1 Objectif

Etudier l'influence des différentes techniques de traitement des données manquantes sur le comportement de la régression logistique.

L'appréhension des données manquantes est un problème difficile. Non pas à cause de sa gestion informatique qui est relativement simple, il suffit de signaler les valeurs manquantes par un code spécifique, mais plutôt à cause des conséquences de leur traitement (suppression des lignes ou des colonnes du fichier ; ou remplacement par une valeur calculée à partir de observations disponibles, on parle alors d'imputation) sur les caractéristiques des modèles élaborés.

Nous en avions parlé dans un précédent document. Il s'agissait alors d'étudier l'impact des différentes techniques de traitement de valeurs manquantes sur les arbres de décision construits avec la méthode C4.5 (Quinlan, 1993)[1] dans le logiciel SIPINA. Aujourd'hui, nous réitérons l'analyse en étudiant leur influence sur les résultats de la **régression logistique**. Nous utiliserons principalement le logiciel **R**, avec la procédure **glm(.)**. Par la suite, nous examinerons le comportement des outils proposés dans des logiciels tels qu'**Orange**, **Knime** et **RapidMiner** placés dans un contexte identique.

Notons, concernant les logiciels, que Tanagra ne gère pas les valeurs manquantes. C'est un choix délibéré. Je ne souhaitais pas que les étudiants, qui sont le principal public de Tanagra, évacuent d'un clic la gestion de données manquantes parce qu'un traitement par défaut est proposé. Réfléchir en amont au problème et proposer une technique en adéquation avec les caractéristiques de l'étude que l'on mène me paraît pédagogiquement incontournable.

Justement, de quelle nature sont ces caractéristiques qui pèsent sur le traitement des valeurs manquantes ? Tout d'abord, il y a **le processus à l'origine de leur formation**[2]. Il peut être complètement aléatoire (**MCAR** : missing completely at random). Dans ce cas, la survenance d'une observation manquante sur une variable ne dépend ni des valeurs de la variable, ni des valeurs d'autres variables de la base. Il peut être aléatoire (**MAR** : missing at random). Dans ce cas, sa survenance ne dépend pas des valeurs de la variable après avoir contrôlé le rôle des autres variables de la base. Par exemple, les cadres sont moins enclins à déclarer leur salaire dans une enquête. Mais dans cette catégorie de personnes, la probabilité d'apparition de valeur manquante ne dépend pas de la valeur du salaire. Il peut être non aléatoire (**MNAR** : missing not at random). Par exemple, les personnes qui ont un salaire élevé préfèrent éluder la question lors d'une enquête. Le problème est alors particulièrement difficile. Il faut passer par un traitement spécifique.

De toutes ces configurations, le cas MCAR est le plus facile à gérer. Nous obtenons des estimations non biaisées des paramètres, même lorsque nous utilisons des techniques très frustes comme la suppression des observations comportant au moins une valeur manquante.

[1] http://tutoriels-data-mining.blogspot.com/2009/10/sipina-traitement-des-donnees.html

[2] http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Missing_Data/Missing.html

L'analyse statistique appiquée sur les données est une autre pièce importante du puzzle. Clairement, l'influence des valeurs manquantes et de leur traitement n'est pas la même dans une analyse prédictive (la régression logistique par exemple), dans un clustering (ex. une classification ascendante hiérarchique) ou dans une analyse factorielle (ex. une analyse en composante principale). La distinction est d'autant plus importante que certaines méthodes intègrent nativement un procédé d'appréhension des valeurs manquantes (ex. l'algorithme NIPALS[3] pour l'analyse en composantes principales et la régression PLS).

Enfin, **le critère d'évaluation du traitement des données manquantes** est la dernière pièce du puzzle. L'objectif n'est pas de retrouver les hypothétiques « vraies » valeurs des observations non-renseignées pour chaque variable. Mais plutôt de proposer des valeurs de remplacement (dans le cas de l'imputation) qui ne dénaturent pas les résultats de l'étude que nous sommes en train de mener. On met souvent en avant le biais et la variance des paramètres estimés des modèles en statistique. Mais dans le cadre spécifique du data mining prédictif, on peut aussi légitimement se poser la question de la performance. Est-ce que la technique de traitement des données manquante permet de produire le modèle le plus efficace possible en prédiction ?

Dans ce tutoriel, nous nous plaçons dans la configuration suivante : (1) les valeurs manquantes sont MCAR, nous avons écrit un programme qui retire de manière complètement aléatoire les valeurs dans l'échantillon d'apprentissage ; (2) nous appliquons la régression logistique sur les données d'apprentissage post-traitées ; (3) nous évaluons les différentes techniques de traitement des données manquantes en observant le taux de bon classement (ou taux de succès) du modèle sur un échantillon test à part qui, lui, ne comporte aucune valeur manquante.

Dans un premier temps, nous mènerons l'expérimentation avec R. Nous opposerons la suppression de ligne (listwise deletion) à l'imputation univariée (remplacement par la moyenne pour les variables quantitatives, par le mode pour les qualitatives). Nous constaterons que cette dernière est une approche très viable dans le cadre MCAR. Dans un deuxième temps, nous étudierons les options proposées dans les logiciels Orange, Knime et RapidMiner. Nous constaterons que malgré leur sophistication, ils ne font pas mieux que l'imputation univariée. Mais rappelons-le encore une fois, nous nous plaçons dans un cadre très particulier (MCAR + régression logistique + taux d'erreur en test). D'autres études seraient le bienvenu (*avis aux amateurs*) dans des configurations différentes.

2 Données

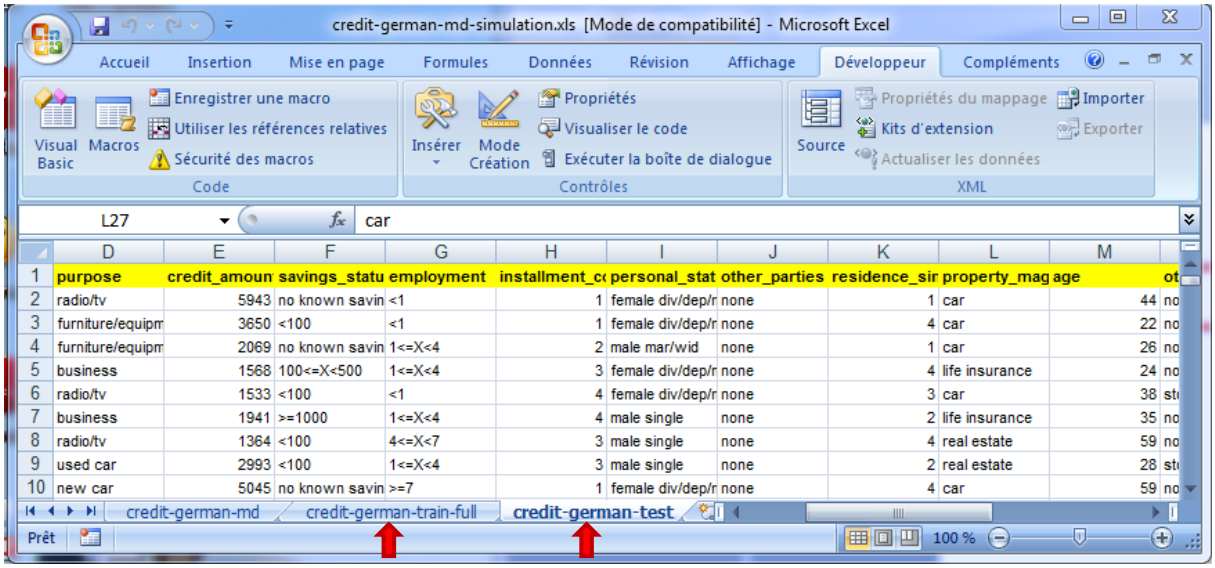
Données disponibles. Nous utilisons les données GERMAN CREDIT DATA dans notre expérimentation[4]. Nous cherchons à prédire la classe d'un individu (bon ou mauvais client) lors d'une demande de crédit. Nous avons scindé aléatoirement le fichier en deux feuilles dans un classeur Excel (**credit-german-md-simulation.xls**) : CREDIT-GERMAN-TRAIN-FULL, réservé à l'apprentissage, comporte 300 observations ; CREDIT-GERMAN-TEST, réservé au test, 700 observations. Pour ce

[3] http://en.wikipedia.org/wiki/Non-linear_iterative_partial_least_squares

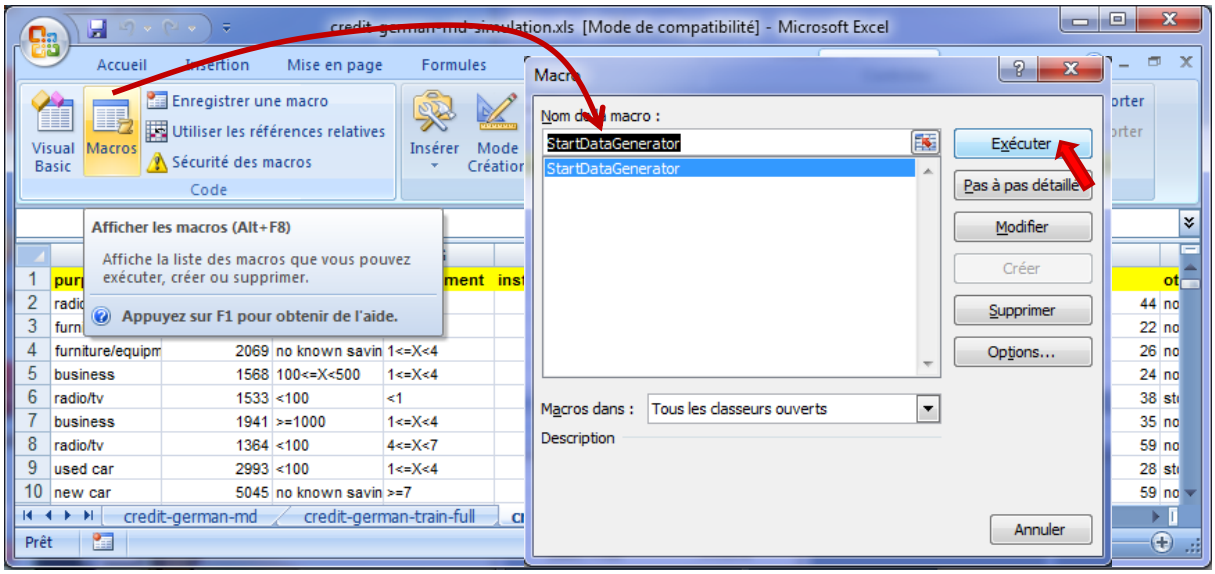
[4] http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

second échantillon, nous avons créé une fois pour toutes un fichier au format texte avec séparateur tabulation CREDIT-GERMAN-TEST.TXT.

Il n'y a aucune valeur manquante dans ces deux échantillons.



Génération d'un échantillon d'apprentissage comportant des valeurs manquantes. Nous utilisons une macro VBA (visual basic pour applications) pour générer un fichier d'apprentissage intégrant des valeurs manquantes dans la première feuille du classeur (CREDIT-GERMAN-MD)[5]. Pour ce faire, nous actionnons le menu DEVELOPPEUR / MACRO d'EXCEL[6] et dans la boîte de paramétrage qui apparaît, nous sélectionnons la macro STARTDATAGENERATOR.

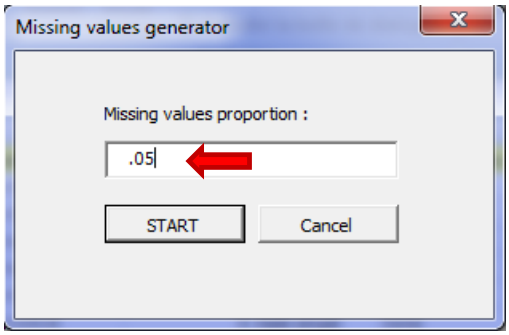


Le programme est démarré, une seconde boîte de paramétrage, propre à la macro, permet de spécifier la proportion de valeurs manquantes qu'il faut générer, en pourcentage des valeurs disponibles. Par exemple, si nous mettons 0.05 (5 %), le programme créera un fichier d'apprentissage

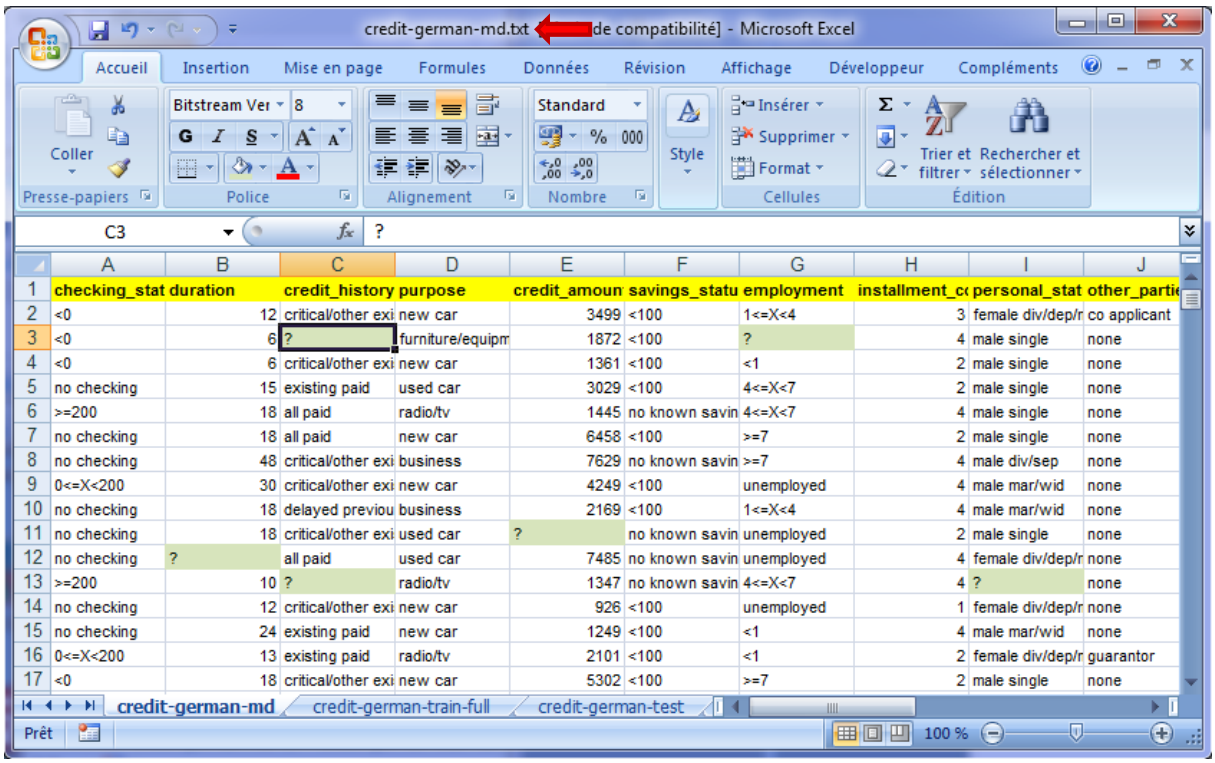
[5] **Attention, il faut veiller à ce que les macros soient activées lorsque vous importez le fichier dans Excel !**

[6] Pour Excel 2003 et antérieures, il faudrait chercher dans le menu OUTILS / MACROS / MACROS.

avec 0.05 * (300 lignes * 20 variables) = 300 cellules **avec l'indicateur « ? » pour signaler l'absence de valeur**. La colonne de la variable à prédire CLASSE n'est pas concernée par cette opération.

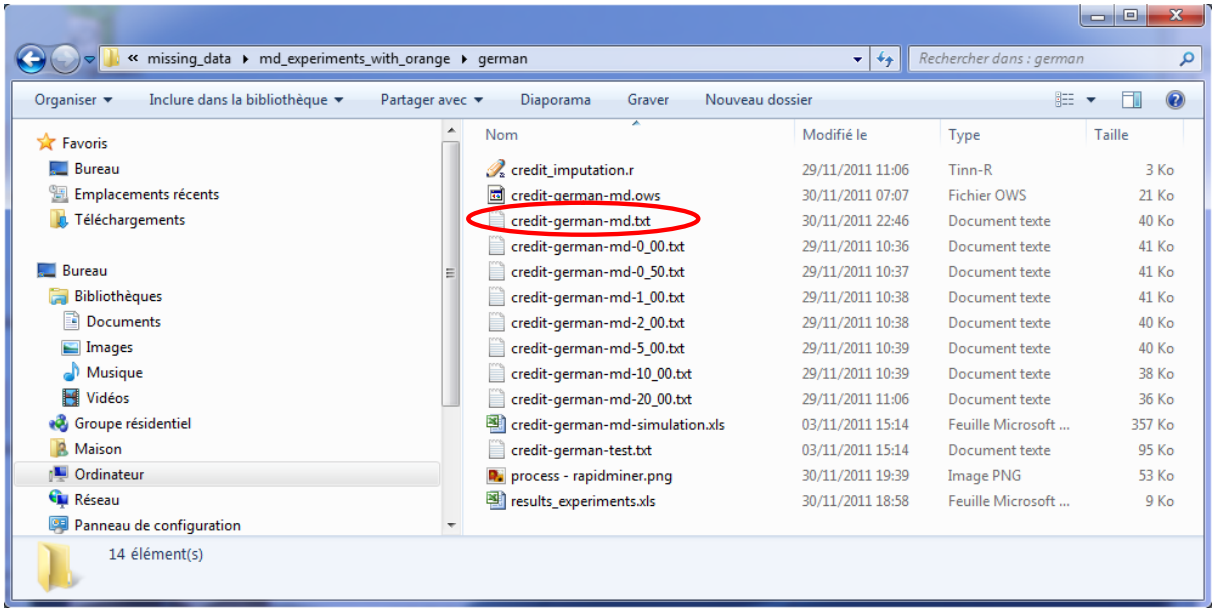


Les données résultantes sont intégrées dans la première feuille du classeur (CREDIT-GERMAN-MD).



