Appendix A1: Random Forest

```
import numpy as np
import gdal
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import cohen_kappa_score
import joblib

# define input raster and output raster path
inpRaster = 'Images/2013_cmpst_filledc.tif'
outRaster = 'Classified/RF_GAMA_2013.tif'

# Read Raster Data
ds = gdal.Open(inpRaster)

# Retrieve raster attributes
rows = ds.RasterYSize
cols = ds.RasterXSize
bands = ds.RasterCount
gt = ds.GetGeoTransform()
proj = ds.GetProjection()

# Read Raster as Array
array = ds.ReadAsArray() #(bands, rows, cols)

#modify structure by stacking bands to dorm one element
```

```
array = np.stack(array,axis=2) #(rows, cols, bands)

array = np.reshape(array, [rows*cols,bands]) # reshape to a 2d array so that it can match with the training data

array_df = pd.DataFrame(array, dtype='int16') # convert array to dataframe to keep both test and training data in the same structure


# Read  training data

gdf = gpd.read_file("ground_truth/2013_truth.shp")

class_names = gdf['Label'].unique() # get class names

print ("class names", class_names)

class_ids = np.arange(class_names.size)+1 # assign ids to class names

print('class ids', class_ids)


df = pd.DataFrame({'Label': class_names, 'id': class_ids}) #create a dataframe of the class names and class ids

#df.to_csv("GAMA_2020 data/class_lookup.csv") # save dataframe as csv for future reference

print('gdf without ids', gdf.head())

gdf['class_id'] = gdf['Label'].map(dict(zip(class_names, class_ids))) #add class ids to the shapefile

print('gdf with ids', gdf.head())


# divide truth data data into test and train data

gdf_train = gdf.sample(frac=0.7)

gdf_test = gdf.drop(gdf_train.index)

print('gdf shape', gdf.shape, 'training shape', gdf_train.shape, 'test', gdf_test.shape)

gdf_train.to_file("ground_truth/GAMA_2013_train.shp")

gdf_test.to_file("ground_truth/GAMA_2013_test.shp")

#enter features to use for training according how they are named in the columns of training data

data = gdf_train[['b1_GAMA_13', 'b2_GAMA_13', 'b3_GAMA_13', 'b4_GAMA_13', 'b5_GAMA_13',

        'b6_GAMA_13', 'b7_GAMA_13', 'b8_GAMA_13', 'b9_GAMA_13']]

#enter training label according to your csv column name
```

```
label = gdf_train['class_id']


data_test = gdf_test[['b1_GAMA_13', 'b2_GAMA_13', 'b3_GAMA_13', 'b4_GAMA_13',
'b5_GAMA_13',
        'b6_GAMA_13', 'b7_GAMA_13', 'b8_GAMA_13', 'b9_GAMA_13']]


label_test = gdf_test['class_id']


####no need to modify the code below###
########################################


#set classifier parameters and train classifier
clf = RandomForestClassifier(n_jobs=-1)
clf.fit(data,label)


#predict classes
y_pred = clf.predict(array_df)
classification = y_pred.reshape((rows,cols)) #reshape predicted classes into a 2d array


# Display map
def color_image_show(img, title):
    fig = plt.figure(figsize=(15,15))
    fig.set_facecolor('white')
    plt.imshow(img)
    plt.title(title)
    plt.show()
color_image_show(classification, 'GAMA Random Forest 2013') # display image


# write classified image as a tiff file
def createGeotiff(outRaster, data, geo_transform, projection):
    # Create a GeoTIFF file with the given data
```

```
    driver = gdal.GetDriverByName('GTiff')
    rows, cols = data.shape
    rasterDS = driver.Create(outRaster, cols, rows, 1, gdal.GDT_Int32)
    rasterDS.SetGeoTransform(geo_transform)
    rasterDS.SetProjection(projection)
    band = rasterDS.GetRasterBand(1)
    band.WriteArray(data)
    rasterDS = None

#export classified image
createGeotiff(outRaster,classification,gt,proj)

#Accuracy assessment
clf.score(data_test,label_test) #check performance of classifier on test data
clf.score(data,label)  #check performance of classifier on train data

# Classification report
x_pred= clf.predict(data_test)
print(classification_report(label_test, x_pred, target_names=class_names))

#confusion matrix
cm = confusion_matrix(label_test, x_pred)
pd.DataFrame(cm, index=class_names, columns=class_names)

# Kappa Score
kappa= cohen_kappa_score(label_test, x_pred)
kappa
```

