FSSM
كلية العلوم السملالية
Faculté des Sciences Semlalia
Marrakech

Cady Ayyad University
Faculty of sciences Semlalia
Master in Data Science

Report of Machine Learning Project

Land cover classification and crop identification using satellite images

Academic Year 2021-2022

Contents Table

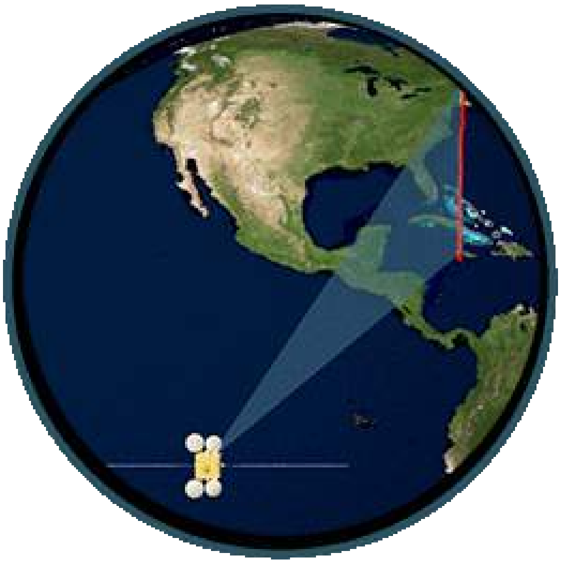
I. Introduction

Land cover classification and crops identification constitute an important economic part of many countries in the world and is one of the most commonly performed tasks in agro-environmental and geospatial research, whose domain ranges from physical geographic observations (i.e information on the land/crop type, land/crop area, configuration of buildings, roads, identifying surface water bodies, etc.) to environmental planning. Land cover classification has become a powerful tool in explaining relationships between ecological changes to socioeconomic activities. The recent advances in high-end computer programming and the availability of free satellite imagery have brought data science and remote sensing to the same platform. It is now possible to quickly perform land cover and crop classifications over a broader spatiotemporal spectrum using simplified machine learning or deep learning algorithms.

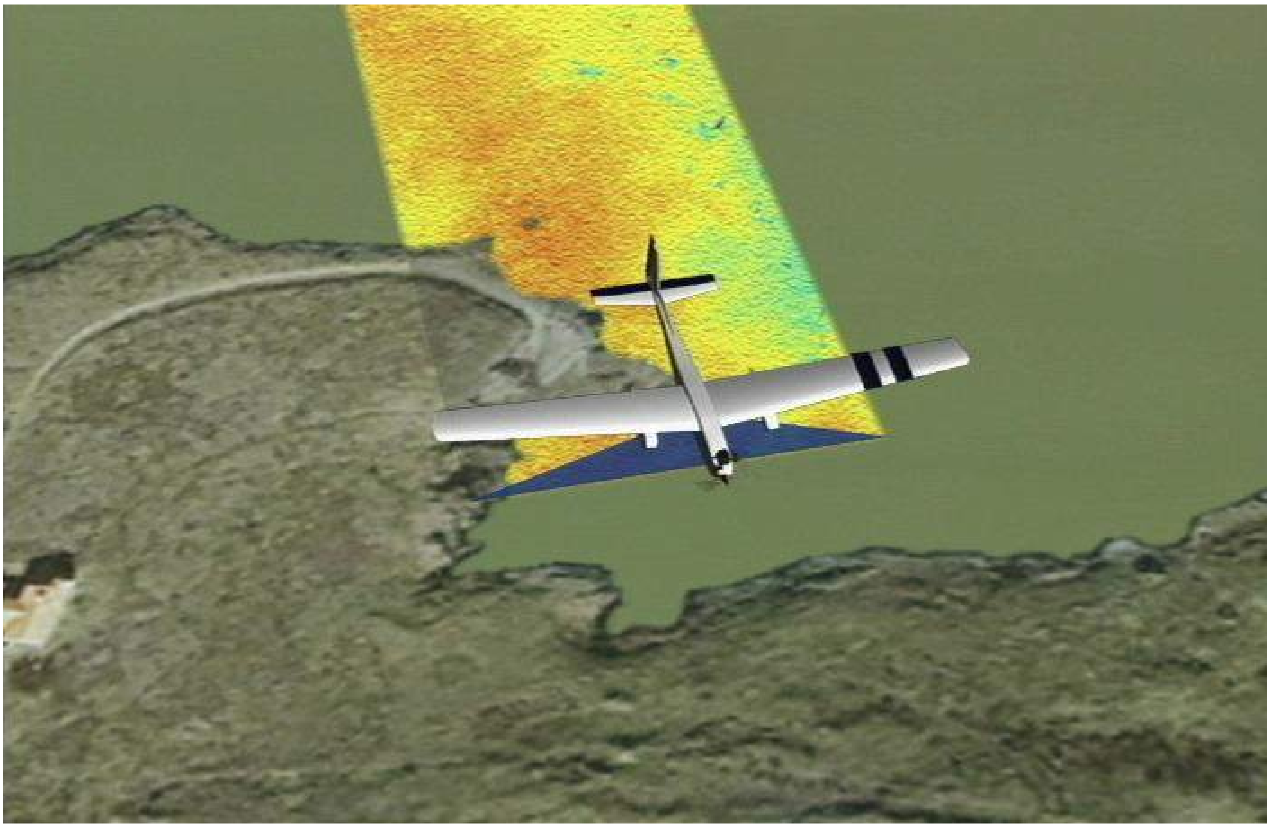
II. Preliminaries

1. Remote sensing:

Remote sensing is the acquisition of information by identifying and measuring an object without making physical contact with it. Remote sensing is used in different fields, like geography, land surveying, and most Earth science disciplines (for example, hydrology, ecology, meteorology, oceanography, glaciology, and geology); it also has military, intelligence, commercial, economic, planning, and humanitarian applications, among others. The term "remote sensing" generally refers to using satellite or aircraft-based sensor technologies to detect and classify objects on Earth by measuring their reflected and emitted radiation(propagated signals, or electromagnetic radiation) at a distance.

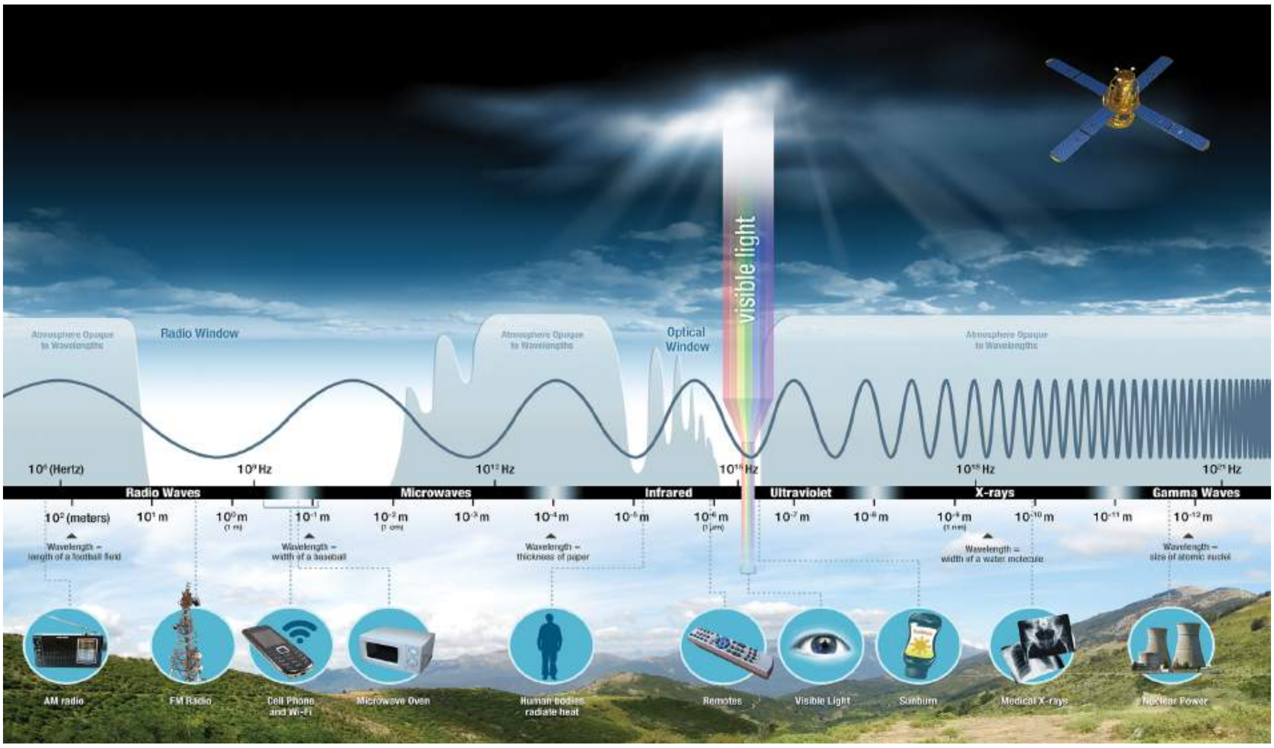


https://www.earthdata.nasa.gov/s3fs-public/2022-02/Air-Quality-Transparent-Blue.gif?VersionId=z3k6nWjVZXyNXN30iMu4GZnVDBVqHYZ0



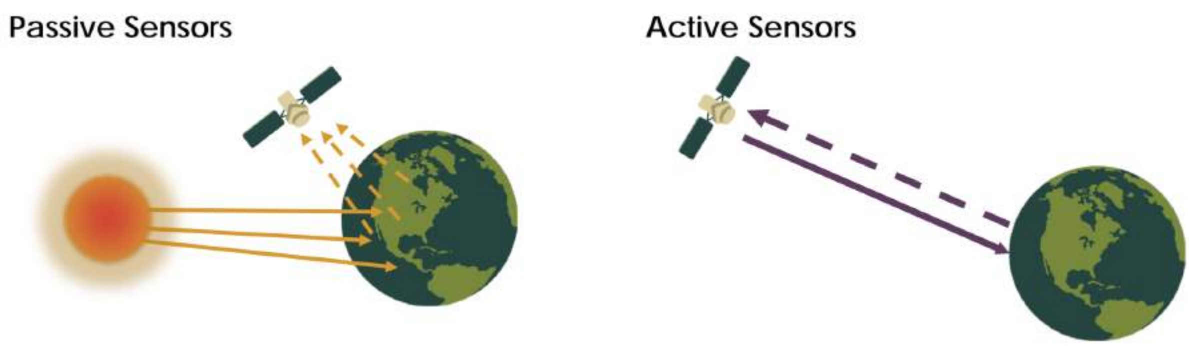
https://svs.gsfc.nasa.gov/vis/a000000/a002600/a002674/frame3.jpg

Many of the objects that make up the Earth's surface reflect and emit electromagnetic energy in unique ways.



http://www.earthdata.nasa.gov/s3fs-public/imported/EMS-Introduction_0.jpeg?VersionId=rSEKLCj0F2zuFCQ6hRTw6zaBBipH.UW

Remote sensing may be split into **"active"** remote sensing (when a signal is emitted by a satellite or aircraft to the object and its reflection detected by the sensor) and **"passive"** remote sensing (when the reflection of sunlight is detected by the sensor).
When Sensors use natural energy from the Sun we talk about **active remote sensing**, and **passive remote sensing** when those that provide their own source of energy.



http://www.earthdata.nasa.gov/s3fs-public/imported/activePassive.png?VersionId=LwgT4UZ4_eLF7Gks6AgKU7l___M7IXIh

2. Satellite Images

2.1 Analog and Digital images

The objects in a real scene can be represented by a two-dimensional image. Remote sensing images are representations of parts of the earth surface as seen from space. The images may be

