##############GEE

Map.centerObject(roi,5);

// // Map.addLayer(roi, {color: "gray"}, "roi");

var elevation = ee.Image("USGS/SRTMGL1\_003").select('elevation').clip(roi);

// slope

var slope = ee.Terrain.slope(elevation);

//aspect

var aspect = ee.Terrain.aspect(elevation);

var cosImg = aspect.divide(180).multiply(Math.PI).cos()

// Call the repository already configured color palette

var palettes = require('users/gena/packages:palettes');

var palette = palettes.colorbrewer.Greys[9].reverse();

// Map.addLayer(elevation, {min:0, max:2500,palette: palette}, "elevation", false);

// Map.addLayer(slope, {min:0, max:60,palette: palette}, "slope", false);

// Map.addLayer(cosImg, {min:-1, max:1,palette: palette}, "cosImg", false);

// Map.addLayer(table, {color: "red"}, "grass", false);

// S2 image decloud and compositing

function maskS2clouds(image) {

var qa = image.select('QA60');

// Bits 10 and 11 are clouds and cirrus, respectively.

var cloudBitMask = 1 << 10;

var cirrusBitMask = 1 << 11;

// Both flags should be set to zero, indicating clear conditions.

var mask = qa.bitwiseAnd(cloudBitMask)

.eq(0)

.and(qa.bitwiseAnd(cirrusBitMask)

.eq(0));

return image.updateMask(mask)

.divide(10000);

}

// calculated features

// VI

function VI(img) {

var ndvi = img.expression(

"(NIR-R)/(NIR+R)", {

"R": img.select("B4"),

"NIR": img.select("B8"),

}).rename("NDVI")

var ndwi = img.expression(

"(G-MIR)/(G+MIR)", {

"G": img.select("B3"),

"MIR": img.select("B8"),

}).rename('NDWI');

var ndbi = img.expression(

"(SWIR-NIR)/(SWIR-NIR)", {

"NIR": img.select("B8"),

"SWIR": img.select("B12"),

}).rename("NDBI");

var savi = img.expression('(NIR - RED) \* (1 + 0.5)/(NIR + RED + 0.5)', {

'NIR': img.select('B8'),

'RED': img.select('B4')

}).float().rename('SAVI');

var ibiA = img.expression('2 \* SWIR1 / (SWIR1 + NIR)', {

'SWIR1': img.select('B6'),

'NIR': img.select('B5')

}).rename(['IBI\_A']);

var ibiB = img.expression('(NIR / (NIR + RED)) + (GREEN / (GREEN + SWIR1))', {

'NIR': img.select('B8'),

'RED': img.select('B4'),

'GREEN': img.select('B3'),

'SWIR1': img.select('B11')

}).rename(['IBI\_B']);

var ibiAB = ibiA.addBands(ibiB);

var ibi = ibiAB.normalizedDifference(['IBI\_A', 'IBI\_B']).rename('IBI');

var rvi = img.expression('NIR/Red', {

'NIR': img.select('B8'),

'Red': img.select('B4')

}).rename('RVI');

var dvi = img.expression('NIR - Red', {

'NIR': img.select('B8'),

'Red': img.select('B4')

}).float().rename('DVI');

return img.addBands(ee.Image.cat([ndvi,ndwi,ndbi,savi,ibi,rvi,dvi]))

}

// image selection

var collection = ee.ImageCollection("COPERNICUS/S2\_SR")

.filterDate('2022-01-01', '2022-12-31')

.filterBounds(roi)

.filter(ee.Filter.lt('CLOUDY\_PIXEL\_PERCENTAGE',20))

.map(maskS2clouds)

.map(VI);

var image1 = collection.median()

.clip(roi);

// print(collection)

// visualization

var visualization = {

min: 0.0,

max: 0.3,

bands: ['B4', 'B3', 'B2'],

};

print(image)

var image = ee.ImageCollection("COPERNICUS/S2\_SR")

.filterDate('2022-01-01', '2022-12-31')

.filterBounds(roi)

.filter(ee.Filter.lt('CLOUDY\_PIXEL\_PERCENTAGE',20))

.map(maskS2clouds)

.median()

.clip(roi)

.select(['B1','B2','B3','B4','B5','B6','B7','B8','B9','B11','B12','B8A']);

//KT

var coefficients = ee.Array([

[0.0356, 0.0822, 0.1360, 0.2611, 0.2964, 0.3338, 0.3877, 0.3895, 0.0949, 0.3882, 0.1366, 0.4750],

[-0.0635, -0.1128, -0.1680, -0.3480, -0.3303, 0.0852, 0.3302, 0.3165, 0.0467, -0.4578, -0.4064, 0.3625],

[0.0649, 0.1363, 0.2802, 0.3072, 0.5288, 0.1379, -0.0001, -0.0807, -0.0302,-0.4064, -0.5602,-0.1389]

]);

var arrayImage1D = image.toArray();

