// 1. Load boundary

var **aoi** = ee.FeatureCollection (boundary path);

// 2. Load satellite image- landsat in this example

var **landsatCollection** = ee.ImageCollection('LANDSAT/LC08/C02/T1\_L2')

.filterBounds(**aoi**)

.filterDate(‘start date in YYYY-MM-DD’, ‘end date in YYYY-MM-DD’)

.filter(ee.Filter.lt('CLOUD\_COVER', value in percentage without percent sign));

// 3. Print image count

print('Number of images for 2023:', **landsatCollection**.size());

// 4. Print dates of images

**landsatCollection**.aggregate\_array('system:time\_start').evaluate(function(dates) {

print('Image dates:', dates.map(function(d) {

return ee.Date(d).format('YYYY-MM-dd').getInfo();

}));

});

// 5. Create median composite and clip

var **compositeImage** = **landsatCollection**.median().clip(**aoi**);

// 6. Select bands

var **bands** = ['SR\_B2', 'SR\_B3', 'SR\_B4', 'SR\_B5', 'SR\_B6', 'SR\_B7'];

var **inputImage** = **compositeImage**.select(**bands**);

// 7. Visualize true color

Map.addLayer(**compositeImage**, {bands: ['SR\_B4', 'SR\_B3', 'SR\_B2'], min: 7000, max: 12000}, 'Raw Composite Image (2023)');

// 8. Compute NDBI

var **ndbi** = **compositeImage**.normalizedDifference(['SR\_B6', 'SR\_B5']).rename('NDBI');

// 9. Mask for built-up areas (suggested threshold: 0.1)

var **builtUpAreas** = **ndbi**.updateMask(ndbi.gt(0.1));

Map.addLayer(**builtUpAreas**, {min: 0.1, max: 1, palette: ['red']}, 'Built-up Areas (NDBI > 0.1)');

// 10. Merge training samples

var **training** = BuiltUps.merge(Agricultures)

.merge(Forests)

.merge(Barelands)

.merge(Bushlands)

.merge(Water)

.merge(Wetlands);

// 11. Sample training data

var **trainImage** = **inputImage**.sampleRegions({

collection: training,

properties: ['Class'],

scale: 30

});

// 12. Split training and testing sets

var **trainingData** = **trainImage**.randomColumn();

var **trainSet** = **trainingData**.filter(ee.Filter.lt('random', 0.8));

var **testSet** = **trainingData**.filter(ee.Filter.gte('random', 0.8));

// 13. Train classifier (Random Forest)

var **classifier** = ee.Classifier.smileRandomForest({

numberOfTrees: 100,

seed: 42

}).train({

features: **trainSet**,

classProperty: 'Class',

inputProperties: **bands**

});

// 14. Classify image

var **classifiedImage** = **inputImage**.classify(classifier);

var **smoothedImage** = **classifiedImage**.focalMode(5, 'square', 'pixels');

// 15. Visualization parameters

var palette = [

'#FFA500', // BuiltUps

'#ADFF2F', // Agriculture

'#1E90FF', // Water

'#FFFF00', // Bushlands

'#000000', // Barelands

'#0000FF', // Wetlands

'#054907’ // Forests

];

Map.addLayer(**classifiedImage**, {min: 0, max: 6, palette: palette}, 'Classified LULC');

Map.addLayer(**smoothedImage**, {min: 0, max: 6, palette: palette}, 'Smoothed LULC');

// 16. Export results

Export.image.toDrive({

image: **classifiedImage**,

description: 'RF\_Classified\_image',

folder: 'EarthEngineExports',

region: **aoi**.geometry(),

scale: 30,

maxPixels: 1e13,

fileFormat: 'GeoTIFF'

});

Export.image.toDrive({

image: **smoothedImage**,

description: 'RF\_Smoothed\_image',

folder: 'EarthEngineExports',

region: **aoi**.geometry(),

scale: 30,

maxPixels: 1e13,

fileFormat: 'GeoTIFF'

});

Export.image.toDrive({

image: **builtUpAreas**,

description: 'NDBI\_Mask’,

folder: 'EarthEngineExports',

region: **aoi**.geometry(),

scale: 30,

maxPixels: 1e13,

fileFormat: 'GeoTIFF'

});

// 17. Accuracy assessment

var **testClassified** = **testSet**.classify(classifier);

var **confusionMatrix** = **testClassified**.errorMatrix('Class', 'classification');

print('Confusion Matrix :', **confusionMatrix**);

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

// 18. Center map

Map.centerObject(**aoi**);