A machine learning algorithm for mapping small reservoirs using Sentinel-2 satellite imagery in Google Earth Engine

Ebenezer K. Siabi
Komlavi Akpoti
Sander J. Zwart

December 2023

Author affiliations

Edenezer K. Siabi[1], Komlavi Akpoti[1], Sander J. Zwart[1]

[1] International Water Management Institute, Accra, Ghana

Suggested citation.

Siabi, E. K.; Akpoti, K.; Zwart, S. J. 2023. *A machine learning algorithm for mapping small reservoirs using Sentinel-2 satellite imagery in Google Earth Engine*. Colombo, Sri Lanka: International Water Management Institute (IWMI). CGIAR Initiative on Aquatic Foods. 13p.

Acknowledgements

This work was carried out with support from the CGIAR Initiative on Aquatic Foods (AqFI). We would like to thank all funders who supported this research through their contributions to the CGIAR Trust Fund.

Cover photo: Giulia Zane (IWMI)

CGIAR Initiative on Aquatic Foods (AqFI)

The CGIAR Initiative on Aquatic Foods aims to tackle systemic challenges to the sustainability and resilience of aquatic food systems, including data gaps that lead to exclusion of the sector from wider food and nutrition policies and programs, and limited research investment. Working closely with research partners in fisheries and aquaculture, civil society, industry, and governments, the Initiative contributes to the reduction of greenhouse gas emissions from the production of aquatic foods and enhance ecological and social resilience through development and dissemination of improved fish strains, better management practices, integrated fish-rice production systems, and fish-friendly irrigation systems. Learn more about AqFI here**:** https://www.cgiar.org/initiative/aquatic-foods/

Disclaimer

This publication has been prepared as an output of the CGIAR Initiative on Aquatic Foods and has not been independently peer-reviewed. Responsibility for editing, proofreading, layout, opinions expressed, and any possible errors lies with the authors and not the institutions involved. The boundaries and names shown, and the designations used on maps do not imply official endorsement or acceptance by IWMI, CGIAR, our partner institutions, or donors.

1. Background

Aquatic Food Systems and Global Challenges. Aquatic foods, encompassing a wide range of species and ecosystems, play a pivotal role in the global food system. They offer substantial benefits, including nutritional value, livelihood support, and economic stability. However, these systems are increasingly under threat due to stressors such as overfishing, pollution, climate change impacts, and governance challenges. These threats compromise the sustainability of aquatic ecosystems, impacting the lives and food security of communities that depend on these resources.

The Resilient Aquatic Food Systems (AqFS) Initiative. The AqFS Initiative, a collaboration between World Fish and the International Water Management Institute (IWMI), is a response to these global challenges. It operates in six regions across 11 countries, tackling systematic issues like data gaps, gender imbalances, and inefficient water management. The initiative aims to harness the potential of aquaculture for multiple uses, addressing food insecurity and promoting sustainable practices.

Focus on Ghana. In Ghana, the AqFS Initiative is particularly focused on integrating Aquatic Food Systems into comprehensive water management plans and developing data-driven strategies to adapt and enhance these systems in the context of climate change. The initiative seeks to leverage the multifunctionality of water bodies, primarily small reservoirs, to improve food security and empower local communities, particularly women and youth.

Project Objectives in Northern Ghana. In Northern Ghana, the initiative's goal is to introduce fish cage culture in small reservoirs. This effort is in partnership with the Fisheries Commission and the Water Research Institute (CSIR-WRI). The project's objectives are multifaceted: enhancing the functionality of water bodies, bolstering food security, and fostering empowerment through aquaculture business development. It also aims to scale successful business models to other communities, especially those around inland valleys.

Mapping Small Reservoirs: A Critical Step. A key component of this initiative is the identification and characterization of small reservoirs suitable for aquaculture. This is where the current project's focus lies. Between 2020 and 2022, IWMI conducted a study to map small reservoirs in the Upper East Region of Ghana. Using satellite imagery from Sentinel-2 and Google Earth Engine, the team mapped these water bodies during six consecutive dry seasons, with high accuracy validated by ground observations.

Advancing the Methodology. In 2023, the methodology to map small reservoirs was further refined to reduce cloud contamination in satellite imagery. This report delves into the enhanced Google Earth scripts used for this purpose, providing a robust framework for high-quality mapping of reservoir extent in northern Ghana.

Relevance and Implications. This report's findings are integral to understanding water availability and assessing the viability of aquaculture in these small reservoirs. The detailed insights into the dynamics of these water bodies are crucial for the broader objectives of the AqFS Initiative in Northern Ghana. They support goals such as enhancing food security, promoting sustainable aquaculture practices, and empowering local communities. Furthermore, the methodologies and insights from this study contribute to addressing the broader challenges faced by aquatic food systems worldwide, in line with the concerns raised by various researchers and initiatives.

