Accuracy Assessment Final

April 23, 2025

[1]:
```
                                                    ###### GITHUB LINK TO COUNTRY␣
 ↪SHAPEFILE ########

##The Github link below provides the zip file which provides the country␣
 ↪shapefile

#https://github.com/NatashaG8/Accuracy-Assessment/raw/main/ParaguayShape.zip
#https://github.com/NatashaG8/Accuracy-Assessment/raw/main/ZambiaShape.zip
#https://github.com/NatashaG8/Accuracy-Assessment/raw/main/ZimbabweShape.zip

#open link and download the shapefile to local device
```

[2]:
```
                                                  ##### CHECKING THE COUNTRY␣
 ↪SHAPEFILE #####
                                                                  ## PARAGUAY ##

import geopandas as gpd

# Define the path to the ZIP archive and the shapefile inside it
shapefile_path = "C:\\Users\\49176\\Downloads\\ParaguayShape.zip"  #use the␣
 ↪path to the zip file you downloaded from the github path provided
shapefile_inside_zip = "/vsizip/{}/ParaguayShape.shp".format(shapefile_path)

# Load the shapefile
aoi = gpd.read_file(shapefile_inside_zip)

# Perform checks
print(f"Number of features: {len(aoi)}")
print("First feature:")
print(aoi.iloc[0])
```

```
Number of features: 1
First feature:
fid                                                    4896.0
iso_a2                                                     PY
NAME                                                 Paraguay
FIPS_10_                                                   PA
ISO_A3                                                    PRY
```

```
WB_A2                                                        PY
WB_A3                                                       PRY
geometry    POLYGON ((-54.35842945927807 -24.7314216957374…
Name: 0, dtype: object
```

[3]:
```
                                                  ##### CHECKING THE COUNTRY␣
 ↪SHAPEFILE #####
                                                                  ## ZAMBIA ##

import geopandas as gpd

# Define the path to the ZIP archive and the shapefile inside it
shapefile_path = "C:\\Users\\49176\\Downloads\\ZambiaShape.zip"  #use the path␣
 ↪to the zip file you downloaded from the github path provided
shapefile_inside_zip = "/vsizip/{}/ZambiaShape.shp".format(shapefile_path)

# Load the shapefile
aoi = gpd.read_file(shapefile_inside_zip)

# Perform checks
print(f"Number of features: {len(aoi)}")
print("First feature:")
print(aoi.iloc[0])
```

```
Number of features: 1
First feature:
fid                                                      280.0
iso_a2                                                      ZM
NAME                                                    Zambia
FIPS_10_                                                    ZA
ISO_A3                                                     ZMB
WB_A2                                                       ZM
WB_A3                                                      ZMB
geometry    POLYGON ((25.25978071856471 -17.7941064876283,…
Name: 0, dtype: object
```

[4]:
```
##### CHECKING THE COUNTRY SHAPEFILE #####
                                                                  ## ZIMBABWE##

import geopandas as gpd

# Define the path to the ZIP archive and the shapefile inside it
shapefile_path = "C:\\Users\\49176\\Downloads\\ZimbabweShape.zip"  #use the␣
 ↪path to the zip file you downloaded from the github path provided
shapefile_inside_zip = "/vsizip/{}/ZimbabweShape.shp".format(shapefile_path)

# Load the shapefile
aoi = gpd.read_file(shapefile_inside_zip)
```

```
# Perform checks
print(f"Number of features: {len(aoi)}")
print("First feature:")
print(aoi.iloc[0])
```

```
Number of features: 1
First feature:
fid                                                      4724.0
iso_a2                                                       ZW
NAME                                                   Zimbabwe
FIPS_10_                                                     ZI
ISO_A3                                                      ZWE
WB_A2                                                        ZW
WB_A3                                                       ZWE
geometry    POLYGON ((32.96908967613309 -17.26611494963315…
Name: 0, dtype: object
```

[5]:
```
                                    ####  RAW DATASET IMAGES BEFORE␣
 ↪RE-CLASSIFICATION ######
                                                        #### PARAGUAY ####

import geopandas as gpd
import ee
import folium
import geemap

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ParaguayShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ParaguayShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
```

```
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Define visualization function with fixes
def visualize_reclassification(dataset_name):
    """Adds dataset layers to the folium map."""
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        raw_image = dynamic_world.select('trees').mean().clip(aoi).
 ↪reproject(crs='EPSG:4326')
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(raw_image, vis_params, "DynamicWorld - Trees")
    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        raw_image = palsar.select('fnf').clip(aoi).reproject(crs='EPSG:4326',␣
 ↪scale=100)
        vis_params = {
            'min': 1,
            'max': 4,
            'palette': ['#00b200', '#83ef62', '#ffff99', '#0000ff']
        }
        map.add_ee_layer(raw_image, vis_params, "PALSAR - Forest/Non-Forest")
    elif dataset_name == "GFW":
```

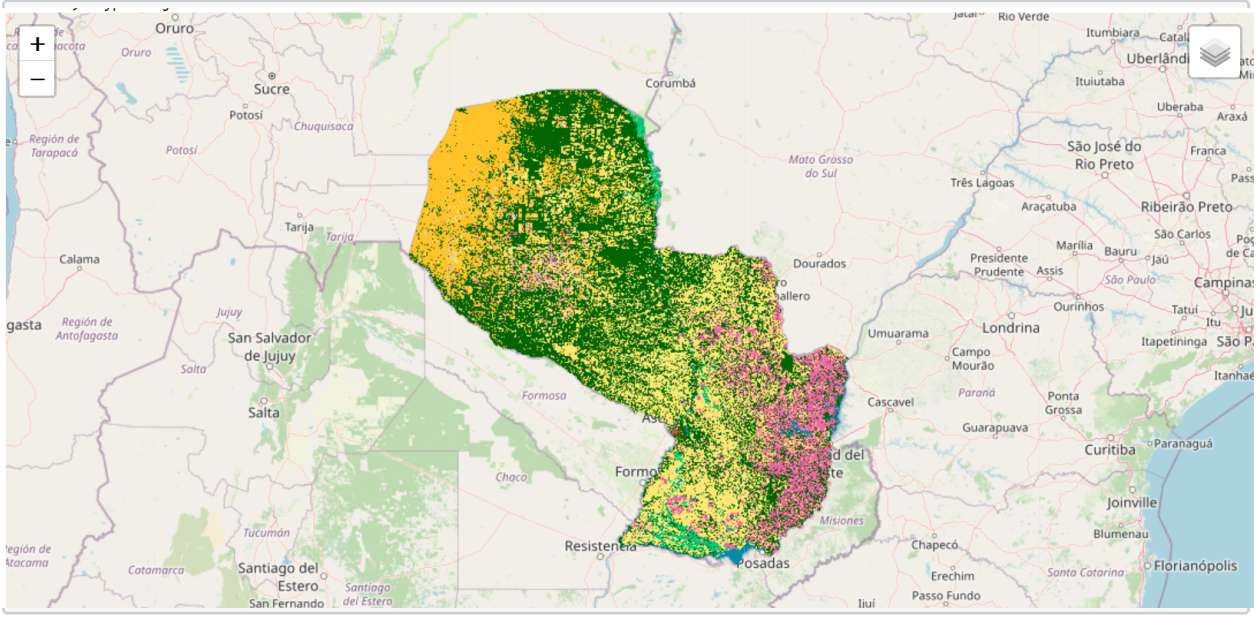
```
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)

        # Create a loss mask for the year 2020 only
        loss_mask_2020 = loss.eq(20)

        # Calculate forest cover after 2020 loss by excluding areas of 2020 loss
        forest_cover_after_loss_2020 = treecover2000.updateMask(loss_mask_2020.
 ↪Not())

        vis_params = {'min': 0, 'max': 100, 'palette': ['white', 'green']}
        map.add_ee_layer(forest_cover_after_loss_2020, vis_params, "GFW -␣
 ↪Forest Cover After 2020 Loss")
    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map').clip(aoi).reproject(crs='EPSG:4326',␣
 ↪scale=100)
        vis_params = {
            'min': 10,
            'max': 100,
            'palette': [
                '#006400', '#ffbb22', '#ffff4c', '#f096ff', '#fa0000',
                '#b4b4b4', '#f0f0f0', '#0064c8', '#0096a0', '#00cf75', '#fae6a0'
            ]
        }
        map.add_ee_layer(esa_image, vis_params, "ESA - WorldCover")

# Visualize datasets
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    visualize_reclassification(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
#map.save("visualized_map_with_loss_2020.html")
map
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                    4896.0
iso_a2                                                     PY
NAME                                                 Paraguay
FIPS_10_                                                   PA
ISO_A3                                                    PRY
WB_A2                                                      PY
```

```
WB_A3                                                        PRY
geometry    POLYGON ((-54.35842945927807 -24.7314216957374…
Name: 0, dtype: object

C:\Users\49176\anaconda3\Lib\site-packages\ee\deprecation.py:207:
DeprecationWarning:

Attention required for UMD/hansen/global_forest_change_2022_v1_10! You are using
a deprecated asset.
To ensure continued functionality, please update it.
Learn more: https://developers.google.com/earth-
engine/datasets/catalog/UMD_hansen_global_forest_change_2022_v1_10

  warnings.warn(warning, category=DeprecationWarning)
```

