

1 Theme

Model deployment with R using the `filehash` package.

Model deployment is the last task of the data mining steps. It corresponds to several aspects e.g. generating a report about the data exploration process, highlighting the useful results; applying models within an organization's decision making process; etc¹.

In this tutorial, we look at the context of predictive data mining. We are concerned about the construction of the model from a labeled dataset; the storage of the model; the distribution of the model, without the dataset used for its construction; the application of the model on new instances in order to assign them a class label from their description (the values of the descriptors).

The integration of the model into the organization's system is a very important task². A simple strategy consists in to use a programming language (C++, Java, etc.) to incorporate the model in a new environment. But this becomes tedious when we must deploy a large number of models. Another solution is the use of a standardized model description, recognized by the majority of CRM tools. The [PMML](#) norm seems a good solution. But, some tools do not know handle this standard. Moreover, the specification changes ([V.4.0.1](#) at this time) make difficult its integration in the tools.

An intermediate way is the use of specific solutions provided by some tools. We describe the `filehash` package for R which allows to deploy a model easily. The main advantage of this solution is that R can be launched under various operating systems. Thus, we can create a model with R under Windows; and apply the model in another environment, for instance with R under Linux. The solution can be easily generalized on a large scale because it is possible to launch R in batch mode. The update of the system will concern only the model file in the future.

We will write three R programs to distinguish the steps of the deployment process. The first one constructs a model from the dataset and stores it into a binary file (filehash format). The second one loads the model in another R session and uses it to label new instances from a second data file. The predictions are stored in a data file (CSV file format). Last, the third program loads the predictions and another data file containing the observed labels for these instances, and calculates the confusion matrix and the generalization error rate.

We use various predictive models in order to check the flexibility of the solutions. We tried the following ones: decision tree (`rpart`); logistic regression (`glm`); linear discriminant analysis (`lda`); linear discriminant analysis from factors of principal component analysis (`lda + pca`). This last one allowed to check if the system remains operational when we manipulate a combination of models.

2 Dataset

We group the various dataset in the « [pima-storing-models.xls](#) » Excel file (<http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/pima-model-deployment.zip>). There are 3 sheets into the workbook:

¹ http://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining

² <http://www.kdnuggets.com/polls/2009/deployment-data-mining-models.htm>

(1) the learning set (apprentissage); (2) the unlabeled dataset (à classer) i.e. the dataset on which we apply the model to obtain the predicted labels; (3) the observed labels of the previous instances (etiquette).

	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 Learning and storing the predictive models

First, we create the predictive model from the learning set. Then, we store the model or the combination of model as an object in a binary file using the **filehash** package³.

```
#clear the memory
rm (list=ls())
#loading the learning set (the labeled dataset)
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 of the decision tree using rpart
library(rpart)
tree.unpruned <- rpart(diabete ~ ., data = pima.train)
#post-pruning
tree.final <- prune(tree.unpruned,cp=0.04)
#printing the decision tree
print(tree.final)
#initializing the logistic regression for the stepwise variable selection
logreg.initial <- glm(diabete ~ 1, data = pima.train, family = binomial)
# forward variable selection using stepAIC
library(MASS)
logreg.final <- stepAIC(logreg.initial,
  scope=list(lower="~1",upper="~age+pedigree+pregnant+bodymass+plasma"),trace=T,direction="forward")
#printing the regression coefficients
print(logreg.final)
# forward selection for the linear discriminant analysis
```