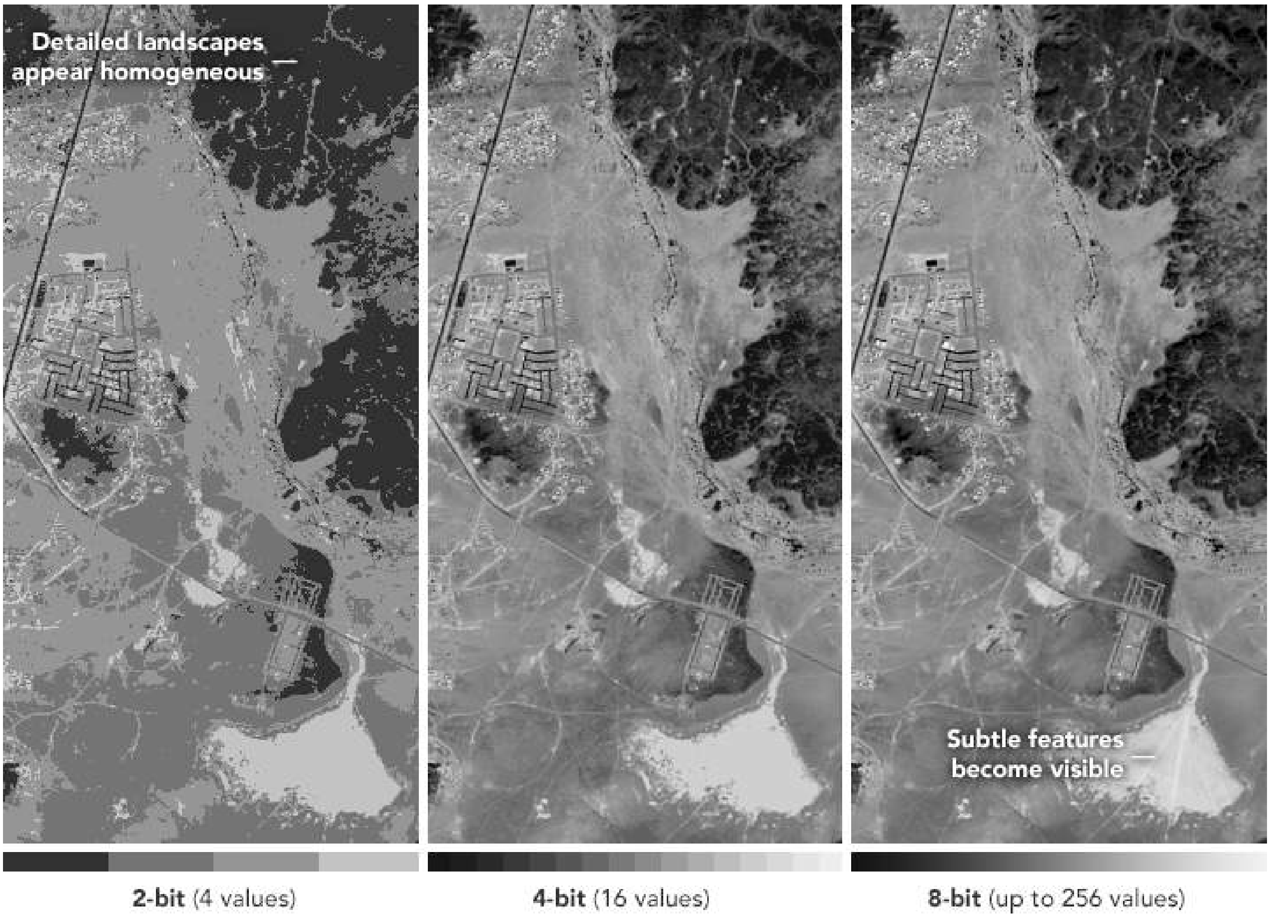
analog or **digital**. Satellite images(Earth observation imagery, spaceborne photography, or simply satellite photo) acquired using electronic sensors are examples of digital images.
In this project, we will use a digital image which is a two-dimensional array of pixels.

2.2 Pixels

In a digital image Each pixel has an **intensity value** and a **location address** (referenced by its row and column numbers).

Intensity value represents the measured electromagnetic energy or radiation that is reflected from the earth's surface back into space and stored as a digital number. This value is normally the average value for the whole ground area covered by the pixel.

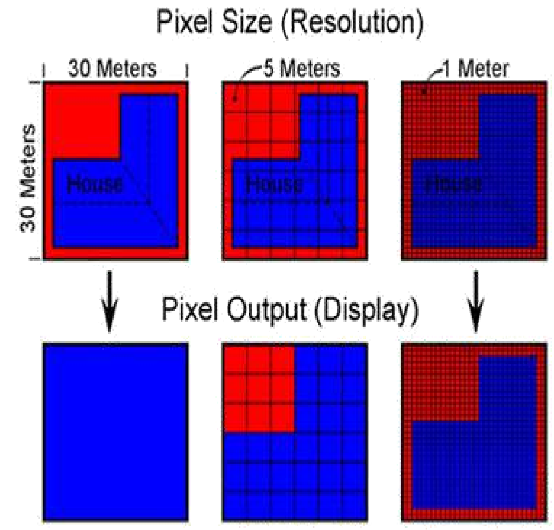
Radiometric resolution is the measure of a sensor's ability to record many levels of brightness, a digital number is stored with a finite number of bits due to the finite storage capacity, the greater the bit depth (number of data bits per pixel) of the images that a sensor records, the higher its radiometric resolution.

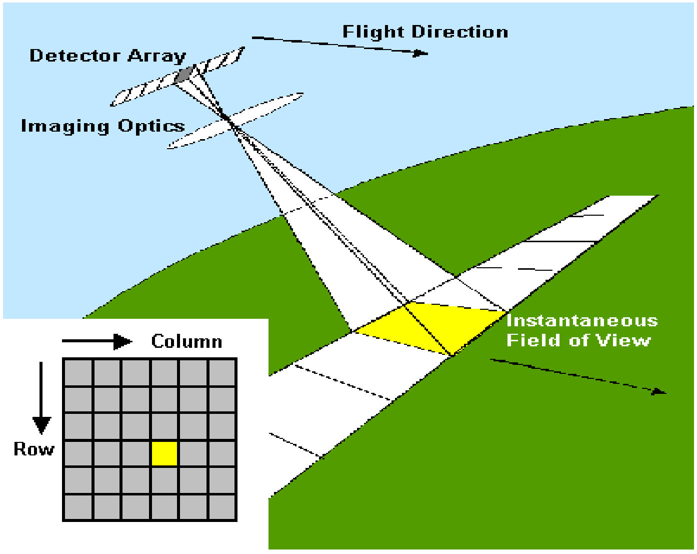


http://www.earthdata.nasa.gov/s3fs-public/2022-02/radiometric_resolution.png?VersionId=SUfbvvyRgjUqC1C5CoB2Br52GvwKq9iZ

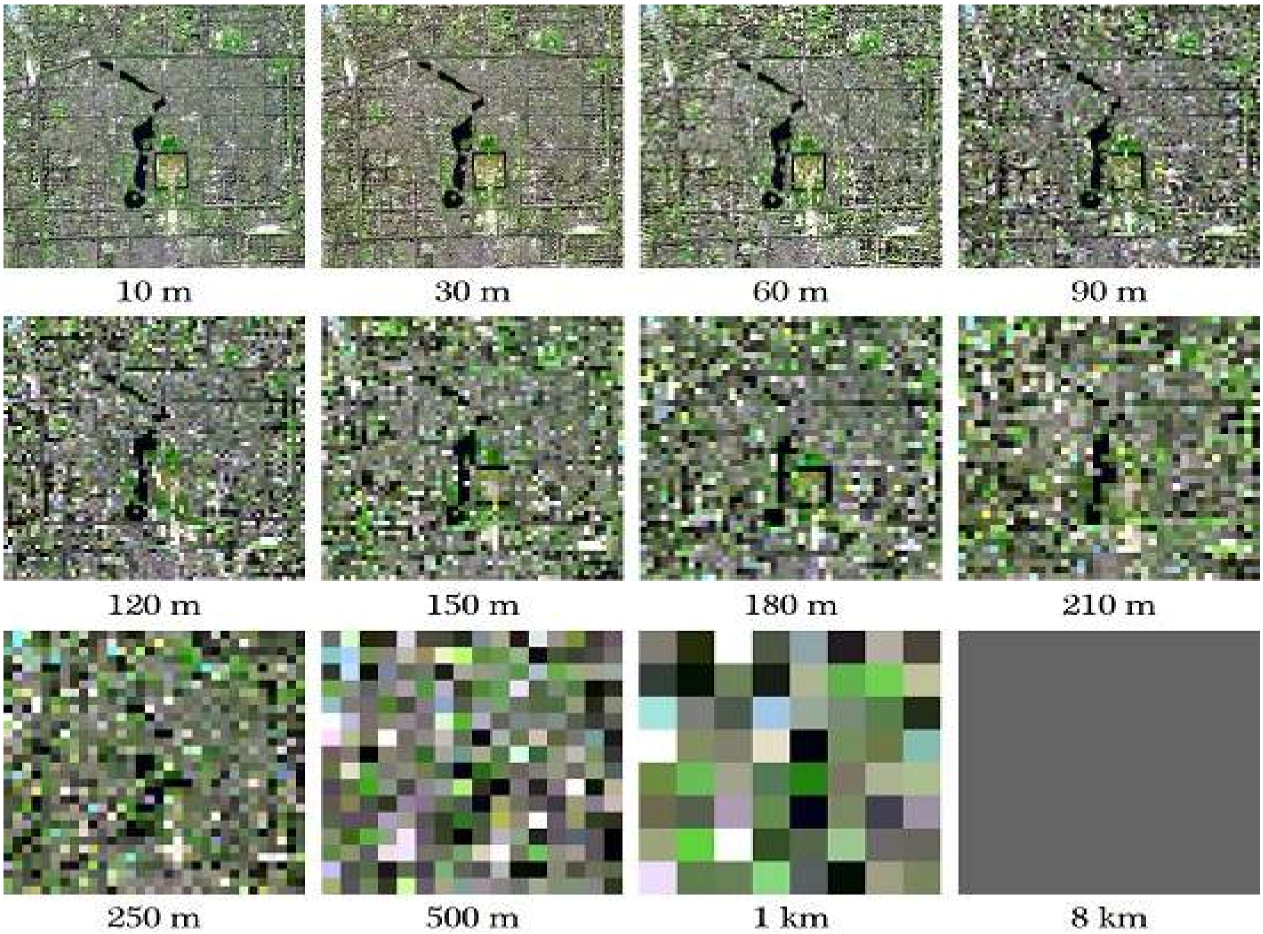
Spatial resolution refers to the size of the smallest object that can be resolved on the ground. Determined by the sensors' instantaneous field of view (**IFOV**), which is a measure of the ground area viewed by a single detector element in a given instant in time.

In a digital image, the resolution is limited by the pixel size, i.e. the smallest resolvable object cannot be smaller than the pixel size. The pixel size is determined by the sampling distance.



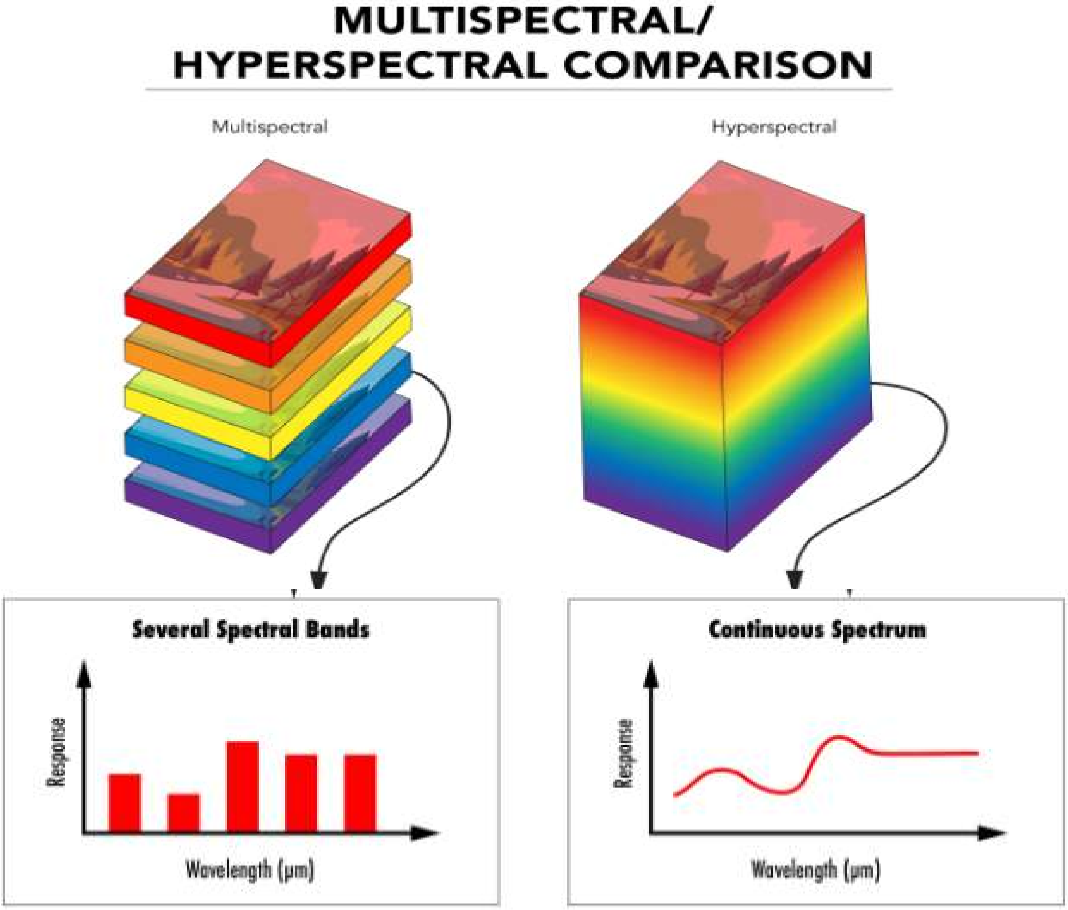


https://seos-project.eu/remotesensing/remotesensing-c03-p02.html https://crisp.nus.edu.sg/~research/tutorial/sweep.gif

