Supplementary Materials: More Than Meets the Eye: Using Sentinel-2 to Map Small Plantations in Complex Forest Landscapes

Keiko Nomura and Edward T. A. Mitchard

The figures below show the spectral responses of an example classified map (random seed 0) to show the variation in spectral responses to the different classes. Note especially the differences between the tree crops in the red-edge and mid-infra red regions.

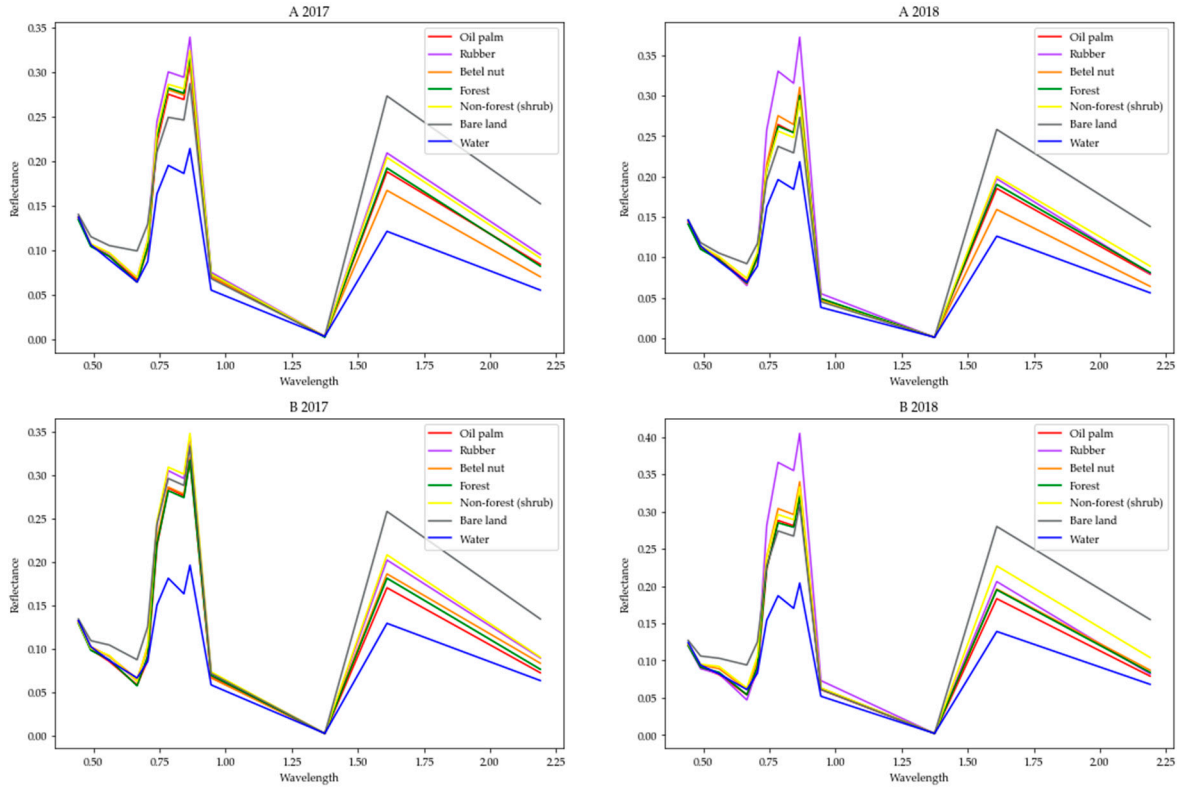

Figure S1. Spectral bands per class.

Google Earth Engine (GEE) code

Instruction: GEE registered users only; find the repository "users/nkeikon/S2/" under the "Reader" in the Script tab on the left.

https://code.earthengine.google.com/?accept_repo=users/nkeikon/S2

For cloning its Git repository, run the following command in a terminal:

https://earthengine.googlesource.com/users/nkeikon/S2

Area A 2017

```
/**** Start of imports. If edited, may not auto-convert in the playground. ****/
var roi = ee.FeatureCollection("users/nkeikon/RemoteSensing/Area_A"),
    palm = ee.FeatureCollection("users/nkeikon/RemoteSensing/palm_A"),
    rubber = ee.FeatureCollection("users/nkeikon/RemoteSensing/rubber_A"),
    betel = ee.FeatureCollection("users/nkeikon/RemoteSensing/betel_A"),
    forest = ee.FeatureCollection("users/nkeikon/RemoteSensing/forest_A"),
    nonforest = ee.FeatureCollection("users/nkeikon/RemoteSensing/nonforest_A"),
    bare = ee.FeatureCollection("users/nkeikon/RemoteSensing/bare_A"),
    water = ee.FeatureCollection("users/nkeikon/RemoteSensing/water_A");
/***** End of imports. If edited, may not auto-convert in the playground. *****/
/*////////////////////////////////////////////////////////////

Below is an example of classification conducted for the study
"More than meets the eye: using Sentinel-2 to map small plantations
in complex forest landscapes" by Keiko Nomura and Edward TA Mitchard.
The default seed '0' is selected for random sampling in this example,
while the study conducted 1,000 classification runs (random seed: 0-999).