2. Methodology

Figure 1 illustrates the step-by-step process for mapping small reservoirs, encompassing the datasets used, cloud removal, algorithm development, as well as the classification and extraction of small reservoir extents. The following sections outline the steps and detail the Google Earth Engine script.

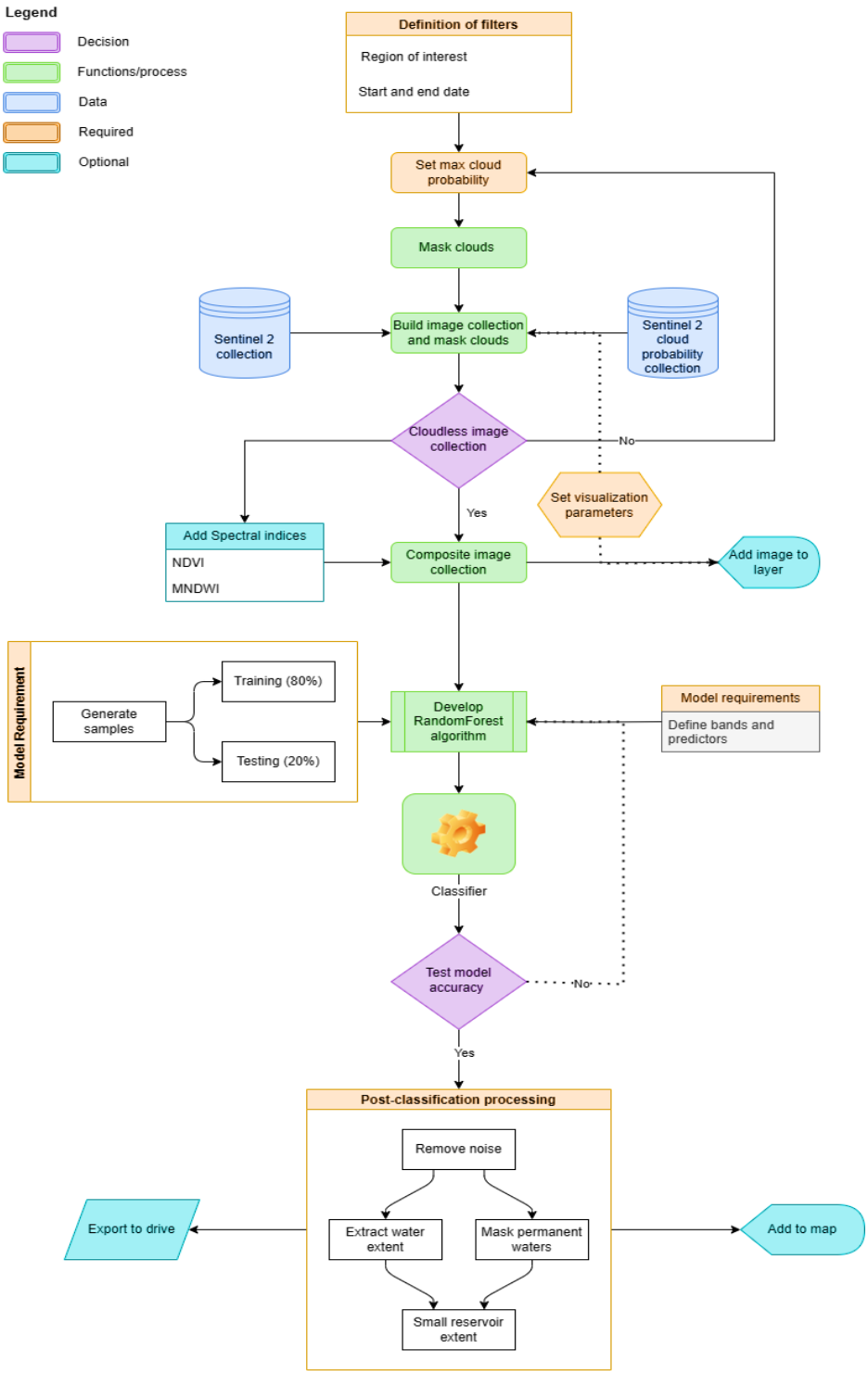

Figure 1. The overall methodology for mapping small reservoirs

2.1. Definition of image collection filters and cloud mask conditions

This section is intended to build filters for Sentinel 2 image collections (Figure 2) over the five northern regions of Ghana (Upper West, Upper East, North-East, Northern, and Savannah) for the study period (November 2018 to April 2023). Here, the region of interest is the boundary of Northern Ghana. The study focused on building monthly composites for each dry year. Therefore, the values currently set for the start and end dates represent the first month (November) of the first dry year (2018). We then set the maximum cloud probability value where values greater than are identified as a cloud. This requires the user to try different threshold values based on their preferences until an optimal threshold is reached. However, a default value of 40% is recommended. This study used an optimal maximum cloud probability threshold of 30% after applying different thresholds. Threshold values lower than our optimal threshold resulted in the removal of portions of the study area and small reservoirs detected as clouds or cloud shadows.

