Biodiversity_GEE_Python_API_NDVI

September 4, 2023

[ ]:
```
# Install necessary libraries
!pip install earthengine-api
```

[ ]:
```
!pip install geemap
!pip install rasterio
```

[ ]:
```
# Import necessary libraries
import ee
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import geemap
import ee
from PIL import Image, ImageDraw, ImageFont
import imageio
import os
from IPython.display import Image, display
from osgeo import gdal
import rasterio
```

[ ]:
```
# Authenticate and initialize the Earth Engine API
ee.Authenticate()
ee.Initialize()
```

[ ]:
```
# Define the region of interest: Sundarbans Mangroves
sundarbans = ee.Geometry.Rectangle([88.0, 21.5, 90.0, 22.5])

# Define the time range
start_year = 2000
end_year = 2023
```

[ ]:
```
# Function to get MODIS NDVI data for a specific year
def get_modis_ndvi(year):
    start_date = f"{year}-01-01"
    end_date = f"{year}-12-31"

    # Use MOD13A2.006 product for NDVI
```

```
    dataset = ee.ImageCollection('MODIS/006/MOD13A2').filterDate(start_date,␣
 ↪end_date).filterBounds(sundarbans)

    # Extract NDVI and scale it. MODIS NDVI values are scaled by a factor of 0.
 ↪0001
    ndvi = dataset.select('NDVI').mean().multiply(0.0001)

    return ndvi
```

[ ]:
```
# Extract yearly NDVI values
yearly_ndvi = {}
for year in range(start_year, end_year+1):
    ndvi_image = get_modis_ndvi(year)
    ndvi_value = ndvi_image.reduceRegion(reducer=ee.Reducer.mean(),
                                         geometry=sundarbans, scale=1000,
                                         maxPixels=1e13).get('NDVI').getInfo()
    yearly_ndvi[year] = ndvi_value
```

[ ]:
```
# Convert to DataFrame for analysis
df = pd.DataFrame(list(yearly_ndvi.items()), columns=['Year', 'NDVI'])

# Export to CSV
df.to_csv('modis_ndvi_2000_2023.csv', index=False)
```

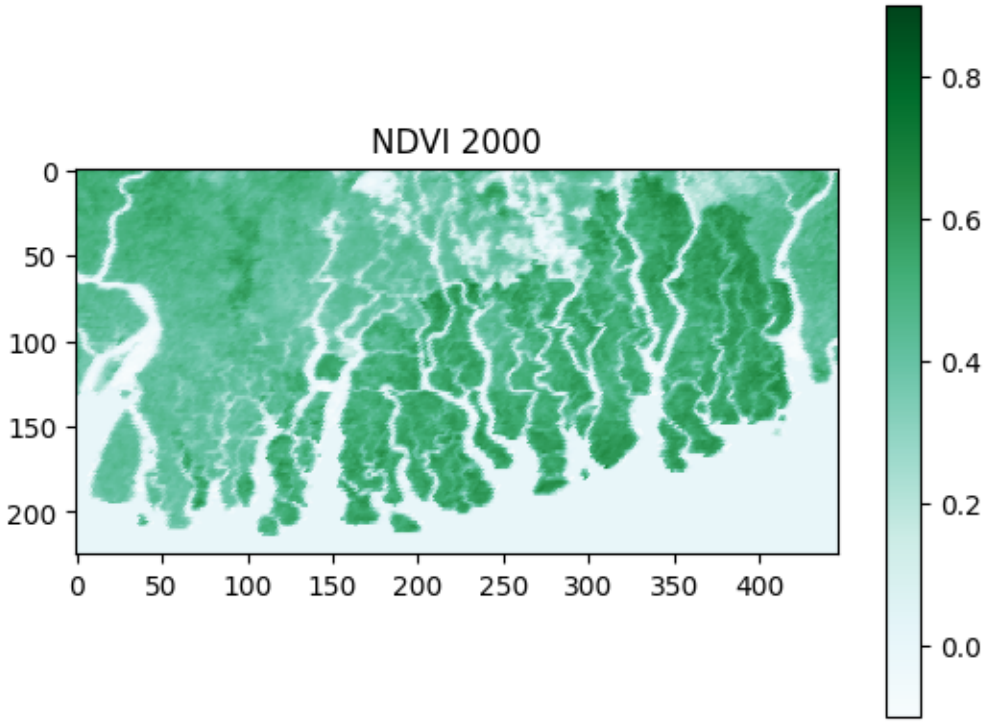
VISUALIZING THE NDVI VALUES ON A MAP

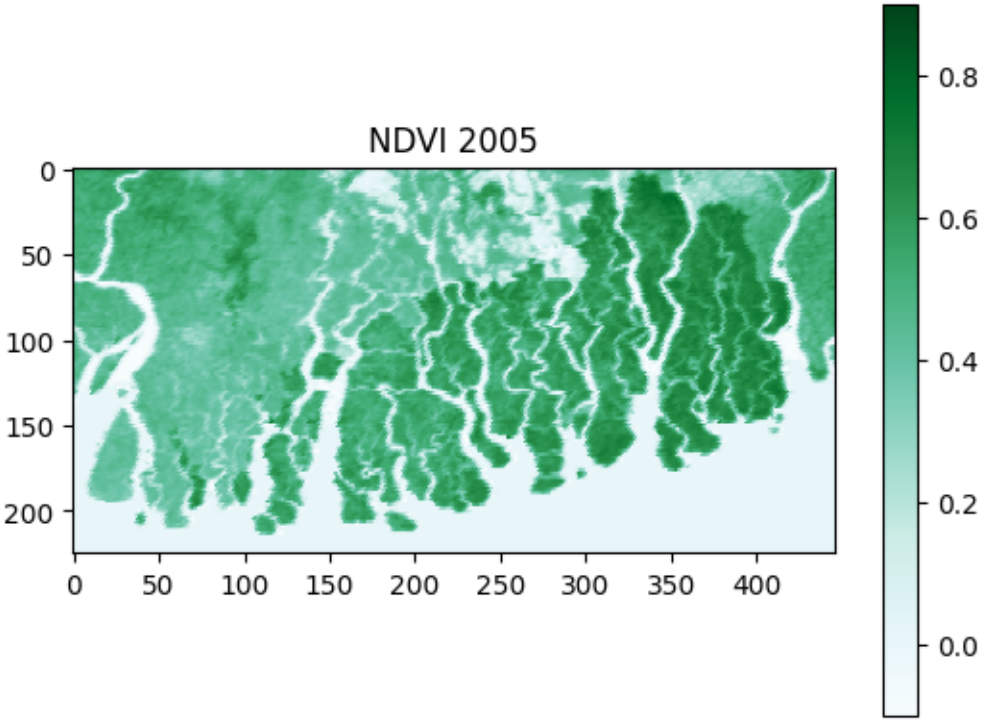
[ ]:
```
# Initialize the Earth Engine module
ee.Initialize()

# Define the region of interest: Sundarbans Mangroves
sundarbans = ee.Geometry.Rectangle([88.0, 21.5, 90.0, 22.5])

# Function to get MODIS NDVI data for a specific year
def get_modis_ndvi(year):
    start_date = f"{year}-01-01"
    end_date = f"{year}-12-31"

    # Use MOD13A2.006 product for NDVI
    dataset = ee.ImageCollection('MODIS/006/MOD13A2').filterDate(start_date,␣
 ↪end_date).filterBounds(sundarbans)

    # Extract NDVI and scale it. MODIS NDVI values are scaled by a factor of 0.
 ↪0001
    ndvi = dataset.select('NDVI').mean().multiply(0.0001)

    return ndvi
```

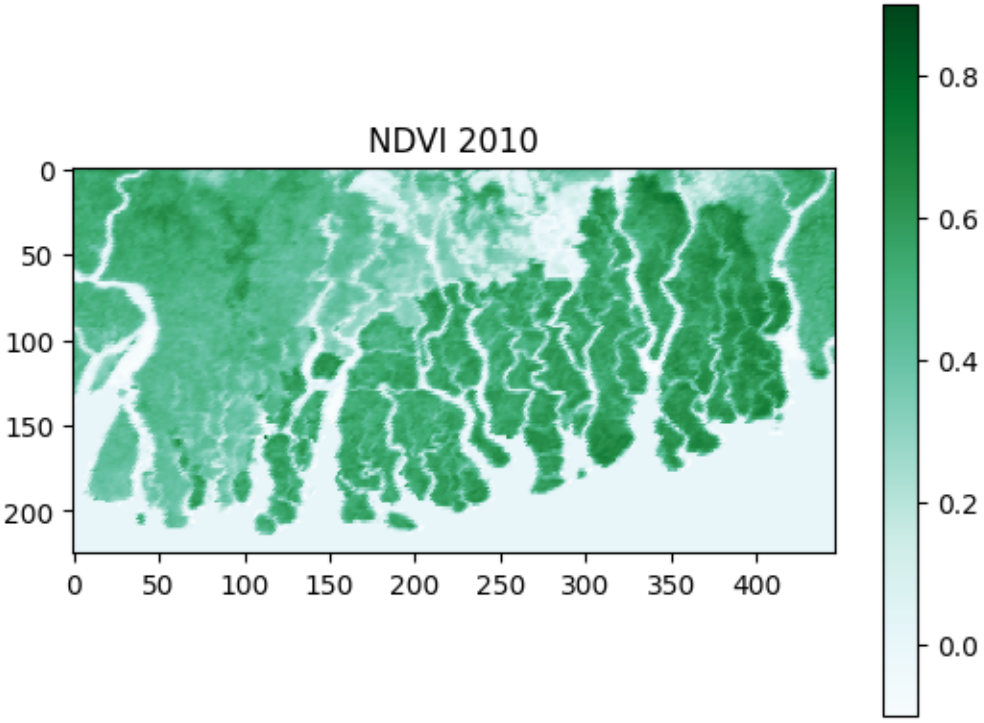
```
# Export each yearly average NDVI image
image_list = []
for year in range(2000, 2024):
    ndvi_image = get_modis_ndvi(year)
    out_path = f'/content/NDVI_{year}.tif'
    geemap.ee_export_image(ndvi_image, filename=out_path, scale=500,␣
 ↪region=sundarbans, file_per_band=False)
    image_list.append(out_path)
```

