Configuration Manual of Potential Coffee Production Hotspots Using Machine Learning Techniques : Nagaland and Manipur, India

MSc Research Project
MSc. In Data Analytics

Nitish Sharma
Student ID: x21154147

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Nitish Sharma
………

Student ID: X21154147……………………………………………………………………………………..……

Programme: MSc. In Data Analytics ……………………………… **Year:** 2022-2023……..

Module: MSc Research Project……………………………………………………………….………

Lecturer: Dr. Catherine Mulwa……………………………………………………………………………

Submission Due Date: 15th December 2022…………………………………………………….………

Project Title: Potential Coffee Production Hotspots Using Machine Learning Techniques : Nagaland and Manipur, India…………………….………

Word Count: 1793…………………………… **Page Count:** 18…………………….….…..………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……………………………………………………………………………………………………

Date: ……………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nitish Sharma
Student ID: x21154147

1 Introduction

The objective of the project is to find new potential hotspots for coffee cultivation by analysing and processing geospatial data from satelite imagery. The satelite data being used in this project is obtained from public datasets available on google earth engine. Unlike traditional route to use GIS and AHP and other machine learning methods on local computer, this research computes and processes data in earth engine using the earth engine code editor. While most of the other datasets are available on earth engine, in order to train the model data needs to be collected from coffee growing locations. This is obtained from GBIF. It can be downloaded using the GBIF python or R package from python or R respectively.It can be also manually downloaded from GBIF website (https://www.gbif.org/occurrence/search). GBIF needs a user to sign up before download. The species needs to be filtered by scientific name to download the data which is "C*offea L*" in this case.

2 Set up Earth Engine and load the datasets

In order to code in earth engine user needs to sign up for free.
After the registration user can create and share scripts.
The GBIF data containing latitude and longitude of coffee occurrence needs to be uploaded to earth engine.
When uploading the file latitude and longitude columns and coordinate reference system needs to be specified. For this research the Occurrence.csv file has been uploaded to users/CoffeeDatasets folder.

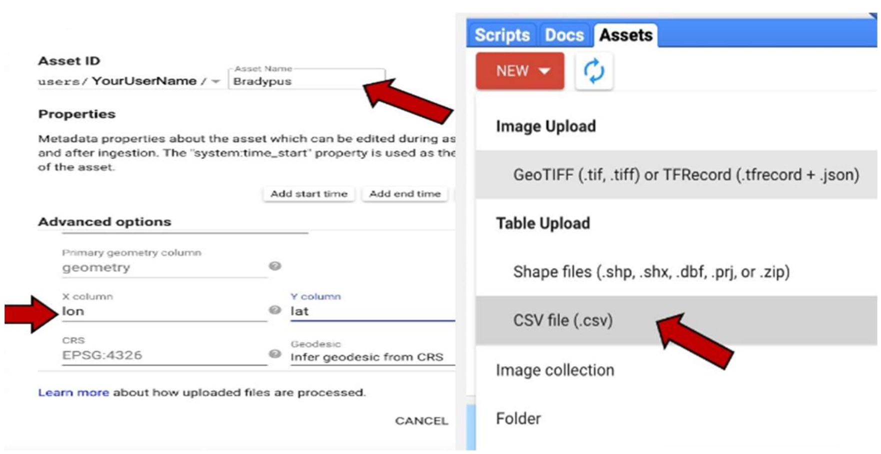


Figure 1 : Loading and Configuring occurrence dataset

3 Load Raster Datasets on earth engine

Following steps are taken to load raster datasets and clip them to area of interest that is India ,on earth engine

1) Load region data and Occurrence datasets into variables :

```
////////////////////////////////////////////////////////////////////////////
// Section 1 - Load species data, AOI, and remove duplicates
////////////////////////////////////////////////////////////////////////////

//////////// 1.1 Load presence data ////////////
var DataRaw = ee.FeatureCollection('users/CoffeeDatasets/Occurence');
print('Original data size:', DataRaw.size());

//////////// 1.2 Define the AOI ////////////
var Region  = ee.FeatureCollection("FAO/GAUL/2015/level2")
var Tar = ee.List(['Mizoram', 'Nagaland', 'Manipur', 'Arunachal Pradesh', 'T
var NE = Region.filter(ee.Filter.inList('ADM1_NAME', Tar));
// var AOI = NE


var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');
var roi = countries.filterMetadata('country_na', 'equals', 'India');
var AOI = roi.geometry().bounds().buffer(1000)
```

Figure 2: Loading Occurrence Data into variables

2) Define spatial resolution to work with and create split screen to show results.

Initially occurrence locations are shown on maps using red dots. These steps have been caried out on section 1.3 and 1.4 .

```
//////////// 1.3 Define spatial resolution to work with (in metres) ////////
var GrainSize = 10;

function RemoveDuplicates(data){
  var randomraster = ee.Image.random().reproject('EPSG:4326', null, GrainSiz
  var randpointvals = randomraster.sampleRegions({collection:ee.FeatureColle
  return randpointvals.distinct('random');
}

//////////// 1.4 Create split maps and link them to compare and visualie the
DataRaw = DataRaw.filter(ee.Filter.bounds(AOI));
var Data = RemoveDuplicates(DataRaw)
print('Final data size:', Data.size());

var left = ui.Map();
var right = ui.Map();
ui.root.clear();
ui.root.add(left);
ui.root.add(right);

// Link maps, so when you drag one map, the other will be moved in sync.
ui.Map.Linker([left, right], 'change-bounds');
```