Conversely, those above our optimal threshold resulted in cloud contamination. We then added functions to mask clouds based on the maximum cloud probability. You will see a function Map.setCenter() in the code snippet below. This sets the viewport to a study area and at a zoom level of 7. Again, Map.setOptions '('satellite) sets a high-resolution satellite image as a basemap imagery. The step-by-step process is shown in the code snippets below.

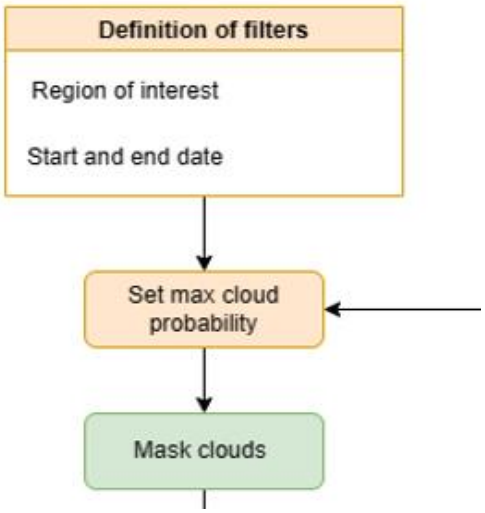

Figure 2. Image collection filters and cloud mask conditions

❖ Import the region of interest and center it. Set other options by adding satellite as the base map.

```
var region = ee.FeatureCollection('projects/ee-siabikebenezer/assets/ROI_Kom');
Map.centerObject(region,7);
Map.setOptions('satellite')
```

❖ Define start and end dates for the image collection and determine maximum cloud probability. Maximum cloud probability may vary depending on the desired outcome. We set the maximum cloud probability threshold to 30 for the small reservoir mapping.

```
var START_DATE = ee.Date('2019-11-01');
var END_DATE = ee.Date('2019-11-30');
var MAX_CLOUD_PROBABILITY = 30;
```

❖ Create a function to mask clouds based on the cloud probability threshold.

```
function maskClouds(img) {
  var clouds = ee.Image(img.get('cloud_mask')).select('probability');
  var isNotCloud = clouds.lt(MAX_CLOUD_PROBABILITY);
  return img.updateMask(isNotCloud);
}
```

❖ The masks for the 10m bands sometimes do not exclude noise at scene edges. Therefore, we apply function to masks from the 20m and 60m bands as well.

```
function mask_Edges(Sent2_img) {
  return Sent2_img.updateMask(
      Sent2_img.select('B8A').mask().updateMask(Sent2_img.select('B9').mask()));
}
```

2.2. Building image collection and masking clouds

There are two main datasets needed to build a cloudless image collection. These are Sentinel 2 surface reflectance and Sentinel 2 cloud probability datasets (Figure 3). These two image collections must be filtered individually by date and region of interest. The filtered collections are joined afterwards. Here, we defined a function to filter the Sentinel 2 and Sentinel 2 cloud probability collections based on our study area (Northern Ghana) and date parameters (i.e. start date and end date). We then joined the two datasets on the system:index property. This generated a copy of the Sentinel 2 image collection for Northern Ghana, where each image has a new Sentinel 2 cloud probability property whose value is the corresponding Sentinel 2 cloud probability image. Here we can set visualization parameters to add the result to the map layer to explore the data for cloud contamination. If the result is fully corrected as desired, users may proceed to the next step. However, if there are cloud contaminations, such as cloud footprints and shadows, users may change the maximum cloud probability threshold until a desired outcome is achieved. As stated above, a maximum cloud probability threshold of 30% was found to be good for the study. The step-by-step process is shown in the code snippets below.