	A	B	C	D	E	F	G	H	I	J
1	checking_stat	duration	credit_history	purpose	credit_amoun	savings_statu	employment	installment_c	personal_stat	other_parti
2	<0	12	critical/other exi	new car	3499	<100	1<=X<4	3	female div/dep/r	co applicant
3	<0	6	?	furniture/equipm	1872	<100	?	4	male single	none
4	<0	6	critical/other exi	new car	1361	<100	<1	2	male single	none
5	no checking	15	existing paid	used car	3029	<100	4<=X<7	2	male single	none
6	>=200	18	all paid	radio/tv	1445	no known savin	4<=X<7	4	male single	none
7	no checking	18	all paid	new car	6458	<100	>=7	2	male single	none
8	no checking	48	critical/other exi	business	7629	no known savin	>=7	4	male div/sep	none
9	0<=X<200	30	critical/other exi	new car	4249	<100	unemployed	4	male mar/wid	none
10	no checking	18	delayed previou	business	2169	<100	1<=X<4	4	male mar/wid	none
11	no checking	18	critical/other exi	used car	?	no known savin	unemployed	2	male single	none
12	no checking	?	all paid	used car	7485	no known savin	unemployed	4	female div/dep/r	none
13	>=200	10	?	radio/tv	1347	no known savin	4<=X<7	4	?	none
14	no checking	12	critical/other exi	new car	926	<100	unemployed	1	female div/dep/r	none
15	no checking	24	existing paid	new car	1249	<100	<1	4	male mar/wid	none
16	0<=X<200	13	existing paid	radio/tv	2101	<100	<1	2	female div/dep/r	guarantor
17	<0	18	critical/other exi	new car	5302	<100	>=7	2	male single	none

Dans le même temps, le fichier d'apprentissage au format texte « CREDIT-GERMAN-MD.TXT » est automatiquement créé dans notre répertoire de travail (le répertoire du fichier XLS).



Pour les férus de programmation, voici le code VBA associé au générateur de données.

```
Private Sub cmdBOk_Click()
'retrieve the proportion from a dialogbox
Dim proportion As Double
proportion = Val(tbProportion.Text)
'checking the range of the value (< 0.5 max.)
If (proportion < 0#) Or (proportion >= 0.5) Then
    proportion = 0.01
End If
'copying the full training set to the first sheet
Sheets("credit-german-train-full").Select
Range("A2:U301").Select
Selection.Copy
Sheets("credit-german-md").Select
Range("A2").Select
ActiveSheet.Paste
'number of cells to clear
Dim nbMd As Long
nbMd = Int(6000# * proportion)
'counter for the number of missing values
Dim counter As Long
counter = 1
'coordinates
Dim i As Long, j As Long
'initialize the random number generator
Randomize (100)
Do While (counter <= nbMd)
    'row
    i = 2 + Int(300# * Rnd)
    'column
    j = 1 + Int(20# * Rnd)
    'checking if the cell is already empty
```

```
    If (Cells(i, j).Value <> "?") Then
        Cells(i, j).Value = "?"
        counter = counter + 1
    End If
Loop
'set the text file name
Dim nomFichier As String
nomFichier = "\credit-german-md.txt"
nomFichier = ThisWorkbook.Path + nomFichier
'checking if the file already exists
Dim fs As Variant
Set fs = CreateObject("Scripting.FileSystemObject")
If (fs.FileExists(nomFichier) = True) Then
    fs.deletefile (nomFichier)
End If
'save the sheet into a text file
ThisWorkbook.SaveAs fileName:=nomFichier, FileFormat:=xlTextWindows
'close the form (dialogbox for the proportion setting)
formMD.Hide
End Sub
```

Notons une information très importante : nous avons inséré 300 valeurs manquantes dans notre ensemble d'apprentissage. Cela ne veut pas dire pour autant qu'il y a 300 observations avec valeurs manquantes. En effet, dans certains cas, certaines observations peuvent comporter deux, voire plus, cellules vides (ex. la ligne n°13 dans notre copie d'écran ci-dessus). Dans les traitements qui suivront, nous compterons les lignes complètes, c.-à-d. ne comportant aucune valeur manquante, dans notre base d'apprentissage.

3 Traitement des valeurs manquantes avec R

3.1 Architecture du script

Nous souhaitons opposer deux techniques de traitement des données manquantes : la suppression d'observations comportant au moins une valeur manquante (listwise deletion) et l'imputation univariée (remplacement par la moyenne pour les variables quantitatives, par le mode pour les qualitatives). Pour évaluer la qualité de la stratégie, nous appliquons les modèles élaborés sur le même échantillon test exempt de valeurs manquantes[7]. La meilleure stratégie est celle qui induit le modèle le plus performant en prédiction. Nous nous basons sur le taux de succès.

Décomposons notre programme R pour comprendre notre démarche.

Chargement des données. Dans un premier temps, nous chargeons l'échantillon d'apprentissage (comportant éventuellement des valeurs manquantes, sauf sur la variable à prédire) et de test (exempt de valeurs manquantes).

[7] L'application d'un modèle sur des individus partiellement décrits est un autre type de problème qui n'entre pas dans le cadre de notre étude. Il n'en est pas moins intéressant. Peut être le sujet d'un futur tutoriel ?

```
#chargement du fichier test
setwd("… votre répertoire …")
data.test <- read.table(file="credit-german-test.txt",dec=".",sep="\t",header=T)
#chargement des données d'apprentissage avec les valeurs manquantes
data.train <- read.table(file="credit-german-md.txt",dec=".",sep="\t",header=T,na.strings="?")
summary(data.train)
```

Suppression des lignes. La première stratégie consiste à supprimer du fichier les lignes comportant au moins une valeur manquante. Elle est un peu brutale, on peut vider très rapidement le fichier avec ça. Mais elle a le mérite de produire des estimations non biaisées des paramètres de la régression dans le contexte MCAR.

```
#première solution : listwise deletion
data.train.omitted <- na.omit(data.train)
summary(data.train.omitted)
print(paste('Remaining observations : ',as.character(nrow(data.train.omitted))))
model.omitted <- glm(classe ~ ., family = binomial, data = data.train.omitted)
```

na.omit(.) supprime de la base les observations présentant au moins une valeur manquante. Nous profitons de ce traitement pour afficher le nombre d'observations complètes dans le fichier d'apprentissage. **model.omitted** est le modèle obtenu à l'issue de ce pré-traitement.

Remplacement univarié. Pour l'imputation univariée, nous utilisons le programme suivant pour générer le modèle prédictif **model.imputed**.

```
#seconde solution : imputation univariée (moyenne, mode)
data.train.imputed <- as.data.frame(lapply(data.train,traiter_missing_data))
summary(data.train.imputed)
model.imputed <- glm(classe ~ ., family = binomial, data = data.train.imputed)
```

La fonction **traiter_missing_data(x)** fait la distinction entre les variables quantitatives et qualitatives (le type factor de R).

```
#traiter les données manquantes
traiter_missing_data <- function(x){
 if (is.factor(x) == T){
  return(traiter.discrete(x))
 } else {
  return(traiter.numeric(x))
 }
}
```

Ainsi, pour le remplacement par la moyenne, nous utiliserons la fonction :

```
#variable numérique, remplacement des NA par la moyenne
traiter.numeric <- function(x){
 y <- x
 z <- na.omit(y)
 if (length(z) < length(y)){
  m <- mean(z)
  y[is.na(y)] <- m
 }
```

```
 return(y)
}
```

L'idée est simple. Nous calculons la moyenne sur les données disponibles. Puis, nous remplaçons les valeurs manquantes par cette moyenne.

De la même manière, nous utilisons le mode pour les variables qualitatives.

```
#variable catégorielle, remplacement des NA par le mode
traiter.discrete <- function(x){
  y <- x
  z <- na.omit(y)
  if (length(z) < length(y)){
    frequence <- table(z)
    m <- which.max(frequence)
    y[is.na(y)] <- levels(x)[m]
  }
  return(y)
}
```

Evaluation des modèles. Dernière étape de notre expérimentation, nous devons évaluer les modèles. Nous utilisons une fonction spécifique pour cela, elle prend en entrée l'échantillon test et le modèle. Elle se charge d'effectuer la prédiction, de construire la matrice de confusion et de calculer le taux de bon classement (= 1 – taux d'erreur)[8].

```
#construction matrice de confusion à partir du modèle
#new.dataset représente l'échantillon test
#model est le modèle
pred_and_confusion_matrix <- function(new.dataset,model){
  #probabilité prédité par le modèle
  data.pred.prob <- predict(model,newdata = new.dataset)
  #affectation
  data.pred.class <- ifelse(data.pred.prob > 0.5,"B","A")
  data.pred.class <- as.factor(data.pred.class)
  #matrice de confusion (classe observée vs. prédite)
  mc <- table(new.dataset$classe,data.pred.class)
  print(mc)
  #taux de bon classement
  ca <- (mc[1,1]+mc[2,2])/sum(mc)
  print(ca)
}
```

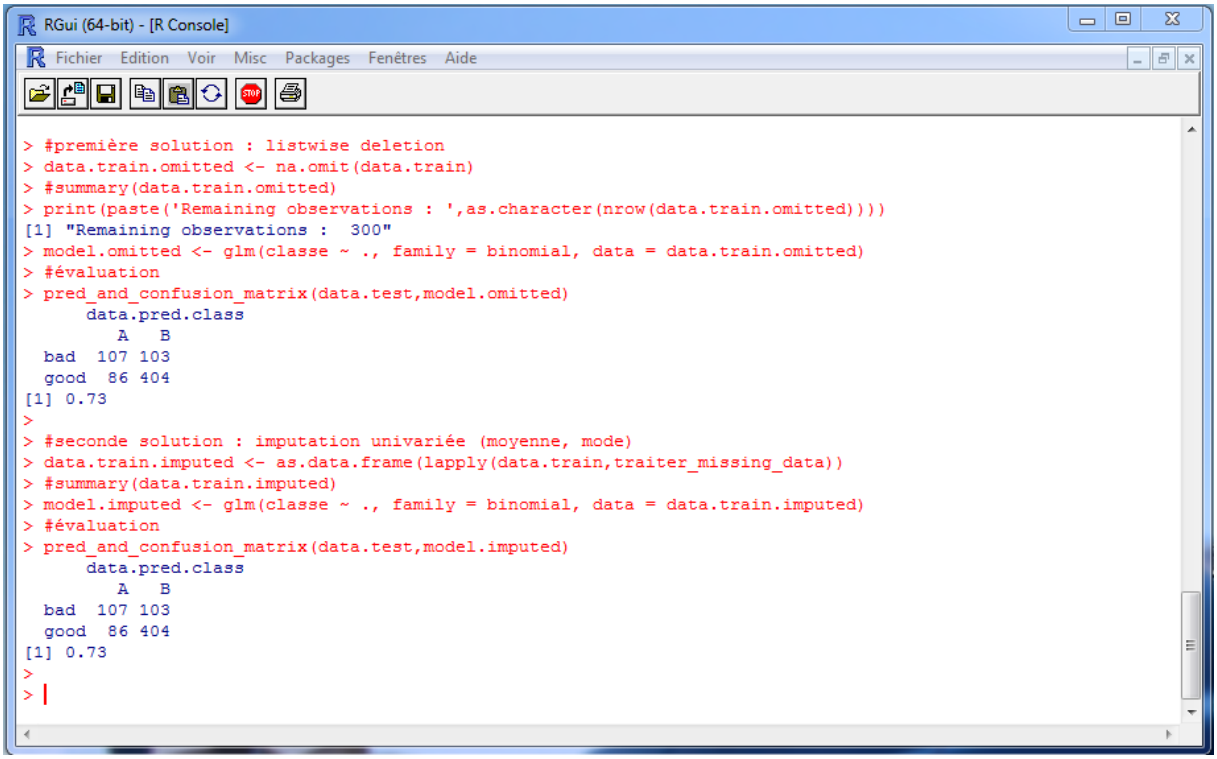
3.2 Organisation de l'expérimentation

L'idée est d'observer l'impact de la stratégie de traitement à mesure que la proportion de valeurs manquantes augmente. Nous essaierons les pourcentages suivants : **0.5%**, **1%**, **2%**, **5%**, **10%**, **20%**.

[8] Nous utilisons le taux de bon classement (ou taux de succès) et non pas le taux d'erreur pour être en adéquation avec les valeurs fournies par Orange dans la suite de ce tutoriel.

Le résultat de référence est la performance du modèle lorsque l'échantillon d'apprentissage est complet (0% de données manquantes). Bien évidemment, dans ce cas, les deux stratégies devraient fournir le même résultat puisqu'aucun traitement n'est effectué.

L'exécution du programme fournit les sorties suivantes.



300 observations sont disponibles après suppression des lignes comportant des valeurs manquantes dans la première stratégie. C'est tout à fait normal puisqu'il n'y en a pas (de valeurs manquantes). Le taux de succès en test est de 73%. Nonobstant les fluctuations d'échantillonnage, nous ne devrions pas faire mieux lorsque nous supprimerons des valeurs dans le fichier d'apprentissage.

3.3 Résultats de l'expérimentation

Les résultats sont recensés dans le tableau suivant.

GERMAN DATASET		Accuracy rate	
% missing	# complete obs.	Listwise Del.	Univ. Imputation
0,00%	300	0,7300	0,7300
0,50%	272	0,7100	0,7257
1,00%	246	0,7129	0,7286
2,00%	201	0,7214	0,7186
5,00%	111	0,6729	0,7114
10,00%	40	ERR	0,7086
20,00%	4	ERR	0,7214

Plusieurs informations attirent notre attention :

- La présence de valeurs manquantes dégrade le modèle. Le contraire aurait été étonnant.
- Tant que la proportion des valeurs manquantes reste faible, jusqu'à 2%, ce qui correspond à 201 observations complètes sur 300, les deux stratégies se valent.
- Lorsque cette proportion augmente (à partir de 5%), la « listwise deletion » dégrade fortement l'apprentissage au point de la rendre impossible pour 10% de valeurs manquantes. En effet, il ne reste que 40 observations après suppression des lignes. R n'a pas voulu lancer la régression.
- A contrario, l'imputation univariée se maintient remarquablement bien**. Dans la configuration MCAR**, la stratégie tient la route même lorsqu'il y a 20% de valeurs manquantes dans l'échantillon d'apprentissage. A la réflexion ce n'est pas très étonnant. Les « ? » étant bien répartis dans l'ensemble des colonnes, il y a finalement peu d'observations manquantes pour chaque variable. Les estimations des valeurs de remplacement sont viables. Un traitement rapide sous Excel nous a permis de constater qu'il y a en moyenne 240 observations disponibles dans chaque colonne.

Encore une fois, nous nous plaçons dans un contexte « de laboratoire » très singulier dans ce tutoriel. Notre principal mérite est d'avoir, autant que faire se peut, délimité le champ de validité de notre expérimentation. Des études complémentaires sont nécessaires pour évaluer le comportement des différentes approches dans d'autres contextes. A ce sujet, je conseille la lecture des excellents ouvrages d'Allison (2001) et de Little & Rubin (2002) référencés en bibliographie.

