USER GUIDE

SPA, Supervised Provenance Analysis

A computer tool based on supervised machine learning algorithms to predict provenance of ceramic fragments

Credits:

Anna Anglisano, UAB (anna.ar.93@gmail.com)

Lluís Casas, UAB (lluis.casas@uab.cat)

Version: 13/05/2022

0. Installation

1. Software required:

R program: (https://www.r-project.org) – recommended version R-4.1.2

R Studio: (https://www.rstudio.com)

2. Application SPA (Supervised Provenance Analysis) Installation

2.1 Where to find all the required documents

Option A [To get the published version]

Download and extract the Supervised_Provenance_Analysis.zip file. Copy the extracted folder to your desired location within your computer.

Option B [To get an eventually updated version]

Download from Githup (https://github.com/)

Sign up on this platform and find the repository: Supervised_Provenance_Analysis. Download and extract the Supervised_Provenance_Analysis.zip file. Copy the extracted folder to your desired location within your computer.

2.2 Install R

Within the copied folder you will find R the installer for Windows. If you use another operating system you could find the corresponding installer in: (https://www.r-project.org)

Install R on your computer, R-4.1.2 version is strongly recommended, on older versions the SPA application could not work as expected.

2.3 Install RStudio

RStudio provides free and open source tools for R and it is required to work with SPA. The installer for Windows can be found into the folder. Installers for different operation systems are available at: https://www.rstudio.com

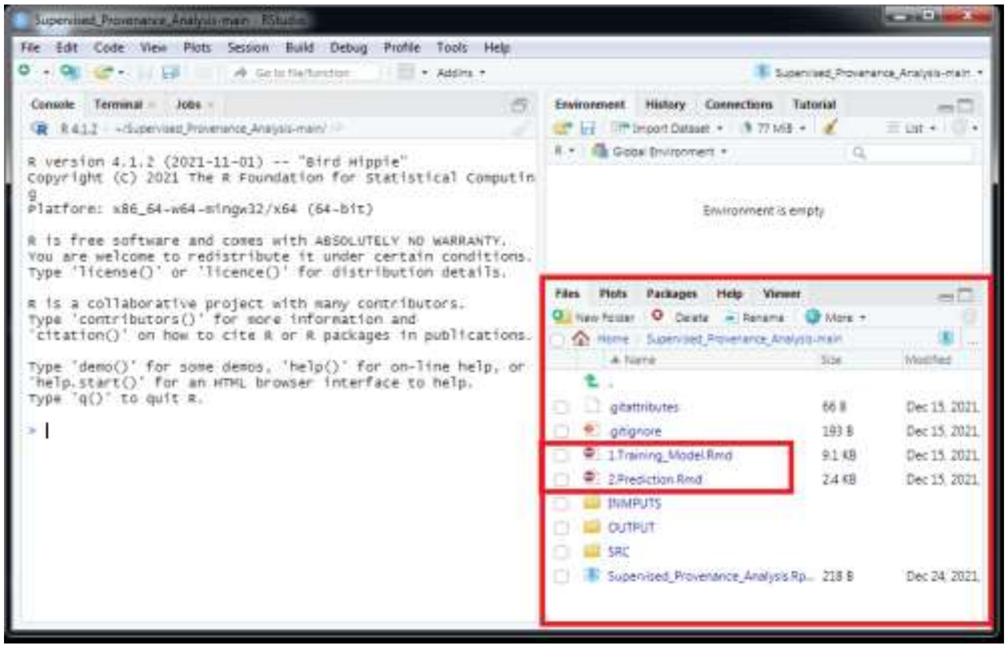
Install RStudio

3. Preparing and running SPA software for the first time

3.1 Open RStudio with preconfigured SPA software:

Go to the extracted folder within your computer and open the file: Supervised_Provenance_Analysis.Rproj

RStudio will open and will be automatically configurated to use SPA software. The software consists mainly in two different codes. The first (1.Training_Model.Rmd) generates a classification model and the second one (2.Prediction.Rmd) produces provenance probabilities for unlabeled samples based on the results of the classification model.


Supervised_Provenance_Analysis-main - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Supervised_Provenance_Analysis-main
Console Terminal Jobs
R 4.1.2 · ~/Supervised_Provenance_Analysis-main/
Environment History Connections Tutorial
Import Dataset
77 MiB
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home Supervised_Provenance_Analysis-main
Name Size Modified

```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computin
g
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

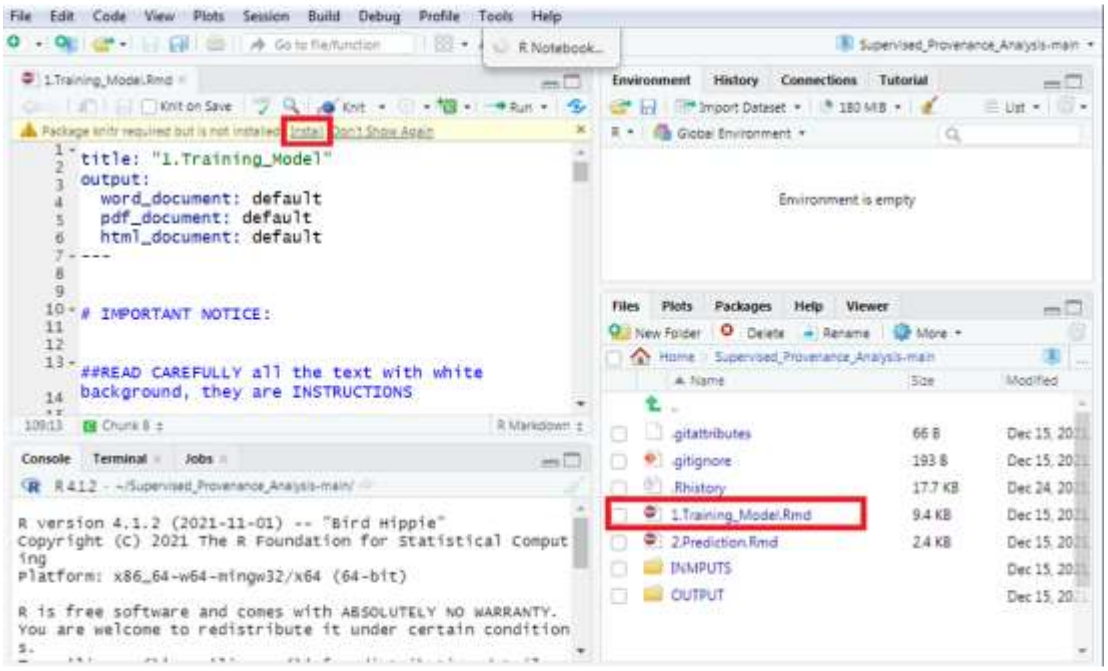
Name	Size	Modified
.gitattributes	66 B	Dec 15, 2021,
.gitignore	193 B	Dec 15, 2021,
1.Training_Model.Rmd	9.1 KB	Dec 15, 2021,
2.Prediction.Rmd	2.4 KB	Dec 15, 2021,
INMPUTS		
OUTPUT		
SRC		
Supervised_Provenance_Analysis.Rp...	218 B	Dec 24, 2021,

Figure UG1. RStudio opened with the preset configuration to run the SPA codes

3.2 Installing first requirements:

Open the first code (1.Training_Model.rmd) by clicking on the document at the bottom-right window.

The code will appear in a window on the top-left of the screen and a message to install the knitr package will appear. Click on Install to start the installation of the package.


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
R Notebook...
Supervised_Provenance_Analysis-main
1.Training_Model.Rmd
Knit on Save
Knit
Run
Package knitr required but is not installed. Install Don't Show Again
Chunk 8
R Markdown
Console Terminal Jobs
R 4.1.2 · ~/Supervised_Provenance_Analysis-main/
Environment History Connections Tutorial
Import Dataset
180 MiB
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home Supervised_Provenance_Analysis-main
Name Size Modified