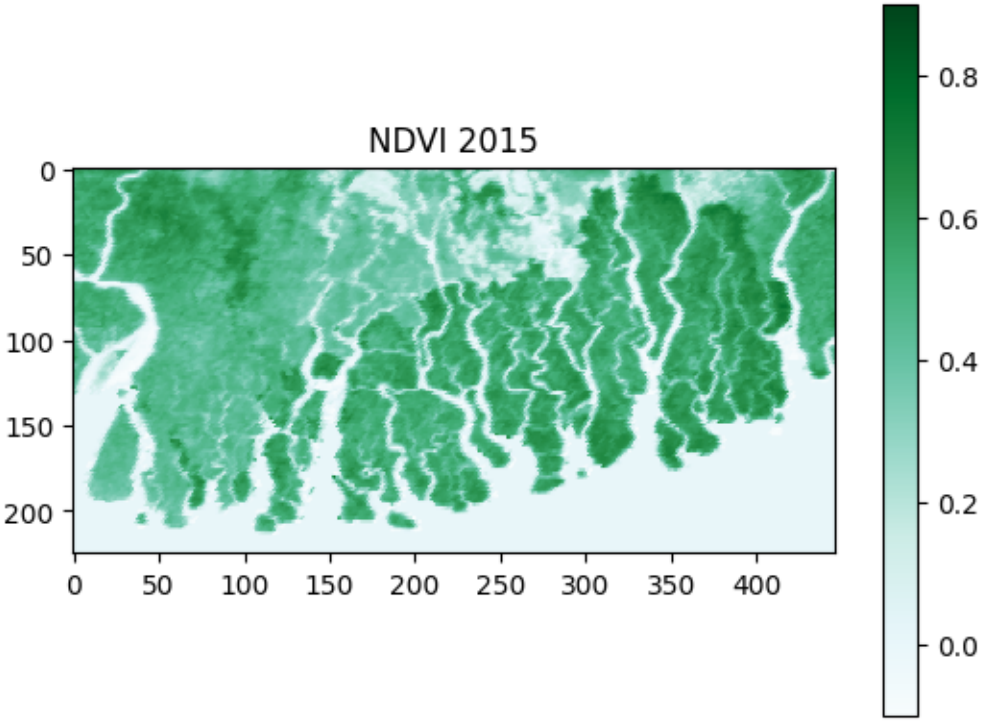
Visualizing a few images

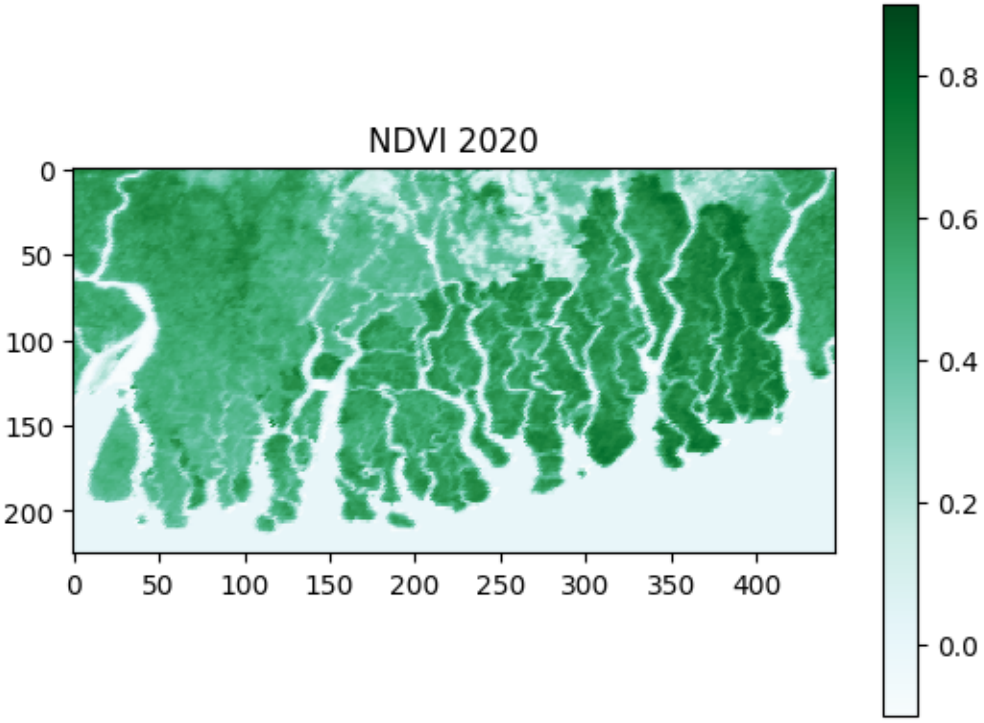
[ ]:
```
# Visualize a few of the downloaded images
for year in [2000, 2005, 2010, 2015, 2020]:
    img_path = f'/content/NDVI_{year}.tif'
    ds = gdal.Open(img_path)
    band = ds.GetRasterBand(1)
    arr = band.ReadAsArray()
    plt.imshow(arr, cmap='BuGn', vmin=-0.1, vmax=0.9)
    plt.title(f'NDVI {year}')
    plt.colorbar()
    plt.show()
```




