

Supplementary Materials File S1

Code to run the classification in Google Earth Engine

```
////*****Faisalabad-Year 1989*****  
////MASK CLOUD LANDSAT////  
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);  
};  
////LOAD LANDSAT DATA////  
var bands= ['B1', 'B2', 'B3', 'B4', 'B5', 'B7'];  
var dataset1 = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR')  
  .filterDate('1989-01-01', '1989-06-30')  
  .map(cloudMaskL457).select(bands);  
var visParams1 = {  
  bands: ['B3', 'B2', 'B1'],  
  min: 0,  
  max: 3000,  
  gamma: 1.3,  
};  
var median1989 = dataset1.median();  
var clipped1989 = median1989.clip(FSD);  
Map.addLayer(clipped1989, visParams1, '1989');  
var ndvi = clipped1989.normalizedDifference(['B4', 'B3']);  
var ndwi = clipped1989.normalizedDifference(['B2', 'B4']);  
var mndwi = clipped1989.normalizedDifference(['B2', 'B5']);  
var ndbi = clipped1989.normalizedDifference(['B7', 'B4']);  
////NDVI Layer////  
var ndvi = clipped1989.normalizedDifference(['B4', 'B3']);  
var ndviPalette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',  
  '74A901', '66A000', '529400', '3E8601', '207401', '056201',  
  '004C00', '023B01', '012E01', '011D01', '011301'];  
Map.addLayer(ndvi,  
  {min:0, max: 1, palette: ndviPalette}, 'NDVI', false);  
////  
//Map.addLayer(ndvi);  
//Map.addLayer(ndwi);  
//Map.addLayer(mndwi);  
//Map.addLayer(ndbi);  
////TRAIN CLASSIFICATION////  
var trainingImage = ee.Image([clipped1989.select('B1'),  
  clipped1989.select('B2'),  
  clipped1989.select('B3'),  
  clipped1989.select('B4'),  
  clipped1989.select('B5'),  
  clipped1989.select('B7');  
var trainingFeatures = buildup  
  .merge(water)  
  .merge(barren)  
  .merge(agriculture)  
  .merge(tree)  
  .merge(grass);
```

```

print(trainingImage);
var predictionBands = trainingImage.bandNames();
print(predictionBands);
var classifierTraining = trainingImage
    .select(predictionBands)
    .sampleRegions({collection: trainingFeatures,
        properties: ['landuse'],
        scale: 30
    });
/////RUN CLASSIFICATION/////
var classifier = ee.Classifier.smileCart().train({
    features: classifierTraining,
    classProperty: 'landuse',
    inputProperties: predictionBands
});
print(classifier.explain());
var classified = trainingImage.select(predictionBands).classify(classifier);
var vizParams = {'bands': ['B3','B2','B1'],
    'min': 0,
    'max': 0.3,
    'gamma': [1.4,1.2,1.0]};
Map.addLayer(trainingImage,vizParams,'True-color composite',false);
Map.addLayer(trainingImage,{'bands':['ndbi','ndvi','mndwi'],
    'min':0,'max':0.75},'NDXI composite',false);
var palette = [
    'ff3c1d', // (1)
    '215c8f', // (2)
    'f2f3e9', // (3)
    'ffc82d', // (4)
    '008800', // (5)
    '00ff2d', // (6)
];
var lulcVis = {min:1,max:6,palette:palette};
Map.addLayer(classified.clip(FSD), lulcVis,'classified');
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix,accuracy);
var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 – commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());

// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 – omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());

// Calculate kappa statistic.
print('Kappa statistic', confusionMatrix.kappa());
/////Export/////
Export.image.toDrive({

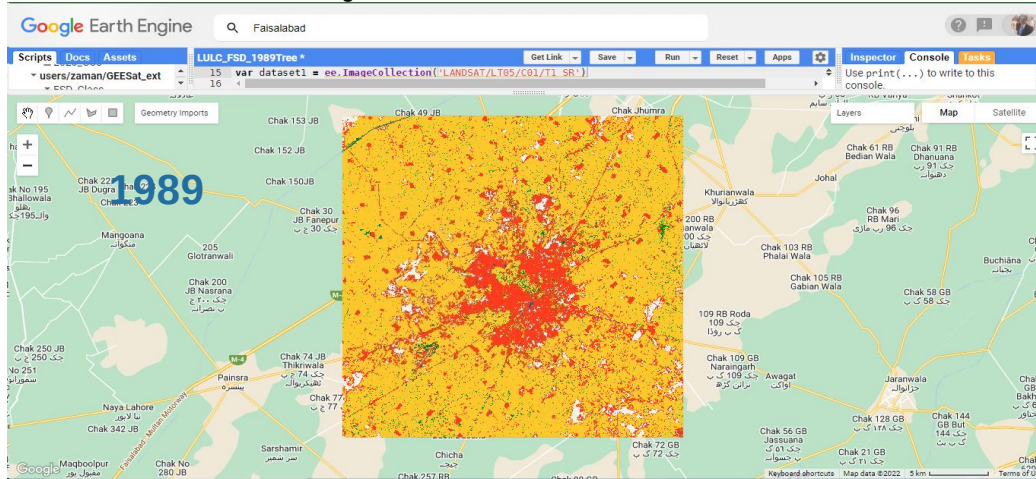
```

```

image: classified,
description: 'FSD_FF_1989',
scale: 30,
region: FSD
});