```
title: "1.Training_Model"
output:
  word_document: default
  pdf_document: default
  html_document: default
---


# IMPORTANT NOTICE:


##READ CAREFULLY all the text with white
background, they are INSTRUCTIONS
```

```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Comput
ing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain condition
s.
```

Name	Size	Modified
.gitattributes	66 B	Dec 15, 20
.gitignore	193 B	Dec 15, 20
.Rhistory	17.7 KB	Dec 24, 20
1.Training_Model.Rmd	9.4 KB	Dec 15, 20
2.Prediction.Rmd	2.4 KB	Dec 15, 20
INMPUTS		Dec 15, 20
OUTPUT		Dec 15, 20

Figure UG2. Click first on ´1.Training_Model.rmd´ to open the code, then click on 'Install' to install knitr package

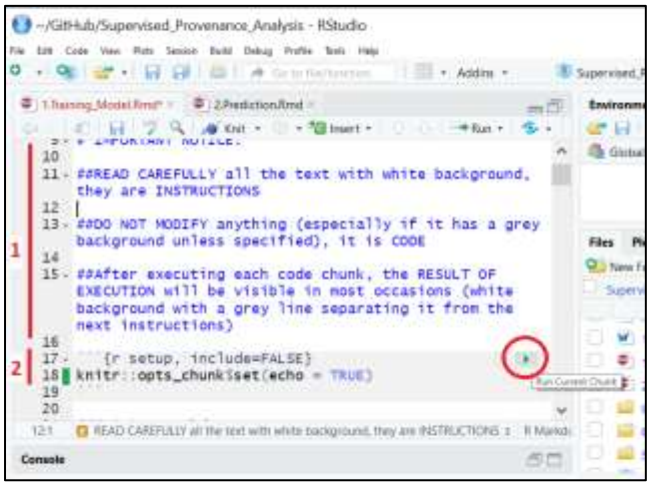
3.3 How to run an R Markdown document, first steps

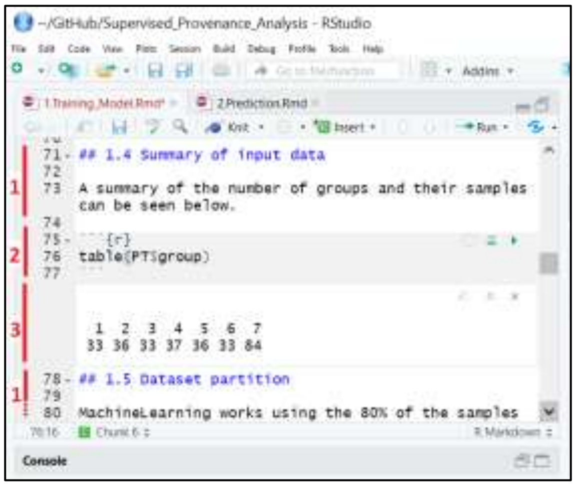
Both codes within the SPA software are R Markdown documents (Rmd extension). This kind of documents contain information/instructions (white background) and actions/code (grey background). Some code actions will generate results. Those will appear below (in black characters on a white background and with a thin grey line at the end). Read carefully each line of the document and do not modify anything unless specified.

To run each chunk of the code, click on the green play button at the beginning of each chunk. *If you are familiar with RStudio, you can also execute "Run All" and analyze the results by reading carefully the information in the code (see section 3.5).*

After clicking, only the current chunk of code will be run. For instance, the chunk within the figure below will install knitr. While it is running a stop button will appear instead of play button. Do not run any other chunk code until the action is completed. Completion can be noticed because a play button will appear again.

Chunk code 1


~/GitHub/Supervised_Provenance_Analysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
1.Training_Model.Rmd*
2.Prediction.Rmd
Knit
Insert
Run
Environment
Files
1
2
Run Current Chunk
Console

~~~
# IMPORTANT NOTICE:

##READ CAREFULLY all the text with white background,
they are INSTRUCTIONS

##DO NOT MODIFY anything (especially if it has a grey
background unless specified), it is CODE

##After executing each code chunk, the RESULT OF
EXECUTION will be visible in most occasions (white
background with a grey line separating it from the
next instructions)

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~
```

```
12:1   READ CAREFULLY all the text with white background, they are INSTRUCTIONS   R Markdown
```

Chunk code 6


~/GitHub/Supervised_Provenance_Analysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
1.Training_Model.Rmd*
2.Prediction.Rmd
Knit
Insert
Run
1
2
3
1
Console

```
## 1.4 Summary of input data

A summary of the number of groups and their samples
can be seen below.

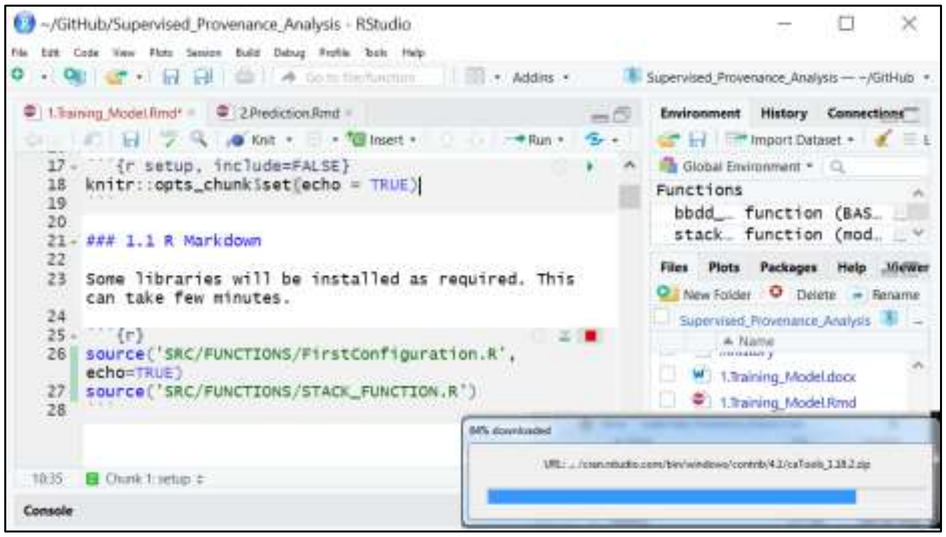
```{r}
table(PT$group)
```

  1 2 3 4 5 6 7
 33 36 33 37 36 33 84

1.5 Dataset partition

MachineLearning works using the 80% of the samples
```

```
78:16   Chunk 6   R Markdown
```

Figure UG3. The background allows the user to distinguish between: 1- Information/instructions (white). 2- Programmed actions/code (grey). 3- The results from the action (white with a thin grey line at the end before next information/instructions). The play button to run the current chunk of the code appears circled in red, while it is running Stop button appears at that place.

3.4 Installing all packages and running SPA software

The second chunk of the code within '1.Traning_model' can take some minutes to run. This will install all packages and libraries required.

To be sure that the task is being done, check if there appears a stop button where the play button was. This indicates that the application has not yet finished running that chunk.
```

