XGBoost with
Sentinel 1 & Sentinel 2

XGBoost with Sentinel 2 and Sentinel 1 Bands

```
#load libraries
library(raster)
library(sp)
library(sf)
library(rgdal)
library (caret)
library(xgboost)
library(rBayesianOptimization)
set.seed(123)

# Setting-up the working directory
setwd("E:/ITC_Twente/Quarter_05_September_2023/00_Advanced_Image_Analysis/XX_Group_Project/04 Model Operation/Finalized_Tiff_Files_01")

###### Import images
B2  = "B2.tif"
B3  = "B3.tif"
B4  = "B4.tif"
B8  = "B8.tif"
NDVI  = "NDVI.tif"
NDWI  = "NDWI.tif"
s1VH  = "VH_Speckle_Filtered.tif"
s1VV  = "VV_Speckle_Filtered.tif"

inraster        = stack(B2, B3, B4, B8, NDVI, NDWI, s1VH, s1VV)
names(inraster) = c("B2", "B3", "B4", "B8", "NDVI", "NDWI", "s1VH", "s1VV")


# Setting-up the working directory
setwd('E:/ITC_Twente/Quarter_05_September_2023/00_Advanced_Image_Analysis/XX_Group_Project/04 Model Operation/Samples_Group')

trainingData  =  shapefile("train_set.shp")
trainingData$Class = as.factor(trainingData$Class)
```

```
summary(trainingData )


barplot(prop.table(table(trainingData$Class)),
       col = rainbow(7),
       ylim = c(0, 0.7),
       main = "Class Distribution")


length(trainingData)
TestingData = shapefile("test_set.shp")
length(TestingData)
TestingData$ClassID


#================================================================================
# Extract raster values for the training samples
#================================================================================
training_data  = extract(inraster, trainingData)
training_data
train <- data.matrix(training_data)
# training_response = as.numeric (as.factor(trainingData$Class))-1
training_response = as.numeric(as.factor(trainingData$ClassID))- 1
training_response
as.numeric(as.factor(trainingData$ClassID))- 1




#================================================================================
# Training the xgboost model
#================================================================================


xgb_model <- xgboost(data = train,
                     label = training_response,
                     booster = "gbtree",
                     eta = 0.5,
                     gamma = 0,
                     max_depth = 50,
          min_child_weight = 1,

          nround=500,

          objective = "multi:softmax",

          num_class = length(unique(training_response)),

)


summary(xgb_model)

mat <- xgb.importance (feature_names = colnames(train),model = xgb_model)

xgb.plot.importance (importance_matrix = mat[1:23])

mat


#============================================================================================

# Classify the entire image:define raster data to use for classification

#===========================================================================================


#change this to your output directory if different

setwd('E:/ITC_Twente/Quarter_05_September_2023/00_Advanced_Image_Analysis/XX_Group_Project/04 Model Operation/Outputs')




#============================================================================================

# Classify the entire image

#===========================================================================================


result   <- predict(xgb_model, inraster [1:(nrow(inraster )*ncol(inraster ))])

res      <- raster(inraster)

res      <- setValues(res,result+1)

res


#============================================================================================

# Assess the classification accuracy

#===========================================================================================

TestingData$ClassID


Testing=extract(res, TestingData) # extracts the value of the classified raster at the validation point locations

Testing
```

```
confusionMatrix(as.factor(Testing), as.factor(TestingData$ClassID) )


confusionMatrix(as.factor(Testing), as.factor(TestingData$ClassID) )$byClass[, 1]


confusionMatrix(as.factor(Testing), as.factor(TestingData$ClassID) )$byClass[]


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


Class_Results = writeRaster(res, 'S2_Based_classification results.tif', overwrite=TRUE)




#=====================================================================================
## Tuning with Trail and Error
#=====================================================================================


xgb_model <- xgboost(data = train,
                  label = training_response,
                  booster = "gbtree",
                  eta = 0.5,
                  gamma = 0,
                  max_depth = 10,
                  min_child_weight = 1,
                  nround=500,
                  objective = "multi:softmax",
                  num_class = length(unique(training_response)),
)



summary(xgb_model)
mat <- xgb.importance (feature_names = colnames(train),model = xgb_model)
xgb.plot.importance (importance_matrix = mat[1:23])
```

```
mat


#====================================================================================================

# Classify the entire image

#===================================================================================================


result   <- predict(xgb_model, inraster [1:(nrow(inraster )*ncol(inraster ))]])

res      <- raster(inraster)

res      <- setValues(res,result+1)

res


#==============================================================================================

# Assess the classification accuracy

#==============================================================================================

TestingData$ClassID


Testing=extract(res, TestingData)

Testing

confusionMatrix(as.factor(Testing), as.factor(TestingData$ClassID) )

confusionMatrix(as.factor(Testing), as.factor(TestingData$ClassID) )$byClass[, 1]

confusionMatrix(as.factor(Testing), as.factor(TestingData$ClassID) )$byClass[]
```