³ We described the utilization package for the handling of very large dataset in a previous tutorial. The data mining algorithms can manipulate datasets (data.frame object) stored in a binary file on the disk. Thus, the memory occupation remains reasonable even if we process very large dataset. See <http://data-mining-tutorials.blogspot.com/2010/06/handling-large-dataset-in-r.html>

```

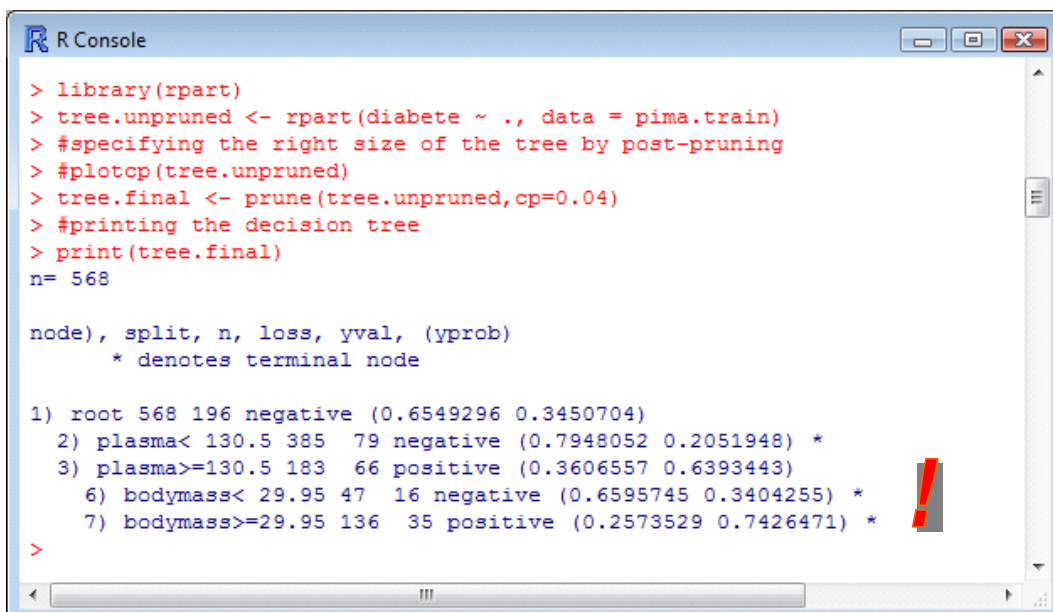
library(klaR)
selection.var <- greedy.wilks(diabete ~ ., data = pima.train, niveau = 0.01)
# linear discriminant analysis on the selected descriptors
lda.final <- lda(selection.var$formula, data = pima.train)
#printing the model coefficients
print(lda.final)
#PCA on the descriptors
pca <- princomp(~age+pedigree+pregnant+bodymass+plasma, data = pima.train, cor = T, scores = T)
#linear discriminant analysis on the first two factors (latent variables)
scores <- as.data.frame(pca$scores)
lda.pca <- lda(pima.train$diabete ~ scores$Comp.1 + scores$Comp.2)
#printing the model coefficients
print(lda.pca)

#removing the previous version of the model file (if necessary)
file.remove("models.db")
#storing the predictive models (including the PCA object) into the « models.db » file
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")

```

3.1 Decision tree

We use the **rpart()** procedure (from the **rpart** package) to create the maximal tree. Then, we use **plotcp()** to detect the good "cp" value (complexity parameter) for the post-pruning process (cp=0.04 here). Last, **prune()** enables to post-prune the tree. We obtain a tree with only 2 variables among the 5 candidate descriptors.



```

R Console
> library(rpart)
> tree.unpruned <- rpart(diabete ~ ., data = pima.train)
> #specifying the right size of the tree by post-pruning
> #plotcp(tree.unpruned)
> tree.final <- prune(tree.unpruned, cp=0.04)
> #printing the decision tree
> print(tree.final)
n= 568

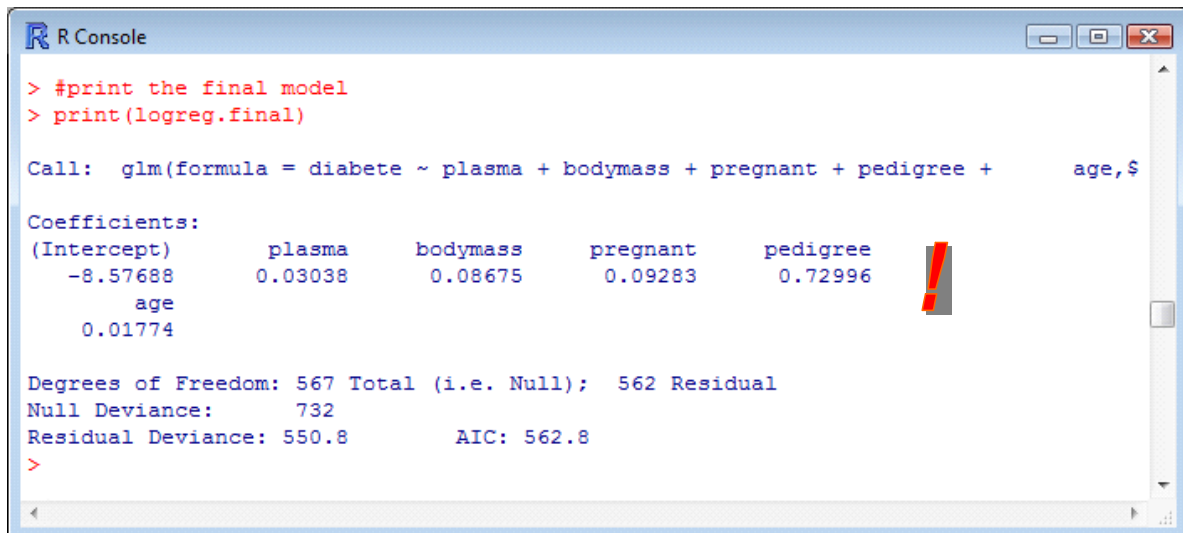
node), split, n, loss, yval, (yprob)
 * denotes terminal node

1) root 568 196 negative (0.6549296 0.3450704)
 2) plasma< 130.5 385 79 negative (0.7948052 0.2051948) *
 3) plasma>=130.5 183 66 positive (0.3606557 0.6393443)
   6) bodymass< 29.95 47 16 negative (0.6595745 0.3404255) *
   7) bodymass>=29.95 136 35 positive (0.2573529 0.7426471) *