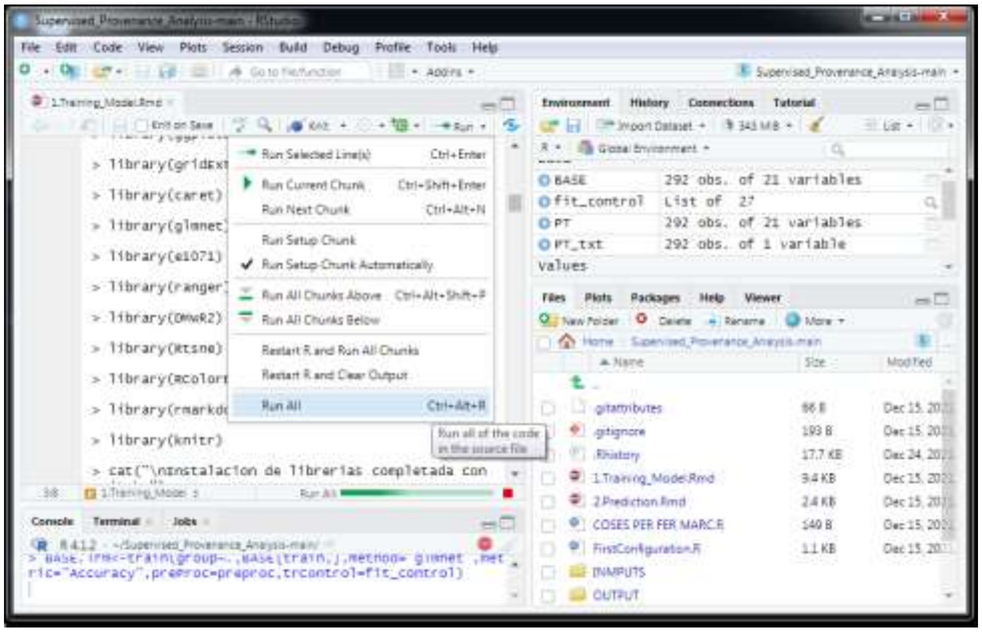
Chunk code 2



~~~
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

### 1.1 R Markdown

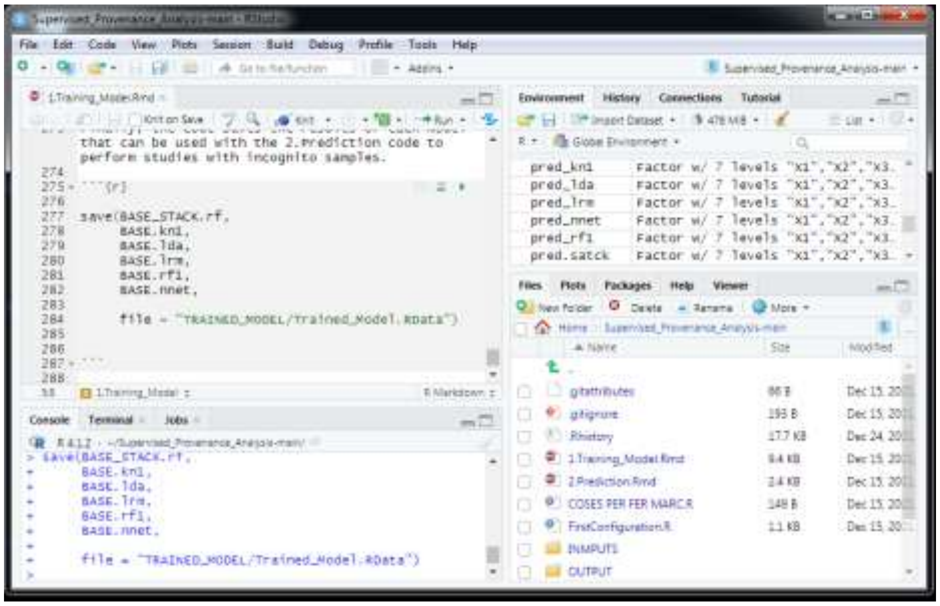
Some libraries will be installed as required. This
can take few minutes.

```{r}
source('SRC/FUNCTIONS/FirstConfiguration.R',
echo=TRUE)
source('SRC/FUNCTIONS/STACK_FUNCTION.R')
```
```

```
Functions
  bbdd_   function (BAS…
  stack_  function (mod…
```

Figure UG4. Screenshot showing the second chunk of the code being run.

Continue the process by reading and running all the information with white background until you have finished all the code without errors.

It is important to run all the chunks with the sequential order. If any of the chunks is omitted, some errors will appear along the code.

3.5 Another way to run all the code

There is an easier way to run the whole '1.Training_Model' code. Expand the "Run" tab and click at the end of the "Run all" line (or Ctrl+Alt+R).

This action will run all the code and the first time it will take from 4 to 10 minutes to complete depending on the PC used. Time will greatly decrease in subsequent executions.



```
> library(gridExt
> library(caret)
> library(glmnet)
> library(e1071)
> library(ranger)
> library(DMwR2)
> library(Rtsne)
> library(RColorB
> library(rmarkdo
> library(knitr)
> cat("\nInstalacion de librerias completada con
```

```
> BASE.irmc<-train(group~.,BASE[train,],method="glmnet",metric="Accuracy",preProc=preproc,trcontrol=fit_control)
```

```
BASE         292 obs. of 21 variables
fit_control  List of 27
PT           292 obs. of 21 variables
PT_txt       292 obs. of 1 variable
```

Run Selected Line(s) Ctrl+Enter
Run Current Chunk Ctrl+Shift+Enter
Run Next Chunk Ctrl+Alt+N
Run Setup Chunk
Run Setup Chunk Automatically
Run All Chunks Above Ctrl+Alt+Shift+P
Run All Chunks Below
Restart R and Run All Chunks
Restart R and Clear Output
Run All Ctrl+Alt+R
Run all of the code in the source file

Figure UG5: Expanded tab showing the 'Run All' option.
```

3.6 The classification model

The final chunk of the code will create a new file: Trained_Model.RData. This is the classification model. Read the next section to know how to interpret the results.

You can notice that the file has been created when the notification appears within the console (in the bottom-left window).