4 Traitement des données manquantes dans d'autres logiciels

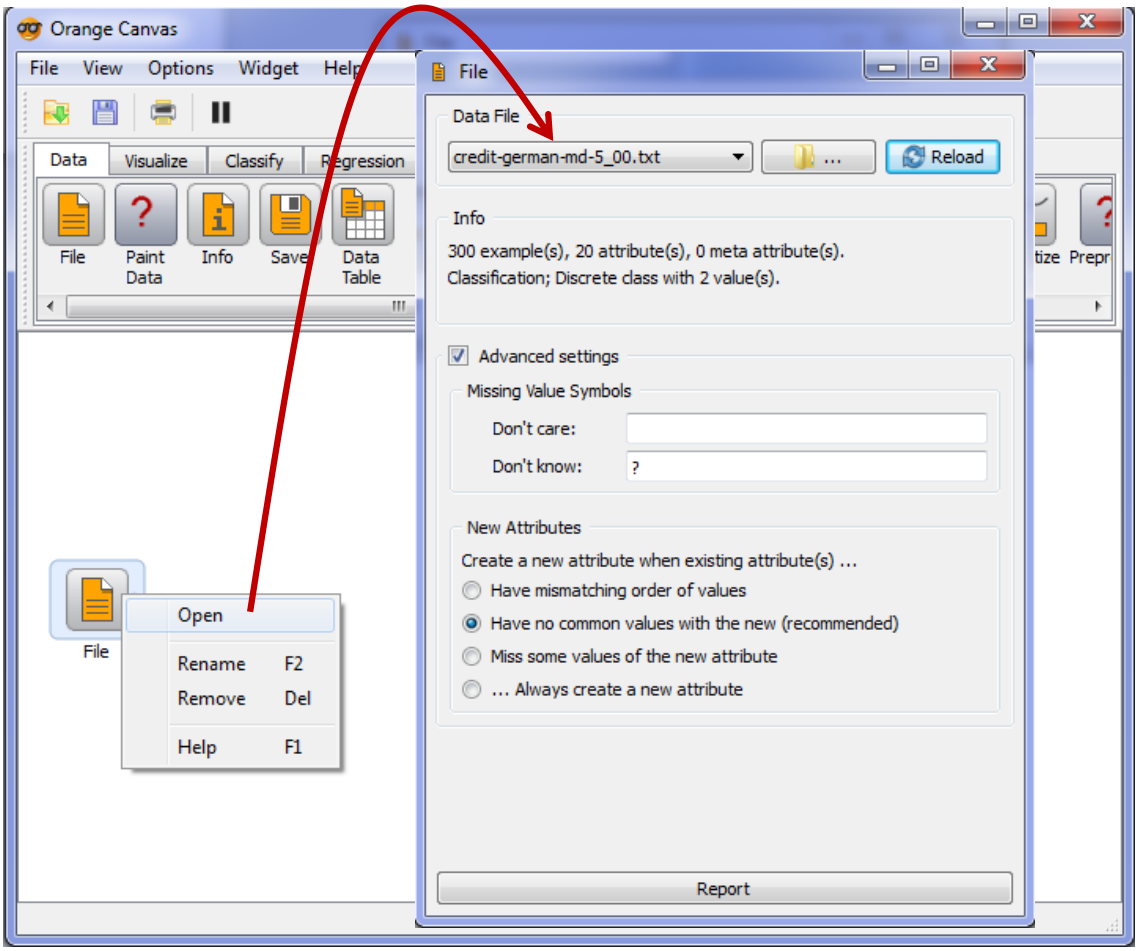
Pour être honnête, l'idée de ce tutoriel m'est venue en découvrant l'outil très élaboré de gestion de données manquantes dans le logiciel Orange. J'ai un peu regardé ce qu'il en était dans les autres logiciels puis, de fil en aiguille, j'ai monté l'expérimentation décrite dans la section précédente pour donner un tour plus scientifique à tout cela.

Nous réalisons un tour d'horizon dans cette section en utilisant le fichier avec 5% de valeurs manquantes (111 observations complètes sur les 300).

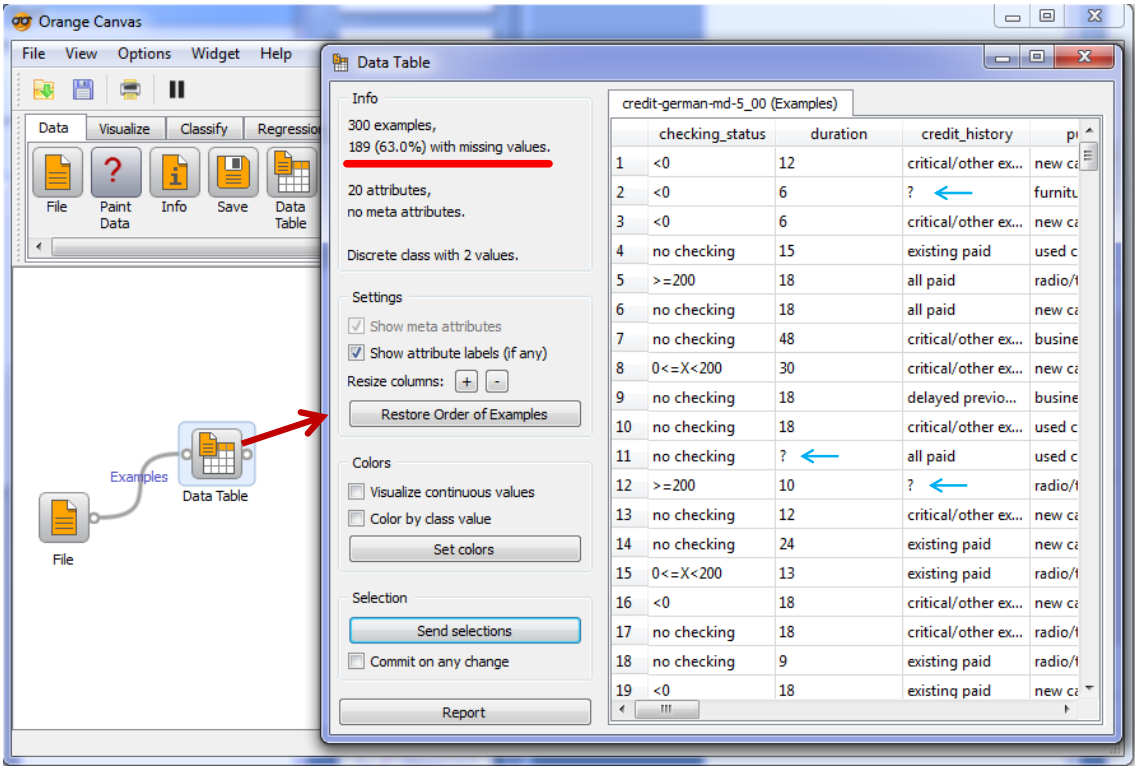
4.1 Traitement des données manquantes avec Orange

4.1.1 Définition du schéma d'analyse.

Imputation des données manquantes lors de l'apprentissage. Au lancement du logiciel, nous retrouvons l'espace de travail qui permet de définir des schémas d'analyse. Nous plaçons le composant FILE (onglet DATA) que nous paramétrons pour charger le fichier CREDIT-GERMAN-MD-5_00.TXT (avec 5.00% de valeurs manquantes). Nous fixons l'option DON'T KNOW à « ? » pour spécifier le code représentant les valeurs manquantes.



Première vérification à faire, nous visualisons les données avec l'outil DATA TABLE (onglet DATA).

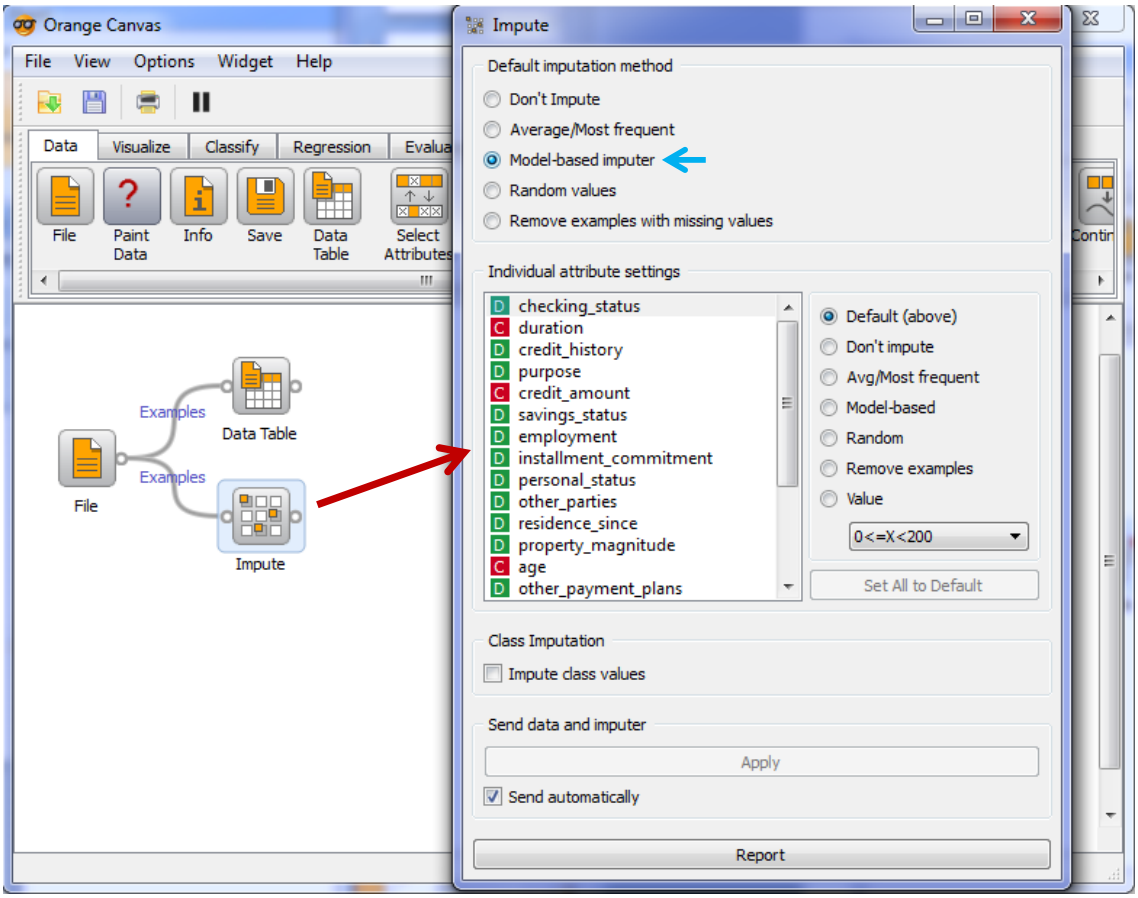


Effectivement, Orange a réalisé correctement le chargement. Il nous annonce que 189 observations contiennent ***au moins*** une valeur manquante (189 + 111 obs. complètes = 300, le compte y est).

Nous plaçons l'outil IMPUTE (onglet DATA) dans le schéma. Nous actionnons le menu OPEN pour accéder à la boîte de paramétrage. Plusieurs options sont possibles pour traiter globalement l'ensemble des variables. Le fichier d'aide du logiciel fournit les indications suivantes :

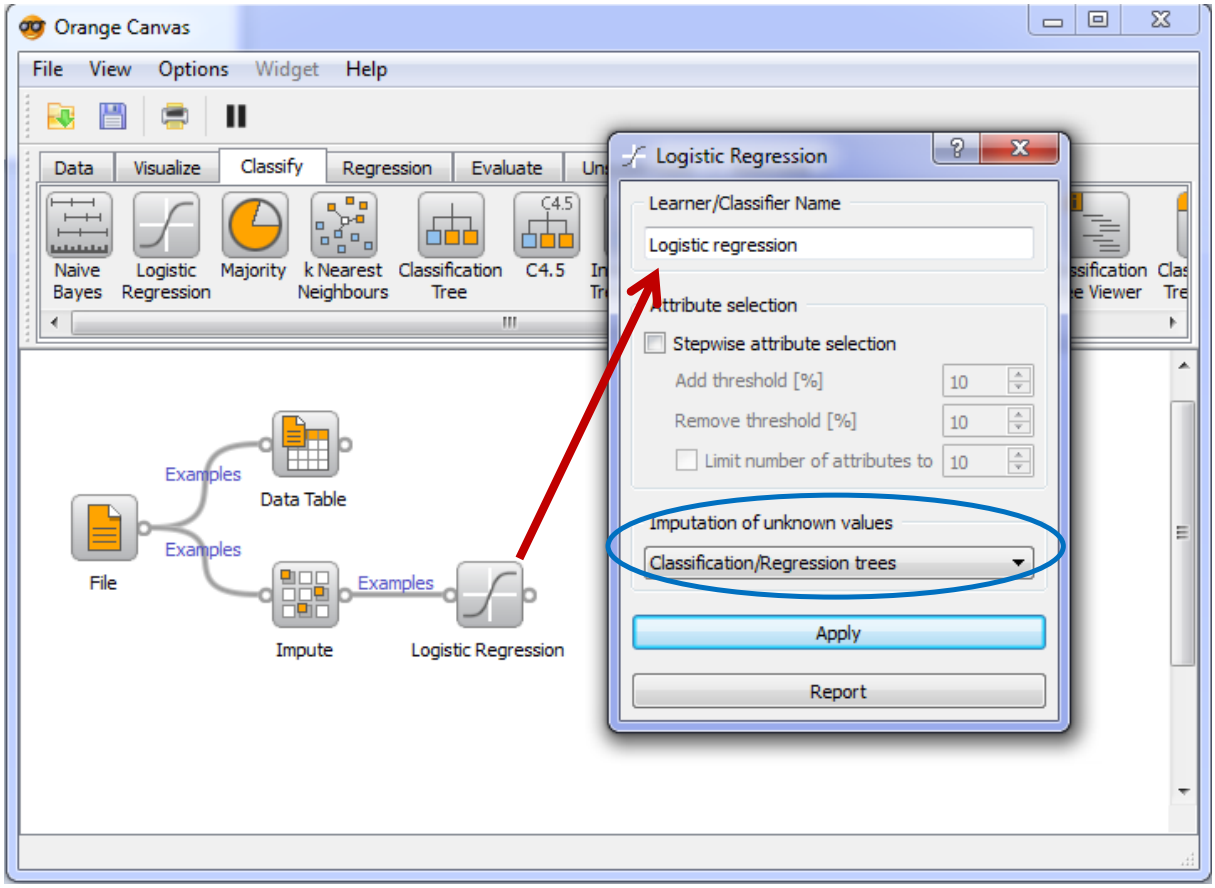
- DON'T IMPUTE. Orange laisse le soin à l'algorithme d'apprentissage (LOGISTIC REGRESSION) en aval de réaliser à la volée l'imputation des valeurs manquantes. Nous y reviendrons plus loin.
- AVERAGE /MOST FREQUENT. C'est l'imputation univariée que nous avons mis en œuvre à l'aide de R dans la section précédente.
- MODEL-BASED IMPUTER. Orange utilise l'algorithme du plus proche voisin pour prédire la valeur manquante d'une variable à partir des autres variables prédictives de la base. Si l'idée semble séduisante, nous nous appuyons sur les liens entre les variables pour produire la meilleure imputation possible, le temps de calcul peut être rapidement prohibitif sur les grandes bases. On parle aussi d'approche « hot deck » dans la littérature.
- RANDOM VALUES. Orange essaie de reconstituer la distribution des données pour chaque variable. Il utilise alors une valeur extraite, en respectant la distribution approximée, de la plage des valeurs possibles.
- REMOVE EXAMPLES WITH MISSING VALUES. Il s'agit de la stratégie « listwise deletion ».

Nous choisissons l'option MODEL BASED IMPUTER.



N.B. : Il est possible de définir une stratégie spécifique pour chaque variable. Nous pouvons même définir la valeur d'imputation de manière ad hoc (INDIVIDUAL ATTRIBUTE SETTNGS = VALUE).

Il nous faut maintenant placer le composant LOGISTIC REGRESSION (onglet CLASSIFY). Nous ne procédons pas à une sélection de variables.

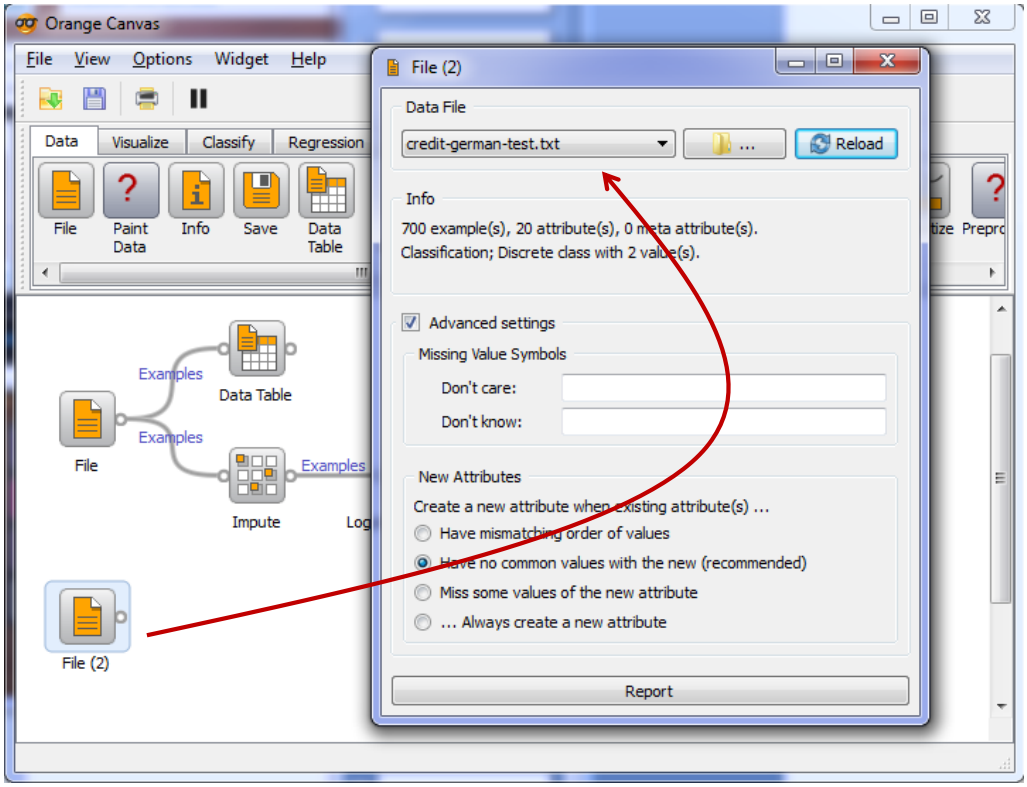


Notons que le composant « régression logistique » dispose d'un mécanisme interne d'imputation des valeurs manquantes (IMPUTATION OF UNKNOWN VALUES). Plusieurs solutions sont proposées : une très classique imputation univariée (moyenne, minimum, maximum) ; ou une imputation multivariée où l'on cherche à prédire les valeurs manquantes d'une variable prédictive à partir des autres variables présentes dans la base. Dans ce cas, un arbre de décision (resp. un arbre de régression) est utilisé pour imputer les valeurs d'une variable qualitative (resp. quantitative). Cette fonctionnalité est activée si nous choisissons l'option DON'T IMPUTE lors du paramétrage du composant IMPUTE.

