Satellite Image Processing

AU1940205	AU1940067	AU1940190	AU1940189
Henil Shah	**Kathan Joshi**	**Devanshu Magiawala**	**Devarsh Sheth**

Synopsis:-

The project intends on performing various operations on satellite images and getting a useful amount of data from it. The satellite imagery used for the project is made available from a site called scihub.copernicus. We then store this dataset in the form of a NumPy array using a python library. Now, using python and QGIS we have performed various operations to generate output imagery.

Transmission of signal from Earth-Satellite-Earth:

The Sun sheds a lot of energy on Earth in visible and non-visible spectral bands. Satellites record this energy after it reflects off the outside of Earth and skips back toward space. Objects on Earth. For example, backwoods, water, asphalt, or snow all reflect various measures of energy. It's these distinctions in reflectivity that empower us to recognize objects through far off detecting.

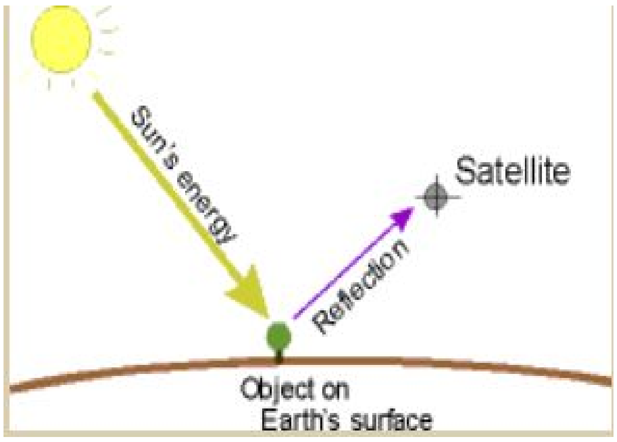
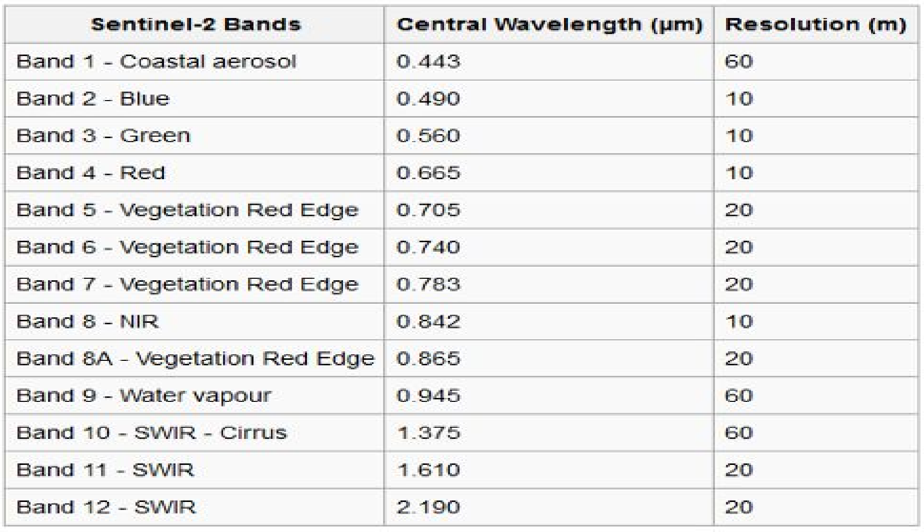


Figure 1
Reference: http://web.sonoma.edu/users/f/freidel/techniques/exer/rem_sens/remsen_e.html

Gatherings of items on Earth have average impressions of energy that help distinguish the objects. For instance, water reflects next to little visible or infrared energy. Snow reflects energy unequivocally, which is the reason it seems white (the mix of all noticeable light frequencies). Vegetation is an intriguing case. Solid vegetation assimilates most obvious light, yet unequivocally reflects infrared. Indeed, any article that had a solid infrared sign and a feeble red noticeable sign would more likely than not be vegetation. Consequently, items can be recognized somewhat dependent on their "spectral signature" or a mix of reflectances in different bands.