~~~
that can be used with the 2.Prediction code to
perform studies with incognito samples.

```{r}

save(BASE_STACK.rf,
     BASE.kn1,
     BASE.lda,
     BASE.lrm,
     BASE.rf1,
     BASE.nnet,

     file = "TRAINED_MODEL/Trained_Model.RData")


```
~~~

```
> save(BASE_STACK.rf,
+      BASE.kn1,
+      BASE.lda,
+      BASE.lrm,
+      BASE.rf1,
+      BASE.nnet,
+
+      file = "TRAINED_MODEL/Trained_Model.RData")
>
```

```
pred_kn1     Factor w/ 7 levels "X1","X2","X3...
pred_lda     Factor w/ 7 levels "X1","X2","X3...
pred_lrm     Factor w/ 7 levels "X1","X2","X3...
pred_mnet    Factor w/ 7 levels "X1","X2","X3...
pred_rf1     Factor w/ 7 levels "X1","X2","X3...
pred.satck   Factor w/ 7 levels "X1","X2","X3...
```

Name	Size	Modified
.gitattributes	66 B	Dec 15, 20
.gitignore	193 B	Dec 15, 20
.Rhistory	17.7 KB	Dec 24, 20
1.Training_Model.Rmd	9.4 KB	Dec 15, 20
2.Prediction.Rmd	2.4 KB	Dec 15, 20
COSES PER FER MARC.R	148 B	Dec 15, 20
FirstConfiguration.R	1.1 KB	Dec 15, 20
INMPUTS		
OUTPUT		

Supervised_Provenance_Analysis-main - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
1.Training_Model.Rmd
Knit on Save
Run
Console Terminal Jobs
R 4.1.2 · ~/Supervised_Provenance_Analysis-main/
Environment History Connections Tutorial
Import Dataset
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home › Supervised_Provenance_Analysis-main

Figure UG6: The Trained_Model.RData file has been generated as informed within the console

4. SUPERVISED PROVENANCE ANALYSIS app

After some explanations on R software, you are now ready to use the SPA tool to make your own studies. However, the first step is to fill the input Excel spreadsheets with your own data.

You will find them within the INPUTS folder:

-Modify the contents of DATABASE 1 MODEL.xlsx with your reference data before running 1.Training_Model.rmd (to train your model)

-Modify the contents of DABASE 2 PREDICTION.xlsx with the data of your samples of unknown provenance before running 2.Prediction.rmd (to produce probabilities to belong to the difference reference classes).

For the two spreadsheets:

-The first column is named “group”. Every sample within DATABASE 1 MODEL.xlsx has to be class-labeled to a specific group. Start from 1 and use as many numbers as groups in your database. In contrast let the column empty of unlabeled samples within DATABASE 2 PREDICTION.xlsx.

-The second column is named “Formula”. Here there is the individual tag of every sample. It is important to avoid duplicated tags in this column.

-From the third column onwards, there are the features that define every sample. In the published case study, these are the chemical weight % (i.e. the chemical composition) of different chemical elements (often expressed in form of oxides), but other variables could be used instead. It is important to use numbers for all the fields within these columns (use 0 for values below detection limits).

The rest of the present document shows all the contents that you will see within RStudio upon execution of the 1.Training_Model.rmd and 2.Prediction.rmd R Markdown documents.
Including 3 different types of texts: information / actions / results.

Information (white background)

Actions/functions (chunks of code with grey background)

Results (they appear directly after running the corresponding chunks, the results are shown with white background with a thin grey line at the end as shown above (section 3.3))

1.Training_Model

IMPORTANT NOTICE:

READ CAREFULLY all the text on white background as this contains INSTRUCTIONS

DO NOT MODIFY anything (especially if it is on grey background) unless specified, it is CODE

After executing each code chunk, the RESULT OF EXECUTION will appear in most occasions (on white background with a grey line separating it from the next instructions)

4.1 R Markdown installation

Some libraries will be installed as required. This can take few minutes.

```
source('SRC/FUNCTIONS/FirstConfiguration.R', echo=TRUE)

source('SRC/FUNCTIONS/STACK_FUNCTION.R')



##
## > cat("\n\n***\n INSTALANDO Y CARGANDO LIBRERIAS \n***\n")
##
##
## ***
##  INSTALANDO Y CARGANDO LIBRERIAS
## ***
##
## > packages_to_check = c("MASS", "randomForest", "readxl",
## +     "dummy", "kknn", "corrplot", "ggplot2", "gridExtra", "caret",
## +     "glmnet", "e1071 ..." ... [TRUNCATED]
##
## > packages_to_install = packages_to_check[!(packages_to_check %in%
## +     installed.packages()[, "Package"])]
##
## > if (length(packages_to_install)) install.packages(packages_to_install,
## +     repos = "http://cran.rstudio.com/", dependencies = TRUE)
##
## > rm(packages_to_check, packages_to_install)
##
## > library(MASS)
##
## > library(randomForest)
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## > library(readxl)
##
## > library(dummy)

## dummy 0.1.3

## dummyNews()

##
## > library(kknn)
##
## > library(corrplot)

## corrplot 0.84 loaded

##
## > library(ggplot2)

## Warning: As of rlang 0.4.0, dplyr must be at least version 0.8.0.
## * dplyr 0.7.6 is too old for rlang 0.4.5.
## * Please update dplyr to the latest version.
## * Updating packages on Windows requires precautions:
##   <https://github.com/jennybc/what-they-forgot/issues/62>

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

##
## > library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':
##
##     combine

##
## > library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```
##
## > library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