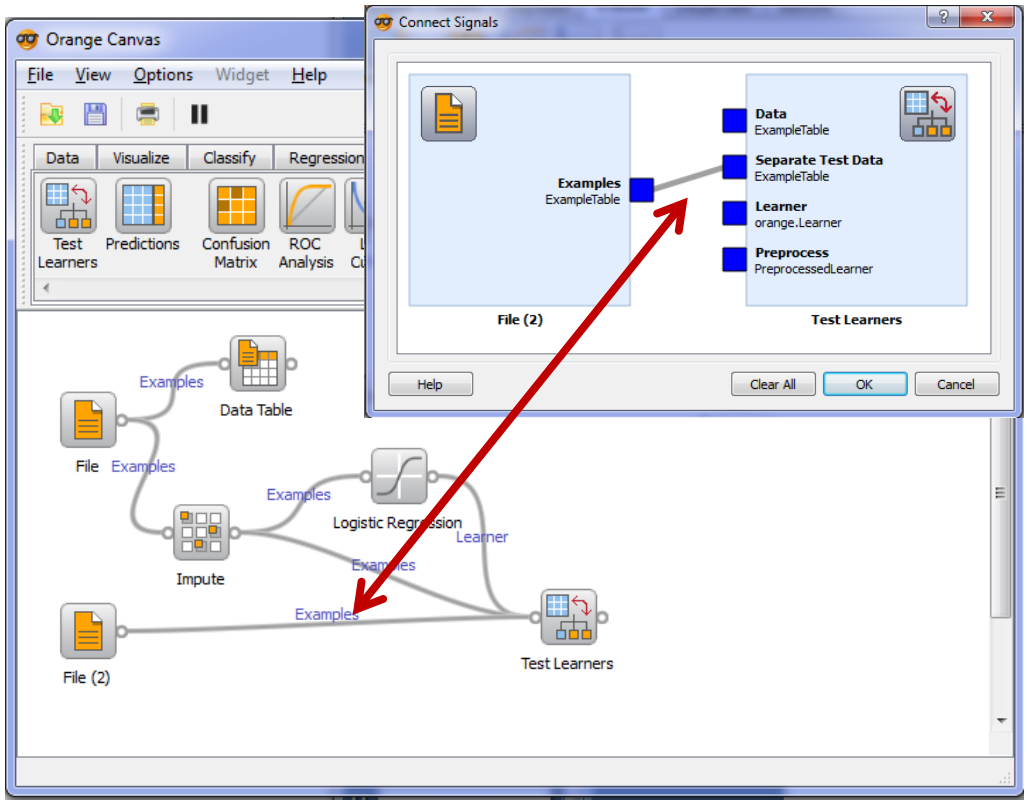
Nous ne l'effectuons pas dans ce didacticiel mais il est possible de visualiser le modèle en utilisant l'outil NOMOGRAM par exemple. Nous détaillons son utilisation et la lecture des résultats dans notre didacticiel consacré au « classifieur bayésien naïf »[9].

Evaluation du modèle sur l'échantillon test. Le modèle est directement construit lorsque nous fermons la boîte de paramétrage. Il ne nous reste plus qu'à évaluer ses performances sur l'échantillon test. Nous insérons une seconde fois le composant FILE. Nous sélectionnons le fichier CREDIT-GERMAN-TEST.TXT.

[9] http://tutoriels-data-mining.blogspot.com/2010/03/le-classifieur-bayesien-naif-revisite.html



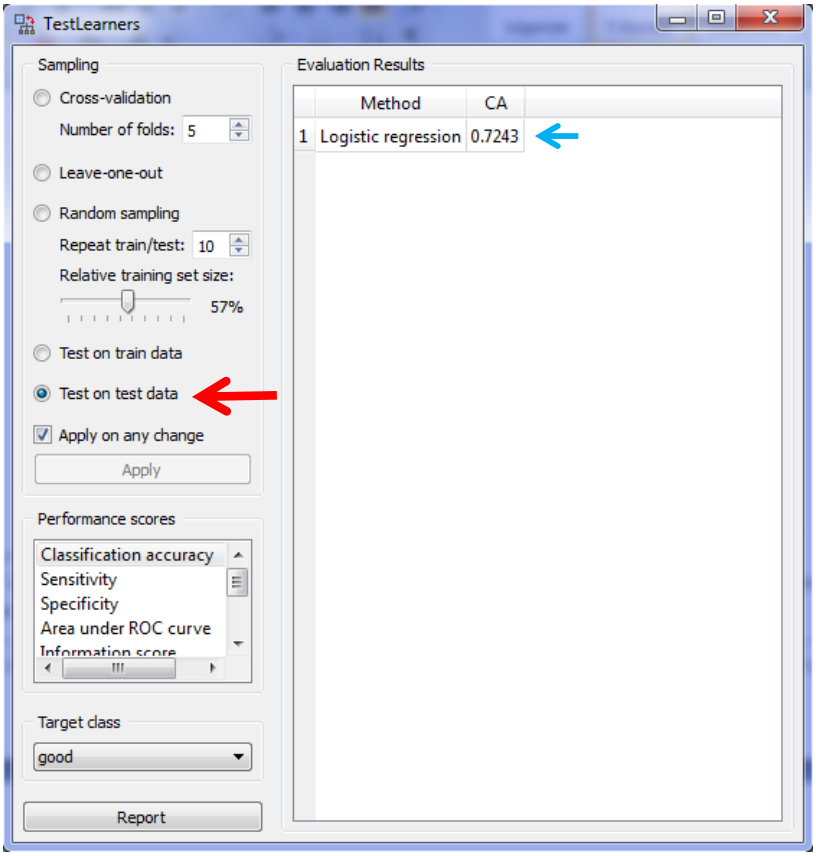
Puis nous plaçons le composant TEST LEARNER (onglet EVALUATE). Nous lui connectons successivement : la régression logistique [LOGISTIC REGRESSION], le fichier d'apprentissage imputé [IMPUTE][10] et le fichier test [FILE (2)].



[10] Je ne comprends pas très bien pourquoi il faut connecter de nouveau ce composant au TEST LEARNER puisqu'il est déjà relié à la régression logistique. Mais, l'évaluation n'est pas possible sans cette opération.

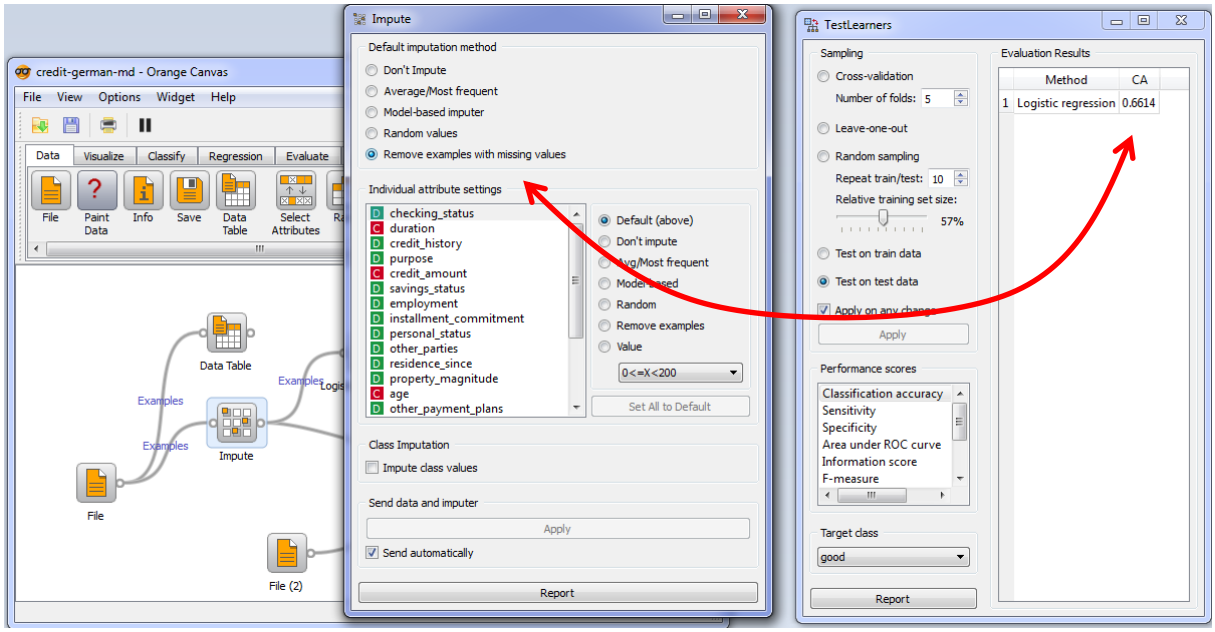
Attention pour cette dernière connexion, il faut spécifier explicitement (en double-cliquant sur le lien) qu'il s'agit de l'échantillon test (SEPARATE TEST DATA).

Il ne nous reste plus qu'à cliquer sur le menu OPEN de TEST LEARNER. Dans la fenêtre de visualisation qui apparaît, nous indiquons d'une part que l'évaluation doit être réalisée à l'aide de l'échantillon test (SAMPLING – TEST ON TEST DATA), d'autre part que seul le taux de bon classement (CLASSIFICATION ACCURACY) nous intéresse dans notre étude. Nous obtenons ainsi un taux en test de 72.43%.



4.1.2 Comparaison des techniques d'imputation

Voyons ce qu'il en est des positions relatives des techniques. Les résultats sont directement comparables parce que nous utilisons exactement les mêmes échantillons d'apprentissage et de test. Orange propose une fonctionnalité très pratique. Il suffit de laisser ouvert la fenêtre TEST LEARNERS et de modifier la technique d'imputation dans la fenêtre de IMPUTE pour voir les résultats directement mis à jour (il faut que l'option SEND AUTOMATICALLY soit cochée dans la boîte de dialogue IMPUTE).



Nous obtenons le tableau suivant à l'issue de l'expérimentation :

Approach	Class. Accuracy
Don't impute (Internal method of Logistic Reg.)	0,7357
Average / Most Frequent	0,7271
Model-based imputer	0,7243
Random Values	0,7357
Remove Examples with missing values	0,6614

Mise à part la suppression des lignes, toutes les méthodes se valent. Très étrangement, dans certains cas le taux de bon classement (73,57%) est meilleur que celui de glm(.) de R construit à partir du fichier de données complet (73%), sans observations manquantes.

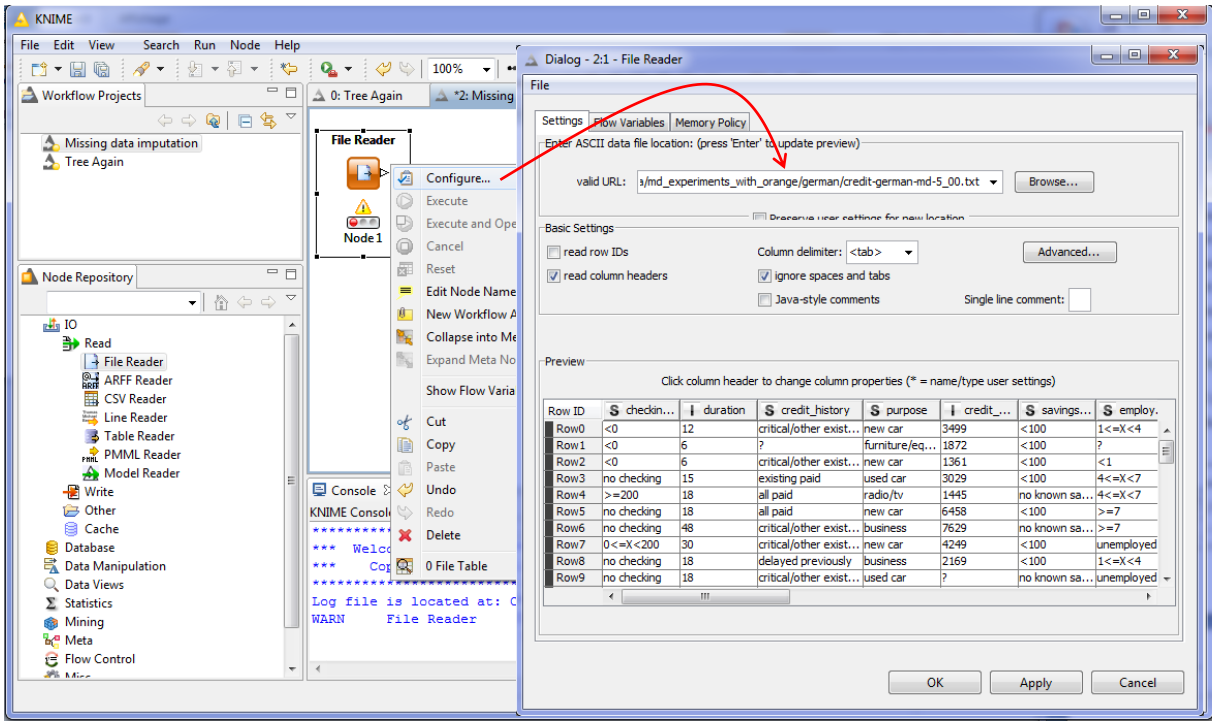
A mon sens il ne faut pas trop s'attarder sur le résultat brut. D'une part parce que nous ne savons pas comment fonctionne exactement la régression logistique d'Orange[11]. D'autre part, parce que nous nous situons dans le cadre très spécifique d'une expérimentation artificielle. Après coup on peut se dire que dans notre configuration MCAR, très extrême il faut le reconnaître, les valeurs étant retirées complètement aléatoirement de l'échantillon d'apprentissage, toute technique d'imputation un tant soit peu raisonnable s'avère viable. Finalement, le grand perdant de ce tutoriel est la suppression de lignes. Elle est très pénalisante dès que la proportion de valeurs manquantes augmente sensiblement (à partir de 5% sur notre fichier).

4.2 Traitement des données manquantes avec KNIME

KNIME propose le composant MISSING VALUES pour le traitement de données manquantes. Nous reproduisons dans cette section le schéma d'analyse que nous avons défini sous Orange.