[5]:
```
<folium.folium.Map at 0x11bc9e43140>
```

[ ]:



[6]:
```
                                        ####  RAW DATASET IMAGES BEFORE␣
 ↪RE-CLASSIFICATION ######

                                                        #### ZAMBIA #####


import geopandas as gpd
import ee
import folium

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
```

```
shapefile_path = "C:\\Users\\49176\\Downloads\\ZambiaShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ZambiaShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Define visualization function with fixes
def visualize_reclassification(dataset_name):
    """Adds dataset layers to the folium map."""
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        raw_image = dynamic_world.select('trees').mean().clip(aoi).
 ↪reproject(crs='EPSG:4326')
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
```

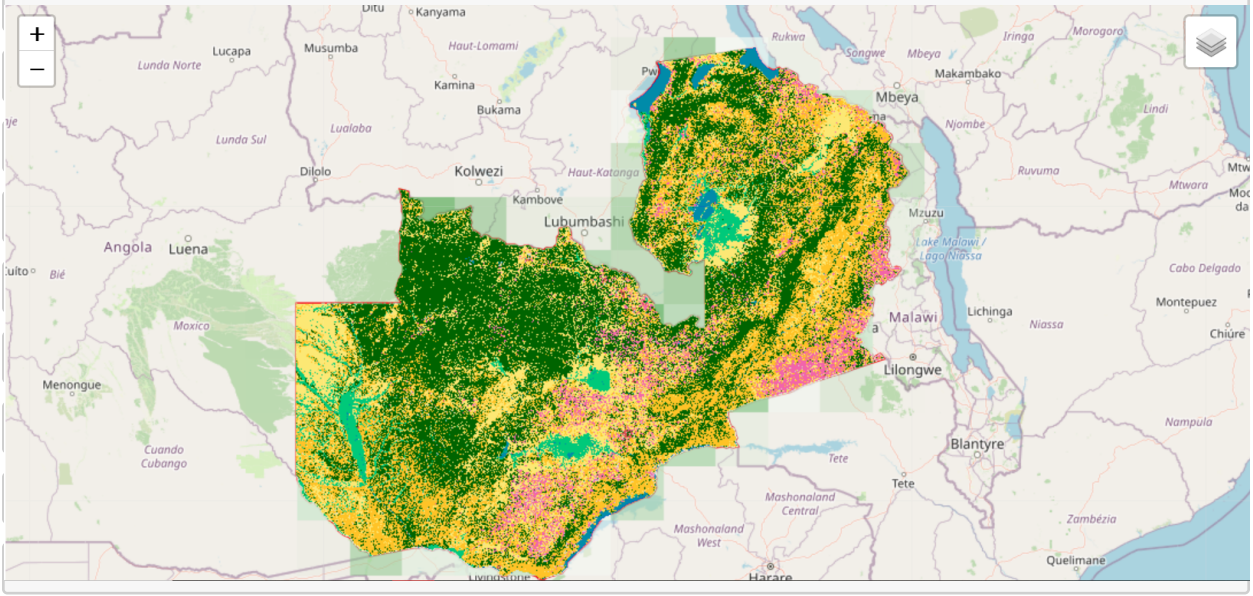
```
        map.add_ee_layer(raw_image, vis_params, "DynamicWorld - Trees")
    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        raw_image = palsar.select('fnf').clip(aoi).reproject(crs='EPSG:4326',␣
↪scale=100)
        vis_params = {
            'min': 1,
            'max': 4,
            'palette': ['#00b200', '#83ef62', '#ffff99', '#0000ff']
        }
        map.add_ee_layer(raw_image, vis_params, "PALSAR - Forest/Non-Forest")
    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)

        # Create a loss mask for the year 2020 only
        loss_mask_2020 = loss.eq(20)

        # Calculate forest cover after 2020 loss by excluding areas of 2020 loss
        forest_cover_after_loss_2020 = treecover2000.updateMask(loss_mask_2020.
↪Not())

        vis_params = {'min': 0, 'max': 100, 'palette': ['white', 'green']}
        map.add_ee_layer(forest_cover_after_loss_2020, vis_params, "GFW -␣
↪Forest Cover After 2020 Loss")
    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map').clip(aoi).reproject(crs='EPSG:4326',␣
↪scale=100)
        vis_params = {
            'min': 10,
            'max': 100,
            'palette': [
                '#006400', '#ffbb22', '#ffff4c', '#f096ff', '#fa0000',
                '#b4b4b4', '#f0f0f0', '#0064c8', '#0096a0', '#00cf75', '#fae6a0'
            ]
        }
        map.add_ee_layer(esa_image, vis_params, "ESA - WorldCover")

# Visualize datasets
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
```

```
    visualize_reclassification(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
#map.save("visualized_map_with_loss_2020.html")
map
```

```
<IPython.core.display.HTML object>

Number of features in AOI: 1
First feature of AOI:
fid                                                      280.0
iso_a2                                                      ZM
NAME                                                    Zambia
FIPS_10_                                                    ZA
ISO_A3                                                     ZMB
WB_A2                                                       ZM
WB_A3                                                      ZMB
geometry    POLYGON ((25.25978071856471 -17.7941064876283,…
Name: 0, dtype: object
```

[6]: <folium.folium.Map at 0x11bccee4ce0>

[ ]:



[7]:
```
                                        ####  RAW DATASET IMAGES BEFORE␣
 ↪RE-CLASSIFICATION ######
                                                          #### ZIMBABWE #####


import geopandas as gpd
import ee
import folium
```

```
# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ZimbabweShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ZimbabweShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Define visualization function with fixes
def visualize_reclassification(dataset_name):
    """Adds dataset layers to the folium map."""
    if dataset_name == "DynamicWorld":
```

```
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        raw_image = dynamic_world.select('trees').mean().clip(aoi).
↪reproject(crs='EPSG:4326')
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(raw_image, vis_params, "DynamicWorld - Trees")
    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        raw_image = palsar.select('fnf').clip(aoi).reproject(crs='EPSG:4326',␣
↪scale=100)
        vis_params = {
            'min': 1,
            'max': 4,
            'palette': ['#00b200', '#83ef62', '#ffff99', '#0000ff']
        }
        map.add_ee_layer(raw_image, vis_params, "PALSAR - Forest/Non-Forest")
    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)

        # Create a loss mask for the year 2020 only
        loss_mask_2020 = loss.eq(20)

        # Calculate forest cover after 2020 loss by excluding areas of 2020 loss
        forest_cover_after_loss_2020 = treecover2000.updateMask(loss_mask_2020.
↪Not())

        vis_params = {'min': 0, 'max': 100, 'palette': ['white', 'green']}
        map.add_ee_layer(forest_cover_after_loss_2020, vis_params, "GFW -␣
↪Forest Cover After 2020 Loss")
    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map').clip(aoi).reproject(crs='EPSG:4326',␣
↪scale=100)
        vis_params = {
            'min': 10,
            'max': 100,
            'palette': [
                '#006400', '#ffbb22', '#ffff4c', '#f096ff', '#fa0000',
                '#b4b4b4', '#f0f0f0', '#0064c8', '#0096a0', '#00cf75', '#fae6a0'
            ]
```

```
        }
        map.add_ee_layer(esa_image, vis_params, "ESA - WorldCover")

# Visualize datasets
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    visualize_reclassification(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
#map.save("visualized_map_with_loss_2020.html")
map
```

```
<IPython.core.display.HTML object>
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                      4724.0
iso_a2                                                       ZW
NAME                                                   Zimbabwe
FIPS_10_                                                     ZI
ISO_A3                                                      ZWE
WB_A2                                                        ZW
WB_A3                                                       ZWE
geometry    POLYGON ((32.96908967613309 -17.26611494963315…
Name: 0, dtype: object
```

[7]: <folium.folium.Map at 0x11bccf16d50>

[ ]:

[ ]:

[ ]:

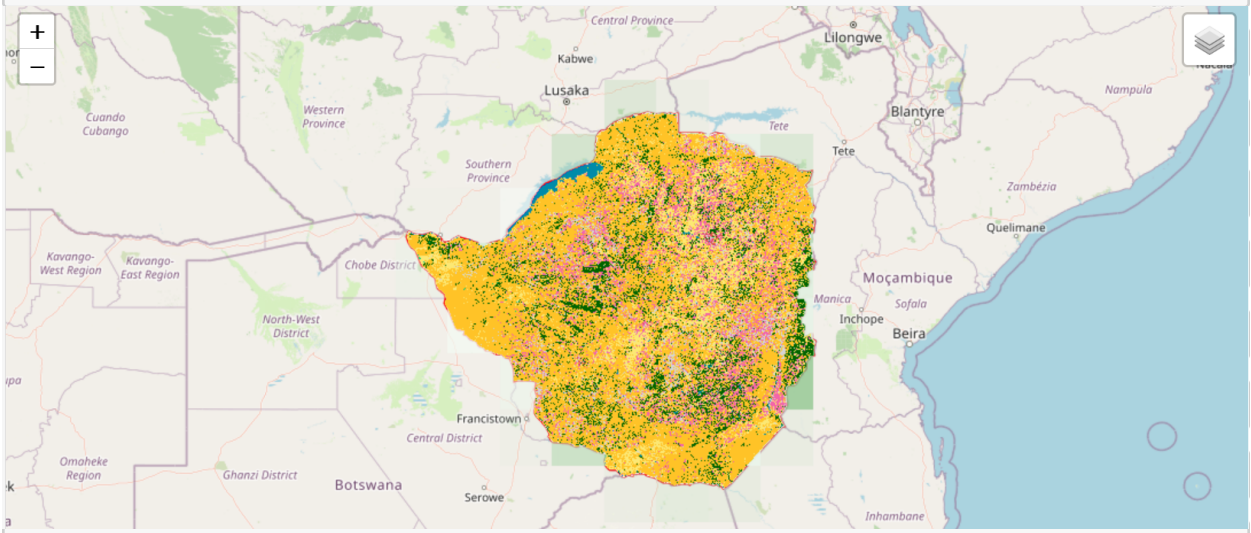
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:



[8]:
```
                                                    ####### RE-CLASSIFICATION OF␣
 ↪IMAGES #########
                                                                ### PARAGUAY #####

import geopandas as gpd
import ee
import folium

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ParaguayShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ParaguayShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer
```

```
# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Reclassify each dataset to forest/non-forest maps
def reclassify_to_forest(dataset_name):
    """Reclassifies dataset to binary forest/non-forest and visualizes on the␣
 ↪map."""
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        forest = dynamic_world.select('trees').mean().clip(aoi).gt(0.3)  #␣
 ↪Threshold for forest
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "DynamicWorld - Forest/Non-Forest")
    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        forest = palsar.select('fnf').clip(aoi).eq(1).Or(palsar.select('fnf').
 ↪eq(2))  # Class 1 and 2 are forests
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "PALSAR - Dense and Non-Dense␣
 ↪Forest")
    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)

        # Create a loss mask for 2020 only
        loss_mask_2020 = loss.eq(20)

        # Exclude 2020 loss from forest cover
        forest_cover_after_loss = treecover2000.updateMask(loss_mask_2020.Not())

        # Reclassify remaining forest (>15% tree cover) and non-forest
        forest = forest_cover_after_loss.gt(10)

        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "GFW - Forest Cover After 2020␣
 ↪Loss")
    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map').clip(aoi)
```

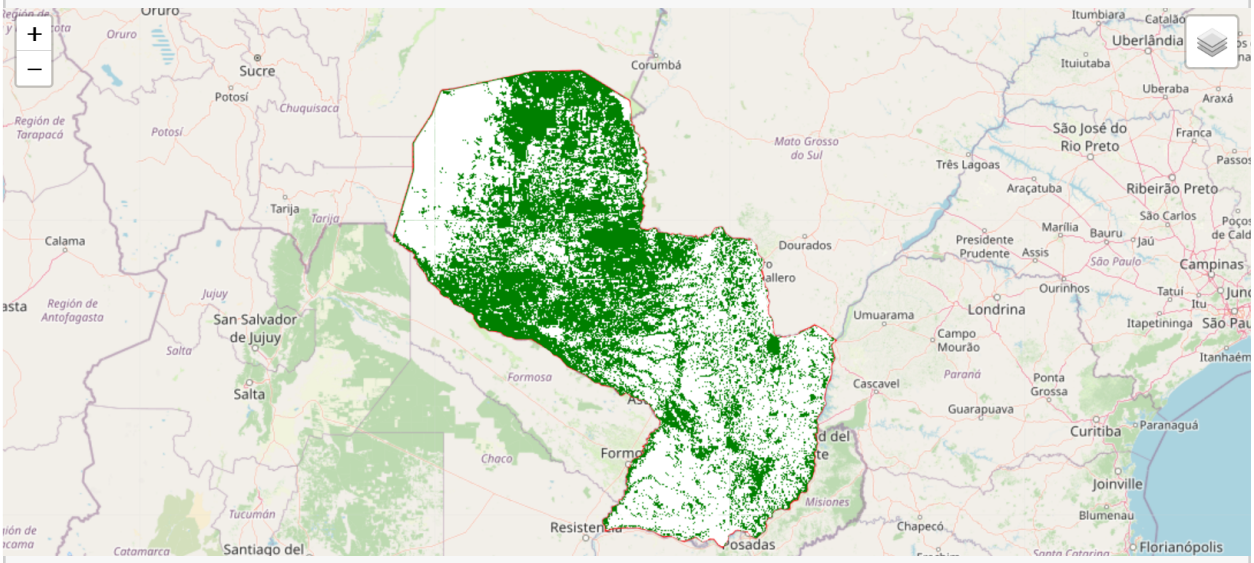
```
        forest = esa_image.eq(10)  # Class 10 represents tree cover in ESA␣
 ↪WorldCover
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "ESA - Forest/Non-Forest")

# Apply reclassification for each dataset
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    reclassify_to_forest(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
#map.save("reclassified_map_with_gfw_2020.html")
map
```

```
<IPython.core.display.HTML object>
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                    4896.0
iso_a2                                                     PY
NAME                                                 Paraguay
FIPS_10_                                                   PA
ISO_A3                                                    PRY
WB_A2                                                      PY
WB_A3                                                     PRY
geometry    POLYGON ((-54.35842945927807 -24.7314216957374…
Name: 0, dtype: object
```

[8]:
```
<folium.folium.Map at 0x11bd015c4a0>
```

[ ]:



[9]:

```
                                                                    #####␣
 ↪RE-CLASSIFICATION OF IMAGES ####
                                                                        #### ZAMBIA␣
 ↪#######

import geopandas as gpd
import ee
import folium
import geemap

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ZambiaShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ZambiaShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
```

```
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Reclassify each dataset to forest/non-forest maps
def reclassify_to_forest(dataset_name):
    """Reclassifies dataset to binary forest/non-forest and visualizes on the␣
 ↪map."""
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        forest = dynamic_world.select('trees').mean().clip(aoi).gt(0.3)  #␣
 ↪Threshold for forest
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "DynamicWorld - Forest/Non-Forest")

    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        forest = palsar.select('fnf').clip(aoi).eq(1).Or(palsar.select('fnf').
 ↪eq(2))  # Class 1 and 2 are forests
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "PALSAR - Dense and Non-Dense␣
 ↪Forest")

    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)

        # Mask areas that experienced loss in 2020
        loss_mask_2020 = loss.eq(20)
        forest_cover_after_loss = treecover2000.updateMask(loss_mask_2020.Not())

        # Reclassify values: below 10 -> 0 (non-forest), above or equal to 10␣
 ↪-> 1 (forest)
        forest = forest_cover_after_loss.gte(10)  # Greater than or equal to 10␣
 ↪becomes 1 (forest)

        # Ensure non-forest areas are explicitly 0
        forest_binary = forest.updateMask(forest).unmask(0)
```

```
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest_binary, vis_params, "GFW - Reclassified Forest/
↪Non-Forest")

    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map').clip(aoi)
        forest = esa_image.eq(10)  # Class 10 represents tree cover in ESA␣
↪WorldCover
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "ESA - Forest/Non-Forest")

# Apply reclassification for each dataset
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    reclassify_to_forest(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
#map.save("reclassified_map_with_gfw_2020.html")
map
```

```
<IPython.core.display.HTML object>

Number of features in AOI: 1
First feature of AOI:
fid                                                      280.0
iso_a2                                                      ZM
NAME                                                    Zambia
FIPS_10_                                                    ZA
ISO_A3                                                     ZMB
WB_A2                                                       ZM
WB_A3                                                      ZMB
geometry    POLYGON ((25.25978071856471 -17.7941064876283,…
Name: 0, dtype: object
```