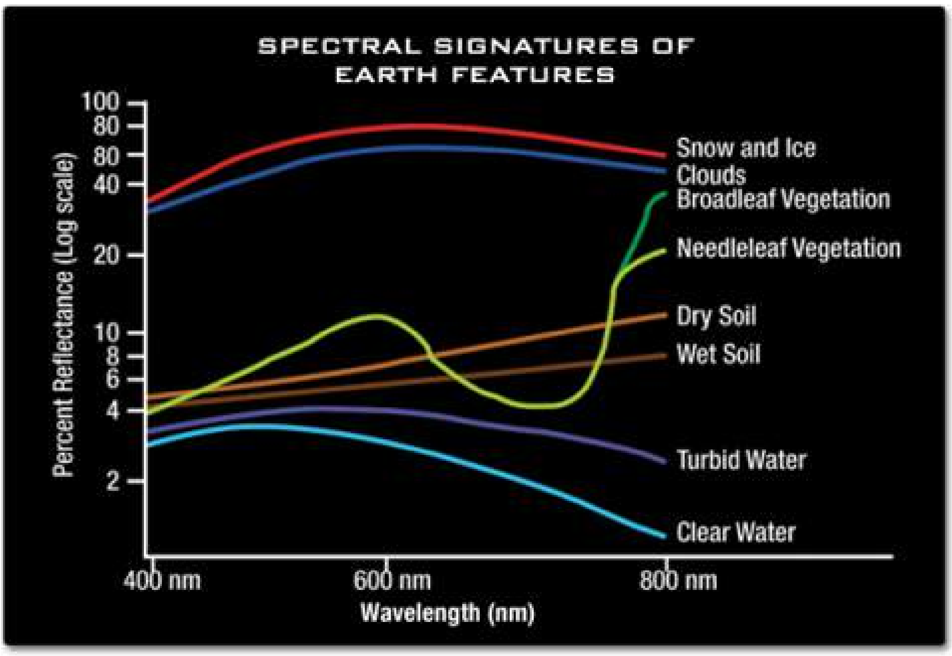


https://www.mdpi.com/2072-4292/12/1/117

Spectral resolution is the ability of a sensor to discern finer wavelengths, that is, having more and narrower bands.It is the wavelength interval size and number of intervals that the sensor is measuring.Many sensors are considered to be **multispectral**, meaning they **have 3-10 bands**. Some sensors have hundreds to even **thousands of bands** and are considered to be **hyperspectral**. The narrower the range of wavelengths for a given band, the finer the spectral resolution.



https://www.edmundoptics.com.sg/contentassets/520f2173de1e4ec482a7be8edcdece8a/figure4.jpg



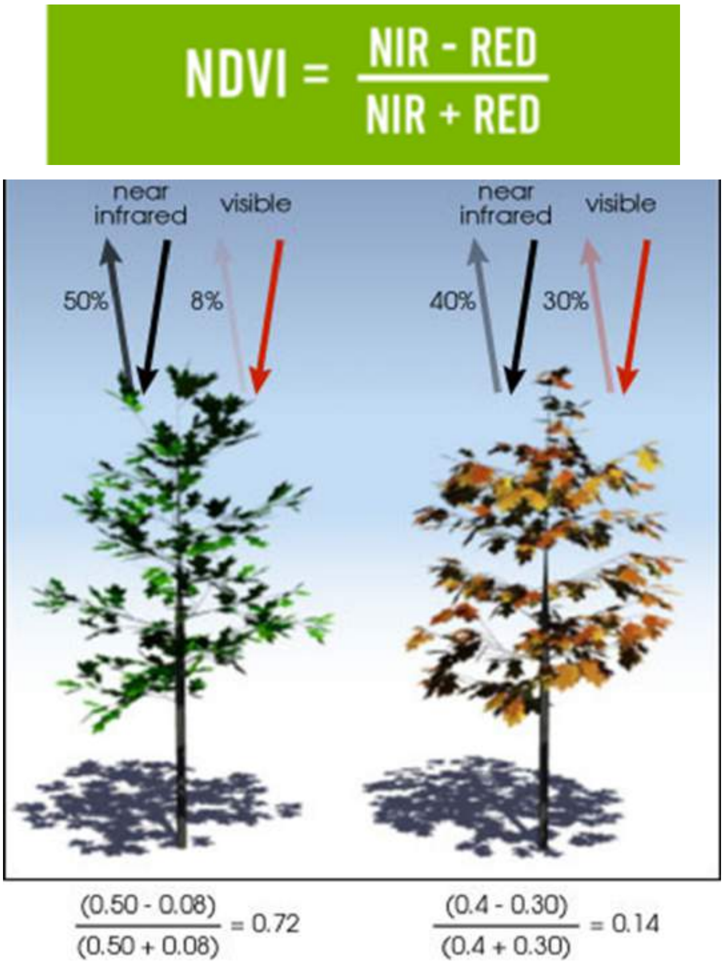
https://science.nasa.gov/files/science-red/s3fs-public/styles/large/public/thumbnails/image/visible-7.jpg?itok=vjkqkwHD

3. Vegetation indices

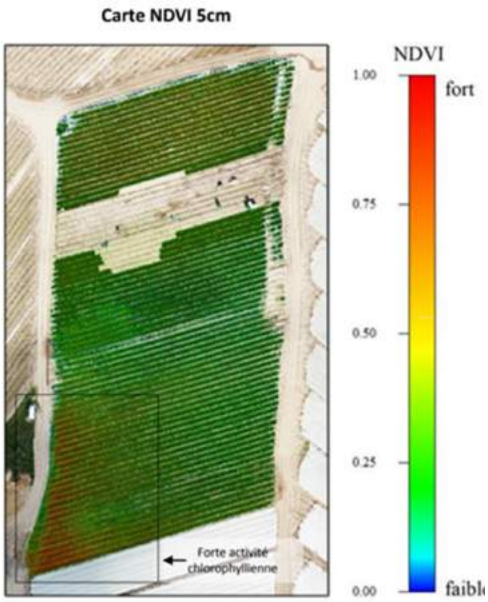
In remote sensing, indices are part of the processing methods called multispectral transformations. They consist of converting luminance measured at the satellite sensor into quantities that have a meaning in the environment.
Based on the multispectral character of satellite data, they can describe the state of a phenomenon. A vegetation index, for example, can indicate the stage of vegetation growth at a given time.
All indices, whether they are vegetation indices, soil indices, water column indices, etc., are based on an empirical approach based on experimental data. Vegetation indices are widely used on the one hand to identify and monitor vegetation dynamics, but also to estimate some biophysical parameters characteristic of plant cover, such as biomass, leaf area index, fraction of photosynthetically active radiation, etc.
for our case we used the following indices:
Usually, these indices are easy to adopt because they require less data.

3.1 NDVI

The NDVI (Normalized difference vegetation index) is the result of a calculation between near-infrared light NIR reflected by the vegetation and visible light R.The NDVI is the difference between the visible red band and the near infrared band.



This index is sensitive to the vigor and quantity of vegetation.
NDVI values range from -1 to +1, with negative values corresponding to surfaces other than vegetation covers, such as snow, water, or clouds for which reflectance in the red is greater than that of the near-infrared. For bare soil, the reflectance is about the same in the red and near infrared, and therefore the NDVI has values close to 0. Vegetation formations, on the other hand, have positive NDVI values, generally between 0.1 and 0.7.
The highest values correspond to the densest cover. dense cover.



NDVI maps are used in :

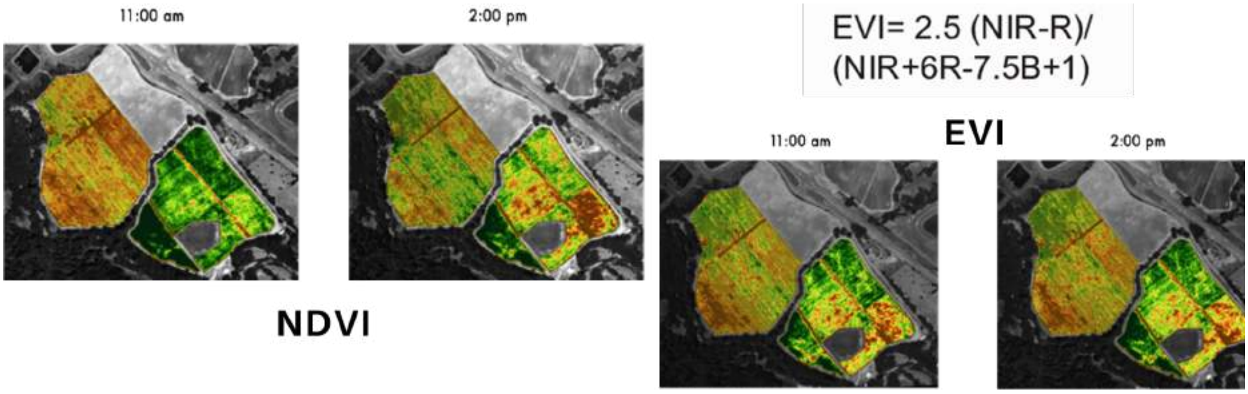
- Gathering information on variability in crop health:
- Identifying possible areas where crops are poor.
- Establishing the development status of crops.
- Detecting problem areas in order to make decisions.
- Differences in vegetation growth in a given area.
- Crop monitoring.
- Yield estimation and scouting.

Usually, NDVI values have a strong correlation in:

- Crop growth stages:
    - Therefore, it is an ideal way to determine the health of crops during the growing season.

3.2 EVI

The Enhanced Vegetation Index (EVI) is calculated in the same way as NDVI but uses additional wavelengths of light to correct for inaccuracies in NDVI, including variations in solar incidence angle, atmospheric conditions such as distortions of light reflected from particles in the air, as well as land cover signals below the vegetation.



[EVI ou NDVI : Quelle est la différence ? | VineView Blog]

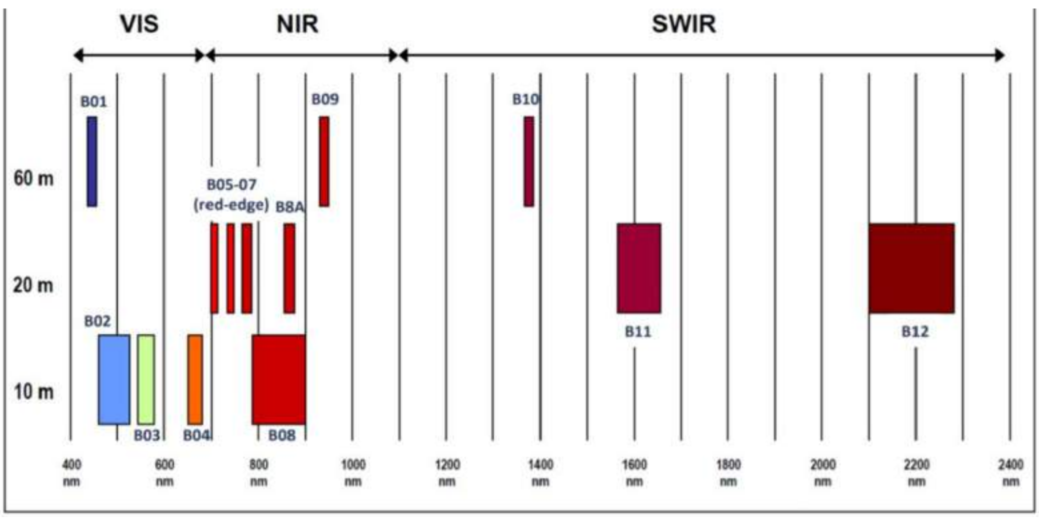
III. DataSet

1. Sentinel-2

SENTINEL-2 is a high-resolution multispectral imaging mission supporting Copernicus Land Monitoring studies, including monitoring of vegetation, land and water cover, and observation of inland waterways and coastal areas. The Sentinel-2 mission is organized around four major themes:

- Land Changes,
- Water Resources,
- Emergency and Hazard Mapping,
- Plant Health and Phenology.

The SENTINEL-2 multispectral instrument (MSI) has 13 spectral bands: four bands at 10 m, six bands at 20 m, and three bands at 60 m spatial resolution.



2. DataSet Of work

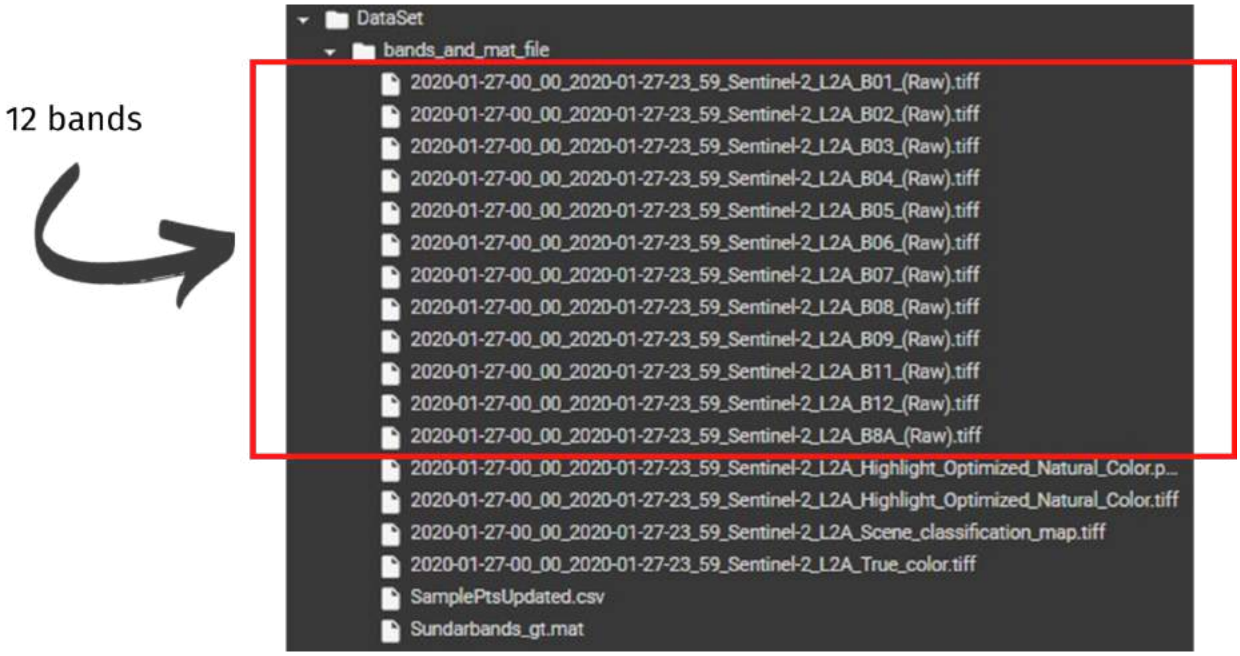
In this project, we worked with a very small part of the Sundarbans region for the task of analyzing satellite imagery.