NDVI 2005
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

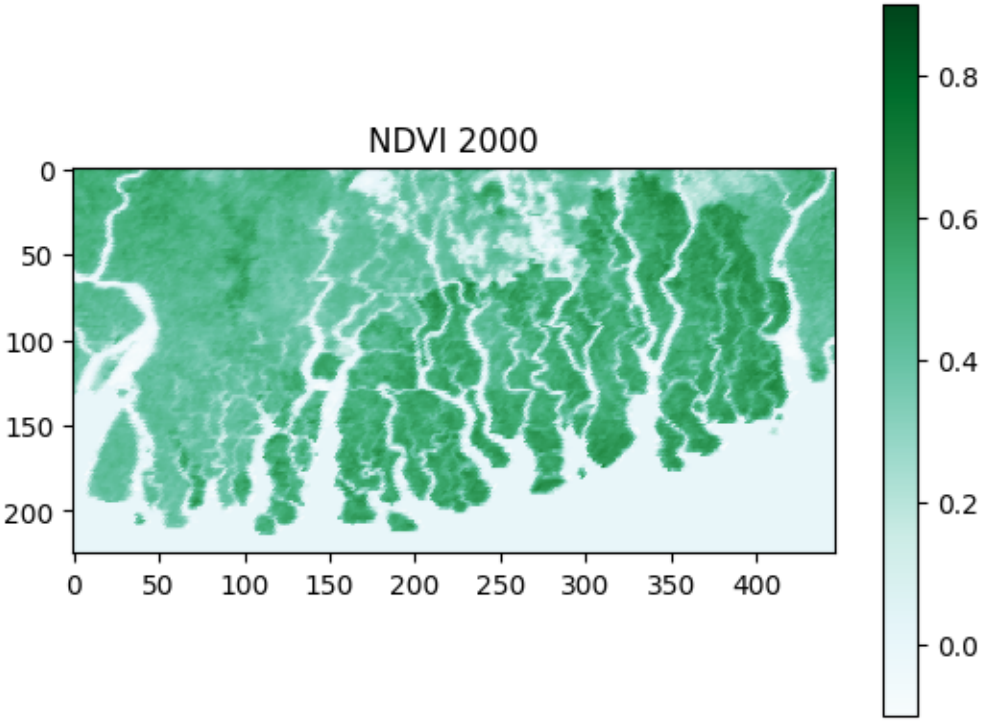

NDVI 2010
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

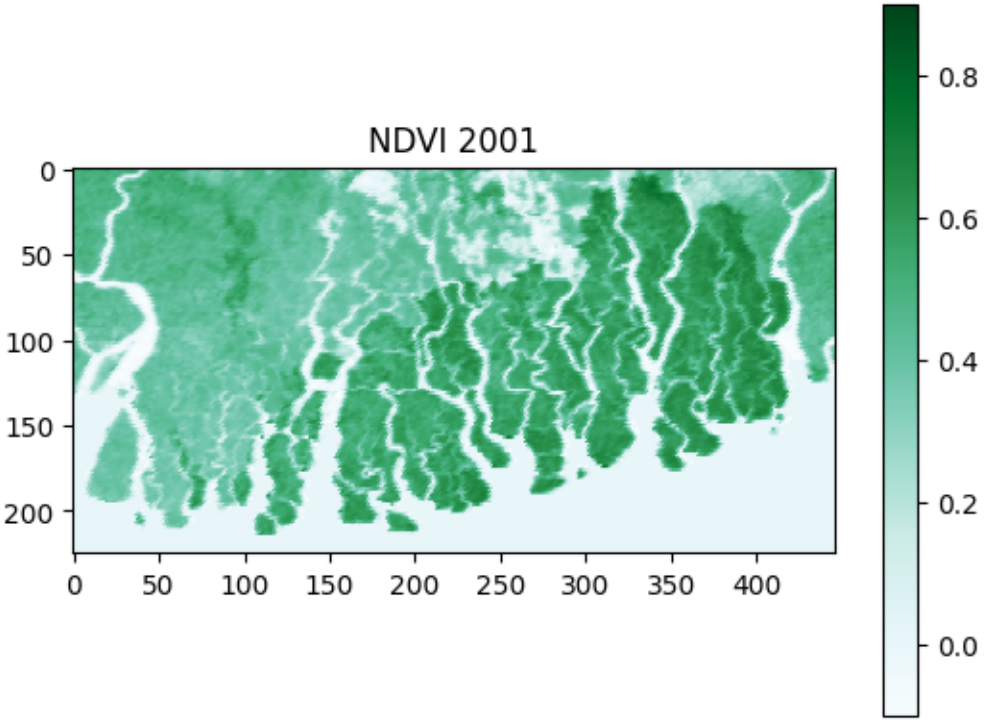

NDVI 2015
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

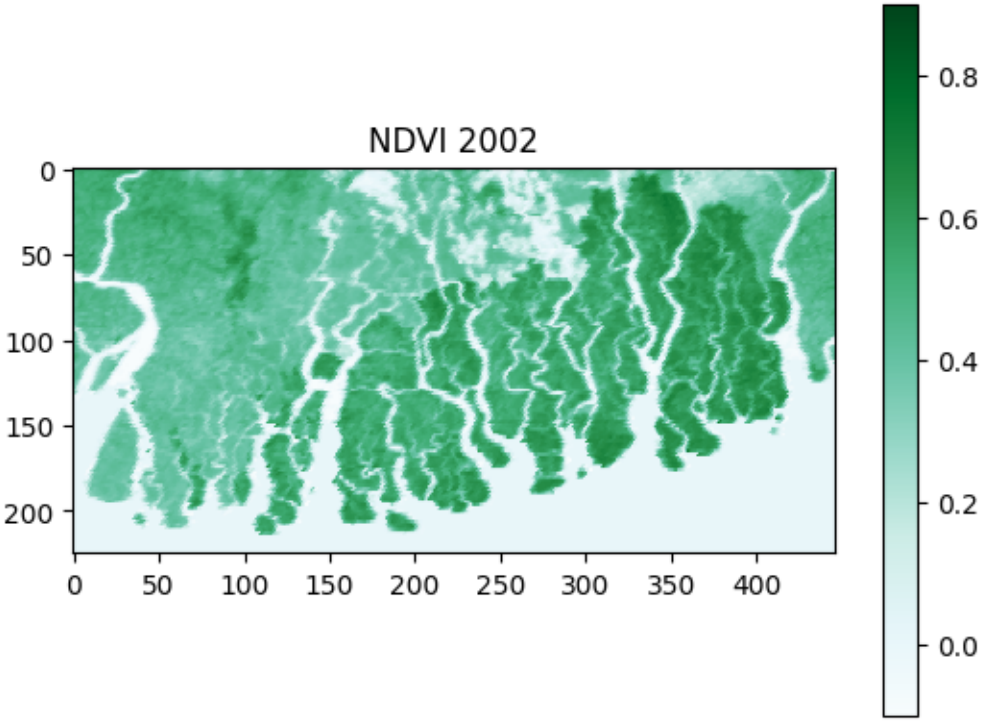

NDVI 2020
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

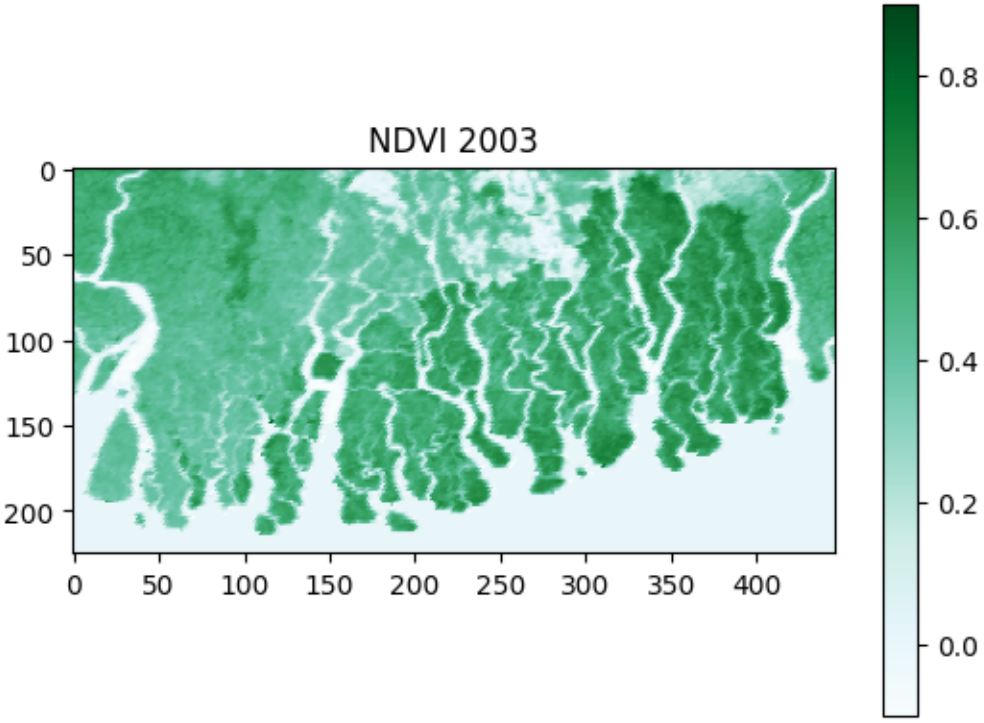
DISPLAY AND EXPORT THE PLOTS

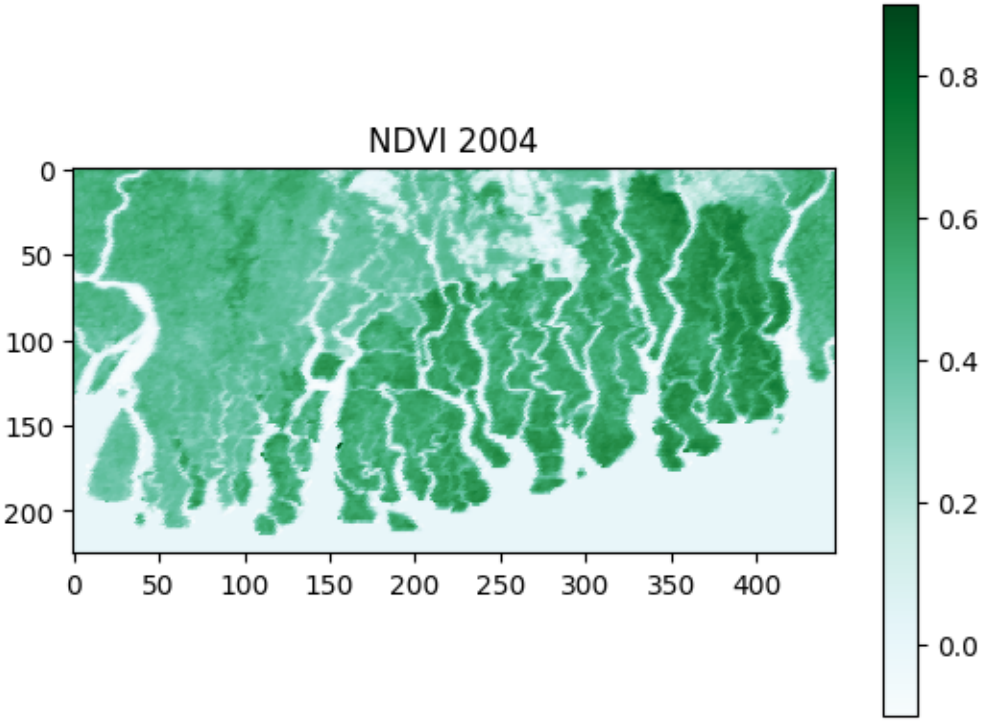
[ ]:
```
# Save the figure

output_dir = '/content/'


# Visualize and save the images for each year from 2000 to 2023
for year in range(2000, 2024):
    img_path = f'/content/NDVI_{year}.tif'
    ds = gdal.Open(img_path)
    band = ds.GetRasterBand(1)
    arr = band.ReadAsArray()

    # Plot the image
    plt.imshow(arr, cmap='BuGn', vmin=-0.1, vmax=0.9)
    plt.title(f'NDVI {year}')
    plt.colorbar()

    # Save the figure
    output_path = os.path.join(output_dir, f'NDVI_{year}.png')
    plt.savefig(output_path, dpi=300, bbox_inches='tight')

    # Display the figure
    plt.show()

    # Close the figure to free up memory
    plt.close()
```