[9]: <folium.folium.Map at 0x11bc9e42990>

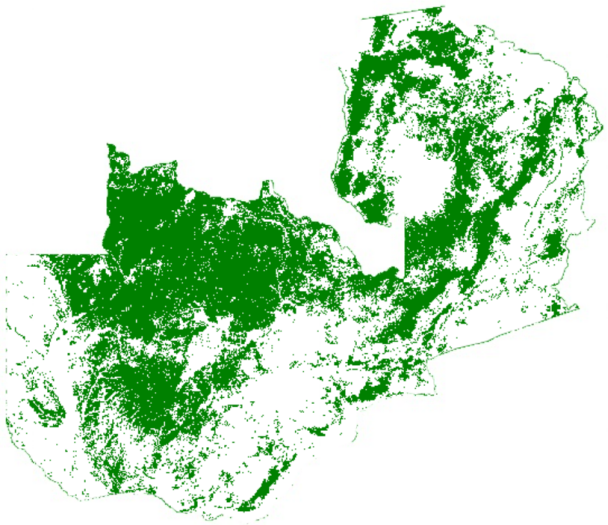
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:



[ ]:

[ ]:

[ ]:

[ ]:

[10]:
```
                                        ####  RAW DATASET IMAGES BEFORE␣
 ↪RE-CLASSIFICATION ######
                                                        #### ZIMBABWE #####



import geopandas as gpd
import ee
import folium
import geemap

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ZimbabweShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ZimbabweShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
```

```
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Reclassify each dataset to forest/non-forest maps
def reclassify_to_forest(dataset_name):
    """Reclassifies dataset to binary forest/non-forest and visualizes on the␣
 ↪map."""
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        forest = dynamic_world.select('trees').mean().clip(aoi).gt(0.3)  #␣
 ↪Threshold for forest
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "DynamicWorld - Forest/Non-Forest")

    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        forest = palsar.select('fnf').clip(aoi).eq(1).Or(palsar.select('fnf').
 ↪eq(2))  # Class 1 and 2 are forests
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "PALSAR - Dense and Non-Dense␣
 ↪Forest")

    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)

        # Mask areas that experienced loss in 2020
        loss_mask_2020 = loss.eq(20)
        forest_cover_after_loss = treecover2000.updateMask(loss_mask_2020.Not())
```

```
        # Reclassify values: below 10 -> 0 (non-forest), above or equal to 10␣
↪-> 1 (forest)
        forest = forest_cover_after_loss.gte(10)  # Greater than or equal to 10␣
↪becomes 1 (forest)

        # Ensure non-forest areas are explicitly 0
        forest_binary = forest.updateMask(forest).unmask(0)

        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest_binary, vis_params, "GFW - Reclassified Forest/
↪Non-Forest")

    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map').clip(aoi)
        forest = esa_image.eq(10)  # Class 10 represents tree cover in ESA␣
↪WorldCover
        vis_params = {'min': 0, 'max': 1, 'palette': ['white', 'green']}
        map.add_ee_layer(forest, vis_params, "ESA - Forest/Non-Forest")

# Apply reclassification for each dataset
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    reclassify_to_forest(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
#map.save("reclassified_map_with_gfw_2020.html")
map
```

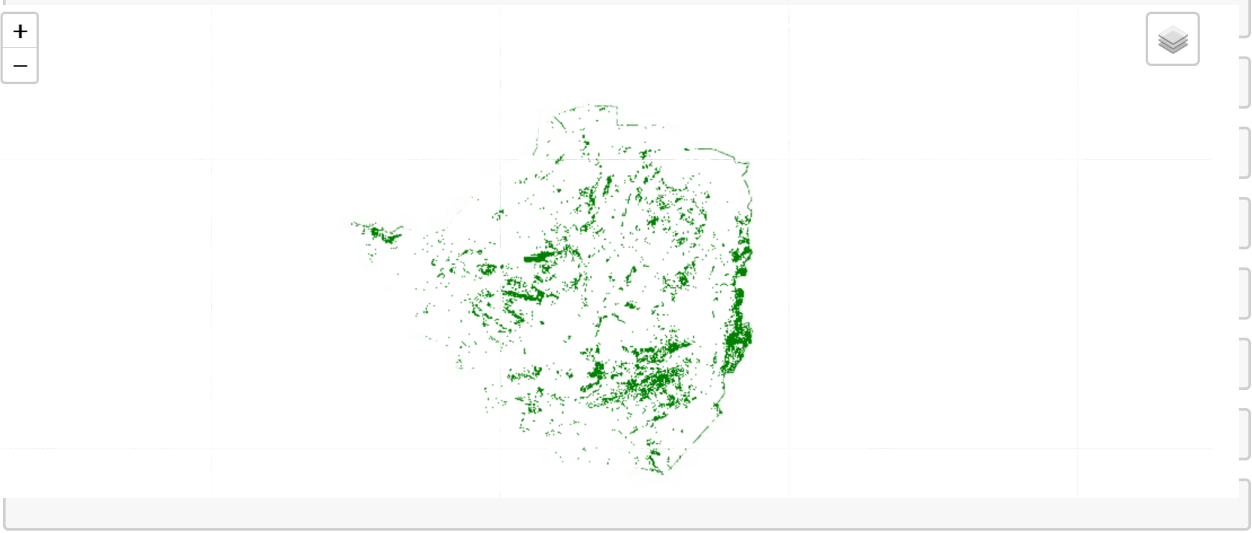
```
<IPython.core.display.HTML object>
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                    4724.0
iso_a2                                                     ZW
NAME                                                 Zimbabwe
FIPS_10_                                                   ZI
ISO_A3                                                    ZWE
WB_A2                                                      ZW
WB_A3                                                     ZWE
geometry    POLYGON ((32.96908967613309 -17.26611494963315…
Name: 0, dtype: object
```

[10]:
```
<folium.folium.Map at 0x11bd045b230>
```

[ ]:

[ ]:



[11]:

```
                                                    #### CALCULATING␣
 ↪FOREST COVER IN HECTARES ####

import geopandas as gpd
import ee
import folium
import geemap

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ParaguayShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ParaguayShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
```

```
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Function to reclassify datasets and calculate forest area
def reclassify_and_calculate_area(dataset_name, scale=500):
    """Reclassifies dataset to binary forest/non-forest, calculates total␣
 ↪forest area, and visualizes on the map."""
    forest = None
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        forest = dynamic_world.select('trees').mean().gt(0.4)  # Threshold for␣
 ↪forest
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
 ↪'green']}, "DynamicWorld Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"DynamicWorld Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
```

```
            .mosaic()
        forest = palsar.select('fnf').eq(1).Or(palsar.select('fnf').eq(2))  #␣
↪Class 1 and 2 are forests
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "PALSAR Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"PALSAR Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)
        loss_mask_2020 = loss.eq(20)
        forest_cover_after_loss = treecover2000.updateMask(loss_mask_2020.Not())
        forest = forest_cover_after_loss.gt(2)
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "GFW Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"GFW Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map')
        forest = esa_image.eq(10)  # Class 10 represents tree cover in ESA␣
↪WorldCover
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "ESA Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
```

```
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"ESA Total Forest Area (ha): {forest_area_ha}")

# Apply reclassification and area calculation for each dataset
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    reclassify_and_calculate_area(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
map.save("fixed_reclassified_map.html")
print("Fixed map saved as fixed_reclassified_map.html")
```

```
<IPython.core.display.HTML object>
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                      4896.0
iso_a2                                                       PY
NAME                                                   Paraguay
FIPS_10_                                                     PA
ISO_A3                                                      PRY
WB_A2                                                        PY
WB_A3                                                       PRY
geometry    POLYGON ((-54.35842945927807 -24.7314216957374…
Name: 0, dtype: object
DynamicWorld Total Forest Area (ha): 18049718.82745839
PALSAR Total Forest Area (ha): 17764185.84840744
GFW Total Forest Area (ha): 20384394.708198767
ESA Total Forest Area (ha): 16807839.178724047
Fixed map saved as fixed_reclassified_map.html
```