*////////////////////////////////////////////////////////////


var areaName = 'A';

var year = '2017';

var startDate = '2017-02-01';

var endDate = '2017-02-28';

var randomSeed = 0;


// Display in the console

print('Area' + " " + areaName +' ' + year);

var checkbox1 = ui.Checkbox('S2 least cloudy', true);

var checkbox2 = ui.Checkbox('NDVI', false);

var checkbox3 = ui.Checkbox('Texture', false);

var checkbox4 = ui.Checkbox('Classified', true);


checkbox1.onChange(function(checked) {Map.layers().get(0).setShown(checked);});

checkbox2.onChange(function(checked) {Map.layers().get(1).setShown(checked);});

checkbox3.onChange(function(checked) {Map.layers().get(2).setShown(checked);});

checkbox4.onChange(function(checked) {Map.layers().get(3).setShown(checked);});


print(checkbox4);

print(checkbox3);

print(checkbox2);

print(checkbox1);


//////////Sentinel-2 data//////////

var s2 = ee.ImageCollection('COPERNICUS/S2')

                    .filterDate(startDate, endDate)

                    .filterBounds(roi);
```

```
// Get the dates of available images

var list = ee.List(s2.aggregate_array("system:time_start")).map(function(d) { return
ee.Date(d)});

print('S2 images of the area during the study period',list);


// Get the cloud score of the images

var getCloudScores = function(img){

    var value = ee.Image(img).get('CLOUDY_PIXEL_PERCENTAGE');

    return ee.Feature(null, {'score': value});

};


var s2clouds = s2.map(getCloudScores);

print ('cloud score', ui.Chart.feature.byFeature(s2clouds));


// Sort images by least cloudy pixel %, select and rename the bands

var s2image = ee.ImageCollection('COPERNICUS/S2')

                    .filterDate(startDate, endDate)

                    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))

                    .sort('CLOUDY_PIXEL_PERCENTAGE')

                    .filterBounds(roi)

                    .map(function(img){

                      var t = img.select([ 'B1','B2','B3','B4','B5','B6','B7','B8','B8A',
'B9','B10', 'B11','B12']).divide(10000);//Rescale to 0-1

                      var out =
t.copyProperties(img).copyProperties(img,['system:time_start']);

                    return out;

                      })

                      .select(['B1','B2','B3','B4','B5','B6','B7','B8','B8A', 'B9','B10',
'B11','B12'],['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2']);


// Obtain the least cloudy image and clip to the ROI

var s2leastCloud = ee.Image(s2image.first());

var s2ROI = s2leastCloud.clip(roi);


var vizParams = {bands: ['red', 'green', 'blue'], min: 0, max: 0.3};

Map.addLayer(s2ROI, vizParams, 'S2 least cloudy');


// Compute the Normalized Difference Vegetation Index (NDVI)

var red = s2ROI.select('red');
```

```
var nir = s2ROI.select('nir');

var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');


var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};

Map.addLayer(ndvi, ndviParams, 'NDVI', false);


// Compute standard deviation of NDVI as texture

var texture = ndvi.reduceNeighborhood({

  reducer: ee.Reducer.stdDev(),

  kernel: ee.Kernel.square(5),

});


Map.addLayer(texture, {min: 0, max: 0.25},'Texture', false);


// Add NDVI and texture bands

var s2final = ee.Image(s2ROI).addBands(ndvi).addBands(texture);


// Add all the appropriate bands

var bands = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2','NDVI','NDVI_stdDev'];

var s2finalWbands = s2final.select(bands);


// Scale the image to 20m

var s2classification =
s2finalWbands.reproject(ee.Projection('EPSG:32647').atScale(20)).reduceResolution({reducer:
ee.Reducer.mean(),maxPixels: 65535});


//////////create training and verification data/////////

/* When conducting multiple runs, use:

   ee.List.sequence(seedMin, seedMax).map(function(n) {

   by setting min and max seed */


// Assign random column to sample

var n = randomSeed;

var randomPalm = palm.randomColumn('random', n);

var randomRubber = rubber.randomColumn('random', n);

var randomBetel = betel.randomColumn('random', n);

var randomForest = forest.randomColumn('random', n);

var randomNonforest = nonforest.randomColumn('random', n);

var randomBare = bare.randomColumn('random', n);
```

```
var randomWater = water.randomColumn('random', n);


// 50:50 for training and testing
var split = 0.5;


var trainingSample = randomBare.filter(ee.Filter.lt('random', split))
  .merge(randomForest.filter(ee.Filter.lt('random', split)))
  .merge(randomNonforest.filter(ee.Filter.lt('random', split)))
  .merge(randomPalm.filter(ee.Filter.lt('random', split)))
  .merge(randomRubber.filter(ee.Filter.lt('random', split)))
  .merge(randomWater.filter(ee.Filter.lt('random', split)))
  .merge(randomBetel.filter(ee.Filter.lt('random', split)));


var testingSample = randomBare.filter(ee.Filter.gte('random', split))
  .merge(randomForest.filter(ee.Filter.gte('random', split)))
  .merge(randomNonforest.filter(ee.Filter.gte('random', split)))
  .merge(randomPalm.filter(ee.Filter.gte('random', split)))
  .merge(randomRubber.filter(ee.Filter.gte('random', split)))
  .merge(randomWater.filter(ee.Filter.gte('random', split)))
  .merge(randomBetel.filter(ee.Filter.gte('random', split)));


// Get the values for training
var training = s2classification.sampleRegions({
  collection: trainingSample,
  properties: ['class'],
  scale: 20,
});


//////////classify and verify/////////
// Create the classifier
var classifier = ee.Classifier.randomForest({
  numberOfTrees: 30,
  variablesPerSplit: 4
  })
    .train(training, 'class');


// Classify the input imagery
var classified = s2classification.classify(classifier,'classification');
```

```
// Create a palette to display the classes
var palette =['ff0000',// palm 0 (red)
              '9933ff',//rubber 1 (purple)
              'FF7F00',//betel 2 (orange)
              '008000',//forest 3 (green)
              'ffff00',//nonforest 4 (yellow)
              'ffffff',//bare 5 (white)
              '0000ff',//river 6 (blue)
];

// Display the classified map
Map.addLayer(classified, {min: 0, max: 6, palette: palette}, 'Classified');
Map.centerObject(roi, 15);

// Get a confusion matrix representing resubstitution accuracy
var trainAccuracy = classifier.confusionMatrix();
print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());

// Sample the input to get validation data
var validation = s2classification.sampleRegions({
  collection: testingSample,
  properties: ['class'],
  scale: 20,
});

// Classify the validation data