Figure 3 : Defining spatial resolution to work with

Figure 4: Split Maps showing occurences of coffee in India

The raster data for different climatic variables is captured using different datasets.
These datasets are described below :

Predictor	**Dataset Type/ Dataset**	**Dataset Description**
Temperature	ImageCollection/ WORLDCLIM/V1/MONTHLY	WorldClim V1 Bioclim provides bioclimatic variables that are derived from the monthly temperature and rainfall in order to generate more biologically meaningful values. It contains 4 bands tmin for minimum temperature, tmax for maximum temperature, tavg for average temperature and perc for mean annual precipitation. The data is spread across these four bands averaged on a monthly basis. For calculation of the temperatures a mean of the temperature is taken for a filtered collection spanning between Jan 2018 – Dec-2020
Elevation	Image/ CGIAR/SRTM90_V4	The original intention behind the production of the Shuttle Radar Topography Mission (SRTM) digital elevation collection was to deliver consistent and high-quality elevation data with a scope that was as close to worldwide as possible. This particular iteration of the SRTM digital elevation data has been processed to remove any gaps in the data as well as make its use more straightforward. It is a single band image consisting of elevation.
Aspect	Image/USGS/SRTMGL1_003	The digital elevation data produced by the Shuttle Radar Topography Mission (SRTM, see Farr et al. 2007) is the

		result of an international research effort that attempted to obtain digital elevation models on a scale that was close to global. With a resolution of 1 arc-second, this SRTM V3 product, also known as SRTM Plus, is made available by NASA JPL. Aspect is calculated using feature engineering on elevation data by applying terrain functions on elevation data and singling out aspect band.
Slope	Image/USGS/SRTMGL1_003	Slope is calculated from elevation data using feature engineering on elevation data.
		Terrain function is applied and slope band is selected .
pH_Scale	Image/ OpenLandMap/SOL/SOL_PH-H2O_USDA-4C1A2A_M/v02	pH of the soil measured in H2O at six different depths (0, 10, 30, 60, 100, and 200 cm) with a resolution of 250 meters.
Soil Organic Content	Image/ OpenLandMap/SOL/SOL_ORGANIC-CARBON_USDA-6A1C_M/v02	The amount of organic carbon in the soil, expressed as x 5 g/kg, was measured at six different depths (0, 10, 30, 60, 100, and 200 cm) with a resolution of 250 m. a conclusion reached after compiling soil data from around the world.
Rainfall	ImageCollection/WORLDCLIM/ V1/BIO	The WorldClim Version 1 Bioclim offers bioclimatic variables that are obtained from the monthly temperature and rainfall in order to provide more biologically meaningful data.It contains of data in various bands. Rainfall data used in this implementation is contained in band bio12 i.e Mean Annual Rainfall and bio13 being used to find rainfall during rainy season.
Hillshade	MODIS/006/MOD44B	The Terra MODIS Vegetation Continuous Fields (VCF) product is a representation of surface vegetation cover estimations on a global scale that is done at the sub-pixel level. Developed to continually portray the terrestrial surface of the Earth as a fraction of fundamental vegetation characteristics. Hillshade is present as a band in the image collection.
Percent Tree Cover	MODIS/006/MOD44B	Percent Tree cover is calculated by selecting percent

		tree cover band in the above image collection. For this the collection is filtered by taking mean for a time period of Jan - 2018 to Dec to 2020.

Table 1 – Raster Predictors and their descriptions

All these data sets are loaded in earth engine as shown below, the feature engineered predictors are also depicted in following figure 5. These steps have been caried out in section 2.3 to 2.5

```
//Feature Engineering slope and aspect from the elevation image

var Terrain = ee.Algorithms.Terrain(ee.Image("USGS/SRTMGL1_003"));


var slope = ee.Terrain.slope(ee.Image("USGS/SRTMGL1_003"));
var aspect = ee.Terrain.aspect(ee.Image("USGS/SRTMGL1_003"));


////////////////// 2.3 Load and process soil data to find soil related fact

var pH_Scale = ee.Image("OpenLandMap/SOL/SOL_PH-H2O_USDA-4C1A2A_M/v02");
var org_content = ee.Image("OpenLandMap/SOL/SOL_ORGANIC-CARBON_USDA-6A1C_M/


////////////////// 2.4 Load and process rainfall and temperature data /////

var BIO = ee.Image("WORLDCLIM/V1/BIO");


////////////////// 2.5 Load NDVI 250 m collection and estimate median annua
var MODIS = ee.ImageCollection("MODIS/006/MOD44B");

//Feature Engineering  : percent tree cover
var MeanPTC = MODIS.filterDate('2018-01-01', '2020-12-31').select(['Percent
```

