1 Objectif

Déploiement des modèles prédictifs avec R.

L'industrialisation est l'étape ultime du data mining. Dans le cadre prédictif, l'objectif est de classer un individu à partir de sa description. Pour les établissement de crédit par exemple, la question posée est la suivante : connaissant l'âge, le revenu, la situation matrimoniale d'un client, est-ce qu'il s'avère risqué de lui prêter de l'argent ? C'est ainsi que sur certains sites web, à partir des informations signalétiques saisies dans des formulaires, on nous fournit un accord de principe (ou un refus) pour un emprunt que nous sollicitons.

L'industrialisation repose sur la possibilité de sauver, de diffuser et d'exploiter le classifieur élaboré lors de la phase d'apprentissage. On parle de déploiement. Schématiquement, nous distinguerons 3 grandes étapes dans le processus data mining :

1. **Construction du modèle prédictif à partir des données disponibles**. Cette phase de modélisation comprend la préparation des données, la recherche du meilleur classifieur, son évaluation. Le data miner y consacre l'essentiel de son temps, ce travail doit être rentabilisée.
2. **Déploiement du modèle**. Une fois que nous avons mis en évidence un classifieur satisfaisant, il doit être sauvé dans un fichier (en tous les cas sur un support non volatile), puis diffusé pour implantation dans les autres services de l'entreprise. ***Seul le modèle est diffusé***. Les données qui ont servi à son élaboration n'ont pas à l'être. C'est d'autant plus vrai que, souvent, nous nous appuyons sur de gros volumes de données pour construire des modèles efficaces.
3. **Mise en œuvre du modèle pour le classement de nouveaux individus.** Une fois le modèle implanté dans un nouvel environnement, il est réellement utilisé pour classer les individus à partir de leur description. C'est la phase d'industrialisation.

L'intégration (le déploiement) des modèles dans les systèmes d'information est donc une étape très importante[1]. Une stratégie très simple, largement utilisée encore de nos jours, consiste à les transcrire manuellement via un langage de programmation quelconque (C++, Java, etc.) dans le nouvel environnement. Mais, elle n'est pas tenable à long terme. L'accélération de l'usage du data mining dans les entreprises, la multiplication des modèles et la nécessité de les mettre à jour régulièrement imposent une automatisation de la phase de déploiement.

Ces dernières années, la norme PMML tend à (*tente de*) s'imposer. Elle est soutenue par un groupe très actif qui propose une description normalisée des modèles. Très séduisante a priori, un standard accepté par tous est un gage d'universalité, cette solution n'est pas sans poser de problèmes : (a) il faut déjà que les outils de data mining sachent produire les fichiers respectant la norme qui, par ailleurs, évolue dans le temps (http://www.dmg.org/v4-0/GeneralStructure.html) ; (b) il faut ensuite que les systèmes d'information, en plus de savoir manipuler les données, sachent interpréter correctement les modèles codés en PMML pour pouvoir les appliquer. Sur le site du « Data Mining group », nous disposons de la liste des logiciels qui savent gérer ces deux aspects. Les acteurs et les méthodes couvertes sont finalement peu nombreux (http://www.dmg.org/products.html).

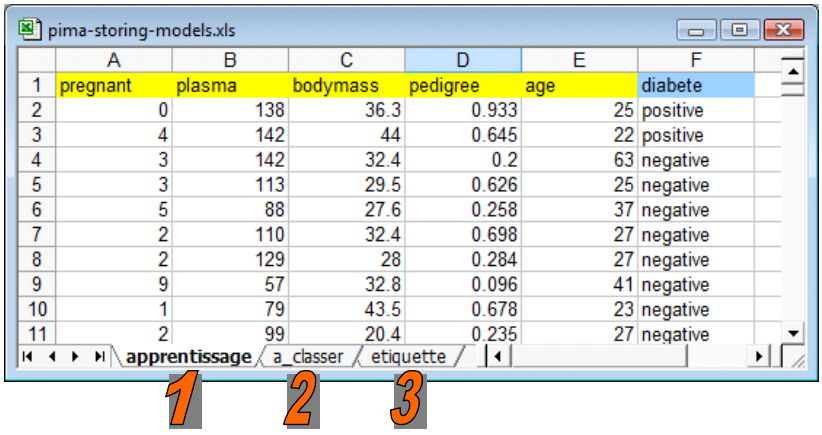
[1] http://www.kdnuggets.com/polls/2009/deployment-data-mining-models.htm

Entre ces deux extrêmes (écriture de programmes vs. PMML), certains logiciels proposent des solutions ad hoc pour l'industrialisation des modèles. Dans ce tutoriel, nous en présentons une pour R. Elle repose sur la possibilité de sauvegarder des modèles dans des fichiers binaires via le package **filehash** (http://cran.r-project.org/web/packages/filehash/index.html). Certes, nous aurons encore besoin du logiciel R dans la phase d'industrialisation (pour le classement de nouveaux individus), mais plusieurs aspects militent en faveur de cette stratégie : R est librement accessible et utilisable dans quelque contexte que ce soit ; il fonctionne indifféremment sous Windows, sous Linux et sous MacOS (http://www.r-project.org/); nous pouvons le piloter en mode batch c.-à-d. tout programme peut faire appel à R en sous main, lui faire exécuter une tâche, et récupérer les résultats.