```

3.2 Logistic regression

For the Logistic Regression, we use the `stepAIC()` command (MASS package) for the variable selection. Then, `glm()` procedure enables to create the model.



```

R Console
> #print the final model
> print(logreg.final)

Call:  glm(formula = diabete ~ plasma + bodymass + pregnant + pedigree + age, $

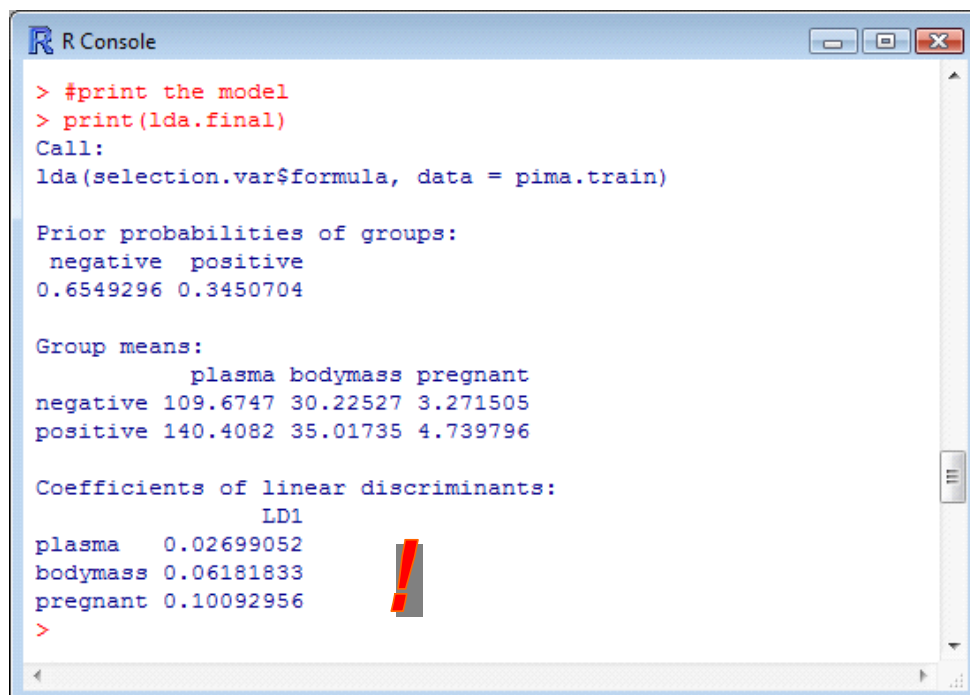
Coefficients:
(Intercept)      plasma      bodymass      pregnant      pedigree
 -8.57688      0.03038      0.08675      0.09283      0.72996
      age
  0.01774

Degrees of Freedom: 567 Total (i.e. Null); 562 Residual
Null Deviance:      732
Residual Deviance: 550.8      AIC: 562.8
>

