```
// Specify the year to retrieve training data from
var year = 2019;

// Export chosen areas
// Quad 1: Northeast part of the country, contains many pineapple plantations
var q1 = ee.Geometry.Polygon([
  [
    [-84.53572969041575, 10.255752237034255],
    [-83.6430905302595, 10.255752237034255],
    [-83.6430905302595, 10.658195193369174],
    [-84.53572969041575, 10.658195193369174],
    [-84.53572969041575, 10.255752237034255]
  ]
]);
// Quad 2: Southwest part of the country, contain pineaple plantations
var q2 =  ee.Geometry.Polygon([
  [
    [-83.59963338015413, 9.07016690516574],
    [-83.26180379031038, 9.07016690516574],
    [-83.26180379031038, 9.51469409273685],
    [-83.59963338015413, 9.51469409273685],
    [-83.59963338015413, 9.07016690516574]
  ]
]);
// Quad 3: Drier area in the northwest, has no plantations
var q3 =  ee.Geometry.Polygon([
  [
    [-85.4997799833845, 10.353034423442695],
    [-85.04659394822825, 10.353034423442695],
    [-85.04659394822825, 10.633901442810517],
    [-85.4997799833845, 10.633901442810517],
    [-85.4997799833845, 10.353034423442695]
  ]
]);
// Quad 4: Central western area of the country, contains major urban landscapes
and no plantations
var q4 =  ee.Geometry.Polygon([
  [
    [-84.43410148793016, 9.639317707774007],
    [-83.94520988636766, 9.639317707774007],
```

```
    [-83.94520988636766, 10.018194587884272],
    [-84.43410148793016, 10.018194587884272],
    [-84.43410148793016, 9.639317707774007]
  ]
]);

// Combine quadrants into a single geometry
var combinedRegion = ee.Geometry.MultiPolygon([
  q1.coordinates(),
  q2.coordinates(),
  q3.coordinates(),
  q4.coordinates()
]);

// Use the combinedRegion as the new ROI
var roi = combinedRegion;

// Define the function to add NDVI to each image
function addNDVI(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
}

// Load Sentinel-2 image collection (Harmonized Surface Reflectance)
var sentinel2 = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
  .filterBounds(roi)
  .filterDate(year + '-01-01', year + '-03-31')
  .map(addNDVI);

// Reduce the image collection to get the pixel with the highest NDVI value
var bestNDVIImage = sentinel2.qualityMosaic('NDVI').clip(roi);

// Define the function to add indices
function addIndices(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
  var savi = image.expression(
    '((NIR - Red) / (NIR + Red + 0.5)) * 1.5',
    {
      'NIR': image.select('B8'),
      'Red': image.select('B4')
    }
  ).rename('SAVI');
  var ndmi = image.normalizedDifference(['B8', 'B11']).rename('NDMI');
  var ndwi = image.normalizedDifference(['B3', 'B8']).rename('NDWI');
```

```
  return image.addBands([ndvi, savi, ndmi, ndwi]);
}

// Apply indices to the best NDVI image
var sentinel2WithIndices = addIndices(bestNDVIImage);

var bands = ['B2', 'B3', 'B4', 'B8', 'B11', 'NDVI', 'SAVI', 'NDMI', 'NDWI'];
var pre_normalized_sentinel = sentinel2WithIndices.select(bands).toFloat();

// Print the bands of the normalized image
print('Sentinel Image Bands:', pre_normalized_sentinel.bandNames());
// Function to calculate and print min and max values across all quadrants
function calculateMinMaxCombined(image, region) {
  // Use reduceRegion to compute min and max across the combined region
  var stats = image.reduceRegion({
    reducer: ee.Reducer.minMax(),  // Compute min and max values
    geometry: region,
    scale: 10,  // Match Sentinel-2 resolution
    maxPixels: 1e13
  });

  // Print the min and max values for the combined region
  print('Min and Max values for all quadrants combined', stats);
}

// Calculate and print min and max values across all quadrants
calculateMinMaxCombined(pre_normalized_sentinel, roi);


function normalizeImage(image) {
  // Normalize Sentinel-2 bands (0 to 10,000)
  var b2_norm = image.select('B2').subtract(0).divide(10000).rename('B2_norm');
  var b3_norm = image.select('B3').subtract(0).divide(10000).rename('B3_norm');
  var b4_norm = image.select('B4').subtract(0).divide(10000).rename('B4_norm');