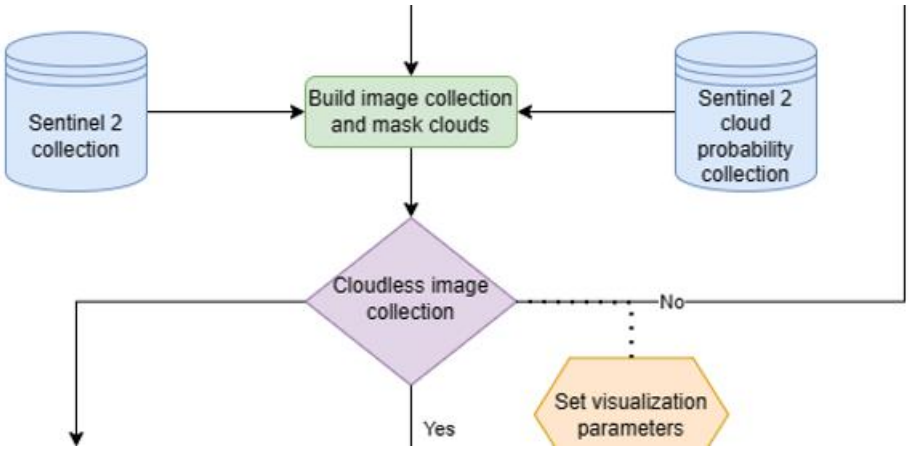

Figure 3. Building image collection and masking clouds

- Sentinel-2 surface reflectance and Sentinel-2 cloud probability are distinct image collections. Each collection needs to undergo similar filtering processes (e.g., based on date and ROI), followed by joining the filtered collections.

- Import Sentinel 2 collection and cloud probability datasets.

```
var Sent2 = ee.ImageCollection('COPERNICUS/S2_SR');
var Sent2_Clouds = ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY');
```

- Filter input image collections by desired data range and region.

```
var criteria = ee.Filter.and(
    ee.Filter.bounds(region), ee.Filter.date(START_DATE, END_DATE));
Sent2 = Sent2.filter(criteria).map(mask_Edges);
Sent2_Clouds = Sent2_Clouds.filter(criteria);
```

- Join the Sentinel 2 collection with the cloud probability dataset on the 'system: index' index' index' index' property and add cloud mask. The outcome is a new Sentinel 2 collection where each image has a new 'cloud 'probability' property whose value is the corresponding cloud probability image.

```
var Sent2WithCloudMask = ee.Join.saveFirst('cloud_mask').apply({
  primary: Sent2,
  secondary: Sent2_Clouds,
  condition:
      ee.Filter.equals({leftField: 'system:index', rightField: 'system:index'})
});

var Sent2CloudMasked30 =
    ee.ImageCollection(Sent2WithCloudMask).map(maskClouds);
```

- Comment out to print the image collection.

```
// print(Sent2CloudMasked30)
```

2.3. Spectral indices and image collection composition

We can add some spectral indices to our image collection. When applied, this will improve the image collection and prediction accuracy of predictive models. For this study, we estimated two spectral indices using a function; the Normalized Difference Vegetation Index (NDVI) and the Modified Normalized Difference Water Index (MNDWI) using the normalizedDifference() function. This function estimates the normalized difference between two bands. For NDVI, the NIR (B8) and RED (B4) were used for the estimation. For MNDWI, we used the GREEN (B3) and SWIR1 (B11) for the estimation (Figure 4). The two spectral indices were mapped across all images in the image collection. We then created a composite image using a selection criterion to each pixel from all the pixels in the image collection. Here, we employed the median() function to generate a composite where each pixel value is the median of all pixels from the image collection. Finally, we set visualization parameters to add the composited map to layer (see Figure 5). We used the Band 4, Band 3 and Band 2 as our visualization bands as well as a minimum and maximum values of 0 and 3000 respectively. The step-by-step process is shown in the code snippets below.

