SUPERVISED CLASSIFICATION IN R: RANDOM FOREST

INTRODUCTION

The objective of this exercise is to practice with machine learning supervised classification methods. The algorithm that we will use in this practice will be Random Forest.

MATERIAL

We will use Sentinel Sentinel-2 MSI images for the Natural Park of “Sierra de las Nieves”, which is in Malaga province (Figure 1).

Figure 1. Location of the study area.

Three images of different seasons will be used: spring, summer, and autumn for 2016. Specific dates are shown in table 1.

Table 1. Names and dates for the Sentinel-2 images

Name	Season	Date
S2A_L1C_20160329_T30SUF_.tif	Spring	25/03/2016
S2A_L1C_20160904_T30SUF_.tif	Summer	04/09/2016
S2A_L1C_20161220_T30SUF_.tif	Autumn	20/12/2016

The Sentinel 2-MSI images are made up of thirteen bands, with spatial resolutions between 10 and 60 meters, as shown in table 2. The bands at 60 meters of resolution have been excluded and the rest have been resampled to 20 meters. As a result, **for each date we have an image of 10 bands that correspond to the original bands 2, 3, 4, 5,6, 7, 8, 8A, 11 and 12, at 20 meters of spatial resolution** (table 2). Specifically, they correspond to the product at level 1C (geometric but not atmospheric correction) and being the values given in reflectivity at the top of the atmosphere.

Table2. Spectral bands and spatial resolution of the Sentinel-2A MSI system. In bold and shaded, the bands available for the exercise, resampled to 20 meters**.** Source: ESA (2015) https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/msi-instrument.

Band	Central wavelength	Spectral region	Original spatial resolution
1	443	Blue	60

2	**490**	**Blue-green**	**10**
3	**560**	**Green**	**10**
4	**665**	**Red**	**10**
5	**705**	**Red edge**	**20**
6	**740**	**Red edge**	**20**
7	**783**	**Near Infrared**	**20**
8	**842**	**Near Infrared**	**10**
8A	**865**	**Near Infrared**	**20**
9	945	Near Infrared	60
10	1380	Medium infrared	60
11	**1610**	**Short Wave Infrared**	**20**
12	**2190**	**Short Wave Infrared**	**20**

The three images are in tif format. Additionally, the following data are given:

- Random sampling with polygons for every category of the study area
- csv containing the spectral signatures of every polygon

INSTRUCTIONS

1. The first step is to load the libraries that we are going to use in the exercise.

```
#load libraries
library(raster)
library(dplyr)
library(mlr)
library(randomForest)
library(ggplot2)
library(clue)
```

2. It is important to fix the “seed” to allow replicability.

```
#defining the seed
set.seed(123, "L'Ecuyer")
```

3. We will also define the working directories.

```
#defining working directories
HDFpath<- "C:/COLOUR/MAPPING" # directory with data
setwd(HDFpath)              # fixing the directory
```

4. Once all this is done, we have our RStudio session ready to start working with the data. Open the shapefile with the training areas.

```
#opening vector file
ROIS<-shapefile("C:/COLOUR/MAPPING/ROIS.shp")
plot(ROIS)
```

If you type the name of the object where we have saved the information of the shapefile "ROIS" you will see the following information:

```
> ROIS
class       : SpatialPolygonsDataFrame
features    : 407
extent      : 302000, 341880, 4042200, 4082940  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=30 +ellps=WGS84 +units=m +no_defs
variables   : 5
names       :  Class_Name, Class_Id, Parts, Length, Area
min values  : caducifolio,        1,     1,    160, 1600
max values  :      urbano,        8,     1,    160, 1600
```

As you can see there is information about the class of the object, the extent, coordinate system, etc ... Note that the notation used by this library is different from the one we use. In this case we find two terms that for us are synonymous with different meanings: "features" and "variables". This library calls features to what are really observations (rows) and variables to features (columns).
Also, we see the names of the columns and the minimum and maximum value.

This shapefile contains the information about the class of each ROI in two different data types "string" or "int", or what is the same text or integers, respectively.

