Creating NDVI Image Using sentinel data for Dharamapuri District, Tamilnadu

Installing the required Libraries

```
!pip·install·sentinelsat
!pip·install·rasterio
!pip·install·folium
!pip·install·geopandas
!pip·install·descartes·
```

```
import folium
import os
import numpy as np
```

```
from sentinelsat import SentinelAPI
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from shapely.geometry import MultiPolygon, Polygon
import rasterio as rio
from rasterio.plot import show
import rasterio.mask
```

Loging in to the copernicus data hub

```
user = 'ss-gis'
password = 'ssgis@3045'

api = SentinelAPI(user, password, 'https://scihub.copernicus.eu/dhus')
```

Reading the Shapefile Geopandas and visualize it with Folium python library.

```
!wget https://www.dropbox.com/s/g24n4lme4722mfo/data.zip
```

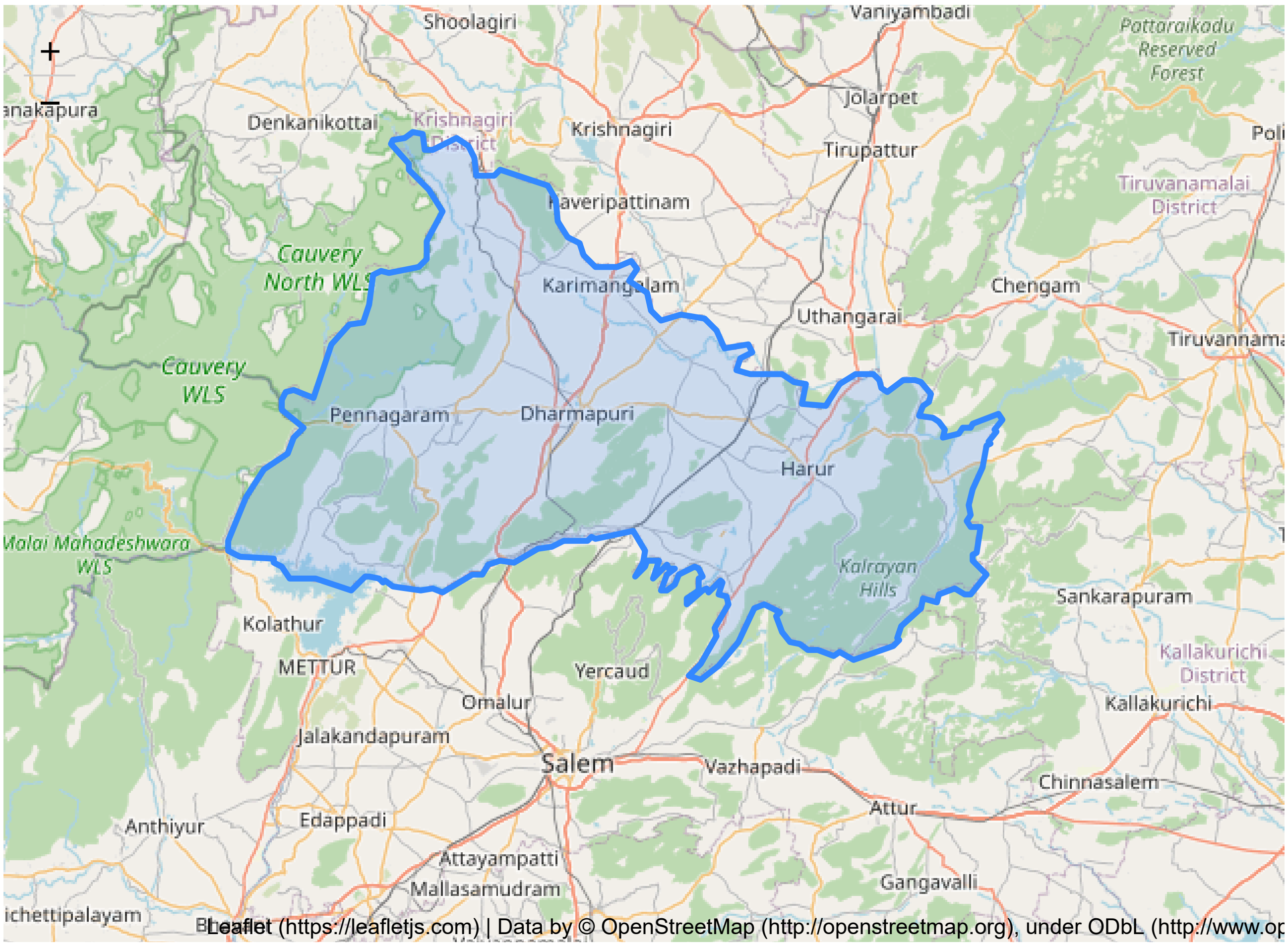
```
!unzip 'data.zip'
```

```
boundary = gpd.read_file('data/Dharmapuri/Dharmapuri.shp')
boundary
```

