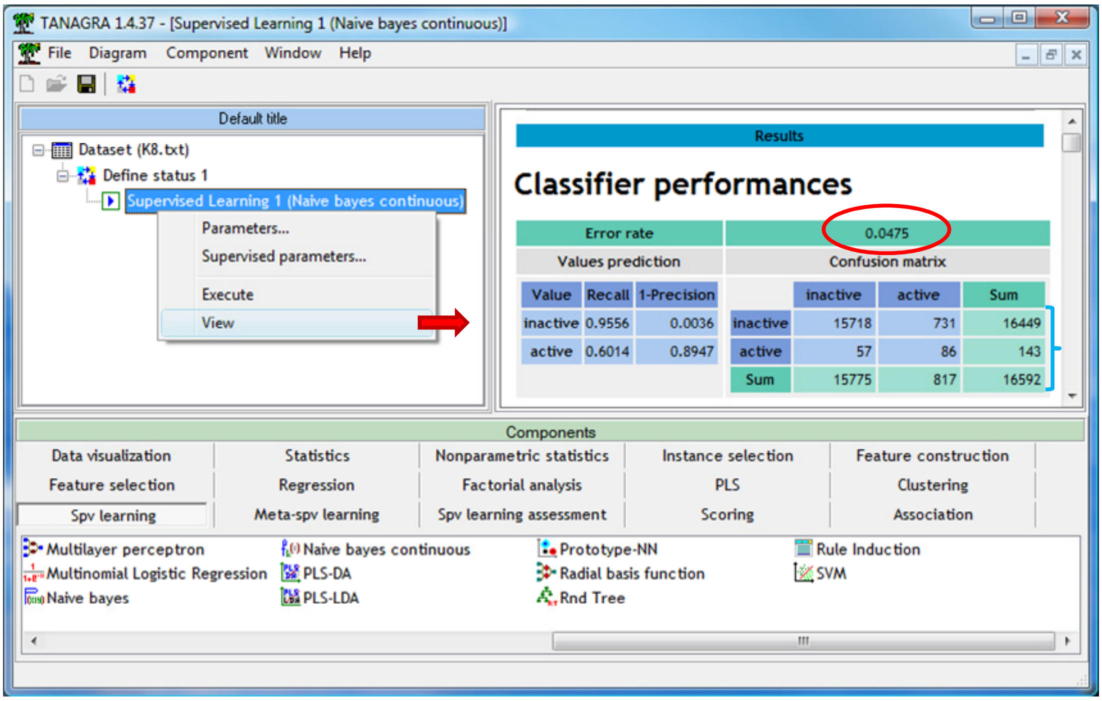


5.2 Learning process

We use the DEFINE STATUS component to specify the status of the variables.



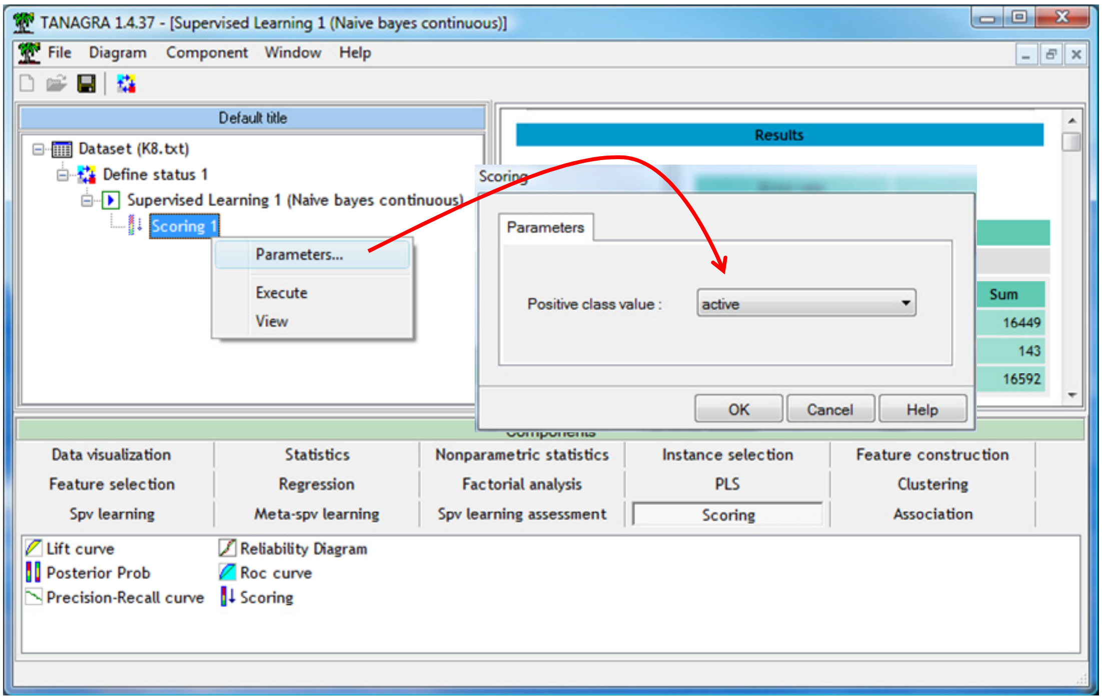
We add the NAIVE BAYES CONTINUOUS component into the diagram. We click on the VIEW menu, we obtain the classifier. By default, Tanagra builds the linear classifier.