[12]:
```
#### CALCULATING FOREST COVER IN HECTARES ####
import ee
import folium
import geemap

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
```

```
shapefile_path = "C:\\Users\\49176\\Downloads\\ZambiaShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ZambiaShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)

# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Function to reclassify datasets and calculate forest area
def reclassify_and_calculate_area(dataset_name, scale=500):
    """Reclassifies dataset to binary forest/non-forest, calculates total␣
 ↪forest area, and visualizes on the map."""
    forest = None
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
```

```
        forest = dynamic_world.select('trees').mean().gt(0.4)  # Threshold for␣
↪forest
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "DynamicWorld Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"DynamicWorld Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        forest = palsar.select('fnf').eq(1).Or(palsar.select('fnf').eq(2))  #␣
↪Class 1 and 2 are forests
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "PALSAR Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"PALSAR Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)
        loss_mask_2020 = loss.eq(20)
        forest_cover_after_loss = treecover2000.updateMask(loss_mask_2020.Not())
        forest = forest_cover_after_loss.gt(2)
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "GFW Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
```

```
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"GFW Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map')
        forest = esa_image.eq(10)  # Class 10 represents tree cover in ESA␣
↪WorldCover
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "ESA Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"ESA Total Forest Area (ha): {forest_area_ha}")

# Apply reclassification and area calculation for each dataset
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    reclassify_and_calculate_area(dataset)

# Add layer control and save the map
folium.LayerControl().add_to(map)
map.save("fixed_reclassified_map.html")
print("Fixed map saved as fixed_reclassified_map.html")
```

```
<IPython.core.display.HTML object>
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                 280.0
iso_a2                                                 ZM
NAME                                               Zambia
FIPS_10_                                               ZA
ISO_A3                                                ZMB
WB_A2                                                  ZM
WB_A3                                                 ZMB
```

```
geometry    POLYGON ((25.25978071856471 -17.7941064876283,…
Name: 0, dtype: object
DynamicWorld Total Forest Area (ha): 30984774.18109145
PALSAR Total Forest Area (ha): 41083928.0337015
GFW Total Forest Area (ha): 33594174.2396203
ESA Total Forest Area (ha): 30432043.19714795
Fixed map saved as fixed_reclassified_map.html
```

[13]:
```python
import ee
import folium
import geemap

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Load the shapefile as AOI
shapefile_path = "C:\\Users\\49176\\Downloads\\ZimbabweShape.zip"
shapefile_inside_zip = f"/vsizip/{shapefile_path}/ZimbabweShape.shp"
aoi_gdf = gpd.read_file(shapefile_inside_zip)
print(f"Number of features in AOI: {len(aoi_gdf)}")
print("First feature of AOI:")
print(aoi_gdf.iloc[0])

# Convert AOI to Earth Engine FeatureCollection
aoi = geemap.gdf_to_ee(aoi_gdf)

# Dynamically compute AOI centroid for map centering
aoi_centroid = aoi.geometry().centroid().coordinates().getInfo()
center = [aoi_centroid[1], aoi_centroid[0]]  # Convert to [latitude, longitude]

# Create a folium map
map = folium.Map(location=center, zoom_start=6)

# Define a function to add Earth Engine layers to folium map
def add_ee_layer(self, ee_object, vis_params, name):
    """Function to add Earth Engine layers to a folium map."""
    if isinstance(ee_object, ee.ImageCollection):
        ee_object = ee_object.mosaic()  # Combine into a single image
    map_id_dict = ee.Image(ee_object).getMapId(vis_params)
    folium.TileLayer(
        tiles=map_id_dict['tile_fetcher'].url_format,
        attr="Map Data &copy; Google Earth Engine",
        name=name,
        overlay=True,
        control=True,
    ).add_to(self)
```

```
# Bind the method to the folium map instance
folium.Map.add_ee_layer = add_ee_layer

# Add AOI boundary to the map
map.add_ee_layer(aoi.style(color="red", width=2), {}, "AOI Boundary")

# Function to reclassify datasets and calculate forest area
def reclassify_and_calculate_area(dataset_name, scale=500):
    """Reclassifies dataset to binary forest/non-forest, calculates total␣
 ↪forest area, and visualizes on the map."""
    forest = None
    if dataset_name == "DynamicWorld":
        dynamic_world = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31')
        forest = dynamic_world.select('trees').mean().gt(0.4)  # Threshold for␣
 ↪forest
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
 ↪'green']}, "DynamicWorld Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"DynamicWorld Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        forest = palsar.select('fnf').eq(1).Or(palsar.select('fnf').eq(2))  #␣
 ↪Class 1 and 2 are forests
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
 ↪'green']}, "PALSAR Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
```

```
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"PALSAR Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "GFW":
        hansen_data = ee.Image('UMD/hansen/global_forest_change_2022_v1_10')
        treecover2000 = hansen_data.select('treecover2000').clip(aoi)
        loss = hansen_data.select('lossyear').clip(aoi)
        loss_mask_2020 = loss.eq(20)
        forest_cover_after_loss = treecover2000.updateMask(loss_mask_2020.Not())
        forest = forest_cover_after_loss.gt(2)
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "GFW Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"GFW Total Forest Area (ha): {forest_area_ha}")

    elif dataset_name == "ESA":
        esa_image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first().select('Map')
        forest = esa_image.eq(10)  # Class 10 represents tree cover in ESA␣
↪WorldCover
        map.add_ee_layer(forest, {'min': 0, 'max': 1, 'palette': ['white',␣
↪'green']}, "ESA Forest")
        # Calculate forest area
        forest_area = ee.Image.pixelArea().updateMask(forest).reduceRegion(
            reducer=ee.Reducer.sum(),
            geometry=aoi.geometry(),
            scale=scale,
            maxPixels=1e13,
            bestEffort=True
        ).getInfo()
        forest_area_ha = forest_area.get('area', 0) / 10000
        print(f"ESA Total Forest Area (ha): {forest_area_ha}")

# Apply reclassification and area calculation for each dataset
datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
for dataset in datasets:
    reclassify_and_calculate_area(dataset)
```

```
# Add layer control and save the map
folium.LayerControl().add_to(map)
map.save("fixed_reclassified_map.html")
print("Fixed map saved as fixed_reclassified_map.html")
```

```
<IPython.core.display.HTML object>
```

```
Number of features in AOI: 1
First feature of AOI:
fid                                                    4724.0
iso_a2                                                     ZW
NAME                                                 Zimbabwe
FIPS_10_                                                   ZI
ISO_A3                                                    ZWE
WB_A2                                                      ZW
WB_A3                                                     ZWE
geometry    POLYGON ((32.96908967613309 -17.26611494963315…
Name: 0, dtype: object
DynamicWorld Total Forest Area (ha): 2497597.6684160465
PALSAR Total Forest Area (ha): 13153283.165624078
GFW Total Forest Area (ha): 9716559.437224543
ESA Total Forest Area (ha): 4621895.446375142
Fixed map saved as fixed_reclassified_map.html
```

[14]:
```
                                        ######CHECKING THE NICFI POINTS ####
                                             #### PARAGUAY #####
import geopandas as gpd

# Define the path to the ZIP file and the shapefile inside it
shapefile_path = "C:\\Users\\49176\\Downloads\\ParaguayNICFIPoints.zip"
shapefile_inside_zip = "/vsizip/{}/ParaguayNICFIPoints.shp".
 ↪format(shapefile_path)

# Load the shapefile
actual_points_NICFI = gpd.read_file(shapefile_inside_zip)

# Perform checks
print(f"Number of features: {len(actual_points_NICFI)}")
print("First feature:")
print(actual_points_NICFI.iloc[0])

# Count occurrences of each class
class_counts = actual_points_NICFI['id'].value_counts()

# Display class distribution with mapping
print("\nClass Mapping: 0 = Non-Forest, 1 = Forest")
print("\nClass Distribution in the Validation Dataset:")
```

```
for cls, count in class_counts.items():
    label = "Non-Forest" if cls == 0 else "Forest"
    print(f"Class {cls} ({label}): {count} points")
```

```
<IPython.core.display.HTML object>
```

```
Number of features: 500
First feature:
id                                                       1
geometry    POINT (-59.99164328026195 -20.20007906081574)
Name: 0, dtype: object

Class Mapping: 0 = Non-Forest, 1 = Forest

Class Distribution in the Validation Dataset:
Class 1 (Forest): 360 points
Class 0 (Non-Forest): 140 points
```

[15]:
```
                                        ###### CHECKING THE NICFI POINTS ####
                                             #### ZAMBIA #####
import geopandas as gpd

# Define the path to the ZIP file and the shapefile inside it
shapefile_path = "C:\\Users\\49176\\Downloads\\Zambia_NICFI_Points.zip"
shapefile_inside_zip = "/vsizip/{}/Points_NICFI_Zambia.shp".
 ↪format(shapefile_path)

# Load the shapefile
actual_points_NICFI = gpd.read_file(shapefile_inside_zip)

# Perform checks
print(f"Number of features: {len(actual_points_NICFI)}")
print("First feature:")
print(actual_points_NICFI.iloc[0])

# Count occurrences of each class
class_counts = actual_points_NICFI['id'].value_counts()

# Display class distribution with mapping
print("\nClass Mapping: 0 = Non-Forest, 1 = Forest")
print("\nClass Distribution in the Validation Dataset:")
for cls, count in class_counts.items():
    label = "Non-Forest" if cls == 0 else "Forest"
    print(f"Class {cls} ({label}): {count} points")
```

```
<IPython.core.display.HTML object>
```

```
Number of features: 500
First feature:
```

```
id                                                       0
geometry    POINT (29.04059101005401 -8.691026309852669)
Name: 0, dtype: object

Class Mapping: 0 = Non-Forest, 1 = Forest

Class Distribution in the Validation Dataset:
Class 0 (Non-Forest): 290 points
Class 1 (Forest): 210 points
```

