Random Forest with Sentinel 1 & Planetscope

WETLAND MAPPING USING RANDOM FOREST CLASSIFIER WITH PLANETSCOPE AND SENTINEL 1 IMAGES

```
#load libraries
library(raster)
library(randomForest)
library(sp)
library(rgdal)
library(ggplot2)
library(caret)
set.seed(123)

#==============================================================================
# Image dataset preparation
#==============================================================================

setwd("D:/Andi/01 ITC University of Twente/05 Quartile 5/Advanced Image Analysis/Project
Assignement/R_data/Satellite_data")

###### Import images
Blue_band = "B2.tif"
Green_band = "B4.tif"
Red_band = "B6.tif"
NIR_band = "B8.tif"
NDVI  = "NDVI.tif"
NDWI  = "NDWI.tif"
S1_VV = "S1_VV.tif"
S1_VH = "S1_VH.tif"

inraster        = stack(Blue_band, Green_band, Red_band, NIR_band, NDVI, NDWI, S1_VV,
S1_VH)
inraster

names(inraster) = c('Blue_band', 'Green_band', 'Red_band', 'NIR_band', 'NDVI', 'NDWI',
'S1_VV', 'S1_VH')

# Remove NA values from the raster stack
inraster_no_na <- reclassify(inraster, cbind(NA, -999))
inraster_no_na
#==============================================================================
# Import training and validation data
#==============================================================================
# Setting-up the working directory
setwd("D:/Andi/01 ITC University of Twente/05 Quartile 5/Advanced Image Analysis/Project
Assignement/R_data/Samples")

trainingData  =  shapefile("Training_point.shp")

testingData = shapefile("Testing_point.shp")

#==============================================================================
# Extract raster values for the training samples
#==============================================================================
training_data  = extract(inraster_no_na, trainingData)
training_response = as.factor(trainingData$Class)

#==============================================================================
#Select the number of input variables(i.e. predictors, features)
#==============================================================================
selection<-  c(1:8) # 8 is from the number of images
training_predictors = training_data[,selection]

#==============================================================================
# Train the random forest
#==============================================================================
```

```
ntree = 300    #number of trees to produce per iteration
mtry = 3       # number of variables used as input to split the variables
r_forest = randomForest(training_predictors, y=training_response, mtry=mtry, ntree =
ntree, keep.forest=TRUE, importance = TRUE, proximity=TRUE)

#================================================================================
#Investigate the OOB (Out-Of-the bag) error
#================================================================================
oob_errors <- as.vector(r_forest$err.rate[, 1])  # Extract OOB error rates
print(oob_errors)

# Set par() parameters to adjust plot margins
par(mar = c(5, 5, 4, 2) + 0.1)

# Plot the OOB Error Rate vs. Number of Trees
plot(x = 1:ntree, y = oob_errors, type = "l", xlab = "Number of Trees", ylab = "OOB Error
Rate",
     main = "OOB Error Rate vs. Number of Trees")

#================================================================================
# Assessment of variable importance
#================================================================================
imp =importance(r_forest)  #for ALL classes individually
imp                        #display importance output in console
varImpPlot(r_forest)
varUsed(r_forest)
importance(r_forest)

#================================================================================================
# Classify the entire image
#================================================================================================
predictor_data = subset(inraster_no_na, selection)
setwd("D:/Andi/01 ITC University of Twente/05 Quartile 5/Advanced Image Analysis/Project
Assignement/R_data/Result")#change this to your output directory if different

predictions = predict(predictor_data, r_forest, format=".tif", overwrite=TRUE,
progress="text", type="response")
# Specify the color palette for the map plot
color_palette <- c("yellow", "blue", "green")

plot(predictions, col = color_palette)
print(predictions)

#================================================================================================
# Assess the classification accuracy
#================================================================================================
Testing=extract(predictions, testingData) # extracts the value of the classified raster at
the validation point locations
Testing
Numeric=as.numeric(as.factor(testingData$Class))
Numeric
conf_mat = confusionMatrix(as.factor(Testing), as.factor(Numeric))
conf_mat

#================================================================================================
# Save the classification results
#================================================================================================

Class_Results = writeRaster(predictions, 'classified_image_andi.tif', overwrite=TRUE, col
= your_colormap)
```