5. Open the file training.csv which contains the spectral signatures.

```
#reading the csv file with the spectral
training <- read.table("training.csv", header = TRUE, sep = ",")
```

6. Print the “training” object on the screen to check its content and that everything is OK. Show up to a maximum of 32 rows, omitting 375. To see only the header you can do the following:

```
> ls(training)
 [1] "Blue0"     "BlueP"     "BlueV"     "Green0"    "GreenP"    "GreenV"    "NIR20"     "NIR2P"     "NIR2V"     "NIR0"      "NIRP"      "NIRV"
[13] "RedEdge10" "RedEdge1P" "RedEdge1V" "RedEdge20" "RedEdge2P" "RedEdge2V" "RedEdge30" "RedEdge3P" "RedEdge3V" "Red0"      "RedP"      "RedV"
[25] "SWIR10"    "SWIR1P"    "SWIR1V"    "SWIR20"    "SWIR2P"    "SWIR2V"    "X"
```

"Ls" lists the features of the dataframe. There are all the bands of the multi-seasonal image and there is one more “X” feature. This characteristic contains the Id of each row. We must remove it as it does not contain information about satellite images and can confuse the classifier. To do this, run the following command:

```
#removing the column with rows IdS
training<-dplyr::select(training, -X)
```

7. So far we have all the information about class labels on the one hand (object: ROIS) and spectral signatures on the other (object: training). Incorporates the column with the class label into the training object and saves the result in a .csv file

```
#incoporating the class labels from the vectorial file into the training data frame
training<-cbind(training, Class_Id=ROIS$Class_Id)
#writing the new data fram with labels
write.csv(training, file="C:/COLOUR/MAPPING/Results/training_RF.csv")
```

8. Now print the “training” object to check that everything is OK.

```
> ls(training)
 [1] "Blue0"     "BlueP"     "BlueV"     "Class_Id"  "Green0"    "GreenP"    "GreenV"    "NIR20"     "NIR2P"     "NIR2V"     "NIR0"      "NIRP"
[13] "NIRV"      "RedEdge10" "RedEdge1P" "RedEdge1V" "RedEdge20" "RedEdge2P" "RedEdge2V" "RedEdge30" "RedEdge3P" "RedEdge3V" "Red0"      "RedP"
[25] "RedV"      "SWIR10"    "SWIR1P"    "SWIR1V"    "SWIR20"    "SWIR2P"    "SWIR2V"
```