Figure 5a – Loading predictor datasets

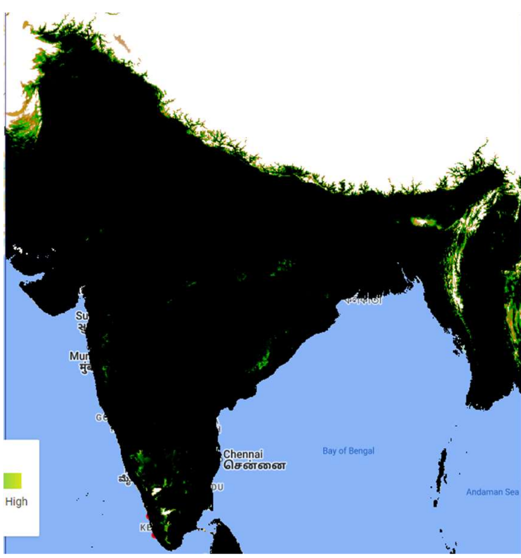
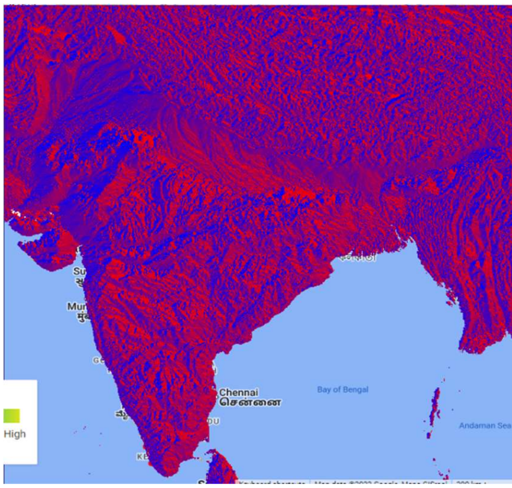


Figure 5b – Predictor - Elevation

Figure 5c – Predictor - Slope



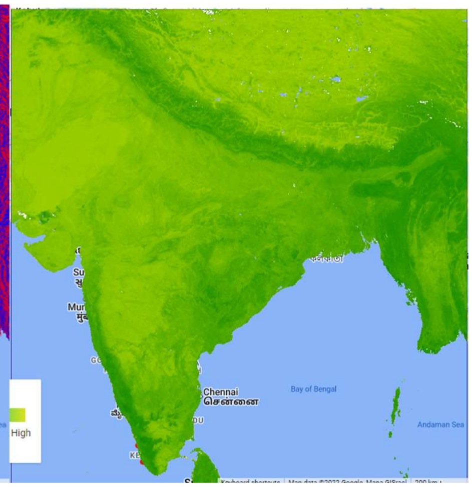
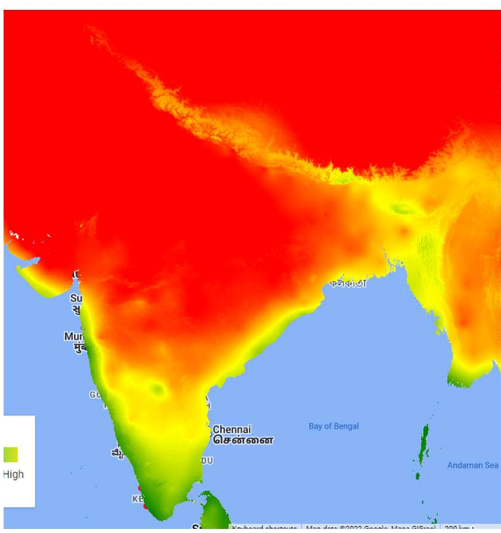
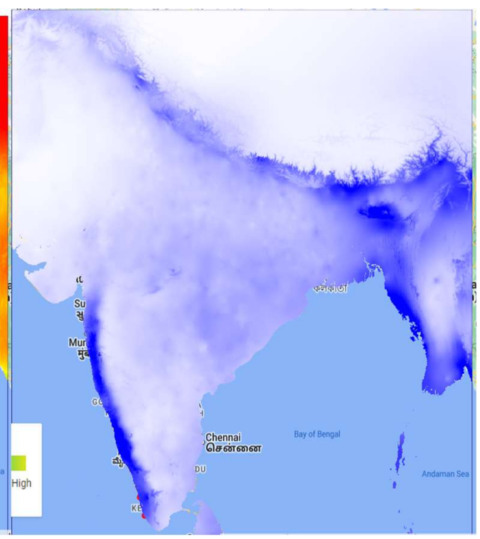


Figure 5d – Predictor – Aspect

Figure 5e – Predictor – pH of soil





5f – Predictor - Annual Temperature Range

5g – Predictor – Mean Annual Rainfall

4 Data Pre Processing

In this research we are working with multiband geospatial data. In such raster images each band comprises of data for one predictor variables. Here in order to combine all the predictor values into a single file we add all the raster layers together using addBands function as shown in figure 6. It is calculated in section 2.6 .

```
////////////////// 2.6 combining predictors defined in above sections to a
//Taking only the needed bands

var pred = BIO.addBands(pH_Scale).addBands(org_content).addBands(Terrain).a
var bands = ['b10','b30_1','bio07','bio01','bio13','bio12','elevation','slo
var preds = pred.select(bands);

var imgNewBands = preds.select(['b10','b30_1','bio07', 'bio13', 'bio12','el
                        .rename(['Soil water pH at 30 cm', 'Soil Organic Content'

var predictors = imgNewBands;

var watermask =  Terrain.select('elevation').gt(0); //Create a water mask
var predictors = predictors.updateMask(watermask).clip(AOI);


print(predictors);
```