Satellites communicate by using radio waves to send signals to the antennas on the Earth. The antennas then capture those signals and process the information coming from those signals



Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

Figure 2
Reference: https://towardsdatascience.com/satellite-imagery-access-and-analysis-in-python-jupyter-notebooks-387971ece84b

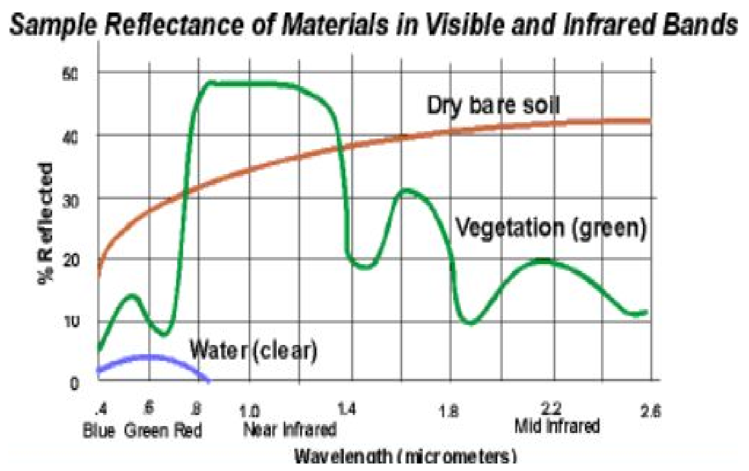
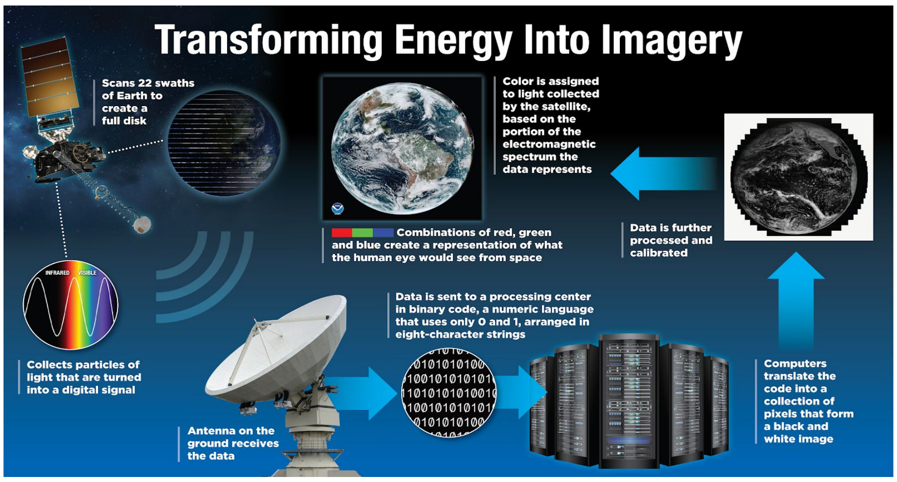


Figure 3

Reference: http://web.sonoma.edu/users/f/freidel/techniques/exer/rem_sens/remsen_e.html

Satellite imagers use remote sensing to collect information about Earth from above. Satellite images are made out of thousands of pixels that the satellite checked into lines and segments. The satellite assembles a gathering of columns into a PC document. This document covers a zone of Earth known as a scene. The scene size fluctuates relying upon the sensor. Sensors on Landsat, the US satellite arrangement, has scenes around 185 km (115 miles) on each side. The outline shows a Landsat scene inclusion for the San Francisco Bay Area. Scenes for other normally utilized sensors range in size from 60 km to 2200 km. A scene of Landsat can have more than 6000 lines and sections of pixels.

For every pixel, the satellite records the measure of energy in at least one band, contingent upon the plan of the sensor. So if the pixel size was 20 meters, the satellite may record one perusing of the measure of blue light, one of the measures of green, one of red, and one every one of two distinctive infrared groups, for a sum of five readings or brightnesses for that one pixel. The smaller the 'pixel' the more precision it can give.



Transforming Energy Into Imagery
Scans 22 swaths of Earth to create a full disk
Color is assigned to light collected by the satellite, based on the portion of the electromagnetic spectrum the data represents
Combinations of red, green and blue create a representation of what the human eye would see from space
Data is further processed and calibrated
INFRARED VISIBLE
Collects particles of light that are turned into a digital signal
Data is sent to a processing center in binary code, a numeric language that uses only 0 and 1, arranged in eight-character strings
Antenna on the ground receives the data
Computers translate the code into a collection of pixels that form a black and white image

Figure 4
Flowchart for the transformation of the signal passing through a system and giving us respective output

Reference: https://goes-r.gov/multimedia/infographics.html?hcb=1

Satellite Imagery Analysis:

Satellite Image Processing is an important field in research and development and consists of the images of earth and satellites taken by the means of artificial satellites. These imageries are taken in digital form and later are processed by the computers to extract meaningful information.