The Sundarbans are one of the largest mangrove areas in the delta formed by the confluence of the Ganges, Brahmaputra and Meghna rivers in the Bay of Bengal. The Sundarbans Forest stretches approximately 10,000 km sq across India and Bangladesh, 40% of which is found in India and is home to many rare and globally threatened wildlife.

This portion of the Sundarbans satellite data is acquired using the Sentinel-2 satellite. The dataset is in the form of 954×298 pixels, with 12 bands with a spectral resolution varying from 10 to 60 meters.

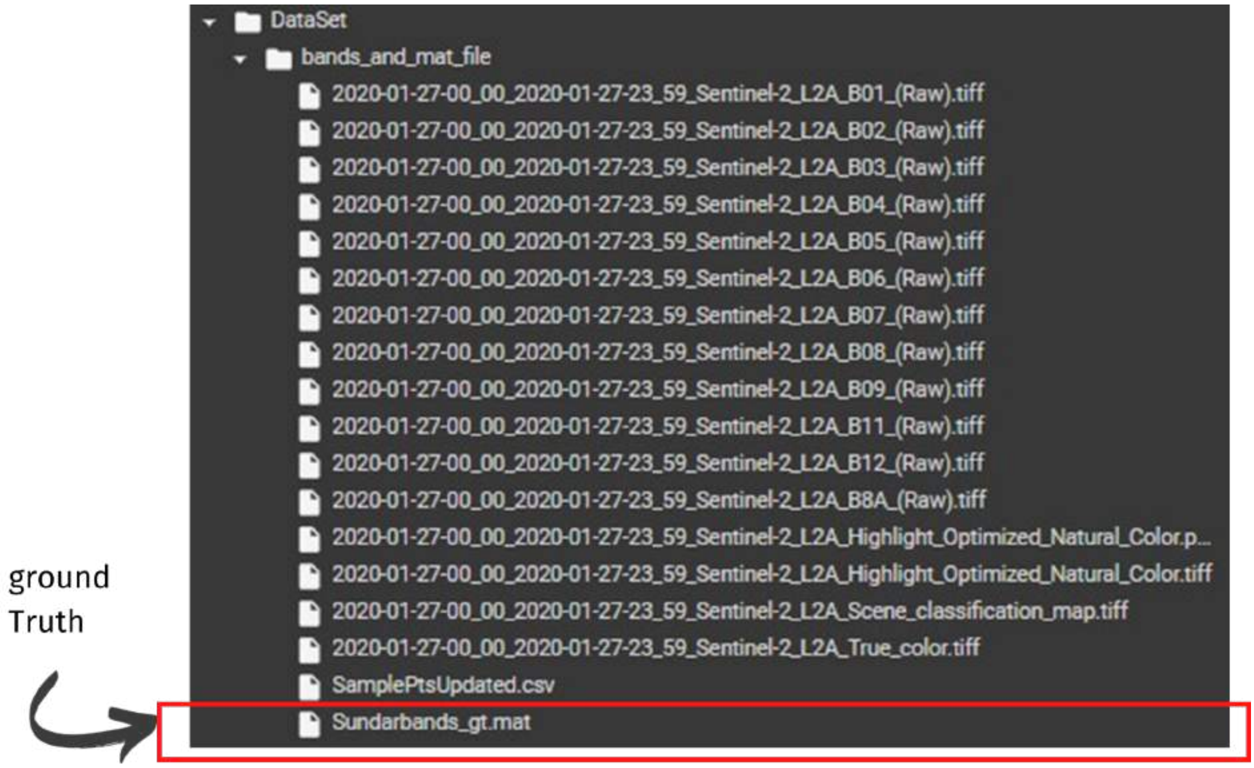
2.1 Bands:

Data Source:
https://github.com/syamkakarla98/Satellite_Imagery_Analysis/tree/main/Data/sundarbans_data

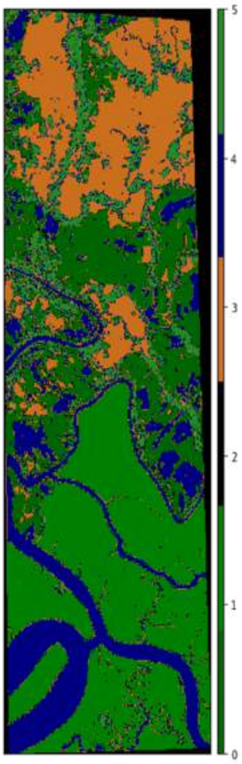


2.2 Ground Truth

It is the result of a process in which a pixel on a satellite image is compared to what is in reality (at the present time) to verify the content of the pixel on the image. That is why we use it in our project for the case of land classification.
Important in the initial supervised classification of an image. When the identity and location of land cover types are known through a combination of fieldwork, maps, and personal experience.
It also helps with atmospheric correction, as satellite images obviously have to pass through the atmosphere, they may be distorted due to absorption in the atmosphere.
Can therefore help to perfectly identify the objects in the satellite photos.



So after importing the ground truth file with extension (.mat) file. The result has the same shape as the bands we imported and each pixel contains the number of the class that exists in reality (trees, crops, water, bare_land, forest, or not identified).

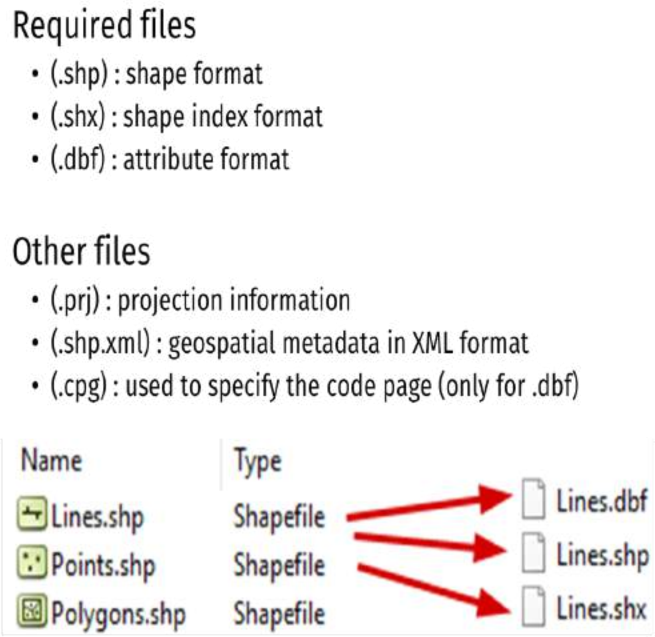


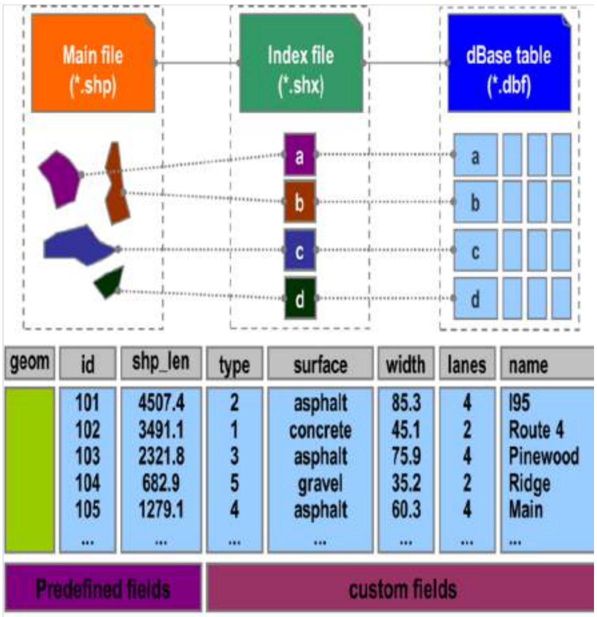
	pixel 0	pixel 1	pixel 2	pixel 3	pixel 4	pixel 5	pixel 6	pixel 7	pixel 8	pixel 9
0	0	0	0	0	0	0	4	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	3	0	0
3	0	0	0	0	0	0	0	3	5	0
4	0	0	4	0	0	0	0	0	5	0
...	...	...	...	...	...	...	...	...	...	...
949	2	2	2	2	2	2	2	4	4	4
950	2	2	2	2	2	2	2	4	4	4
951	2	2	2	2	2	2	2	4	4	4
952	2	2	2	2	2	2	2	4	4	4
953	2	2	2	2	2	2	2	4	4	4

954 rows × 298 columns

2.3 Shapefile:

For the case of crop classification we have used a shapefile which is a file format for geographic information systems (**GIS**) that allows us to archive the location, shape and attributes of geographic features.



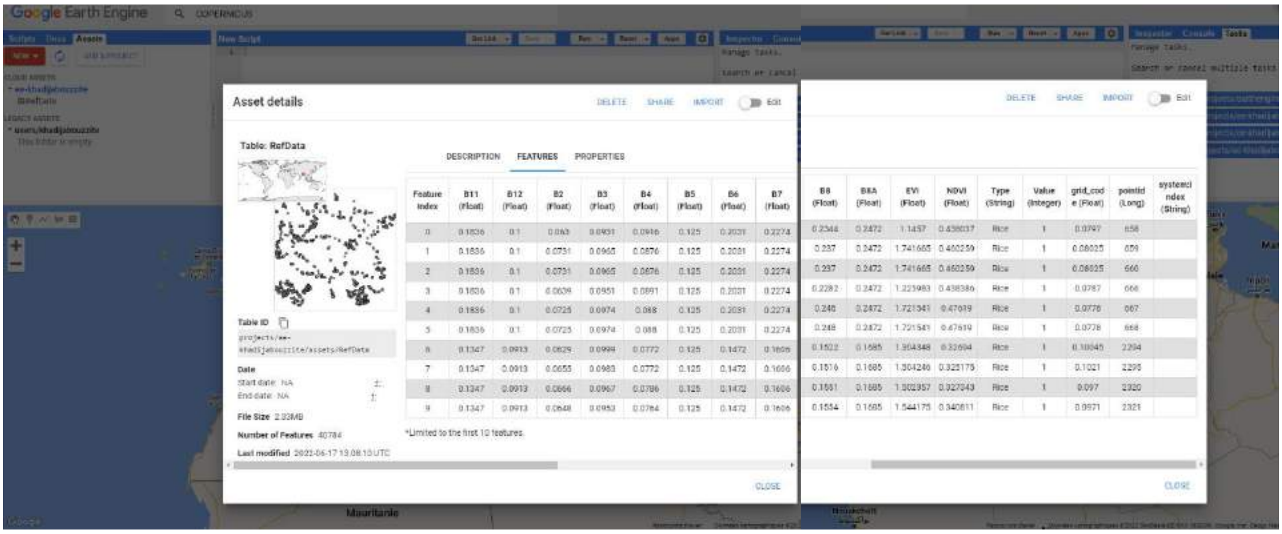


gisgeography.com transportgeography.org

for each geometry type (lines points Polygons etc) we have these files. The first 3 are mandatory for reading a shape file. Take the example of geometry type **“Polygons”**:
(.shp) file for the geometry of the entity itself, which will be indexed by **(.shx) file** to ensure a search forwards or backward in a fast way and the **(.dbf) file** contains the columnar attributes for each shape. **(.prj) file** for the coordinates in the projection system. **(.cpg)** file for text encoding (ex **UTF8**).

2.3.1 Google earth engine

Google Earth Engine is a cloud-based geospatial analysis platform that allows users to visualize and analyze global satellite images and geospatial datasets with planetary-scale analysis capabilities. Scientists and non-profit organizations use Earth Engine for remote sensing research, epidemic forecasting, natural resource management, etc.
In our case, we used it to read the shape file, after inserting all the extension files. The shape file as shown below contains the geometry type, the intensity of pixels of 12 bands, and the class of each pixel (Mustard, Lentil, Wheat, Maize, Potato , Rice, or Others).