```

3.3 Linear discriminant analysis

`greedy.wilks()` [package *klaR* package] allows to detect the relevant variables. Then, `lda()` is used for the construction of the model on the selected attributes.



```

R Console
> #print the model
> print(lda.final)

Call:
lda(selection.var$formula, data = pima.train)

Prior probabilities of groups:
  negative positive
0.6549296 0.3450704

Group means:
      plasma bodymass pregnant
negative 109.6747 30.22527 3.271505
positive 140.4082 35.01735 4.739796

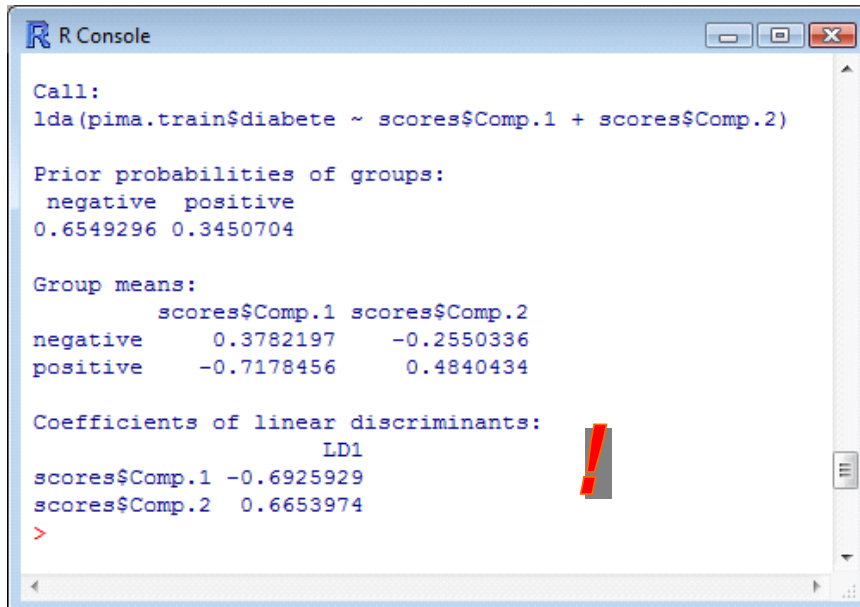
Coefficients of linear discriminants:
          LD1
plasma  0.02699052
bodymass 0.06181833
pregnant 0.10092956
>

```

3.4 Linear discriminant analysis from the factors of PCA

First, `princomp()` allows to calculate the latent variables. We use the two first ones in order to create the model with `lda()`. This strategy allows to regularize the linear discriminant approach, especially

when we have redundant descriptors (see <http://data-mining-tutorials.blogspot.com/2010/04/linear-discriminant-analysis-on-pca.html>).



```

R Console

Call:
lda(pima.train$diabete ~ scores$Comp.1 + scores$Comp.2)

Prior probabilities of groups:
negative positive
0.6549296 0.3450704

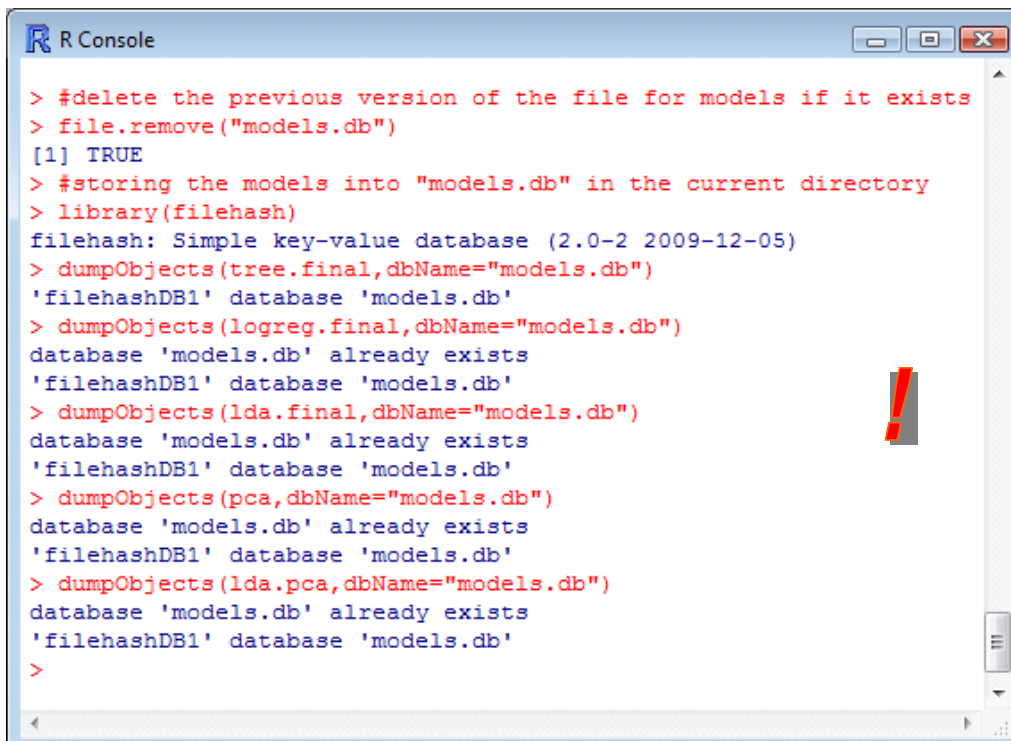
Group means:
      scores$Comp.1 scores$Comp.2
negative  0.3782197   -0.2550336
positive -0.7178456    0.4840434

Coefficients of linear discriminants:
          LD1
scores$Comp.1 -0.6925929
scores$Comp.2  0.6653974
>

