Code: Cropland Classification using Landsat 8 and Landsat 5 metrics

```
// 1. Set initial variables
print("location of interest", roi)

// 2. Get an image from a suitable collection
// This is the Image collection of Landsat 8 and Landsat 5 Level 2 (Surface reflectance), Collection 2, Tier 1
var l8SR = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2");
var l5SR = ee.ImageCollection("LANDSAT/LT05/C02/T1_L2");

// IMAGE ENHANCEMENT
// 3. Enhance the image with cloud masking
// Define a function for cloud masking for Landsat 8

var maskCloud_Landsat = function (image) {
  var qaMask = image.select('QA_PIXEL').bitwiseAnd(parseInt('11111', 2)).eq(0);
  var saturationMask = image.select('QA_RADSAT').eq(0);

  // Apply the scaling factors to the appropriate bands. This will change the reflectance values.
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);
  var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);

  // Replace the original bands with the scaled ones and apply the masks.
  return image.addBands(opticalBands, null, true)
      .addBands(thermalBands, null, true)
      .updateMask(qaMask)
      .updateMask(saturationMask);
}

// Define a function for cloud masking for Landsat 5
var maskCloud_Landsat5 = function (images) {
  var qaMaskl5 = images.select('QA_PIXEL').bitwiseAnd(parseInt('11111', 2)).eq(0);
  var saturationMaskl5 = images.select('QA_RADSAT').eq(0);

// Apply the scaling factors to the appropriate bands. This will change the reflectance values.
  var opticalBandsl5 = images.select('SR_B.').multiply(0.0000275).add(-0.2);
  var thermalBandsl5 = images.select('ST_B6').multiply(0.00341802).add(149.0);

// Replace the original bands with the scaled ones and apply the masks.
  return images
      .addBands(opticalBandsl5, null, true)
      .addBands(thermalBandsl5, null, true)
      .updateMask(qaMaskl5)
      .updateMask(saturationMaskl5);
}

// Standardize band names L8
    var bands_LS8 = ["SR_B2"  ,"SR_B3"   ,"SR_B4" ,"SR_B5" ,"SR_B6"   ,"SR_B7"   ,
"QA_PIXEL","QA_RADSAT"']
```

```
   var bands_LS5 = ["SR_B1"  ,"SR_B2"   ,"SR_B3" ,"SR_B4" ,"SR_B5"   ,"SR_B7"   ,
"QA_PIXEL","QA_RADSAT"]
   var bands_std = ["blue","green","red","nir","swir1","swir2", "QA_PIXEL","QA_RADSAT"]

var collectionl8 = l8SR
                .filter(ee.Filter.calendarRange(7,10,'month'))
                .map(maskCloud_Landsat) //Mask cloud and cloud shadow from the image
                .select(bands_LS8, bands_std) //Standardize band names

var collectionl5 = l5SR
                .filter(ee.Filter.calendarRange(7,10,'month'))
                .map(maskCloud_Landsat5) //Mask cloud and cloud shadow from the image
                .select(bands_LS5, bands_std) //Standardize band names

// 4. Make a cloud free composite
var imageVisParam_SR_cloudMasked =
                {
                  "bands":["red","green","blue"], // These are the Red - Green - Blue bands on the Landsat 8
satellite
                  "min":0,"max":0.3,       // Minimum reflectance value is 0 (0 is black), and maximum
reflectance value is 40000 (40000 is white), depends on satellite
                };

var imageVisParam_SR_cloudMaskedl5 =
                {
                  "bands":["red_p75","green_p75","blue_p75"], // These are the Red - Green - Blue bands on the
Landsat 8 satellite
                  "min":0,"max":0.3,       // Minimum reflectance value is 0 (0 is black), and maximum
reflectance value is 40000 (40000 is white), depends on satellite
                };

// 5. Calculate an index
var addNDVI = function(image){
  return image.addBands(image.normalizedDifference(['nir', 'red']).rename('ndvi'))
}

var addEVI = function (image){
  var evi = image.expression(
    "2.5*((nir-red)/(nir+6*red-7.5*blue+1))",
    {
      nir : image.select("nir"),
      red : image.select("red"),
      blue : image.select("blue"),
    }
    ).rename("evi")

  return image.addBands(evi)
}
```

```
var collectionEnhanced2013_14 = collectionl8.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2013,2014,'year')).map(addNDVI).map(addEVI)
var collectionEnhanced2015_16 = collectionl8.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2015,2016,'year')).map(addNDVI).map(addEVI)
