//draw your study area and name it “aoi” and import the DEM 30m and name it “DEM”

//License :shall be used only for education purposes

//No responsibility for otherwise used

var cloudMaskL457 = function(image) {

var qa = image.select('pixel\_qa');

// If the cloud bit (5) is set and the cloud confidence (7) is high

// or the cloud shadow bit is set (3), then it's a bad pixel.

var cloud = qa.bitwiseAnd(1 << 5)

.and(qa.bitwiseAnd(1 << 7))

.or(qa.bitwiseAnd(1 << 3));

// Remove edge pixels that don't occur in all bands

var mask2 = image.mask().reduce(ee.Reducer.min());

return image.updateMask(cloud.not()).updateMask(mask2);

};

//area of interest

var aoi1 = aoi

// load the landsat data

var composite = ee.ImageCollection('LANDSAT/LT05/C01/T1\_SR')

.filterDate('1994-06-01', '1995-09-30').filterBounds(aoi1)

.map(cloudMaskL457);

var visParams = {

bands: ['B3', 'B2', 'B1'],

min: 0,

max: 3000,

gamma: 1.4,

};

Map.addLayer(composite.median().clip(aoi1), visParams);

//===================================================================================

var image=composite.median().select(['B1','B2','B3','B4','B5','B6','B7']).clip(aoi1).int()

//new indext est===========================================================================0

var NDBI = image.normalizedDifference(['B5','B4']).rename('NDBI');

var NDBai = image.normalizedDifference(['B5','B6']).rename('NDBai')

var newIndex= NDBI.addBands(NDBai)

//GLCM===================================================================================000

var glcm\_image1 = image.select('B1')

var glcm\_image2 = image.select('B2')

var glcm\_image3 = image.select('B3')

var glcm\_image4 = image.select('B4')

var glcm\_image5 = image.select('B5')

var glcm\_image6 = image.select('B6')

var glcm\_image7 = image.select('B7')

var glcm1 = glcm\_image1.glcmTexture({size: 7}).select(['B1\_contrast','B1\_diss'])

var glcm2 = glcm\_image2.glcmTexture({size: 7})//.select(['B2\_var','B2\_contrast','B2\_diss'])

var glcm3 = glcm\_image3.glcmTexture({size: 7})//.select(['B3\_var','B3\_contrast','B3\_diss'])

var glcm4 = glcm\_image4.glcmTexture({size: 7})//.select(['B4\_var','B4\_contrast','B4\_diss'])

var glcm5 = glcm\_image5.glcmTexture({size: 7})//.select(['B5\_var','B5\_contrast','B5\_diss'])

var glcm6 = glcm\_image6.glcmTexture({size: 7})//.select(['B6\_var','B6\_contrast','B6\_diss'])

var glcm7 = glcm\_image7.glcmTexture({size: 7})//.select(['B7\_var','B7\_contrast' ,'B7\_diss'])

var glcmfin =glcm1//.addBands(glcm2).addBands(glcm3).addBands(glcm4).addBands(glcm5).addBands(glcm6).addBands(glcm7).int()

//============================================================================================

//DEM

var Maskeddata= glcmfin.addBands(newIndex).addBands(image)

print(Maskeddata)

//============================================================================================

var training =T.filterBounds(aoi1)

//print (training.reduceColumns(ee.Reducer.frequencyHistogram(),['remapped']),'Training');

var band = Maskeddata.bandNames()

//sample the training data

var training\_samp = Maskeddata.select(band).sampleRegions({

collection: training,

properties: ['remapped'],

scale: 30

});

// Make a Random Forest classifier and train it.

var classifier = ee.Classifier.smileRandomForest({numberOfTrees:50,variablesPerSplit:7}).train({

features: training\_samp,

classProperty: 'remapped',

inputProperties: band

});

//====================================================================== variable importance

//Run the classification 3

var classified = Maskeddata.select(band).classify(classifier);

var dict = classifier.explain();

print('Explain:',dict);

var variable\_importance = ee.Feature(null, ee.Dictionary(dict).get('importance'));

//print(variable\_importance,'varr')

var chart =

ui.Chart.feature.byProperty(variable\_importance)

.setChartType('ColumnChart')

.setOptions({

title: 'Random Forest Variable Importance',

legend: {position: 'none'},

hAxis: {title: 'Bands'},

vAxis: {title: 'Importance'}

});

print(chart);

// Get a confusion matrix representing resubstitution accuracy.

print('RF error matrix: ', classifier.confusionMatrix());

print('RF accuracy: ', classifier.confusionMatrix().accuracy());

// export the classification result

Export.image.toDrive({

image: classified,

description: 'classifiedNew',

scale: 30,

maxPixels:1e13,

region: aoi1

});