```

Screenshot of the classified image



*****Faisalabad-Year 1989*****

```

//////MASK CLOUD LANDSAT//////
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);
};

//////LOAD LANDSAT DATA//////
var bands= ['B1', 'B2', 'B3', 'B4', 'B5', 'B7'];
var dataset1 = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR')
  .filterDate('1999-01-01', '1999-06-30')
  .map(cloudMaskL457).select(bands);

var visParams1 = {
  bands: ['B7', 'B4', 'B2'],
  min: 0,
  max: 3000,
  gamma: 1.3,
};

////// Landcover Classes Selection overlay Band combination//////
var median1999 = dataset1.median();
var clipped1999 = median1999.clip(FSD);
Map.addLayer(clipped1999, visParams1, '1999');
var ndvi = clipped1999.normalizedDifference(['B4', 'B3']);
var ndwi = clipped1999.normalizedDifference(['B2', 'B4']);
var mndwi = clipped1999.normalizedDifference(['B2', 'B5']);
var ndbi = clipped1999.normalizedDifference(['B7', 'B4']);
//Map.addLayer(ndvi);
//Map.addLayer(ndwi);
//Map.addLayer(mndwi);
//Map.addLayer(ndbi);
//////////NDVI Layer//////////
var ndvi = clipped1989.normalizedDifference(['B4', 'B3']);

```

```

var ndviPalette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
    '74A901', '66A000', '529400', '3E8601', '207401', '056201',
    '004C00', '023B01', '012E01', '011D01', '011301'];
Map.addLayer(ndvi,
    {min:0, max: 1, palette: ndviPalette}, 'NDVI', false);
/////TRAIN CLASSIFICATION/////
var trainingImage = ee.Image([clipped1999.select('B1'),
    clipped1999.select('B2'),
    clipped1999.select('B3'),
    clipped1999.select('B4'),
    clipped1999.select('B5'),
    clipped1999.select('B7')];
var trainingFeatures = buildup
    .merge(water)
    .merge(barren)
    .merge(agriculture)
    .merge(tree)
    .merge(grass);
print(trainingImage);
var predictionBands = trainingImage.bandNames();
print(predictionBands);
var classifierTraining = trainingImage
    .select(predictionBands)
    .sampleRegions({collection: trainingFeatures,
        properties: ['landuse'],
        scale: 30
    });
/////RUN CLASSIFICATION/////
var classifier = ee.Classifier.smileCart().train({
    features: classifierTraining,
    classProperty: 'landuse',
    inputProperties: predictionBands
});
print(classifier.explain());
var classified = trainingImage.select(predictionBands).classify(classifier);
var vizParams = {'bands': ['B3','B2','B1'],
    'min': 0,
    'max': 0.3,
    'gamma': [1.4,1.2,1.0]};
Map.addLayer(trainingImage,vizParams,'True-color composite',false);
Map.addLayer(trainingImage,{'bands':['ndbi','ndvi','mndwi'],
    'min':0,'max':0.75},'NDXI composite',false);
var palette = [
    'ff3c1d',
    '215c8f',
    'f2f3e9',
    'ffc82d',
    '008800',
    '00ff2d',
];
var lulcVis = {min:1,max:6,palette:palette};
Map.addLayer(classified.clip(FSD), lulcVis,'classified');
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix,accuracy);

```

```

var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

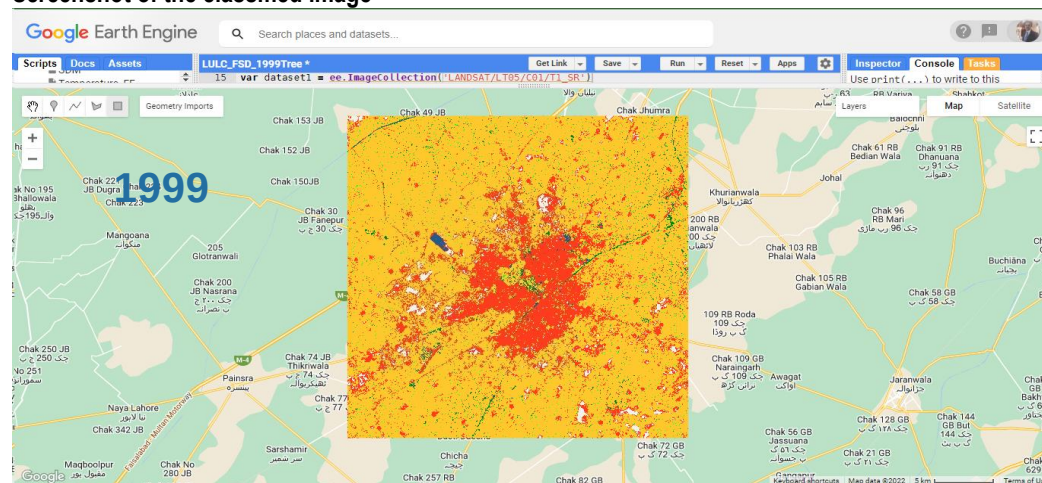
// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 - commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());

// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 - omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());

// Calculate kappa statistic.
print("Kappa statistic", confusionMatrix.kappa());
/////Export/////
Export.image.toDrive({
  image: classified,
  description: 'FSD_Final_1999',
  scale: 30,
  region: FSD
});