Figure 6 : Loading predictors into a single multiband image

After adding all the predictors correlation is determined between predictors and correlated variables are then removed from final raster. Snippets of correlation matrix and the code are shown in figure 7, figure 8 and figure 9. It is calculated in section 2.7.

```
///////////////// 2.7 Calculating correlation between predictors /////////////////

var DataCor = predictors.sample({scale: GrainSize, numPixels: 5000, geometries: true}); //Generate 5000 random points
var PixelVals = predictors.sampleRegions({collection: DataCor, scale: GrainSize, tileScale: 16}); //Extract covariate values

// // To check all pairwise correlations we need to map the reduceColumns function across all pairwise combinations of predictors
var CorrAll = predictors.bandNames().map(function(i){
    var tmp1 = predictors.bandNames().map(function(j){
      var tmp2 = PixelVals.reduceColumns({
        reducer: ee.Reducer.spearmansCorrelation(),
        selectors: [i, j]
      });
    return tmp2.get('correlation');
    });
    return tmp1;
  });
print('Variables correlation matrix',CorrAll);
```

Figure 7 – Calculating the correlation matrix

```
  Variables correlation matrix
 ▾List (13 elements)
   ▸0: List (13 elements)
   ▸1: List (13 elements)
   ▸2: List (13 elements)
   ▸3: List (13 elements)
   ▸4: List (13 elements)
   ▸5: List (13 elements)
   ▸6: List (13 elements)
   ▸7: List (13 elements)
   ▸8: List (13 elements)
   ▸9: List (13 elements)
   ▸10: List (13 elements)
   ▸11: List (13 elements)
   ▸12: List (13 elements)
```

Figure 8 – Correlation matrix collapsed view

```
 Variables correlation matrix
▾List (13 elements)
  ▾0: List (13 elements)
     0: 1
     1: -0.4723475491821722
     2: 0.7180379168810642
     3: -0.7068010986461689
     4: -0.7949950429336743
     5: 0.19110033045994
     6: -0.27127396714478824
     7: -0.02495231796895045
     8: 0.01395853301492471
     9: -0.7242286246999539
     10: -0.32361640860421614
     11: 0.02919389315565078
     12: -0.1503583132195778
  ▾1: List (13 elements)
     0: -0.47234754918217214
     1: 1
     2: -0.21696139003219134
     3: 0.1682033906860725
     4: 0.24551799320696974
     5: 0.30736553506476183
     6: 0.4420943435996417
     7: 0.05836400900676092
     8: 0.023125398020023627
     9: 0.39498885249320304
     10: -0.24132671766570218
     11: -0.4412806593431163
     12: -0.33987247503449836
  ▾2: List (13 elements)
     0: 0.7180379168810643
```
Figure 9 – Correlation matrix Expanded view

In order to produce pseudo-absences for the purpose of this research, a method of environmental profiling that involving two stages was utilized. A k-means clustering based on Euclidean distance for a subset of 1,000 randomly chosen occurrences in order to restrict the area for the creation of pseudo-absences to pixels that were more dissimilar to the environmental profile of the occurrence data, were done . This allowed to create a more accurate picture of the distribution of the data. This was followed by creation of a grid over area of interest . Figure 10 and 11 shows its implementation.

```
////////////////// 2.9 Defining blocks to fold randomly for cross validation//////////////////


// Creating an image for the presence locations. The pixels having occurences will be removed fr
// This will prevent having presence and pseudo-absences in the same pixel.

var mask = Data
  .reduceToImage({
    properties: ['random'],
    reducer: ee.Reducer.first()
}).reproject('EPSG:4326', null, ee.Number(GrainSize)).mask().neq(1).selfMask();

// Extract local covariate values from multiband predictor image at presence points

var PixelVals = predictors.sampleRegions({collection: Data.randomColumn({seed:5}).sort('random')

// Instantiate the clusterer and train it.

var clusterer = ee.Clusterer.wekaKMeans({nClusters:2, distanceFunction:"Euclidean", fast: true})

// Cluster the input using the trained clusterer.

var ClResult = predictors.cluster(clusterer);

// Retain cluster class mode dissimilar to occurrence data
var ClMask = ClResult.select(['cluster']).eq(1);

var AreaForPA =  mask.updateMask(ClMask);

Map.addLayer(AreaForPA, {},'Area to create pseudo-absences', 0);
```
Figure 10 – Defining blocks for cross validation

```
///////////////// 2.10 Define a function to create a grid over AOI /////////////////

function makeGrid(Geometry, scale) {

  // pixelLonLat returns an image with each pixel labeled with longitude and

  // latitude values.

  var lonLat = ee.Image.pixelLonLat();
  // Select the longitude and latitude bands, multiply by a large number then
  // truncate them to integers.

  var lonGrid = lonLat
    .select('longitude')
    .multiply(100000)
    .toInt();
  var latGrid = lonLat
    .select('latitude')
    .multiply(100000)
    .toInt();
  return lonGrid
    .multiply(latGrid)
    .reduceToVectors({
      geometry: Geometry,
      scale: scale,
      geometryType: 'polygon',
    });
}
// Create grid and remove cells outside AOI
var Scale = 50000; // Set range in m to create spatial blocks
var Grid = makeGrid(AOI, Scale);
Map.addLayer(Grid, {},'Grid for spatail block cross validation', 0);
```