##
## > library(e1071)
##
## > library(ranger)

##
## Attaching package: 'ranger'

## The following object is masked from 'package:randomForest':
##
##     importance

##
## > library(DMwR)

## Loading required package: grid

##
## > library(Rtsne)
##
## > library(RColorBrewer)
##
## > library(rmarkdown)
##
## > library(knitr)
##
## > cat("\nInstalacion de librerias completada con exito\n")
##
## Instalacion de librerias completada con exito
```

4.2 Inputs

It is crucial to have modified the DATABASE 1 MODEL.xlsx file introducing your reference class-labeled data before running any chunk within 1.Training_Model.

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    292 obs. of  21 variables:
##  $ group: Factor w/ 7 levels "1","2","3","4",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ Na2O : num  0.322 0.258 0.217 0.427 0.363 0.318 0 0.519 0 0.332 ...
##  $ MgO  : num  1.83 2.02 2.36 1.92 1.47 2.61 2.08 2.12 2.53 1.57 ...
##  $ Al2O3: num  9.5 8.7 11.1 10.7 11.9 13.5 11.2 11.6 10.4 7.48 ...
##  $ SiO2 : num  56.6 48.4 53.8 56.5 72.4 67.3 53.3 57.8 50.2 39.8 ...
##  $ SO3  : num  0 0 0 0.0593 0 0.0115 0.131 0.225 0 0 ...
##  $ Cl   : num  0.082 0.0643 0.0815 0.0687 0.0846 0.105 0.0657 0.0895 0.0702
0.0559 ...
##  $ K2O  : num  4.02 3.71 5 3.49 3.55 5.04 4.44 3.47 4.1 2.52 ...
##  $ TiO2 : num  1.17 1.22 1.24 1.65 1.56 1.45 1.46 1.93 1.28 1.44 ...
##  $ Cr2O3: num  0.0126 0.012 0.0242 0.0317 0.0302 0.0266 0.0284 0.06 0.0249 0
.0488 ...
##  $ MnO  : num  0.0308 0.0374 0.0317 0.0544 0.0467 0.027 0.0605 0.0605 0.062
0.0532 ...
##  $ Fe2O3: num  4.99 6.19 8.53 8.76 7.33 7.92 10.4 8.75 8.76 7.41 ...
##  $ NiO  : num  0 0 0.0032 0.0053 0.0038 0.0043 0.0045 0.0121 0.003 0 ...
##  $ CuO  : num  0.0129 0.0076 0.0121 0.0102 0.0135 0.0067 0.0116 0.0179 0.009
8 0.0233 ...
##  $ ZnO  : num  0.0182 0.0154 0.0193 0.0164 0.0178 0.0229 0.0416 0.0221 0.022
5 0.0249 ...
##  $ Rb2O : num  0.0306 0.0319 0.0428 0.0228 0.0196 0.0353 0.0316 0.0224 0.030
7 0.0179 ...
##  $ SrO  : num  0.0288 0.0311 0.0276 0.0323 0.0173 0.0207 0.0363 0.031 0.0331
0.0403 ...
##  $ Y2O3 : num  0.0042 0.0049 0.0034 0.0051 0.0029 0.0082 0.0072 0.0065 0.004
5 0.0049 ...
##  $ ZrO2 : num  0.107 0.0926 0.0449 0.107 0.0993 0.0912 0.0546 0.137 0.0527 0
.0878 ...
##  $ Nb2O5: num  0.0016 0.0024 0.0025 0.0037 0.0017 0.0049 0.0042 0.0045 0.003
5 0.004 ...
##  $ BaO  : num  0 0.0008 0.0252 0 0 0.0256 0.231 0 0.0085 0.0078 ...
```

4.3 Checklist

Some basic verifications will be carried out below

Make sure that here do not appear “NA” or “duplicated sample”.

```
anyNA(PT)

## [1] FALSE
```

Make sure that there are no variables without values in the database. The expected result is: **FALSE**. If the result is TRUE, check your spreadsheet again.

```
anyDuplicated(PT)

## [1] 0
```

Make sure that there are not duplicated values on the dataset. The result should be 0 here.

4.4 Summary of input data

A summary of the number of groups and their samples can be seen below.

The accuracy should increase:

1. Having more samples in each group - *it means more information to learn how to classify.*
2. Having less number of groups - *it means that it will be easier to distinguish between 2 or 3 different groups than between 10*

```
table(PT$group)
```

```
##
##  1  2  3  4  5  6  7
## 33 36 33 37 36 33 84
```

4.5 Dataset partition

Supervised Machine Learning works using the 80% of the samples (randomly selected) to train the model and then the rest 20% is used to test the model and as a result of this an accuracy value is produced. The accuracy value indicates how successful has been the classification model.

In this case, the selection of samples is not completely random as we force also to select 80% of every single group to train the model. Sample selection within a group keeps being random.

Seed

The random selection is determined by a **seed**. To make sure that the accuracy of your model is robust, you can try different using different seeds which are defined using numbers (do not use 0). Accuracy probably will change using different seeds, but not so much if the model is robust enough.

set.seed (*change this number*)

```
set.seed(28)
train<-createDataPartition(
  PT$group,
  times = 1,
  p = 0.8,
  list = TRUE
)$Resample
```

4.6 Supervised models

Different classification models can be used. The models that have been included within the code are those described within the published manuscript. These selected models have been extensively and successfully used in different fields of science and technology.

All models have been optimized using the *fit_control* function.

```
fit_control<-trainControl(method="repeatedcv",number=10,repeats=2,savePredictio
ns="final",classProbs=TRUE)
preproc=c("center","scale")
```

Parameters to validate a model

Accuracy: Indicates the ratio between hits and fails obtained by the model in the test step. (the closer to 1, the better).

The confusion matrix gives more details as the ratio between hits and fails is given per each group in the test step. (a matrix with a diagonal full of 1 would mean a perfect performance).

Sensitivity: (True Positive Rate, TP/(TP+FN)) capacity to detect true positives. (the closer to 1, the better).

Specificity: (True Negative Rate, TN/ (TN+FP)) capacity to detect false positives (the closer to 1, the better)

Balance accuracy: It can be interpretated as the accuracy of each group (better when it is close from 1)

The description of models has been taken from Anglisano et al. 2020

4.6.1 Generalized linear models - Glmnet

Description
These are generalization models of a linear relationship between the output variable (class) and a set of input variables (features) where the distribution of the output variable can be non-normal and non-continuous and the function linking input and output variables can be more complex than a simple identity function. Specifically, the Glmnet algorithm incorporates regularization (i.e., reduction of variance) by the lasso and elastic-net methods to avoid overfitting (i.e., noise fitting).

```
BASE.lrm<-train(group~.,BASE[train,],method="glmnet",metric="Accuracy",preProc=
preproc,trControl=fit_control)
pred_lrm<-predict(BASE.lrm,BASE[-train,])
confusionMatrix(pred_lrm,BASE$group[-train])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction X1 X2 X3 X4 X5 X6 X7
##         X1  6  0  0  0  2  0  1
##         X2  0  7  0  0  1  0  0
##         X3  0  0  6  0  1  0  0
##         X4  0  0  0  6  0  0  0
##         X5  0  0  0  1  2  0  0
##         X6  0  0  0  0  0  6  1
##         X7  0  0  0  0  1  0 14
##
## Overall Statistics
##
##                Accuracy : 0.8545
##                  95% CI : (0.7334, 0.935)
##     No Information Rate : 0.2909
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8259
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: X1 Class: X2 Class: X3 Class: X4 Class: X5
## Sensitivity             1.0000    1.0000    1.0000    0.8571   0.28571
## Specificity             0.9388    0.9792    0.9796    1.0000   0.97917
## Pos Pred Value          0.6667    0.8750    0.8571    1.0000   0.66667
## Neg Pred Value          1.0000    1.0000    1.0000    0.9796   0.90385
## Prevalence              0.1091    0.1273    0.1091    0.1273   0.12727
## Detection Rate          0.1091    0.1273    0.1091    0.1091   0.03636
## Detection Prevalence    0.1636    0.1455    0.1273    0.1091   0.05455
## Balanced Accuracy       0.9694    0.9896    0.9898    0.9286   0.63244
##                      Class: X6 Class: X7
## Sensitivity             1.0000    0.8750
## Specificity             0.9796    0.9744
## Pos Pred Value          0.8571    0.9333
## Neg Pred Value          1.0000    0.9500
## Prevalence              0.1091    0.2909
## Detection Rate          0.1091    0.2545
## Detection Prevalence    0.1273    0.2727
## Balanced Accuracy       0.9898    0.9247
```