```

Screenshot of the classified image



*****Faisalabad Year 2009*****

```

/////MASK CLOUD LANDSAT/////
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);
};

/////LOAD LANDSAT DATA/////
var bands= ['B1', 'B2', 'B3', 'B4', 'B5', 'B7'];
var dataset1 = ee.ImageCollection('LANDSAT/LE07/C01/T1_SR')
  .filterDate('2009-01-01', '2009-06-30')
  .map(cloudMaskL457).select(bands);

var visParams1 = {
  bands: ['B3', 'B4', 'B5'],
  min: 0,

```

```

    max: 3000,
    gamma: 1.3,
  };
  var median2009 = dataset1.median();
  var clipped2009 = median2009.clip(FSD);
  Map.addLayer(clipped2009, visParams1, '2009');
  var trueColor321 = clipped2009.select(['B3', 'B4', 'B5']);
  var trueColor321Vis = {
    min: 0,
    max: 2500,
    gamma: [0.95, 1.1, 1],
  };
  var trainingImage = ee.Image([clipped2009.select('B1'),
    clipped2009.select('B2'),
    clipped2009.select('B3'),
    clipped2009.select('B4'),
    clipped2009.select('B5'),
    clipped2009.select('B7'),
  ]);
  var trainingFeatures = buildup
    .merge(water)
    .merge(barren)
    .merge(agriculture)
    .merge(tree)
    .merge(grass);
  print(trainingImage);
  var predictionBands = trainingImage.bandNames();
  print(predictionBands);
  var classifierTraining = trainingImage
    .select(predictionBands)
    .sampleRegions({collection: trainingFeatures,
      properties: ['landuse'],
      scale: 30
    });
  /////RUN CLASSIFICATION/////
  var classifier = ee.Classifier.smileCart().train({
    features: classifierTraining,
    classProperty: 'landuse',
    inputProperties: predictionBands
  });
  print(classifier.explain());
  var classified = trainingImage.select(predictionBands).classify(classifier);
  //var visParams =
  {"opacity":1,"bands":["B4","B3","B2"],"min":0.06860050559043884,"max":0.4413536787033081,"gamma":1};
  var vizParams = {'bands': ['B3','B2','B1'],
    'min': 0.06860050559043884,
    'max': 0.4413536787033081,
    'gamma': [1.4, 1.2, 1.0]};
  Map.addLayer(trainingImage, vizParams, 'True-color composite', false);
  var palette = [
    'ff3c1d',
    '215c8f',
    'f2f3e9',
    'ffc82d',
    '008800',
    '00ff2d',
  ];
  var lulcVis = {min:1,max:6,palette:palette};
  Map.addLayer(classified.clip(FSD), lulcVis, 'classified');
  ////////////Accuracy////////

```

Screenshot of the classified image



```

var visParams1 = {
  bands: ['B7', 'B4', 'B2'],
  // min: 0,
  // max: 1000,
  gamma: 1.8,
};
var medianI8toa = I8toa.median();
var clippedI8toa = medianI8toa.clip(FSD);
Map.addLayer(clippedI8toa, visParams1, '2019');
var ndvi = clippedI8toa.normalizedDifference(['B5', 'B4']);
var ndwi = clippedI8toa.normalizedDifference(['B3', 'B5']);
var mndwi = clippedI8toa.normalizedDifference(['B5', 'B4']);
var mndwi37 = clippedI8toa.normalizedDifference(['B3', 'B7']);
var ndmi = clippedI8toa.normalizedDifference(['B5', 'B6']);
var uinx = clippedI8toa.normalizedDifference(['B7', 'B4']);
var trainingImage = ee.Image([clippedI8toa.select('B2'),
  clippedI8toa.select('B3'),
  clippedI8toa.select('B4'),
  clippedI8toa.select('B5'),
  clippedI8toa.select('B6'),
  clippedI8toa.select('B7'),
  clippedI8toa.select('B10'),
  clippedI8toa.select('B11'),
]);

var vizParams = {'bands': ['B5', 'B4', 'B3'],
  'min': 0,
  'max': 0.3,
  'gamma': [1.4, 1.2, 1.0]};
Map.addLayer(trainingImage, vizParams, 'True-color composite', false);
Map.addLayer(trainingImage, {'bands': ['ndwi', 'ndvi', 'mndwi'],
  'min': 0, 'max': 0.75, 'NDXI composite', false});
*/
//////////NDWI Layer//////////
var ndwi = clippedI8toa.normalizedDifference(['B3', 'B5']);
var ndwiViz = {min: -0.7, max: 1, palette: ['brown', 'black', '0000FF', '00FFFF'], bands: ['nd']};
Map.addLayer(ndwi, ndwiViz, 'NDWI', false);

var trainingFeatures = buildup
  .merge(water)
  .merge(barren)
  .merge(agriculture)
  .merge(tree)
  .merge(grass);
print(trainingImage);
var predictionBands = trainingImage.bandNames();
print(predictionBands);
var classifierTraining = trainingImage
  .select(predictionBands)
  .sampleRegions({collection: trainingFeatures,
    properties: ['landuse'],
    scale: 30
  });
/////////RUN CLASSIFICATION/////////
var classifier = ee.Classifier.smileCart().train({
  features: classifierTraining,
  classProperty: 'landuse',
  inputProperties: predictionBands
});
print(classifier.explain());