Figure 11 – Creating Grid over AOI

5 Fit the Models and Evaluate the results

This research utilises Random Forest, Gradient Boosting and Classification and Regeression trees. All these have been implemented in section 3.1, 3.2 and 3.3 the figures 12, 13a , 13b and 14 show fitting and evaluation for random forest :

```
//////////// 3.1.1.b Define species development model function and Activate the random forest clas:

function SDM_RF(x) {
  var Seed = ee.Number(x);

  // Randomly split blocks for training and validation
  var GRID = ee.FeatureCollection(Grid).randomColumn({seed:Seed}).sort('random');
  var TrainingGrid = GRID.filter(ee.Filter.lt('random', split));  // Filter points with 'random'
  var TestingGrid = GRID.filter(ee.Filter.gte('random', split));  // Filter points with 'random'

  // Presence
  var PresencePoints = ee.FeatureCollection(Data);
  PresencePoints = PresencePoints.map(function(feature){return feature.set('PresAbs', 1)});
  var TrPresencePoints = PresencePoints.filter(ee.Filter.bounds(TrainingGrid));  // Filter prese
  var TePresencePoints = PresencePoints.filter(ee.Filter.bounds(TestingGrid));  // Filter presen

  // Pseudo-absences
  var TrPseudoAbsPoints = AreaForPA.sample({region: TrainingGrid, scale: GrainSize, numPixels: T
  TrPseudoAbsPoints = TrPseudoAbsPoints.randomColumn().sort('random').limit(ee.Number(TrPresence
  TrPseudoAbsPoints = TrPseudoAbsPoints.map(function(feature){
      return feature.set('PresAbs', 0);
      });

  var TePseudoAbsPoints = AreaForPA.sample({region: TestingGrid, scale: GrainSize, numPixels: Te
  TePseudoAbsPoints = TePseudoAbsPoints.randomColumn().sort('random').limit(ee.Number(TePresence
  TePseudoAbsPoints = TePseudoAbsPoints.map(function(feature){
      return feature.set('PresAbs', 0);
      });

  // Merge presence and pseudo-absencepoints
  var trainingPartition = TrPresencePoints.merge(TrPseudoAbsPoints);
  var testingPartition = TePresencePoints.merge(TePseudoAbsPoints);

  // Extract local covariate values from multiband predictor image at training points
  var trainPixelVals = predictors.sampleRegions({collection: trainingPartition, properties: ['Pr

  // Classify using random forest
  var Classifier = ee.Classifier.smileRandomForest({
    numberOfTrees: 500, //The number of decision trees to create.
    variablesPerSplit: null, //The number of variables per split. If unspecified, uses the squa
    minLeafPopulation: 10,//Only create nodes whose training set contains at least this many po
    bagFraction: 0.5,//The fraction of input to bag per tree. Default: 0.5.
    maxNodes: null,//The maximum number of leaf nodes in each tree. If unspecified, defaults to
    seed: Seed//The randomization seed.
   });
```

Figure 12 – Implementing Random forest species development model

```
//////////// 3.1.2.a Developing the Habitat suitability Maps ////////////
// Set visualization parameters
var visParams = {
  min: 0,
  max: 1,
  palette: ["#440154FF","#482677FF","#404788FF","#33638DFF","#287D8EFF",
  "#1F968BFF","#29AF7FFF","#55C667FF","#95D840FF","#DCE319FF"],
};

// Extract all model predictions
var images = ee.List.sequence(0,ee.Number(numiter).multiply(4).subtract(1),4).map(function(x){
  return results.get(x)});

// Calculate mean of all individual model runs
var GAUL = ee.FeatureCollection("FAO/GAUL/2015/level2");
var States = ee.List(['Manipur','Nagaland']);
var NM = GAUL.filter(ee.Filter.inList('ADM1_NAME', States));

var ModelAverage = ee.ImageCollection.fromImages(images).mean().clip(NM);

// Add final habitat suitability layer and presence locations to the map
right.addLayer(ModelAverage, visParams, 'Habitat Suitability predicted by Random Forest Classifier


// Create legend for habitat suitability map.
var legend = ui.Panel({style: {position: 'bottom-left', padding: '8px 15px'}});

legend.add(ui.Label({
  value: "Habitat suitability",
  style: {fontWeight: 'bold', fontSize: '18px', margin: '0 0 4px 0', padding: '0px'}
}));

legend.add(ui.Thumbnail({
  image: ee.Image.pixelLonLat().select(0),
  params: {
    bbox: [0,0,1,0.1],
    dimensions: '200x20',
    format: 'png',
    min: 0,
    max: 1,
    palette: ["#440154FF","#482677FF","#404788FF","#33638DFF","#287D8EFF",
  "#1F968BFF","#29AF7FFF","#55C667FF","#95D840FF","#DCE319FF"]
  },
  style: {stretch: 'horizontal', margin: '8px 8px', maxHeight: '40px'},
}));

legend.add(ui.Panel({
  widgets: [
    ui.Label('Low', {margin: '0px 0px', textAlign: 'left', stretch: 'horizontal'}),
    ui.Label('Medium', {margin: '0px 0px', textAlign: 'center', stretch: 'horizontal'}),
    ui.Label('High', {margin: '0px 0px', textAlign: 'right', stretch: 'horizontal'}),
    ],layout: ui.Panel.Layout.Flow('horizontal')
}));
```