The Project is done in Python and modified with QGIS.

The project firstly recognizes the different spectral bands of the imagery and then divides it individually into an array very similar to a NumPy array and then perform individual operations: -

- ❏ Rasterio Library is used to store the raster data as a NumPy array (Note: This array starts with index 1 and not 0)
  - ❏ Raster data is a grid of regularly sized pixels.
  - ❏ Each pixel has different values, based on the number of spectral bands the imagery contains.

QGIS(Quantum Geographic Information System):

- QGIS is open-source software, functioning as a geographic information system.
- It allows the user to analyze and edit geographical information (spatial information).
- We used QGIS to enhance the quality of output images; the colour scale is formed with QGIS.
- QGIS underpins both raster and vector layers; vector information is put away as one or the other point, line, or polygon highlights. Different arrangements of raster pictures are upheld, and the product can georeference pictures.

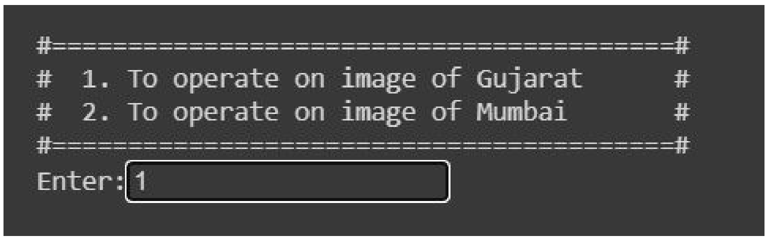
Some of the operations shown in this project are as follows:

- RGB Image
- NDVI Image(Vegetation Index)
- NRG Image
- NDWI Image(Water Index)

User Interface:-

A simple user interface is created by python coding. It is based on a number selecting process. Firstly, an option is given to select between the image the user would like to access.

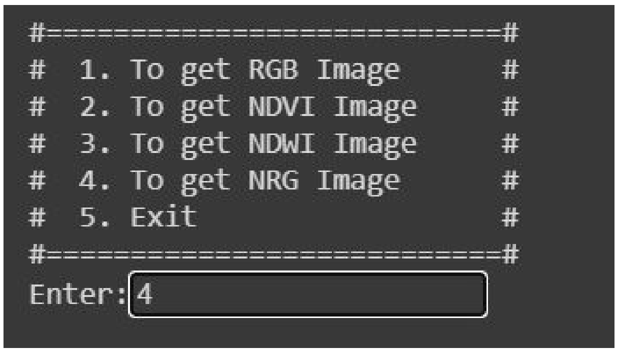
For example, if you enter 1 it will select the image of Gujarat:



```
#=========================================#
#  1. To operate on image of Gujarat      #
#  2. To operate on image of Mumbai       #
#=========================================#
Enter: 1
```

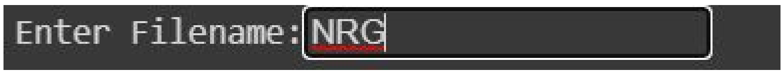
Further going on, it can be seen that 4 selection options are given to select which type of image the user would like to get as output and also an exit option which would exit the program

Here for instance option 4 is selected:



```
#==========================#
#  1. To get RGB Image     #
#  2. To get NDVI Image    #
#  3. To get NDWI Image    #
#  4. To get NRG Image     #
#  5. Exit                 #
#==========================#
Enter: 4
```

Next step after selecting one of the four options, the user is entitled to give the name of the filename which would be generated.



```
Enter Filename: NRG
```

And the process is done. The file (image file) would be generated after processing for some time. The selected output is first generated and then downloaded:

NRG.tiff

Code of the Project:

Note: The following code contains the dataset for both the images which are stored in the drive. To run this, one needs to download the dataset from here. https://drive.google.com/drive/folders/12dxeH-RSafYEWYqepE-_IkkXPXLEpWtN?usp=sharing

```
import rasterio as rio
def RGB(filename):
  # To create an RGB image
        filename = filename + '.tiff'
        with rio.open(filename,'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()


def NDVI(filename):
  # read Red(b4) and NIR(b8) as arrays
      red = b4.read()
      nir = b8.read()

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

      # Write the NDVI image into the given filename
      meta = b4.meta
      meta.update(driver='GTiff')
      meta.update(dtype=rio.float32)
      filename = filename + '.tif'
      with rio.open(filename, 'w', **meta) as dst:
          dst.write(ndvi.astype(rio.float32))

def NDWI(filename):
      # Calculating NDWI
      gre = b3.read()
      nir = b8.read()

      ndwi = (gre.astype(float) - nir.astype(float))/(nir+gre)
      # Write the NDWI image into the given filename
      meta1 = b8.meta
      meta1.update(driver='GTiff')
      meta1.update(dtype=rio.float32)
      filename = filename + '.tif'
      with rio.open(filename, 'w', **meta1) as dst:
          dst.write(ndwi.astype(rio.float32))

def NRG(filename):
      # Generating NRG Image
      filename = filename + '.tiff'
      with rio.open(filename,'w',driver='Gtiff', width=b4.width,
height=b4.height,
                    count=3,crs=b4.crs,transform=b4.transform,
dtype=b4.dtypes[0]) as nrg:
          nrg.write(b3.read(1),1)
          nrg.write(b4.read(1),2)
          nrg.write(b8.read(1),3)
          nrg.close()
```

```
image_no = int(input("""
  #=========================================#
  #  1. To operate on image of Gujarat      #
  #  2. To operate on image of Mumbai       #
  #=========================================#
  Enter:"""))

if image_no == 1:
  # Open Bands 8, 4, 3 and 2 with Rasterio
  R10 =
'/content/drive/MyDrive/ECE_project/S2A_MSIL2A_20201206T055221_N0214_R0
48_T42QXJ_20201206T075916.SAFE/GRANULE/L2A_T42QXJ_A028502_20201206T0552
23/IMG_DATA/R10m'
  b8 = rio.open(R10+'/T42QXJ_20201206T055221_B08_10m.jp2')
  b4 = rio.open(R10+'/T42QXJ_20201206T055221_B04_10m.jp2')
  b3 = rio.open(R10+'/T42QXJ_20201206T055221_B03_10m.jp2')
  b2 = rio.open(R10+'/T42QXJ_20201206T055221_B02_10m.jp2')

  while 1==1:
      img_type = int(input("""
      #=============================#
      #  1. To get RGB Image        #
      #  2. To get NDVI Image     #
      #  3. To get NDWI Image     #
      #  4. To get NRG Image      #
      #  5. Exit                  #
      #===========================#
      Enter:"""))

      if img_type == 1:
        filename=input("Enter Filename:")
        RGB(filename)

      elif img_type == 2:
        filename=input("Enter Filename:")
        NDVI(filename)

      elif img_type == 3:
        filename=input("Enter Filename:")
        NDWI(filename)

      elif img_type == 4:
        filename=input("Enter Filename:")
        NRG(filename)

      elif img_type ==5:
        print("Thank you for using our program!!!!!")
        break;

      else:
        print("Please enter valid image type..!!!")


elif image_no == 2:
    # Open Bands 8, 4, 3 and 2 with Rasterio
    R10 =
'/content/drive/MyDrive/ECE_project/S2A_MSIL2A_20201203T054211_N0214_R0
05_T43QBA_20201203T075112(NEW).SAFE/GRANULE/L2A_T43QBA_A028459_20201203
T054212/IMG_DATA/R10m'
    b8 = rio.open(R10+'/T43QBA_20201203T054211_B08_10m.jp2')
    b4 = rio.open(R10+'/T43QBA_20201203T054211_B04_10m.jp2')
    b3 = rio.open(R10+'/T43QBA_20201203T054211_B03_10m.jp2')
    b2 = rio.open(R10+'/T43QBA_20201203T054211_B02_10m.jp2')

    while 1==1:
        img_type = int(input("""
        #===========================#
        #  1. To get RGB Image      #
        #  2. To get NDVI Image     #
        #  3. To get NDWI Image     #
        #  4. To get NRG Image      #
        #  5. Exit                  #
        #===========================#
        Enter:"""))
        if img_type == 1:
                filename=input("Enter Filename:")
                RGB(filename)

        elif img_type == 2:
                filename=input("Enter Filename:")
                NDVI(filename)

        elif img_type == 3:
                filename=input("Enter Filename:")
                NDWI(filename)

        elif img_type == 4:
                filename=input("Enter Filename:")
                NRG(filename)
        elif img_type == 5:
                print("Thank you for using our program!!!!")
                break;
        else:
          print("Please enter valid image type..!!!")
    else:
      print("Please select valid image ..!!!")
```