```

3.5 Storing the models

All the models are stored into the "models.db" file using the **dumpObjects()** procedure [*filehash* package]. We have an identifier to distinguish them. About the LDA on PCA factors, we must store both the PCA and the LDA objects.



```

R Console

> #delete the previous version of the file for models if it exists
> file.remove("models.db")
[1] TRUE
> #storing the models into "models.db" in the current directory
> library(filehash)
filehash: Simple key-value database (2.0-2 2009-12-05)
> dumpObjects(tree.final, dbName="models.db")
'filehashDB1' database 'models.db'
> dumpObjects(logreg.final, dbName="models.db")
database 'models.db' already exists
'filehashDB1' database 'models.db'
> dumpObjects(lda.final, dbName="models.db")
database 'models.db' already exists
'filehashDB1' database 'models.db'
> dumpObjects(pca, dbName="models.db")
database 'models.db' already exists
'filehashDB1' database 'models.db'
> dumpObjects(lda.pca, dbName="models.db")
database 'models.db' already exists
'filehashDB1' database 'models.db'
>

```

The "models.db" file may be distributed (e.g. data storage device such as usb flash drive, network, etc.) to anyone who wants to use the models to classify new individuals i.e. to get the class value or the score from the values of the descriptors.

4 Classifying unlabeled instances

Two prerequisites are needed so that the models can be applied on a file with unlabelled instances: (1) we must have an installed version of R software with the required packages, whatever the operating system; (2) the variable names into the data file must be the same ones than the original data file (learning set).

4.1 Deployment with R under Windows

```
#clear the memory
rm (list=ls())

#load the unlabeled dataset from the second sheet of the Excel file
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")

#connection to the file containing the models
library(filehash)
db.models <- dbInit("models.db")
#assigning an environment to the models
env.models <- db2env(db.models)
#printing the available models
print(ls(env.models))

#loading the required packages
library(rpart)
library(MASS)

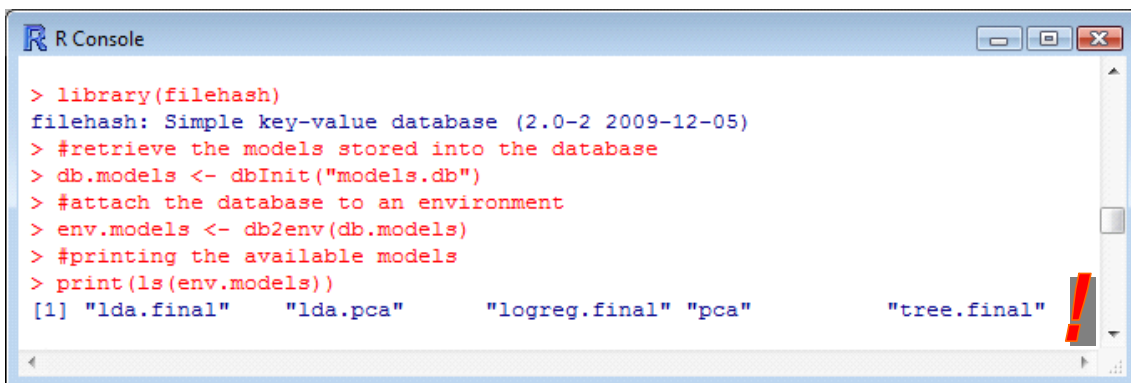
#classifying the unlabeled instances using the models
pred.tree <- predict(env.models$tree.final,newdata=pima.unlabeled,type="class")
pred.logreg <- as.factor(ifelse(predict(env.models$logreg.final, newdata=pima.unlabeled,type="response")>0.5,
"positive", "negative"))
pred.lda <- predict(env.models$lda.final,newdata=pima.unlabeled)$class
#attention, there are two steps for the combination of PCA and LDA
#calculating the scores of individuals on the factors of PCA
scores.pca <- predict(env.models$pca,newdata=pima.unlabeled)
scores <- as.data.frame(scores.pca)
#predicting with the LDA from the scores on the factors
pred.lda.pca <- predict(env.models$lda.pca,newdata=pima.unlabeled)$class

#creating a data frame structure from the various predictions
predictions <- data.frame(pred.tree,pred.logreg,pred.lda,pred.lda.pca)
print(summary(predictions))

#storing the predictions into a CSV data file
file.remove("predictions.txt")
write.table(predictions,file="predictions.txt",dec=".",sep="t",row.names=F,quote=F)
```