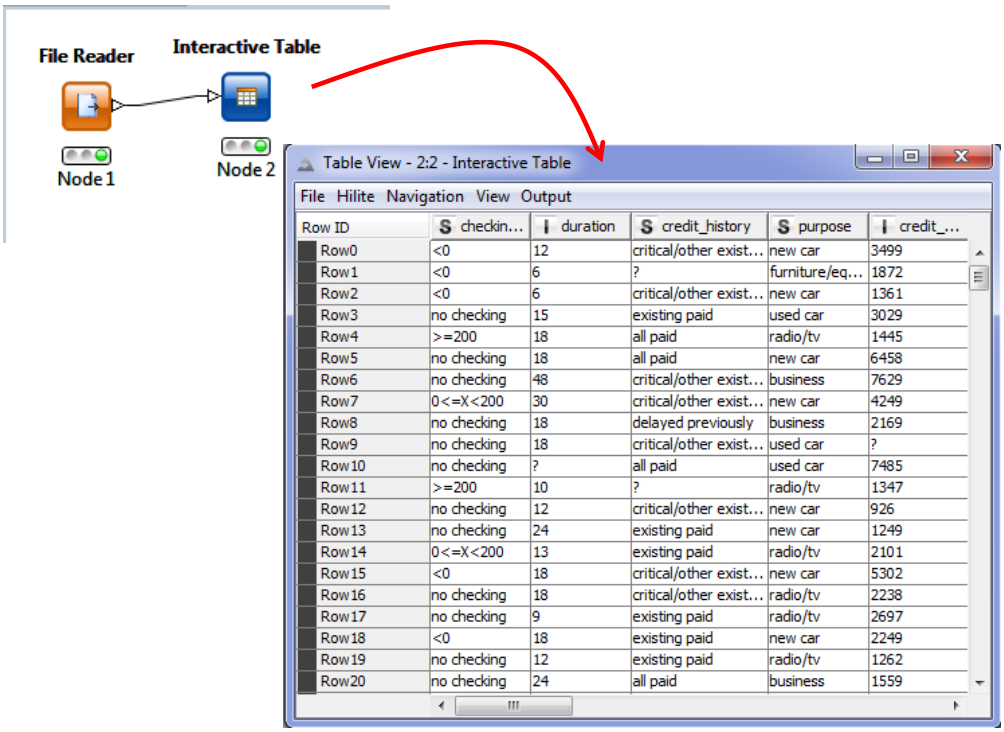
[11] Curieusement, le taux de succès en test du modèle élaboré à partir du fichier d'apprentissage complet est de 72.14%, j'avoue que ce résultat assez singulier m'a laissé un peu perplexe…

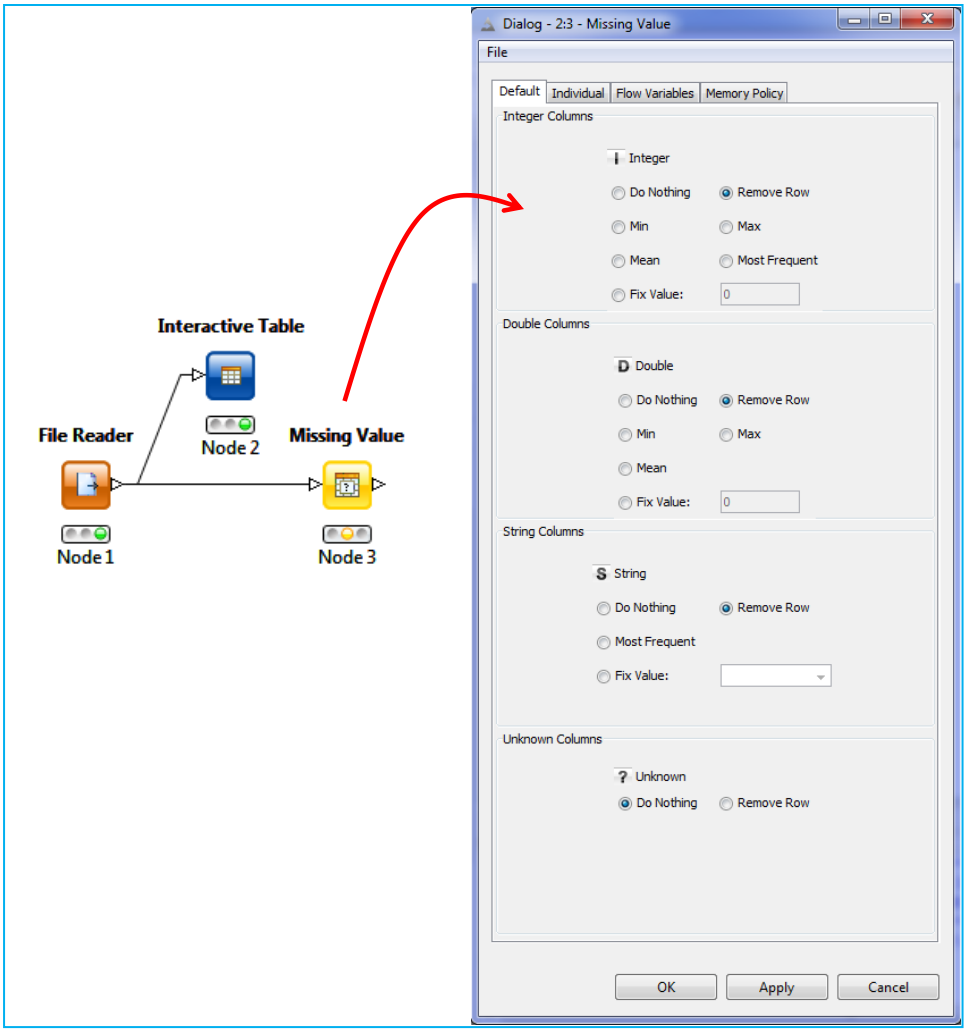
Construction du modèle sur l'échantillon d'apprentissage traité par suppression de lignes. Nous importons le fichier CREDIT-GERMAN-MD-5_00.TXT à l'aide du composant FILE READER (branche IO / READ du dépôt des méthodes). Les valeurs manquantes sont symbolisées par le caractère « ? ».



Nous lançons l'importation effective du fichier en cliquant sur le menu EXECUTE.

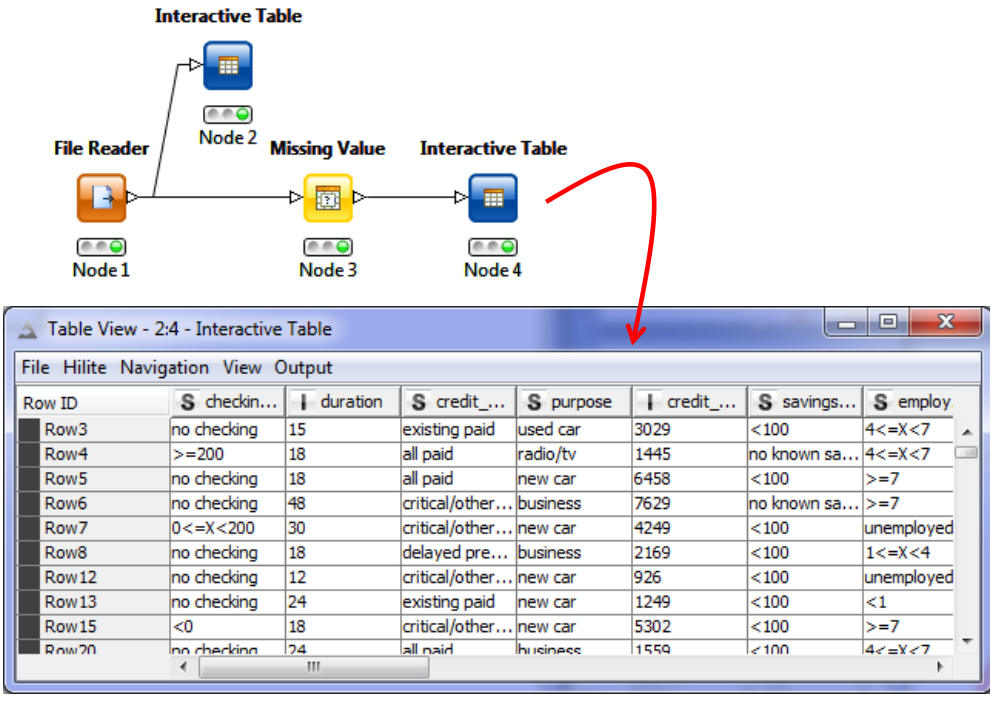
Pour vérifier l'importation, nous visualisons les données à l'aide du composant INTERACTIVE TABLE (DATA VIEWS). Nous lui connections le composant précédent et nous cliquons sur le menu EXECUTE AND OPEN VIEWS.





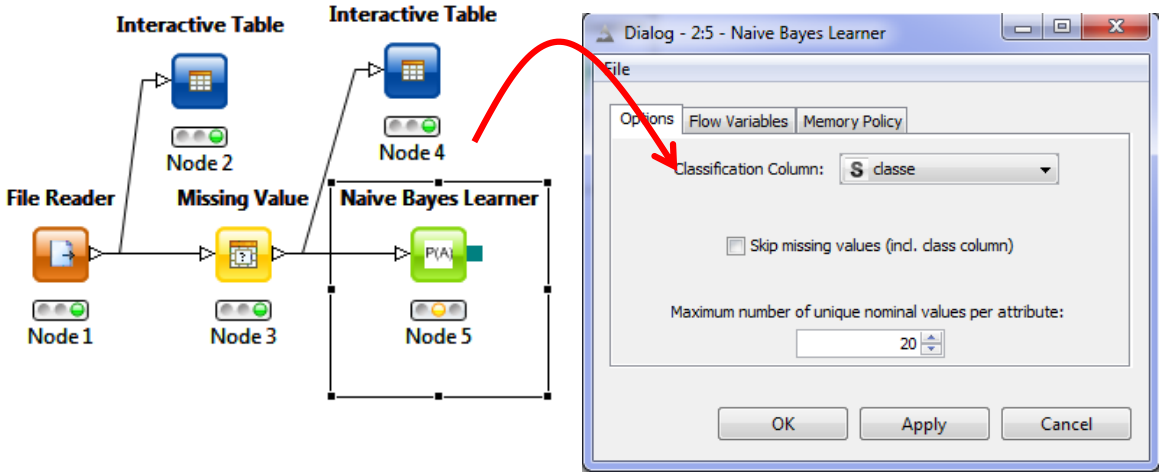
Nous plaçons maintenant le composant MISSING VALUE (DATA MANIPULATION / COLUMN / TRANSFORM). Nous le paramétrons (menu CONFIGURE) de la manière à supprimer les lignes (REMOVE ROW), tant pour les variables numériques (entières - INTEGER ou réelles DOUBLE) que pour les variables qualitatives (STRING). Il s'agit de la stratégie « listwise deletion ».

De nouveau, pour voir l'état de l'ensemble de données, nous utilisons le composant INTERACTIVE TABLE. Nous constatons que les lignes ROW0, ROW1, ROW2, ROW9, ROW10, etc. ont été exclues de l'ensemble d'apprentissage.

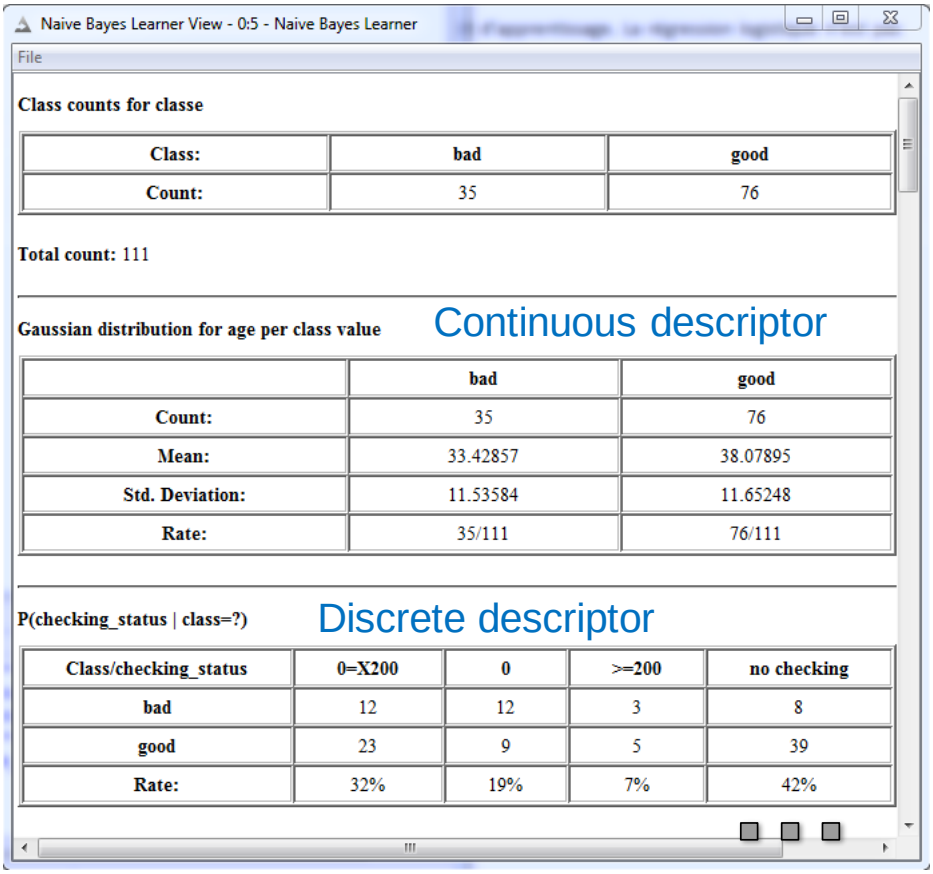


Row ID	checkin...	duration	credit_...	purpose	credit_...	savings...	employ
Row3	no checking	15	existing paid	used car	3029	<100	4<=X<7
Row4	>=200	18	all paid	radio/tv	1445	no known sa...	4<=X<7
Row5	no checking	18	all paid	new car	6458	<100	>=7
Row6	no checking	48	critical/other...	business	7629	no known sa...	>=7
Row7	0<=X<200	30	critical/other...	new car	4249	<100	unemployed
Row8	no checking	18	delayed pre...	business	2169	<100	1<=X<4
Row12	no checking	12	critical/other...	new car	926	<100	unemployed
Row13	no checking	24	existing paid	new car	1249	<100	<1
Row15	<0	18	critical/other...	new car	5302	<100	>=7
Row20	no checking	24	all paid	business	1559	<100	4<=X<7

Il reste maintenant à brancher le composant d'apprentissage. La régression logistique n'est pas disponible dans Knime (**04/12/2011** : *En réalité, elle y est. Voir page 21*). Nous lui substituons le modèle bayésien naïf NAIVE BAYES (modèle d'indépendance conditionnelle, MINING / BAYES)[12]. Il induit aussi un séparateur linéaire dans l'espace de représentation. Les performances en prédiction devraient être similaires. Nous désignons la variable à prédire CLASSE.



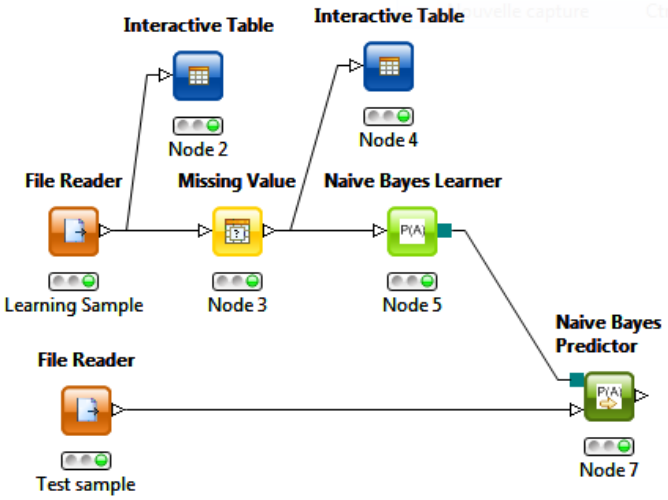
Nous actionnons le menu EXECUTE AND OPEN VIEWS. Knime fournit les distributions conditionnelles (moyenne et variances conditionnelles pour les variables quantitatives, avec l'hypothèse d'une distribution gaussienne ; le tableau de contingence pour les variables qualitatives).



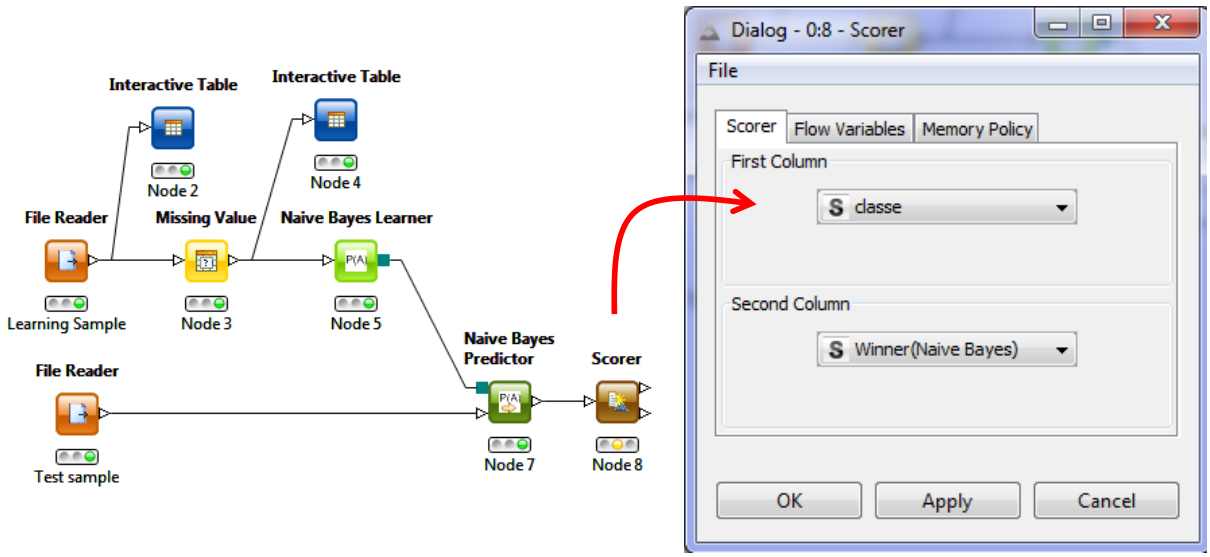
[12] http://eric.univ-lyon2.fr/~ricco/cours/slides/naive_bayes_classifier.pdf

Evaluation sur l'échantillon test. Nous devons maintenant appliquer ce modèle sur l'échantillon test (credit-german-test.txt). Nous chargeons ce dernier à l'aide d'un second FILE READER.