The resubstitution error rate is 4.75%. But it is not really useful because we have a very highly imbalanced dataset. The error rate of the default classifier is 0.86%. The computation time is 10.7 seconds. The memory occupation of the model is negligible in relation to the memory occupation of the dataset.

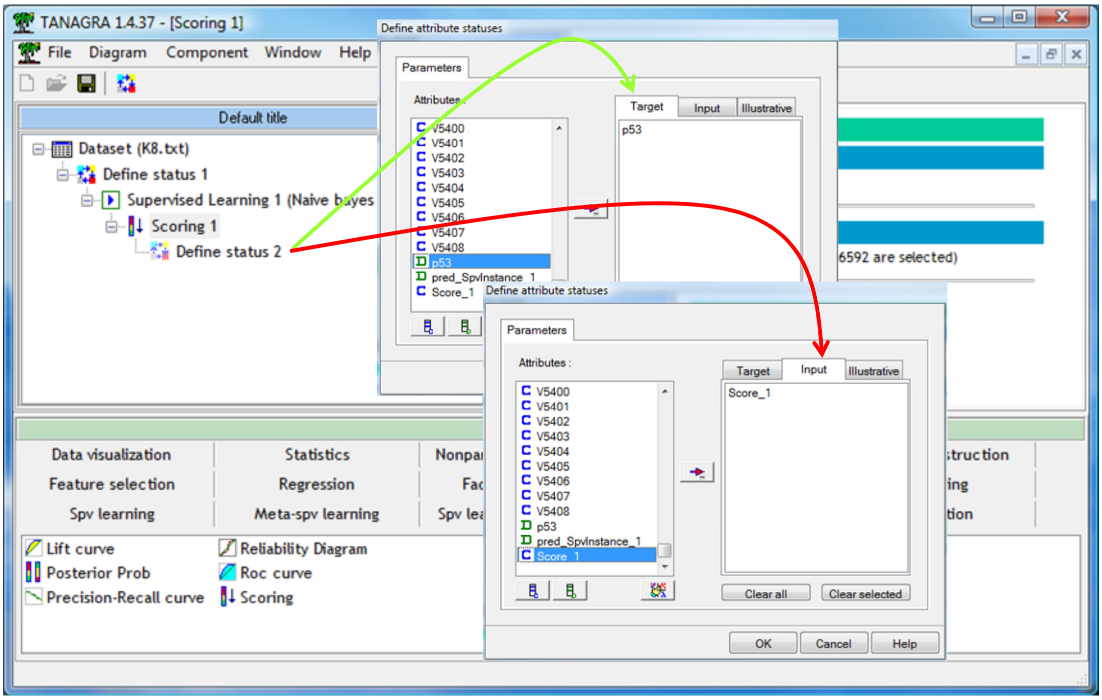
5.3 Assessment using the ROC curve

We want to assess the classifier using a ROC curve.