Nous écrirons trois programmes distincts pour différencier les étapes. Le premier construit les modèles à partir des données d'apprentissage et les stocke dans un fichier binaire. Le second charge les modèles et les utilise pour classer les individus non étiquetés d'un second ensemble de données. Les prédictions sont sauvées dans un fichier CSV. Enfin, le troisième charge les prédictions et la vraie classe d'appartenance conservée dans un troisième fichier, il construit les matrices de confusion et calcule les taux d'erreur. Les méthodes de data mining utilisés sont : les arbres de décision (rpart) ; la régression logistique (glm) ; l'analyse discriminante linéaire (lda) ; et l'analyse discriminante sur facteurs de l'ACP (princomp + lda). Avec ce dernier cas, on montre que la stratégie reste opérationnelle même lorsque la prédiction nécessite un enchaînement d'opérations complexes.

2 Données

Nous regroupons les données dans un classeur Excel « pima-storing-models.xls » (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/pima-model-deployment.zip). Nous disposons de 3 feuilles : (1) les données d'apprentissage étiquetées (apprentissage) ; (2) les observations à classer (a_classer), pour lesquelles nous disposons uniquement des valeurs des variables prédictives ; (3) les étiquettes des individus de la feuille précédente (étiquette).



	A	B	C	D	E	F
1	pregnant	plasma	bodymass	pedigree	age	diabete
2	0	138	36.3	0.933	25	positive
3	4	142	44	0.645	22	positive
4	3	142	32.4	0.2	63	negative
5	3	113	29.5	0.626	25	negative
6	5	88	27.6	0.258	37	negative
7	2	110	32.4	0.698	27	negative
8	2	129	28	0.284	27	negative
9	9	57	32.8	0.096	41	negative
10	1	79	43.5	0.678	23	negative
11	2	99	20.4	0.235	27	negative

3 Construction et stockage des modèles prédictifs

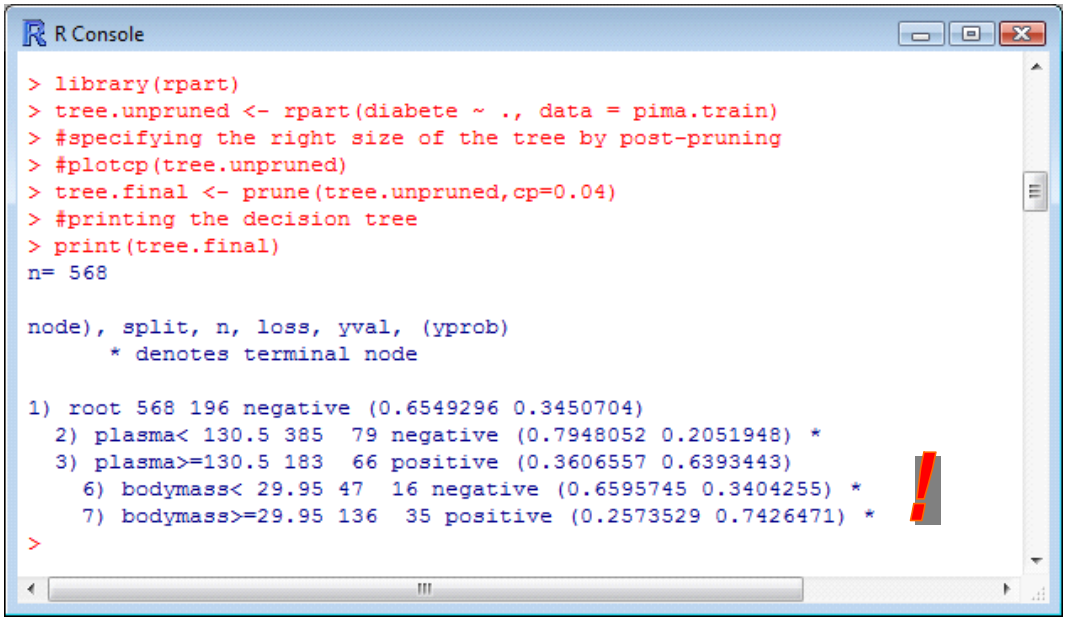
La première étape consiste à construire les modèles prédictifs à partir des données d'apprentissage. Nous les stockons par la suite dans un fichier externe en utilisant les fonctionnalités du package « filehash » que nous avons longuement présenté par ailleurs (http://tutoriels-data-mining.blogspot.com/2010/06/traitement-des-tres-grands-fichiers.html).

```
#vider la mémoire
rm (list=ls())
#charger les données
library(xlsReadWrite)
setwd("D:/DataMining/Databases_for_mining/dataset_for_soft_dev_and_comparison/idt-spad")
pima.train <- read.xls(file="pima-storing-models.xls",colNames=T,sheet="apprentissage")

#induction d'un arbre de décision avec rpart
library(rpart)
tree.unpruned <- rpart(diabete ~ ., data = pima.train)
#post-élagage
tree.final <- prune(tree.unpruned,cp=0.04)
#impression de l'arbre de décision
print(tree.final)

#initialisation de la régression logistique avec uniquement la constante
logreg.initial <- glm(diabete ~ 1, data = pima.train, family = binomial)
#sélection de variables – recherche forward avec stepAIC
library(MASS)
logreg.final <- stepAIC(logreg.initial,
      scope=list(lower="~1",upper="~age+pedigree+pregnant+bodymass+plasma"),trace=T,direction="forward")
#impression du modèle définitif
print(logreg.final)

#fsélection forward pour l'analyse discriminante
library(klaR)
selection.var <- greedy.wilks(diabete ~ ., data = pima.train, niveau = 0.01)
#analyse discriminante avec les variables sélectionnées
lda.final <- lda(selection.var$formula, data = pima.train)
#impression du modèle définitif
print(lda.final)

#ACP sur les variables prédictives
pca <- princomp(~age+pedigree+pregnant+bodymass+plasma, data = pima.train, cor = T, scores = T)
#analyse discriminante sur les deux premiers facteurs de l'ACP
scores <- as.data.frame(pca$scores)
lda.pca <- lda(pima.train$diabete ~ scores$Comp.1 + scores$Comp.2)
#impression du modèle
print(lda.pca)

#suppression de la version précédente du fichier de stockage (si nécessaire)
file.remove("models.db")
#stockage des modèles prédictifs (et intermédiaires comme l'ACP) dans « models.db »
library(filehash)
dumpObjects(tree.final,dbName="models.db")
dumpObjects(logreg.final,dbName="models.db")
dumpObjects(lda.final,dbName="models.db")
dumpObjects(pca,dbName="models.db")
dumpObjects(lda.pca,dbName="models.db")
```