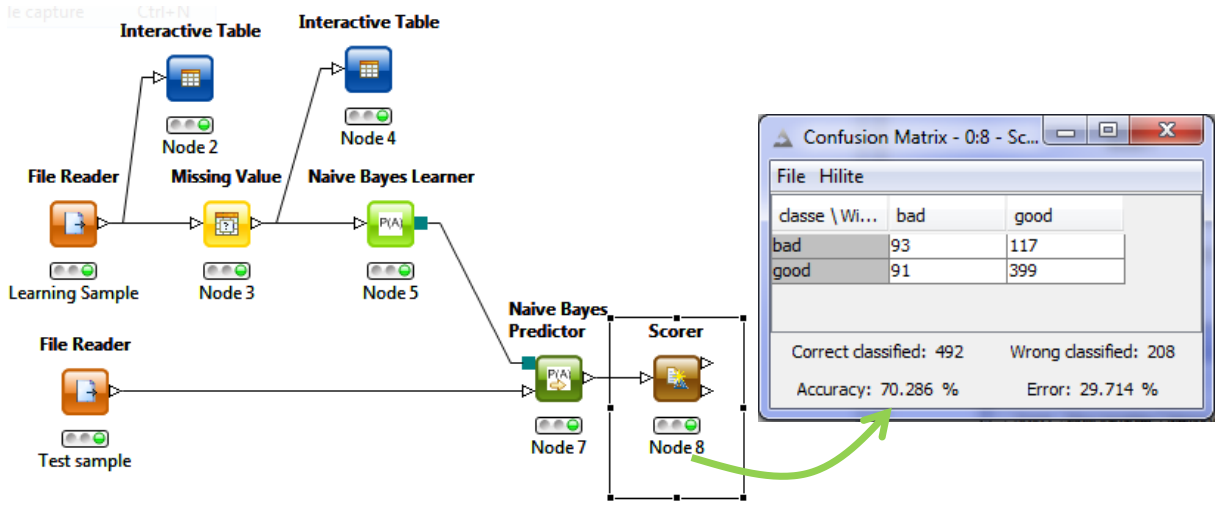
Puis nous calculons les prédictions à l'aide du composant NAIVE BAYES PREDICTOR auquel nous relions : le modèle appris (NAIVE BAYES LEARNER) ; les données sur lesquelles doivent être calculées les prédictions (FILE READER – échantillon test).



Passons à la matrice de confusion. Nous utilisons l'outil SCORER (MINING / SCORING / SCORER). Nous le paramétrons (menu CONFIGURE) de manière à opposer la classe observée (FIRST ROW) et la classe prédite par le modèle [SECOND COLUMN – WINNER (NAIVE BAYES)].



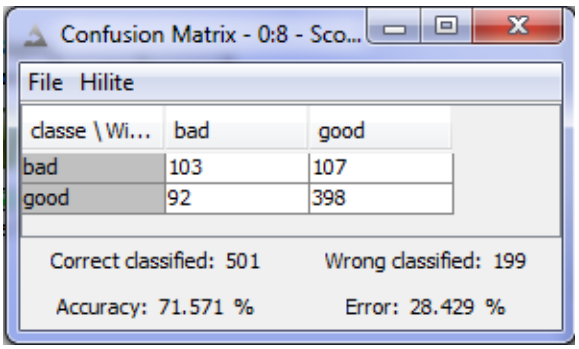
Nous actionnons le menu EXECUTE AND OPEN VIEWS. Nous constatons un taux de succès de 70.286%.



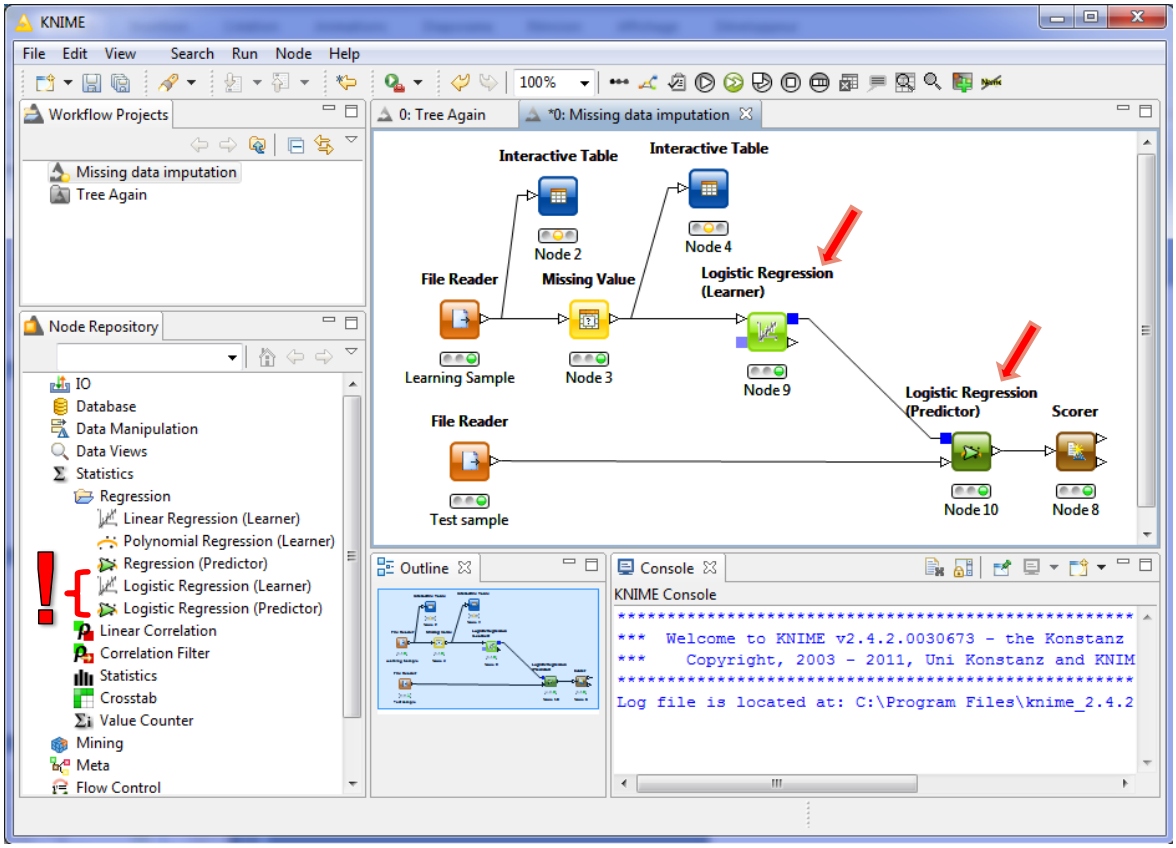
Le classifieur bayésien naïf souffre moins de la suppression de lignes que la régression logistique. En effet, le modèle élaboré à partir des données complètes (sans valeurs manquantes) propose un taux d'erreur de 72%. Cette caractéristique est vraisemblablement due au faible nombre de paramètres estimés dans le modèle d'indépendance conditionnelle. Sa dépendance aux données d'apprentissage – sa variance - est plus faible. Il est (un peu) moins sensible à la réduction de l'échantillon.

Remarque : Il y a des limites à tout bien sûr. Lorsque nous passons à 10% de valeurs manquantes, l'apprentissage reste encore possible contrairement à la régression logistique. Mais le très faible nombre d'observations exploitées (40 lignes sur les 300 initialement disponibles) se répercute sur le taux de succès : 63,429%.

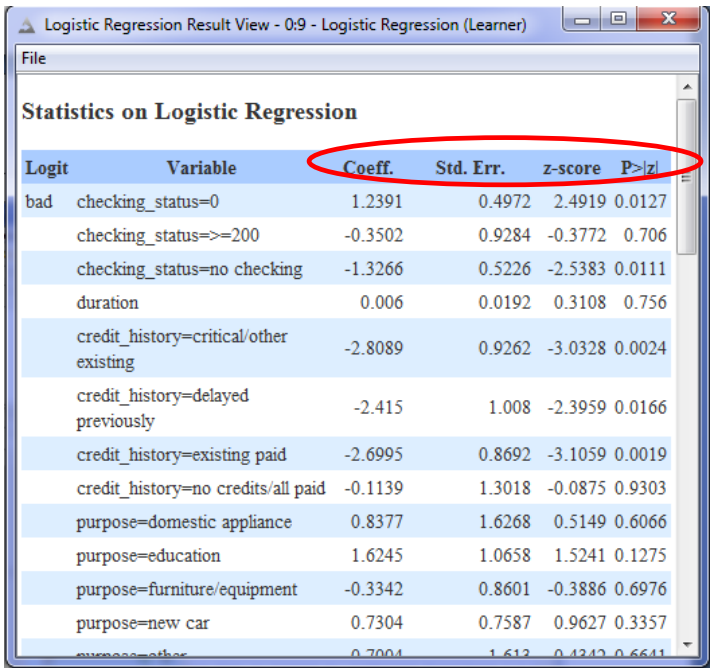
Imputation univariée. Lorsque nous sélectionnons l'imputation univariée (MEAN pour les colonnes INTEGER et DOUBLE / MOST FREQUENT pour les STRING) dans le composant IMPUTE, nous obtenons un taux de succès de 71.571% sur l'échantillon test. Encore une fois, dans un cadre très similaire (classifieur bayésien naïf à la place de la régression logistique), l'imputation univariée surpasse très facilement la suppression de lignes dans le cadre des données manquantes MCAR.



La régression logistique sous Knime. **Correction, 04/12/2011**. Très peu de temps après la publication de ce tutoriel, Loïc Lucel – qu'il en soit très chaleureusement remercié – m'a indiqué que la régression logistique était en réalité présente dans le logiciel Knime. Il fallait chercher dans la branche STATISTICS / REGRESSION du « Node Repository ». J'ai donc reproduit l'analyse en substituant les outils de la régression logistique (LEANER et PREDICTOR) à ceux du classifieur bayésien naïf. Nous avons le diagramme suivant :



Les variables prédictives qualitatives sont automatiquement recodées. Les sorties de la régression logistique respectent les standards du domaine[13]. Nous disposons du coefficient calculé, de l'estimation de son écart-type, de la statistique de test de significativité et de la p-value associée.



Logistic Regression Result View - 0:9 - Logistic Regression (Learner)

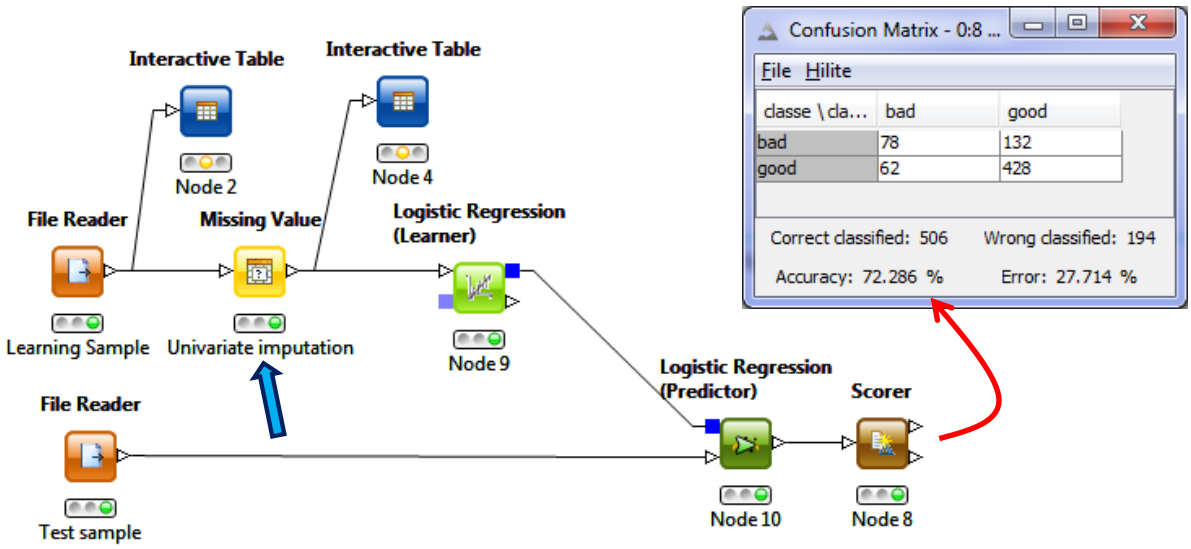
File

Statistics on Logistic Regression

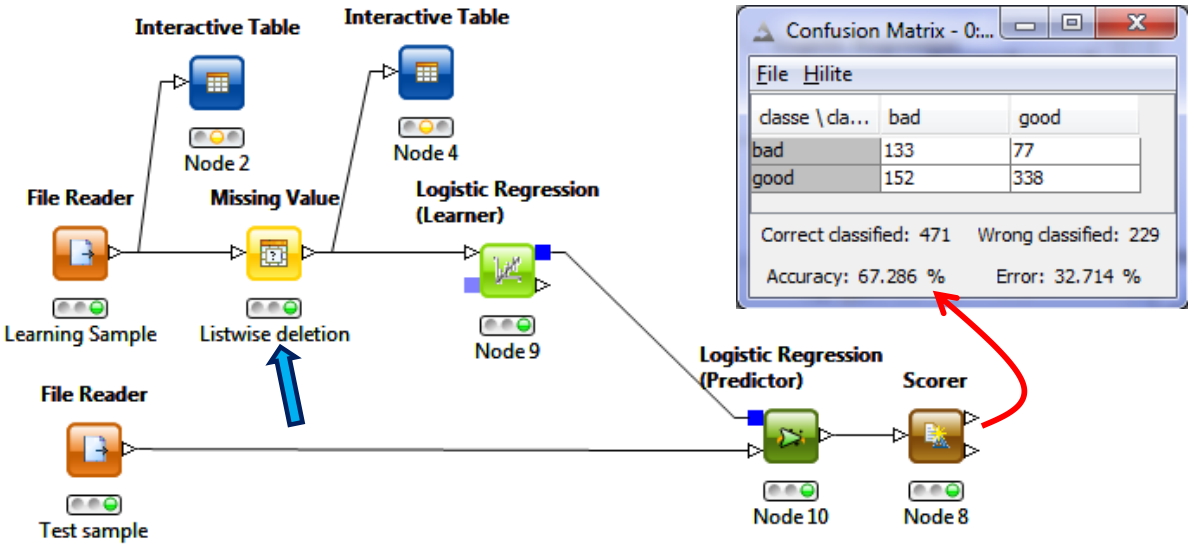
Logit	Variable	Coeff.	Std. Err.	z-score	P>|z|
bad	checking_status=0	1.2391	0.4972	2.4919	0.0127
	checking_status=>=200	-0.3502	0.9284	-0.3772	0.706
	checking_status=no checking	-1.3266	0.5226	-2.5383	0.0111
	duration	0.006	0.0192	0.3108	0.756
	credit_history=critical/other existing	-2.8089	0.9262	-3.0328	0.0024
	credit_history=delayed previously	-2.415	1.008	-2.3959	0.0166
	credit_history=existing paid	-2.6995	0.8692	-3.1059	0.0019
	credit_history=no credits/all paid	-0.1139	1.3018	-0.0875	0.9303
	purpose=domestic appliance	0.8377	1.6268	0.5149	0.6066
	purpose=education	1.6245	1.0658	1.5241	0.1275
	purpose=furniture/equipment	-0.3342	0.8601	-0.3886	0.6976
	purpose=new car	0.7304	0.7587	0.9627	0.3357

[13] Décidément, j'apprécie beaucoup ce logiciel. Tout comme Orange, il sait résister à la tentation de la course aux armements qui consiste à programmer tout un tas de techniques d'apprentissage dont on a du mal parfois à en saisir la teneur réelle. Et les outils, lorsqu'ils sont intégrés dans le logiciel, sont souvent d'excellente facture.

Nous avons exécuté le « workflow » avec la base d'apprentissage comportant 5% de données manquantes. En utilisant **l'imputation univariée**, nous obtenons un taux de succès de **72.286%**.



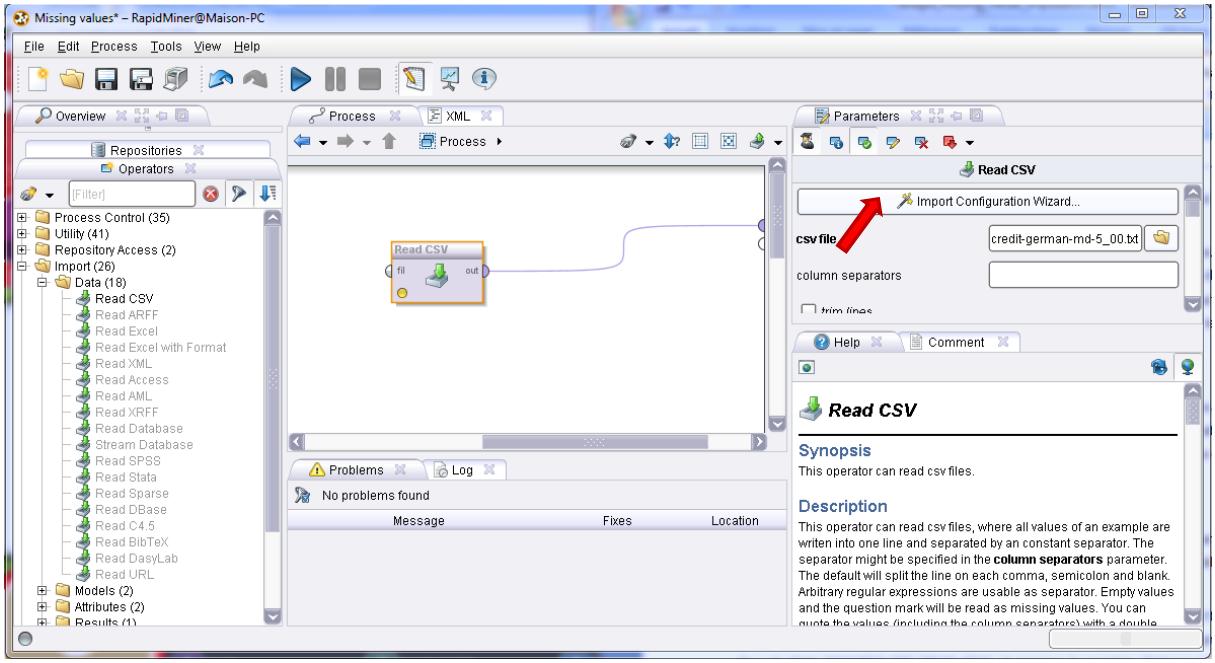
Avec la stratégie « **listwise deletion** » (Remove row), il passe à **67.286%**.