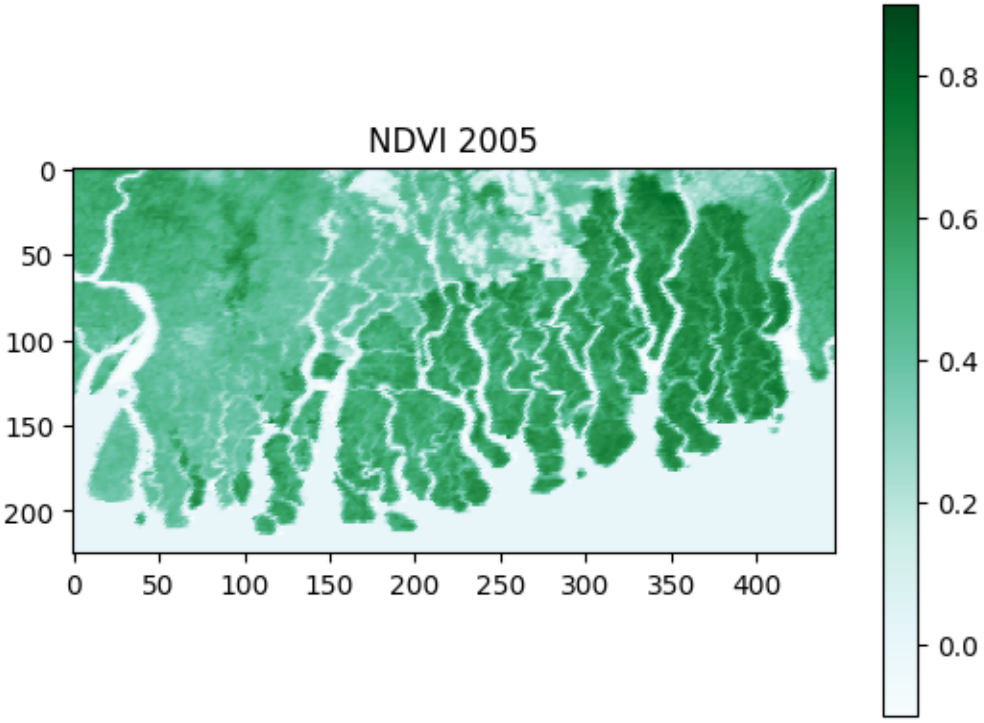

NDVI 2000
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

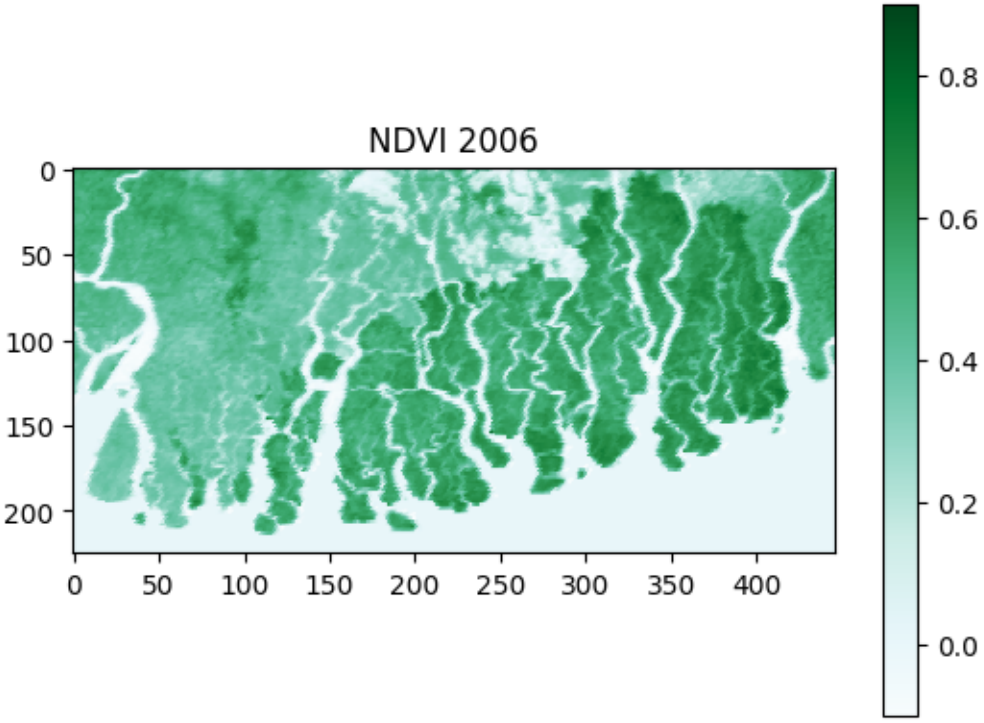

NDVI 2001
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

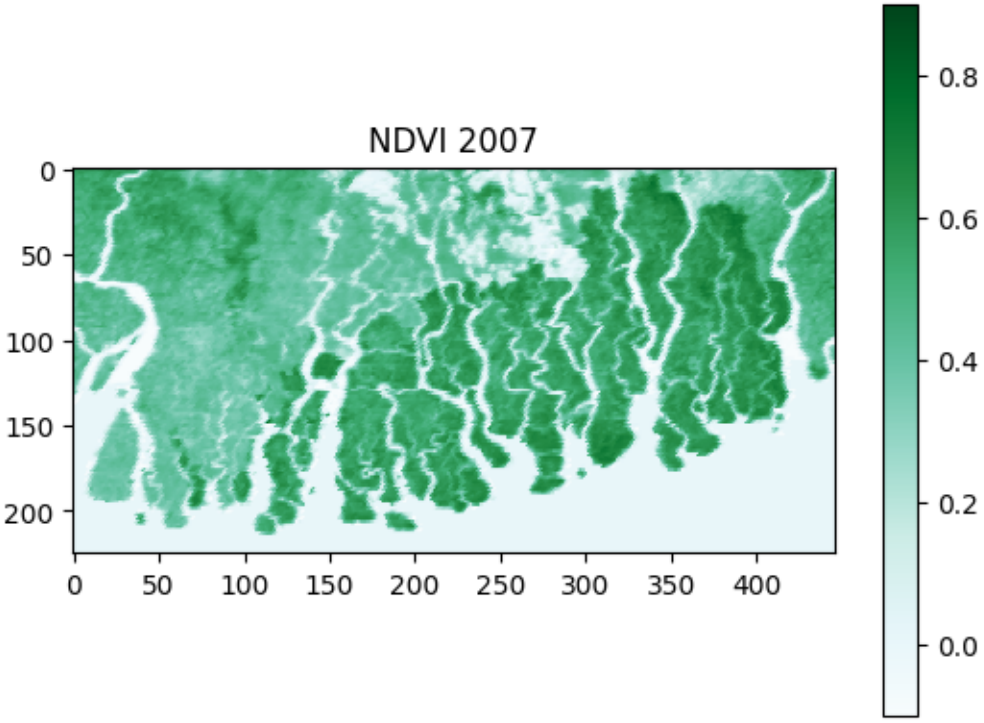

NDVI 2002
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

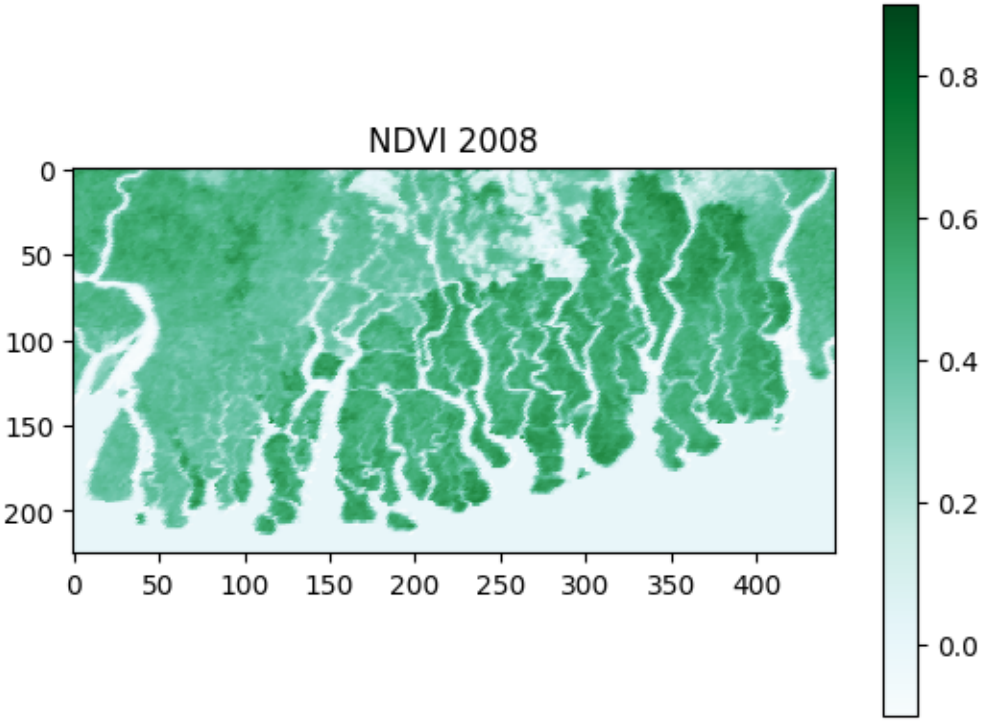

NDVI 2003
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

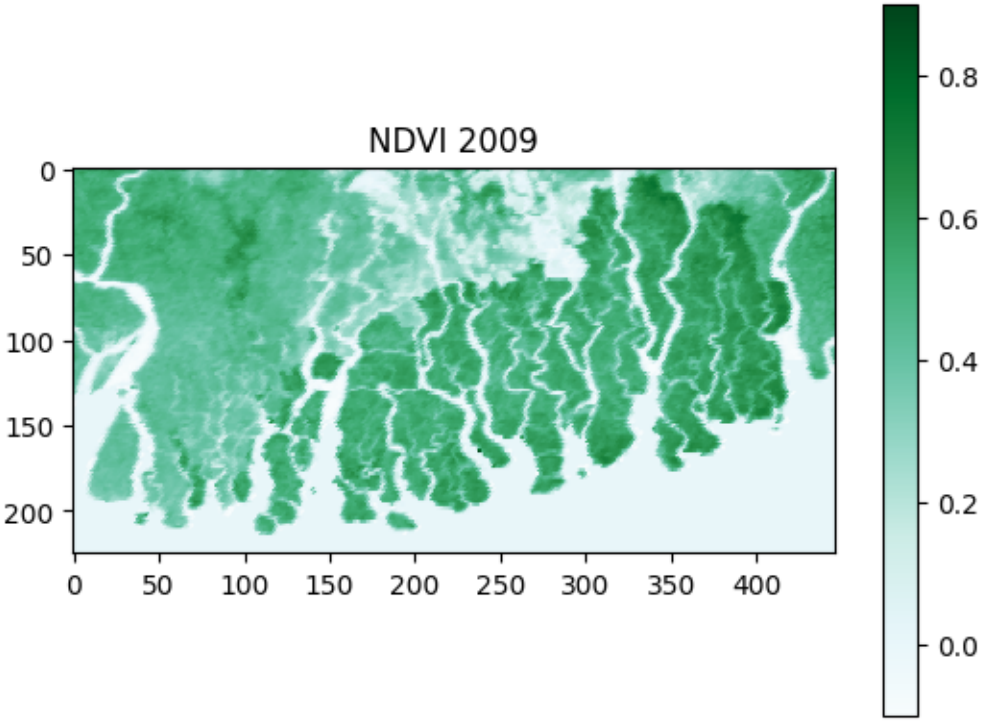

NDVI 2004
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

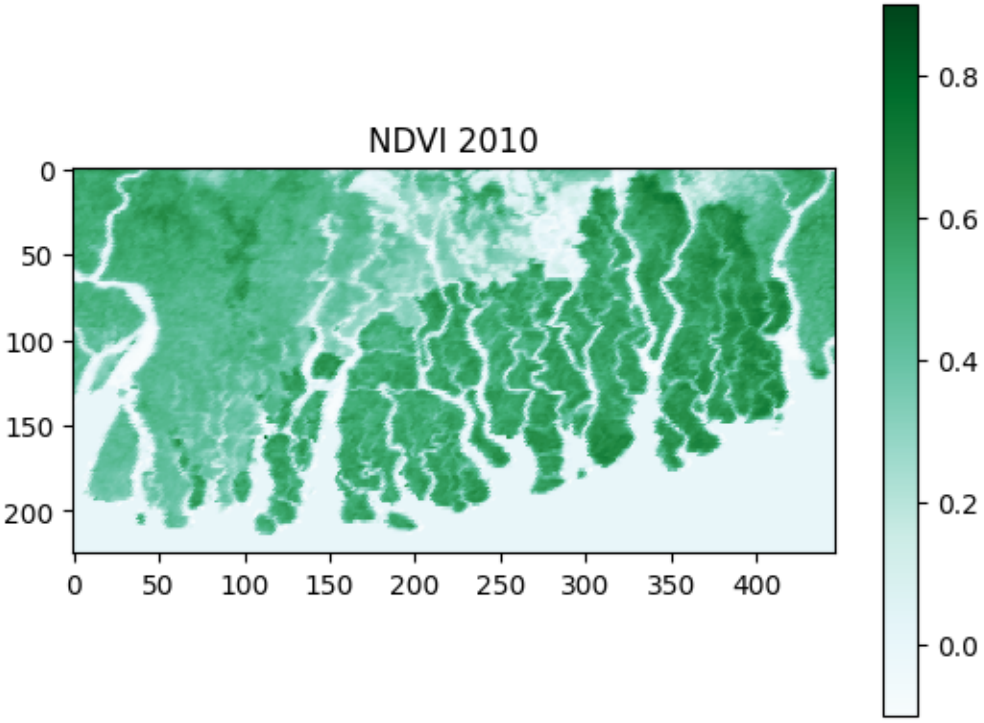

NDVI 2005
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

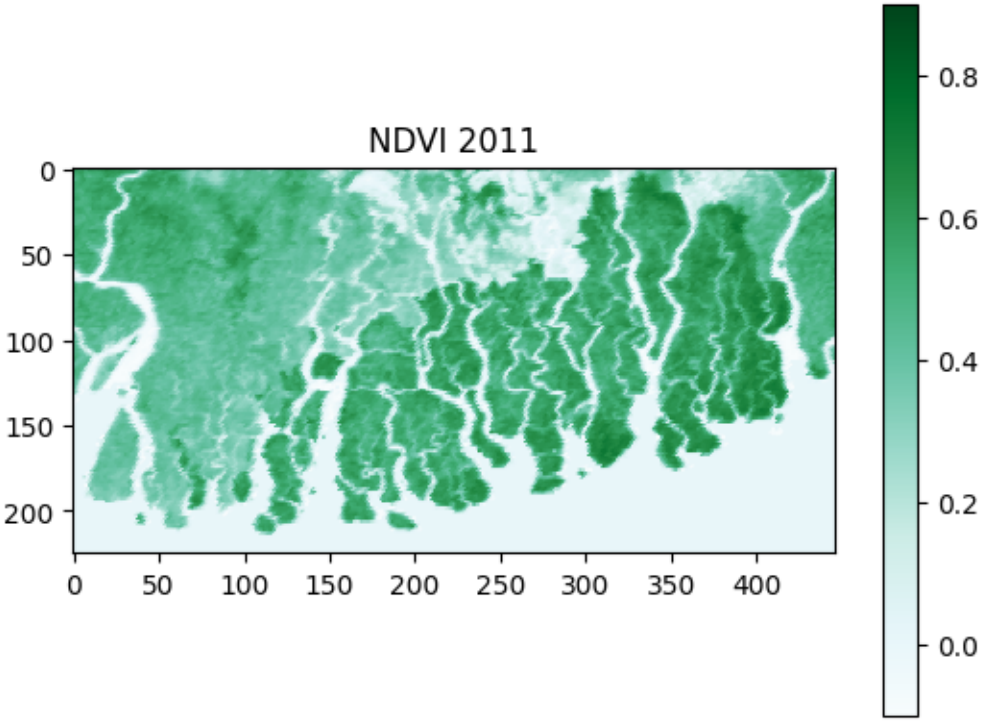

NDVI 2006
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

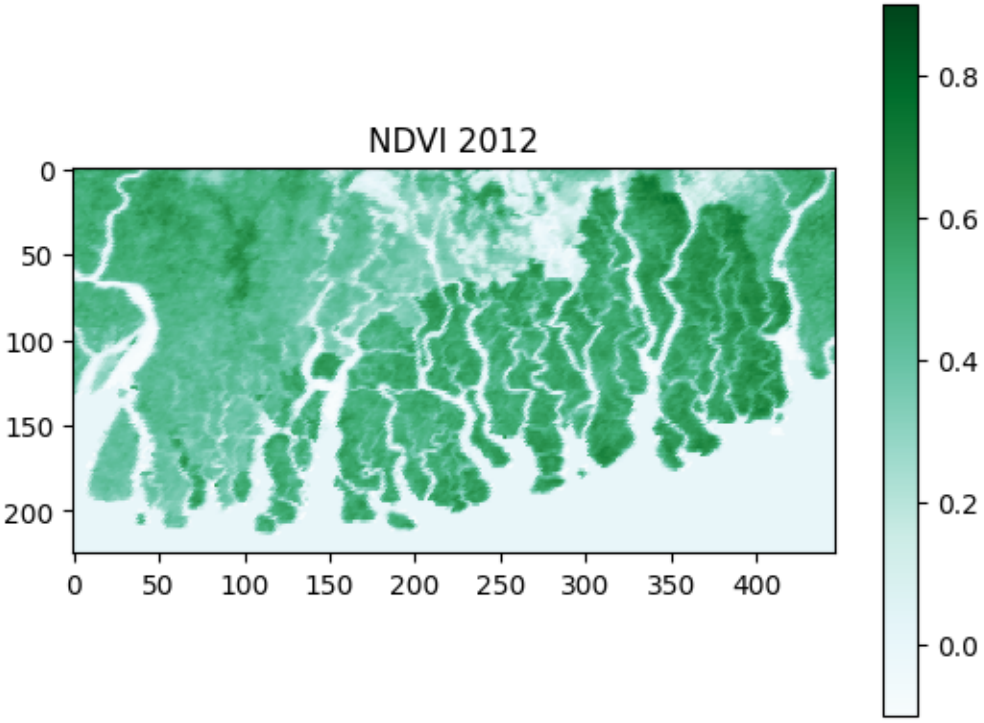

NDVI 2007
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

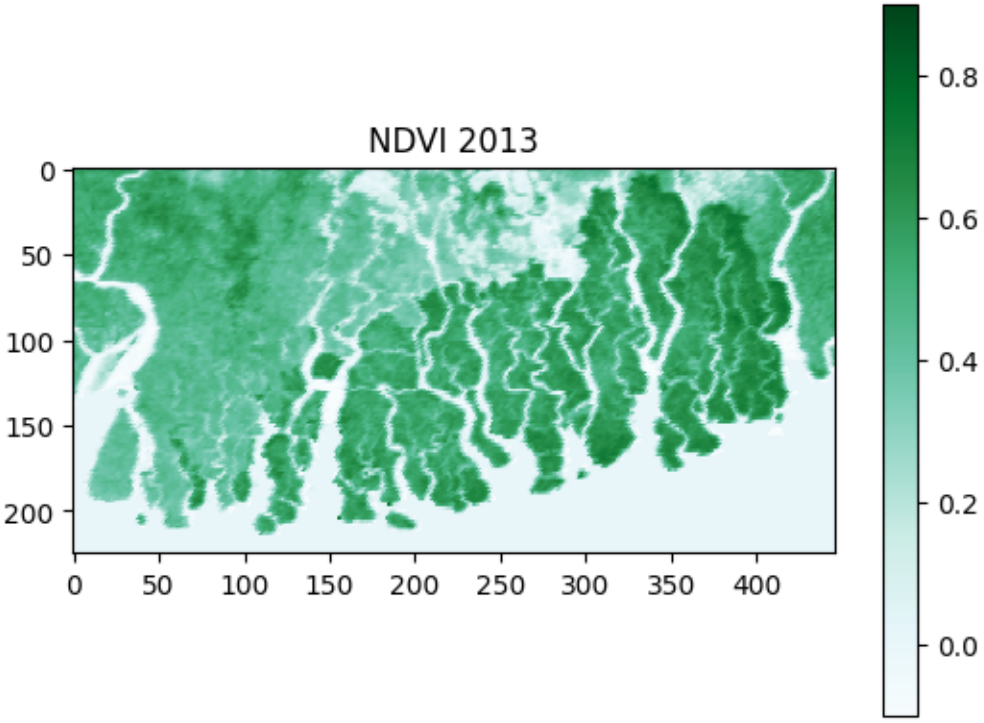

NDVI 2008
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

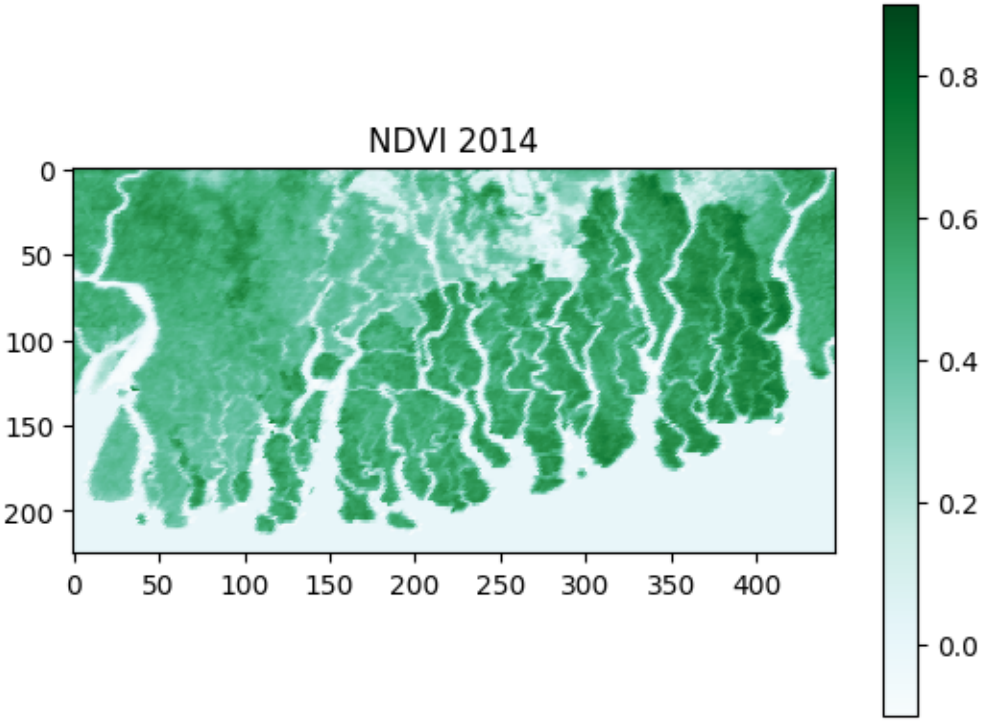

NDVI 2009