Les données étiquetées de la première feuille du classeur sont chargées avec l'instruction **read.xls(.)** *[package xlsReadWrite]*. 4 techniques prédictives sont implémentées dans ce didacticiel.

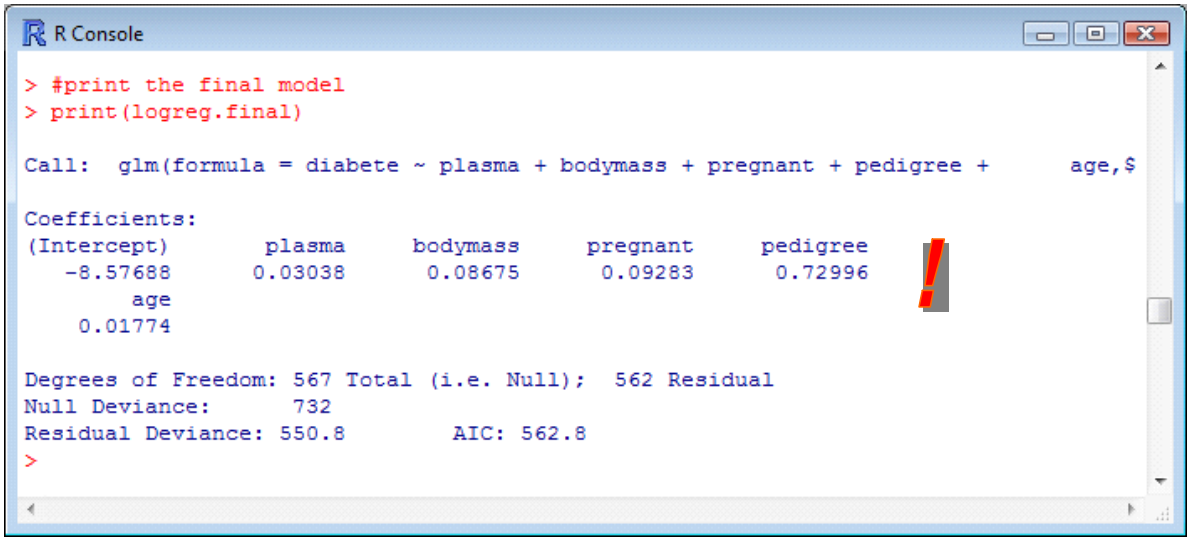
3.1 Arbre de décision

Nous utilisons **rpart()** [*package rpart*] pour générer l'arbre de décision maximal, **prune()** se charge du post élagage. Nous le montrons pas ici, nous avons utilisé la fonction **plotcp()** pour détecter la valeur adéquate du paramètre « cp ». Nous obtenons un arbre impliquant seulement 2 des 5 variables prédictives.