```

```

var classified = trainingImage.select(predictionBands).classify(classifier);
var vizParams = {'bands': ['B3','B2','B1'],
  'min': 0,
  'max': 0.3,
  'gamma': [1.4,1.2,1.0]};
//Map.addLayer(trainingImage,vizParams,'True-color composite',false);
//Map.addLayer(trainingImage,{bands:['ndbi','ndvi','mndwi'],
//  'min':0,'max':0.75},'NDXI composite',false);
var palette = [
  'ff3c1d',
  '215c8f',
  'f2f3e9',
  'ffc82d',
  '008800',
  '00ff2d',
];
var lulcVis = {min:1,max:6,palette:palette};
Map.addLayer(classified.clip(FSD), lulcVis,'classified');
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix,accuracy);
var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

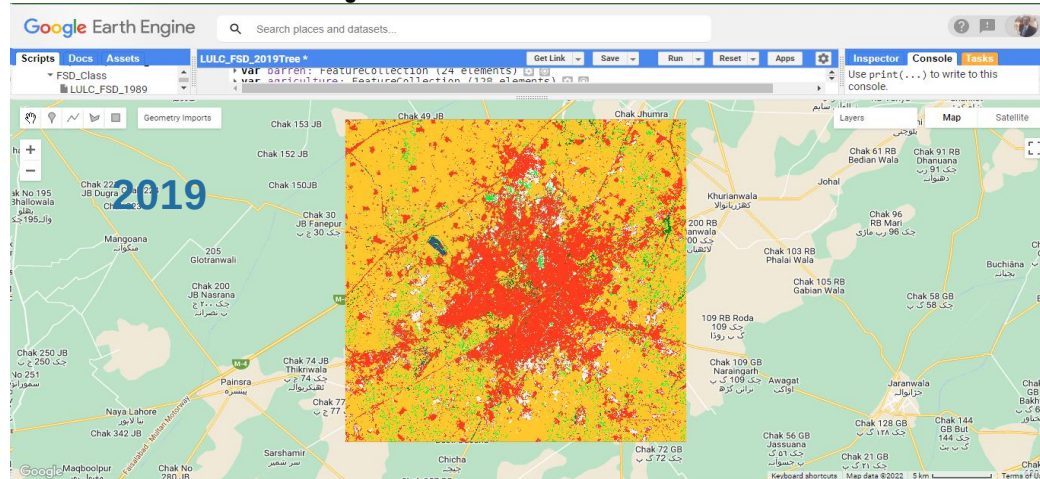
// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 – commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());

// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 – omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());

// Calculate kappa statistic.
print('Kappa statistic', confusionMatrix.kappa());
//////////Export////////
Export.image.toDrive({
  image: classified,
  description: 'FSD_FF_2019',
  scale: 30,
  region: FSD
});

```

Screenshot of the classified image



////*****Jhang-Year 1989*****

//////MASK CLOUD LANDSAT//////

```
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);  
};
```

//////LOAD LANDSAT DATA//////

```
var bands= ['B1', 'B2', 'B3', 'B4', 'B5', 'B7'];  
var dataset1 = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR')  
  .filterDate('1989-01-01', '1989-06-30')  
  .map(cloudMaskL457).select(bands);  
var visParams1 = {  
  bands: ['B3', 'B2', 'B1'],  
  min: 0,  
  max: 3000,  
  gamma: 1.3,  
};  
var median1989 = dataset1.median();  
var clipped1989 = median1989.clip(JHG);  
Map.addLayer(clipped1989, visParams1, '1989');  
var ndvi = clipped1989.normalizedDifference(['B4', 'B3']);  
var ndwi = clipped1989.normalizedDifference(['B2', 'B4']);  
var mndwi = clipped1989.normalizedDifference(['B2', 'B5']);  
var ndbi = clipped1989.normalizedDifference(['B7', 'B4']);  
//////NDVI Layer//////  
var ndvi = clipped1989.normalizedDifference(['B4', 'B3']);  
var ndviPalette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',  
  '74A901', '66A000', '529400', '3E8601', '207401', '056201',  
  '004C00', '023B01', '012E01', '011D01', '011301'];
```

```
Map.addLayer(ndvi,  
  {min:0, max: 1, palette: ndviPalette, 'NDVI', false};
```

