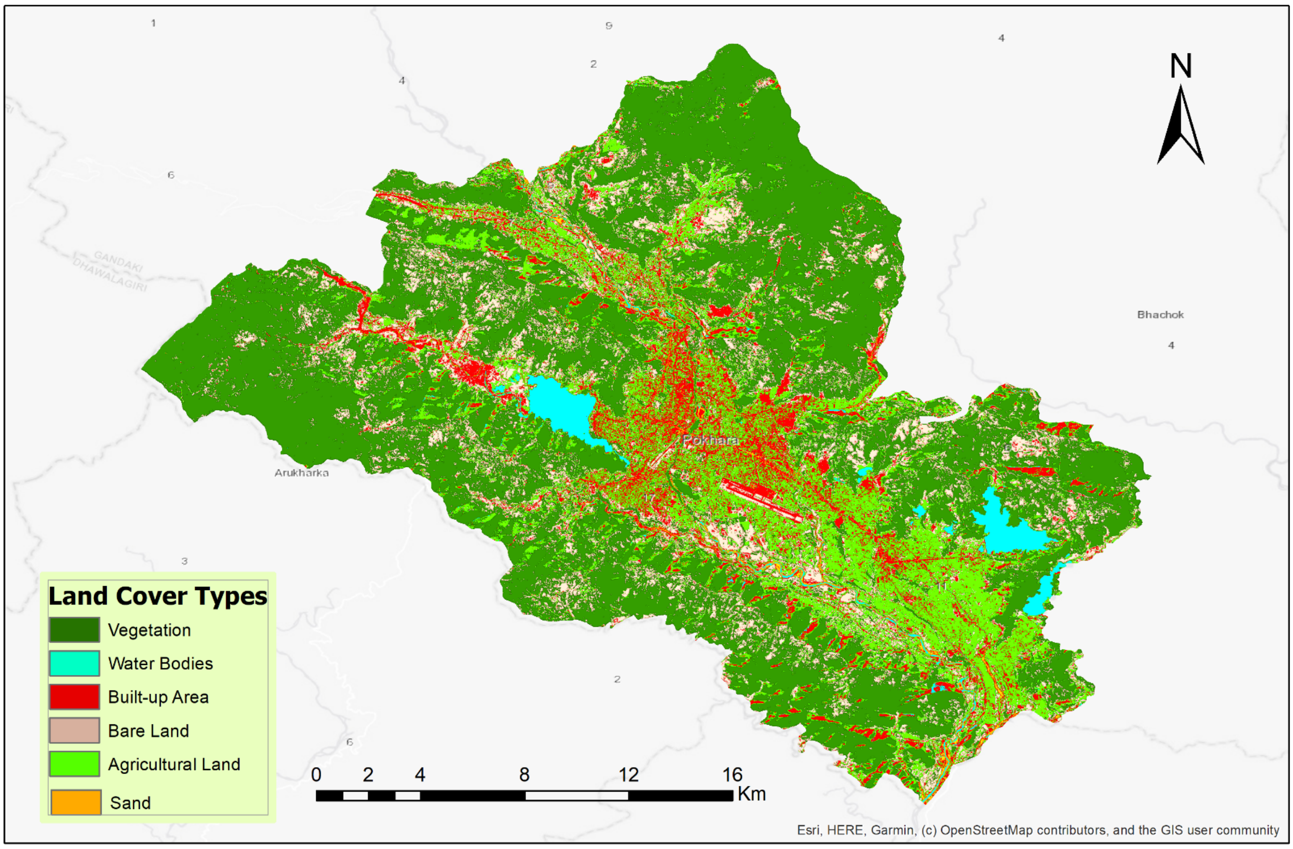
Prepared LULC Map:

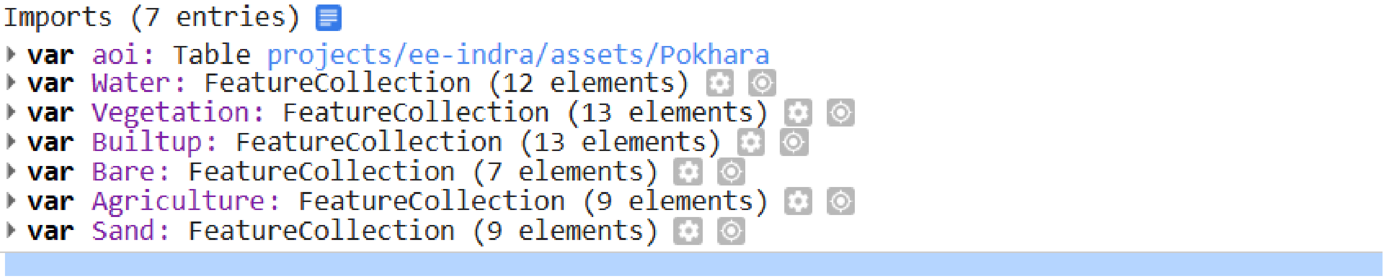
Prepared LULC Map:

