var ROI = ee.FeatureCollection("projects/ee-osamaamjad256/assets/nowshera");

//var ROI = ee.FeatureCollection('projects/ee-kimutailawrence19/assets/ROI');

// Set map center to the aoi for making sure we have the correct study area

Map.centerObject(ROI, 9)

// Define period of analysis

var start = '2022-01-01';

var end = '2022-12-31';

var season = ee.Filter.date(start,end);

print(season);

// Import Sentinel-1 collection

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

// Filter Sentinel-1 collection for study area, date ranges and polarization components

var sCollection = sentinel1

//filter by aoi and time

.filterBounds(ROI)

.filter(season)

// Filter to get images with VV and VH dual polarization

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

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

// Filter to get images collected in interferometric wide swath mode.

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

// Also filter based on the orbit: descending or ascending mode

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

var asc = sCollection.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());

// 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();

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

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

// Merge points together

var newfc = builtup.merge(vegetation).merge(barren);

print(newfc, 'newfc');

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

//overlay

var training = composite.sampleRegions({

collection:newfc,

properties:['landcover'],

scale:30

});

///SPLITS: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 CLASSIFER-------------------/

// var classifier = ee.Classifier.smileRandomForest(numberOfTrees, variablesPerSplit, minLeafPopulation, bagFraction, maxNodes, seed)

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 = [

'grey', // urban (0) // grey

'blue', // water (1) // blue

'green' // forest (2) // green

];

Map.addLayer(classified.clip(ROI),{min: 0, max: 9,palette: palette},"classification");

Map.centerObject(ROI,10);

var confusionMatrix =classifier.confusionMatrix();

print(confusionMatrix,'Error matrix: ');

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