//////

//Map.addLayer(ndvi);

//Map.addLayer(ndwi);

//Map.addLayer(mndwi);

//Map.addLayer(ndbi);

//////TRAIN CLASSIFICATION//////

```

var trainingImage = ee.Image([clipped1989.select('B1'),
                              clipped1989.select('B2'),
                              clipped1989.select('B3'),
                              clipped1989.select('B4'),
                              clipped1989.select('B5'),
                              clipped1989.select('B7')];
var trainingFeatures = builtup
  .merge(water)
  .merge(barren)
  .merge(agriculture)
  .merge(tree)
  .merge(grass);
print(trainingImage);
var predictionBands = trainingImage.bandNames();
print(predictionBands);
var classifierTraining = trainingImage
  .select(predictionBands)
  .sampleRegions({collection: trainingFeatures,
                  properties: ['landuse'],
                  scale: 30
                  });
/////RUN CLASSIFICATION/////
var classifier = ee.Classifier.smileCart().train({
  features: classifierTraining,
  classProperty: 'landuse',
  inputProperties: predictionBands
});
print(classifier.explain());
var classified = trainingImage.select(predictionBands).classify(classifier);
var vizParams = {'bands': ['B3', 'B2', 'B1'],
                 'min': 0,
                 'max': 0.3,
                 'gamma': [1.4, 1.2, 1.0]};
Map.addLayer(trainingImage, vizParams, 'True-color composite', false);
Map.addLayer(trainingImage, {'bands': ['ndbi', 'ndvi', 'mndwi'],
                             'min': 0, 'max': 0.75, 'NDXI composite', false});

var palette = [
  'ff3c1d', // (1)
  '215c8f', // (2)
  'f2f3e9', // (3)
  'ffc82d', // (4)
  '008800', // (5)
  '00ff2d', // (6)
];
var lulcVis = {min: 1, max: 6, palette: palette};
Map.addLayer(classified.clip(JHG), lulcVis, 'classified');
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix, accuracy);
var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

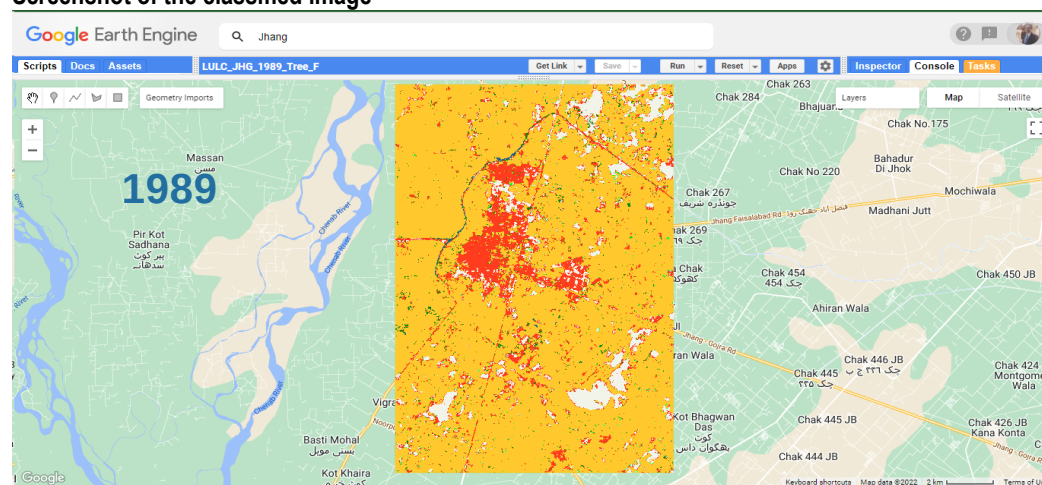
```

```
// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 – commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());
```

```
// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 – omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());
```

```
// Calculate kappa statistic.
print('Kappa statistic', confusionMatrix.kappa());
/////Export/////
Export.image.toDrive({
  image: classified,
  description: 'JHG_FF_1989',
  scale: 30,
  region: JHG
});
```