<u>NRG(Near-Infrared Red Green) Images:</u>

The three basic components of visible light are red, green, and blue light (RGB). In theory, any visible colour can be created by combining these colours in some way. The NRG image is also known as a "false colour" image, which means that the colours have been assigned to three different wavelengths that your eyes might not normally see. False-color (or pseudocolour) refers to a group of colour rendering methods used to display images in colour which were recorded in the visible or non-visible parts of the electromagnetic spectrum. The blue band has a poor transmission and hence this type of images can be used

Landsat 7 images are colour composites, made by assigning the three primary colours to three bands of the Enhanced Thematic Mapper (ETM+) sensor. These images are not colour photographs; they are "false colour" images (green fields won't necessarily look green in the image).

RGB = NRG (Red, Green, Blue = Near Infrared, Red, Green, or "energy")

Red = Near IR (ETM+ band 4)

Green = Red (ETM+ band 3)

Blue = Green (ETM+ band 2)

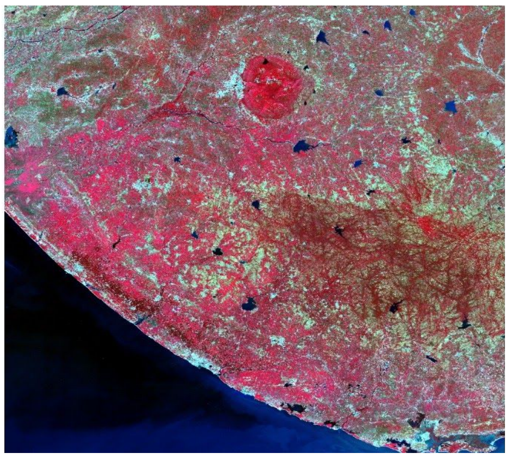
SWIR – a part of the range with wavelengths in the range of 1.628-1.652 nm.

Code for NRG:

```
def NRG(filename):
      # Generating NRG Image
      filename = filename + '.tiff'
      with rio.open(filename,'w',driver='Gtiff', width=b4.width,
height=b4.height,
                    count=3,crs=b4.crs,transform=b4.transform,
dtype=b4.dtypes[0]) as nrg:
          nrg.write(b3.read(1),1)
          nrg.write(b4.read(1),2)
          nrg.write(b8.read(1),3)
          nrg.close()
```

Output:

Original Image


NRG Image

RGB(Red Green Blue) Images:

The satellite image is formed with the use of primary colours: red, green, and blue. Looking at the history, the first enhanced images were formed for severe convection eg. MCS (Mesoscale convective system) for revealing the coldest tops and the growth rate there. The first RGB combination was developed for the USA's polar-orbiting satellites and it was used for discriminating the low clouds from high and middle clouds. A large number of RGB's were developed by institutes and weather services.

The three colours red, green, and blue are assigned for three MSG channels or, if proper, channel contrasts. All different colours are produced as a blend of these three basic tones. The decision among channels and channel contrasts relies upon the landscapes that are being searched for.

This will be exhibited with the help of Natural Colour RGB, which is formed from the bands 03 (R), 02 (G), and 01 (B). Every one of these three noticeable channels contains data about reflected daylight (and therefore the optical thickness of clouds), yet each channel additionally adds its actual claim to fame:

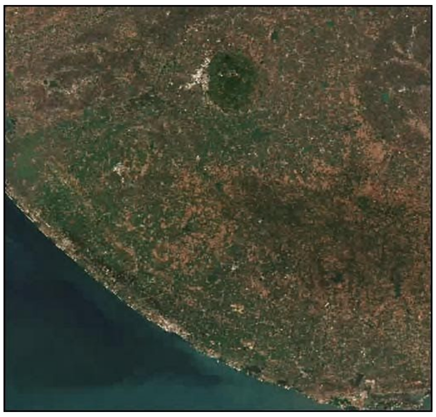
band 03 (1.6 micrometre): particle phase and size;band 02 (0.8 micrometre): "greenness" of vegetation;band 01 (0.6 micrometer): optical thickness