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

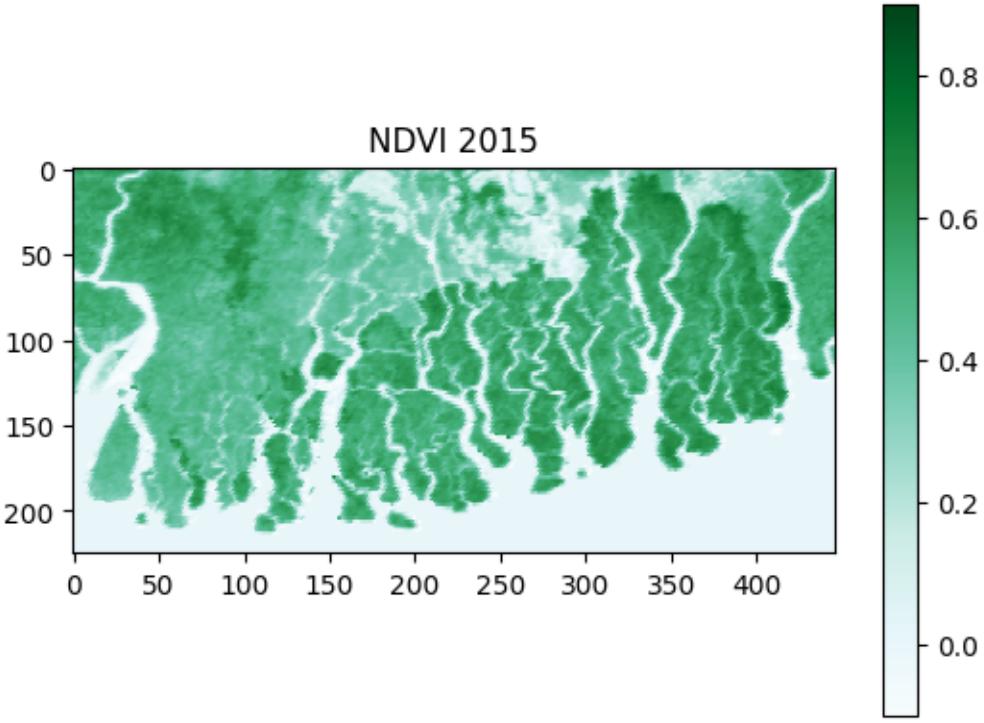

NDVI 2010
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

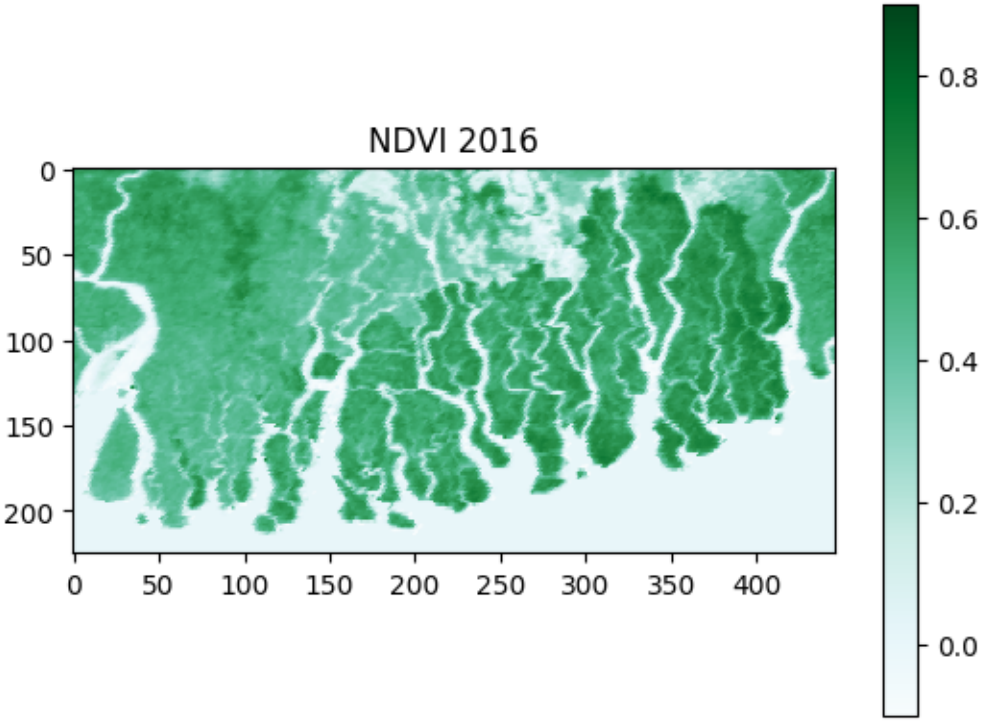

NDVI 2011
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

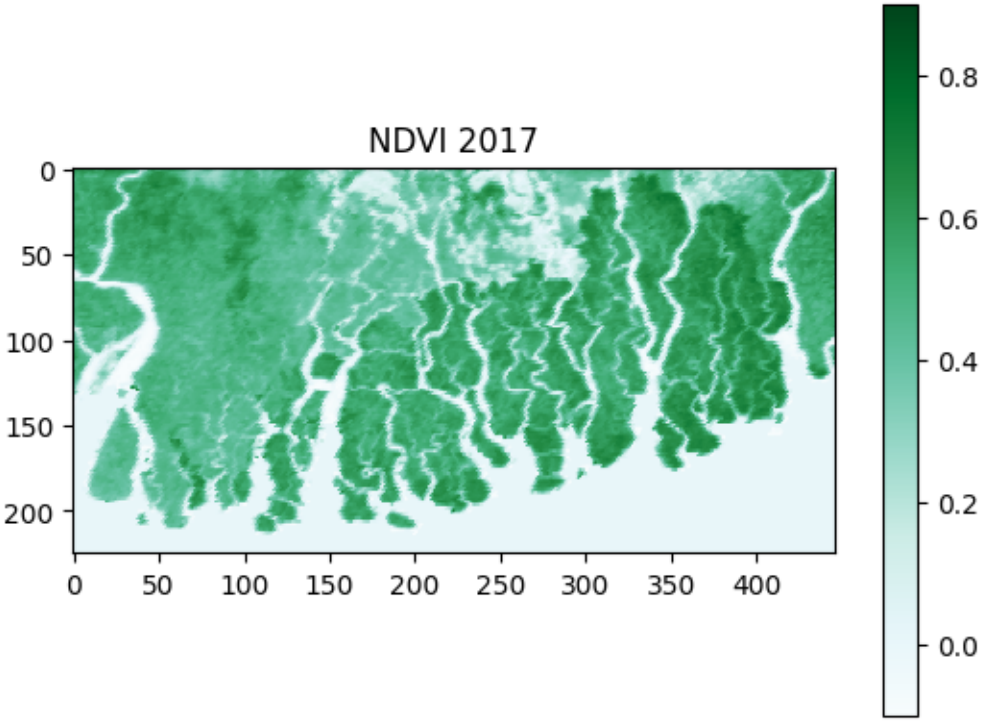

NDVI 2012
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

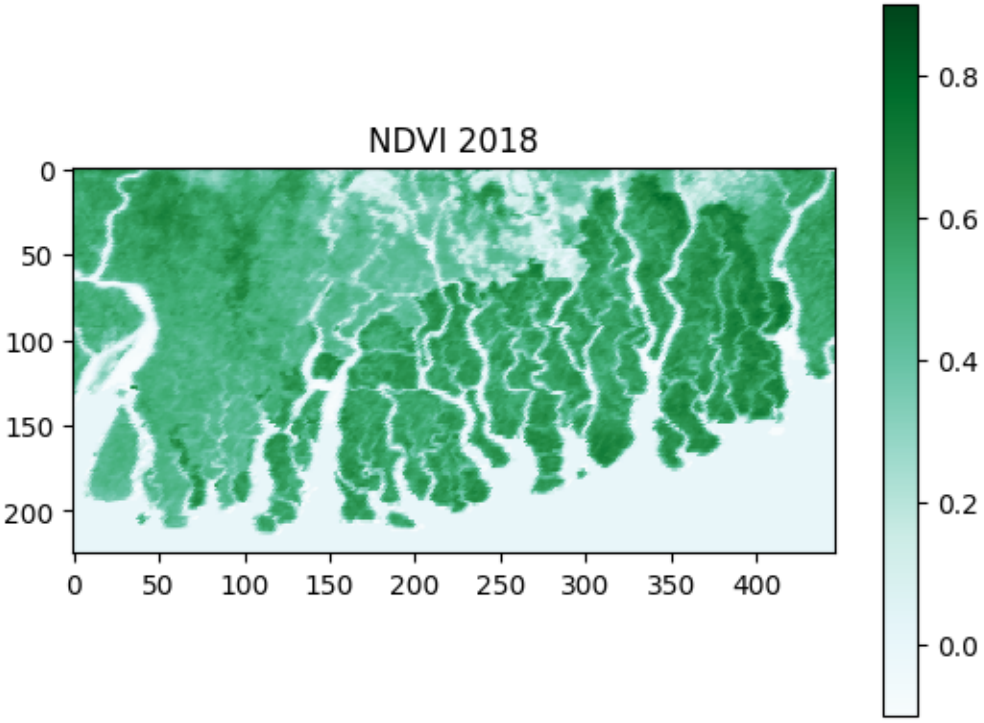

NDVI 2013
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

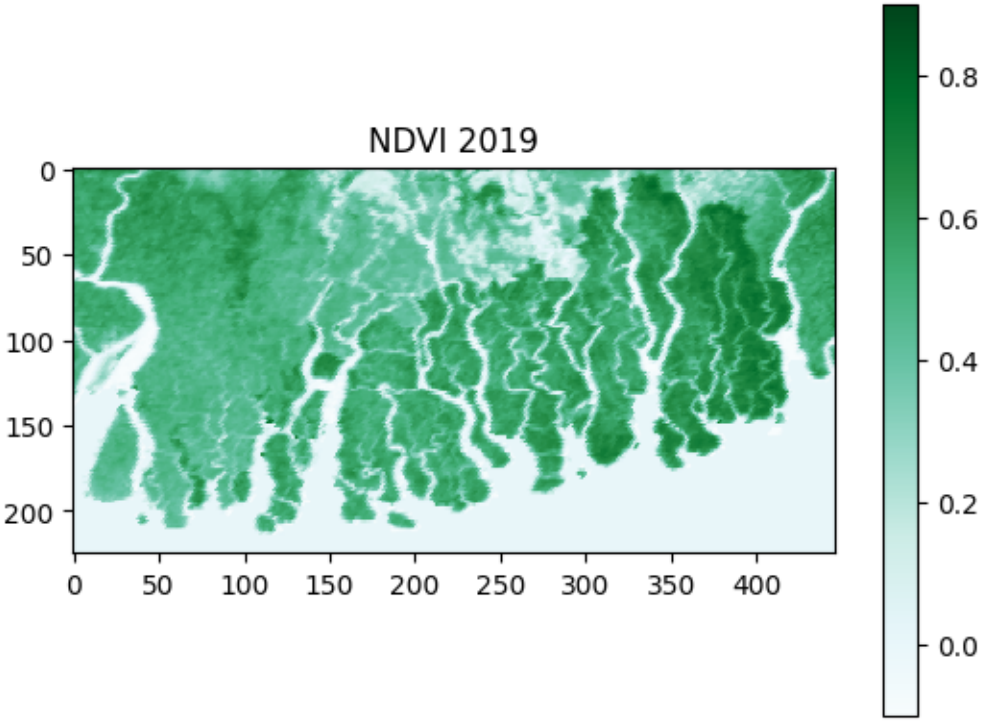

NDVI 2014
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

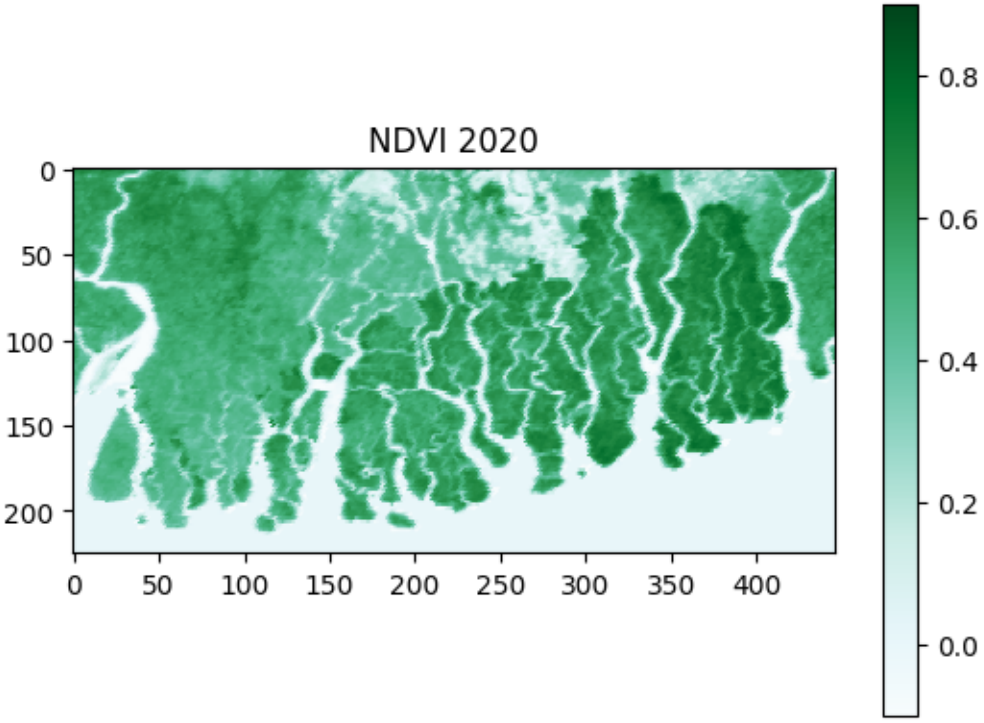

NDVI 2015
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

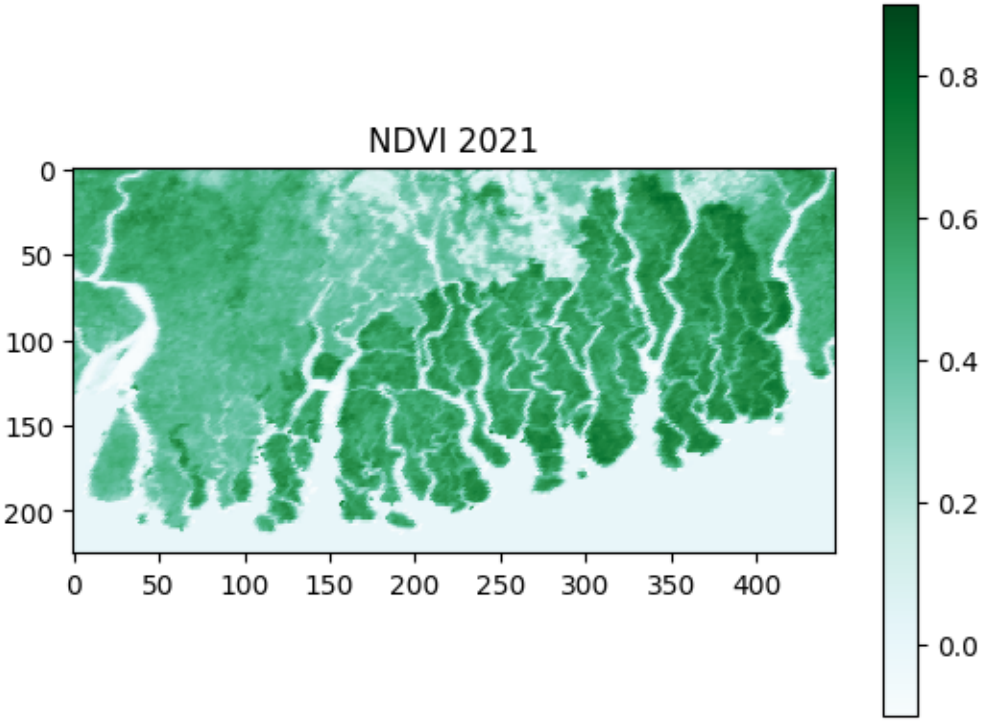

NDVI 2016
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

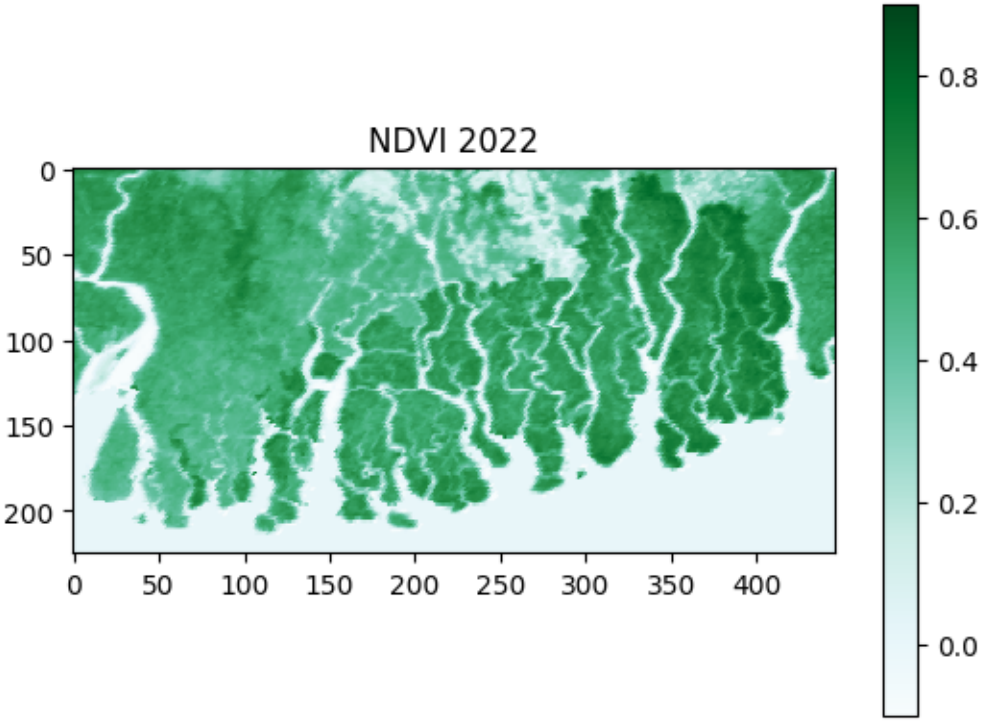

NDVI 2017
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

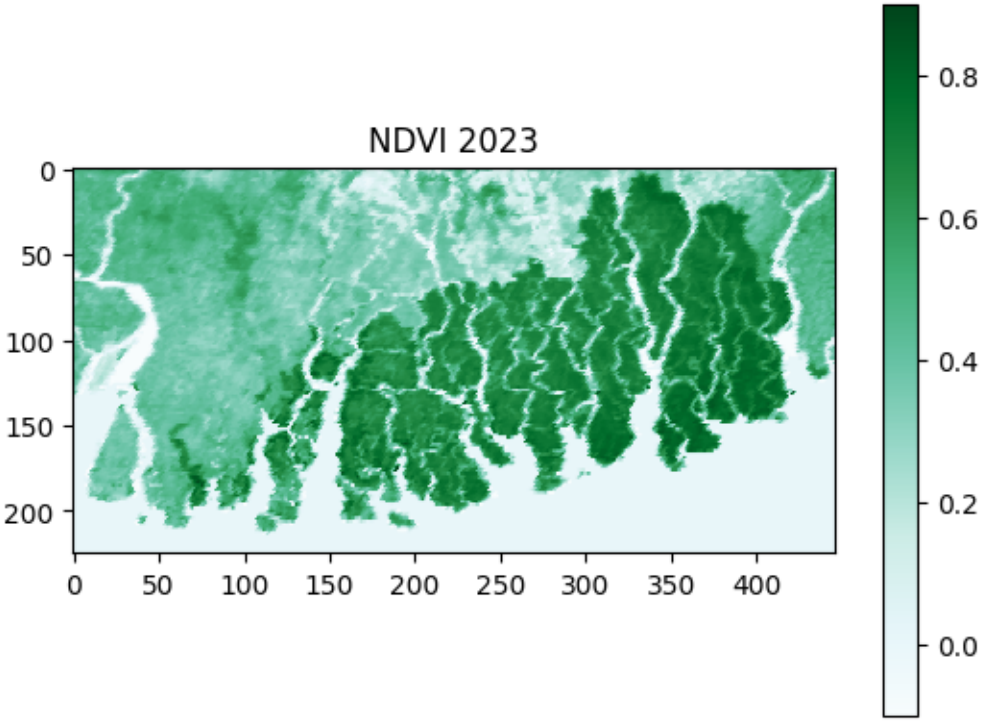

NDVI 2018
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0