9. It seems we have all the features included, but are they in the correct data format? To see the format of each feature we will do the following:

```
#checking the data type and basic stats of every feature
summary(training)
```

```
> summary(training)
     BlueV            GreenV             RedV          RedEdge1V        RedEdge2V      RedEdge3V        NIRV          NIR2V          SWIR1V
 Min.   : 795.5   Min.   : 677.0   Min.   : 495.0   Min.   : 667.8   Min.   :1185   Min.   :1360   Min.   :1308   Min.   :1499   Min.   : 686.2
 1st Qu.: 906.1   1st Qu.: 856.5   1st Qu.: 724.4   1st Qu.:1026.2   1st Qu.:1774   1st Qu.:2060   1st Qu.:1966   1st Qu.:2246   1st Qu.:1476.6
 Median :1121.2   Median :1129.2   Median :1212.5   Median :1444.2   Median :2055   Median :2346   Median :2252   Median :2600   Median :2297.2
 Mean   :1186.8   Mean   :1218.7   Mean   :1376.5   Mean   :1571.4   Mean   :2120   Mean   :2426   Mean   :2345   Mean   :2688   Mean   :2427.2
 3rd Qu.:1445.1   3rd Qu.:1518.8   3rd Qu.:1908.9   3rd Qu.:2036.9   3rd Qu.:2368   3rd Qu.:2738   3rd Qu.:2655   3rd Qu.:3054   3rd Qu.:3250.6
 Max.   :2013.8   Max.   :2421.8   Max.   :3425.8   Max.   :3603.0   Max.   :3963   Max.   :4410   Max.   :4278   Max.   :4920   Max.   :5129.2
     SWIR2V           BlueP            GreenP             RedP          RedEdge1P         RedEdge2P      RedEdge3P        NIRP          NIR2P
 Min.   : 316.8   Min.   : 740.0   Min.   : 555.0   Min.   : 366.5   Min.   : 519.2   Min.   :1038   Min.   :1199   Min.   :1202   Min.   :1273
 1st Qu.: 755.1   1st Qu.: 858.9   1st Qu.: 755.1   1st Qu.: 497.8   1st Qu.: 859.1   1st Qu.:1660   1st Qu.:1884   1st Qu.:1891   1st Qu.:2039
 Median :1519.2   Median : 991.8   Median : 938.2   Median : 844.5   Median :1165.8   Median :2028   Median :2294   Median :2261   Median :2446
 Mean   :1549.0   Mean   :1109.5   Mean   :1058.7   Mean   :1015.1   Mean   :1325.4   Mean   :2196   Mean   :2511   Mean   :2502   Mean   :2692
 3rd Qu.:2245.4   3rd Qu.:1305.2   3rd Qu.:1306.1   3rd Qu.:1407.2   3rd Qu.:1722.8   3rd Qu.:2739   3rd Qu.:3019   3rd Qu.:3006   3rd Qu.:3218
 Max.   :3821.2   Max.   :2203.0   Max.   :2298.2   Max.   :2565.0   Max.   :2794.8   Max.   :3947   Max.   :4774   Max.   :4699   Max.   :5140
     SWIR1P           SWIR2P            BlueO            GreenO             RedO          RedEdge1O        RedEdge2O        RedEdge3O          NIRO
 Min.   : 602.5   Min.   : 296.2   Min.   : 768.0   Min.   : 456.8   Min.   : 253.2   Min.   : 251.2   Min.   : 305.2   Min.   : 276.8   Min.   : 244.5
 1st Qu.:1379.1   1st Qu.: 669.2   1st Qu.: 952.5   1st Qu.: 734.8   1st Qu.: 519.5   1st Qu.: 770.4   1st Qu.:1312.9   1st Qu.:1428.1   1st Qu.:1411.0
 Median :1982.0   Median :1304.0   Median :1108.0   Median : 953.5   Median : 918.5   Median :1162.2   Median :1762.0   Median :1936.2   Median :1923.8
 Mean   :2100.3   Mean   :1415.4   Mean   :1203.4   Mean   :1051.1   Mean   :1028.5   Mean   :1249.3   Mean   :1794.1   Mean   :1983.0   Mean   :1980.0
 3rd Qu.:2714.6   3rd Qu.:1952.8   3rd Qu.:1344.8   3rd Qu.:1258.5   3rd Qu.:1337.2   3rd Qu.:1583.6   3rd Qu.:2227.6   3rd Qu.:2500.2   3rd Qu.:2493.5
 Max.   :5157.5   Max.   :4006.2   Max.   :4082.8   Max.   :3981.5   Max.   :4464.2   Max.   :4811.5   Max.   :5165.0   Max.   :5213.5   Max.   :5346.5
     NIR2O            SWIR1O            SWIR2O           Class_Id
 Min.   : 231.2   Min.   :  80.25   Min.   :  39.75   Min.   :1.000
 1st Qu.:1520.2   1st Qu.:1071.75   1st Qu.: 551.00   1st Qu.:3.000
 Median :2071.0   Median :1846.25   Median :1181.00   Median :5.000
 Mean   :2141.2   Mean   :1891.30   Mean   :1284.42   Mean   :4.541
 3rd Qu.:2685.2   3rd Qu.:2426.75   3rd Qu.:1735.50   3rd Qu.:7.000
 Max.   :5389.8   Max.   :6122.25   Max.   :4732.50   Max.   :8.000
```

The feature "Class_Id" contains integers. Therefore, R considers it to be of type numeric, when in fact it contains the class labels and should be categorical.

Convert the Class_Id characteristic into a categorical type so that we can do the classification.

```
#converting the class labels in categorical
training$Class_Id<-as.factor(training$Class_Id)
```