var collectionEnhanced2017_18 = collectionl8.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2017,2018,'year')).map(addNDVI).map(addEVI)
var collectionEnhanced2019_20 = collectionl8.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2019,2020,'year')).map(addNDVI).map(addEVI)
var collectionEnhanced2021_22 = collectionl8.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2021,2022,'year')).map(addNDVI).map(addEVI)

var collectionEnhanced2006_08 = collectionl5.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2006,2008,'year')).map(addNDVI).map(addEVI)
var collectionEnhanced2009_11 = collectionl5.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2009,2011,'year')).map(addNDVI).map(addEVI)

var NDVIEnhanced2021_22 = collectionl8.filterBounds(roi).map(function(image){return
image.clip(roi)}).filter(ee.Filter.calendarRange(2021,2022,'year')).map(addNDVI)
var NDVIimageEnhanced2021_22 = NDVIEnhanced2021_22.median()

var imageEnhanced2013_14 = collectionEnhanced2013_14.median()
var imageEnhanced2015_16 = collectionEnhanced2015_16.median()
var imageEnhanced2017_18 = collectionEnhanced2017_18.median()
var imageEnhanced2019_20 = collectionEnhanced2019_20.median()
var imageEnhanced2021_22 = collectionEnhanced2021_22.median()

var imageEnhanced2006_08 = collectionEnhanced2006_08.median()
var imageEnhanced2009_11 = collectionEnhanced2009_11.median()

// Merge all the collections
var mergedCollection = collectionEnhanced2013_14
  .merge(collectionEnhanced2015_16)
  .merge(collectionEnhanced2017_18)
  .merge(collectionEnhanced2019_20)
  .merge(collectionEnhanced2021_22);

// Select the Landsat 8 image to use as the reference image
var referenceImage = mergedCollection.median();

// Define a function to apply spectral matching using NDVI and EVI
var spectralMatch = function(image) {
  var ndviDiff = image.select('ndvi').subtract(referenceImage.select('ndvi'));
  var eviDiff = image.select('evi').subtract(referenceImage.select('evi'));
  var ndviRatio = image.select('ndvi').divide(referenceImage.select('ndvi'));
  var eviRatio = image.select('evi').divide(referenceImage.select('evi'));
  var ndviMatched = image.select('nir').add(ndviDiff).multiply(ndviRatio);
  var eviMatched = image.select('nir').add(eviDiff).multiply(eviRatio);
```

```
  var matchedImage =
image.addBands(ndviMatched.rename('NDVI_matched')).addBands(eviMatched.rename('EVI_matched')
);
  return matchedImage;
};

var l8Matched2013_2014 = collectionEnhanced2013_14.map(spectralMatch).median();
var l8Matched2015_2016 = collectionEnhanced2015_16.map(spectralMatch).median();
var l8Matched2017_2018 = collectionEnhanced2017_18.map(spectralMatch).median();
var l8Matched2019_2020 = collectionEnhanced2019_20.map(spectralMatch).median();
var l8Matched2021_2022 = collectionEnhanced2021_22.map(spectralMatch).median();

var l5Matched2006_2008 = collectionEnhanced2006_08.map(spectralMatch).median();
var l5Matched2009_2011 = collectionEnhanced2009_11.map(spectralMatch).median();

// 6. Classification
// 6.1. Obtain training data
  // CHANGE THIS TO SUIT YOUR CLASS SCHEMES and COLOR

  var classes = ["Agriculture","ForestPastures","NonAgriculture"]
  var colorChart = ["green","red","yellow"]
  var visParams_classes = {min: 0, max: 4, palette: colorChart}

var trainingPoints = ee.FeatureCollection([Agriculture, ForestPastures, NonAgriculture])
print("Training Points", trainingPoints)

// 6.2 Use these bands for classification.
  // CHANGE THESE BANDS ACCORDING TO YOUR TARGET
  var predictionBands = ["blue","green","red","nir","swir1","swir2", "ndvi","evi"]
  var classProperty = 'Class';

// 6.3 Sample the composite to generate training data.
  var training = l8Matched2021_2022.select(predictionBands).sampleRegions({
    collection: trainingPoints,
    properties: [classProperty],
    scale: 30
  });

// Load the validation data from your CSV file
  var validationData = ee.FeatureCollection('FilePath');

  // Sample the image at the points and add the class property from the CSV file
  var validation = l8Matched2021_2022.select(predictionBands).sampleRegions({
    collection: validationData,
    properties: [classProperty],
    scale: 30
  });
```

```
// 6.4 Train a Random Forest classifier.