Screenshot of the classified image



*****Jhang-Year 1999*****

```
//////MASK CLOUD LANDSAT//////
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);
};
//////LOAD LANDSAT DATA//////
var bands= ['B1', 'B2', 'B3', 'B4', 'B5', 'B7'];
var dataset1 = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR')
  .filterDate('1999-01-01', '1999-06-30')
  .map(cloudMaskL457).select(bands);
var visParams1 = {
  bands: ['B7', 'B4', 'B2'],
  min: 0,
  max: 3000,
  gamma: 1.3,
};
////// Landcover Classes Selection overlay Band combination//////
var median1999 = dataset1.median();
var clipped1999 = median1999.clip(JHG);
```

```

Map.addLayer(clipped1999, visParams1, '1999');
var ndvi = clipped1999.normalizedDifference(['B4', 'B3']);
var ndwi = clipped1999.normalizedDifference(['B2', 'B4']);
var mndwi = clipped1999.normalizedDifference(['B2', 'B5']);
var ndbi = clipped1999.normalizedDifference(['B7', 'B4']);
//Map.addLayer(ndvi);
//Map.addLayer(ndwi);
//Map.addLayer(mndwi);
//Map.addLayer(ndbi);
//////////NDVI Layer//////////
var ndvi = clipped1989.normalizedDifference(['B4', 'B3']);
var ndviPalette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
'74A901', '66A000', '529400', '3E8601', '207401', '056201',
'004C00', '023B01', '012E01', '011D01', '011301'];
Map.addLayer(ndvi,
{min:0, max: 1, palette: ndviPalette}, 'NDVI', false);

//////TRAIN CLASSIFICATION//////
var trainingImage = ee.Image([clipped1999.select('B1'),
clipped1999.select('B2'),
clipped1999.select('B3'),
clipped1999.select('B4'),
clipped1999.select('B5'),
clipped1999.select('B7')];
var trainingFeatures = builtup
.merge(water)
.merge(barren)
.merge(agriculture)
.merge(tree)
.merge(grass);
print(trainingImage);
var predictionBands = trainingImage.bandNames();
print(predictionBands);
var classifierTraining = trainingImage
.select(predictionBands)
.sampleRegions({collection: trainingFeatures,
properties: ['landuse'],
scale: 30
});
//////RUN CLASSIFICATION//////
var classifier = ee.Classifier.smileCart().train({
features: classifierTraining,
classProperty: 'landuse',
inputProperties: predictionBands
});
print(classifier.explain());
var classified = trainingImage.select(predictionBands).classify(classifier);
var vizParams = {'bands': ['B3', 'B2', 'B1'],
'min': 0,
'max': 0.3,
'gamma': [1.4, 1.2, 1.0]};
Map.addLayer(trainingImage, vizParams, 'True-color composite', false);
Map.addLayer(trainingImage, {'bands': ['ndbi', 'ndvi', 'mndwi'],
'min': 0, 'max': 0.75}, 'NDXI composite', false);
var palette = [
'ff3c1d',
'215c8f',
'f2f3e9',
'ffc82d',
'008800',

```

```

'00ff2d',
];
var lulcVis = {min:1,max:6,palette:palette};
Map.addLayer(classified.clip(JHG), lulcVis,'classified');
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix,accuracy);
var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

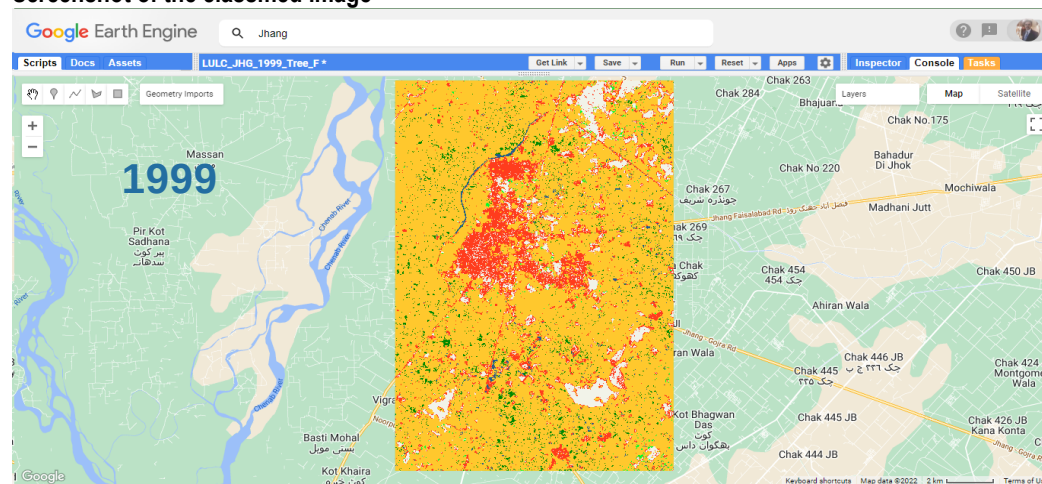
// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 - commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());

// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 - omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());

// Calculate kappa statistic.
print("Kappa statistic", confusionMatrix.kappa());
////////Export////////
Export.image.toDrive({
  image: classified,
  description: 'JHG_Final_1999',
  scale: 30,
  region: JHG
});

```

Screenshot of the classified image



*****Jhang-Year 2009*****

```

////////MASK CLOUD LANDSAT////////
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);
};
/////LOAD LANDSAT DATA/////
var bands= ['B1', 'B2', 'B3', 'B4', 'B5', 'B7'];
var dataset1 = ee.ImageCollection('LANDSAT/LE07/C01/T1_SR')
    .filterDate('2009-01-01', '2009-06-30')
    .map(cloudMaskL457).select(bands);
var visParams1 = {
  bands: ['B3', 'B4', 'B5'],
  min: 0,
  max: 3000,
  gamma: 1.3,
};
var median2009 = dataset1.median();
var clipped2009 = median2009.clip(JHG);
Map.addLayer(clipped2009, visParams1, '2009');
var trueColor321 = clipped2009.select(['B3', 'B4', 'B5']);
var trueColor321Vis = {
  min: 0,
  max: 2500,
  gamma: [0.95, 1.1, 1],
};
var trainingImage = ee.Image([clipped2009.select('B1'),
  clipped2009.select('B2'),
  clipped2009.select('B3'),
  clipped2009.select('B4'),
  clipped2009.select('B5'),
  clipped2009.select('B7'),
]);
var trainingFeatures = builtup
    .merge(water)
    .merge(barren)
    .merge(agriculture)
    .merge(tree)
    .merge(grass)
print(trainingImage);
var predictionBands = trainingImage.bandNames();
print(predictionBands);
var classifierTraining = trainingImage
    .select(predictionBands)
    .sampleRegions({collection: trainingFeatures,
  properties: ['landuse'],
  scale: 30
});
/////RUN CLASSIFICATION/////
var classifier = ee.Classifier.smileCart().train({
  features: classifierTraining,
  classProperty: 'landuse',
  inputProperties: predictionBands
});
print(classifier.explain());
var classified = trainingImage.select(predictionBands).classify(classifier);
//var visParams =
{"opacity":1,"bands":["B4","B3","B2"],"min":0.06860050559043884,"max":0.4413536787033081,"gamma":1};
var vizParams = {'bands': ['B3','B2','B1'],
  'min': 0.06860050559043884,
  'max': 0.4413536787033081,
  'gamma': [1.4,1.2,1.0]};

```

```

Map.addLayer(trainingImage,vizParams,'True-color composite',false);
var palette = [
  'ff3c1d',
  '215c8f',
  'f2f3e9',
  'ffc82d',
  '008800',
  '00ff2d',
];
var lulcVis = {min:1,max:6,palette:palette};
Map.addLayer(classified.clip(JHG), lulcVis,'classified');
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix,accuracy);
var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

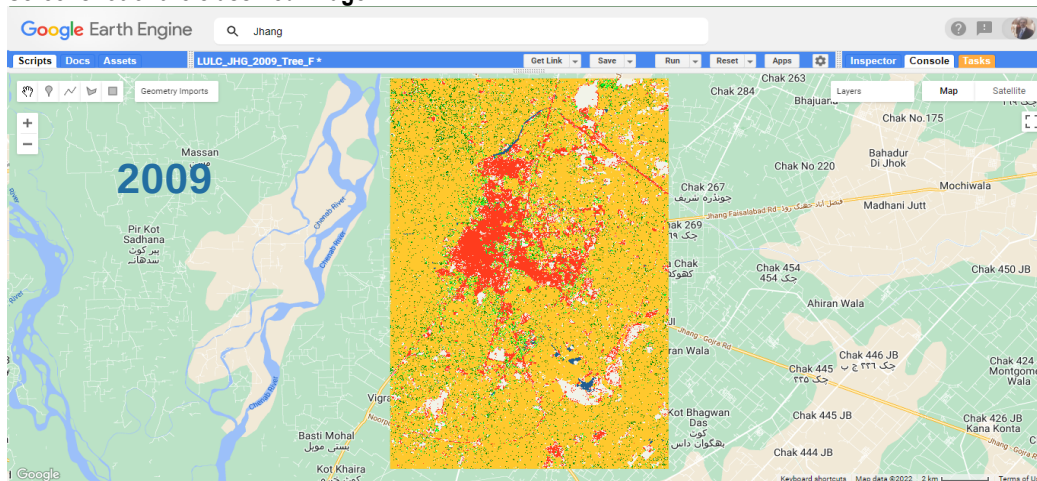
// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 – commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());

// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 – omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());

// Calculate kappa statistic.
print('Kappa statistic', confusionMatrix.kappa());
////////Export////////
Export.image.toDrive({
  image: classified,
  description: 'JHG_FF_2009',
  scale: 30,
  region: JHG
});

```

Screenshot of the classified image



*****Jhang-Year 2019*****

```
//////MASK CLOUD LANDSAT//////
var maskClouds = function(image) {
  var quality = image.select('BQA');
  var cloud01 = quality.eq(61440);
  var cloud02 = quality.eq(53248);
  var cloud03 = quality.eq(28672);
  var mask = cloud01.or(cloud02).or(cloud03).not();
  return image.updateMask(mask);
};
//////LOAD LANDSAT DATA//////
var bands = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'B11'];
var l8toa = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')
  .filterDate('2019-01-01', '2019-06-30').select(bands).map(maskClouds);
var visParams1 = {
  bands: ['B7', 'B4', 'B2'],
  // min: 0,
  // max: 1000,
  gamma: 1.8,
};
var medianl8toa = l8toa.median();
var clippedl8toa = medianl8toa.clip(JHG);
Map.addLayer(clippedl8toa, visParams1, '2019');
var ndvi = clippedl8toa.normalizedDifference(['B5', 'B4']);
var ndwi = clippedl8toa.normalizedDifference(['B3', 'B5']);
var mndwi = clippedl8toa.normalizedDifference(['B5', 'B4']);
var mndwi37 = clippedl8toa.normalizedDifference(['B3', 'B7']);
var ndmi = clippedl8toa.normalizedDifference(['B5', 'B6']);
var uinx = clippedl8toa.normalizedDifference(['B7', 'B4']);
var trainingImage = ee.Image([clippedl8toa.select('B2'),
  clippedl8toa.select('B3'),
  clippedl8toa.select('B4'),
  clippedl8toa.select('B5'),
  clippedl8toa.select('B6'),
  clippedl8toa.select('B7'),
  clippedl8toa.select('B10'),
  clippedl8toa.select('B11'),
]);

var vizParams = {'bands': ['B5', 'B4', 'B3'],
  'min': 0,
  'max': 0.3,
  'gamma': [1.4, 1.2, 1.0]};
Map.addLayer(trainingImage, vizParams, 'True-color composite', false);
Map.addLayer(trainingImage, {'bands': ['ndwi', 'ndvi', 'mndwi'],
  'min': 0, 'max': 0.75}, 'NDXI composite', false);
*/
//////////NDWI Layer//////////
var ndwi = clippedl8toa.normalizedDifference(['B3', 'B5']);
var ndwiViz = {min: -0.7, max: 1, palette: ['brown', 'black', '0000FF', '00FFFF'], bands: ['nd']};
Map.addLayer(ndwi, ndwiViz, 'NDWI', false);

var trainingFeatures = buildup
  .merge(water)
  .merge(barren)
  .merge(agriculture)
  .merge(tree)
  .merge(grass);
print(trainingImage);
var predictionBands = trainingImage.bandNames();
```

```

print(predictionBands);
var classifierTraining = trainingImage
  .select(predictionBands)
  .sampleRegions({collection: trainingFeatures,
    properties: ['landuse'],
    scale: 30
  });
/////RUN CLASSIFICATION/////
var classifier = ee.Classifier.smileCart().train({
  features: classifierTraining,
  classProperty: 'landuse',
  inputProperties: predictionBands
});
print(classifier.explain());
var classified = trainingImage.select(predictionBands).classify(classifier);
var vizParams = {'bands': ['B3','B2','B1'],
  'min': 0,
  'max': 0.3,
  'gamma': [1.4,1.2,1.0]};
//Map.addLayer(trainingImage,vizParams,'True-color composite',false);
//Map.addLayer(trainingImage,{'bands':['ndbi','ndvi','mndwi'],
//  'min':0,'max':0.75},'NDXI composite',false);
var palette = [
  'ff3c1d',
  '215c8f',
  'f2f3e9',
  'ffc82d',
  '008800',
  '00ff2d',
];
var lulcVis = {min:1,max:6,palette:palette};
Map.addLayer(classified.clip(JHG), lulcVis,'classified');
var confusionMatrix = classifier.confusionMatrix();
//////////Accuracy////////
sample = sample.randomColumn();
var split = 0.8; // Roughly 80% training, 20% testing.
var training = sample.filter(ee.Filter.lt('random', split));
var validation = sample.filter(ee.Filter.gte('random', split));
var confusionMatrix = classifier.confusionMatrix()
var accuracy = confusionMatrix.accuracy()
print(confusionMatrix,accuracy);
var confusionMatrix = ee.ConfusionMatrix(array);
print("Constructed confusion matrix", confusionMatrix);

// Calculate overall accuracy.
print("Overall accuracy", confusionMatrix.accuracy());

// Calculate consumer's accuracy, also known as user's accuracy or
// specificity and the complement of commission error (1 – commission error).
print("Consumer's accuracy", confusionMatrix.consumersAccuracy());

// Calculate producer's accuracy, also known as sensitivity and the
// compliment of omission error (1 – omission error).
print("Producer's accuracy", confusionMatrix.producersAccuracy());

// Calculate kappa statistic.
print('Kappa statistic', confusionMatrix.kappa());
/////Export/////
Export.image.toDrive({
  image: classified,

```

```
description: 'JHG_FF_2019',  
scale: 30,  
region: JHG  
});
```