NDVI 2019
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0


NDVI 2020
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0


NDVI 2021
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0


NDVI 2022
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0


NDVI 2023
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0.8
0.6
0.4
0.2
0.0

Creating the GIF

[ ]:
```
# Directory where the images are saved
image_dir = '/content/'

# List to store the images
images = []

# Function to normalize the array values to 0-1 for visualization
def normalize(array):
    array_min, array_max = array.min(), array.max()
    return (array - array_min) / (array_max - array_min)

# Define a custom colormap (red-yellow-green)
colors = ["blue", "green"]
cmap = plt.cm.colors.LinearSegmentedColormap.from_list("", colors)

# Loop through each year and read the images
for year in range(2000, 2024):
    image_path = os.path.join(image_dir, f'NDVI_{year}.tif')

    # Open the image using rasterio
    with rasterio.open(image_path) as src:
        array = src.read(1)

    # Normalize the array values
    norm_array = normalize(array)

    # Apply the colormap
    colored_array = (cmap(norm_array) * 255).astype(np.uint8)

    # Convert the colored array to an image
    img = Image.fromarray(colored_array)

    # Convert to RGB (to ensure compatibility with imageio)
    img_rgb = img.convert('RGB')

    # Add the year label to the image
    draw = ImageDraw.Draw(img_rgb)
    font = ImageFont.load_default()
    draw.text((20, 20), str(year), font=font, fill="white")

    # Append to the images list
    images.append(img_rgb)

# Create the GIF
out_gif_path = '/content/NDVI_timelapse.gif'
```

```
images[0].save(out_gif_path, save_all=True, append_images=images[1:], loop=0,␣
 ↪duration=1000)
```

Labelling the GIF

[ ]:
```
# Directory where the images are saved
image_dir = '/content/'

# List to store the images
images = []

# Function to normalize the array values to 0-1 for visualization
def normalize(array):
    array_min, array_max = array.min(), array.max()
    return (array - array_min) / (array_max - array_min)


for year in range(2000, 2024):
    img_path = f"/content/NDVI_{year}.tif"
    img = rasterio.open(img_path).read(1)

    # Normalize the image array between 0 and 1 for visualization
    img_norm = (img - img.min()) / (img.max() - img.min())

    # Convert the normalized image array to an RGB image using the GnBu colormap
    img_rgb = (255 * plt.cm.GnBu_r(img_norm)).astype(np.uint8)

    # Convert the RGB image to a PIL Image
    img_rgb = Image.fromarray(img_rgb[:, :, :3])

    # Add the year label to the image
    draw = ImageDraw.Draw(img_rgb)
    font = ImageFont.load_default()
    draw.text((20, 20), str(year), font=font, fill="black")

    # Append the labeled image to the images list
    images.append(img_rgb)

# Create the GIF
out_gif_path = '/content/NDVI_timelapse_2.gif'
images[0].save(out_gif_path, save_all=True, append_images=images[1:], loop=0,␣
 ↪duration=1000)
```