Land Use Land Cover Map of Pokhara-2024



Esri, HERE, Garmin, (c) OpenStreetMap contributors, and the GIS user community

Code :



```
Imports (7 entries)
▸ var aoi: Table projects/ee-indra/assets/Pokhara
▸ var Water: FeatureCollection (12 elements)
▸ var Vegetation: FeatureCollection (13 elements)
▸ var Builtup: FeatureCollection (13 elements)
▸ var Bare: FeatureCollection (7 elements)
▸ var Agriculture: FeatureCollection (9 elements)
▸ var Sand: FeatureCollection (9 elements)
```

```
//Filtering imagery
var image = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
.filterDate("2024-10-01", "2024-12-30")
// Pre-filter to get less cloudy granules.
.filter(ee.Filter.lt("CLOUDY_PIXEL_PERCENTAGE", 20))
.filterBounds(aoi)
.map(maskS2clouds)
.map(function (img) {
return img.clip(aoi);
}) .median();
```

```
image = selectBands(image);
print(image);

// visualize imagery
Map.addLayer(image,{}, "dataset_rgbviz", true);
Map.centerObject(aoi,10)

//========================================================
// Machine Learning Model
//========================================================
//merge samples
var sample = Water
.merge(Agriculture)
.merge(Vegetation)
.merge(Builtup)
.merge(Sand)
.merge(Bare)
.randomColumn();

// split train and test
var train = sample.filter(ee.Filter.lte("random", 0.8));
var test = sample.filter(ee.Filter.gt("random", 0.8));

// Extract image values
var trainSample = image.sampleRegions({
collection: train,
scale: 10,
properties: ["Class"],
});
var testSample = image.sampleRegions({
collection: test,
scale: 10,
properties: ["Class"],
});

// Legend dict
var legend = {
LULC_class_values: [1, 2, 3, 4, 5, 6],
LULC_class_palette: [
"228B22",
"00BFFF",
"DC143C",
"D2B48C",
"7CFC00",
"F4A460",
],
};

// Train rf model
var rf_model = ee.Classifier.smileRandomForest(50).train(
trainSample,
"Class",
image.bandNames()
);

// 5. Accuracy test
var cm = testSample
.classify(rf_model, "predict")
.errorMatrix("Class", "predict");
print("Confusion matrix", cm, "Accuracy", cm.accuracy(), "Kappa", cm.kappa());

var lulc = image.classify(rf_model, "LULC").toByte().set(legend);
Map.addLayer(lulc, {}, "lulc", true);

// Filter Cloud S2
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);
}

//function to select bands
function selectBands(image) {
var ndvi = image
.expression("(NIR - RED) / (NIR + RED)", {
NIR: image.select("B8"),
RED: image.select("B4"),
})
.rename("NDVI");
```

```
var ndbi = image
.expression("(SWIR - NIR) / (SWIR + NIR)", {
NIR: image.select("B8"),
SWIR: image.select("B11"),
})
.rename("NDBI");

var mndwi = image
.expression("(GREEN - SWIR1) / (GREEN + SWIR1)", {
GREEN: image.select("B3"),
SWIR1: image.select("B11"),
})
.rename("MNDWI");

var ndsli = image
.expression("(RED - SWIR1) / (RED + SWIR1)", {
RED: image.select("B4"),
SWIR1: image.select("B11"),
})
.rename("NDSLI");

//required bands selection
var bands = ["B4", "B3", "B2", "B8", "B11", "B12"];
image = image.select(bands);

// add NDVI, NDBI, NDSLI bands to image
image = image.addBands(ee.Image([ndvi, mndwi, ndbi, ndsli]));
return image;
}

// Legend
//============================================================
// set position of panel
var legend = ui.Panel({
style: {
position: "bottom-left",
padding: "8px 15px",
},
});

// Create legend title
var legendTitle = ui.Label({
value: "LULC legend",
style: {
```

```
fontWeight: "bold",
fontSize: "18px",
margin: "0 0 4px 0",
padding: "0",
},
});

// Add the title to the panel
legend.add(legendTitle);

// Creates and styles 1 row of the legend.
var makeRow = function (color, name) {
// Create the label that is actually the colored box.
var colorBox = ui.Label({
style: {
backgroundColor: "#" + color,
// Use padding to give the box height and width.
padding: "8px",
margin: "0 0 4px 0",
},
});

// Create the label filled with the description text.
var description = ui.Label({
value: name,
style: { margin: "0 0 4px 6px" },
});

// return the panel
return ui.Panel({
widgets: [colorBox, description],
layout: ui.Panel.Layout.Flow("horizontal"),
});
};

// Palette with the colors
var palette = [
"228B22",
"00BFFF",
"DC143C",
"D2B48C",
"7CFC00",
"F4A460",
];
```

```
// name of the legend
var names = [
"Vegetation",
"Water",
"Urban",
"Bare Land",
"Agriculture",
"Sand",
];

// Add color and and names
for (var i = 0; i < names.length; i++) {
legend.add(makeRow(palette[i], names[i]));
}

// add legend to map
Map.add(legend);

// Export lulc to google drive
Export.image.toDrive({
image: lulc,
scale: 10,
region: aoi,
crs: "EPSG:4326",
maxPixels: 1e13,
folder: "LULC",
description: "Pokhara_LULC_2024",
formatOptions: {
cloudOptimized: true,
},
});
```