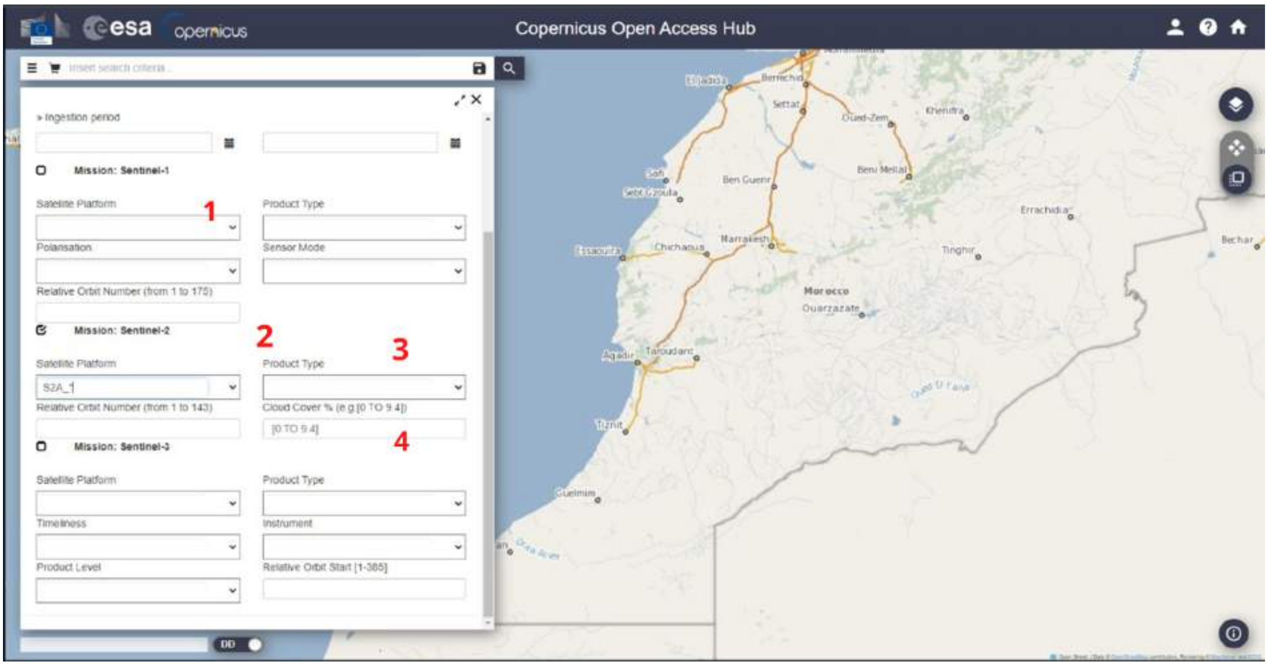


and we also use its API to extract the satellite images available in our area of interest, exploit them and we can use it for the mapping of the classification result directly on the map of the platform but in our case we found some problems with it.

2.4 Download the test Data

There are many sites for downloading satellite images. For the purpose of this project, we used Scihub, one of the most used programs for downloading satellite images.
Scihub is the official website of the **European Space Agency's Copernicus Programme**: (Earth observation program based on satellite and ground data).
The data are updated as soon as the satellite photo is received, and are free. The site is accessible at **scihub.copernicus.eu**.

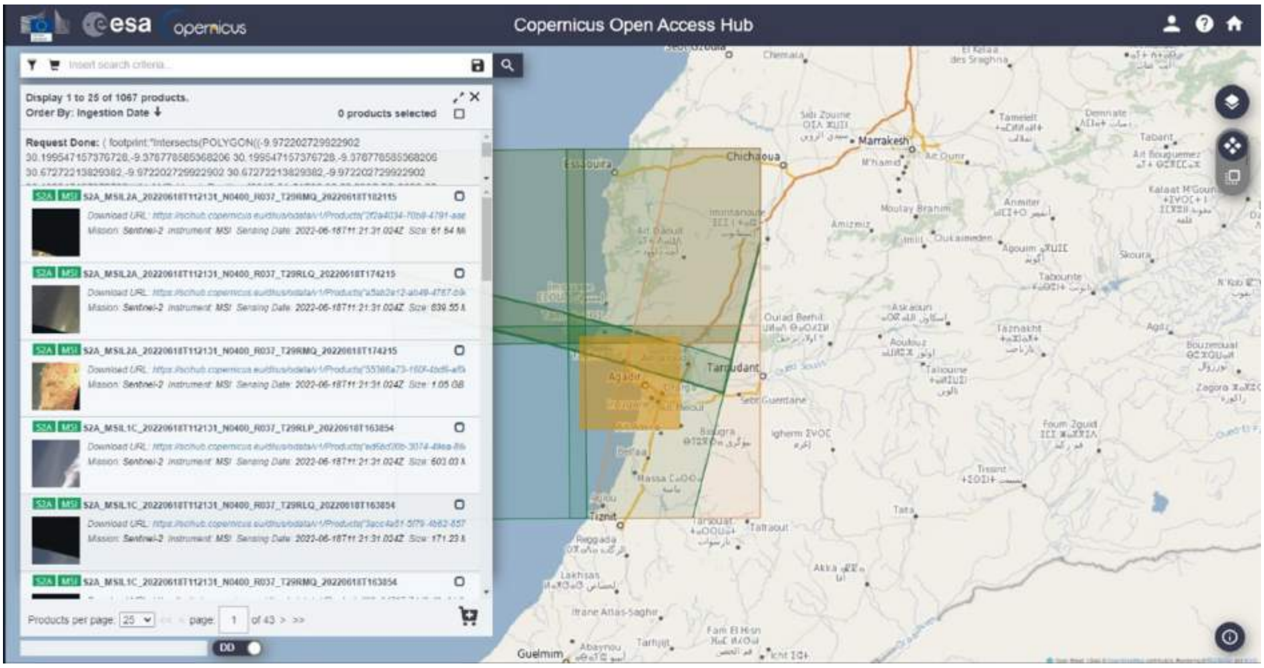
To access the images, we need to create an account. Then we have different settings in the interface to filter and download the data.



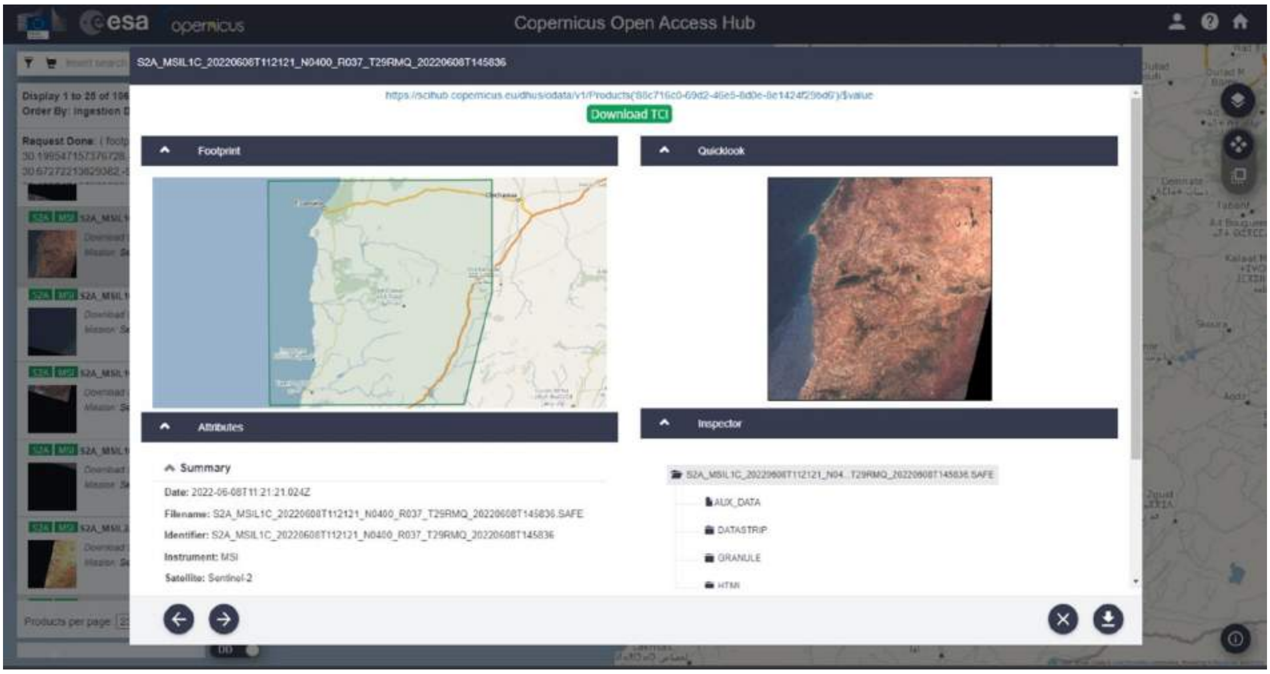
1 - Sensing period: allows to filter the images at a given period
2 - Mission: allows to choose the data according to the satellite: Sentinel 1, 2, or 3
3- Product type: allows to choose the processed or not images:

- 1C is an ortho-rectified image in TOA (Top-of-Atmosphere) reflectance with a cloud mask
- 2A is an ortho-rectified image in BOA (Bottom-of-Atmosphere) reflectance.

For Sentinel 2, the fourth field Cloud Coverage is used to filter out images with too much cloud coverage by selecting the area of interest and launching the search for images available in that area.
The results are displayed in a list, with information for each image.
The images are also displayed on the map.



Then we can choose the satellite images that suit us and download them.



IV. Practical demonstration

1. The Packages Used

In this project we need some special packages such as
ipyleaflet: it allows us to interact with the maps in the Jupyter notebook.

Rasterio: this is a highly useful module for raster processing that you can use for reading and writing several different raster formats in Python. Rasterio is based on GDAL and Python automatically registers all known GDAL drivers for reading supported formats when importing the module.

EarthPy: is a python package that makes it easier to plot and work with spatial raster and vector data using open-source tools. Earthpy depends upon geopandas which has a focus on vector data and rasterio which facilitates the input and output of raster data files. It also requires matplotlib for plotting operations.

2. The Code

2.1 Land Cover Classification

Step 1: upload the data

This code conducts us to upload the 11 bands for the study area.

```
# Data Directory
os.chdir('/content/drive/MyDrive/project 6/train')

# Read bands
sentinel_bands = glob("*B?*.tiff")
sentinel_bands.sort()

# Composite the bands
l = []
for i in sentinel_bands:
	with rio.open(i, 'r') as f:
	l.append(f.read(1))

# Data as array
arr_st = np.stack(l)
arr_st.shape
(11, 954, 298)
```

Step 2: upload the Ground Truth

The ground truth of the satellite image is read using the load mat method from the scipy.io package. The ground truth has 6 classes which include water, crops, trees, bare land, etc. which is two-dimensional (954 × 298).

```
# Ground Truth
y_data = loadmat('/content/drive/MyDrive/project 6/train/Sundarbands_gt.mat')['gt']
display(y_data)

array([[0, 0, 0, ..., 2, 2, 2],
       [0, 0, 0, ..., 2, 2, 2],
       [0, 0, 0, ..., 2, 2, 2],
       ...,
       [2, 2, 2, ..., 2, 2, 2],
       [2, 2, 2, ..., 2, 2, 2],
       [2, 2, 2, ..., 2, 2, 2]], dtype=int32)
```

Step 3: Data Visualization

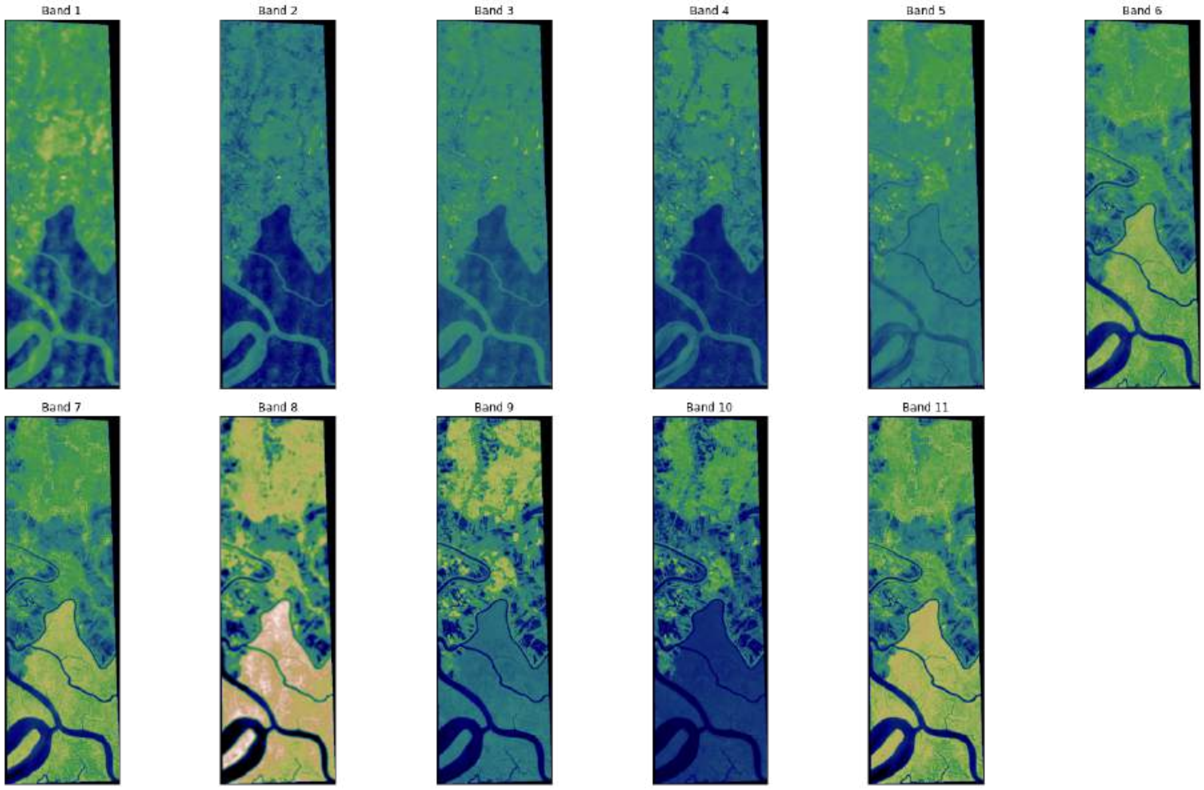
These Sundarbans data have multiple bands that contain data ranging from visible to infrared. So it is hard to visualize the data for humans. Creating an RGB Composite Image makes it easier to understand the data effectively. To plot RGB composite images, we have plotted the red, green, and blue bands, which are bands 4, 3, and 2, respectively. Since Python uses a zero-based index system, we need to subtract a value of 1 from each index. Therefore, the index for the red band is 3, green is 2, and blue is 1. Let's see the code to plot the RGB composite image along with the stretch applied.