Let us see the main results:

- 5 models (objects) are available into the "models.db" file.

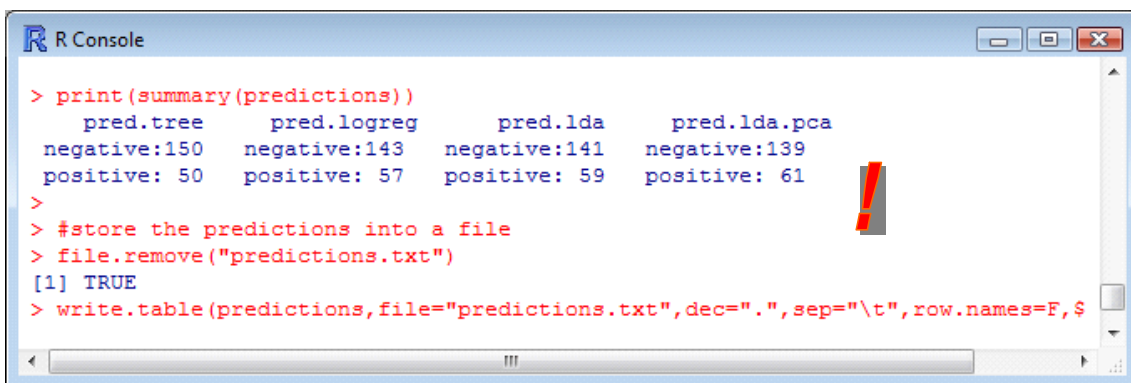


```

> library(filehash)
filehash: Simple key-value database (2.0-2 2009-12-05)
> #retrieve the models stored into the database
> db.models <- dbInit("models.db")
> #attach the database to an environment
> env.models <- db2env(db.models)
> #printing the available models
> print(ls(env.models))
[1] "lda.final"      "lda.pca"        "logreg.final"  "pca"            "tree.final"

```

- The application of the models on the unlabeled instances works fine. The predictions are correctly stored into the "predictions.txt" data file.

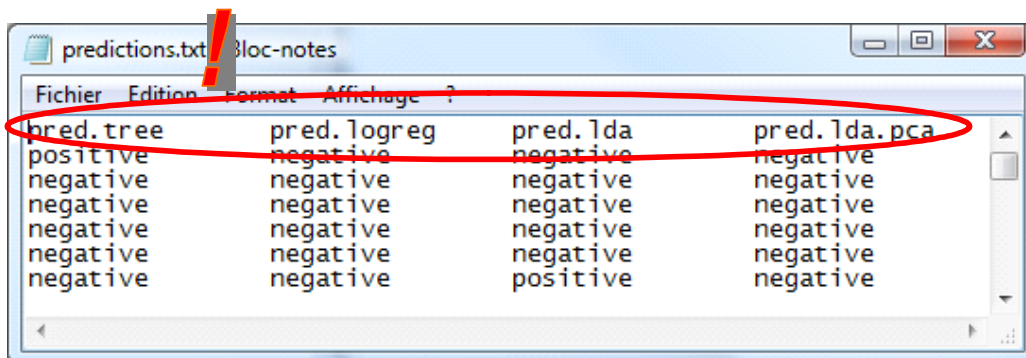


```

> print(summary(predictions))
  pred.tree  pred.logreg  pred.lda  pred.lda.pca
negative:150 negative:143 negative:141 negative:139
positive: 50 positive: 57 positive: 59 positive: 61
>
> #store the predictions into a file
> file.remove("predictions.txt")
[1] TRUE
> write.table(predictions,file="predictions.txt",dec=".",sep="\t",row.names=F,$