[16]:

```
###### CHECKING THE NICFI POINTS ####
                #### ZIMBABWE #####
import geopandas as gpd

# Define the path to the ZIP file and the shapefile inside it
shapefile_path = "C:\\Users\\49176\\Downloads\\ZimNICFIPoints.zip"
shapefile_inside_zip = "/vsizip/{}/ZimNICFIPoints.shp".format(shapefile_path)

# Load the shapefile
actual_points_NICFI = gpd.read_file(shapefile_inside_zip)

# Perform checks
print(f"Number of features: {len(actual_points_NICFI)}")
print("First feature:")
print(actual_points_NICFI.iloc[0])

# Count occurrences of each class
class_counts = actual_points_NICFI['id'].value_counts()

# Display class distribution with mapping
print("\nClass Mapping: 0 = Non-Forest, 1 = Forest")
print("\nClass Distribution in the Validation Dataset:")
for cls, count in class_counts.items():
    label = "Non-Forest" if cls == 0 else "Forest"
    print(f"Class {cls} ({label}): {count} points")
```

```
<IPython.core.display.HTML object>

Number of features: 500
First feature:
id                                                       0
geometry    POINT (28.80569050938217 -16.70740115670537)
Name: 0, dtype: object

Class Mapping: 0 = Non-Forest, 1 = Forest

Class Distribution in the Validation Dataset:
Class 1 (Forest): 262 points
Class 0 (Non-Forest): 238 points
```

[17]:
```
import ee
import geopandas as gpd
import geemap
import pandas as pd

# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Function to load data for a given country
def load_country_data(country_name):
    if country_name == "Paraguay":
        aoi_path = "C:\\Users\\49176\\Downloads\\ParaguayShape.zip"
        points_path = "C:\\Users\\49176\\Downloads\\ParaguayNICFIPoints.zip"
        aoi_file = "ParaguayShape.shp"
        points_file = "ParaguayNICFIPoints.shp"
    elif country_name == "Zambia":
        aoi_path = "C:\\Users\\49176\\Downloads\\ZambiaShape.zip"
        points_path = "C:\\Users\\49176\\Downloads\\Zambia_NICFI_Points.zip"
        aoi_file = "ZambiaShape.shp"
        points_file = "Points_NICFI_Zambia.shp"
    elif country_name == "Zimbabwe":
        aoi_path = "C:\\Users\\49176\\Downloads\\ZimbabweShape.zip"
        points_path = "C:\\Users\\49176\\Downloads\\ZimNICFIPoints.zip"
        aoi_file = "ZimbabweShape.shp"
        points_file = "ZimNICFIPoints.shp"
    else:
        raise ValueError("Unsupported country name.")

    aoi_gdf = gpd.read_file(f"/vsizip/{aoi_path}/{aoi_file}")
    points_gdf = gpd.read_file(f"/vsizip/{points_path}/{points_file}")
    return geemap.gdf_to_ee(aoi_gdf), geemap.gdf_to_ee(points_gdf)

# Function to process each dataset
def process_dataset(dataset_name, aoi_fc, actual_points_fc, country_name,␣
 ↪results):
    classification_image = None
    resolution = None

    if dataset_name == "DynamicWorld":
        image = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi_fc) \
            .filterDate('2020-01-01', '2020-12-31') \
            .select(['trees']).mean()
        classification_image = image.gte(0.5).rename('classification')
        resolution = 10
    elif dataset_name == "PALSAR":
```

```
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi_fc) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        classification_image = palsar.expression(
            'fnf == 1 || fnf == 2 ? 1 : 0',
            {'fnf': palsar.select('fnf')}
        ).rename('classification')
        resolution = 25
    elif dataset_name == "GFW":
        image = ee.Image('UMD/hansen/global_forest_change_2022_v1_10').
↪select('treecover2000')
        classification_image = image.gte(10).rename('classification')
        resolution = 30
    elif dataset_name == "ESA":
        image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
            .first()
        classification_image = image.expression(
            'b("Map") == 10 ? 1 : 0',
            {'Map': image.select('Map')}
        ).rename('classification')
        resolution = 10

    samples = classification_image.sampleRegions(
        collection=actual_points_fc,
        properties=['id'],
        scale=resolution
    ).map(lambda f: f.set('classification', ee.Number(f.get('classification')).
↪toInt()))

    cm = samples.errorMatrix('id', 'classification')
    matrix = cm.getInfo()

    def safe_val(r, c):
        try:
            return matrix[r][c]
        except:
            return 0

    TN = safe_val(0, 0)
    FP = safe_val(0, 1)
    FN = safe_val(1, 0)
    TP = safe_val(1, 1)

    precision = TP / (TP + FP) if (TP + FP) > 0 else 0
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0
```

```
    f1 = (2 * precision * recall) / (precision + recall) if (precision +␣
↪recall) > 0 else 0
    accuracy = cm.accuracy().getInfo()
    kappa = cm.kappa().getInfo()

    squared_diff = samples.map(lambda f: f.set(
        'squared_diff', ee.Number(f.get('id')).subtract(f.
↪get('classification')).pow(2)
    ))
    mean_squared = squared_diff.reduceColumns(
        reducer=ee.Reducer.mean(),
        selectors=['squared_diff']
    ).get('mean')
    rmse = ee.Number(mean_squared).sqrt().getInfo()

    tpr = TP / (TP + FN) if (TP + FN) > 0 else 0
    fpr = FP / (FP + TN) if (FP + TN) > 0 else 0
    auc = tpr / (tpr + fpr) if (tpr + fpr) > 0 else 0

    results.append({
        "Country": country_name,
        "Dataset": dataset_name,
        "TP": TP,
        "TN": TN,
        "FP": FP,
        "FN": FN,
        "Precision": precision,
        "Recall": recall,
        "F1 Score": f1,
        "Accuracy": accuracy,
        "Kappa": kappa,
        "RMSE": rmse,
        "AUC": auc
    })

def run_all_countries():
    results = []
    countries = ["Paraguay", "Zambia", "Zimbabwe"]
    datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]

    for country in countries:
        aoi_fc, points_fc = load_country_data(country)
        for dataset in datasets:
            process_dataset(dataset, aoi_fc, points_fc, country, results)

    df = pd.DataFrame(results)
    df = df.sort_values(by=["Country", "Dataset"])
```

```
    print("\nCombined Classification Results:\n")
    for country in df["Country"].unique():
        print("=" * 80)
        print(f"{country}".center(80))
        print("=" * 80)
        print(df[df["Country"] == country].drop(columns="Country").
 ↪to_string(index=False))
        print("\n")

    df.to_csv("combined_classification_results.csv", index=False)
    print("Results saved to 'combined_classification_results.csv'")

# Execute
run_all_countries()
```