  // Define the hyperparameters
  var nTrees = 200;  // Number of trees in the forest
  var maxDepth =75; // Maximum depth of the trees in the forest

  var classifier = ee.Classifier.smileRandomForest({
    numberOfTrees: nTrees,
    maxNodes: maxDepth,
    seed: 0
  }).train({
    features: training,
    classProperty: classProperty,
  });

// 6.5 Classify the image.
  var classified2013_14 = l8Matched2013_2014.classify(classifier);
  var classified2015_16 = l8Matched2015_2016.classify(classifier);
  var classified2017_18 = l8Matched2017_2018.classify(classifier);
  var classified2019_20 = l8Matched2019_2020.classify(classifier);
  var classified2021_22 = l8Matched2021_2022.classify(classifier);
  var classified2009_11 = l5Matched2009_2011.classify(classifier);
  var classified2006_08 = l5Matched2006_2008.classify(classifier);


var trainedClassifierV = ee.Classifier.smileRandomForest({
  numberOfTrees: nTrees,
  maxNodes: maxDepth,
  seed: 0
}).train({
  features: training,
  classProperty: classProperty,
  inputProperties: predictionBands
});

// Classify the validation data using the trained classifier
var classifiedV = validation.classify(trainedClassifierV);

// Calculate the accuracy assessment to the console
var validationAccuracy = classifiedV.errorMatrix(classProperty, 'classification');
var kappaCoefficient = validationAccuracy.kappa();
var producersAccuracy = validationAccuracy.producersAccuracy();
var consumersAccuracy = validationAccuracy.consumersAccuracy();

// 7 Smoothing
var smoothing = function (classifiedImage){
// Define a kernel.
var kernel = ee.Kernel.circle({radius: 1});
// Perform an erosion followed by a dilation, display.
```

```
var opened = classifiedImage
          .focal_min({kernel: kernel, iterations: 2})
          .focal_max({kernel: kernel, iterations: 2});
  return ee.Image(opened).copyProperties(classifiedImage)
}

var smoothed2013_14 = ee.Image(smoothing(classified2013_14)).clip(roi)
var smoothed2015_16 = ee.Image(smoothing(classified2015_16)).clip(roi)
var smoothed2017_18 = ee.Image(smoothing(classified2017_18)).clip(roi)
var smoothed2019_20 = ee.Image(smoothing(classified2019_20)).clip(roi)
var smoothed2021_22 = ee.Image(smoothing(classified2021_22)).clip(roi)

var smoothed2006_08 = ee.Image(smoothing(classified2006_08)).clip(roi)
var smoothed2009_11 = ee.Image(smoothing(classified2009_11)).clip(roi)

// 8. Calculate area
  var irrigation2013_14 = smoothed2013_14.eq(0)
  var irrigation2015_16 = smoothed2015_16.eq(0)
  var irrigation2017_18 = smoothed2017_18.eq(0)
  var irrigation2019_20 = smoothed2019_20.eq(0)
  var irrigation2021_22 = smoothed2021_22.eq(0)
  var irrigation2006_08 = smoothed2006_08.eq(0)
  var irrigation2009_11 = smoothed2009_11.eq(0)

var areaImage2013_14 = irrigation2013_14.multiply(ee.Image.pixelArea())
var area2013_14 = areaImage2013_14.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
  })
print("Irrigated Area of 2013_14 is ",area2013_14)

var areaImage2015_16 = irrigation2015_16.multiply(ee.Image.pixelArea())
var area2015_16 = areaImage2015_16.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
  })
print("Irrigated Area of 2015_16 is ",area2015_16)

var areaImage2017_18 = irrigation2017_18.multiply(ee.Image.pixelArea())
var area2017_18 = areaImage2017_18.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
```

```
  })
print("Irrigated Area of 2017_18 is ", area2017_18)

var areaImage2019_20 = irrigation2019_20.multiply(ee.Image.pixelArea())
var area2019_20 = areaImage2019_20.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
  })
print("Irrigated Area of 2019_20 is ", area2019_20)


var areaImage2021_22 = irrigation2021_22.multiply(ee.Image.pixelArea())
var area2021_22 = areaImage2021_22.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
  })
print("Irrigated Area of 2021_22 is ", area2021_22)

var areaImage2006_08 = irrigation2006_08.multiply(ee.Image.pixelArea())
var area2006_08 = areaImage2006_08.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
  })
print("Irrigated Area of 2006_08 is ",area2006_08)

var areaImage2009_11 = irrigation2009_11.multiply(ee.Image.pixelArea())
var area2009_11 = areaImage2009_11.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: roi,
  scale: 30,
  maxPixels: 1e10
  })
print("Irrigated Area of 2009_11 is ",area2009_11)
```