4.6.2 Random Forest - RF

Description
This algorithm is based on the concept of decision tree (a series of yes/no questions asked to the data that in the end lead to a predicted class). The RF model deals with many decision trees (i.e., a forest) using random sampling to build the trees and random subsets of features when splitting nodes of the trees.

```
BASE.rf1<-train(group~.,BASE[train,],method="ranger",metric="Accuracy",preProc=
preproc,trControl=fit_control)
pred_rf1<-predict(BASE.rf1,BASE[-train,])
confusionMatrix(pred_rf1,BASE$group[-train])

## Confusion Matrix and Statistics
##
##           Reference
## Prediction X1 X2 X3 X4 X5 X6 X7
##         X1  4  0  0  0  4  0  1
##         X2  1  6  0  0  0  0  0
##         X3  0  0  6  0  1  0  0
##         X4  0  0  0  7  0  1  0
##         X5  0  0  0  0  2  0  0
##         X6  1  0  0  0  0  5  1
##         X7  0  1  0  0  0  0 14
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.6703, 0.8957)
##     No Information Rate : 0.2909
##     P-Value [Acc > NIR] : 7.696e-15
##
##                   Kappa : 0.7607
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: X1 Class: X2 Class: X3 Class: X4 Class: X5
## Sensitivity            0.66667    0.8571    1.0000    1.0000   0.28571
## Specificity            0.89796    0.9792    0.9796    0.9792   1.00000
## Pos Pred Value         0.44444    0.8571    0.8571    0.8750   1.00000
## Neg Pred Value         0.95652    0.9792    1.0000    1.0000   0.90566
## Prevalence             0.10909    0.1273    0.1091    0.1273   0.12727
## Detection Rate         0.07273    0.1091    0.1091    0.1273   0.03636
## Detection Prevalence   0.16364    0.1273    0.1273    0.1455   0.03636
## Balanced Accuracy      0.78231    0.9182    0.9898    0.9896   0.64286
##                      Class: X6 Class: X7
## Sensitivity            0.83333    0.8750
## Specificity            0.95918    0.9744
## Pos Pred Value         0.71429    0.9333
## Neg Pred Value         0.97917    0.9500
## Prevalence             0.10909    0.2909
## Detection Rate         0.09091    0.2545
## Detection Prevalence   0.12727    0.2727
## Balanced Accuracy      0.89626    0.9247
```

4.6.3 Fit Neural Networks - ANN

Description
A mathematical mimic of human learning where individual processing elements are organized in layers. The input layer receives the weighted values of the features of an object to produce new values through so called activation functions; these values will be also weighted and transferred to new layers until reaching the output which is made of as many elements as classes. The obtained values are used to assign a class to the object.

```
BASE.nnet<-train(group~.,BASE[train,],method="nnet",metric="Accuracy",preProc=p
reproc,trControl=fit_control)
pred_nnet<-predict(BASE.nnet,BASE[-train,])

confusionMatrix(pred_nnet,BASE$group[-train])

## Confusion Matrix and Statistics
##
##           Reference
## Prediction X1 X2 X3 X4 X5 X6 X7
##         X1  5  0  0  0  2  1  1
##         X2  0  7  0  0  1  0  0
##         X3  0  0  6  0  1  0  0
##         X4  0  0  0  7  0  1  1
##         X5  0  0  0  0  2  0  0
##         X6  1  0  0  0  0  2  0
##         X7  0  0  0  0  1  2 14
##
## Overall Statistics
##
##                Accuracy : 0.7818
##                  95% CI : (0.6499, 0.8819)
##     No Information Rate : 0.2909
##     P-Value [Acc > NIR] : 6.967e-14
##
##                   Kappa : 0.7366
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: X1 Class: X2 Class: X3 Class: X4 Class: X5
## Sensitivity            0.83333    1.0000    1.0000    1.0000   0.28571
## Specificity            0.91837    0.9792    0.9796    0.9583   1.00000
## Pos Pred Value         0.55556    0.8750    0.8571    0.7778   1.00000
## Neg Pred Value         0.97826    1.0000    1.0000    1.0000   0.90566
## Prevalence             0.10909    0.1273    0.1091    0.1273   0.12727
## Detection Rate         0.09091    0.1273    0.1091    0.1273   0.03636
## Detection Prevalence   0.16364    0.1455    0.1273    0.1636   0.03636
## Balanced Accuracy      0.87585    0.9896    0.9898    0.9792   0.64286
##                      Class: X6 Class: X7
## Sensitivity            0.33333    0.8750
## Specificity            0.97959    0.9231
## Pos Pred Value         0.66667    0.8235
## Neg Pred Value         0.92308    0.9474
## Prevalence             0.10909    0.2909
## Detection Rate         0.03636    0.2545
## Detection Prevalence   0.05455    0.3091
## Balanced Accuracy      0.65646    0.8990
```

4.6.4 k-Nearest Neighbour - kknn

Description
Its basic idea is that a new object will be classified according to the class that have their k-nearest neighbors.

```
BASE.kn1<-train(group~.,BASE[train,],method="kknn",metric="Accuracy",preProc=pr
eproc,trControl=fit_control)

pred_kn1<-predict(BASE.kn1,BASE[-train,])
confusionMatrix(pred_kn1,BASE$group[-train])

## Confusion Matrix and Statistics
##
##           Reference
## Prediction X1 X2 X3 X4 X5 X6 X7
##         X1  4  0  0  0  1  0  0
##         X2  1  6  0  0  0  0  0
##         X3  1  0  6  0  0  0  0
##         X4  0  0  0  7  0  2  0
##         X5  0  0  0  0  1  1  1
##         X6  0  1  0  0  1  3  0
##         X7  0  0  0  0  4  0 15
##
## Overall Statistics
##
##                Accuracy : 0.7636
##                  95% CI : (0.6298, 0.8677)
##     No Information Rate : 0.2909
##     P-Value [Acc > NIR] : 5.693e-13
##
##                   Kappa : 0.7124
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: X1 Class: X2 Class: X3 Class: X4 Class: X5
## Sensitivity            0.66667    0.8571    1.0000    1.0000   0.14286
## Specificity            0.97959    0.9792    0.9796    0.9583   0.95833
## Pos Pred Value         0.80000    0.8571    0.8571    0.7778   0.33333
## Neg Pred Value         0.96000    0.9792    1.0000    1.0000   0.88462
## Prevalence             0.10909    0.1273    0.1091    0.1273   0.12727
## Detection Rate         0.07273    0.1091    0.1091    0.1273   0.01818
## Detection Prevalence   0.09091    0.1273    0.1273    0.1636   0.05455
## Balanced Accuracy      0.82313    0.9182    0.9898    0.9792   0.55060
##                      Class: X6 Class: X7
## Sensitivity            0.50000    0.9375
## Specificity            0.95918    0.8974
## Pos Pred Value         0.60000    0.7895
## Neg Pred Value         0.94000    0.9722
## Prevalence             0.10909    0.2909
## Detection Rate         0.05455    0.2727
## Detection Prevalence   0.09091    0.3455
## Balanced Accuracy      0.72959    0.9175
```