Figure 13 a - Visualizing Random Forest results

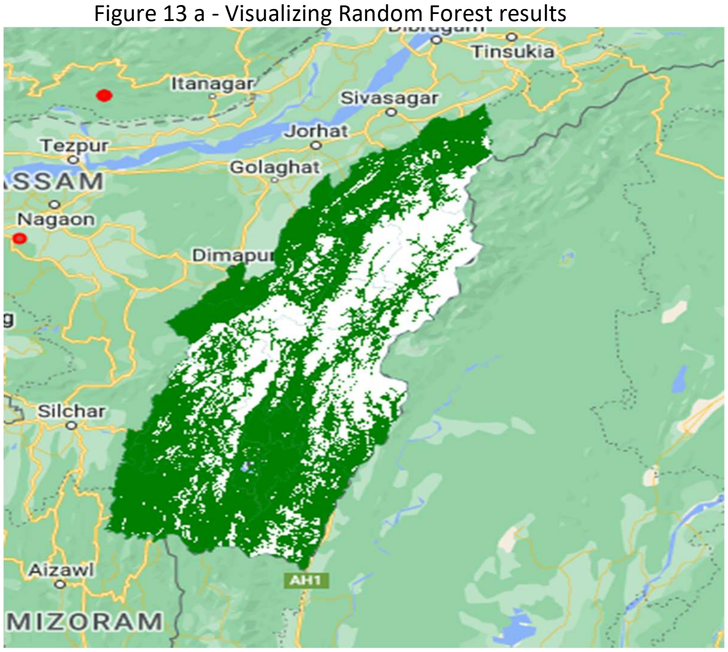


Figure 13 b- Visualizing potential hotspots predicted by ranmdom forest

```
//////////// 3.1.3.b Calculate AUC of the Receiver Operator Characteristic ////////////
function getAUCROC(x){
  var X = ee.Array(x.aggregate_array('FPR'));
  var Y = ee.Array(x.aggregate_array('TPR'));
  var X1 = X.slice(0,1).subtract(X.slice(0,0,-1));
  var Y1 = Y.slice(0,1).add(Y.slice(0,0,-1));
  return X1.multiply(Y1).multiply(0.5).reduce('sum',[0]).abs().toList().get(0);
}

function AUCROCaccuracy(x){
  var HSM = ee.Image(images.get(x));
  var TData = ee.FeatureCollection(TestingDatasets.get(x));
  var Acc = getAcc(HSM, TData);
  return getAUCROC(Acc);
}


var AUCROCs = ee.List.sequence(0,ee.Number(numiter).subtract(1),1).map(AUCROCaccuracy);
print('AUC-ROC:', AUCROCs);
print('Mean AUC-ROC', AUCROCs.reduce(ee.Reducer.mean()));


//////////// 3.1.3.c Calculate AUC of Precision Recall Curve ////////////
function getAUCPR(roc){
  var X = ee.Array(roc.aggregate_array('TPR'));
  var Y = ee.Array(roc.aggregate_array('Precision'));
  var X1 = X.slice(0,1).subtract(X.slice(0,0,-1));
  var Y1 = Y.slice(0,1).add(Y.slice(0,0,-1));
  return X1.multiply(Y1).multiply(0.5).reduce('sum',[0]).abs().toList().get(0);
}

function AUCPRaccuracy(x){
  var HSM = ee.Image(images.get(x));
  var TData = ee.FeatureCollection(TestingDatasets.get(x));
  var Acc = getAcc(HSM, TData);
  return getAUCPR(Acc);
}

var AUCPRs = ee.List.sequence(0,ee.Number(numiter).subtract(1),1).map(AUCPRaccuracy);
print('AUC-PR:', AUCPRs);
print('Mean AUC-PR', AUCPRs.reduce(ee.Reducer.mean()));
```

Figure 14 - Validating Random Forest Results

The implementation of Gradient boosting is done in section 3.2 of the project. It consits of various steps. The figures 15, 16a, 16b and 17 walk through implementation and validation of Gradient Boosting algorithm.

```
//////////// 3.2.1.b Define species development model function and Activate the Gradient Boosting 

function SDM_GB(x) {
    var Seed = ee.Number(x);

    // Randomly split blocks for training and validation
    var GRID = ee.FeatureCollection(Grid).randomColumn({seed:Seed}).sort('random');
    var TrainingGrid = GRID.filter(ee.Filter.lt('random', split));  // Filter points with 'random'
    var TestingGrid = GRID.filter(ee.Filter.gte('random', split));  // Filter points with 'random'

    // Presence
    var PresencePoints = ee.FeatureCollection(Data);
    PresencePoints = PresencePoints.map(function(feature){return feature.set('PresAbs', 1)});
    var TrPresencePoints = PresencePoints.filter(ee.Filter.bounds(TrainingGrid));  // Filter prese
    var TePresencePoints = PresencePoints.filter(ee.Filter.bounds(TestingGrid));  // Filter presen

    // Pseudo-absences
    var TrPseudoAbsPoints = AreaForPA.sample({region: TrainingGrid, scale: GrainSize, numPixels: T
    TrPseudoAbsPoints = TrPseudoAbsPoints.randomColumn().sort('random').limit(ee.Number(TrPresence
    TrPseudoAbsPoints = TrPseudoAbsPoints.map(function(feature){
        return feature.set('PresAbs', 0);
        });

    var TePseudoAbsPoints = AreaForPA.sample({region: TestingGrid, scale: GrainSize, numPixels: Te
    TePseudoAbsPoints = TePseudoAbsPoints.randomColumn().sort('random').limit(ee.Number(TePresence
    TePseudoAbsPoints = TePseudoAbsPoints.map(function(feature){
        return feature.set('PresAbs', 0);
        });

    // Merge presence and pseudo-absencepoints
    var trainingPartition = TrPresencePoints.merge(TrPseudoAbsPoints);
    var testingPartition = TePresencePoints.merge(TePseudoAbsPoints);

    // Extract local covariate values from multiband predictor image at training points
    var trainPixelVals = predictors.sampleRegions({collection: trainingPartition, properties: ['Pr


    // Classify using a gradient boosting
    var Classifiergb = ee.Classifier.smileGradientTreeBoost({
      numberOfTrees:500, //The number of decision trees to create.
      shrinkage: 0.005, //The shrinkage parameter in (0, 1) controls the learning rate of procedur
      samplingRate: 0.7, //The sampling rate for stochastic tree boosting. Default 0.07
      maxNodes: null, //The maximum number of leaf nodes in each tree. If unspecified, defaults to
      loss: "LeastAbsoluteDeviation", //Loss function for regression. One of: LeastSquares, LeastA
      seed:Seed //The randomization seed.
    });
```