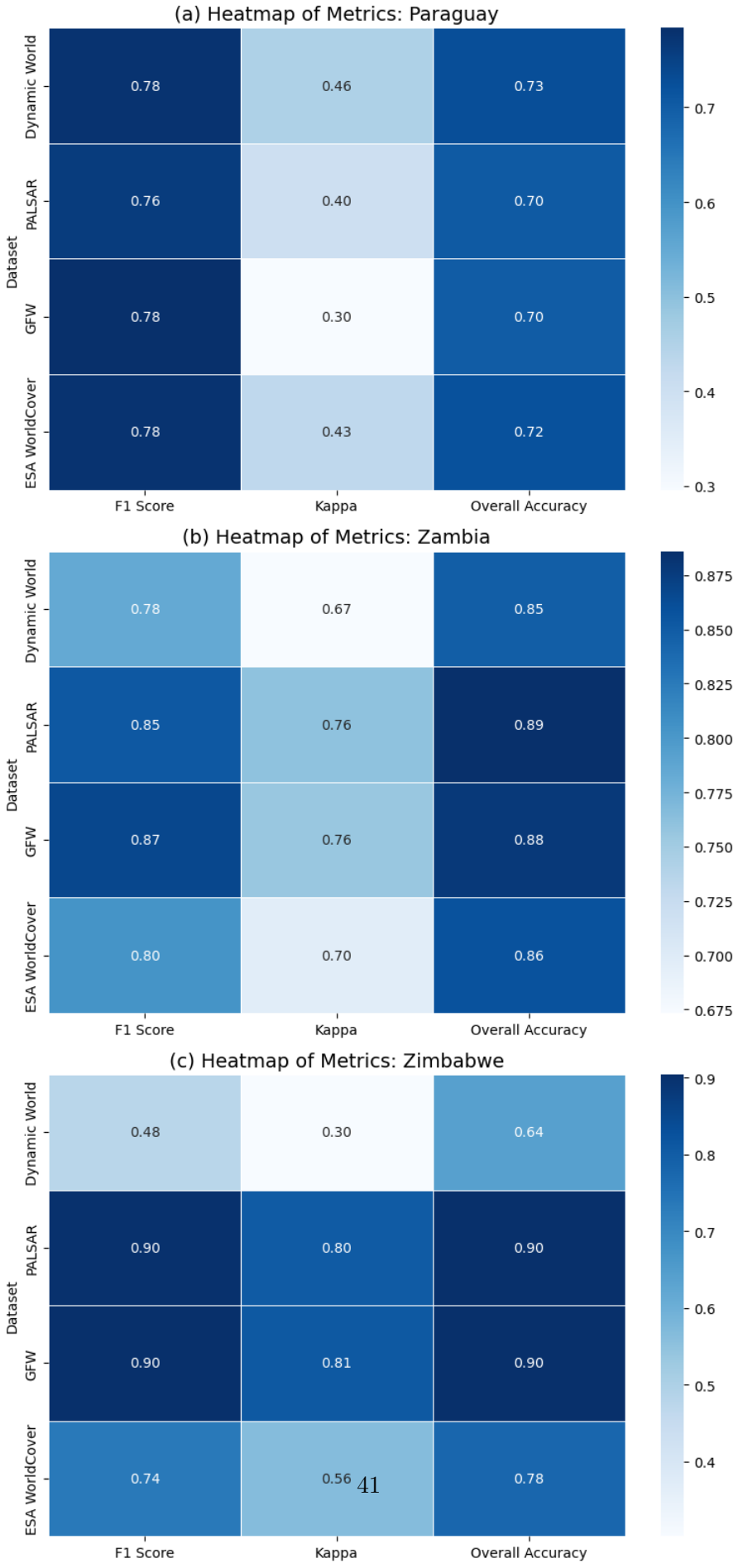
```
<IPython.core.display.HTML object>


Combined Classification Results:

================================================================================
                                    Paraguay
================================================================================
     Dataset  TP  TN  FP  FN  Precision   Recall  F1 Score  Accuracy    Kappa
RMSE      AUC
DynamicWorld 237 127  13 123   0.948000 0.658333  0.777049     0.728 0.456000
0.521536 0.876387
         ESA 243 118  22 117   0.916981 0.675000  0.777600     0.722 0.428924
0.527257 0.811159
         GFW 272  78  62  88   0.814371 0.755556  0.783862     0.700 0.295907
0.547723 0.630464
      PALSAR 234 118  22 126   0.914062 0.650000  0.759740     0.704 0.401682
0.544059 0.805310


================================================================================
                                     Zambia
================================================================================
     Dataset  TP  TN  FP  FN  Precision   Recall  F1 Score  Accuracy    Kappa
RMSE      AUC
DynamicWorld 138 286   4  72   0.971831 0.657143  0.784091     0.848 0.673427
0.389872 0.979442
         ESA 146 283   7  64   0.954248 0.695238  0.804408     0.858 0.697202
0.376829 0.966446
         GFW 199 240  50  11   0.799197 0.947619  0.867102     0.878 0.755844
0.349285 0.846064
      PALSAR 166 277  13  44   0.927374 0.790476  0.853470     0.886 0.761146
```

```
0.337639 0.946334


================================================================================
                                    Zimbabwe
================================================================================
     Dataset  TP  TN  FP  FN  Precision   Recall  F1 Score  Accuracy    Kappa
RMSE      AUC
DynamicWorld  82 238   0 180   1.000000 0.312977  0.476744     0.640 0.302499
0.600000 1.000000
         ESA 155 234   4 107   0.974843 0.591603  0.736342     0.778 0.563624
0.471169 0.972376
         GFW 226 226  12  36   0.949580 0.862595  0.904000     0.904 0.808441
0.309839 0.944776
      PALSAR 219 232   6  43   0.973333 0.835878  0.899384     0.902 0.804936
0.313050 0.970723


Results saved to 'combined_classification_results.csv'
```

[18]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Real metrics from the classification results
data = {
    "Country": ["Paraguay"]*4 + ["Zambia"]*4 + ["Zimbabwe"]*4,
    "Dataset": ["Dynamic World", "PALSAR", "GFW", "ESA WorldCover"]*3,
    "F1 Score": [
        0.777049, 0.759740, 0.783862, 0.777600,  # Paraguay
        0.784091, 0.853470, 0.867102, 0.804408,  # Zambia
        0.476744, 0.899384, 0.904000, 0.736342   # Zimbabwe
    ],
    "Kappa": [
        0.456000, 0.401682, 0.295907, 0.428924,  # Paraguay
        0.673427, 0.761146, 0.755844, 0.697202,  # Zambia
        0.302499, 0.804936, 0.808441, 0.563624   # Zimbabwe
    ],
    "Overall Accuracy": [
        0.728, 0.704, 0.700, 0.722,  # Paraguay
        0.848, 0.886, 0.878, 0.858,  # Zambia
        0.640, 0.902, 0.904, 0.778   # Zimbabwe
    ]
}

# Create DataFrame
df = pd.DataFrame(data)
```

```
# Create a plot for each country
fig, axes = plt.subplots(3, 1, figsize=(8, 16))
countries = ["Paraguay", "Zambia", "Zimbabwe"]
titles = ["(a) Heatmap of Metrics: Paraguay", "(b) Heatmap of Metrics: Zambia",␣
 ↪"(c) Heatmap of Metrics: Zimbabwe"]

for i, country in enumerate(countries):
    ax = axes[i]
    country_df = df[df["Country"] == country].set_index("Dataset")[["F1 Score",␣
 ↪"Kappa", "Overall Accuracy"]]

    sns.heatmap(
        country_df,
        annot=True,
        fmt=".2f",
        cmap="Blues",
        linewidths=0.5,
        cbar=True,
        ax=ax
    )
    ax.set_title(titles[i], fontsize=14)
    ax.set_xlabel("")
    ax.set_ylabel("Dataset")

plt.tight_layout()
plt.show()
```

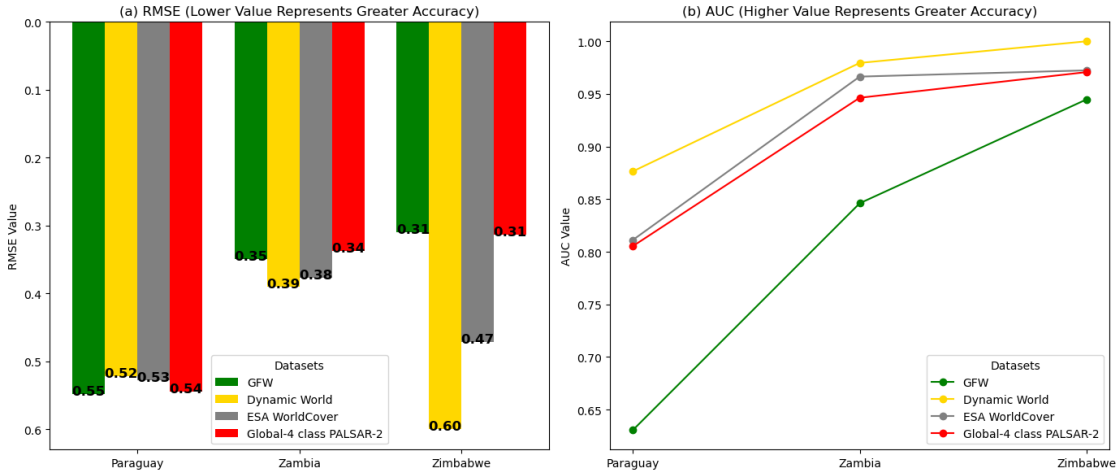
```
<IPython.core.display.HTML object>
```


(a) Heatmap of Metrics: Paraguay
Dataset
Dynamic World
PALSAR
GFW
ESA WorldCover
0.78
0.46
0.73
0.76
0.40
0.70
0.78
0.30
0.70
0.78
0.43
0.72
F1 Score
Kappa
Overall Accuracy
0.7
0.6
0.5
0.4
0.3
(b) Heatmap of Metrics: Zambia
Dataset
Dynamic World
PALSAR
GFW
ESA WorldCover
0.78
0.67
0.85
0.85
0.76
0.89
0.87
0.76
0.88
0.80
0.70
0.86
F1 Score
Kappa
Overall Accuracy
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.675
(c) Heatmap of Metrics: Zimbabwe
Dataset
Dynamic World
PALSAR
GFW
ESA WorldCover
0.48
0.30
0.64
0.90
0.80
0.90
0.90
0.81
0.90
0.74
0.56
0.78
F1 Score
Kappa
Overall Accuracy
0.9
0.8
0.7
0.6
0.5
0.4

[19]:
```
import matplotlib.pyplot as plt
import numpy as np