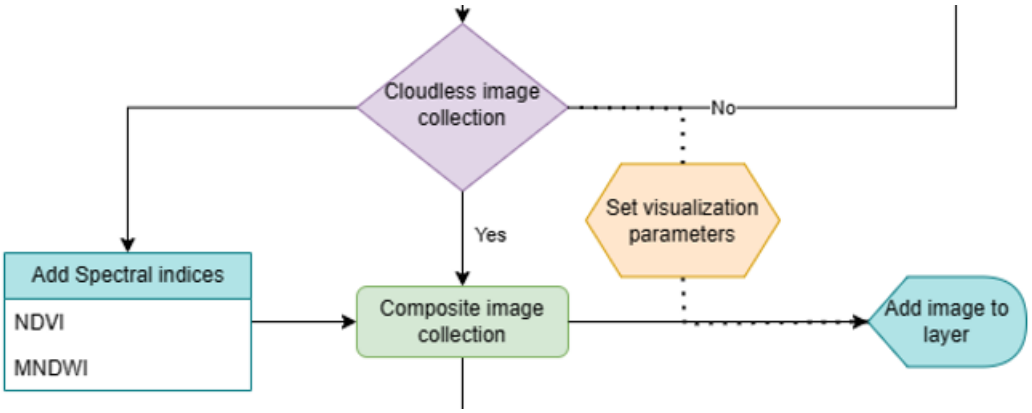

Figure 4. Spectral indices and image collection composition

- Define function to maps spectral indices to enhance classification accuracy

```
var addSpec = function(img){
   // MNDWI (Modified Normalized difference water index - Hanqui Xu, 2006)
  var mndwi = img.normalizedDifference(['B3', 'B6']).rename('MNDWI');
  // NDVI
  var ndvi = img.normalizedDifference(['B5','B4']).rename('NDVI');
  return img
    .addBands(mndwi)
    .addBands(ndvi)
};

var Sent2_Spec = Sent2CloudMasked30.map(addSpec)
```

- Composite the Sentinel 2 image collection

```
var composite = Sent2_Spec
                //uses the median reducer
                .median()
                // clips the composite to our ROI
                .clip(region)
  print(composite)
```

- Define visualization parameters for the image collection

```
var rgbVis = {min: 0, max: 3000, bands: ['B4', 'B3', 'B2']};
```