var validated = validation.classify(classifier);

// Get a confusion matrix representing expected accuracy
var testAccuracy = validated.errorMatrix('class', 'classification');
print ('Validation accuracy exported to "Tasks"');
//print('Validation error matrix: ', testAccuracy);
//print('Validation overall accuracy: ', testAccuracy.accuracy());

var visualization = classified.visualize({
  palette: palette,
  min: 0,
```

```
  max: 6

});


// Create a legend

var labels = ['Oil palm','Rubber', 'Betel nut', 'Forest', 'Non-forest (shrub)', 'Bare land',
'Water'];

var add_legend = function(title, lbl, pal) {

  var legend = ui.Panel({style: {position: 'bottom-left'}}), entry;

  legend.add(ui.Label({value: title, style: { fontWeight: 'bold', fontSize: '18px', margin:
'0 0 4px 0', padding: '0px' } }));

  for (var x = 0; x < lbl.length; x++){

    entry = [ ui.Label({style:{color: pal[x], border:'1px solid black', margin: '0 0 4px
0'}, value: '  '}),

      ui.Label({ value: labels[x], style: { margin: '0 0 4px 4px' } }) ];

    legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));

  } Map.add(legend); };


add_legend('Legend', labels, palette);


////////// Calculate area by class//////////

var names = ['0 oil palm', '1 rubber', '2 betel nut', '3 forest', '4 non-forest','5 bare
land','6 water'];

var count = classified.eq([0, 1, 2, 3, 4, 5, 6]).rename(names);

var total = count.multiply(ee.Image.pixelArea());

var area = total.reduceRegion(ee.Reducer.sum(), roi, 20);

print('Area by class (m2)', area);


//////////////export images and results//////////////

Export.image.toDrive({

  image: visualization,

  description: 'classifiedMap_colour'+'_'+areaName + year + '_'+ randomSeed,

  region: roi,

  crs: 'EPSG:32647',

  scale: 20

});


Export.image.toDrive({

  image: classified,

  description: 'classifiedMap_raster'+'_'+areaName + year + '_'+ randomSeed,

  region: roi,

  crs: 'EPSG:32647',
```

```
  scale: 20
});


var exportAccuracy = ee.Feature(null, {matrix: testAccuracy.array()});
var exportAccuracyNumber = ee.Feature(null, {matrix: testAccuracy.accuracy()});


Export.table.toDrive({
  collection: ee.FeatureCollection(exportAccuracy),
  description: 'AccuracyMatrix'+'_'+areaName + year + '_'+ randomSeed,
  fileFormat: 'CSV'
});


//////////chart wavelengths by class/////////
// Chart S2 spectral bands
var classifiedImage = classified.select(['classification']);
var bands1 = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2'];
var newImage = s2classification
    .select(bands1)
    .addBands(classifiedImage);


var wavelengths = [0.443, 0.490, 0.560, 0.665, 0.705, 0.740, 0.783, 0.842, 0.865, 0.945,
1.375, 1.610, 2.190];
var options = {
  lineWidth: 1,
  pointSize: 2,
  hAxis: {title: 'Wavelength (micrometers)'},
  vAxis: {title: 'Reflectance'},
  title: 'Spectra in Area'+ ' ' + areaName,
  colors: ['ff0000',// palm 0 (red)
           '9933ff',//rubber 1 (purple)
           'FF7F00',//betel 2 (orange)
           '008000',//forest 3 (green)
           'ffff00',//nonforest 4 (yellow)
           'D3D3D3',//bare 5 (grey)
           '0000ff',//river 6 (blue)
]};


var chart = ui.Chart.image.byClass(
```

```
    newImage, 'classification', roi, ee.Reducer.mean(), 20, labels, wavelengths)
    .setOptions(options);


print(chart);


// Chart indices
var bands2 = ['NDVI','NDVI_stdDev'];
var newImage2 = s2classification
    .select(bands2)
    .addBands(classifiedImage);


var xaxis = ['NDVI', 'Texture'];
var options2 = {
  lineWidth: 1,
  pointSize: 2,
  title: 'NDVI and Texture indices in Area'+ ' ' + areaName,
  colors: ['ff0000',// palm 0 (red)
           '9933ff',//rubber 1 (purple)
           'FF7F00',//betel 2 (orange)
           '008000',//forest 3 (green)
           'ffff00',//nonforest 4 (yellow)
           'D3D3D3',//bare 5 (grey)
           '0000ff',//river 6 (blue)
]};


var chart2 = ui.Chart.image.byClass(
    newImage2, 'classification', roi, ee.Reducer.mean(), 20, labels, xaxis)
    .setOptions(options2);


print(chart2);
```

Area A 2018

```
/**** Start of imports. If edited, may not auto-convert in the playground. ****/
var roi = ee.FeatureCollection("users/nkeikon/RemoteSensing/Area_A"),
    palm = ee.FeatureCollection("users/nkeikon/RemoteSensing/palm_A"),
    rubber = ee.FeatureCollection("users/nkeikon/RemoteSensing/rubber_A"),
    betel = ee.FeatureCollection("users/nkeikon/RemoteSensing/betel_A"),
    forest = ee.FeatureCollection("users/nkeikon/RemoteSensing/forest_A"),
    nonforest = ee.FeatureCollection("users/nkeikon/RemoteSensing/nonforest_A"),
    bare = ee.FeatureCollection("users/nkeikon/RemoteSensing/bare_A"),
    water = ee.FeatureCollection("users/nkeikon/RemoteSensing/water_A");
/***** End of imports. If edited, may not auto-convert in the playground. *****/
/*////////////////////////////////////////////////////////////

Below is an example of classification conducted for the study
"More than meets the eye: using Sentinel-2 to map small plantations
```

```
in complex forest landscapes" by Keiko Nomura and Edward TA Mitchard.
The default seed '0' is selected for random sampling in this example,
while the study conducted 1,000 classification runs (random seed: 0-999).

*////////////////////////////////////////////////////////////

var areaName = 'A';

var year = '2018';

var startDate = '2018-02-01';

var endDate = '2018-02-28';

var randomSeed = 0;


// Display in the console

print('Area' + " " + areaName +' ' + year);

var checkbox1 = ui.Checkbox('S2 composite', true);

var checkbox2 = ui.Checkbox('NDVI', false);

var checkbox3 = ui.Checkbox('Texture', false);

var checkbox4 = ui.Checkbox('Classified', true);


checkbox1.onChange(function(checked) {Map.layers().get(0).setShown(checked);});

checkbox2.onChange(function(checked) {Map.layers().get(1).setShown(checked);});

checkbox3.onChange(function(checked) {Map.layers().get(2).setShown(checked);});

checkbox4.onChange(function(checked) {Map.layers().get(3).setShown(checked);});


print(checkbox4);

print(checkbox3);

print(checkbox2);

print(checkbox1);


//////////Sentinel-2 data/////////

var s2 = ee.ImageCollection('COPERNICUS/S2')

                    .filterDate(startDate, endDate)

                    .filterBounds(roi);


// Get the dates of images

var list = ee.List(s2.aggregate_array("system:time_start")).map(function(d) { return
ee.Date(d)});

print('S2 images of the area during the study period',list);


// Get the cloud score of the images

var getCloudScores = function(img){