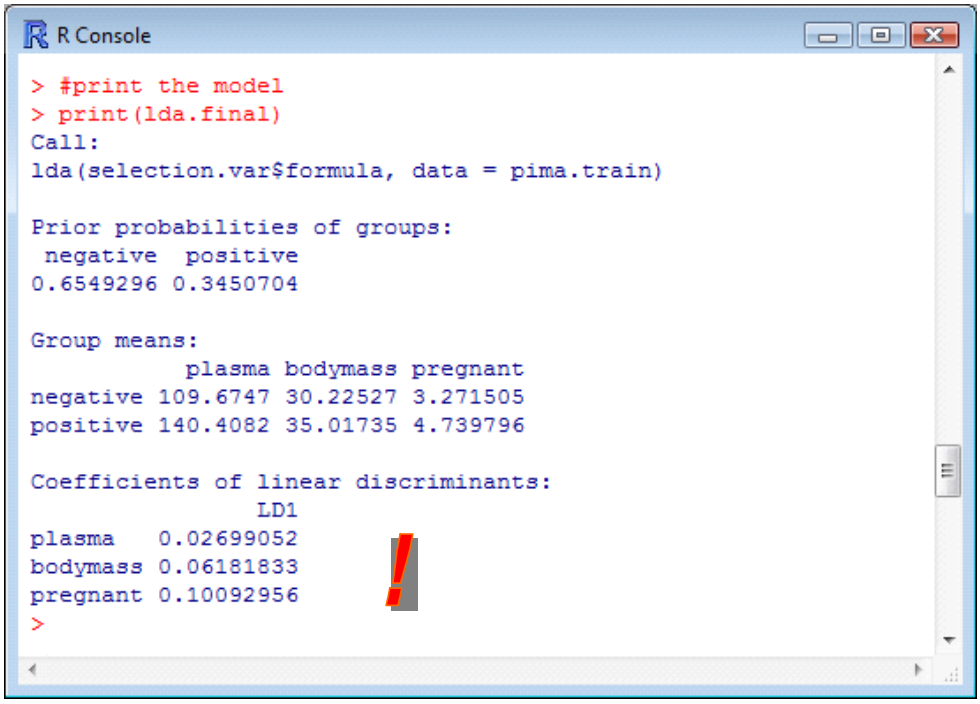
3.2 Régression logistique

Pour la régression logistique, nous utilisons la commande **stepAIC()** [*package MASS*] pour détecter automatiquement les variables pertinentes. Le premier appel à **glm()** sert à initialiser la procédure de recherche. Toutes les variables prédictives ont finalement été retenues.



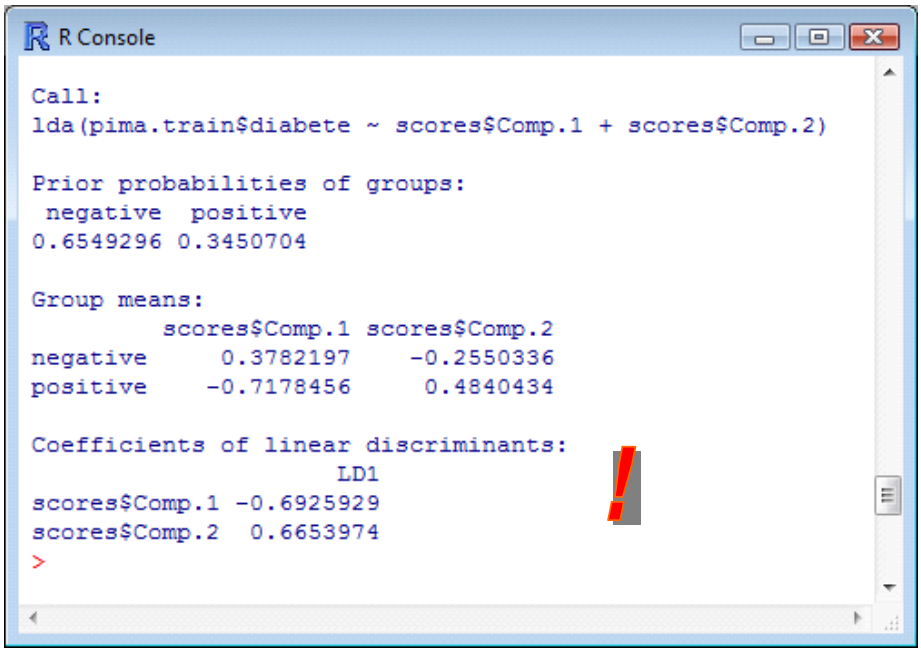
3.3 Analyse discriminante

Nous utilisons tout d'abord la commande **greedy.wilks()** [*package klaR*] pour détecter les variables prédictives pertinentes, puis **lda()** pour réaliser l'analyse discriminante sur les variables sélectionnées.



3.4 Analyse discriminante sur facteurs de l'ACP

Nous procédons en deux temps. L'analyse en composantes principales **princomp()** projette les individus dans un nouvel espace de représentation. Nous utilisons les deux premiers facteurs comme variables prédictives dans l'analyse discriminante **lda()** subséquente.