- Add layer to the map

```
Map.addLayer(composite, rgbVis, 'S2 Image RGB');
```

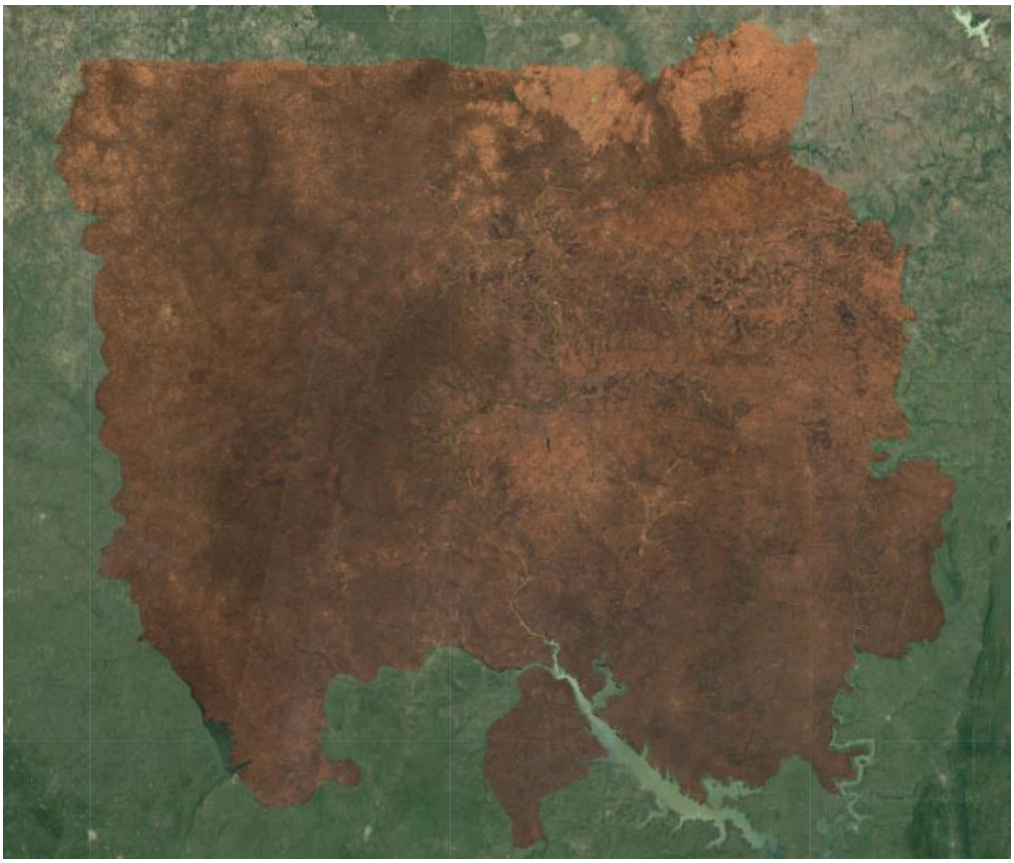

Figure 5. Sentinel 2 image over the five northern regions of Ghana

2.4. Developing Random Forest model for mapping

We developed a predictive model to map small reservoirs in the study area. There are several machine learning algorithms for mapping; however, we utilized the Random Forest algorithm for our study. We generated training samples from the high resolution basemap imagery provided by Google Maps. The aim is to classify each source pixel into two classes – water and non-water. We generated at least 80-100 feature collections (encompassing points and polygons) for each class using the drawing tool in the code editor. Each feature collection was given a property called 'water' with values of 0 and 1, showing whether the feature collection is non-water or water respectively. Here, we merged the water and non-water samples and divided them into training (80%) and testing (20%) datasets. Four bands (B3, B4, and B5 and MNDWI) were utilized as predictors. Subsequently, we used the .smileRandomForest classifier for the algorithm development with 100 trees and 4 randomly selected predictors per split. The term "smile" pertains to the Statistical Machine Intelligence and Learning Engine (SMILE), a JAVA library utilized by Google Earth Engine to implement these algorithms. The model was built using the training set and applied to all image pixels to generate a two-class image (Figure 6).

Obtaining a quantitative estimation of the classification's accuracy is crucial. We used the test dataset to validate our predictions. This is done once our model has been trained and applied to the entire image. In validating the model, the classified values were compared with the test set. We used the ee.Classifier.confusionMatrix function to estimate the confusion matrix representing the expected accuracy.

The next step is to clean the classified image from noise to obtain a good overall accuracy score. Here, we

used the .connectedPixelCount function to mask unconnected pixels. Also, we masked all permanent waters using a 100m buffered stream network vector. The result is then added to the map layer (see Figure 7). The step-by-step process is shown in the code snippets below.