Code for RGB:

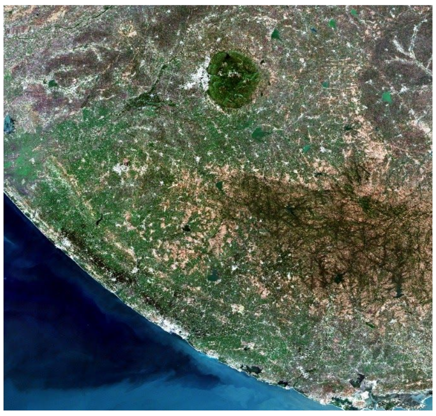
```
def RGB(filename):
  # To create an RGB image
        filename = filename + '.tiff'
        with rio.open(filename,'w',driver='Gtiff', width=b4.width,
height=b4.height,
                    count=3,crs=b4.crs,transform=b4.transform,
dtype=b4.dtypes[0]) as rgb:
          rgb.write(b2.read(1),1)
          rgb.write(b3.read(1),2)
          rgb.write(b4.read(1),3)
          rgb.close()
```

Output:

Original Image



RGB Image



<u>NDVI(Normalised Difference Vegetation Index) Image:</u>

Calculating NDVI (Normalized Difference Vegetation Index) is an indicator that helps us to know the presence of green vegetation from satellite images. For calculating NDVI we require Red Band and Near-Infrared Band (NIR). The band numbers depend on the satellite images. Sentinal satellite images have red in the 4th band and NIR in the 8th band. The calculation formula is:

NDVI = (nir – red) / (nir + red)

The output of the satellite image as NDVI is shown as:

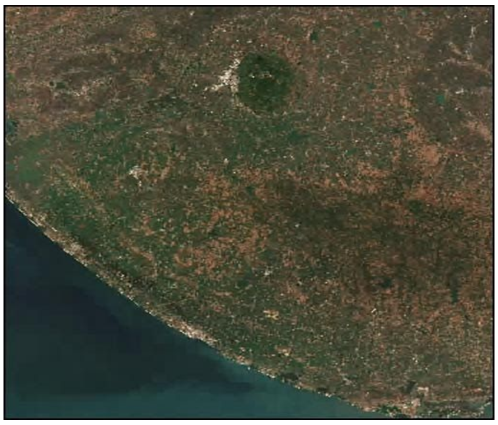
The low vegetation area is shown as red. These spectral reflectances are themselves proportions of the reflected over the approaching radiation in each spectral band independently, subsequently, they take on qualities somewhere in the range of 0.0 and 1.0. By plan, the NDVI itself subsequently fluctuates between - 1.0 and +1.0. NDVI is practically, yet not directly, comparable to the straightforward infrared/red proportion (NIR/VIS).

Code for NDVI:

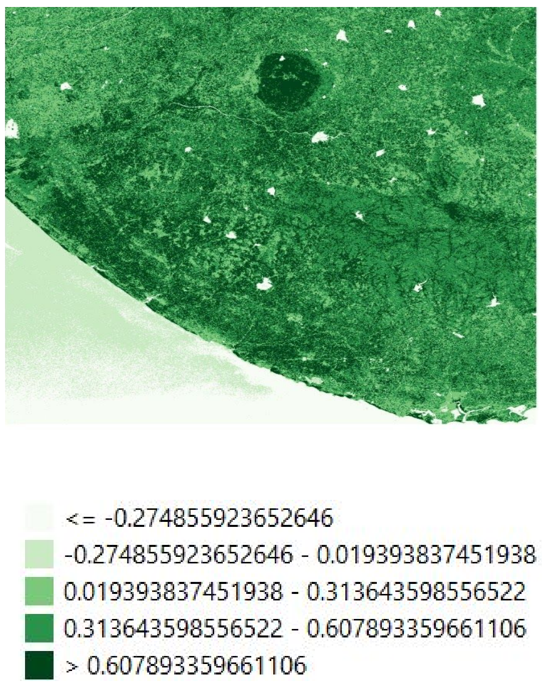
```
def NDVI(filename):
  # read Red(b4) and NIR(b8) as arrays
      red = b4.read()
      nir = b8.read()
      # Calculating ndvi
      ndvi = (nir.astype(float)-red.astype(float))/(nir+red)

      # Write the NDVI image into the given filename
      meta = b4.meta
      meta.update(driver='GTiff')
      meta.update(dtype=rio.float32)
      filename = filename + '.tif'
      with rio.open(filename, 'w', **meta) as dst:
          dst.write(ndvi.astype(rio.float32))
```