```

- We observe that 4 columns are stored into "predictions.txt".



```

pred.tree  pred.logreg  pred.lda  pred.lda.pca
positive   negative    negative  negative
negative   negative    negative  negative
negative   negative    negative  negative
negative   negative    negative  negative
negative   negative    negative  negative
negative   negative    positive  negative

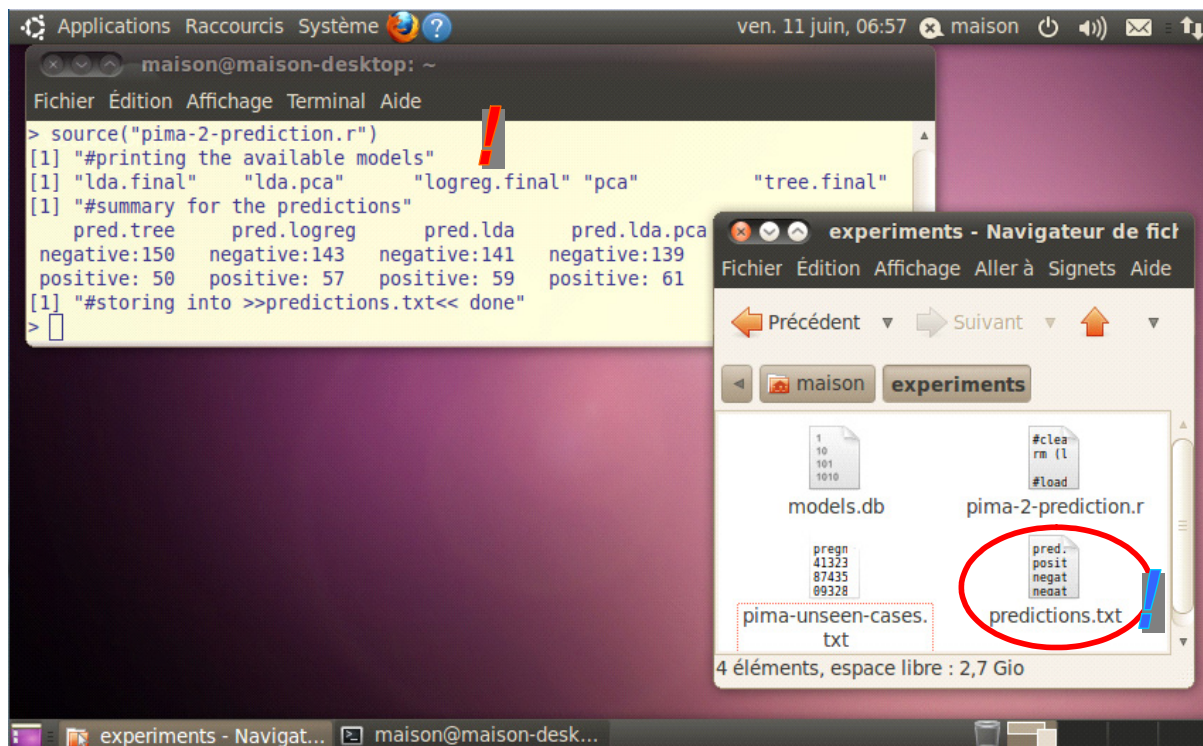
```

4.2 Deployment with R under Linux

The "models.db" file generated with R under Windows can be used with R under another operating system. This is an important property. It greatly expands the capacity of the system. The model can be used on a large variety of computer provided that R software is present.

We show the results under Linux (Ubuntu)⁴. The "predictions.txt" data file is properly generated.

⁴ Note: Because the xlsReadWrite package seems not work properly under Linux, I transform the source files (learning set, unlabeled instances, labels) into a CSV file. But, that does not modify the global framework



5 Assessing the predictions

This part is not essential. By way of comparison, since we have the labels of the instances to classify, it would be interesting to analyze the behavior of the various models.