Make a “summary” again and check if Class_Id has changed

```
> summary(training)
     BlueV            GreenV             RedV          RedEdge1V        RedEdge2V      RedEdge3V        NIRV          NIR2V          SWIR1V
 Min.   : 795.5   Min.   : 677.0   Min.   : 495.0   Min.   : 667.8   Min.   :1185   Min.   :1360   Min.   :1308   Min.   :1499   Min.   : 686.2
 1st Qu.: 906.1   1st Qu.: 856.5   1st Qu.: 724.4   1st Qu.:1026.2   1st Qu.:1774   1st Qu.:2060   1st Qu.:1966   1st Qu.:2246   1st Qu.:1476.6
 Median :1121.2   Median :1129.2   Median :1212.5   Median :1444.2   Median :2055   Median :2346   Median :2252   Median :2600   Median :2297.2
 Mean   :1186.8   Mean   :1218.7   Mean   :1376.5   Mean   :1571.4   Mean   :2120   Mean   :2426   Mean   :2345   Mean   :2688   Mean   :2427.2
 3rd Qu.:1445.1   3rd Qu.:1518.8   3rd Qu.:1908.9   3rd Qu.:2036.9   3rd Qu.:2368   3rd Qu.:2738   3rd Qu.:2655   3rd Qu.:3054   3rd Qu.:3250.6
 Max.   :2013.8   Max.   :2421.8   Max.   :3425.8   Max.   :3603.0   Max.   :3963   Max.   :4410   Max.   :4278   Max.   :4920   Max.   :5129.2

     SWIR2V           BlueP            GreenP             RedP          RedEdge1P         RedEdge2P      RedEdge3P        NIRP          NIR2P
 Min.   : 316.8   Min.   : 740.0   Min.   : 555.0   Min.   : 366.5   Min.   : 519.2   Min.   :1038   Min.   :1199   Min.   :1202   Min.   :1273
 1st Qu.: 755.1   1st Qu.: 858.9   1st Qu.: 755.1   1st Qu.: 497.8   1st Qu.: 859.1   1st Qu.:1660   1st Qu.:1884   1st Qu.:1891   1st Qu.:2039
 Median :1519.2   Median : 991.8   Median : 938.2   Median : 844.5   Median :1165.8   Median :2028   Median :2294   Median :2261   Median :2446
 Mean   :1549.0   Mean   :1109.5   Mean   :1058.7   Mean   :1015.1   Mean   :1325.4   Mean   :2196   Mean   :2511   Mean   :2502   Mean   :2692
 3rd Qu.:2245.4   3rd Qu.:1305.2   3rd Qu.:1306.1   3rd Qu.:1407.2   3rd Qu.:1722.8   3rd Qu.:2739   3rd Qu.:3019   3rd Qu.:3006   3rd Qu.:3218
 Max.   :3821.2   Max.   :2203.0   Max.   :2298.2   Max.   :2565.0   Max.   :2794.8   Max.   :3947   Max.   :4774   Max.   :4699   Max.   :5140

     SWIR1P           SWIR2P            BlueO            GreenO             RedO          RedEdge1O        RedEdge2O        RedEdge3O          NIRO
 Min.   : 602.5   Min.   : 296.2   Min.   : 768.0   Min.   : 456.8   Min.   : 253.2   Min.   : 251.2   Min.   : 305.2   Min.   : 276.8   Min.   : 244.5
 1st Qu.:1379.1   1st Qu.: 669.2   1st Qu.: 952.5   1st Qu.: 734.8   1st Qu.: 519.5   1st Qu.: 770.4   1st Qu.:1312.9   1st Qu.:1428.1   1st Qu.:1411.0
 Median :1982.0   Median :1304.0   Median :1108.0   Median : 953.5   Median : 918.5   Median :1162.2   Median :1762.0   Median :1936.2   Median :1923.8
 Mean   :2100.3   Mean   :1415.4   Mean   :1203.4   Mean   :1051.1   Mean   :1028.5   Mean   :1249.3   Mean   :1794.1   Mean   :1983.0   Mean   :1980.0
 3rd Qu.:2714.6   3rd Qu.:1952.8   3rd Qu.:1344.8   3rd Qu.:1258.5   3rd Qu.:1337.2   3rd Qu.:1583.6   3rd Qu.:2227.6   3rd Qu.:2500.2   3rd Qu.:2493.5
 Max.   :5157.5   Max.   :4006.2   Max.   :4082.8   Max.   :3981.5   Max.   :4464.2   Max.   :4811.5   Max.   :5165.0   Max.   :5213.5   Max.   :5346.5

     NIR2O            SWIR1O            SWIR2O           Class_Id
 Min.   : 231.2   Min.   :  80.25   Min.   :  39.75   7      :57
 1st Qu.:1520.2   1st Qu.:1071.75   1st Qu.: 551.00   5      :51
 Median :2071.0   Median :1846.25   Median :1181.00   1      :50
 Mean   :2141.2   Mean   :1891.30   Mean   :1284.42   2      :50
 3rd Qu.:2685.2   3rd Qu.:2426.75   3rd Qu.:1735.50   3      :50
 Max.   :5389.8   Max.   :6122.25   Max.   :4732.50   4      :50
                                                      (Other):99
```

10. Once you have the file ready for training, you can start training the classifiers. The steps to follow depend on the specific library to be used. In our case we will use the "mlr" library as it is currently the most complete library with machie learning algorithms https://mlr.mlr-org.com/.