Figure 15 – Implementation of Gradient boost species distribution model

```
});

//////////// 3.2.2.b Developing the potential Hotspot map ////////////

// Extract all model predictions

var images = ee.List.sequence(0,ee.Number(numiter).multiply(4).subtract(1),4).map(function(x){
  return results.get(x)});

// Calculate mean of all individual model runs
var GAUL = ee.FeatureCollection("FAO/GAUL/2015/level2");
var States = ee.List(['Manipur','Nagaland']);
var NM = GAUL.filter(ee.Filter.inList('ADM1_NAME', States));

var ModelAverage = ee.ImageCollection.fromImages(images).mean().clip(NM);

// Add final habitat suitability layer and presence locations to the map
right.addLayer(ModelAverage, visParams, 'Habitat Suitability predicted by Gradient Boosting Classi


// Distribution map

// Extract all model predictions
var images2 = ee.List.sequence(1,ee.Number(numiter).multiply(4).subtract(1),4).map(function(x){
  return results.get(x)});

// Calculate mean of all indivudual model runs
var DistributionMap = ee.ImageCollection.fromImages(images2).mode().clip(NM);

// Add final distribution map and presence locations to the map
right.addLayer(DistributionMap,
  {palette: ["white", "green"], min: 0, max: 1},
  'Potential distribution predicted by Gradient Boosting Classifiers');
```

Figure 16a – Visualizing Gradient Boosting Results

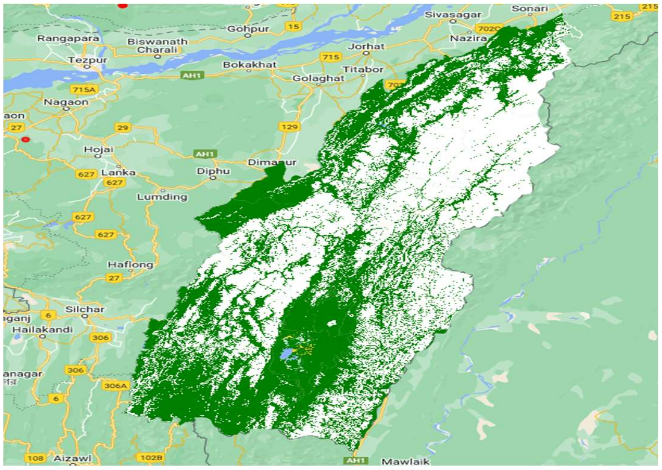


Figure 16-b Visualizing hotspots detected by Gradient Boosting Trees

```
//////////// 3.3.3.b Calculate AUC of the Receiver Operator Characteristic ////////////
function getAUCROC(x){
  var X = ee.Array(x.aggregate_array('FPR'));
  var Y = ee.Array(x.aggregate_array('TPR'));
  var X1 = X.slice(0,1).subtract(X.slice(0,0,-1));
  var Y1 = Y.slice(0,1).add(Y.slice(0,0,-1));
  return X1.multiply(Y1).multiply(0.5).reduce('sum',[0]).abs().toList().get(0);
}

function AUCROCaccuracy(x){
  var HSM = ee.Image(images.get(x));
  var TData = ee.FeatureCollection(TestingDatasets.get(x));
  var Acc = getAcc(HSM, TData);
  return getAUCROC(Acc);
}

var AUCROCs = ee.List.sequence(0,ee.Number(numiter).subtract(1),1).map(AUCROCaccuracy);
print('AUC-ROC:', AUCROCs);
print('Mean AUC-ROC', AUCROCs.reduce(ee.Reducer.mean()));

//////////// 3.3.3.c Calculate AUC of Precision Recall Curve ////////////
function getAUCPR(roc){
  var X = ee.Array(roc.aggregate_array('TPR'));
  var Y = ee.Array(roc.aggregate_array('Precision'));
  var X1 = X.slice(0,1).subtract(X.slice(0,0,-1));
  var Y1 = Y.slice(0,1).add(Y.slice(0,0,-1));
  return X1.multiply(Y1).multiply(0.5).reduce('sum',[0]).abs().toList().get(0);
}

function AUCPRaccuracy(x){
  var HSM = ee.Image(images.get(x));
  var TData = ee.FeatureCollection(TestingDatasets.get(x));
  var Acc = getAcc(HSM, TData);
  return getAUCPR(Acc);
}

var AUCPRs = ee.List.sequence(0,ee.Number(numiter).subtract(1),1).map(AUCPRaccuracy);
print('AUC-PR:', AUCPRs);
print('Mean AUC-PR', AUCPRs.reduce(ee.Reducer.mean()));
```