RGB Composite Image of all bands:

```
ep.plot_rgb( arr_st,   rgb=(3, 2, 1),  stretch=True,   str_clip=0.02,  figsize=(12, 12),
title="RGB Composite Image",)
plt.show()
```

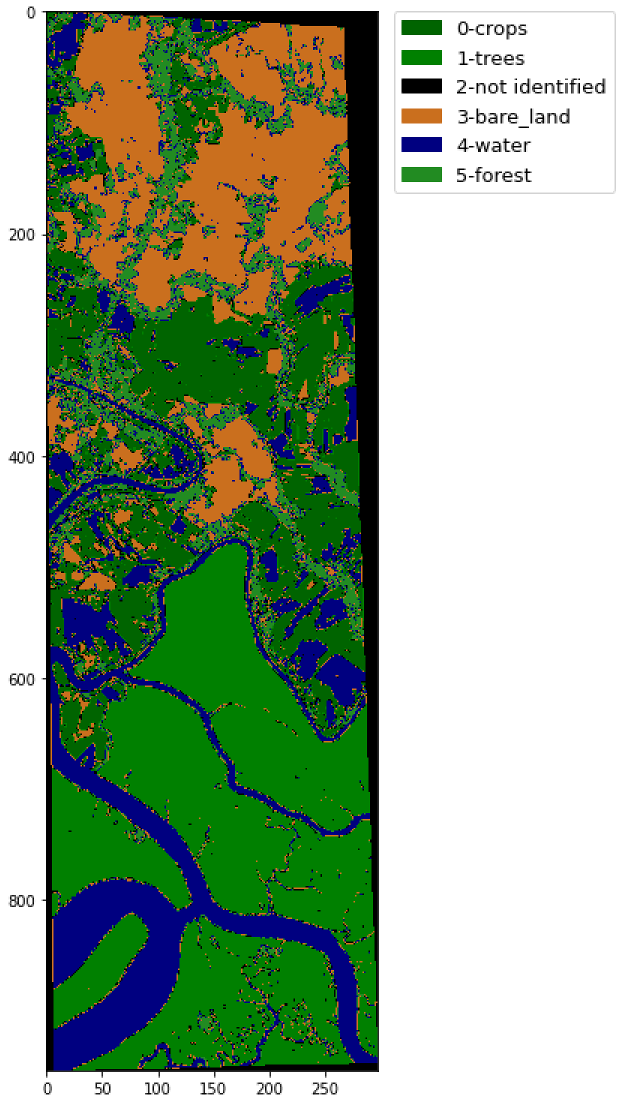
The 11 bands visualization:

```
ep.plot_bands(arr_st,  cmap = 'gist_earth',  figsize = (20, 12), cols = 6,  cbar = False)
plt.show()
```



Visualization of the ground truth:

```
cat_names = ["0-crops", "1-trees", "2-not identified", "3-bare_land", "4-water", "5-forest"]
f, ax = plt.subplots(figsize=(13, 13))
im_ax=ax.imshow(y_data, cmap=ListedColormap(['darkgreen', 'green', 'black', '#CA6F1E',
'navy', 'forestgreen']))
leg_neg = ep.draw_legend(im_ax = im_ax, titles = cat_names)
plt.show()
```



Step 4: Preprocessing

In this case we need to make the data in format were we have in the line every pixel of the image and in the column the different bounds, NDVi and EVI, and we need also to scale the data so here is the process:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

x = np.moveaxis(arr_st, 0, -1)

X_data = x.reshape(-1, 11)
scaler = StandardScaler().fit(X_data)
X_scaled = scaler.transform(X_data)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_data.ravel(),
                                                    test_size=0.20, stratify = y_data.ravel())
print(f"X_train Shape: {X_train.shape}\nX_test Shape: {X_test.shape}\ny_train Shape: {y_train.shape}\ny_test Shape:{y_test.shape}")
```

```
X_train Shape: (227433, 11)
X_test Shape: (56859, 11)
y_train Shape: (227433,)
y_test Shape:(56859,)
```

Step 5: Training the model

Since we have tabular data and multi-class problem, we choose to work with XGBoost.

Gradient boosting is a technique attracting attention for its prediction speed and accuracy, especially with large and complex data. Gradient boosting is a type of supervised machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model to minimize the error.

```
from xgboost import XGBClassifier
import xgboost as xgb
xgb_model  = XGBClassifier()
xgb_model.fit(X_train, y_train)

xgb_model_pred = xgb_model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, xgb_model_pred)*100}")

print(classification_report(y_test, xgb_model_pred))
```

```
Accuracy: 98.71436360118891
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     10814
           1       1.00      0.99      1.00     15714
           2       1.00      1.00      1.00      4064
           3       0.99      0.99      0.99     11193
           4       0.99      0.99      0.99      9030
           5       0.96      0.95      0.96      6044

    accuracy                           0.99     56859
   macro avg       0.99      0.99      0.99     56859
weighted avg       0.99      0.99      0.99     56859
```

Step 6: GridSearchCV

In order to choose the best hyperparameters for our model and the right estimator we used gridSearchCv which is a technique to search through the best parameter values from the given set of the grid of parameters. It is basically a cross-validation method. the model and the parameters are required to be fed in. Best parameter values are extracted and then the predictions are made.

```
from xgboost import cv

from xgboost import XGBClassifier
import xgboost as xgb
# use a full grid over all parameters
xgb_train=xgb.DMatrix(X_train, label=y_train)
DM_test =  xgb.DMatrix(data = X_test, label = y_test)

xgb_grid = {'n_estimators':[100,150,200,250],'max_depth': [8,10,15,20]}
xgb_model  = XGBClassifier()
xgb_grid_search = GridSearchCV(estimator = xgb_model, param_grid = xgb_grid, cv = 5, verbose = 1)

xgb_grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
GridSearchCV(cv=5, estimator=XGBClassifier(),
             param_grid={'max_depth': [8], 'n_estimators': [200]}, verbose=1)
```

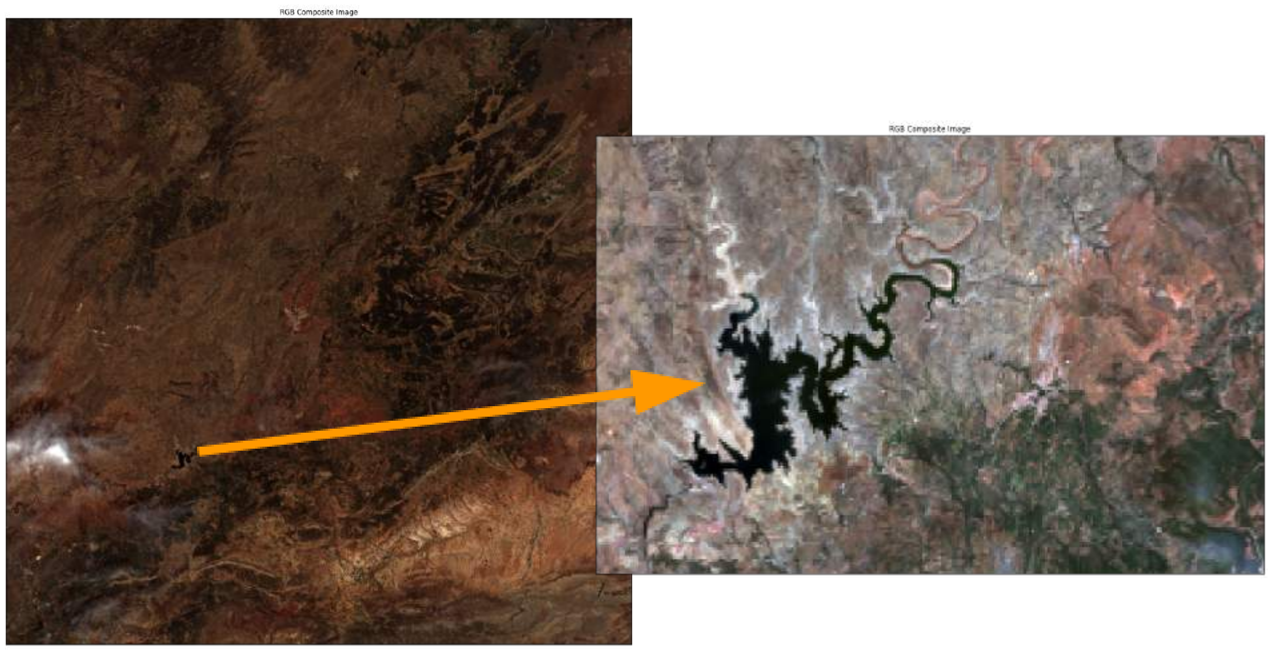
let's make a prediction:

For this step, in the test phase, we choose to upload bands of the region “BARRAGE EL HANSALI (Oum ER-Rbia River)” from European Space Agency's Copernicus Programme:

```
os.chdir('/content/drive/MyDrive/project 6/test/rbi333/R60m')

# Read bands
sentinel_bands = glob("*.jp2")
sentinel_bands.sort()

# Composite the bands
bands_arr = []
for i in sentinel_bands:
  with rio.open(i, 'r') as f:
    g=f.read(1)
    bands_arr.append(g)
    print(i,"has shape",g.shape)
    # Data as array
```

```
T30STB_20220508T105619_B01_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B02_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B03_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B04_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B05_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B06_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B07_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B09_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B11_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B12_60m.jp2 has shape (1830, 1830)
T30STB_20220508T105619_B8A_60m.jp2 has shape (1830, 1830)
```

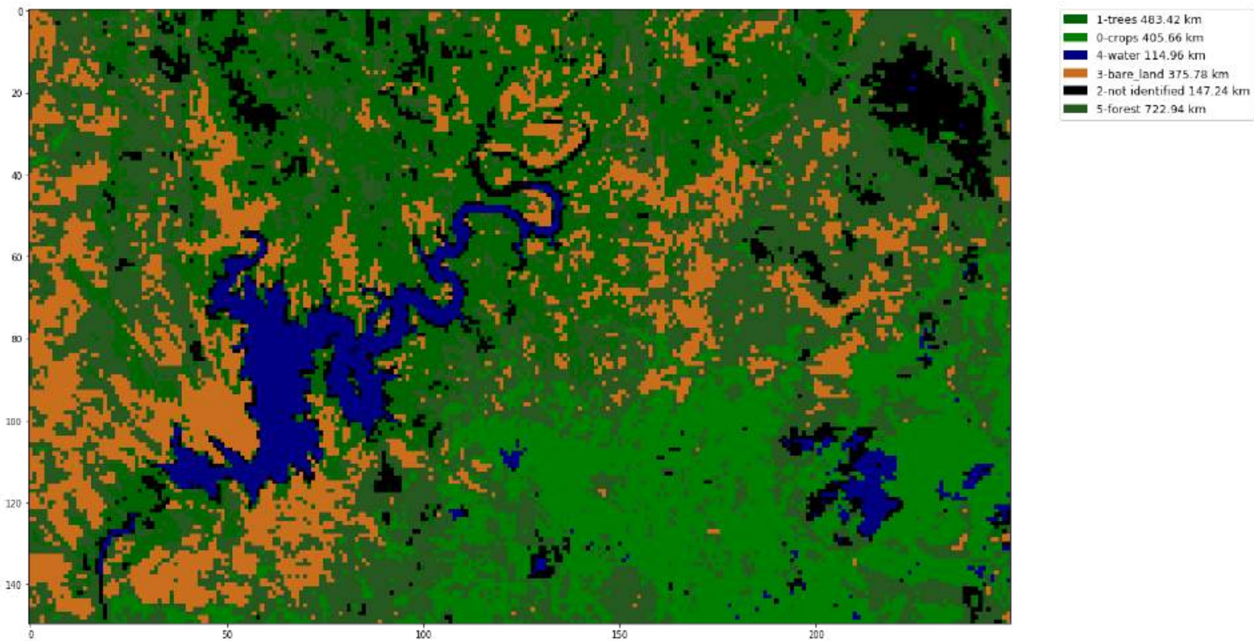
RGB Composite Image of all bands for this place