    var value = ee.Image(img).get('CLOUDY_PIXEL_PERCENTAGE');
```

```
    return ee.Feature(null, {'score': value});
};


var s2clouds = s2.map(getCloudScores);
print ('cloud score', ui.Chart.feature.byFeature(s2clouds));


// Obtain images with <10% cloudy pixel %, select and rename bands
var s2image = ee.ImageCollection('COPERNICUS/S2')
                    .filterDate(startDate, endDate)
                    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))
                    .filterBounds(roi)
                    .map(function(img){
                      var t = img.select([ 'B1','B2','B3','B4','B5','B6','B7','B8','B8A',
'B9','B10', 'B11','B12']).divide(10000);//Rescale to 0-1
                      var out =
t.copyProperties(img).copyProperties(img,['system:time_start']);
                    return out;
                      })
                      .select(['B1','B2','B3','B4','B5','B6','B7','B8','B8A', 'B9','B10',
'B11','B12'],['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2']);


// Create a median value image and clip to the ROI
var s2median = s2image.median();
var s2medianROI = s2median.clip(roi);
var vizParams = {bands: ['red', 'green', 'blue'], min: 0, max: 0.3};
Map.addLayer(s2medianROI, vizParams, 'S2 composite');


// Compute the Normalized Difference Vegetation Index (NDVI)
var red = s2medianROI.select('red');
var nir = s2medianROI.select('nir');
var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');


var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};
Map.addLayer(ndvi, ndviParams, 'NDVI', false);


// Compute standard deviation of NDVI as texture
var texture = ndvi.reduceNeighborhood({
  reducer: ee.Reducer.stdDev(),
  kernel: ee.Kernel.square(5),
```

```
});


Map.addLayer(texture, {min: 0, max: 0.25},'Texture', false);


// Add NDVI and texture bands

var s2final = ee.Image(s2medianROI).addBands(ndvi).addBands(texture);


// Add all the appropriate bands

var bands = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2','NDVI','NDVI_stdDev'];

var s2finalWbands = s2final.select(bands);


// Scale the image to 20m

var s2classification =
s2finalWbands.reproject(ee.Projection('EPSG:32647').atScale(20)).reduceResolution({reducer:
ee.Reducer.mean(),maxPixels: 65535});


//////////create training and verification data//////////

/* When conducting multiple runs, use:

   ee.List.sequence(seedMin, seedMax).map(function(n) {

   by setting min and max seed */


// Assign random column to sample

var n = randomSeed;

var randomPalm = palm.randomColumn('random', n);

var randomRubber = rubber.randomColumn('random', n);

var randomBetel = betel.randomColumn('random', n);

var randomForest = forest.randomColumn('random', n);

var randomNonforest = nonforest.randomColumn('random', n);

var randomBare = bare.randomColumn('random', n);

var randomWater = water.randomColumn('random', n);


// 50:50 for training and testing

var split = 0.5;


var trainingSample = randomBare.filter(ee.Filter.lt('random', split))

  .merge(randomForest.filter(ee.Filter.lt('random', split)))

  .merge(randomNonforest.filter(ee.Filter.lt('random', split)))

  .merge(randomPalm.filter(ee.Filter.lt('random', split)))

  .merge(randomRubber.filter(ee.Filter.lt('random', split)))
```

```
  .merge(randomWater.filter(ee.Filter.lt('random', split)))
  .merge(randomBetel.filter(ee.Filter.lt('random', split)));

var testingSample = randomBare.filter(ee.Filter.gte('random', split))
  .merge(randomForest.filter(ee.Filter.gte('random', split)))
  .merge(randomNonforest.filter(ee.Filter.gte('random', split)))
  .merge(randomPalm.filter(ee.Filter.gte('random', split)))
  .merge(randomRubber.filter(ee.Filter.gte('random', split)))
  .merge(randomWater.filter(ee.Filter.gte('random', split)))
  .merge(randomBetel.filter(ee.Filter.gte('random', split)));

// Get the values for training
var training = s2classification.sampleRegions({
  collection: trainingSample,
  properties: ['class'],
  scale: 20,
});

//////////classify and verify/////////
// Create the classifier
var classifier = ee.Classifier.randomForest({
  numberOfTrees: 30,
  variablesPerSplit: 4
  })
    .train(training, 'class');

// Classify the input imagery
var classified = s2classification.classify(classifier,'classification');

// Create a palette to display the classes
var palette =['ff0000',// palm 0 (red)
              '9933ff',//rubber 1 (purple)
              'FF7F00',//betel 2 (orange)
              '008000',//forest 3 (green)
              'ffff00',//nonforest 4 (yellow)
              'ffffff',//bare 5 (white)
              '0000ff',//river 6 (blue)
];
```

```
// Display the classified map
Map.addLayer(classified, {min: 0, max: 6, palette: palette}, 'Classified');
Map.centerObject(roi, 15);


// Get a confusion matrix representing resubstitution accuracy
var trainAccuracy = classifier.confusionMatrix();
print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());


// Sample the input to get validation data
var validation = s2classification.sampleRegions({
  collection: testingSample,
  properties: ['class'],
  scale: 20,
});


// Classify the validation data
var validated = validation.classify(classifier);


// Get a confusion matrix representing expected accuracy
var testAccuracy = validated.errorMatrix('class', 'classification');
print('Validation accuracy exported to "Tasks"');
//print('Validation error matrix: ', testAccuracy);
//print('Validation overall accuracy: ', testAccuracy.accuracy());


var visualization = classified.visualize({
  palette: palette,
  min: 0,
  max: 6
});


// Create a legend
var labels = ['Oil palm','Rubber', 'Betel nut', 'Forest', 'Non-forest (shrub)', 'Bare land',
'Water'];
var add_legend = function(title, lbl, pal) {
  var legend = ui.Panel({style: {position: 'bottom-left'}}), entry;
  legend.add(ui.Label({value: title, style: { fontWeight: 'bold', fontSize: '18px', margin:
'0 0 4px 0', padding: '0px' } }));
  for (var x = 0; x < lbl.length; x++){
```

```
    entry = [ ui.Label({style:{color: pal[x], border:'1px solid black', margin: '0 0 4px
0'}, value: '■■'}),

      ui.Label({ value: labels[x], style: { margin: '0 0 4px 4px' } }) ];

    legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));

  } Map.add(legend); };


add_legend('Legend', labels, palette);


////////// Calculate area by class//////////

var names = ['0 oil palm', '1 rubber', '2 betel nut', '3 forest', '4 non-forest','5 bare
land','6 water'];

var count = classified.eq([0, 1, 2, 3, 4, 5, 6]).rename(names);