    The first thing this library requires is to set up a task, where we must specify the type of task (classification, regression, clustering ...), the training data, and the feature that contains the class label.

```
#creating the task
clasificacion.task <- makeClassifTask(id = "nieves", data = training, target = "Class_Id")
```

11. The next step would be to tune the hyperparameter range of the model. In the case of Random Forest, we only need two hyperparameters to optimise it: the number of random features (mtry) and the number of trees in the forest (ntree). If we tried to apply another

classifier we could have a look to the help and see the parameters that need to be optimised for that specific algorithm. You can see the information regarding the algorithms included in the mlr library at: https://mlr.mlr-org.com/articles/tutorial/integrated_learners.html

Set up the range of variation of hyperparameters, and the way to evaluate the best combination of them. It is proposed to use an evaluation based on a cross validation of 10 folders, using the classification error (mmce) as a metric.

```
#tuning the hyperparameters; mtry is tipically close to sqrt(número características)
ps.rf <- makeParamSet(makeDiscreteParam("mtry", values = (3:6)),
                      makeDiscreteParam("ntree", values = c(1000, 5000, 10000)))

ctrl <- makeTuneControlGrid()
rdesc <- makeResampleDesc("CV", iters = 10)
res <- tuneParams("classif.randomForest", task = clasificacion.task,
                  resampling = rdesc, par.set = ps.rf, measures = mmce,  control = ctrl)
res
```

```
[Tune] Started tuning learner classif.randomForest for parameter set:
          Type len Def          Constr Req Tunable Trafo
mtry  discrete   -   -         3,4,5,6   -    TRUE     -
ntree discrete   -   - 1000,5000,10000   -    TRUE     -
With control class: TuneControlGrid
Imputation value: 1
[Tune-x] 1: mtry=3; ntree=1000
[Tune-y] 1: mmce.test.mean=0.1645732; time: 0.1 min
[Tune-x] 2: mtry=4; ntree=1000
[Tune-y] 2: mmce.test.mean=0.1621341; time: 0.1 min
[Tune-x] 3: mtry=5; ntree=1000
[Tune-y] 3: mmce.test.mean=0.1596341; time: 0.1 min
[Tune-x] 4: mtry=6; ntree=1000
[Tune-y] 4: mmce.test.mean=0.1645732; time: 0.1 min
[Tune-x] 5: mtry=3; ntree=5000
[Tune-y] 5: mmce.test.mean=0.1670732; time: 0.4 min
[Tune-x] 6: mtry=4; ntree=5000
[Tune-y] 6: mmce.test.mean=0.1621341; time: 0.4 min
[Tune-x] 7: mtry=5; ntree=5000
[Tune-y] 7: mmce.test.mean=0.1620732; time: 0.4 min
[Tune-x] 8: mtry=6; ntree=5000
[Tune-y] 8: mmce.test.mean=0.1645732; time: 0.4 min
[Tune-x] 9: mtry=3; ntree=10000
[Tune-y] 9: mmce.test.mean=0.1670732; time: 0.7 min
[Tune-x] 10: mtry=4; ntree=10000
[Tune-y] 10: mmce.test.mean=0.1670122; time: 0.7 min
[Tune-x] 11: mtry=5; ntree=10000
[Tune-y] 11: mmce.test.mean=0.1670122; time: 0.7 min
[Tune-x] 12: mtry=6; ntree=10000
[Tune-y] 12: mmce.test.mean=0.1646341; time: 0.8 min
[Tune] Result: mtry=5; ntree=1000 : mmce.test.mean=0.1596341
> res
Tune result:
Op. pars: mtry=5; ntree=1000
mmce.test.mean=0.1596341
```

In this case we have tested 12 different Random Forest models. The best model is the one that considers 5 random variables in each tree and 1000 classification trees, with a classification error of 15.96%. This means that the overall accuracy is 84.04%.

12. We would have already decided the optimal configuration of the algorithm. Now you have to set the parameters and create an object with the trained model.

```
#setting hyperparameters and training the model
lrn <- setHyperPars(makeLearner("classif.randomForest", predict.type = "prob"), par.vals = res$x, importance=TRUE)
modelo.rf <- train(lrn, clasificacion.task)
```

Note that the name given by mlr to the Random Forest algorithm for classification is “classif.randomForest”. We have specified that the prediction type will be the probability of class membership (prob).