4.6.5 Linear discriminant analysis (LDA)

Description
Similarly to the PCA logic, delineates a new set of variables defined as linear combinations of the initial features reducing the dimensionality of the problem, but instead of looking for the maximum variance, LDA maximizes the separability among classes (the distance between their means) and simultaneously minimizes the internal scatter within each class.

```
BASE.lda<-train(group~.,BASE[train,],method="lda",metric="Accuracy",preProc=pre
proc,trControl=fit_control)
pred_lda<-predict(BASE.lda,BASE[-train,])
confusionMatrix(pred_lda,BASE$group[-train])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction X1 X2 X3 X4 X5 X6 X7
##         X1  6  0  0  0  2  0  1
##         X2  0  7  0  0  1  1  0
##         X3  0  0  6  0  1  0  0
##         X4  0  0  0  6  0  1  0
##         X5  0  0  0  0  2  1  0
##         X6  0  0  0  0  0  3  1
##         X7  0  0  0  1  1  0 14
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.6703, 0.8957)
##     No Information Rate : 0.2909
##     P-Value [Acc > NIR] : 7.696e-15
##
```

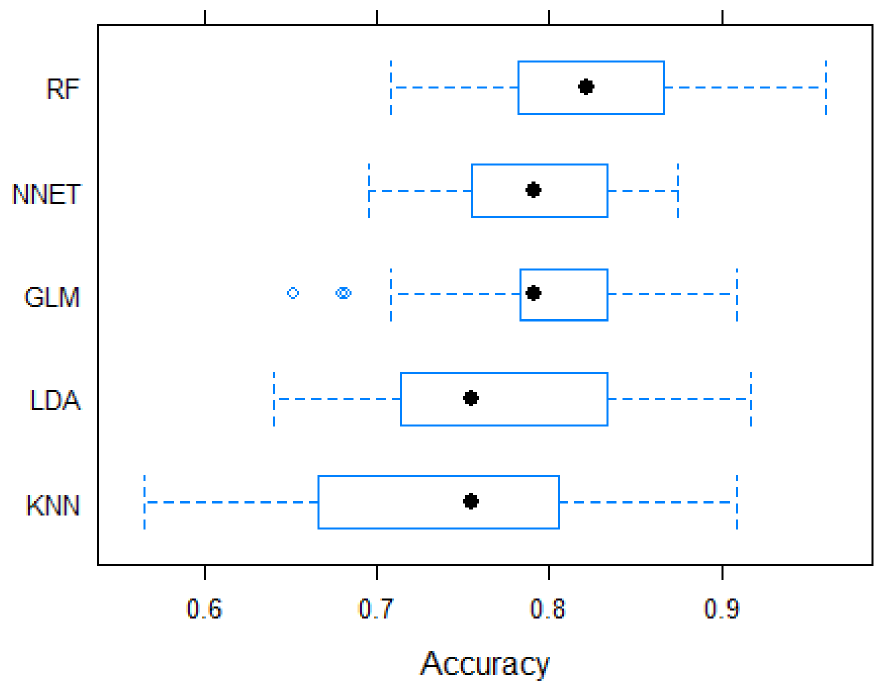
```
##                   Kappa : 0.7595
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: X1 Class: X2 Class: X3 Class: X4 Class: X5
## Sensitivity             1.0000    1.0000    1.0000    0.8571   0.28571
## Specificity             0.9388    0.9583    0.9796    0.9792   0.97917
## Pos Pred Value          0.6667    0.7778    0.8571    0.8571   0.66667
## Neg Pred Value          1.0000    1.0000    1.0000    0.9792   0.90385
## Prevalence              0.1091    0.1273    0.1091    0.1273   0.12727
## Detection Rate          0.1091    0.1273    0.1091    0.1091   0.03636
## Detection Prevalence    0.1636    0.1636    0.1273    0.1273   0.05455
## Balanced Accuracy       0.9694    0.9792    0.9898    0.9182   0.63244
##                      Class: X6 Class: X7
## Sensitivity            0.50000    0.8750
## Specificity            0.97959    0.9487
## Pos Pred Value         0.75000    0.8750
## Neg Pred Value         0.94118    0.9487
## Prevalence             0.10909    0.2909
## Detection Rate         0.05455    0.2545
## Detection Prevalence   0.07273    0.2909
## Balanced Accuracy      0.73980    0.9119
```

4.6.6 Stack of models

Description
With the aim of improving the performance of the classification, information from multiple models can be used to generate a new model (a model of models) using a random forest approach to the predictions from different models.

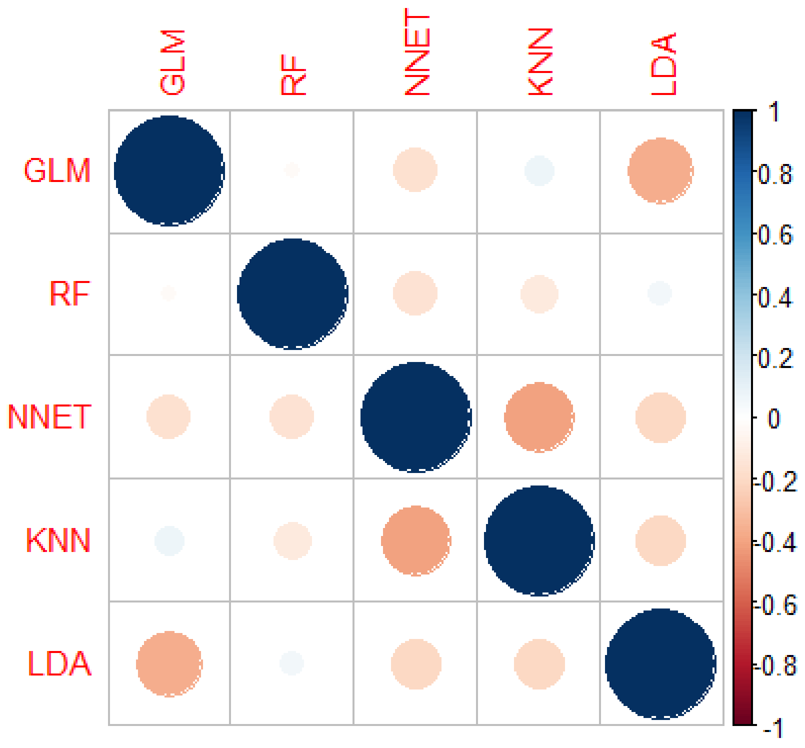
The next graph shows accuracy variability of all models. It uses the results from 10 random seeds. Make sure that there are not important variations with values. The models with smaller variability are more robust. Avoid the use (i.e. disregard) of a model with strong accuracy variations to classify unlabeled samples.

```
model_list<-list(GLM=BASE.lrm,RF=BASE.rf1,NNET=BASE.nnet,KNN=BASE.kn1,LDA=BASE.
lda)

res <- resamples(model_list)
bwplot(res,metric="Accuracy")
```



It is important to check that there is no correlation between the new variables (here, the different classification models).

```
model_cor<-modelCor(res)
model_cor

##              GLM          RF       NNET         KNN         LDA
## GLM   1.00000000 -0.02291717 -0.1603149  0.07201213 -0.36812539
## RF   -0.02291717  1.00000000 -0.1551214 -0.11814591  0.05021576
## NNET -0.16031490 -0.15512141  1.0000000 -0.40588241 -0.20201898
## KNN   0.07201213 -0.11814591 -0.4058824  1.00000000 -0.20359458
## LDA  -0.36812539  0.05021576 -0.2020190 -0.20359458  1.00000000

corrplot(model_cor)
```



Large great circles should appear only within the diagonal. If not, one of two correlated models should be deleted from the stack.

