In [ ]:

```
# packages
import os
from osgeo import gdal, ogr, gdal_array # I/O image data
import joblib
import numpy as np # math and array handling
import matplotlib.pyplot as plt # plot figures
from matplotlib.colors import ListedColormap # to import certain defined color palettes for plotting your results
from sklearn.ensemble import RandomForestClassifier # classifier
import pandas as pd # handling large data as table sheets
from sklearn.metrics import classification_report, accuracy_score,confusion_matrix  # calculating measures for accuracy assessment
from sklearn.neighbors import KNeighborsClassifier
from skimage import exposure  # for adjustment of rasterstack (histogram equalization, etc)

# Tell GDAL to throw Python exceptions, and register all drivers
gdal.UseExceptions()
gdal.AllRegister()
```

In [ ]:

```
# path where my data is located
folder_src = r"C:\Users\rwolff\Documents\Lac Bam SSD\Test\Neuer Ordner\TDX_3"
folder_src_shape = r"C:\Users\rwolff\Documents\Lac Bam SSD\Shapefiles classification  angepasste Klassen"

# path where I want to save my results
folder_results = r"E:\CSV_27 Aufnahmen\TSX"
```

In [ ]:

```
# how many cores should be used?
n_cores = -1
# -1 -> all available cores
```

In [ ]:

```
#import os

directory = folder_src
directory_shapes = folder_src_shape
iterator = 0

for filename_tif, filename_shape in zip(os.listdir(directory),[f for f in os.listdir(directory_shapes) if f.endswith('.shp')]):
  file = os.path.join(directory, filename_tif)
  if os.path.isfile(file):
    print(file)
    s2_stack=file
    print(s2_stack)
    filename = filename_tif
    # load image data
    #In this script we are Using gdal.open() instead of rio.open()
    img_ds = gdal.Open(s2_stack, gdal.GA_ReadOnly)

    img = np.zeros((img_ds.RasterYSize, img_ds.RasterXSize, img_ds.RasterCount),
               gdal_array.GDALTypeCodeToNumericTypeCode(img_ds.GetRasterBand(1).DataType))
    for b in range (img.shape[2]):
        img[:, :, b] = img_ds.GetRasterBand(b + 1).ReadAsArray()

    print("Raster format is:", gdal_array.GDALTypeCodeToNumericTypeCode(img_ds.GetRasterBand(1).DataType))

# store the variables above in a more meaningful way. You will use these variables later.
    row = img_ds.RasterYSize
    col = img_ds.RasterXSize
    band_number = img_ds.RasterCount

    print("Raster number of rows: {}".format(row))
    print("Raster number of columns: {}".format(col))
    print("Raster number of bands: {}".format(band_number))

# Take our full image and reshape into long 2d array (nrow * ncol, nband) for classification
    #new_shape = (img.shape[0] * img.shape[1], img.shape[2])
    #img_as_array = img[:, :, : int(img.shape[2])].reshape(new_shape)

    #print('Reshaped from {o} to {n}'.format(o=img.shape, n=img_as_array.shape))

    training = folder_src_shape + "\\" + filename_shape
# what is the numerical attribute of your classes in the shapefile?
    attribute = 'id'
# load training data and show all shapefile attributes
    print(training)
    shape_dataset = ogr.Open(training)
    shape_layer = shape_dataset.GetLayer()

# extract the names of all attributes (fieldnames) in the shape file
```

```
    attributes = [] # empty list where the attributes will be saved
    ldefn = shape_layer.GetLayerDefn() # encapsulates the attribute schema of the features of the layer
    for n in range(ldefn.GetFieldCount()):
        fdefn = ldefn.GetFieldDefn(n)
        attributes.append(fdefn.name)

# print the attributes
    print('Available attributes in the shapefile are: {}'.format(attributes))
# copy the structure of your Sentinel2 image to pass this information to the new rasterized polygons
    mem_drv = gdal.GetDriverByName('MEM')
    mem_raster = mem_drv.Create('',img_ds.RasterXSize,img_ds.RasterYSize,1,gdal.GDT_Byte)
    mem_raster.SetProjection(img_ds.GetProjection())
    mem_raster.SetGeoTransform(img_ds.GetGeoTransform())
    mem_band = mem_raster.GetRasterBand(1)
    mem_band.Fill(0)
    mem_band.SetNoDataValue(0)



    att_ = 'ATTRIBUTE='+attribute

# rasterize your polygons
    err = gdal.RasterizeLayer(mem_raster, [1], shape_layer, None, None, [1], [att_,"ALL_TOUCHED=TRUE"])
    assert err == gdal.CE_None

    roi = mem_raster.ReadAsArray()
# Number of training pixels:
    n_samples = (roi > 0).sum()
    print('{n} training samples'.format(n=n_samples))

# What are our classification labels?
    labels = np.unique(roi[roi > 0])
    print('training data include {n} classes: {classes}'.format(n=labels.size, classes=labels))

    # Subset the image dataset with the training image = X
    # Mask the classes on the training dataset = y
    # These will have n_samples rows


    X = img[roi > 0, :]
    y = roi[roi > 0]

    if iterator == 0:
        X_concat = X.copy()
        y_concat = y.copy()
    else:
        X_concat1 = X_concat.copy()
        y_concat1 = y_concat.copy()
        X_concat = np.concatenate((X_concat1, X), axis=0)
        y_concat = np.concatenate((y_concat1, y), axis=0)
    iterator+=1
```

In [ ]:
```
klassen,Anzahl=np.unique(y_concat , return_index=False, return_inverse=Fals
e, return_counts=True, axis=None)
print(Anzahl)
print(klassen)
```

In [ ]:
```
## SAFE X_concat to CSV

array_X = X_concat
df = pd.DataFrame(array_X)
display(df) #Alternative zu print()

#1. Schritt: CSV-File wird geschrieben
output_name = 'X_concat TDX_3test.csv'
output_folder = r"E:\CSV Data\TSX"
output_data = os.path.join(output_folder, output_name)
if os.path.isfile(output_data):   #checks, if file already exists
    print ("\nFile already exists in {}! Image was not saved!".format(outpu
t_data))
else:
    df.to_csv(output_data)
    print ("\nCsv-file was successfully saved in {}!".format(output_data))

## SAFE y_concat to CSV

array_y = y_concat
df = pd.DataFrame(array_y)
display(df) #Alternative zu print()

#1. Schritt: CSV-File wird geschrieben
output_name = 'y_concat TDX_3test.csv'
output_folder = r"E:\CSV Data\TSX"
output_data = os.path.join(output_folder, output_name)
if os.path.isfile(output_data):   #checks, if file already exists
    print ("\nFile already exists in {}! Image was not saved!".format(outpu
t_data))
else:
    df.to_csv(output_data)
    print ("\nCsv-file was successfully saved in {}!".format(output_data))
```

In [ ]:
```
klassen,Anzahl=np.unique(array_y, return_index=False, return_inverse=False,
return_counts=True, axis=None)
print(Anzahl)
print(klassen)
```

In [ ]: