SVM with Sentinel 1 & Planetscope

```
var wetland = ee.FeatureCollection("projects/ee-sevenwah970705/assets/wetland"),
      nonwetland = ee.FeatureCollection("projects/ee-sevenwah970705/assets/nonwetland"),
      image = ee.Image("projects/ee-sevenwah970705/assets/Ohio_Wetland_Planet"),
      water = ee.FeatureCollection("projects/ee-sevenwah970705/assets/water");
      sar = ee.Image("projects/ee-sevenwah970705/assets/Ohio_Wetland_S1");

// Add VV and VH bands
image = image.addBands(sar.select('VV').rename('VV'));
image = image.addBands(sar.select('VH').rename('VH'));

// Calculate NDVI and NDWI
var ndvi = image.normalizedDifference(['b4', 'b3']);
var ndwi = image.normalizedDifference(['b2', 'b4']);

// Add NDVI and NDWI bands
image = image.addBands(ndvi.rename('NDVI'));
image = image.addBands(ndwi.rename('NDWI'));

var landcoverPalette = ['yellow', 'green', 'blue'];
print(image);

// Add true-color composite to map
Map.addLayer(image, {bands: ['b4', 'b3', 'b2'], max: 3000}, 'True color image');

var classNames = water.merge(nonwetland).merge(wetland);
print(classNames);

var bands = ['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'VV', 'VH', 'NDVI', 'NDWI'];

var training = image.select(bands).sampleRegions({
  collection: classNames,
  properties: ['ClassID'],
  scale: 30,
  tileScale: 12
});

var withRandom = training.randomColumn('random');
var split = 0.7;
var trainingPartition = withRandom.filter(ee.Filter.lt('random', split));
var testingPartition = withRandom.filter(ee.Filter.gte('random', split));

// Set SVM parameters
var svmOptions = {
  kernelType: 'RBF', // Change the kernel type (options: 'linear', 'poly', 'rbf', 'sigmoid',
```

```
'precomputed')
  gamma: 0.1
  cost: 10
};

var classifier = ee.Classifier.libsvm(svmOptions)

//classification method: smileCart() randomForest() minimumDistance libsvm
//using libsvm to trian the sample
var classifier = ee.Classifier.libsvm().train({
  features: trainingPartition,
  classProperty:  'ClassID',
});
print(classifier).

//get the classification
var classified = image.select(bands).classify(classifier);
//Display the classification map
Map.centerObject(classNames, 11);
Map.addLayer(classified, {palette: landcoverPalette, min: 0, max: 2 }, 'classified');
print (classified, "11");
//Display the classification map
var test = testingPartition.classify(classifier);
var confusionMatrix = test.errorMatrix('ClassID', 'classification');
print ('confusionMatrix', confusionMatrix);
print ('overall accuracy', confusionMatrix.accuracy());
print ('kappa accuracy', confusionMatrix.kappa());

Export.image.toDrive({
  image: classified,
  description: 'wheat',
  folder: 'C',
  scale: 30,
  maxPixels: 34e10
});
```