Appendix A2: Support Vector Machine

```
import numpy as np
import gdal
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import cohen_kappa_score
import joblib

# define input raster and output raster path
inpRaster = 'Images/2013_cmpst_filledc.tif'
outRaster = 'Classified/RF_GAMA_2013.tif'

# Read Raster Data
ds = gdal.Open(inpRaster)

# Retrieve raster attributes
rows = ds.RasterYSize
cols = ds.RasterXSize
bands = ds.RasterCount
gt = ds.GetGeoTransform()
proj = ds.GetProjection()

# Read Raster as Array
array = ds.ReadAsArray() #(bands, rows, cols)
```

```
#modify structure by stacking bands to dorm one element

array = np.stack(array,axis=2) #(rows, cols, bands)

array = np.reshape(array, [rows*cols,bands]) # reshape to a 2d array so that it can match with the training data

array_df = pd.DataFrame(array, dtype='int16') # convert array to dataframe to keep both test and training data in the same structure


# Read  training data

gdf = gpd.read_file("ground_truth/2013_truth.shp")

class_names = gdf['Label'].unique() # get class names

print ("class names", class_names)

class_ids = np.arange(class_names.size)+1 # assign ids to class names

print('class ids', class_ids)


df = pd.DataFrame({'Label': class_names, 'id': class_ids}) #create a dataframe of the class names and class ids

#df.to_csv("GAMA_2020 data/class_lookup.csv") # save dataframe as csv for future reference

print('gdf without ids', gdf.head())

gdf['class_id'] = gdf['Label'].map(dict(zip(class_names, class_ids))) #add class ids to the shapefile

print('gdf with ids', gdf.head())


# divide truth data data into test and train data

gdf_train = gdf.sample(frac=0.7)

gdf_test = gdf.drop(gdf_train.index)

print('gdf shape', gdf.shape, 'training shape', gdf_train.shape, 'test', gdf_test.shape)

gdf_train.to_file("ground_truth/GAMA_2013_train.shp")

gdf_test.to_file("ground_truth/GAMA_2013_test.shp")


#enter features to use for training according how they are named in the columns of training data
```

```
data = gdf_train[['b1_GAMA_13', 'b2_GAMA_13', 'b3_GAMA_13', 'b4_GAMA_13', 'b5_GAMA_13',
        'b6_GAMA_13', 'b7_GAMA_13', 'b8_GAMA_13', 'b9_GAMA_13']]


#enter training label according to your csv column name
label = gdf_train['class_id']


data_test = gdf_test[['b1_GAMA_13', 'b2_GAMA_13', 'b3_GAMA_13', 'b4_GAMA_13', 'b5_GAMA_13',
        'b6_GAMA_13', 'b7_GAMA_13', 'b8_GAMA_13', 'b9_GAMA_13']]


label_test = gdf_test['class_id']
####no need to modify the code below###
########################################
#set classifier parameters and train classifier
#set classifier parameters and train classifier


clf = SVC(kernel = 'linear')
clf.fit(data,label)
#predict class
y_pred = clf.predict(array_df)
classification = y_pred.reshape((rows,cols)) #reshape predicted classes into a 2d array


# display image
color_image_show(classification, 'GAMA Support Vector Machine 2022')


#export classified image
createGeotiff(outRaster,classification,gt,proj)


#Accuracy assessment
```

```
clf.score(data_test,label_test) #check performance of classifier on test data
clf.score(data,label)  #check performance of classifier on train data
# classification report
x_pred= clf.predict(data_test)
print(classification_report(label_test, x_pred, target_names=class_names))

#confusion matrix
cm = confusion_matrix(label_test, x_pred)
pd.DataFrame(cm, index=class_names, columns=class_names)

# Kappa Score
kappa= cohen_kappa_score(label_test, x_pred)
kappa
```

Appendix A3: Classification Report

| **RF GAMA 2022** | | | | |
|---|---|---|---|---|
| | Precision | Recall | F1-Score | Support |
| Built-up | 0.93 | 0.95 | 0.94 | 109 |
| Water | 1 | 0.98 | 0.99 | 64 |
| Vegetation | 0.98 | 0.99 | 0.99 | 117 |
| Transition | 0.86 | 0.8 | 0.83 | 45 |
| **Confusion Matrix** | | | | |
| | Built-up | Water | Vegetation | Transition |
| Built-up | 104 | 0 | 0 | 5 |
| Water | 1 | 63 | 0 | 0 |
| Vegetation | 0 | 0 | 116 | 1 |
| Transition | 7 | 0 | 2 | 36 |
| **Overall Accuracy** | **0.916417** | | | |
| **Kappa Score** | **0.933241** | | | |

| **SVM GAMA 2022** | | | | |
|---|---|---|---|---|
| | Precision | Recall | F1-Score | Support |
| Built-up | 0.96 | 0.88 | 0.92 | 104 |
| Water | 1 | 1 | 1 | 68 |
| Vegetation | 1 | 0.99 | 1 | 124 |
| Transition | 0.73 | 0.9 | 0.8 | 39 |
| **Confusion Matrix** | | | | |
| | Built-up | Water | Vegetation | Transition |
| Built-up | 92 | 0 | 0 | 12 |
| Water | 0 | 68 | 0 | 0 |
| Vegetation | 0 | 0 | 123 | 1 |
| Transition | 4 | 0 | 0 | 35 |
| **Overall Accuracy** | **0.94925373** | | | |
| **Kappa Score** | **0.929248506** | | | |