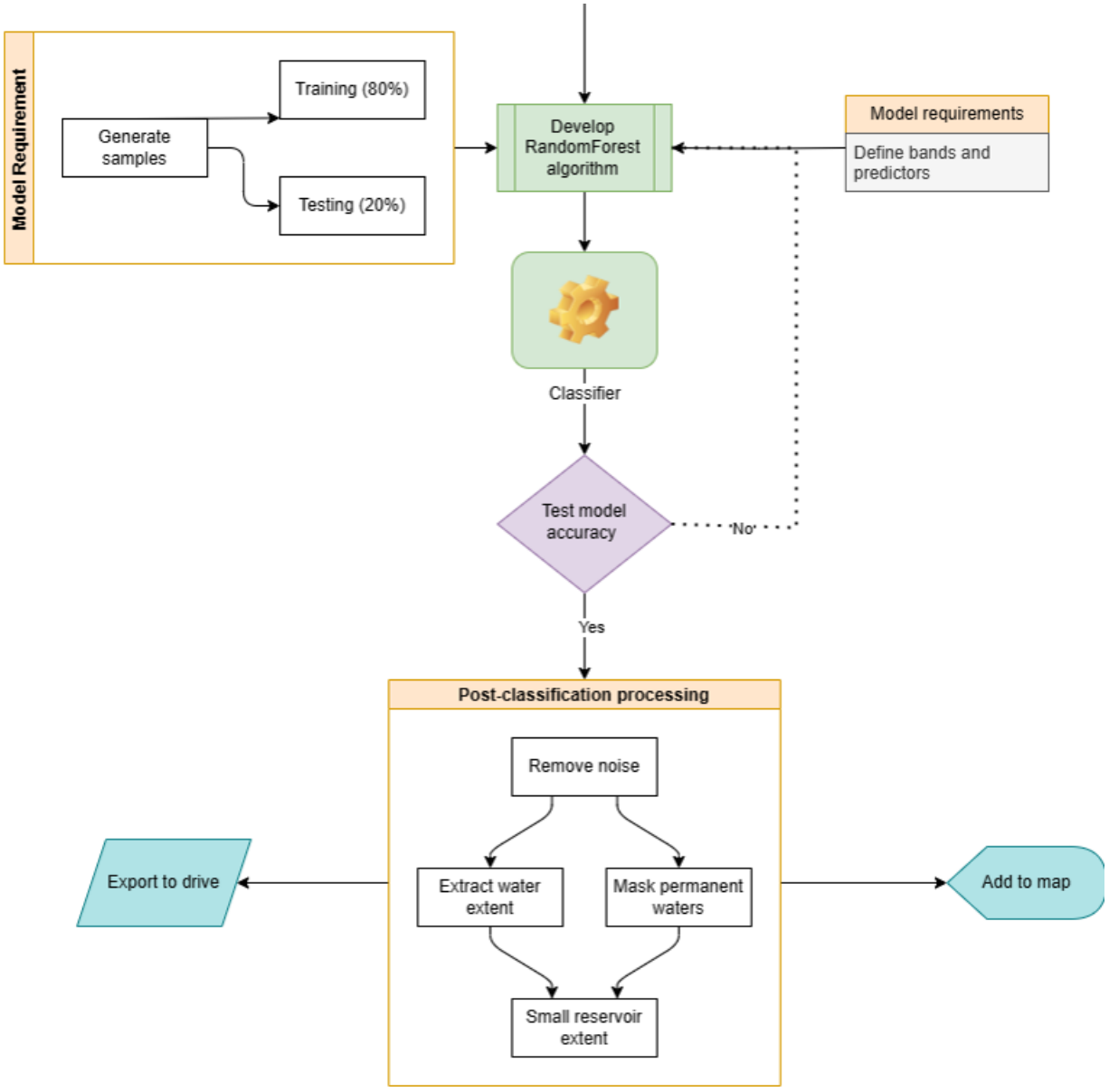

Figure 6. Random forest model framework for mapping

- Generate training samples and predictors. After drawing training points and polygons, merge water and non-water samples

```
var classes = Water.merge(nonWater)
```

- Define the Bands to include in the model

```
var bands = ['B4','B5','B3', 'MNDWI']
```

- Declare a variable called image to select the bands of interest and clip to ROI

```
var image = composite.select(bands).clip(region)
```

❖ Assemble samples for the RandomForest model.

```
// Assemble samples for the model
var samples = image.sampleRegions({
    collection: classes, //set of geometries selected for training
    properties: ['water'],
    scale: 10 //make each sample the same size as landsat pixel
    }).randomColumn('random'); //creates a column with random numbers
```

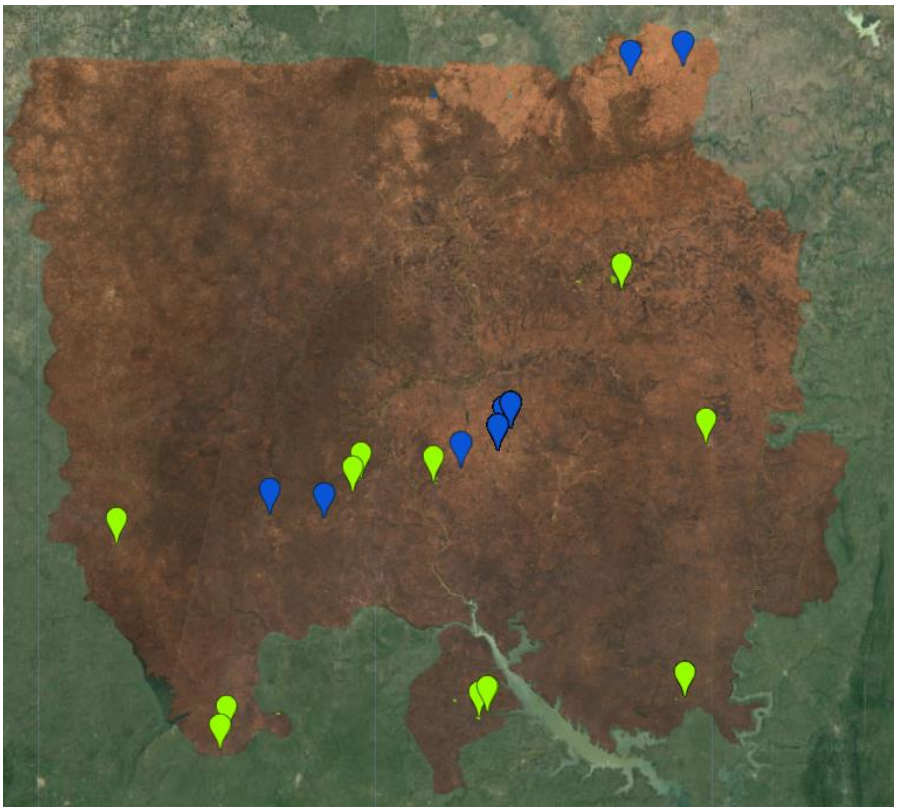

Figure 7. Sample points and polygons for training and testing

❖ Split samples into training and testing

```
var split = 0.8; //80% of training, 20% for testing
var training = samples.filter(ee.Filter.lt('random', split)); //subset training
var testing = samples.filter(ee.Filter.gte('random', split)); //subset testing data
```

❖ Comment out to print these variables to see how much training and testing data were used.

```
// print('Overall Samples n =', samples.aggregate_count('.all'));
// print('Training Samples n =', training.aggregate_count('.all'));
// print('Testing Samples n =', testing.aggregate_count('.all'));
```

❖ Begin Random forest classification using .smileRandomForest for a model with 100 trees and 4 randomly selected predictors per split.

```
var classifier = ee.Classifier.smileRandomForest(100,4).train({
    features: training.select(['B4','B5','B3', 'MNDWI', 'water']),
    classProperty: 'water', //Pull the water property from classes
    inputProperties: bands
    });
```