Check the information about the model "model.rf" by directly printing its name on the screen and using the "summary" command.

```
> model.rf
Model for learner.id=classif.randomForest; learner.class=classif.randomForest
Trained on: task.id = nieves; obs = 407; features = 30
Hyperparameters: mtry=6,ntree=1e+03,importance=TRUE
> summary(model.rf)
              Length Class                Mode
learner        16    classif.randomForest list
learner.model  19    randomForest.formula list
task.desc      13    ClassifTaskDesc      list
subset        407    -none-               numeric
features       30    -none-               character
factor.levels   1    -none-               list
time            1    -none-               numeric
dump            0    -none-               NULL
```

Inside model.rf there is an object which is where the model and almost all the information about it are actually stored. Take a look at what it contains by listing model.rf $ learner.model:

```
#checking model information
ls(model.rf$learner.model)
```

```
> #checking model information
> ls(model.rf$learner.model)
 [1] "call"          "classes"       "confusion"       "err.rate"      "forest"        "importance"
 [7] "importanceSD"  "inbag"         "localImportance" "mtry"          "ntree"         "oob.times"
[13] "predicted"     "proximity"     "terms"           "test"          "type"          "votes"
[19] "y"
```

There is a lot of valuable information inside model.rf $ learner.model such as: the confusion matrix, the importance of each feature in the model training (importance), the predicted class for each training area (predicted), and the observed class of each training area (y). Have a look to it:

```
#printing on screen model information
model.rf$learner.model$confusion
model.rf$learner.model$importance
```

```
> #printing on screen model information
> model.rf$learner.model$confusion
   1  2  3  4  5  6  7  8 class.error
1 46  0  1  1  0  0  0  2  0.08000000
2  0 45  0  0  0  0  5  0  0.10000000
3  0  0 40  0  0  0  4  6  0.20000000
4  0  0  1 40  8  0  0  1  0.20000000
5  0  0  0  5 44  0  0  2  0.13725490
6  0  0  2  0  0 47  0  0  0.04081633
7  0  7  3  0  0  1 42  4  0.26315789
8  1  0  4  0  2  0  2 41  0.18000000
```

	1	2	3	4	5	6	7	8
BlueV	0.043028647	0.069378434	0.036511938	0.044059303	0.1235525656	0.0543019014	0.041889881	0.024915587
GreenV	0.022584545	0.028987740	0.021403884	0.020516880	0.2073281492	0.0456767264	0.027710675	0.021464908
RedV	0.074619353	0.076973729	0.083265551	0.047877086	0.1331215415	0.0877898495	0.057496104	0.062504380
RedEdge1V	0.031139952	0.037317000	0.042994488	0.020959875	0.1248690400	0.0462460185	0.026792750	0.032592375
RedEdge2V	0.042829379	0.006912830	0.018206342	0.034145461	0.0274331071	0.0027272070	0.009139401	0.036174920
RedEdge3V	0.071101908	0.010400065	0.022051652	0.056027350	0.0123509059	0.0020144635	0.010730300	0.039017824
NIRV	0.057498362	0.025171588	0.029811043	0.056801167	0.0081667015	0.0012936699	0.019358956	0.029388957
NIR2V	0.045642174	0.029240751	0.033196522	0.067461149	0.0130202299	0.0008711282	0.027973307	0.025683475
SWIR1V	0.038426043	0.049795942	0.082276786	0.108132820	0.0853444469	0.0394234673	0.046363245	0.032509639
SWIR2V	0.095657274	0.063985491	0.070446049	0.103937284	0.1039711546	0.0798207413	0.065178742	0.041853544
BlueP	0.030022980	0.204459315	0.073889443	0.032084381	0.0637099177	0.0506387338	0.104706596	0.056383704
GreenP	0.010080761	0.118343157	0.042909217	0.023985148	0.0304660601	0.0269758057	0.074223758	0.030492966
RedP	0.040137184	0.329106343	0.104290210	0.058587057	0.0878070003	0.0818404412	0.130620409	0.071321287
RedEdge1P	0.012026730	0.167772941	0.057143168	0.038660629	0.0549130526	0.0279299208	0.049082788	0.033224795
RedEdge2P	0.021411681	0.017997761	0.053838447	0.018945029	0.0212000358	0.0421809238	0.015792450	0.050195927
RedEdge3P	0.026493228	0.019801059	0.052266167	0.025322059	0.0274954809	0.0671364462	0.018967465	0.043397260
NIRP	0.027567709	0.019517934	0.043930109	0.013048619	0.0371901694	0.0641798531	0.024518161	0.029450382
NIR2P	0.027239171	0.018624857	0.046069751	0.017740523	0.0293627632	0.0701396556	0.026546943	0.032354250
SWIR1P	0.007965678	0.041448955	0.059512935	0.083556197	0.0227037846	0.0338102738	0.038965676	0.019219583
SWIR2P	0.042317354	0.129173865	0.134273908	0.110794708	0.1198025621	0.1537929468	0.135643823	0.068274727
BlueO	0.006837430	0.019765261	0.001826641	0.033335073	0.0054026166	0.0130847968	0.021152260	0.010911276
GreenO	0.006346834	0.006051840	0.006123238	0.004958700	0.0150625450	0.0291213018	0.036328321	0.007395626
RedO	0.006071140	0.019621723	0.008642811	0.038032623	0.0341186622	0.0441413979	0.019890350	0.010267888
RedEdge1O	0.001896185	0.004217398	0.002351722	0.003122954	0.0101983068	0.0083852606	0.016135088	0.003099480
RedEdge2O	0.023479087	0.004481132	0.007477263	0.011565938	0.0121325104	0.0024018191	0.010850030	0.017460041
RedEdge3O	0.023545970	0.005874679	0.008471613	0.017862927	0.0114296002	0.0019544994	0.011093939	0.022169674
NIRO	0.017938696	0.018655109	0.014930165	0.012623498	0.0063752692	0.0050405642	0.050262022	0.030588473
NIR2O	0.015971791	0.023798557	0.017481712	0.021053380	0.0114813874	0.0037901606	0.053596934	0.032980578
SWIR1O	0.002952746	0.012421316	0.007506724	0.033038603	-0.0024214845	0.0021412947	0.071835496	0.009889677
SWIR2O	0.001420910	0.002067523	0.005724446	0.019740827	-0.0001103657	0.0054839603	0.065275124	0.009216152