This is also a way to check the credibility of the system. If the individuals are classified arbitrarily, the system for the storage and the retrieve of the models using "filehash" can be questionable.

```

#clear the memory
rm(list=ls())

#function for the calculation of the confusion matrix and the error rate
error_rate <- function(y,ypred){
  mc <- table(y,ypred)
  error <- (mc[1,2]+mc[2,1])/sum(mc)
  print(mc)
  print(error)
}

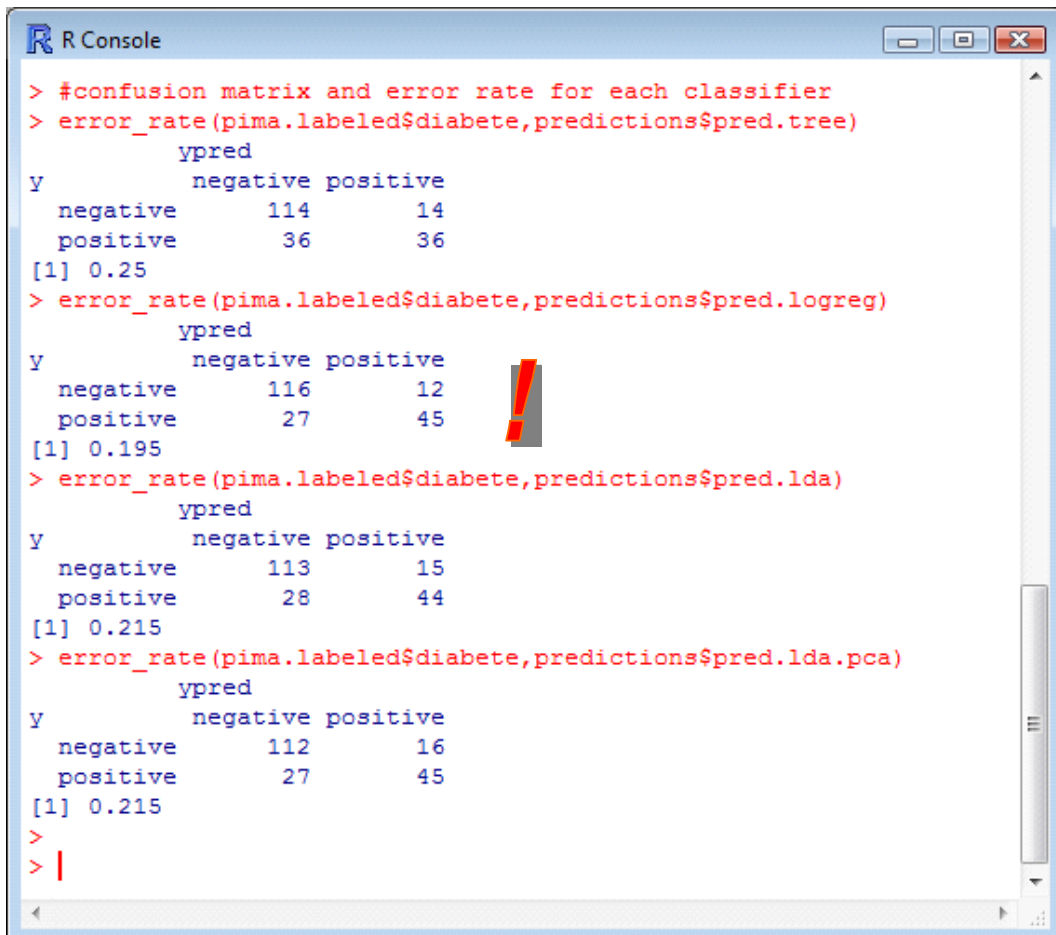
#loading the labels of the instances from the third sheet of the workbook
library(xlsReadWrite)
setwd("D:/DataMining/Databases_for_mining/dataset_for_soft_dev_and_comparison/idt-spad")
pima.labeled <- read.xls(file="pima-storing-models",colNames=T, sheet="etiquette")

#loading the predictions of the models
predictions <- read.table("predictions.txt",header=T,dec=".",sep="\t")

#confusion matrix and error rate for each prediction
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)
```

The logistic regression is the better; the decision is the worst, about our dataset in any case. Of course, this result cannot be applied to other datasets.



```
R Console
> #confusion matrix and error rate for each classifier
> error_rate(pima.labeled$diabete,predictions$pred.tree)
      ypred
Y      negative positive
negative  114      14
positive   36      36
[1] 0.25
> error_rate(pima.labeled$diabete,predictions$pred.logreg)
      ypred
Y      negative positive
negative  116      12
positive   27      45
[1] 0.195
> error_rate(pima.labeled$diabete,predictions$pred.lda)
      ypred
Y      negative positive
negative  113      15
positive   28      44
[1] 0.215
> error_rate(pima.labeled$diabete,predictions$pred.lda.pca)
      ypred
Y      negative positive
negative  112      16
positive   27      45
[1] 0.215
>
> |
```

6 Conclusion

In this tutorial, we show that it is easy to deploy predictive models developed under the R software using the [filehash](#) package. Indeed, the models file is in binary format, only R knows to read it. But R is free and it can run under various operating systems. The proposed solution is a credible alternative to the systems implemented by other software.