❖ Test the accuracy of the model.

```
var validation = testing.classify(classifier);
var testAccuracy = validation.errorMatrix('water', 'classification');
print('Validation error matrix RF:', testAccuracy);
print('Validation overall accuracy RF:', testAccuracy.accuracy());
```

```
Validation error matrix RF:                                    JSON
▸ [[9168,9],[33,2400]]                                         JSON

Validation overall accuracy RF:                                JSON
0.9963824289405685
```

Example of Validation error matrix and the overall accuracy of the RandomForest model

❖ Classify the merged composite using the randomforest model

```
var classifiedrf = image.select(bands)// select the predictors
                        .classify(classifier); //.classify applies the random forest
```

❖ Remove noise from model results by masking unconnected pixels

```
var pixelcount = classifiedrf.connectedPixelCount(5,false);
var countmask = pixelcount.select(0).gt(2);
```

❖ Mask the results to only display water extent

```
var classMask = classifiedrf.select('classification').gt(0)
var classed = classifiedrf.updateMask(countmask).updateMask(classMask);
```

❖ Mask all permanent waters to display small reservoir extent

```
var SRE = classed
var RiverNet = ee.FeatureCollection('projects/ee-siabikebenezer/assets/Buff_new1')
var mask = ee.Image.constant(1).clip(RiverNet.geometry()).mask().not()
SRE = SRE.updateMask(mask)
```

❖ Add the final result to the map (Figure 8)

```
Map.addLayer(SRE, {min: 1, max: 1, palette: 'red, red'}, 'Masked Small Reservoir extent');
```

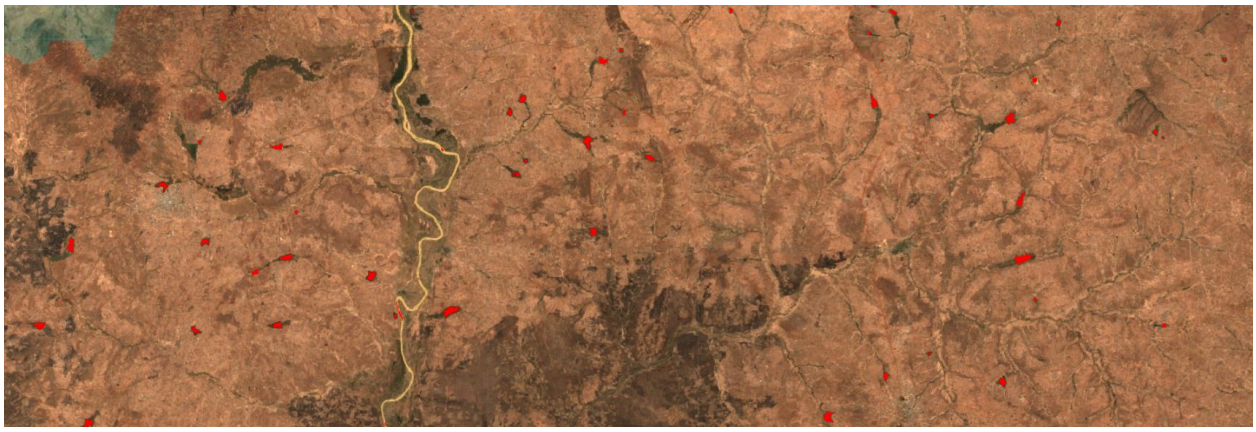

Figure 8. Small Reservoir extent classified by the Random Forest model

2.5. Exporting Small Reservoir Extent

After defining the required parameters, the Export.image.toDrive function is used to export the classified Small reservoir extent in a raster format. Here, the image is the classified small reservoir extent. The description represents the file name. The region represents the region of interest (Northern Ghana). We set a scale for Earth Engine to estimate a crsTransform parameter. Finally, the maxPixels parameter is designed to avoid the unintended creation of excessively large exports. If the default value doesn't align with a user's desired output image, it can be adjusted by increasing maxPixels. Our maxPixels was set to 1e13 for the export.

❖ Define parameters to export small reservoir extent to drive.

```
Export.image.toDrive({
  image: SRE,
  description: 'SREExtent',
  region: region,
  scale: 10,
  maxPixels: 1e13
  });
```

3. Code availability

The full code is available as open access at Github:

https://github.com/AKPOTI/Technical-note---Small-reservoir-mapping/blob/main/GEE_Codes.js

Basic small reservoir explorer app: https://siabikebenezer.users.earthengine.app/view/small-reservoir-extent-explorer