var total = count.multiply(ee.Image.pixelArea());

var area = total.reduceRegion(ee.Reducer.sum(), roi, 20);

print('Area by class (m2)',area);


//////////////export images and results//////////////

Export.image.toDrive({

  image: visualization,

  description: 'classifiedMap_colour'+'_'+areaName + year + '_'+ randomSeed,

  region: roi,

  crs: 'EPSG:32647',

  scale: 20

});


Export.image.toDrive({

  image: classified,

  description: 'classifiedMap_raster'+'_'+areaName + year + '_'+ randomSeed,

  region: roi,

  crs: 'EPSG:32647',

  scale: 20

});


var exportAccuracy = ee.Feature(null, {matrix: testAccuracy.array()});

var exportAccuracyNumber = ee.Feature(null, {matrix: testAccuracy.accuracy()});


Export.table.toDrive({

  collection: ee.FeatureCollection(exportAccuracy),

  description: 'AccuracyMatrix'+'_'+areaName + year + '_'+ randomSeed,

  fileFormat: 'CSV'
```

```
});


//////////chart wavelengths by class//////////

// Chart S2 spectral bands

var classifiedImage = classified.select(['classification']);

var bands1 = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2'];

var newImage = s2classification

    .select(bands1)

    .addBands(classifiedImage);


var wavelengths = [0.443, 0.490, 0.560, 0.665, 0.705, 0.740, 0.783, 0.842, 0.865, 0.945,
1.375, 1.610, 2.190];

var options = {

  lineWidth: 1,

  pointSize: 2,

  hAxis: {title: 'Wavelength (micrometers)'},

  vAxis: {title: 'Reflectance'},

  title: 'Spectra in Area'+ ' ' + areaName,

  colors: ['ff0000',// palm 0 (red)

              '9933ff',//rubber 1 (purple)

              'FF7F00',//betel 2 (orange)

              '008000',//forest 3 (green)

              'ffff00',//nonforest 4 (yellow)

              'D3D3D3',//bare 5 (grey)

              '0000ff',//river 6 (blue)

]

};


var chart = ui.Chart.image.byClass(

    newImage, 'classification', roi, ee.Reducer.mean(), 20, labels, wavelengths)

    .setOptions(options);


print(chart);


// Chart indices

var bands2 = ['NDVI','NDVI_stdDev'];

var newImage2 = s2classification

    .select(bands2)
```

```
    .addBands(classifiedImage);


var xaxis = ['NDVI', 'Texture'];
var options2 = {
  lineWidth: 1,
  pointSize: 2,
  title: 'NDVI and Texture indices in Area'+ ' ' + areaName,
  colors: ['ff0000',// palm 0 (red)
              '9933ff',//rubber 1 (purple)
              'FF7F00',//betel 2 (orange)
              '008000',//forest 3 (green)
              'ffff00',//nonforest 4 (yellow)
              'D3D3D3',//bare 5 (grey)
              '0000ff',//river 6 (blue)
]
};


var chart2 = ui.Chart.image.byClass(
    newImage2, 'classification', roi, ee.Reducer.mean(), 20, labels, xaxis)
    .setOptions(options2);


print(chart2);
```

Area B 2017

```
/**** Start of imports. If edited, may not auto-convert in the playground. ****/
var roi = ee.FeatureCollection("users/nkeikon/RemoteSensing/Area_B"),
    palm = ee.FeatureCollection("users/nkeikon/RemoteSensing/palm_B"),
    rubber = ee.FeatureCollection("users/nkeikon/RemoteSensing/rubber_B"),
    betel = ee.FeatureCollection("users/nkeikon/RemoteSensing/betel_B"),
    forest = ee.FeatureCollection("users/nkeikon/RemoteSensing/forest_B"),
    nonforest = ee.FeatureCollection("users/nkeikon/RemoteSensing/nonforest_B"),
    bare = ee.FeatureCollection("users/nkeikon/RemoteSensing/bare_B"),
    water = ee.FeatureCollection("users/nkeikon/RemoteSensing/water_B");
/***** End of imports. If edited, may not auto-convert in the playground. *****/
/*////////////////////////////////////////////////////////////

Below is an example of classification conducted for the study
"More than meets the eye: using Sentinel-2 to map small plantations
in complex forest landscapes" by Keiko Nomura and Edward TA Mitchard.
The default seed '0' is selected for random sampling in this example,
while the study conducted 1,000 classification runs (random seed: 0-999).

*////////////////////////////////////////////////////////////


var areaName = 'B';

var year = '2017';

var startDate = '2017-02-01';

var endDate = '2017-02-28';

var randomSeed = 0;
```

```
// Display in the console
print('Area' + " " + areaName +' ' + year);
var checkbox1 = ui.Checkbox('S2 composite', true);
var checkbox2 = ui.Checkbox('NDVI', false);
var checkbox3 = ui.Checkbox('Texture', false);
var checkbox4 = ui.Checkbox('Classified', true);

checkbox1.onChange(function(checked) {Map.layers().get(0).setShown(checked);});
checkbox2.onChange(function(checked) {Map.layers().get(1).setShown(checked);});
checkbox3.onChange(function(checked) {Map.layers().get(2).setShown(checked);});
checkbox4.onChange(function(checked) {Map.layers().get(3).setShown(checked);});

print(checkbox4);
print(checkbox3);
print(checkbox2);
print(checkbox1);

//////////Sentinel-2 data//////////
var s2 = ee.ImageCollection('COPERNICUS/S2')
                    .filterDate(startDate, endDate)
                    .filterBounds(roi);

// Get the dates of images
var list = ee.List(s2.aggregate_array("system:time_start")).map(function(d) { return
ee.Date(d)});
print('S2 images of the area during the study period',list);

// Get the cloud score of the images
var getCloudScores = function(img){
    var value = ee.Image(img).get('CLOUDY_PIXEL_PERCENTAGE');
    return ee.Feature(null, {'score': value});
};

var s2clouds = s2.map(getCloudScores);
print ('cloud score', ui.Chart.feature.byFeature(s2clouds));

// Obtain images with <10% cloudy pixel %, select and rename bands
var s2image = ee.ImageCollection('COPERNICUS/S2')
```

```
                    .filterDate(startDate, endDate)

                    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))

                    .filterBounds(roi)

                    .map(function(img){

                      var t = img.select([ 'B1','B2','B3','B4','B5','B6','B7','B8','B8A',
'B9','B10', 'B11','B12']).divide(10000);//Rescale to 0-1

                      var out =
t.copyProperties(img).copyProperties(img,['system:time_start']);

                    return out;

                      })

                      .select(['B1','B2','B3','B4','B5','B6','B7','B8','B8A', 'B9','B10',
'B11','B12'],['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2']);


// Create a median value image and clip to the ROI

var s2median = s2image.median();

var s2medianROI = s2median.clip(roi);

var vizParams = {bands: ['red', 'green', 'blue'], min: 0, max: 0.3};

Map.addLayer(s2medianROI, vizParams, 'S2 composite');


// Compute the Normalized Difference Vegetation Index (NDVI)