Output:

Original Image



NDVI Image



Applications: -

To determine the change in % of vegetation across a certain period based on the intensity of it over a certain area. Example time-lapse could help us to keep a record over depletion of vegetation.

NDWI(Normalized Difference Water Index) Images:

The Normalized Difference Water Index (**NDWI**) is known to be strongly related to the plant water content. It is, therefore, a very good proxy for plant water stress.

Remote sensing of land and the NDWI index can control irrigation in real-time, significantly improving agriculture, especially in areas were meeting the need for water is difficult.

NDWI = (green-nir)/(nir+green)

Nir: near-infrared band; Green: green band

Applications: -

To determine the change of % of water bodies across a certain period based on the intensity of it over a certain area. Example time-lapse could help us to keep a note on water-scarce.

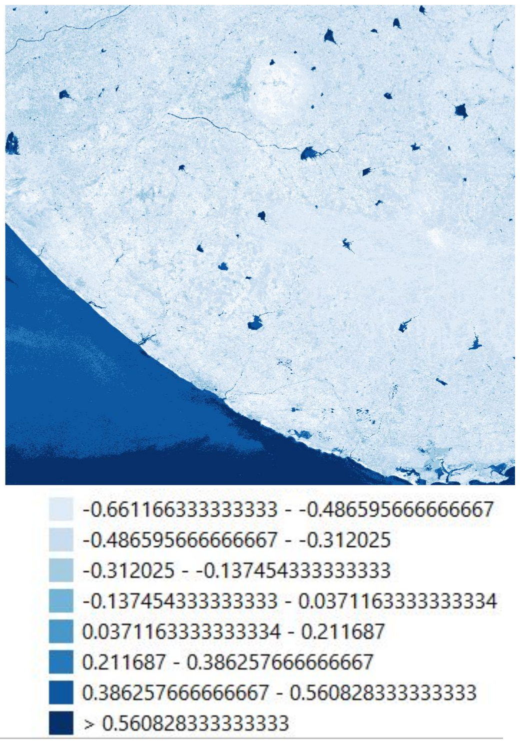
Code for NDWI:

```python
def NDWI(filename):
      # Calculating NDWI
      gre = b3.read()
      nir = b8.read()
      ndwi = (gre.astype(float) - nir.astype(float))/(nir+gre)
      # Write the NDWI image into the given filename
      meta1 = b8.meta
      meta1.update(driver='GTiff')
      meta1.update(dtype=rio.float32)
      filename = filename + '.tif'
      with rio.open(filename, 'w', **meta1) as dst:
          dst.write(ndwi.astype(rio.float32))
```

Output:

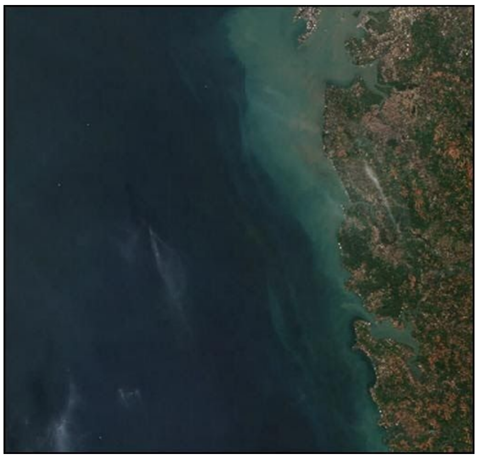
Original Image

NDWI Image



Different SWIR bands can be used to characterize the water absorption in the generalized form of NDWI. Note, that SWIR band with near-infrared gives the index to calculate the water content of leaves. while near-infrared with the green band gives the index to calculate the water content in different water bodies.