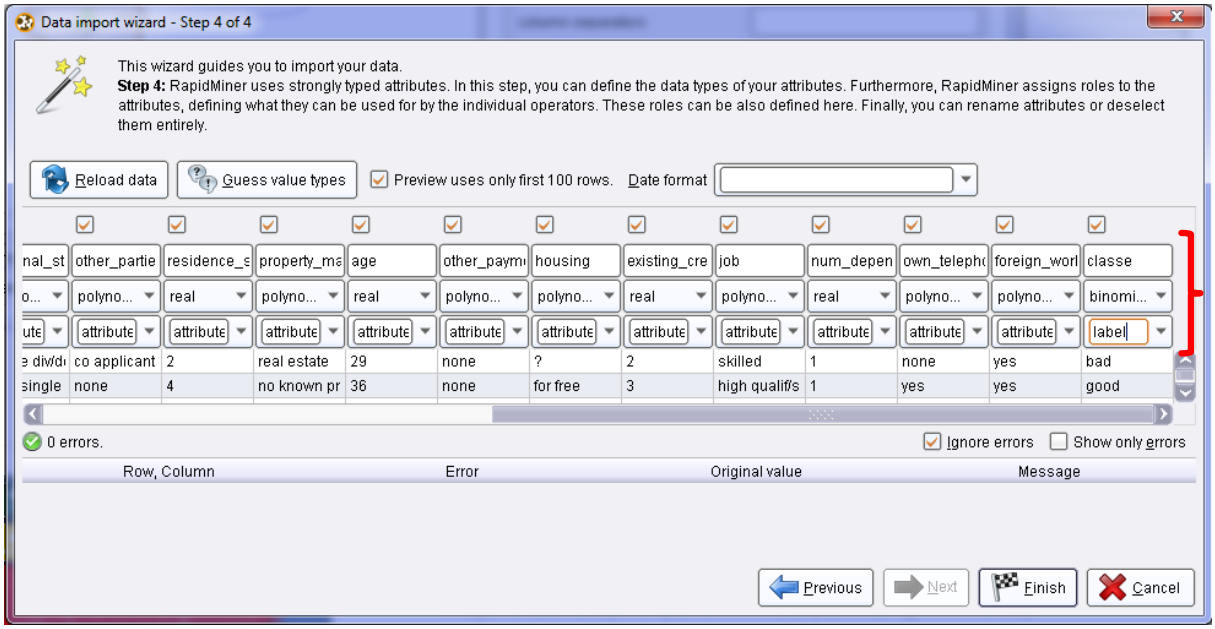
4.3 Traitement des données manquantes avec RapidMiner

Deux outils sont nécessaires pour détecter les valeurs manquantes dans RapidMiner. Par la suite, l'imputation se fait naturellement avec un composant dédié. Outre la suppression de lignes, le logiciel propose l'imputation univariée (moyenne /mode). Les techniques plus sophistiquées ne sont pas disponibles.

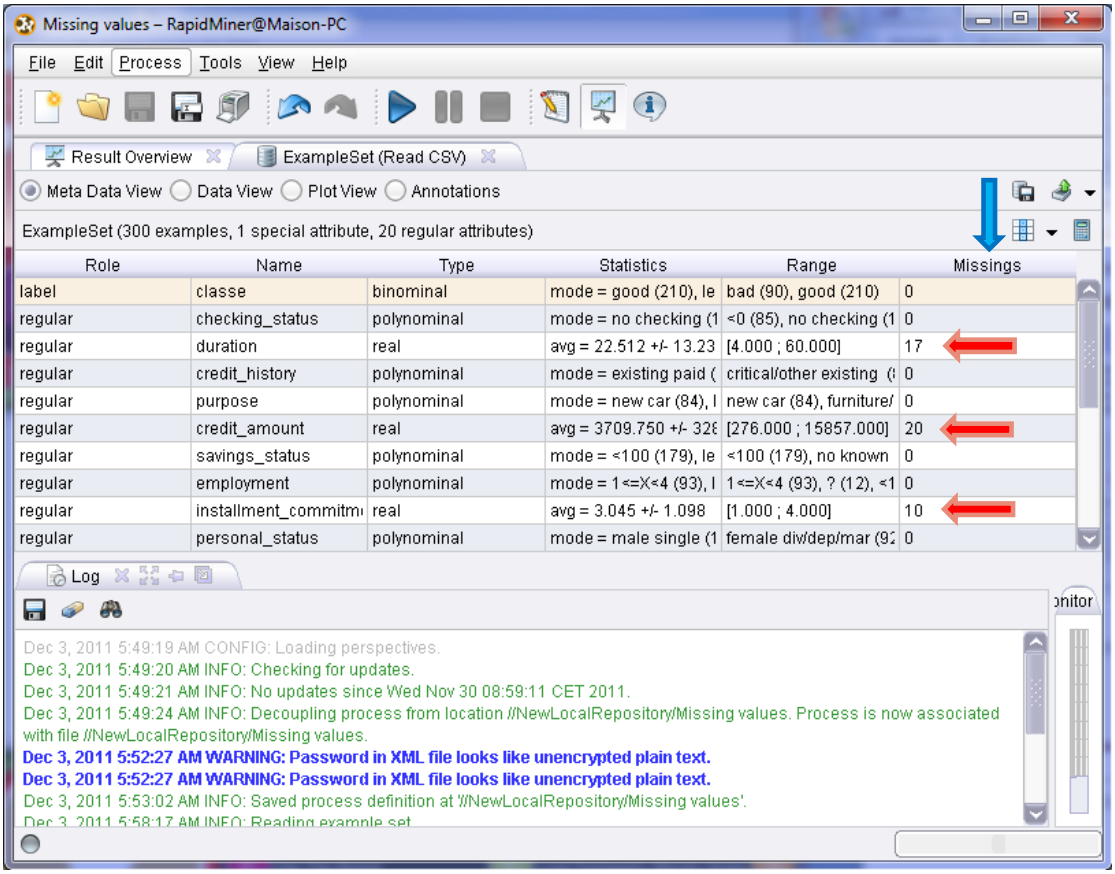
Détection des valeurs manquantes. Après avoir démarré RapidMiner, nous créons un nouveau « Process ». Nous utilisons le composant READ CSV (branche IMPORT / DATA dans OPERATORS) pour importer les données. Son paramétrage peut être compliqué, mieux vaut passer par le wizard (bouton IMPORTATION CONFIGURATION WIZARD) pour paramétrer la lecture du fichier.



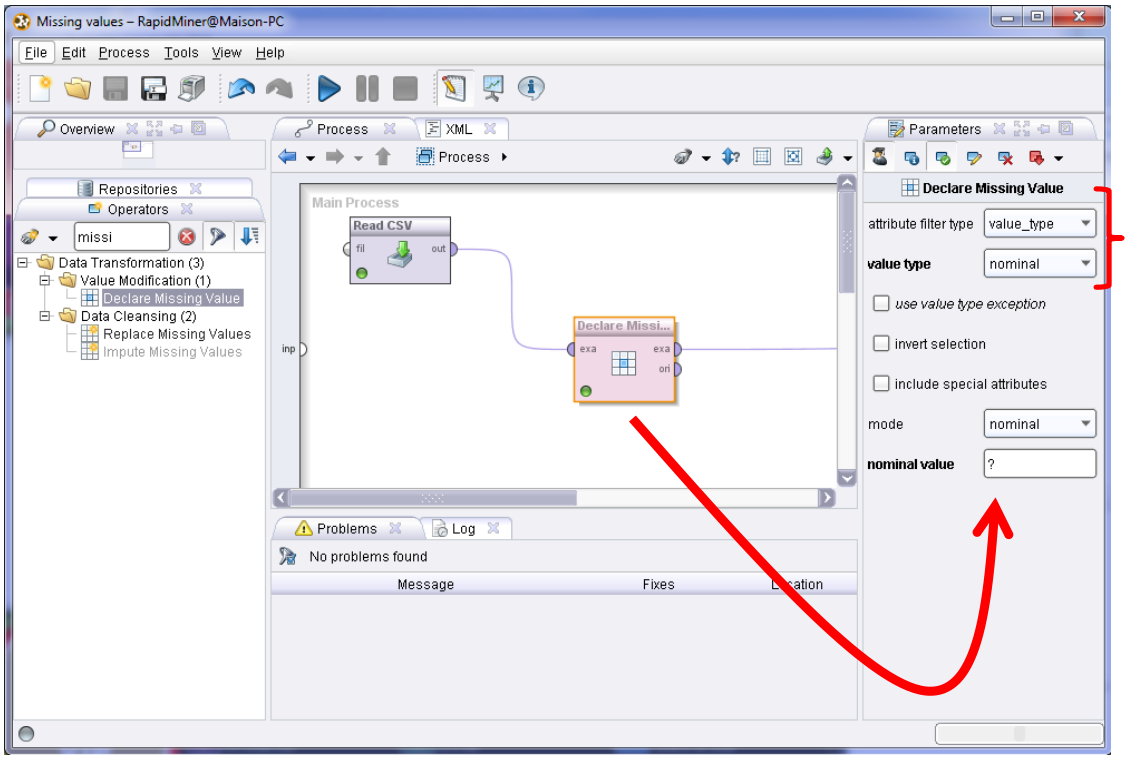
Le typage des variables est important. Dans notre cas, toutes les variables numériques doivent être spécifiées REAL, les autres sont POLYNOMINAL (variables catégorielles à plus de 2 modalités) ou BINOMINAL (variables catégorielles à 2 modalités, c'est le cas de la CLASSE). De plus, nous devons indiquer au logiciel la variable à prédire CLASSE (LABEL).



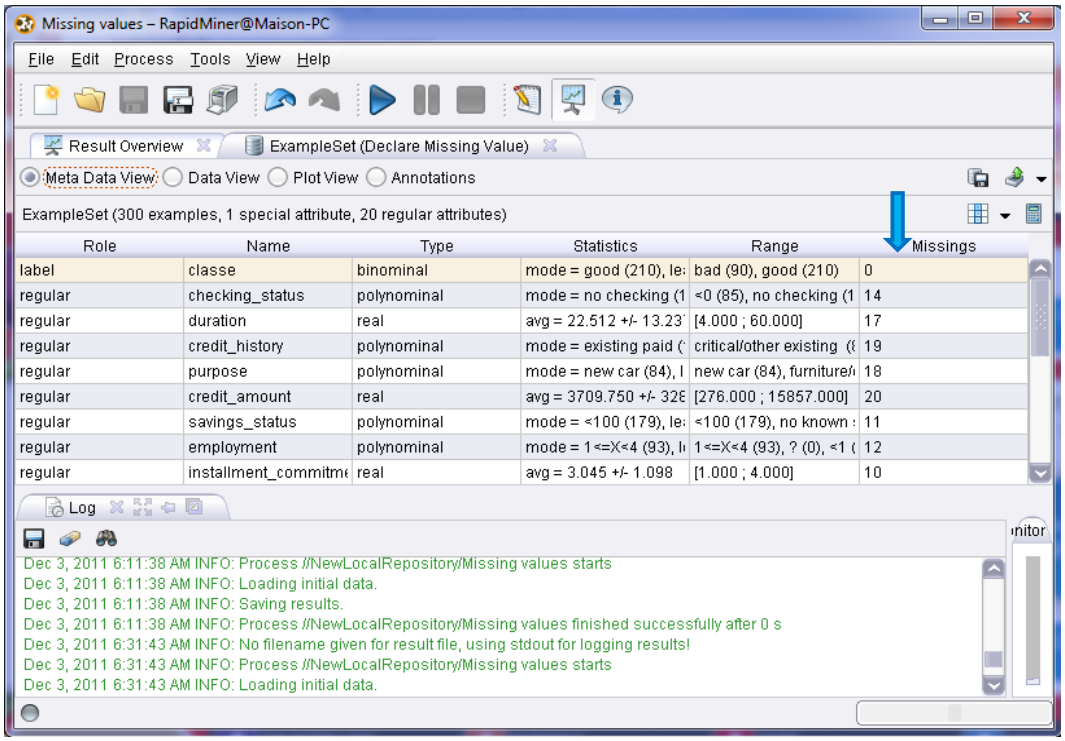
RapidMiner se sert de cette première étape pour détecter les valeurs manquantes. Pour toutes celles indiquées REAL, le caractère « ? » est incohérent avec la définition de la variable. Le logiciel en déduit qu'il s'agit d'une valeur manquante. Nous le constatons en exécutant le PROCESS (menu PROCESS / RUN) (***Remarque*** : il est très vraisemblable que RapidMiner vous demande de sauvegarder la description des traitements au préalable). Dans l'onglet des résultats EXAMPLE SET (READ CSV), il recense le nombre d'observations manquantes pour chaque variable REAL (ex. 17 pour DURATION, 20 pour CREDIT_AMOUNT, etc.).



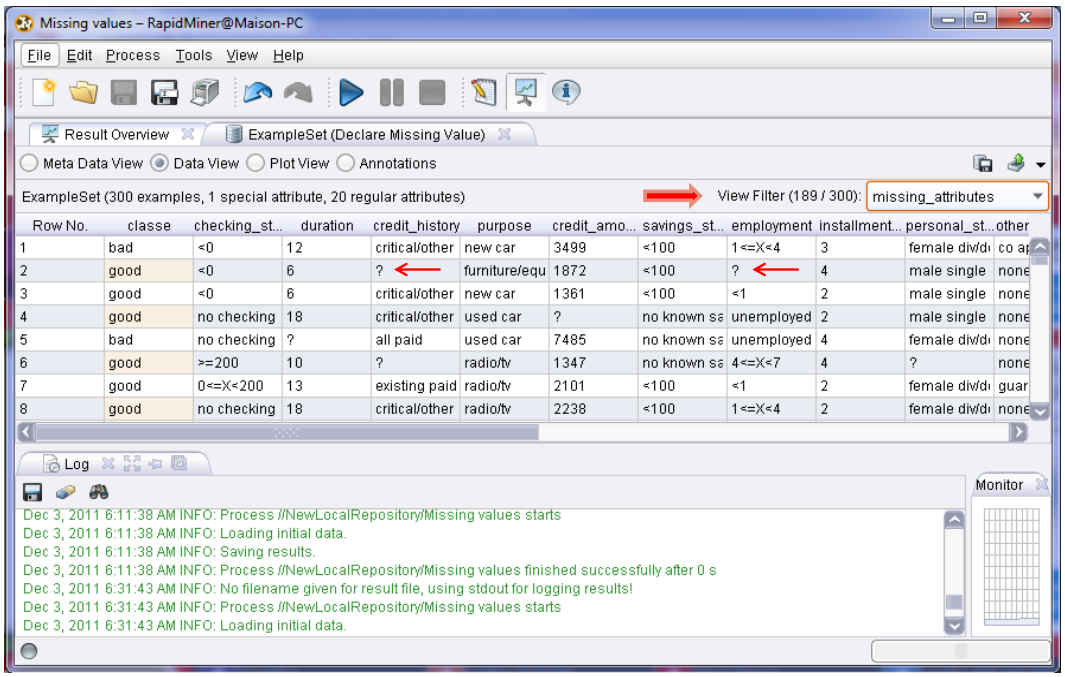
Revenons à la fenêtre de définition des traitements. Nous insérons le composant DECLARE MISSING VALUES (DATA TRANSFORMATION / VALUE MODIFICATION) dans l'espace de travail. Nous lui connectons READ CSV. Nous le paramétrons de manière à ce qu'il ne traite que les variables nominales, en lui indiquant que le code « ? » désigne les valeurs manquantes.



