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 co
lor 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 eq
ualization, etc)

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

In [ ]:
```
# CSV-File einlesen X_concat
input_name = 'X_concat TDX_3.csv'
input_folder = r"E:\CSV Data\TSX"
input_data = os.path.join(input_folder, input_name)

df = pd.read_csv(input_data, index_col = 0)
display(df)
array_X= df.to_numpy()
display(array_X, type(array_X))

# CSV-File einlesen y_concat
input_name = 'y_concat TDX_3.csv'
input_folder = r"E:\CSV Data\TSX"
input_data = os.path.join(input_folder, input_name)

df = pd.read_csv(input_data, index_col = 0)
display(df)
array_y= df.to_numpy()

display(array_y, type(array_y))
```

In [ ]:
```
X_concat = array_X
y_concat = array_y
```

In [ ]:
```
print(X_concat)
```

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

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

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_concat, y_concat, tes
t_size=0.3, random_state=42)
```

In [ ]:
```
# Oversampling
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X_train, y_train)

print('Our X matrix is sized: {sz}'.format(sz=X_resampled.shape))
print('Our y array is sized: {sz}'.format(sz=y_resampled.shape))
```

In [ ]:
```
X_train = X_resampled
y_train = y_resampled
print('Our X matrix is sized: {sz}'.format(sz=X_train.shape))
print('Our y array is sized: {sz}'.format(sz=y_train.shape))
```

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

In [ ]:
```
## RANDOM FOREST TRAIN / TEST
est = 500
rf = RandomForestClassifier(n_estimators=est, verbose=1, n_jobs=n_cores, ra
ndom_state=42, max_features ="sqrt")
X_train = np.nan_to_num(X_train)
rf.fit(X_train, y_train)
```

In [ ]:
```
#Prediction class for the testing set
X_test = np.nan_to_num(X_test)
y_pred = rf.predict(X_test)
print('Shape prediction {}'.format(y_pred.shape))
```

In [ ]:
```
# Confusion Matrix
# View confusion matrix for test data and predictions
confusion_matrix(y_test, y_pred)
```

In [ ]:
```
# Check the model performance
print(classification_report(y_test, y_pred))
```

In [ ]:
```
# path where my data is located
folder_NewData_src = r"C:\Users\rwolff\Documents\Lac Bam SSD\Aufnahmen Clip
ped\Neue Aufnahmen\Neue Aufnahmen TDX_3 Clipped"
folder_src_shape = r"C:\Users\rwolff\Documents\Lac Bam SSD\Vektordaten_Merg
ed all"

# path where I want to save my results
folder_results = r"E:\results\500"
```

In [ ]:
```
#import os

directory = folder_NewData_src
directory_shapes = folder_src_shape
iterator = 0

for filename_tif, filename_shape in zip(os.listdir(directory),[f for f in o
s.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.RasterCo
unt),
                   gdal_array.GDALTypeCodeToNumericTypeCode(img_ds.GetRasterBan
d(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 va
riables 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) f
or 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.sha
pe))

    img_as_array = np.nan_to_num(img_as_array)

    training = folder_src_shape + "\\" + filename_shape

# Now predict for each pixel
    #img_as_array = np.nan_to_num(img_as_array)
    class_prediction = rf.predict(img_as_array)
    print('Shape prediction {}'.format(class_prediction.shape))
```

```
    class_prediction = class_prediction.reshape(img[:, :, 0].shape)
    print('Reshaped back to {}'.format(class_prediction.shape))
# assing colors to each class. The order of the colors depends on the order
of the landclasses
    custom_cmap = ListedColormap(["orange","blue","green","purple"])
# for example, here water = "lightseagreen"

# plot your classification
    fig,ax = plt.subplots(figsize=(20,20))
    ax.set_title('Random Forest Classification\n n-trees = 200 TDX 1', font
size = 35) # use '\n' to start a new line

# indicate which file will be plotted, add colors
    plot_rf =ax.imshow(class_prediction, cmap=custom_cmap)
# set parameters for colorbar
    cbar_rf = plt.colorbar(plot_rf,shrink=0.6)
    cbar_rf.set_ticks([1,2,3,4])
    cbar_rf.set_ticklabels(["fields", "open water", "flooded vegetation", "
wetlands"])
    cbar_rf.ax.tick_params(labelsize=25) # adapt font size of ticks
# show your plot
#plt.show()

 # TEST BLOCK ZUM SPEICHERN DER DATEN !!!!
    plotname = "\\" + filename[8:-4]
    print(folder_results)
    print(plotname)
# export your plot as an PNG image
#fig.savefig(os.path.join(folder_results, "plotname1.png"), bbox_inches='ti
ght')
#fig.savefig(os.path.join(folder_results, "FIG%d%d.png"), bbox_inches='tigh
t')
    fig.savefig(os.path.join(folder_results, str(filename)+".png"), bbox_in
ches='tight')

# define where the image will be saved
#classification_image=os.path.join(folder_results,'plotname1.tif')
    classification_image=os.path.join(folder_results, str(filename)+'.tif')
    cols = img.shape[1]
    rows = img.shape[0]

# define structure of your output file
    driver = gdal.GetDriverByName("gtiff")
    outdata = driver.Create(classification_image, cols, rows, 1, gdal.GDT_B
yte)
    outdata.SetGeoTransform(img_ds.GetGeoTransform()) ##sets same geotransf
orm as input
    outdata.SetProjection(img_ds.GetProjection()) ##sets same projection as
input

# specify which image you want to save
    outdata.GetRasterBand(1).WriteArray(class_prediction)

#saves to disk!!
    outdata.FlushCache()
```

```
    print('Image saved to: {}'.format(classification_image))
```

In [ ]:
```
# RANDOM FOREST SPEICHERN!  mit JOBLIB TEST!!!!!!
#import joblib
#from sklearn.ensemble import RandomForestClassifier
#joblib.dump(rf, "RandomForest_TDX_2_over.joblib")
```

In [ ]:
```

```