```
bands_arr=np.array(bands_arr)
list2=[]

for i in bands_arr:
  list2.append(i[:][:,450:700])

np.shape(list2)

bands_arr1=np.array(list2)

bands_arr1=bands_arr1[:][:,1200:1350]
bands_arr1.shape

bands_arr1 = np.stack(bands_arr1)
bands_arr1.shape

ep.plot_rgb(
    bands_arr1,
    rgb=(3, 2, 1),
    stretch=True,
    str_clip=0.02,
    figsize=(20, 20),
    title="RGB Composite Image",
)

plt.show()
```

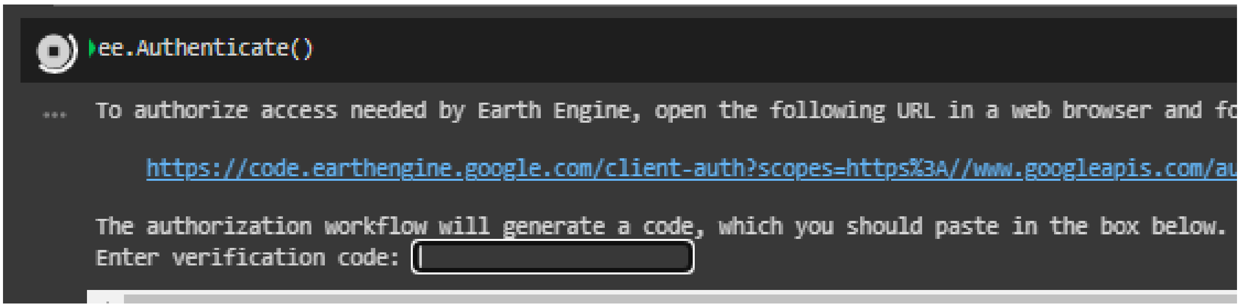
```
pred =loaded_model.predict(test)
```

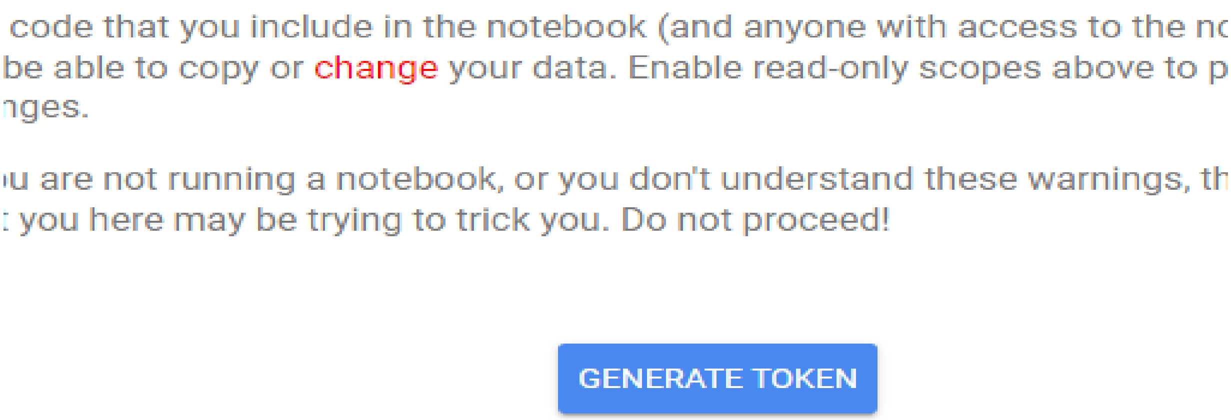
```
(unique, counts) = np.unique(pred, return_counts=True)
surface=(counts*60)/1000

cat_names = ["1-trees "+str(surface[0])+" km",\
             "0-crops "+str(surface[1])+" km",\
             "4-water "+str(surface[2])+" km", \
             "3-bare_land "+str(surface[3])+" km", \
             "2-not identified "+str(surface[4])+" km",\
             "5-forest "+str(surface[5])+" km"]

f, ax = plt.subplots(figsize=(20,20))
im_ax = ax.imshow(pred.reshape((bands_arr1.shape[1], bands_arr1.shape[2])),\
                  cmap=ListedColormap(['darkgreen', 'green', 'navy', '#CA6F1E', 'black', '#25591f']))
#CA6F1E
leg_neg = ep.draw_legend(im_ax = im_ax, titles = cat_names)
plt.show()
```



2.2 Crops Identification:

In this study, GEE has been used to complete the task along with python API.
First of all, we need to connect to the GEE with the help of API ee with the following steps.





edu.uiz.ac.ma

to continue to

Earth Engine Notebook Client - khadija.bouzzite@edu.uiz.ac.ma

Khadija BOUZZITE
khadija.bouzzite@edu.uiz.ac.ma

Sign in or provide access to Earth Engine Notebook Client - khadija.bouzzite@edu.uiz.ac.ma

To sign in or provide access:

1. Copy the authorization code from the Authorization code section.
2. Navigate to Earth Engine Notebook Client - khadija.bouzzite@edu.uiz.ac.ma.
3. Paste the authorization code on the Earth Engine Notebook Client - khadija.bouzzite@edu.uiz.ac.ma screen.

Authorization code

Please copy this code, switch to your application and paste it there:

4/1AX4XfWgEpDy6Nw0No5ZAShil_87r0YiSFJyzuP1ryAWJT
ZoJbFXtJTGNz-E

```
ee.Authenticate()
```

```
To authorize access needed by Earth Engine, open the following URL in

    https://code.earthengine.google.com/client-auth?scopes=https%3A//

The authorization workflow will generate a code, which you should pas
Enter verification code: 4/1AX4XfWgEpDy6Nw0No!
```

```
ee.Authenticate()
```

```
To authorize access needed by Earth Engine, open the following URL in a web browser and follow

    https://code.earthengine.google.com/client-auth?scopes=https%3A//www.googleapis.com/auth/ea

The authorization workflow will generate a code, which you should paste in the box below.
Enter verification code: 4/1AX4XfWgEpDy6Nw0No5ZAShil_87r0YiSFJyzuP1ryAWJTZoJbFXtJTGNz-E

Successfully saved authorization token.
```

Then Add the Study area shapefile from directory.

```
roi = ee.FeatureCollection("projects/ee-khadijabouzzite/assets/RefData") #
aoi = roi.geometry()       # Convert Feature Collecion to Geometry as String
```

Then add Earth Engine data based on the required filters (period, type of product, cloud contamination, etc).

```
# Monthly dataset collection
dataset = ee.ImageCollection("COPERNICUS/S2_SR")\
                .filterDate('2020-10-01', '2021-03-31')\
                .filterBounds(roi)\
                .filter('CLOUDY_PIXEL_PERCENTAGE <= 5')
```

Median Image Calculation:

Reduces an image collection by calculating the median of all values at each pixel across the stack of all matching bands. Bands are matched by name.

```
oct_mar_med = dataset.median().select('B2','B3','B4','B8','B5','B6','B7','B8A','B11','B12').clip(roi)
oct_mar_med
```

Indices Calculation:

NDVI:

```
oct_mar_ndvi = oct_mar_med.normalizedDifference(['B8', 'B4']).rename('ndvi').toDouble()
```

EVI:

```
oct_mar_evi = oct_mar_med.expression('2.5*((NIR-R)/(NIR+6*R+1-7.5*B))',{
                                     'NIR':oct_mar_med.select('B8'),
                                     'R':oct_mar_med.select('B4'),
                                     'B':oct_mar_med.select('B2')})\
                                     .rename('evi')
```

Then we add these indices to sour data with the following code.

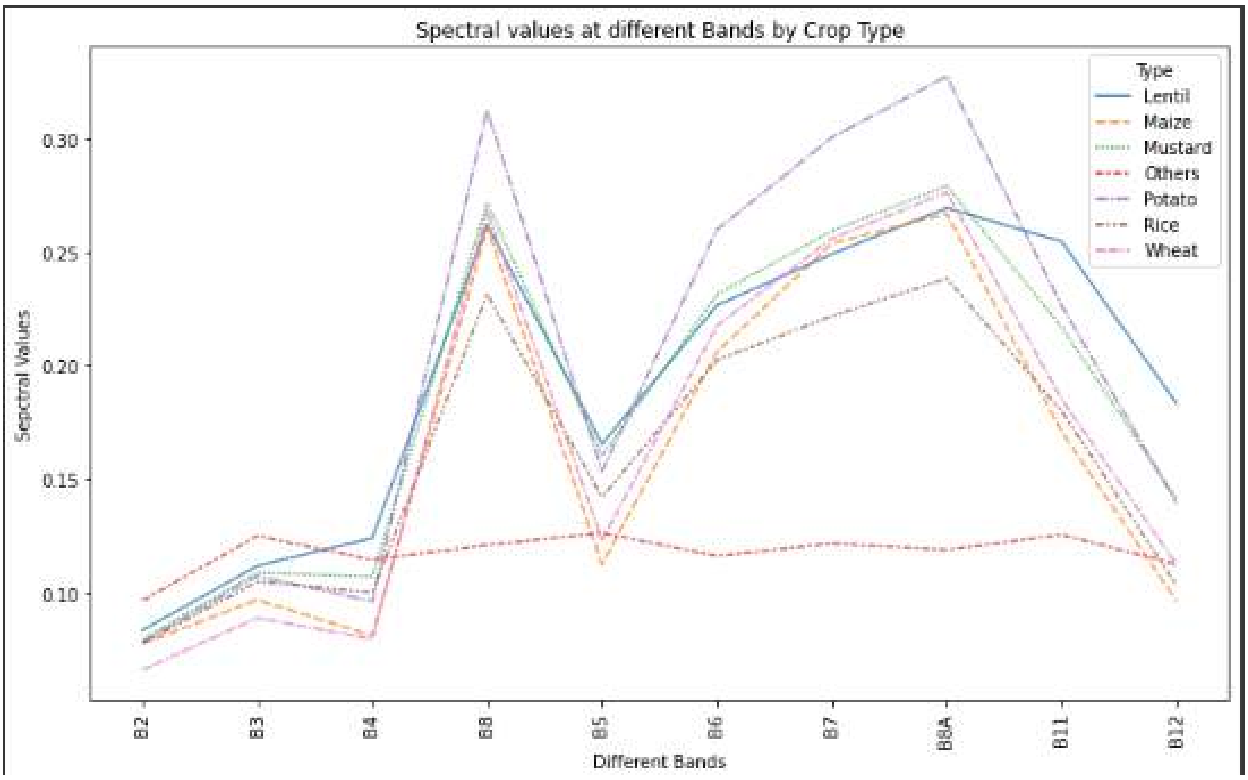
```
oct_mar_comp = oct_mar_ref.addBands(oct_mar_ndvi).addBands(oct_mar_evi)
```

Crop-specific spectral response has been analyzed below:

```
data1=data.drop(['Value','NDVI','EVI','pointid', 'grid_code'], axis=1) #
me_spec=data1.groupby(['Type']).mean()
me_spec_t=me_spec.transpose()
me_spec_t
```

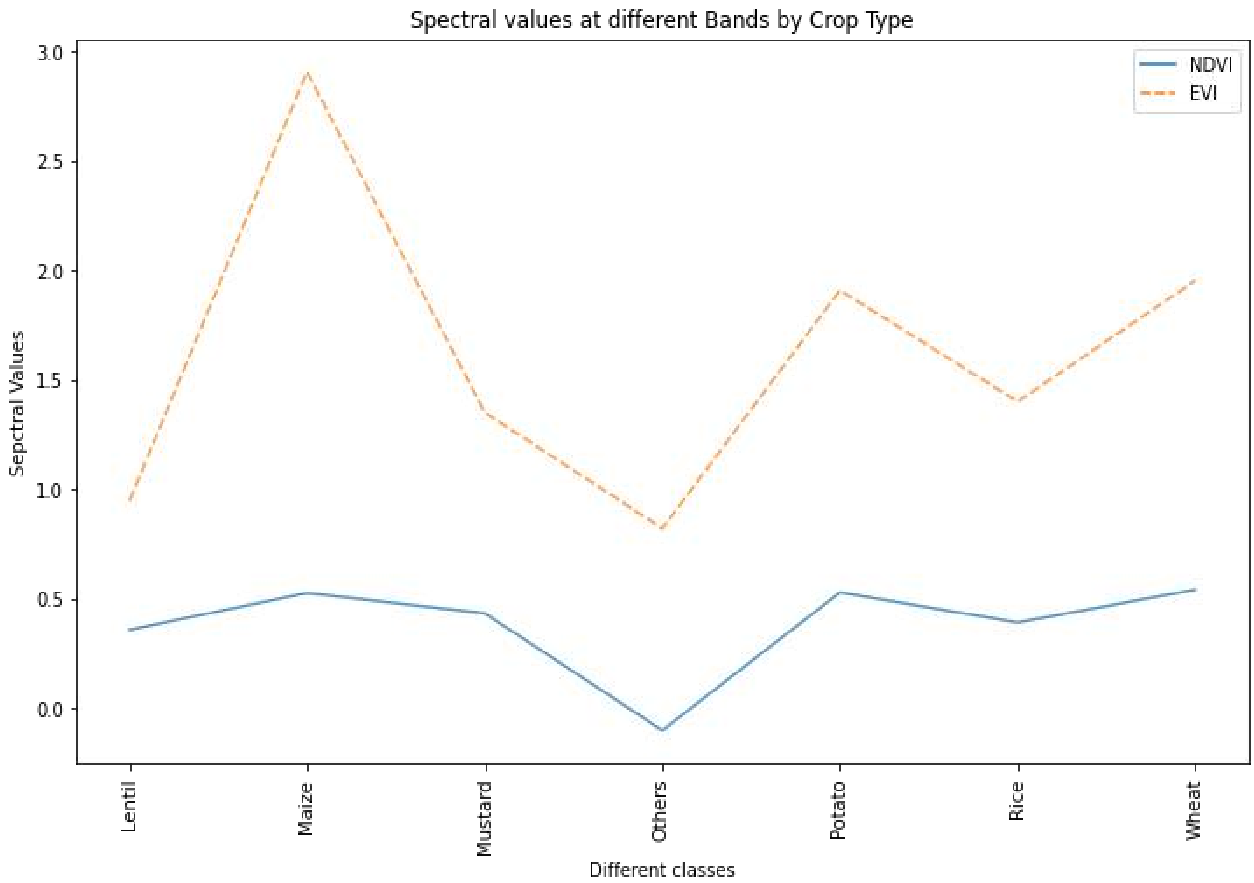
Type	Lentil	Maize	Mustard	Others	Potato	Rice	Wheat
B2	0.083534	0.077572	0.079573	0.096671	0.078404	0.077699	0.066113
B3	0.111863	0.096939	0.108807	0.125035	0.107411	0.104933	0.088976
B4	0.124080	0.081127	0.107191	0.114570	0.096299	0.100092	0.079622
B8	0.262024	0.260879	0.271557	0.121021	0.312261	0.231420	0.267996
B5	0.165522	0.112621	0.160313	0.126355	0.153658	0.142223	0.123366
B6	0.226557	0.205998	0.231341	0.116221	0.259868	0.202418	0.217591
B7	0.249095	0.253787	0.259280	0.121744	0.300461	0.221817	0.255961
B8A	0.269411	0.266923	0.279192	0.118926	0.327353	0.238328	0.276281
B11	0.254620	0.170973	0.216983	0.125592	0.226354	0.179965	0.185085
B12	0.183351	0.096306	0.141231	0.112544	0.139435	0.103225	0.112439