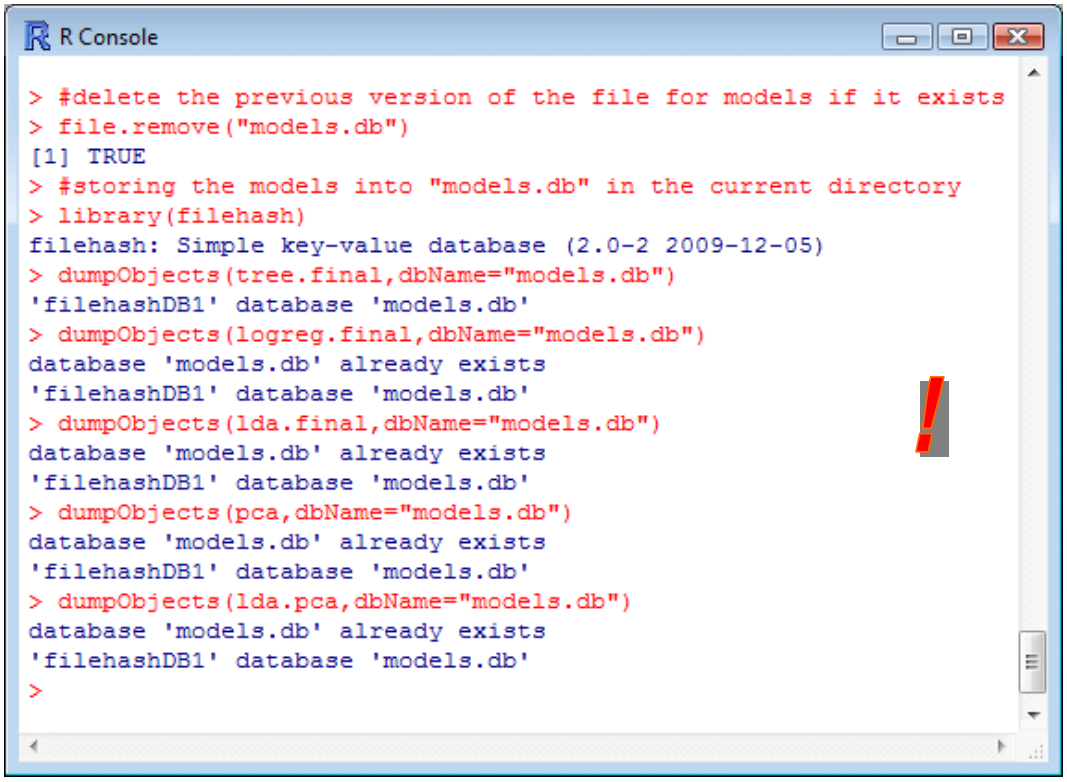


Nous avions présenté cette approche dans un de nos tutoriels. C'est une manière, entres autres, de traiter le problème des prédicteurs fortement corrélés (http://tutoriels-data-mining.blogspot.com/2008/03/analyse-discriminante-sur-axes.html).

3.5 Stockage des modèles

Tous les modèles sont stockés dans le seul fichier « models.db » à l'aide de la fonction **dumpObjects() [***package filehash***]**. Ils peuvent donc co-exister. Nous les distinguerons à l'aide de leur identifiant lors du chargement ultérieur.

Concernant l'analyse discriminante sur facteurs, nous devons à la fois sauver l'objet « pca » correspondant au résultat de l'ACP, et « lda.pca » correspondant à l'analyse discriminante sur les 2 premiers facteurs. Tous deux seront nécessaires lors de la prédiction sur de nouveaux individus.



Le fichier « models.db » peut être copié sur n'importer quel support et distribué à quiconque veut exploiter les modèles pour classer de nouveaux individus. C'est ce que nous ferons dans la section suivante.

4 Classement de nouveaux individus

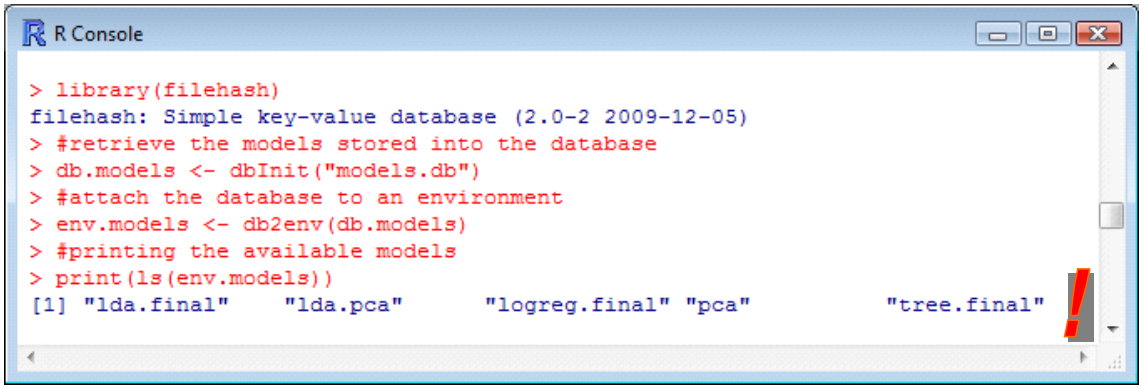
Deux préalables sont nécessaires pour que l'on puisse appliquer les modèles sur un fichier d'observations non étiquetées: (1) l'utilisateur doit disposer du logiciel R sur sa machine, avec les packages adéquats, qu'importe le système d'exploitation ; (2) les noms des variables du fichier décrivant les individus à classer doivent correspondre à ceux de l'échantillon d'apprentissage.

