```
//
/*
* This script executes an image classification procedure on a pre-Haiyan image stack consisting of the following datasets:
*
* Landsat- 7 and 8, comprised of 7 bands and 5 indices (total = 12)
*
*
* A total of 12 layers were classified using Random Forest (RF) algorithm with 5 land cover types found in Leyte, Philippines. Accuracy assessments were done, of which overall accuracy, user's and producer's accuracies were computed.


/*******************************
 DEFINE RANDOM SEED
*******************************/
var seed = 1500;
/*******************************
 DEFINE EXTENT AND VIEW
*******************************/
// Set center of map view
Map.setCenter(124.7,11.0, 8); // Zoom in and center display to Leyte region, the Philippines
// Define and display box extents covering Leyte region
var box = ee.Geometry.Rectangle(124.2,9.8, 125.4,11.6);
/*******************************
 LOAD DATASETS
*******************************/
// LANDSAT
// Load Landsat image assets
var leyte_pre = ee.Image ('users/Ghaffarian-Saman/Leyte_Pre');
// Mosaic Landsat image assets
var composite = ee.ImageCollection(leyte_pre).mosaic()
 .select([0,1,2,3,4,5,6,7],['B2', 'B3', 'B4', 'B5', 'B6', 'B10', 'B7', 'YR']);
```

```
// Calculate indices from Landsat bands

var nir = composite.select(['B5']);

var ndvi = composite.normalizedDifference(['B5', 'B4']); // Normalised Difference Vegetation Index (NDVI)

var lswi = composite.normalizedDifference(['B5', 'B6']); // Land Surface Water Index (LSWI)

var ndti = composite.normalizedDifference(['B6', 'B7']); // Normalised Difference Till Index (NDTI)

var stvi = composite.expression('((b("B6") - b("B4")) / (b("B6") + b("B4") + 0.1)) * (1.1 - (b("B7") / 2))'); // SoilAdjusted Total Vegetation Index (SATVI)

var evi = composite.expression('2.5 * ((b("B5") - b("B4")) / (b("B5") + 6 * b("B4") - 7.5 * b("B2") + 1))'); // Enhanced Vegetation Index


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

 CREATE COMPOSITE STACK

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

// Rename band filenames in metadata

ndvi = ndvi.rename('NDVI');

lswi = lswi.rename('LSWI');

ndti = ndti.rename('NDTI');

stvi = stvi.rename('SATVI');

evi = evi.rename('EVI');


var stackLandsat = composite.addBands(ndvi).addBands(lswi).addBands(ndti).addBands(stvi).addBands(evi);

var bandsLandsat = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI'];


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

 DEFINE REGIONS OF INTEREST

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

// Merge regions of interest

var leyteROI = Forest.merge(BuiltUp).merge(Agriculture).merge(BareSoil).merge(WaterBody).merge(OtherVeg);

// Initialise random column and values for ROI feature collection

leyteROI = leyteROI.randomColumn('random1', seed);
```

```
var train = leyteROI.filter(ee.Filter.lte('random1', 0.7));
var test = leyteROI.filter(ee.Filter.gt('random1', 0.7));

Map.addLayer(train, {'color': '000000'}, 'ROI Train', true);
Map.addLayer(test, {'color': 'FF0000'}, 'ROI Test', true);
// Initialise random column and values for ROI feature collection
train = train.randomColumn('random', seed);
test = test.randomColumn('random', seed);

// Create training ROIs from the image dataset
var roiTrainLandsat = stackLandsat.select(bandsLandsat).sampleRegions({
collection: train,
properties: ['ClassID', 'random'],
scale: 30
});
// Create testing ROIs from the image dataset
var roiTestLandsat = stackLandsat.select(bandsLandsat).sampleRegions({
collection: test,
properties: ['ClassID', 'random'],
scale: 30
});

// Partition the regions of interest into training and testing areas
var trainingLandsat = roiTrainLandsat.filter(ee.Filter.lte('random', 0.7));
var testingLandsat = roiTestLandsat.filter(ee.Filter.lte('random', 0.7));

// Print number of regions of interest for training and testing at the console
//print(testingLandsat);
print(trainingLandsat);
print('Training, Landsat, n =', trainingLandsat.aggregate_count('.all'));
print('Testing, Landsat, n =', testingLandsat.aggregate_count('.all'));
```

```
/******************************
 EXECUTE CLASSIFICATION
******************************/
// LANDSAT ONLY
// Classification using Random Forest algorithm
var classifierLandsat = ee.Classifier.randomForest(100,0,10,0.5,false,2018).train({
 features: trainingLandsat.select(['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI','ClassID']),
 classProperty: 'ClassID',
 inputProperties: bandsLandsat
});

// Classify the validation data
var validationLandsat = testingLandsat.classify(classifierLandsat);
// Calculate accuracy metrics
var emL = validationLandsat.errorMatrix('ClassID', 'classification'); // Error matrix
var oaL = emL.accuracy(); // Overall accuracy
var uaL = emL.consumersAccuracy().project([1]); // Consumer's accuracy
var paL = emL.producersAccuracy().project([0]); // Producer's accuracy

print('Overall Accuracy, Landsat: ', oaL);
print('User\'s Accuracy (rows), Landsat:', uaL);
print('Producer\'s Accuracy (cols), Landsat:', paL);

// Classify the image Random Forest algorithm
var classifiedLandsat = stackLandsat.select(bandsLandsat).classify(classifierLandsat);

/******************************
 DISPLAY RGB COMPOSITES
******************************/
Map.addLayer(composite, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3}, 'RGB Landsat', false);
```

```
/*******************************
 DISPLAY CLASSIFICATION
*******************************/
// Create a palette for displaying the classified images
var palette = ['008000', 'ff0000',  '808000',  'ffff00','0000ff', '00ff00'];
// Display classified image
Map.addLayer(classifiedLandsat, {min: 0, max: 5, palette: palette}, 'Classification, Landsat', true);
// Display classified mode image
// Define classification legend
var colors = ['008000', 'ff0000',  '808000',  'ffff00','0000ff', '00ff00'];
var names = ["Forest",
 "Built-up",
 "Agriculture",
 "Bare soil",
 "Water body",
 "Other Vegetation"];
var legend = ui.Panel({style: {position: 'bottom-left'}});
legend.add(ui.Label({
 value: "Land Cover Classification",
 style: {
 fontWeight: 'bold',
 fontSize: '16px',
 margin: '0 0 4px 0',
 padding: '0px'
 }
}));
// Iterate classification legend entries
var entry;
for (var x = 0; x<6; x++){
 entry = [
 ui.Label({style:{color:colors[x],margin: '0 0 4px 0'}, value: ''}),
```

```
 ui.Label({
 value: names[x],
 style: {
 margin: '0 0 4px 4px'
 }
 })
 ];
 legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal'))); }
// Display classification legend
Map.add(legend);
/*******************************
 EXPORT CLASSIFIED IMAGES
*******************************/
// Classified images for Landsat only
Export.image.toDrive({
 image: classifiedLandsat.uint8(),
 description: 'Classification_SetA_Pre_L_30m_RF',
 folder: 'Google Earth Engine',
 region: box,
 scale: 30,
 maxPixels: 300000000,
});

/*******************************
 EXPORT TABLES
*******************************/
// Export computed statistics for all regions of interest as a csv file

// For Landsat only
Export.table.toDrive(trainingLandsat, 'Training_SetA_Pre_L_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationLandsat, 'Validation_SetA_Pre_L_30m_RF', 'Google Earth Engine');
```