How to delete one model from stack? In the first chunk of code in section 2.6 "Stack of models" you will find the following action:

model_list<-list(GLM=BASE.lrm,RF=BASE.rf1,NNET=BASE.nnet,KNN=BASE.kn1,LDA=BASE.lda)

Let us assume, for example, that you need to remove the Random Forest model from the stack. To do so, remove "**RF=BASE.rf,**" within the action. Don't forget to delete only one coma.

The results of Stack of models are shown below.

```
BASE_STACK<-stack_function(model_list,BASE)
BASE_STACK.rf<-train(group~.,BASE_STACK[train,], method="rf",metric="Accuracy")
pred.satck<-predict(BASE_STACK.rf,BASE_STACK[-train,])
confusionMatrix(pred.satck,BASE$group[-train])

## Confusion Matrix and Statistics
##
##             Reference
## Prediction X1 X2 X3 X4 X5 X6 X7
##         X1  4  0  0  0  4  0  1
##         X2  1  6  0  0  0  0  0
##         X3  0  0  6  0  1  0  0
##         X4  0  0  0  7  0  1  0
##         X5  0  0  0  0  2  0  0
##         X6  1  0  0  0  0  5  1
##         X7  0  1  0  0  0  0 14
##
## Overall Statistics
##
```

```
##                Accuracy : 0.8
##                  95% CI : (0.6703, 0.8957)
##     No Information Rate : 0.2909
##     P-Value [Acc > NIR] : 7.696e-15
##
##                   Kappa : 0.7607
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: X1 Class: X2 Class: X3 Class: X4 Class: X5
## Sensitivity            0.66667    0.8571    1.0000    1.0000   0.28571
## Specificity            0.89796    0.9792    0.9796    0.9792   1.00000
## Pos Pred Value         0.44444    0.8571    0.8571    0.8750   1.00000
## Neg Pred Value         0.95652    0.9792    1.0000    1.0000   0.90566
## Prevalence             0.10909    0.1273    0.1091    0.1273   0.12727
## Detection Rate         0.07273    0.1091    0.1091    0.1273   0.03636
## Detection Prevalence   0.16364    0.1273    0.1273    0.1455   0.03636
## Balanced Accuracy      0.78231    0.9182    0.9898    0.9896   0.64286
##                      Class: X6 Class: X7
## Sensitivity            0.83333    0.8750
## Specificity            0.95918    0.9744
## Pos Pred Value         0.71429    0.9333
## Neg Pred Value         0.97917    0.9500
## Prevalence             0.10909    0.2909
## Detection Rate         0.09091    0.2545
## Detection Prevalence   0.12727    0.2727
## Balanced Accuracy      0.89626    0.9247
```

4.7 Saving the results

Finally, the code saves the training and test results of each model that can be used with the 2.Prediction code to perform studies with unlabeled samples.

```
save(BASE_STACK.rf,
     BASE.kn1,
     BASE.lda,
     BASE.lrm,
     BASE.rf1,
     BASE.nnet,

     file = "TRAINED_MODEL/Trained_Model.RData")
```

Here the train and test step finishes and to move to the production of cluster prediction open the file **2.Prediction.rmd** and follow its instructions

2.Prediction

IMPORTANT NOTICE:

To start the production of cluster predictions, first you must train the classification models. DO it by opening the file **1.Training _Model.rmd** and follow its instructions. Only if this is already done, follow instructions below.

4.8 Predictions

The first step is to install knitr and libraries necessaries to do provenance predictions using the results from the generated models

The actions above are needed to load the results from the trained models, some functions needed to make predictions and to load the database with samples of unknown provenance.

```
load("TRAINED_MODEL/Trained_Model.RData")

source('SRC/FUNCTIONS/STACK_FUNCTION.R')
source('SRC/FUNCTIONS/WRITE_FUNCTION.R')

ruta<-"INMPUTS/DATABASE 2 PREDICTION.xlsx"

PT <- read_excel(ruta, sheet = 1)
PT_txt<-PT[,2]
PT[,2]<-NULL

BASE<-PT
BASE$group<-NULL

head(read_excel(ruta, sheet = 1))

## # A tibble: 6 x 22
##   group Formula  Na2O   MgO Al2O3  SiO2    SO3     Cl   K2O  TiO2  Cr2O3
##   <lgl> <chr>   <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 NA    2020_0~ 0.596  1.72  13.4  66   0      0.0765  4.16  1.54 0.0654
## 2 NA    2020_0~ 0.581  1.8   12.5  65.8 0.0286 0.0679  4.5   1.59 0.0425
## 3 NA    2020_0~ 0.528  1.88  12.7  66.7 0      0.0688  4.43  1.61 0.038
## 4 NA    CM54    0.592  1.93  13.1  63.8 0      0.0919  4.53  1.51 0.0413
## 5 NA    CM55    0.645  1.88  12.8  66.5 0      0.0912  4.17  1.64 0.0415
## 6 NA    CMVP1   0.655  1.91  12.2  65.7 0      0.0897  4.46  1.72 0.0617
## # ... with 11 more variables: MnO <dbl>, Fe2O3 <dbl>, NiO <dbl>,
## #   CuO <dbl>, ZnO <dbl>, Rb2O <dbl>, SrO <dbl>, Y2O3 <dbl>, ZrO2 <dbl>,
## #   Nb2O5 <dbl>, BaO <dbl>
```

The head of the data from the document: **DATABASE 2 PREDICTION** can be seen on the results above.

The next chunk of code will make de predictions but the results will not be displayed below.

```
model_list<-list(GLM=BASE.lrm,RF=BASE.rf1,NNET=BASE.nnet,KNN=BASE.kn1,LDA=BASE.
lda)

BASE_STACK<-stack_function(model_list,BASE,inf_group=FALSE)
predicciorf1<-predict(object=BASE_STACK.rf,newdata = BASE_STACK)
probstak<-predict(object=BASE_STACK.rf,newdata=BASE_STACK,type="prob")

BASE_STACK_PROB<-stack_function_prob(model_list,BASE,inf_group=FALSE)



BASE_STACK_PROB_TOTAL<-as.data.frame(c(BASE_STACK[,!is.element(names(BASE_STACK
),"group")],BASE_STACK_PROB))
```

To produce the results, the following action will generate a .csv document with them.

```
BASE_IMPRIMIR<-bbdd_print(PT_txt,BASE_STACK_PROB_TOTAL,predicciorf1,probstak)
imprimeix(nom="prediction_results.csv",df=BASE_IMPRIMIR)
```

The results are now available on your computer, you will find a .csv document on the folder: /Supervised_Provenance_Analysis/OUTPUT

Have a look at them and Good luck!