4.1 Déploiement avec R sous Windows

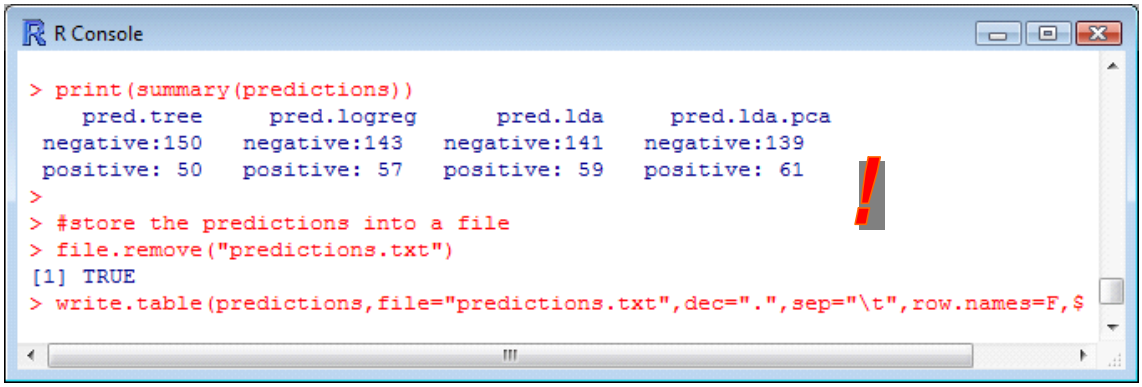
```
#vider la mémoire
rm (list=ls())

#charger les données de la seconde feuille du classeur Excel
library(xlsReadWrite)
setwd("D:/DataMining/Databases_for_mining/dataset_for_soft_dev_and_comparison/idt-spad")
pima.unlabeled <- read.xls(file="pima-storing-models.xls",colNames=T,sheet="a_classer")

#connexion à la base contenant les modèles
library(filehash)
db.models <- dbInit("models.db")
#rattacher la base à un environnement
env.models <- db2env(db.models)
#lister les modèles disponibles pour vérification
print(ls(env.models))

#charger les packages adéquats – correspondants aux modèles à manipuler
library(rpart)
library(MASS)

#classer les individus non étiquetés – en utilisant chaque modèle
pred.tree <- predict(env.models$tree.final,newdata=pima.unlabeled,type="class")
pred.logreg <- as.factor(ifelse(predict(env.models$logreg.final, newdata=pima.unlabeled,type="response")>0.5,
"positive", "negative"))
pred.lda <- predict(env.models$lda.final,newdata=pima.unlabeled)$class
#attention, prédiction en deux temps pour l'analyse discriminante sur facteurs
#calcul des coordonnées factorielles (sur les axes de l'ACP)
scores.pca <- predict(env.models$pca,newdata=pima.unlabeled)
scores <- as.data.frame(scores.pca)
#prédiction de l'analyse discriminante à partir de ces coordonnées factorielles
pred.lda.pca <- predict(env.models$lda.pca,newdata=pima.unlabeled)$class

#créer un data frame pour les prédictions
predictions <- data.frame(pred.tree,pred.logreg,pred.lda,pred.lda.pca)
print(summary(predictions))

#stockage des prédictions dans un fichier CSV
file.remove("predictions.txt")
write.table(predictions,file="predictions.txt",dec=".",sep="\t",row.names=F,quote=F)
```

Reprenons les principaux résultats :

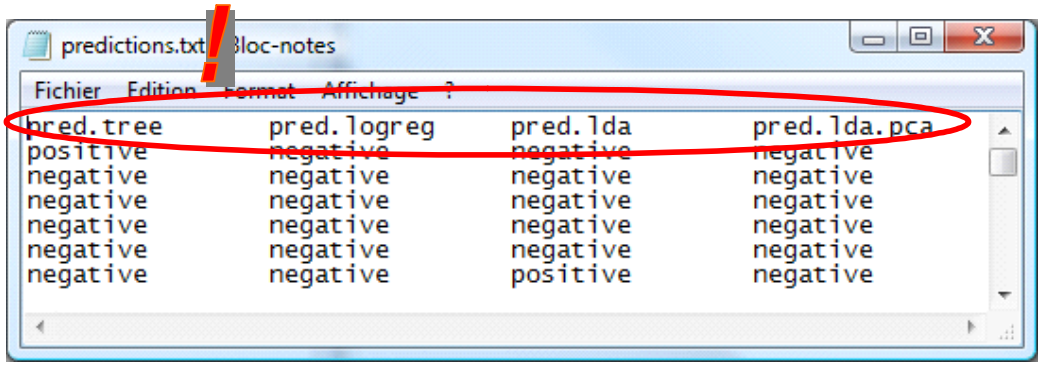
- 5 objets utilisables pour la prédiction sont disponibles dans le fichier « models.db ».