Figure 17 – Evaluating Gradient Boosting Results

The implementation of Classification and regression trees is done in section 3.3 of the project. It consists of various steps. The figures 18, 19a, 19b and 20 walk through implementation and validation of Gradient Boosting algorithm.

```
//////////// 3.3.1.b Define species development model function and Activate the Gradient Boosting 

function SDM_c(x) {
    var Seed = ee.Number(x);

    // Randomly split blocks for training and validation
    var GRID = ee.FeatureCollection(Grid).randomColumn({seed:Seed}).sort('random');
    var TrainingGrid = GRID.filter(ee.Filter.lt('random', split));  // Filter points with 'random'
    var TestingGrid = GRID.filter(ee.Filter.gte('random', split));  // Filter points with 'random'

    // Presence
    var PresencePoints = ee.FeatureCollection(Data);
    PresencePoints = PresencePoints.map(function(feature){return feature.set('PresAbs', 1)});
    var TrPresencePoints = PresencePoints.filter(ee.Filter.bounds(TrainingGrid));  // Filter prese
    var TePresencePoints = PresencePoints.filter(ee.Filter.bounds(TestingGrid));  // Filter presen

    // Pseudo-absences
    var TrPseudoAbsPoints = AreaForPA.sample({region: TrainingGrid, scale: GrainSize, numPixels: T
    TrPseudoAbsPoints = TrPseudoAbsPoints.randomColumn().sort('random').limit(ee.Number(TrPresence
    TrPseudoAbsPoints = TrPseudoAbsPoints.map(function(feature){
        return feature.set('PresAbs', 0);
        });

    var TePseudoAbsPoints = AreaForPA.sample({region: TestingGrid, scale: GrainSize, numPixels: Te
    TePseudoAbsPoints = TePseudoAbsPoints.randomColumn().sort('random').limit(ee.Number(TePresence
    TePseudoAbsPoints = TePseudoAbsPoints.map(function(feature){
        return feature.set('PresAbs', 0);
        });
```

Figure 18 – Implementation of species distribution model using cart

```
// Distribution map

// Extract all model predictions
var images2 = ee.List.sequence(1,ee.Number(numiter).multiply(4).subtract(1),4).map(function(x){
  return results.get(x)});

// Calculate mean of all indivudual model runs
var DistributionMap = ee.ImageCollection.fromImages(images2).mode().clip(NM);

// Add final distribution map and presence locations to the map
right.addLayer(DistributionMap,
  {palette: ["white", "green"], min: 0, max: 1},
  'Potential distribution predicted by CART Classifiers');
```

Figure 19 a – Visualizations generated using CART

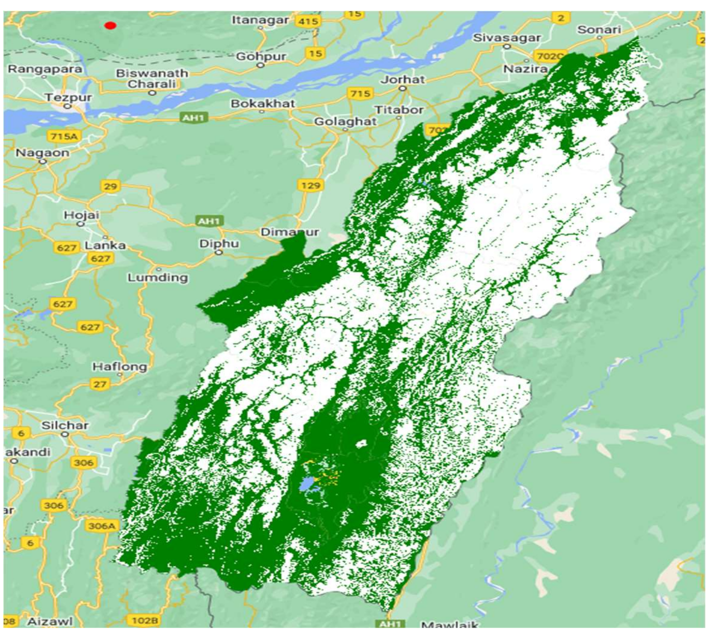


Figure 19b – Potential hotspots detected by CART in Nagaland and Manipur

6 Export Results to Drive

The metrics and evaluation results as well as the generated maps can be exported to google drive as a raster or vector data. Google earth engine allows export to csv as well as TIFF and shapefiles for visualization and further analysis of the file in a GIS environment. The file can be exported to other cloud accounts as well for large images. It also offers exporting data to google one account. For this research we export the data to google drive using the free tier,which allows a combined space of 15 gb. The export code has snippet can be seen in figure 20.

```
Export.table.toDrive({
  collection: ee.FeatureCollection(AUCROCs
                        .map(function(element){
                        return ee.Feature(null,{AUCROC:element})})),
  description: 'AUCROC',
  folder : 'CART Results',
  fileFormat: 'CSV',
});

Export.table.toDrive({
  collection: ee.FeatureCollection(AUCPRs
                        .map(function(element){
                        return ee.Feature(null,{AUCPR:element})})),
  description: 'AUCPR',
  folder : 'CART Results',
  fileFormat: 'CSV',
});

Export.table.toDrive({
  collection: ee.FeatureCollection(Metrics),
  description: 'Metrics',
  folder : 'CART Results',
  fileFormat: 'CSV',
});
```

Figure 20 – Export to drive