//print("arrayImage1D",arrayImage1D)

Map.addLayer(arrayImage1D,{},"arrayImage1D")

var arrayImage2D = arrayImage1D.toArray(1);

//print("arrayImage2D",arrayImage2D)

Map.addLayer(arrayImage2D,{},"arrayImage2D")

// Multiplicative transformation

var componentsImage = ee.Image(coefficients)

.matrixMultiply(arrayImage2D)

.arrayProject([0])

.arrayFlatten([[

'brightness', 'greenness', 'wetness'

]]);

var greenness = componentsImage.select('greenness')

var wetness = componentsImage.select('wetness')

var composite = image1.addBands(elevation)

.addBands(slope)

.addBands(cosImg)

.addBands(wetness)

.addBands(greenness)

var visParam = {

min: -0.1,

max: 0.9,

palette: 'FFFFFF, CE7E45, DF923D, F1B555, FCD163, 99B718, 74A901, 66A000, 529400,' +

'3E8601, 207401, 056201, 004C00, 023B01, 012E01, 011D01, 011301'

};

Map.addLayer( composite.select("NDVI"), visParam,'NDVI',false);

print('composite',composite)

Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3},'original',false);

//output composite

Export.image.toAsset({

image: composite,

description: 'composite12',

scale: 10,

region: roi,

crs:'EPSG:4326',

maxPixels:1e13

});

// // Export.image.toDrive({

// // image: image.select('SR\_B.\*'),

// // description: 'class\_img',

// // folder: 'class\_img',

// // fileNamePrefix: '2020class',

// // region:roi,

// // scale: 10,

// // maxPixels: 1e13

// });

//The pixels contained in the category

// var s6 = ee.List([forest,grassland,field,dry,village,water,pond,town]);

// var fvc = s6.map(function(img){

// var ndvi = composite.sampleRegions({

// collection:img,

// scale:10,

// geometries:true,

// });

// return ndvi

// });

// print(fvc)

// print('forest',ee.FeatureCollection(fvc.get(0)).size())

// The 5 category elements are synthesized into one element set

var classNames = table;

print(classNames)

// Select the band required for classification

var bands = [ 'B2', 'B3', 'B4', 'B5', 'B6','B7','B8','B8A','B11','B9','B12'

,'NDBI','NDVI','elevation',"slope",'IBI','SAVI','NDWI','RVI']

var training = composite.select(bands)

.sampleRegions({

collection: classNames,

properties: ['CLCL'],

scale: 30

});

// print(train.first())

// Setting variable data

var withRandom = training.randomColumn('random');

var split = 0.7;

var trainingP = withRandom.filter(ee.Filter.lt('random', split)); // 70% training sample

var testingP = withRandom.filter(ee.Filter.gte('random', split)); // 30% verification sample

// Set the classifier parameters

var classifier = ee.Classifier.smileRandomForest(60).train({

features: trainingP,

classProperty: "CLCL",

inputProperties: bands

});

print('explain',classifier.explain())

var class\_img = composite.select(bands).classify(classifier);

var palette = [

'#FF0000', // field

'#00FF66', // dry

'#FF00FF', // town

'#FFA07A', // village

'#F0F8FF', // forest

'#FF4500',// grassland

'#00FFFF',//water

'#0000FF'// pond

];

var palette1 = palettes.colorbrewer.Accent[8];

Map.addLayer(class\_img, {min: 1, max:9, palette: palette1}, 'classified');

Export.image.toDrive({

image: class\_img,

description: 'yuanmouforests',

folder: 'yuanmouforests',

fileNamePrefix: 'yuanmouforests',

region:roi,

scale:10,

maxPixels: 1e13

});

// Training accuracy

var trainAccuracy = classifier.confusionMatrix().accuracy();

print('trainAccuracy', trainAccuracy);

// prediction accuracy

var testAccuracy = testingP

.classify(classifier)

.errorMatrix('CLCL', 'classification')

.accuracy();

print('testAccuracy', testAccuracy);

// parameters tune

var numTrees = ee.List.sequence(5, 100, 5);

var accuracies = numTrees.map(function(t) {

var classifier = ee.Classifier.smileRandomForest(t)

.train({

features: trainingP,

classProperty: 'CLCL',

inputProperties: bands

});

return testingP

.classify(classifier)

.errorMatrix('CLCL', 'classification')

.accuracy();

});

print(ui.Chart.array.values({

array: ee.Array(accuracies),

axis: 0,

xLabels: numTrees

}));

// precision evaluation

var test = testingP.classify(classifier);

// confusion matrix

var confusionMatrix = test.errorMatrix('CLCL', 'classification')

// print('confusionMatrix',confusionMatrix);// The confusion matrix is displayed on the panel

print('overall accuracy', confusionMatrix.accuracy());// The overall accuracy is displayed on the panel

print('kappa accuracy', confusionMatrix.kappa());// The overall accuracy is displayed on the panel