- L'application des modèles sur l'échantillon non étiqueté a correctement fonctionné. Les prédictions ont bien été stockées dans le fichier « predictions.txt ».



- Un coup d'œil rapide au fichier montre les 4 colonnes générées.

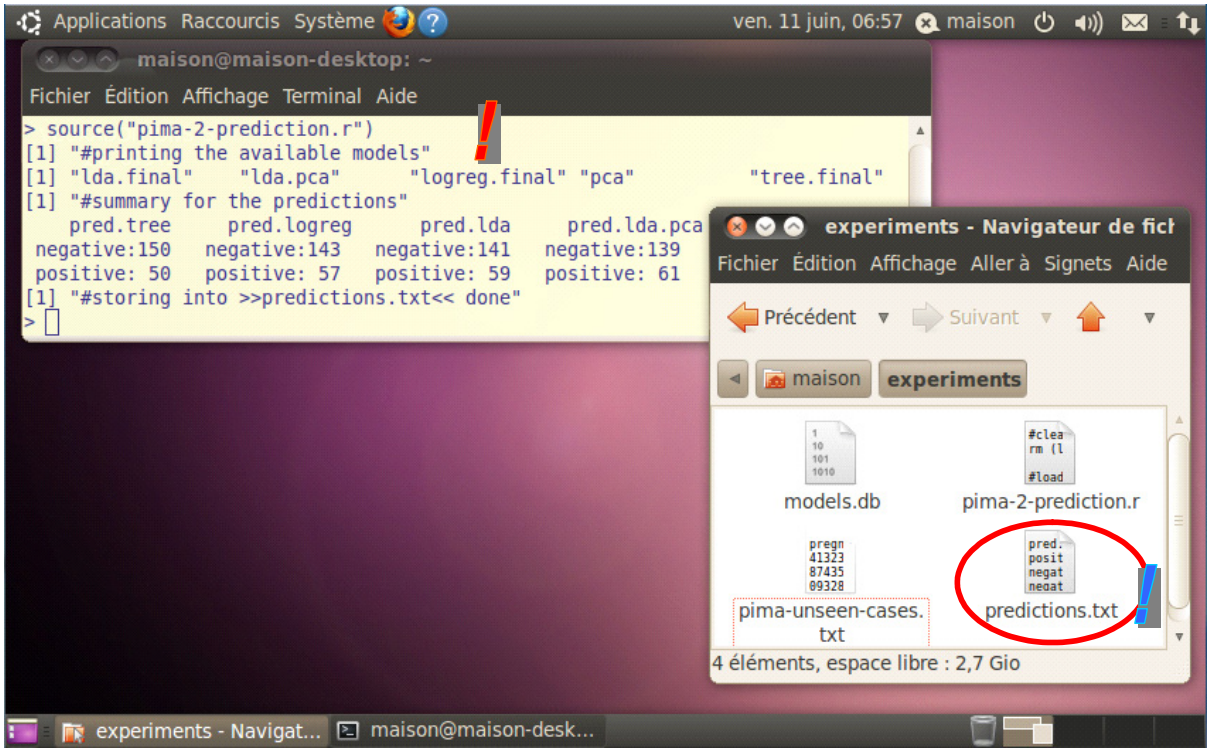


4.2 Déploiement avec R sous Linux

Le fichier « models.db » généré avec R sous Windows peut être exploité sous d'autres systèmes d'exploitation (pourvu que R y soit présent), en respectant exactement le même canevas. Cette caractéristique élargit considérablement la portée du dispositif.

Dans ce qui suit, nous montrons les résultats sous Linux (Ubuntu). Petite concession quand même, le package xlsReadWrite n'est pas accessible, nous avons donc transformé la seconde feuille du classeur Excel en fichier CSV. Cela ne change rien fondamentalement.

Au final, le fichier « prédictions.txt » est correctement généré.