var red = s2medianROI.select('red');

var nir = s2medianROI.select('nir');

var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');


var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};

Map.addLayer(ndvi, ndviParams, 'NDVI', false);


// Compute standard deviation of NDVI as texture

var texture = ndvi.reduceNeighborhood({

  reducer: ee.Reducer.stdDev(),

  kernel: ee.Kernel.square(5),

});


Map.addLayer(texture, {min: 0, max: 0.25},'Texture', false);


// Add NDVI and texture bands

var s2final = ee.Image(s2medianROI).addBands(ndvi).addBands(texture);


// Add all the appropriate bands
```

```
var bands = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2','NDVI','NDVI_stdDev'];

var s2finalWbands = s2final.select(bands);


// Scale the image to 20m

var s2classification =
s2finalWbands.reproject(ee.Projection('EPSG:32647').atScale(20)).reduceResolution({reducer:
ee.Reducer.mean(),maxPixels: 65535});


//////////create training and verification data//////////

/* When conducting multiple runs, use:

   ee.List.sequence(seedMin, seedMax).map(function(n) {

   by setting min and max seed */


// Assign random column to sample

var n = randomSeed;

var randomPalm = palm.randomColumn('random', n);

var randomRubber = rubber.randomColumn('random', n);

var randomBetel = betel.randomColumn('random', n);

var randomForest = forest.randomColumn('random', n);

var randomNonforest = nonforest.randomColumn('random', n);

var randomBare = bare.randomColumn('random', n);

var randomWater = water.randomColumn('random', n);


// 50:50 for training and testing

var split = 0.5;


var trainingSample = randomBare.filter(ee.Filter.lt('random', split))

  .merge(randomForest.filter(ee.Filter.lt('random', split)))

  .merge(randomNonforest.filter(ee.Filter.lt('random', split)))

  .merge(randomPalm.filter(ee.Filter.lt('random', split)))

  .merge(randomRubber.filter(ee.Filter.lt('random', split)))

  .merge(randomWater.filter(ee.Filter.lt('random', split)))

  .merge(randomBetel.filter(ee.Filter.lt('random', split)));


var testingSample = randomBare.filter(ee.Filter.gte('random', split))

  .merge(randomForest.filter(ee.Filter.gte('random', split)))

  .merge(randomNonforest.filter(ee.Filter.gte('random', split)))

  .merge(randomPalm.filter(ee.Filter.gte('random', split)))

  .merge(randomRubber.filter(ee.Filter.gte('random', split)))
```

```
  .merge(randomWater.filter(ee.Filter.gte('random', split)))
  .merge(randomBetel.filter(ee.Filter.gte('random', split)));


// Get the values for training
var training = s2classification.sampleRegions({
  collection: trainingSample,
  properties: ['class'],
  scale: 20,
});


//////////classify and verify//////////
// Create the classifier
var classifier = ee.Classifier.randomForest({
  numberOfTrees: 30,
  variablesPerSplit: 4
  })
    .train(training, 'class');


// Classify the input imagery
var classified = s2classification.classify(classifier,'classification');


// Create a palette to display the classes
var palette =['ff0000',// palm 0 (red)
              '9933ff',//rubber 1 (purple)
              'FF7F00',//betel 2 (orange)
              '008000',//forest 3 (green)
              'ffff00',//nonforest 4 (yellow)
              'ffffff',//bare 5 (white)
              '0000ff',//river 6 (blue)
];


// Display the classified map
Map.addLayer(classified, {min: 0, max: 6, palette: palette}, 'Classified');
Map.centerObject(roi, 15);


// Get a confusion matrix representing resubstitution accuracy
var trainAccuracy = classifier.confusionMatrix();
print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());
```

```
// Sample the input to get validation data
var validation = s2classification.sampleRegions({
  collection: testingSample,
  properties: ['class'],
  scale: 20,
});

// Classify the validation data
var validated = validation.classify(classifier);

// Get a confusion matrix representing expected accuracy
var testAccuracy = validated.errorMatrix('class', 'classification');
//print('Validation error matrix: ', testAccuracy);
//print('Validation overall accuracy: ', testAccuracy.accuracy());
print('Validation accuracy exported to "Tasks"');

var visualization = classified.visualize({
  palette: palette,
  min: 0,
  max: 6
});

// Create a legend
var labels = ['Oil palm','Rubber', 'Betel nut', 'Forest', 'Non-forest (shrub)', 'Bare land',
'Water'];
var add_legend = function(title, lbl, pal) {
  var legend = ui.Panel({style: {position: 'bottom-left'}}), entry;
  legend.add(ui.Label({value: title, style: { fontWeight: 'bold', fontSize: '18px', margin:
'0 0 4px 0', padding: '0px' } }));
  for (var x = 0; x < lbl.length; x++){
    entry = [ ui.Label({style:{color: pal[x], border:'1px solid black', margin: '0 0 4px
0'}, value: '■■'}),
      ui.Label({ value: labels[x], style: { margin: '0 0 4px 4px' } }) ];
    legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));
  } Map.add(legend); };

add_legend('Legend', labels, palette);

////////// Calculate area by class//////////
```

```
var names = ['0 oil palm', '1 rubber', '2 betel nut', '3 forest', '4 non-forest','5 bare
land','6 water'];

var count = classified.eq([0, 1, 2, 3, 4, 5, 6]).rename(names);

var total = count.multiply(ee.Image.pixelArea());

var area = total.reduceRegion(ee.Reducer.sum(), roi, 20);

print('Area by class (m2)',area);


//////////////export images and results//////////////

Export.image.toDrive({

  image: visualization,

  description: 'classifiedMap_colour'+'_'+areaName + year + '_'+ randomSeed,

  region: roi,

  crs: 'EPSG:32647',

  scale: 20

});


Export.image.toDrive({

  image: classified,

  description: 'classifiedMap_raster'+'_'+areaName + year + '_'+ randomSeed,

  region: roi,

  crs: 'EPSG:32647',

  scale: 20

});


var exportAccuracy = ee.Feature(null, {matrix: testAccuracy.array()});

var exportAccuracyNumber = ee.Feature(null, {matrix: testAccuracy.accuracy()});


Export.table.toDrive({

  collection: ee.FeatureCollection(exportAccuracy),

  description: 'AccuracyMatrix'+'_'+areaName + year + '_'+ randomSeed,

  fileFormat: 'CSV'

});


//////////chart wavelengths by class//////////

// Chart S2 spectral bands

var classifiedImage = classified.select(['classification']);

var bands1 = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2'];

var newImage = s2classification
```

```
    .select(bands1)
    .addBands(classifiedImage);


var wavelengths = [0.443, 0.490, 0.560, 0.665, 0.705, 0.740, 0.783, 0.842, 0.865, 0.945,