Nous obtenons le nombre d'observations manquantes pour l'ensemble des variables.[14]



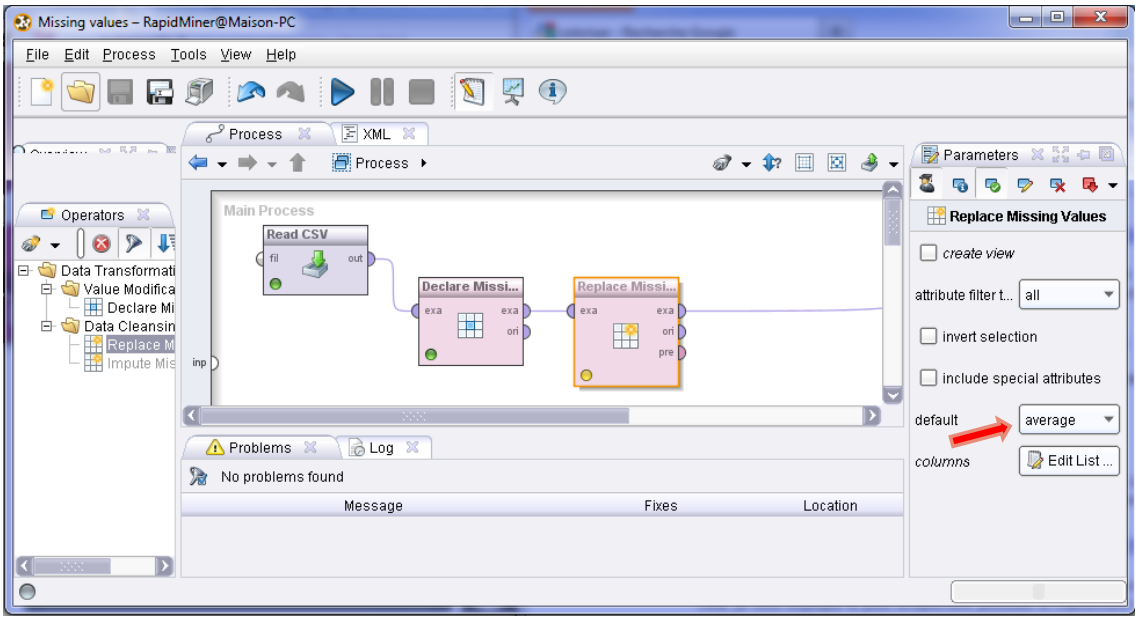
En sélectionnant l'option DATA VIEW dans l'onglet des résultats, nous pouvons visualiser les données en les filtrant de différentes manières. Nous n'affichons que les lignes comportant au moins une valeur manquante (MISSING_ATTRIBUTES). RapidMiner nous indique qu'ils sont au nombre de 189[15].



[14] Ca paraît un peu compliqué tout ça. La difficulté vient de la nécessité de faire comprendre à RapidMiner le code des valeurs manquantes dans un fichier texte. Lorsque les données proviennent d'une base, le processus est nettement simplifié comme le montre ce tutoriel : http://www.youtube.com/watch?v=0IVZmAk0pI4

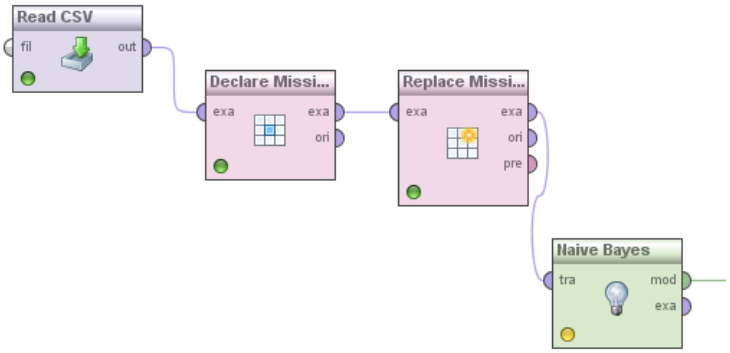
[15] Dans le fichier à 5% de valeurs manquantes, il y a 111 observations complètes dixit le traitement dans R. Ainsi, 111 + 189 = 300 observations, la taille initiale de notre échantillon d'apprentissage.

Imputation et construction du modèle. Nous utilisons REPLACE MISSING VALUES pour l'imputation[16] (DATA TRANSFORMATION / DATA CLEANSING). Nous lui connectons l'opérateur précédent.



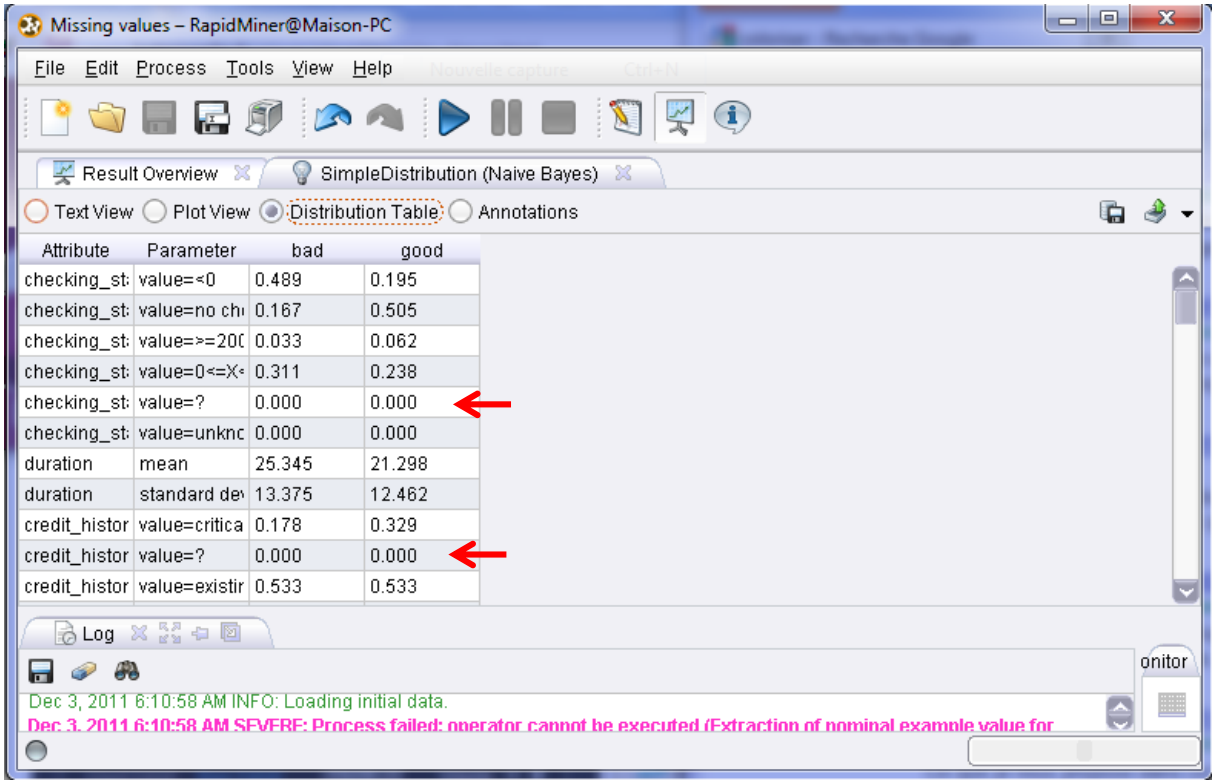
AVERAGE est le type de remplacement proposé par défaut. En réalité, cette option inclus le remplacement par le mode pour les variables nominales.

Nous pouvons maintenant introduire l'outil d'apprentissage supervisé. La régression logistique proposée dans RapidMiner ne correspond pas à la méthode que l'on retrouve usuellement dans les logiciels de statistique. Il m'a paru préférable d'utiliser le classifieur bayésien naïf encore une fois.

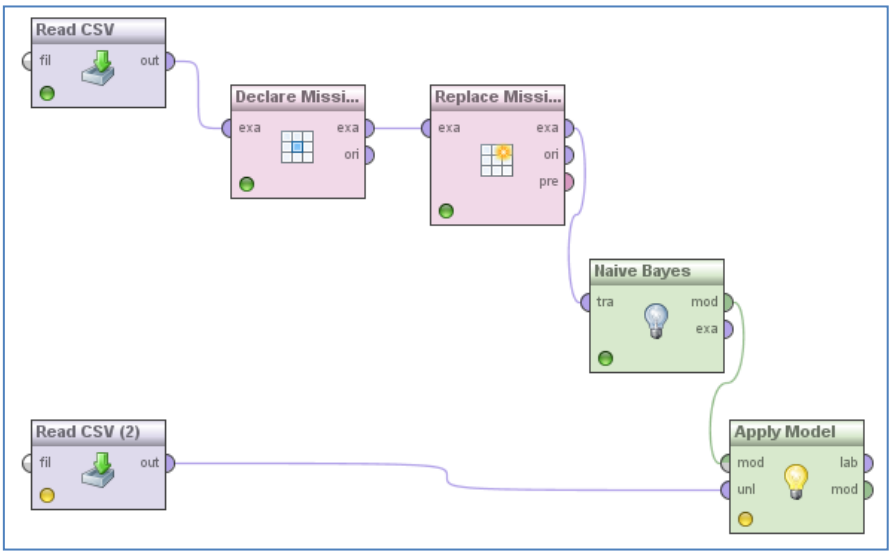


Nous lançons l'exécution. Dans le tableau décrivant les distributions conditionnelles, nous constatons que les modalités « ? » des variables ne correspond à aucune observation maintenant. Les imputations ont bien été effectuées.

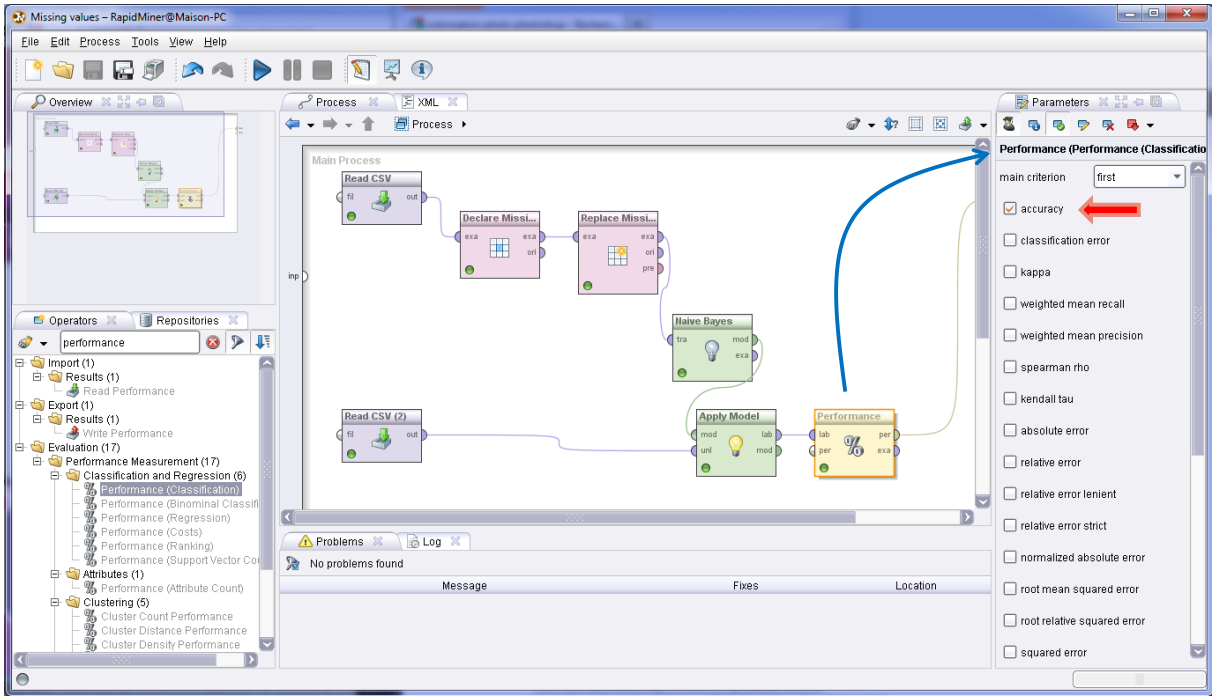
[16] Un outil IMPUTE MISSING VALUES existe. Mais il semble encore non stabilisé. Son utilisation n'est pas recommandée dixit la documentation du logiciel.



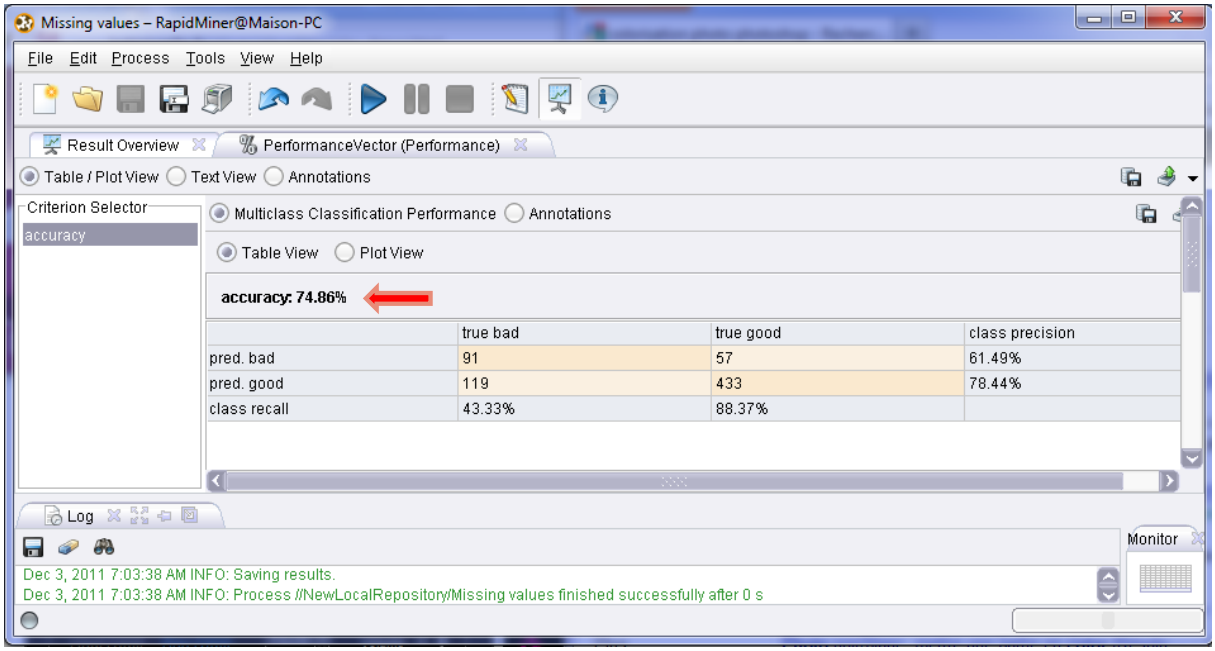
Evaluation sur l'échantillon test. Pour l'évaluation, nous devons appliquer ce modèle sur l'échantillon test et confronter la prédiction avec les valeurs observées de la variable cible. Nous insérons une nouvelle fois le composant READ CSV pour charger l'échantillon test (CREDIT-GERMAN-TEST.TXT). Nous le connectons au composant APPLY MODEL (MODELING / MODEL APPLICATION) qui permet d'appliquer le classifieur sur un échantillon.



Enfin, l'opérateur PERFORMANCE CLASSIFICATION (EVALUATION / PERFORMANCE MEASUREMENT / CLASSIFICATION AND REGRESSION) permet de mesurer le taux de succès (ACCURACY).



Nous exécutons le diagramme. Nous obtenons un taux de succès de 74.86% sur l'échantillon de 700 observations.



5 Expérimentation sur la base WAVE à 2 classes

Pour donner une plus grande assise à notre étude, nous avons réitéré l'expérimentation sous R sur la base WAVE[17] réduite à 2 classes (21 variables prédictives quantitatives). Nous disposons de 500 observations en apprentissage et, surtout, de 32867 observations en test. L'estimation du taux de succès n'en sera que plus précise.

[17] http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+1%29

Nous obtenons le tableau suivant :

WAVE DATASET		Accuracy rate	
% missing	# complete obs.	Listwise Del.	Univ. Imputation
0,00%	500	0,9150	0,9150
0,50%	451	0,9147	0,9150
1,00%	405	0,9082	0,9162
2,00%	331	0,9068	0,9160
5,00%	177	0,8731	0,9174
10,00%	72	0,7847	0,9192
20,00%	6	ERR	0,9188

Les résultats corroborent ceux obtenus sur la base GERMAN. Lorsque la proportion de valeur manquante augmente, l'imputation univariée tient parfaitement son rôle alors que la suppression de lignes dégrade fortement la construction du modèle prédictif. Assez curieusement, on aurait même dit que l'introduction de valeurs manquantes pour les remplacer par la moyenne ou le mode améliorerait la qualité des modèles. Les différences sont infimes, gardons-nous bien de ce genre de conclusion. Mais j'avoue que ces résultats m'intriguent quelque peu, et ce n'est pas faute d'avoir vérifié l'exactitude des taux de succès recueillis[18].

6 Conclusion

On ne le répétera jamais assez, le traitement des données manquantes est un problème difficile. Il faut faire des choix en fonction de facteurs que l'on ne maîtrise pas toujours très bien (le processus de formation des valeurs manquantes notamment). Dans ce tutoriel, nous avons essayé de montrer les solutions proposées par plusieurs logiciels. Nous constatons que dans le cadre des données manquantes totalement MCAR, l'imputation univariée (moyenne / mode) convient très bien dans la construction de modèles prédictifs linéaires, dans la mesure où notre principal critère d'évaluation est l'efficacité en prédiction.

7 Bibliographie

Allison, P.D. (2001), « Missing Data ». Sage University Papers Series on Quantitative Applications in the Social Sciences, 07-136. Thousand Oaks, CA : Sage.

Little, R.J.A., Rubin, D.B. (2002), « Statistical Analysis with Missing Data », 2nd Edition, New York : John Wiley.

[18] De toute manière, les données réellement utilisées et les scripts sont accessibles sur notre site de tutoriels. R, tout comme Orange, Knime et RapidMiner, sont eux-mêmes librement téléchargeables sur leurs sites de distribution respectifs. Tout le monde peut reproduire à l'identique les résultats décrits dans ce document. C'est une règle que l'on devrait voir instaurer pour les publications dites « scientifiques ».