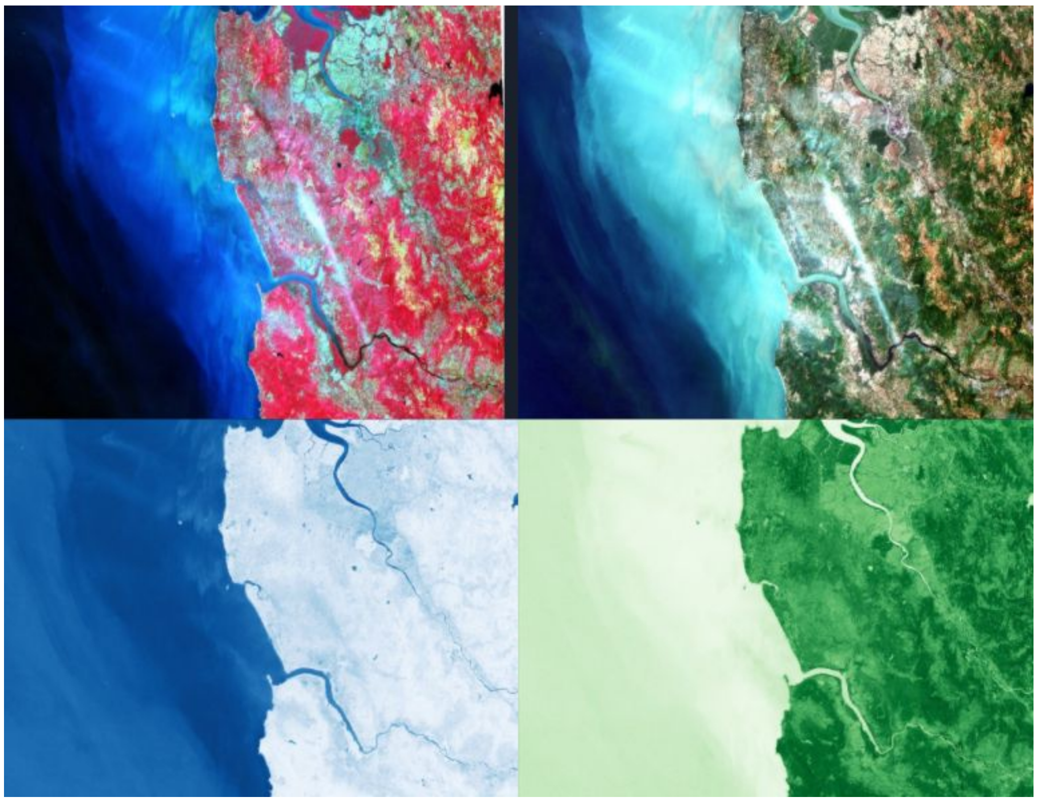
<u>Images of some other coordinates(Mumbai): NRG, RGB, NDWI, and NDVI Respectively:-</u>



Original Image (Image-2: Mumbai)

After all similar kinds of processing of the above image

Outputs:



Conclusion:-

All the images obtained after processing and enhancing are of much larger size(average 500-700 Mb) the more the size, the more accurate and precise image is obtained. Zooming in using the QGIS software gives great detailed clarity of the surroundings of the particular area selected. The rivers, vegetation can be seen in a clear amount, mountains can be spotted out and other similar data. One can use these images to analyse the topography of a particular area and can also analyse that area, its vegetation and water level and predict something or may take a decision based on it.

References:-

A. https://www.colby.edu/biology/BI352/Labs/satelliteim_info.pdf
B. https://eos.com/ndwi/
C. https://scihub.copernicus.eu/dhus/#/home
D. https://towardsdatascience.com/satellite-imagery-access-and-analysis-in-python-jupyter-notebooks-387971ece84b
E. https://medium.com/dataseries/satellite-imagery-analysis-with-python-a06eea5465ea
F. https://www.youtube.com/watch?v=j15MryznWn4
G. https://www.youtube.com/watch?v=txhjhjWqF7c
H. https://en.wikipedia.org/wiki/Normalized_difference_water_index
I. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-1-W2/43/2019/isprs-archives-XLII-1-W2-43-2019.pdf
J. http://web.sonoma.edu/users/f/freidel/techniques/exer/rem_sens/remsen_e.html#:~:text=Satellites%20record%20this%20energy%20after,identify%20objects%20through%20remote%20sensing.
K. http://edo.jrc.ec.europa.eu/documents/factsheets/factsheet_ndwi.pdf