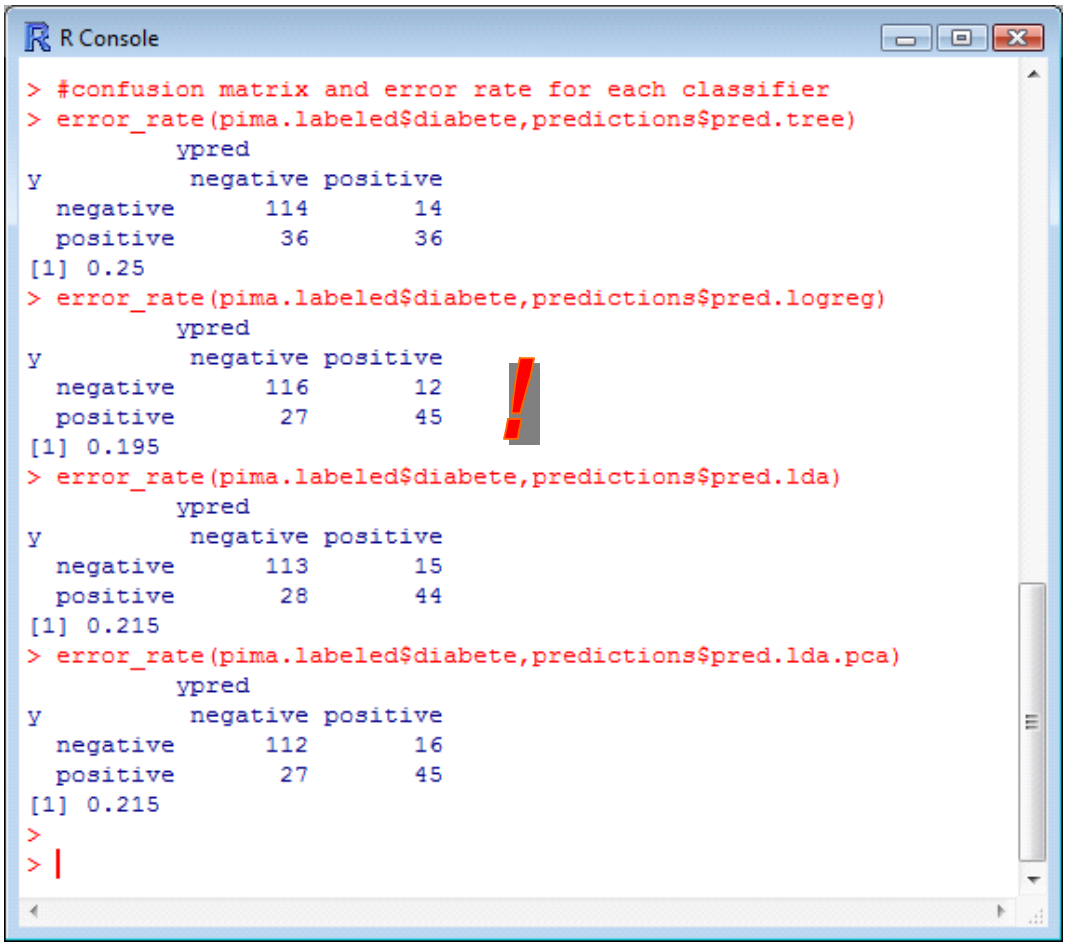
5 Évaluation des prédictions

Cette partie n'est pas indispensable. A titre de curiosité, puisque nous disposons par ailleurs des vraies étiquettes du second ensemble de données dans la troisième feuille de notre classeur Excel, il serait intéressant de voir comment se comportent les prédictions des différents modèles. C'est aussi une manière de vérifier la crédibilité du dispositif. Si les individus sont classés n'importe comment, la sauvegarde des modèles peut être remise en cause.

```
#vider la mémoire
rm(list=ls())

#fonction pour la matrice de confusion et le taux d'erreur
error_rate <- function(y,ypred){
  mc <- table(y,ypred)
  error <- (mc[1,2]+mc[2,1])/sum(mc)
  print(mc)
  print(error)
}

#charger les étiquettes des observations dans la 3ème feuille du classeur Excel
library(xlsReadWrite)
setwd("D:/DataMining/Databases_for_mining/dataset_for_soft_dev_and_comparison/idt-spad")
pima.labeled <- read.xls(file="pima-storing-models",colNames=T,sheet="etiquette")

#charger les prédictions des modèles
predictions <- read.table("predictions.txt",header=T,dec=".",sep="\t")

#matrice de confusion et taux d'erreur pour chaque prédiction
```

```
error_rate(pima.labeled$diabete,predictions$pred.tree)
error_rate(pima.labeled$diabete,predictions$pred.logreg)
error_rate(pima.labeled$diabete,predictions$pred.lda)
error_rate(pima.labeled$diabete,predictions$pred.lda.pca)
```

La régression logistique se révèle être la meilleure en prédiction ; l'arbre de décision, sur ce fichier en tous les cas, est le moins performant.



6 Conclusion

Dans ce didacticiel, nous montrons qu'il est facile de déployer les modèles prédictifs élaborés sous le logiciel R. Certes, le fichier à diffuser est au format binaire, seul R sait le lire. Cette limitation ne peut être ignorée. Mais, en contrepartie, R est gratuit, il peut fonctionner sous différents systèmes d'exploitation – nous avons ainsi montré qu'un fichier de modèles élaboré sous Windows est exploitable sous Linux – et nous pouvons le piloter en mode batch. Bref, la solution proposée est une alternative crédible aux stratégies adoptées par les autres logiciels.