  var b8_norm = image.select('B8').subtract(0).divide(10000).rename('B8_norm');
  var b11_norm =
image.select('B11').subtract(0).divide(10000).rename('B11_norm');

  // Normalize indices (-1 to 1)
  var ndvi_norm = image.select('NDVI').add(1).divide(2).rename('NDVI_norm'); //
[-1, 1] -> [0, 1]
  var savi_norm = image.select('SAVI').add(1.5).divide(3).rename('SAVI_norm');
```

```
  var ndmi_norm = image.select('NDMI').add(1).divide(2).rename('NDMI_norm');
  var ndwi_norm = image.select('NDWI').add(1).divide(2).rename('NDWI_norm');

  // Combine normalized bands into a single image
  var normalizedImage = image
    .addBands([b2_norm, b3_norm, b4_norm, b8_norm, b11_norm, ndvi_norm,
savi_norm, ndmi_norm, ndwi_norm])
    .select([
      'B2_norm', 'B3_norm', 'B4_norm', 'B8_norm', 'B11_norm',
      'NDVI_norm', 'SAVI_norm', 'NDMI_norm', 'NDWI_norm'
    ]); // Select only normalized bands

  return normalizedImage;
}

// Apply normalization to the Sentinel-2 image
var finalNormalizedImage = normalizeImage(pre_normalized_sentinel);

// Print the normalized image's bands for verification
print('Normalized Image Bands:', finalNormalizedImage.bandNames());
// Function to calculate and print min and max values across all quadrants

// Visualization
Map.setCenter(-84.09262671948392, 9.92844479883414);
Map.addLayer(finalNormalizedImage, {bands: ['B4_norm', 'B3_norm', 'B2_norm'],
min: 0, max: .3}, 'Normalized RGB Image');

// Define pineapple plantations feature collection
var pineapple = ee.FeatureCollection('projects/gbsc-gcp-lab-emordeca/assets/
costa_rica/plantations/2019/CRpineapple2019');

// Rasterize the pineapple plantations
var rasterized = pineapple
  .map(function(feature) {
    return feature.set('value', 1);  // Assign a value to each feature
  })
  .reduceToImage({
    properties: ['value'],
    reducer: ee.Reducer.first()  // Use 'first' to assign the value
  })
  .rename('class')  // Rename the band to "class"
  .unmask(0);  // Set unmasked (no data) areas to 0

var finalMaskImage = rasterized.clip(roi);
```

```
Map.addLayer(finalMaskImage, {min: 0, max: 1, palette: ['white', 'green']},
'Rasterized Pineapple Plantations');


// Clip images and masks to quadrants and reproject
var mask_q1 = finalMaskImage.clip(q1).reproject({crs:
finalNormalizedImage.projection(), scale: 10});
var mask_q2 = finalMaskImage.clip(q2).reproject({crs:
finalNormalizedImage.projection(), scale: 10});
var mask_q3 = finalMaskImage.clip(q3).reproject({crs:
finalNormalizedImage.projection(), scale: 10});
var mask_q4 = finalMaskImage.clip(q4).reproject({crs:
finalNormalizedImage.projection(), scale: 10});

var image_q1 = finalNormalizedImage.clip(q1).reproject({crs:
finalNormalizedImage.projection(), scale: 10});
var image_q2 = finalNormalizedImage.clip(q2).reproject({crs:
finalNormalizedImage.projection(), scale: 10});
var image_q3 = finalNormalizedImage.clip(q3).reproject({crs:
finalNormalizedImage.projection(), scale: 10});
var image_q4 = finalNormalizedImage.clip(q4).reproject({crs:
finalNormalizedImage.projection(), scale: 10});



// Function to export mask and image for a specific quadrant
function exportQuadrant(mask, image, region, name, year) {
  Export.image.toDrive({
    image: mask,
    description: 'mask_' + name + '_' + year,
    folder: 'DeepLearningPineappleData',
    scale: 10,
    region: region,
    maxPixels: 1e13,
    crs: 'EPSG:4326'
  });

  Export.image.toDrive({
    image: image,
    description: 'image_' + name + '_' + year,
    folder: 'DeepLearningPineappleData',
    scale: 10,
    region: region,
    maxPixels: 1e13,
```

```
    crs: 'EPSG:4326'
  });
}

var image_q1_float32 = image_q1.toFloat();
var image_q2_float32 = image_q2.toFloat();
var image_q3_float32 = image_q3.toFloat();
var image_q4_float32 = image_q4.toFloat();

// Export each quadrant using the export function
exportQuadrant(mask_q1, image_q1_float32, q1, 'q1', year);
exportQuadrant(mask_q2, image_q2_float32, q2, 'q2', year);
exportQuadrant(mask_q3, image_q3_float32, q3, 'q3', year);
exportQuadrant(mask_q4, image_q4_float32, q4, 'q4', year);
```