```
plt.figure(figsize=(12,7))
sns.lineplot(data=me_spec_t, alpha=0.75)
plt.title("Spectral values at different Bands by Crop Type")
plt.xlabel("Different Bands")
plt.ylabel("Sepctral Values")
plt.xticks(rotation=90)

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



As we see here every type of crops has specific spectral

Type	NDVI	EVI
Lentil	0.357680	0.946214
Maize	0.525370	2.906864
Mustard	0.432894	1.350845
Others	-0.101686	0.820563
Potato	0.527948	1.908365
Rice	0.390974	1.400188
Wheat	0.540792	1.952659



Here is the data of the first classification problem after preprocessing:

	pointid	grid_code	Type	Value	B2	B3	B4	B8	B5	B6	B7	B8A	B11	B12	NDVI	EVI
0	312	0.08075	Rice	1	0.0666	0.0850	0.0897	0.1554	0.1125	0.1356	0.1504	0.1682	0.1599	0.0896	0.268054	0.845778
1	313	0.08070	Rice	1	0.0660	0.0830	0.0915	0.1724	0.1120	0.1379	0.1514	0.1694	0.1616	0.0926	0.306556	0.892936
2	314	0.08075	Rice	1	0.0662	0.0852	0.0902	0.1665	0.1120	0.1379	0.1514	0.1694	0.1616	0.0926	0.297234	0.902745
3	315	0.08085	Rice	1	0.0672	0.0851	0.0910	0.1681	0.1125	0.1356	0.1504	0.1682	0.1599	0.0896	0.297569	0.916984
4	316	0.08145	Rice	1	0.0653	0.0865	0.0926	0.1654	0.1125	0.1356	0.1504	0.1682	0.1599	0.0896	0.282171	0.786687

Since in this project we focused on the crops, So we need to drop the irrelevant bands, those that haven’t a vegetation impact, and this is the final data format:

```
df_model = pd.get_dummies(data[['pointid', 'grid_code', 'Type', 'Value', 'B2', 'B3', 'B4', 'B8', 'B5', 'B6', 'B7', 'B8A', 'B11', 'B12', 'NDVI', 'EVI']])
# df_model.tail()
df_model.drop(columns=['pointid', 'grid_code'],inplace=True)
# df_model.tail()
y = df_model.pop('Value')

df_model = df_model.loc[:, ['B2', 'B3', 'B4', 'B8','NDVI', 'EVI']] #Model_Scheme (RGB + NIR Bands + Indices)
X = df_model
X.columns
```

```
Index(['B2', 'B3', 'B4', 'B8', 'NDVI', 'EVI'], dtype='object')
```

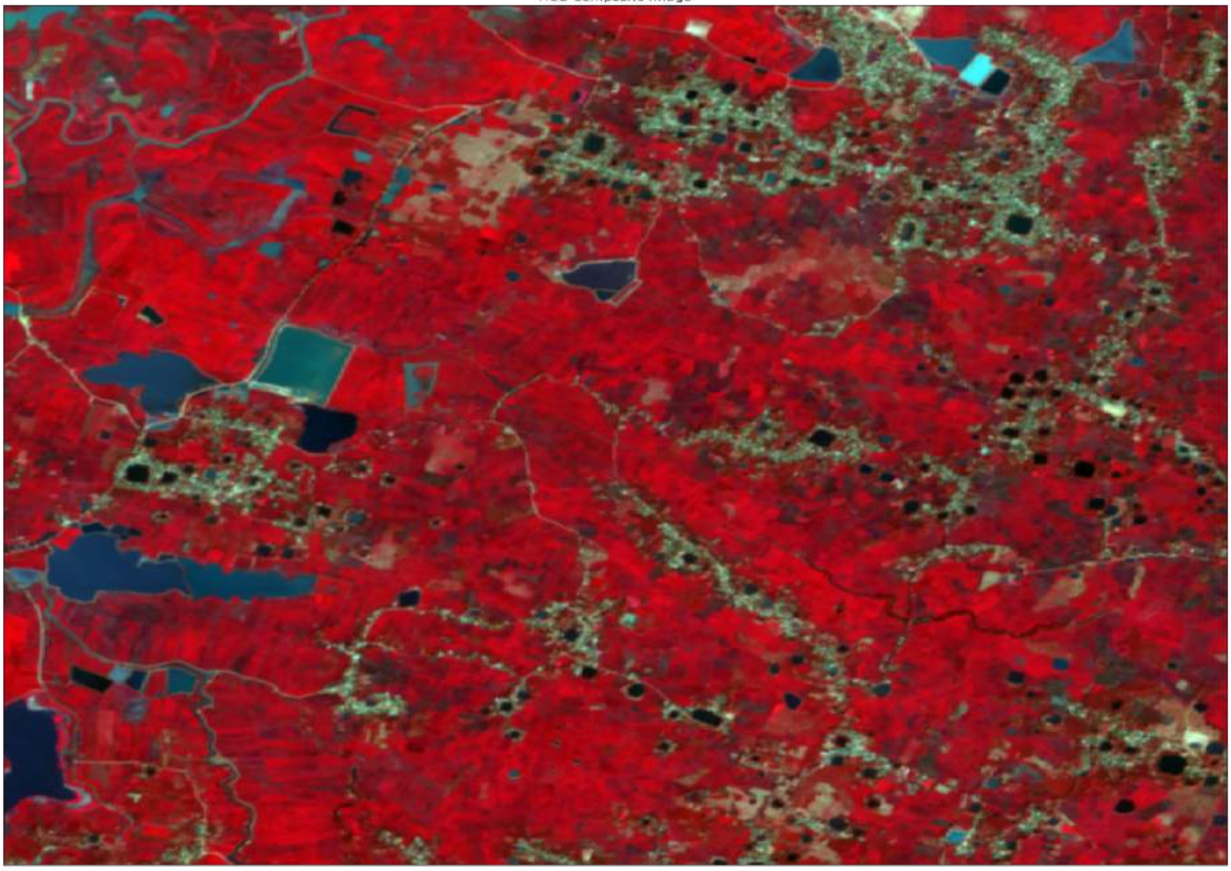
In this classification problem we used random forest classifier

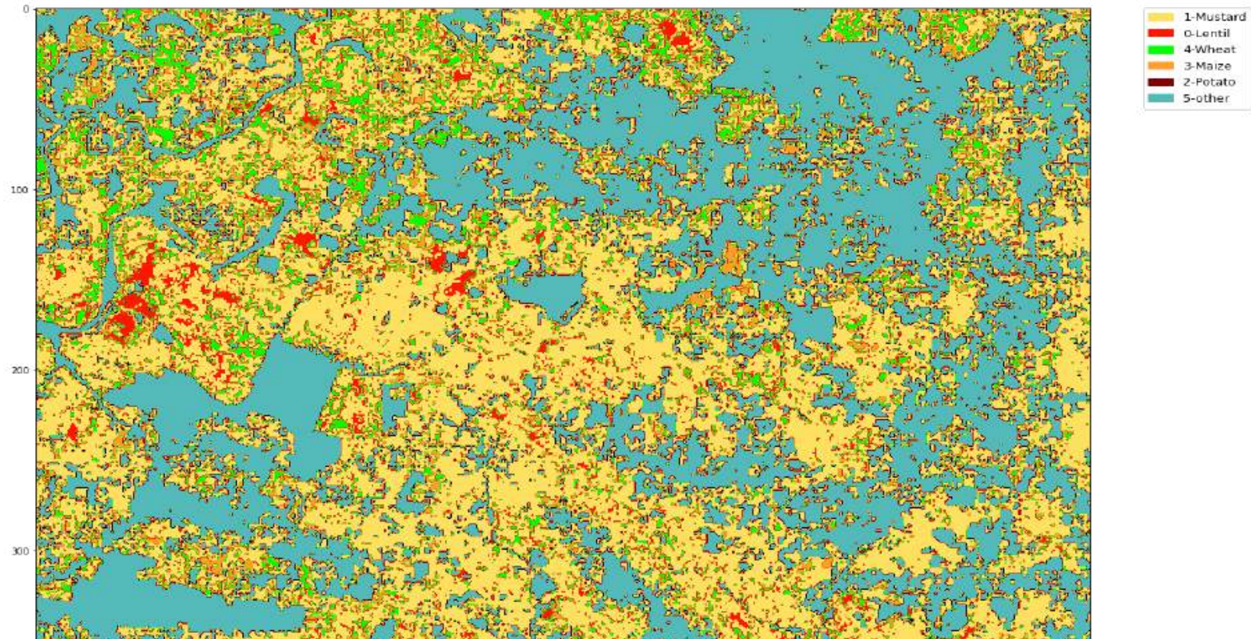
```
[ ] from sklearn.ensemble import RandomForestClassifier

    rf = RandomForestClassifier(n_estimators=50) # Try with different number of estimators
    rf = rf.fit(X_train, y_train.values.ravel())

    # Prediction Using Random Forest Classifier
    from sklearn.metrics import accuracy_score, confusion_matrix

    Y_pred_rf = rf.predict(X_test)

    Class_Acc_RF4 = accuracy_score(y_test, Y_pred_rf)*100
    print(Class_Acc_RF4)
    # print("Classification accuracy of RF is", Class_Acc_RF)
    print(classification_report(y_test, Y_pred_rf))
```

```
93.34749918273945
              precision    recall  f1-score   support

           1       0.88      0.93      0.91      3335
           2       0.74      0.66      0.70       210
           3       0.92      0.93      0.92      1748
           4       0.86      0.85      0.85       808
           5       0.71      0.16      0.26        62
           6       0.68      0.35      0.47       153
           7       0.99      0.98      0.98      5920

    accuracy                           0.93     12236
   macro avg       0.83      0.69      0.73     12236
weighted avg       0.93      0.93      0.93     12236
```

One of the difficulties that we have faced in this project is that we don't meet the place in Morocco where we have the different types of crops that the model had training for, so as testing place we chose "Rajshahi (Bangladesh). we follow the same steps as in the first project to predict and this is the result:

RGB Composite Image





V. Conclusion

This project was divided into two classification problems which are Land cover classification and crop identification. We have discussed satellite imagery compared to normal images, several steps that are taken for Data acquisition and preparation, and also the machine learning models used for accomplishing this work.

References

1. https://arxiv.org/ftp/arxiv/papers/2010/2010.06497.pdf
2. Dynamic World, Near real-time global 10 m land use land cover mapping (nature.com)
3. 1709.00029.pdf (arxiv.org)
4. Remote Sensing | Free Full-Text | Land Cover Maps Production with High Resolution Satellite Image Time Series and Convolutional Neural Networks: Adaptations and Limits for Operational Systems (mdpi.com)
5. 9df3e6a0bbf3a104bd92aac1c6afaf5b.Classification of Satellite Images.pdf (ijesc.org)
6. Hyperspectral and Multispectral Imaging | Edmund Optics
7. Principles of Remote Sensing - Centre for Remote Imaging, Sensing and Processing, CRISP (nus.edu.sg)
8. What is Remote Sensing? | Earthdata (nasa.gov)
9. Remote sensing - Wikipedia