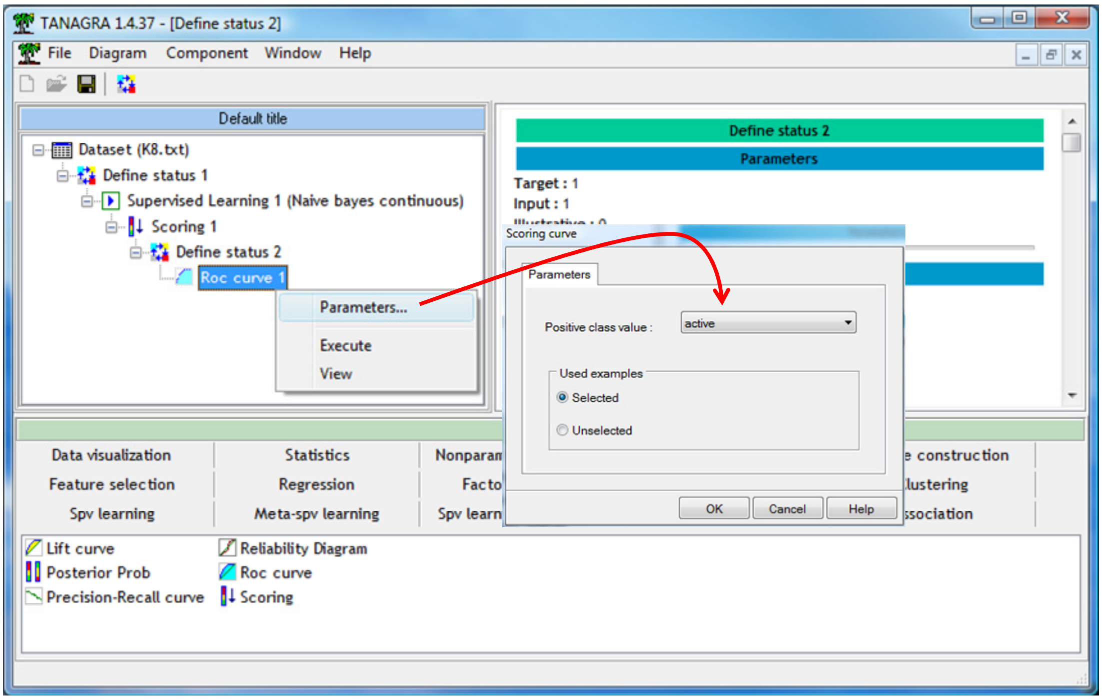


First, we must assign a "score" to each instance. We add the SCORING (SCORING tab) into the diagram. We set the following parameter (p53 = active is the positive class).

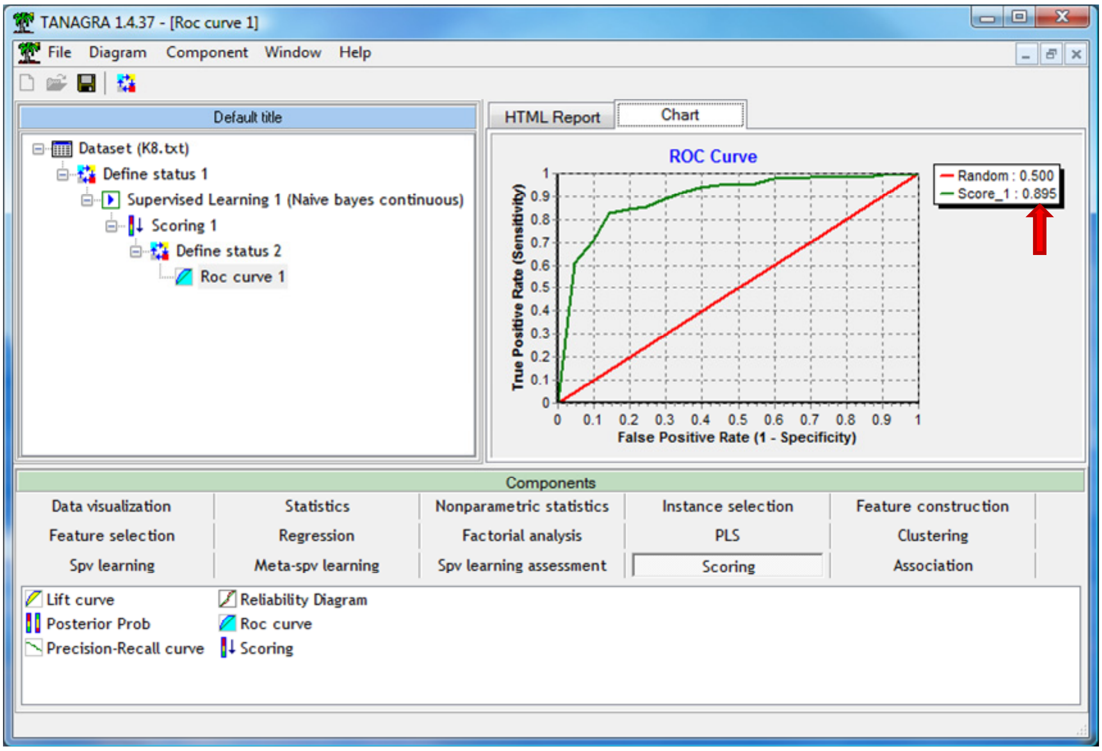
Then, using the DEFINE STATUS component, we set the class attribute as TARGET, the score column as INPUT.



We add the ROC CURVE component into the diagram. We set the following parameters: p53 = active is the positive class value; we build the curve on the learning set (the aim is simply to show the feasibility of the calculations).



We click on the VIEW menu. We obtain the ROC curve. The area under curve (AUC = 0.895) criterion shows that the model is finally better than the default classifier.



5.4 Computation time and memory occupation

We summarize here the computation time and the memory occupation at each step of the process.

Step	Computation time (sec.)	Memory occupation (KB)
Launching the tool	-	3 620
Loading the dataset	25.7	489 252
Learning the model	10.7	490 928
Scoring the instances	5.9	491 008
Constructing the ROC curve	0.6	508 088

At the same level of performance, I doubt that we can do better - faster, less memory - with another method that produces a linear classifier.

6 Conclusion

In this tutorial, we show that it is possible to produce an explicit model in the form of linear combination of variables, and possibly their squared, for the naive bayes classifier with continuous predictors. This document is the counterpart of the paper dedicated to the discrete predictors (see http://data-mining-tutorials.blogspot.com/2010/07/naive-bayes-classifier-for-discrete.html).

The result is not trivial because we accumulate all the benefits. On the one hand, the learning process is very fast, we can handle very large databases. On the other hand, we obtain an explicit model which is interpretable (we can analyze the influence on each attribute in the prediction process). The deployment is made easier: a simple set of coefficients [(number of values of the target attribute) times (number of predictors + 1) in the worst case] is necessary for its diffusion.