// Set map center to the AOI to make sure we have the correct study area

Map.centerObject(AOI, 9);

// Import Sentinel-1 collection

var sentinel1 = ee.ImageCollection('COPERNICUS/S1\_GRD')

.filterBounds(AOI)

.filterDate('2023-08-01', '2023-09-30')

.filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))

.filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VH'))

.filter(ee.Filter.eq('instrumentMode', 'IW'));

var desc = sentinel1.filter(ee.Filter.eq('orbitProperties\_pass', 'DESCENDING'));

var asc = sentinel1.filter(ee.Filter.eq('orbitProperties\_pass', 'ASCENDING'));

// Inspect number of tiles returned after the search; we will use the one with more tiles

print("descending tiles ", desc.size());

print("ascending tiles ", asc.size());

// Also Inspect one file

print(asc.first());

print(desc.first());

// Create a composite from means at different polarizations and look angles.

var composite = ee.Image.cat([

asc.select('VH').mean(),

asc.select('VV').mean(),

desc.select('VH').mean()

]).focal\_median();

var composite = composite.clip(AOI);

// Display as a composite of polarization and backscattering characteristics.

Map.addLayer(composite, {min: [-25, -20, -25], max: [0, 10, 0]}, 'composite');

// Define the land cover classes

var water = /\* color: #0000ff \*/ ee.FeatureCollection(

[ee.Feature(

ee.Geometry.Point([70.9104, 32.6105]), // Add sample points for water

{"landcover": 1}),

ee.Feature(

ee.Geometry.Point([70.9124, 32.6125]),

{"landcover": 1})]

);

var forest = /\* color: #00ff00 \*/ ee.FeatureCollection(

[ee.Feature(

ee.Geometry.Point([70.9304, 32.6205]), // Add sample points for forest

{"landcover": 2}),

ee.Feature(

ee.Geometry.Point([70.9324, 32.6225]),

{"landcover": 2})]

);

var barren = /\* color: #ffb6b6 \*/ ee.FeatureCollection(

[ee.Feature(

ee.Geometry.Point([70.9504, 32.6305]), // Add sample points for barren land

{"landcover": 3}),

ee.Feature(

ee.Geometry.Point([70.9524, 32.6325]),

{"landcover": 3})]

);

var urban = /\* color: #ff0000 \*/ ee.FeatureCollection(

[ee.Feature(

ee.Geometry.Point([70.9704, 32.6405]), // Add sample points for urban areas

{"landcover": 4}),

ee.Feature(

ee.Geometry.Point([70.9724, 32.6425]),

{"landcover": 4})]

);

// Merge points together

var newfc = water.merge(forest).merge(barren).merge(urban);

print(newfc, 'newfc');

var Bands\_selection = ['VV', 'VH'];

// Overlay

var training = composite.sampleRegions({

collection: newfc,

properties: ['landcover'],

scale: 10

});

// Split: Training (75%) & Testing samples (25%).

var Total\_samples = training.randomColumn('random');

var training\_samples = Total\_samples.filter(ee.Filter.lessThan('random', 0.75));

print(training\_samples, "Training Samples");

var validation\_samples = Total\_samples.filter(ee.Filter.greaterThanOrEquals('random', 0.75));

print(validation\_samples, "Validation\_Samples");

// Random Forest Classifier

var classifier = ee.Classifier.smileRandomForest(10).train({

features: training,

classProperty: 'landcover',

inputProperties: Bands\_selection

});

var classified = composite.classify(classifier);

// Define a palette for the Land Use classification.

var palette = [

'Blue', // water (1)

'green', // forest (2)

'fec89a', // barren (3)

'Red' // urban (4)

];

var classified = classified.clip(AOI);

Map.addLayer(classified, {min: 1, max: 4, palette: palette}, "classification");

// Print the confusion matrix and overall accuracy

var confusionMatrix = classifier.confusionMatrix();

print(confusionMatrix, 'Error matrix: ');

print(confusionMatrix.accuracy(), 'Training Overall Accuracy: ');

// Export the classified image to Google Drive

Export.image.toDrive({

image: classified,

description: 'Classified\_Image',

region: AOI,

scale: 10,

crs: 'EPSG:4326',

maxPixels: 1e13

});