# Dataset labels and countries
datasets = ['GFW', 'Dynamic World', 'ESA WorldCover', 'Global-4 class PALSAR-2']
countries = ['Paraguay', 'Zambia', 'Zimbabwe']

# Updated RMSE values
rmse_data = {
    'GFW': [0.547723, 0.349285, 0.309839],
    'Dynamic World': [0.521536, 0.389872, 0.600000],
    'ESA WorldCover': [0.527257, 0.376829, 0.471169],
    'Global-4 class PALSAR-2': [0.544059, 0.337639, 0.313050]
}

# Updated AUC values
auc_data = {
    'GFW': [0.630464, 0.846064, 0.944776],
    'Dynamic World': [0.876387, 0.979442, 1.000000],
    'ESA WorldCover': [0.811159, 0.966446, 0.972376],
    'Global-4 class PALSAR-2': [0.805310, 0.946334, 0.970723]
}

# Set up the figure and axes
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# Colors
colors = {
    'GFW': 'green',
    'Dynamic World': 'gold',
    'ESA WorldCover': 'gray',
    'Global-4 class PALSAR-2': 'red'
}

# Bar plot for RMSE
bar_width = 0.2
x = np.arange(len(countries))

for i, dataset in enumerate(datasets):
    values = rmse_data[dataset]
    bars = axs[0].bar(x + i * bar_width, values, width=bar_width,␣
 ↪label=dataset, color=colors[dataset])
    for j, val in enumerate(values):
        axs[0].text(x[j] + i * bar_width, val + 0.005, f"{val:.2f}",
```

```
                    ha='center', va='bottom', fontsize=12, fontweight='bold',␣
 ↪color='black')  # <- Enlarged and bold

axs[0].set_title('(a) RMSE (Lower Value Represents Greater Accuracy)')
axs[0].set_ylabel('RMSE Value')
axs[0].set_xticks(x + bar_width * 1.5)
axs[0].set_xticklabels(countries)
axs[0].legend(title='Datasets')
axs[0].invert_yaxis()

# Line plot for AUC
for dataset in datasets:
    axs[1].plot(countries, auc_data[dataset], marker='o', label=dataset,␣
 ↪color=colors[dataset])

axs[1].set_title('(b) AUC (Higher Value Represents Greater Accuracy)')
axs[1].set_ylabel('AUC Value')
axs[1].legend(title='Datasets')

plt.tight_layout()
plt.show()
```

```
<IPython.core.display.HTML object>
```



[20]:
```
                    # Example if doing accuracy assessment for one country e.g␣
 ↪Zambia ##

import ee
import geopandas as gpd
import geemap
import pandas as pd
```

```
# Authenticate and Initialize Earth Engine
ee.Authenticate()
ee.Initialize()

# Function to load data for one country (Zambia in this example)
def load_country_data():
    aoi_path = "C:\\Users\\49176\\Downloads\\ZambiaShape.zip"
    points_path = "C:\\Users\\49176\\Downloads\\Zambia_NICFI_Points.zip"
    aoi_file = "ZambiaShape.shp"
    points_file = "Points_NICFI_Zambia.shp"

    aoi_gdf = gpd.read_file(f"/vsizip/{aoi_path}/{aoi_file}")
    points_gdf = gpd.read_file(f"/vsizip/{points_path}/{points_file}")
    return geemap.gdf_to_ee(aoi_gdf), geemap.gdf_to_ee(points_gdf)

# Function to process each dataset
def process_dataset(dataset_name, aoi_fc, actual_points_fc, country_name,␣
 ↪results):
    classification_image = None
    resolution = None

    if dataset_name == "DynamicWorld":
        image = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1') \
            .filterBounds(aoi_fc) \
            .filterDate('2020-01-01', '2020-12-31') \
            .select(['trees']).mean()
        classification_image = image.gte(0.5).rename('classification')
        resolution = 10
    elif dataset_name == "PALSAR":
        palsar = ee.ImageCollection("JAXA/ALOS/PALSAR/YEARLY/FNF4") \
            .filterBounds(aoi_fc) \
            .filterDate('2020-01-01', '2020-12-31') \
            .mosaic()
        classification_image = palsar.expression(
            'fnf == 1 || fnf == 2 ? 1 : 0',
            {'fnf': palsar.select('fnf')}
        ).rename('classification')
        resolution = 25
    elif dataset_name == "GFW":
        image = ee.Image('UMD/hansen/global_forest_change_2022_v1_10').
 ↪select('treecover2000')
        classification_image = image.gte(10).rename('classification')
        resolution = 30
    elif dataset_name == "ESA":
        image = ee.ImageCollection('ESA/WorldCover/v100') \
            .filterDate('2020-01-01', '2020-12-31') \
```

```
            .first()
        classification_image = image.expression(
            'b("Map") == 10 ? 1 : 0',
            {'Map': image.select('Map')}
        ).rename('classification')
        resolution = 10

    # Sample points and calculate confusion matrix
    samples = classification_image.sampleRegions(
        collection=actual_points_fc,
        properties=['id'],
        scale=resolution
    ).map(lambda f: f.set('classification', ee.Number(f.get('classification')).
↪toInt()))

    cm = samples.errorMatrix('id', 'classification')
    matrix = cm.getInfo()

    def safe_val(r, c):
        try:
            return matrix[r][c]
        except:
            return 0

    TN = safe_val(0, 0)
    FP = safe_val(0, 1)
    FN = safe_val(1, 0)
    TP = safe_val(1, 1)

    precision = TP / (TP + FP) if (TP + FP) > 0 else 0
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0
    f1 = (2 * precision * recall) / (precision + recall) if (precision +␣
↪recall) > 0 else 0
    accuracy = cm.accuracy().getInfo()
    kappa = cm.kappa().getInfo()

    # RMSE
    squared_diff = samples.map(lambda f: f.set(
        'squared_diff', ee.Number(f.get('id')).subtract(f.
↪get('classification')).pow(2)
    ))
    mean_squared = squared_diff.reduceColumns(
        reducer=ee.Reducer.mean(),
        selectors=['squared_diff']
    ).get('mean')
    rmse = ee.Number(mean_squared).sqrt().getInfo()
```

```
    # AUC
    tpr = TP / (TP + FN) if (TP + FN) > 0 else 0
    fpr = FP / (FP + TN) if (FP + TN) > 0 else 0
    auc = tpr / (tpr + fpr) if (tpr + fpr) > 0 else 0

    results.append({
        "Country": country_name,
        "Dataset": dataset_name,
        "TP": TP,
        "TN": TN,
        "FP": FP,
        "FN": FN,
        "Precision": precision,
        "Recall": recall,
        "F1 Score": f1,
        "Accuracy": accuracy,
        "Kappa": kappa,
        "RMSE": rmse,
        "AUC": auc
    })

# Run for Zambia only
def run_zambia():
    country = "Zambia"
    datasets = ["DynamicWorld", "PALSAR", "GFW", "ESA"]
    results = []

    aoi_fc, points_fc = load_country_data()
    for dataset in datasets:
        process_dataset(dataset, aoi_fc, points_fc, country, results)

    df = pd.DataFrame(results)
    print("\nZambia Classification Results:")
    print(df.to_string(index=False))
    df.to_csv("zambia_classification_results.csv", index=False)
    print("\nSaved to 'zambia_classification_results.csv'")

# Execute for Zambia
run_zambia()
```

```
<IPython.core.display.HTML object>


Zambia Classification Results:
Country      Dataset  TP  TN  FP  FN  Precision   Recall  F1 Score  Accuracy
Kappa     RMSE      AUC
 Zambia DynamicWorld 138 286   4  72   0.971831 0.657143  0.784091     0.848
0.673427 0.389872 0.979442
```

```
 Zambia       PALSAR 166 277  13  44   0.927374 0.790476  0.853470     0.886
0.761146 0.337639 0.946334
 Zambia          GFW 199 240  50  11   0.799197 0.947619  0.867102     0.878
0.755844 0.349285 0.846064
 Zambia          ESA 146 283   7  64   0.954248 0.695238  0.804408     0.858
0.697202 0.376829 0.966446

Saved to 'zambia_classification_results.csv'
```