1.375, 1.610, 2.190];
var options = {
  lineWidth: 1,
  pointSize: 2,
  hAxis: {title: 'Wavelength (micrometers)'},
  vAxis: {title: 'Reflectance'},
  title: 'Spectra in Area'+ ' ' + areaName,
  colors: ['ff0000',// palm 0 (red)
           '9933ff',//rubber 1 (purple)
           'FF7F00',//betel 2 (orange)
           '008000',//forest 3 (green)
           'ffff00',//nonforest 4 (yellow)
           'D3D3D3',//bare 5 (grey)
           '0000ff',//river 6 (blue)
]};


var chart = ui.Chart.image.byClass(
    newImage, 'classification', roi, ee.Reducer.mean(), 20, labels, wavelengths)
    .setOptions(options);


print(chart);


// Chart indices
var bands2 = ['NDVI','NDVI_stdDev'];
var newImage2 = s2classification
    .select(bands2)
    .addBands(classifiedImage);


var xaxis = ['NDVI', 'Texture'];
var options2 = {
  lineWidth: 1,
  pointSize: 2,
  title: 'NDVI and Texture indices in Area'+ ' ' + areaName,
  colors: ['ff0000',// palm 0 (red)
           '9933ff',//rubber 1 (purple)
```

```
            'FF7F00',//betel 2 (orange)
            '008000',//forest 3 (green)
            'ffff00',//nonforest 4 (yellow)
            'D3D3D3',//bare 5 (grey)
            '0000ff',//river 6 (blue)
]};

var chart2 = ui.Chart.image.byClass(
    newImage2, 'classification', roi, ee.Reducer.mean(), 20, labels, xaxis)
    .setOptions(options2);

print(chart2);
```

Area B 2018

```
/**** Start of imports. If edited, may not auto-convert in the playground. ****/
var roi = ee.FeatureCollection("users/nkeikon/RemoteSensing/Area_B"),
    palm = ee.FeatureCollection("users/nkeikon/RemoteSensing/palm_B"),
    rubber = ee.FeatureCollection("users/nkeikon/RemoteSensing/rubber_B"),
    betel = ee.FeatureCollection("users/nkeikon/RemoteSensing/betel_B"),
    forest = ee.FeatureCollection("users/nkeikon/RemoteSensing/forest_B"),
    nonforest = ee.FeatureCollection("users/nkeikon/RemoteSensing/nonforest_B"),
    bare = ee.FeatureCollection("users/nkeikon/RemoteSensing/bare_B"),
    water = ee.FeatureCollection("users/nkeikon/RemoteSensing/water_B");
/***** End of imports. If edited, may not auto-convert in the playground. *****/
/*////////////////////////////////////////////////////////////

Below is an example of classification conducted for the study
"More than meets the eye: using Sentinel-2 to map small plantations
in complex forest landscapes" by Keiko Nomura and Edward TA Mitchard.
The default seed '0' is selected for random sampling in this example,
while the study conducted 1,000 classification runs (random seed: 0-999).

*////////////////////////////////////////////////////////////


var areaName = 'B';

var year = '2018';

var startDate = '2018-03-01';

var endDate = '2018-03-31';

var randomSeed = 0;


// Display in the console

print('Area' + " " + areaName +' ' + year);

var checkbox1 = ui.Checkbox('S2 composite', true);

var checkbox2 = ui.Checkbox('NDVI', false);

var checkbox3 = ui.Checkbox('Texture', false);

var checkbox4 = ui.Checkbox('Classified', true);


checkbox1.onChange(function(checked) {Map.layers().get(0).setShown(checked);});

checkbox2.onChange(function(checked) {Map.layers().get(1).setShown(checked);});
```

```
checkbox3.onChange(function(checked) {Map.layers().get(2).setShown(checked);});

checkbox4.onChange(function(checked) {Map.layers().get(3).setShown(checked);});


print(checkbox4);

print(checkbox3);

print(checkbox2);

print(checkbox1);


//////////Sentinel-2 data//////////

var s2 = ee.ImageCollection('COPERNICUS/S2')

                    .filterDate(startDate, endDate)

                    .filterBounds(roi);


// Get the dates of images

var list = ee.List(s2.aggregate_array("system:time_start")).map(function(d) { return
ee.Date(d)});

print('S2 images of the area during the study period',list);


// Get the cloud score of the images

var getCloudScores = function(img){

    var value = ee.Image(img).get('CLOUDY_PIXEL_PERCENTAGE');

    return ee.Feature(null, {'score': value});

};


var s2clouds = s2.map(getCloudScores);

print ('cloud score', ui.Chart.feature.byFeature(s2clouds));


// Obtain images with <10% cloudy pixel %, select and rename bands

var s2image = ee.ImageCollection('COPERNICUS/S2')

                    .filterDate(startDate, endDate)

                    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))

                    .filterBounds(roi)

                    .map(function(img){

                      var t = img.select([ 'B1','B2','B3','B4','B5','B6','B7','B8','B8A',
'B9','B10', 'B11','B12']).divide(10000);//Rescale to 0-1

                      var out =
t.copyProperties(img).copyProperties(img,['system:time_start']);

                    return out;

                      })

                      .select(['B1','B2','B3','B4','B5','B6','B7','B8','B8A', 'B9','B10',
```

```
'B11','B12'],['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2']);


// Create a median value image and clip to the ROI

var s2median = s2image.median();

var s2medianROI = s2median.clip(roi);

var vizParams = {bands: ['red', 'green', 'blue'], min: 0, max: 0.3};

Map.addLayer(s2medianROI, vizParams, 'S2 composite');


// Compute the Normalized Difference Vegetation Index (NDVI)

var red = s2medianROI.select('red');

var nir = s2medianROI.select('nir');

var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');


var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};

Map.addLayer(ndvi, ndviParams, 'NDVI', false);


// Compute standard deviation of NDVI as texture

var texture = ndvi.reduceNeighborhood({

  reducer: ee.Reducer.stdDev(),

  kernel: ee.Kernel.square(5),

});


Map.addLayer(texture, {min: 0, max: 0.25},'Texture', false);


// Add NDVI and texture bands

var s2final = ee.Image(s2medianROI).addBands(ndvi).addBands(texture);


// Add all the appropriate bands

var bands = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2','NDVI','NDVI_stdDev'];

var s2finalWbands = s2final.select(bands);


// Scale the image to 20m

var s2classification =
s2finalWbands.reproject(ee.Projection('EPSG:32647').atScale(20)).reduceResolution({reducer:
ee.Reducer.mean(),maxPixels: 65535});


//////////create training and verification data//////////

/* When conducting multiple runs, use:

   ee.List.sequence(seedMin, seedMax).map(function(n) {
```

```
   by setting min and max seed */


// Assign random column to sample
var n = randomSeed;
var randomPalm = palm.randomColumn('random', n);
var randomRubber = rubber.randomColumn('random', n);