	cartodb_id	censuscode	dt_cen_cd	st_cen_cd	st_nm	district	
0	148	630.0	29	33	Tamil Nadu	Dharmapuri	POLYGON

```
map = folium.Map([12.11, 78.2091], zoom_start=9)

folium.GeoJson(boundary).add_to(map)
map
```



With the above code, the boundary shapefile is read using Geopandas and called as boundary, then later created an empty base map in Folium centred around coordinates in the area, and defined it as map. Finally, The Geopanda data has been added to the base mapand we can

Creating the footprint

```
bound = gpd.read_file('data/Dharmapuri/overlay.shp')

footprint = None
for i in bound['geometry']:
    footprint = i
```

Created a query for Sentinel 2 images Level 2A with cloud coverage between 0 and 10 that fall or intersect with the footprint (Area of study). For the time period, we are interested in Sentinel Level 2A satellite images taken between '20220501' and '20220530'

```
products = api.query(footprint,
                    date = ('20220501', '20220530'),
                    platformname = 'Sentinel-2',
                    processinglevel = 'Level-2A',
                    cloudcoverpercentage = (0,10))
```

From here we can create a GeodataFrame or Dataframe from the product dictionary and sort them according to cloud coverage percentage.

```
products_gdf = api.to_geodataframe(products)
products_gdf_sorted = products_gdf.sort_values(['cloudcoverpercentage'], ascending=[True])
products_gdf_sorted
```

Let us say we are interested in the first satellite image since this has the least cloud coverage of all available images. we can simply call download and provide the product name

```
api.download("053a906a-6f8d-4769-9a20-47432d197d96")
```

```
!wget https://www.dropbox.com/s/hxilp617uvlbot6/S2A_MSIL1C_20220528T050701_N0400_R019_T43P
```

```
!unzip 'S2A_MSIL1C_20220528T050701_N0400_R019_T43PHP_20220528T070315.zip'
```

Create RGB Image

```
source ='S2A_MSIL1C_20220528T050701_N0400_R019_T43PHP_20220528T070315/S2A_MSIL1C_20220528T
```

```
# Open Bands 4, 3 and 2 with Rasterio
b4 = rio.open(source+'/T43PHP_20220528T050701_B04.jp2')
b3 = rio.open(source+'/T43PHP_20220528T050701_B03.jp2')
b2 = rio.open(source+'/T43PHP_20220528T050701_B02.jp2')
```

```
with rio.open('RGB_Image.tiff','w',driver='Gtiff', width=b4.width, height=b4.height,
              count=3,crs=b4.crs,transform=b4.transform, dtype=b4.dtypes[0]) as rgb:
    rgb.write(b2.read(1),1)
    rgb.write(b3.read(1),2)
    rgb.write(b4.read(1),3)
    rgb.close()
```

```
raster = rio.open("RGB_Image.tiff")
```

NDVI

To calculate the NDVI, we need Red band and Near-Infrared Band (NIR).Sentinel Images have red in 4th band and NIR in the 8th band.

