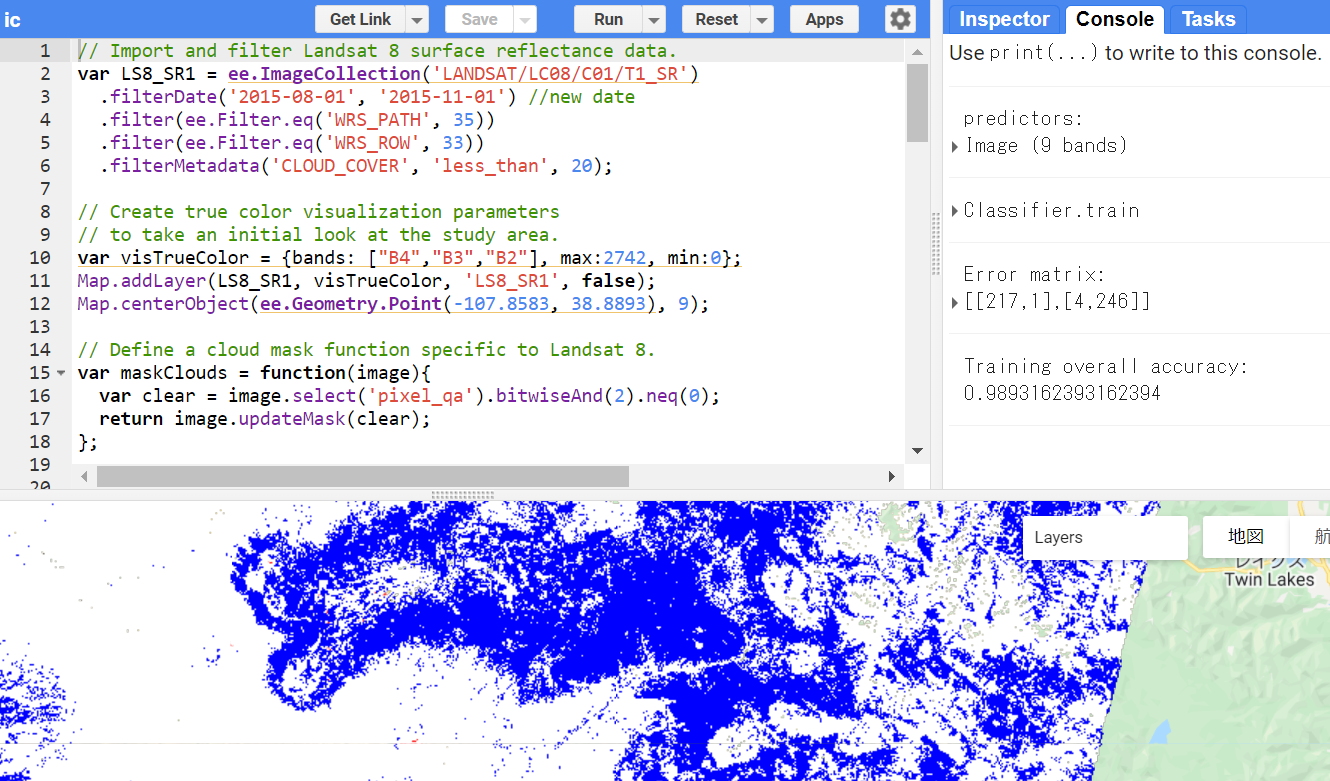
Supervised Classification

Random Forest (RF) https://ecodata.nrel.colostate.edu/gdpe-gee-remote-sensing-lessons/module7.html#google-earth-engine-image-classification-workflow



// Import and filter Landsat 8 surface reflectance data.

var LS8\_SR1 = ee.ImageCollection('LANDSAT/LC08/C01/T1\_SR')

.filterDate('2015-08-01', '2015-11-01') //new date

.filter(ee.Filter.eq('WRS\_PATH', 35))

.filter(ee.Filter.eq('WRS\_ROW', 33))

.filterMetadata('CLOUD\_COVER', 'less\_than', 20);

// Create true color visualization parameters

// to take an initial look at the study area.

var visTrueColor = {bands: ["B4","B3","B2"], max:2742, min:0};

Map.addLayer(LS8\_SR1, visTrueColor, 'LS8\_SR1', false);

Map.centerObject(ee.Geometry.Point(-107.8583, 38.8893), 9);

// Define a cloud mask function specific to Landsat 8.

var maskClouds = function(image){

var clear = image.select('pixel\_qa').bitwiseAnd(2).neq(0);

return image.updateMask(clear);

};

// Apply the cloud mask function to the previously filtered image

// collection and calculate the median.

var LS8\_SR2 = LS8\_SR1

.map(maskClouds)

.median();

Map.addLayer(LS8\_SR2, visTrueColor, 'LS8\_SR2 - masked');

// First define individual bands as variables.

var red = LS8\_SR2.select('B4').rename("red")

var green= LS8\_SR2.select('B3').rename("green")

var blue = LS8\_SR2.select('B2').rename("blue")

var nir = LS8\_SR2.select('B5').rename("nir")

var swir1 = LS8\_SR2.select('B6').rename("swir1")

var swir2 = LS8\_SR2.select('B7').rename("swir2")

// Then, calculate three different vegetation indices: NDVI, NDWI, and TCB.

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

var ndwi = green.subtract(nir).divide(green.add(nir)).rename('ndwi');

var TCB = LS8\_SR2.expression(

"0.3037 \* B2 + 0.2793 \* B3 + 0.4743 \* B4 + 0.5585 \* B5 + 0.5082 \* B6 + 0.1863 \* B7" , {

'B2': blue,

'B3': green,

'B4': red,

'B5': nir,

'B6': swir1,

'B7': swir1

}).rename("TCB");

// Combine the predictors into a single image.

var predictors = nir

.addBands(blue)

.addBands(green)

.addBands(red)

.addBands(swir1)

.addBands(swir2)

.addBands(ndvi)

.addBands(TCB)

.addBands(ndwi)

print('predictors: ', predictors);

// Load the field sampling locations.

var PA = ee.FeatureCollection('users/GDPE-GEE/Module7\_PresAbs');

Map.addLayer(PA.style({color: 'red', pointSize: 3, width: 1, fillColor: 'white'}),{}, 'Merged\_Presence\_Absence');

// Determine the values of each predictor at each training data location.

var samples = predictors.sampleRegions({

collection: PA,

properties: ['presence'],

scale: 30 });

// Using the sampled data, build a randomForest model.

// Using a specific seed (random number) exactly replicates your model each time you run it.

var trainingclassifier = ee.Classifier.smileRandomForest({

numberOfTrees: 10,

seed: 7})

.train({

features: samples,

classProperty: 'presence'});

print(trainingclassifier);

// Print Confusion Matrix and Overall Accuracy.

var confusionMatrix = trainingclassifier.confusionMatrix();

print('Error matrix: ', confusionMatrix);

print('Training overall accuracy: ', confusionMatrix.accuracy());

// Apply the model to the extent of the loaded predictor image.

var classified = predictors.classify(trainingclassifier);

Map.addLayer(classified, {min:0, max:1, palette:['white', 'blue']}, 'classified')