var randomBetel = betel.randomColumn('random', n);
var randomForest = forest.randomColumn('random', n);
var randomNonforest = nonforest.randomColumn('random', n);
var randomBare = bare.randomColumn('random', n);
var randomWater = water.randomColumn('random', n);


// 50:50 for training and testing
var split = 0.5;


var trainingSample = randomBare.filter(ee.Filter.lt('random', split))
  .merge(randomForest.filter(ee.Filter.lt('random', split)))
  .merge(randomNonforest.filter(ee.Filter.lt('random', split)))
  .merge(randomPalm.filter(ee.Filter.lt('random', split)))
  .merge(randomRubber.filter(ee.Filter.lt('random', split)))
  .merge(randomWater.filter(ee.Filter.lt('random', split)))
  .merge(randomBetel.filter(ee.Filter.lt('random', split)));


var testingSample = randomBare.filter(ee.Filter.gte('random', split))
  .merge(randomForest.filter(ee.Filter.gte('random', split)))
  .merge(randomNonforest.filter(ee.Filter.gte('random', split)))
  .merge(randomPalm.filter(ee.Filter.gte('random', split)))
  .merge(randomRubber.filter(ee.Filter.gte('random', split)))
  .merge(randomWater.filter(ee.Filter.gte('random', split)))
  .merge(randomBetel.filter(ee.Filter.gte('random', split)));


// Get the values for training
var training = s2classification.sampleRegions({
  collection: trainingSample,
  properties: ['class'],
  scale: 20,
});
```

```
//////////classify and verify//////////
// Create the classifier
var classifier = ee.Classifier.randomForest({
  numberOfTrees: 30,
  variablesPerSplit: 4
  })
    .train(training, 'class');

// Classify the input imagery
var classified = s2classification.classify(classifier,'classification');

// Create a palette to display the classes
var palette =['ff0000',// palm 0 (red)
              '9933ff',//rubber 1 (purple)
              'FF7F00',//betel 2 (orange)
              '008000',//forest 3 (green)
              'ffff00',//nonforest 4 (yellow)
              'ffffff',//bare 5 (white)
              '0000ff',//river 6 (blue)
];

// Display the classified map
Map.addLayer(classified, {min: 0, max: 6, palette: palette}, 'Classified');
Map.centerObject(roi, 15);

// Get a confusion matrix representing resubstitution accuracy
var trainAccuracy = classifier.confusionMatrix();
print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());

// Sample the input to get validation data
var validation = s2classification.sampleRegions({
  collection: testingSample,
  properties: ['class'],
  scale: 20,
});

// Classify the validation data
var validated = validation.classify(classifier);
```

```
// Get a confusion matrix representing expected accuracy
var testAccuracy = validated.errorMatrix('class', 'classification');
//print('Validation error matrix: ', testAccuracy);
//print('Validation overall accuracy: ', testAccuracy.accuracy());
print('Validation accuracy exported to "Tasks"');


var visualization = classified.visualize({
  palette: palette,
  min: 0,
  max: 6
});


// Create a legend
var labels = ['Oil palm','Rubber', 'Betel nut', 'Forest', 'Non-forest (shrub)', 'Bare land',
'Water'];
var add_legend = function(title, lbl, pal) {
  var legend = ui.Panel({style: {position: 'bottom-left'}}), entry;
  legend.add(ui.Label({value: title, style: { fontWeight: 'bold', fontSize: '18px', margin:
'0 0 4px 0', padding: '0px' } }));
  for (var x = 0; x < lbl.length; x++){
    entry = [ ui.Label({style:{color: pal[x], border:'1px solid black', margin: '0 0 4px
0'}, value: '  '}),
      ui.Label({ value: labels[x], style: { margin: '0 0 4px 4px' } }) ];
    legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));
  } Map.add(legend);
  };


add_legend('Legend', labels, palette);


//////////calculate area by class/////////
var names = ['0 oil palm', '1 rubber', '2 betel nut', '3 forest', '4 non-forest','5 bare
land','6 water'];
var count = classified.eq([0, 1, 2, 3, 4, 5, 6]).rename(names);
var total = count.multiply(ee.Image.pixelArea());
var area = total.reduceRegion(ee.Reducer.sum(), roi, 20);
print('Area by class (m2)',area);


//////////export images and results/////////
Export.image.toDrive({
```

```
  image: visualization,
  description: 'classifiedMap_colour'+'_'+areaName + year + '_'+ randomSeed,
  region: roi,
  crs: 'EPSG:32647',
  scale: 20
});


Export.image.toDrive({
  image: classified,
  description: 'classifiedMap_raster'+'_'+areaName + year + '_'+ randomSeed,
  region: roi,
  crs: 'EPSG:32647',
  scale: 20
});


var exportAccuracy = ee.Feature(null, {matrix: testAccuracy.array()});
var exportAccuracyNumber = ee.Feature(null, {matrix: testAccuracy.accuracy()});


Export.table.toDrive({
  collection: ee.FeatureCollection(exportAccuracy),
  description: 'AccuracyMatrix'+'_'+areaName + year + '_'+ randomSeed,
  fileFormat: 'CSV'
});


//////////chart wavelengths by class//////////
var classifiedImage = classified.select(['classification']);
var bands1 = ['aerosol', 'blue', 'green', 'red', 'red1','red2','red3','nir','red4','h2o',
'cirrus','swir1', 'swir2'];
var newImage = s2classification
    .select(bands1)
    .addBands(classifiedImage);


var wavelengths = [0.443, 0.490, 0.560, 0.665, 0.705, 0.740, 0.783, 0.842, 0.865, 0.945,
1.375, 1.610, 2.190];
var options = {
  lineWidth: 1,
  pointSize: 2,
  hAxis: {title: 'Wavelength (micrometers)'},
  vAxis: {title: 'Reflectance'},
  title: 'Spectra in Area'+ ' ' + areaName,
```

```
  colors: ['ff0000',// palm 0 (red)
           '9933ff',//rubber 1 (purple)
           'FF7F00',//betel 2 (orange)
           '008000',//forest 3 (green)
           'ffff00',//nonforest 4 (yellow)
           'D3D3D3',//bare 5 (grey)
           '0000ff',//river 6 (blue)
]};

var chart = ui.Chart.image.byClass(
    newImage, 'classification', roi, ee.Reducer.mean(), 20, labels, wavelengths)
    .setOptions(options);

print(chart);

// Chart indices
var bands2 = ['NDVI','NDVI_stdDev'];
var newImage2 = s2classification
    .select(bands2)
    .addBands(classifiedImage);

var xaxis = ['NDVI', 'Texture'];
var options2 = {
  lineWidth: 1,
  pointSize: 2,
  title: 'NDVI and Texture indices in Area'+ ' ' + areaName,
  colors: ['ff0000',// palm 0 (red)
           '9933ff',//rubber 1 (purple)
           'FF7F00',//betel 2 (orange)
           '008000',//forest 3 (green)
           'ffff00',//nonforest 4 (yellow)
           'D3D3D3',//bare 5 (grey)
           '0000ff',//river 6 (blue)
]};

var chart2 = ui.Chart.image.byClass(
    newImage2, 'classification', roi, ee.Reducer.mean(), 20, labels, xaxis)
    .setOptions(options2);
```

```
print(chart2);
```