The formula for NDVI calculation is: nir - red /(nir + red).

First we need to read the 4th and 8th bands as arrays and creating NDVI Index using the above formula.

```
b4 = rio.open(source+'/T43PHP_20220528T050701_B04.jp2')
b8 = rio.open(source+'/T43PHP_20220528T050701_B08.jp2')
```

```
red = b4.read()
nir = b8.read()
```

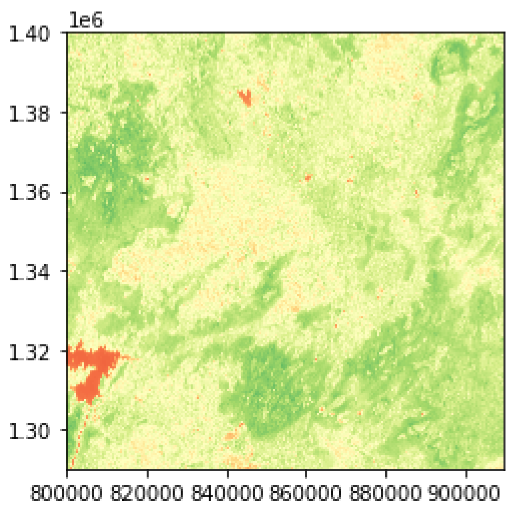
```
ndvi = (nir.astype(float)-red.astype(float))/(nir+red)
```

```
meta = b4.meta

meta.update(driver='GTiff')
meta.update(dtype=rasterio.float32)

with rasterio.open('NDVI2.tif', 'w', **meta) as dst:
    dst.write(ndvi.astype(rasterio.float32))
```

```
ndvi = rio.open(r'NDVI2.tif')
show(ndvi, transform=ndvi.transform, cmap='RdYlGn')
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9ee9d8210>
```

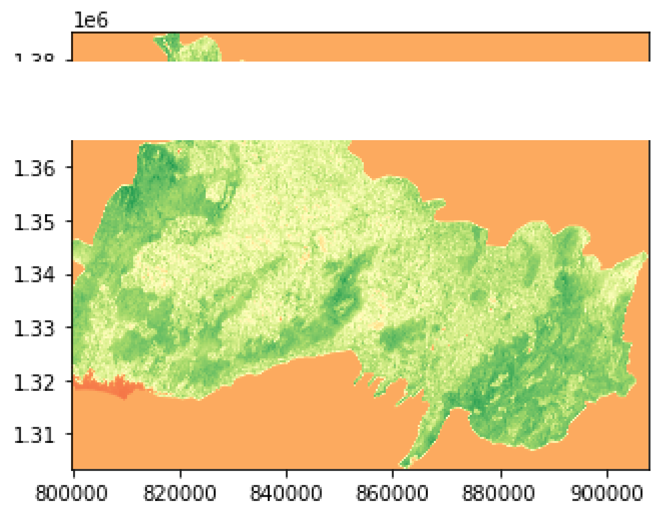
Mask Satellite image

```
boundary_proj = boundary.to_crs({'init': 'epsg:32643'})

with rio.open("/content/NDVI2.tif") as src:
    out_image, out_transform = rio.mask.mask(src, boundary_proj.geometry,crop=True)
    out_meta = src.meta.copy()
    out_meta.update({"driver": "GTiff",
                 "height": out_image.shape[1],
                 "width": out_image.shape[2],
                 "transform": out_transform})

with rio.open("NDVI_masked.tif", "w", **out_meta) as dest:
    dest.write(out_image)


ndvi_mask = rio.open(r'NDVI_masked.tif')
show(ndvi_mask, transform=ndvi_mask.transform, cmap='RdYlGn')
```


1e6
1.36
1.35
1.34
1.33
1.32
1.31
800000
820000
840000
860000
880000
900000

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9ed47dd90>
```