What are the best and worst ranked classes? What classes are confused with each other? What are the most important variables in the general model? Which are the most important for each category? Are there differences between the importance derived from "mean decrease in accuracy" and "mean decrease in Gini"?

13. Get ready the satellite image to apply the model. Open the multiband tiff image "multiestacional.tif" and rename the bands:

```
#opening the multiband tif
multiseasonal<-brick("multiseasonal.tif")
names(multiseasonal)<-c("BlueV", "GreenV", "RedV", "RedEdge1V", "RedEdge2V",
                        "RedEdge3V", "NIRV", "NIR2V", "SWIR1V", "SWIR2V",
                        "BlueP", "GreenP", "RedP", "RedEdge1P", "RedEdge2P",
                        "RedEdge3P", "NIRP", "NIR2P", "SWIR1P", "SWIR2P",
                        "BlueO", "GreenO", "RedO", "RedEdge1O", "RedEdge2O",
                        "RedEdge3O", "NIRO", "NIR2O", "SWIR1O", "SWIR2O")
```

14. The last step of the automatic classification would be to apply the trained model to the "multi-seasonal" satellite image. To do this, we must convert the image into a dataframe first, since the models obtained by the mlr library can only be applied to dataframe. Then we will have to convert the prediction into a raster image.

```
#applying the model to predict the map
new_data=as.data.frame(as.matrix(multiseasonal))
pred.rf<-predict(model.rf, newdata=new_data)
mapa.rf = multiseasonal[[1]]
mapa.rf[] = pred.rf$data$response
mapa.rf
```

15. Don't forget to save the model and the map.

```
#saving models
save(model.rf, file="C:/COLOUR/MAPPING/Results/RandomForest_model.Rdata")
#saving the map in tif format
writeRaster(mapa.rf, filename = "C:/COLOUR/MAPPING/Results/RandomForest.tif",
            format="GTiff", datatype = "FLT4S", overwrite = TRUE)
```

Try to perform a classification with the classification algorithms: classification trees (rpart package), artificial neural networks and Support vector machines consulting the web about the different classifiers of the mlr package. Compare the accuracy of the models and maps obtained.
For the optimization of the hyperparameters you can follow the following reference:

Rodriguez-Galiano, V.F. and M. Chica-Rivas, Evaluation of different machine learning methods for land cover mapping of a Mediterranean area using multi-seasonal Landsat images and Digital Terrain Models. International Journal of Digital Earth, 2014. 7(6): p. 492-509