Solar Detection

The indices I have explored for the solar detection are as follows:

NDMI

NDWI

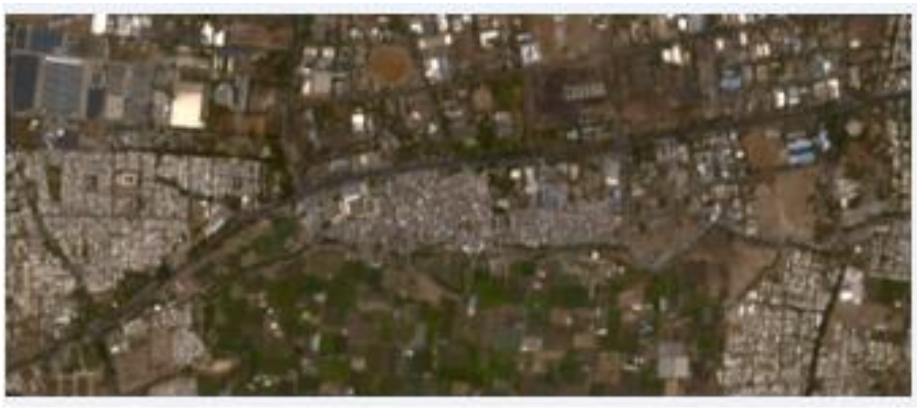
NDBI

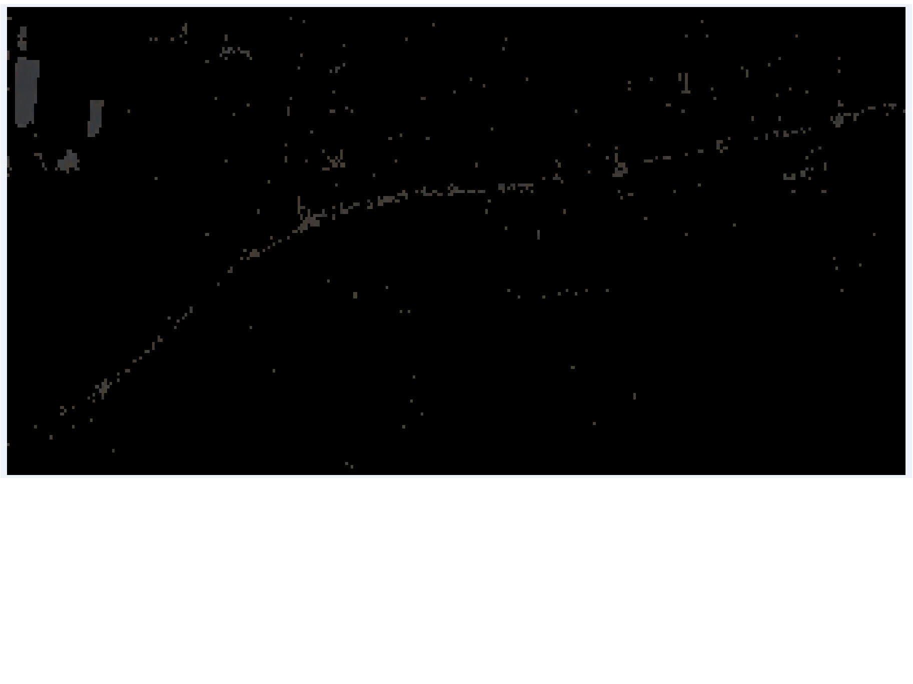
NDVI

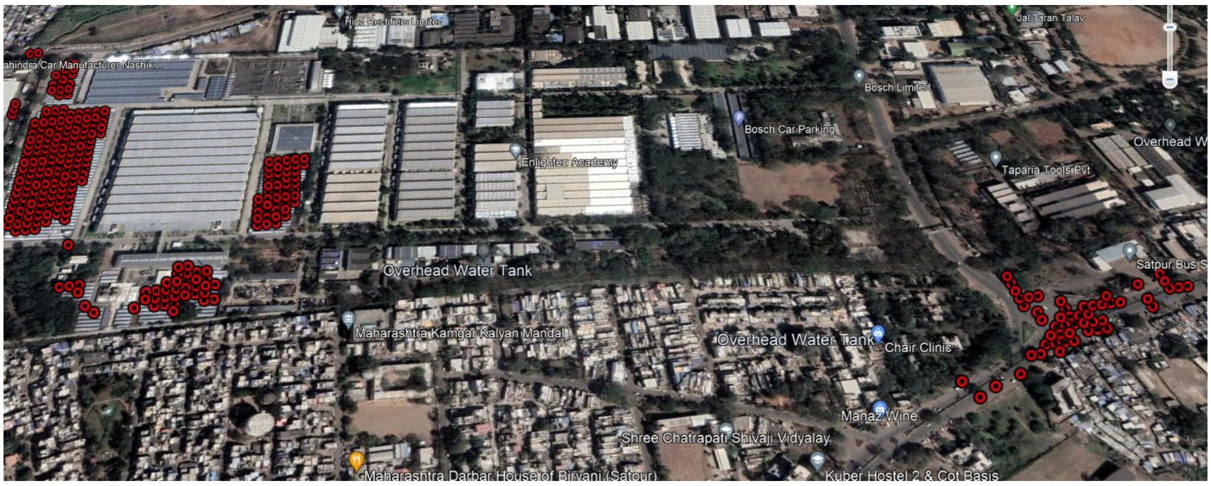
BAUC (Surface temp index)

This are the indices which don't show any change on the solar or buildup areas.

So the method I tried is Colour segmentation of NCI(Natural Colour Index






Mahindra Car Manufacturer Nashik
Jal Taran Talav
Bosch Limited
Bosch Car Parking
Enlighten Academy
Overhead W
Taparia Tools Pvt
Satpur Bus S
Overhead Water Tank
Maharashtra Kamgar Kalyan Mandal
Overhead Water Tank
Chair Clinic
Manaz Wine
Shree Chatrapati Shivaji Vidyalay
Maharashtra Darbar House of Biryani (Satpur)
Kuber Hostel 2 & Cot Basis

Thus running this code for the grids of whole nashik even water bodies, road, barren farms etc are getting detected which reduces the accuracy so much that its hard to detect only solars.

The alternate method I suggested was to train a AI mode for this but due to unviability of training data. First I have to make dataset and define the target vectors and after that train the model which is technically not possible in this resources so I have to drop tat plan too.

So I am submitting the code for colour segmentation and explain it below.

Code Explanation

This code performs the following tasks:

1. Image Segmentation:
    - Reads a TIFF image ('trimbak_road.tiff') using the imageio library.
    - Defines a range of colors representing light creme to dark brown in the RGB color space (lower_light_creme and upper_dark_brown).
    - Creates a mask that selects pixels within the specified color range.
    - Applies the mask to the original image to isolate pixels within the specified color range.
    - Saves the segmented pixels to a new TIFF file ('light_creme_to_dark_brown_segmented.tif').
2. Geospatial Information Extraction:
    - Uses the rasterio library to extract geospatial information from the original TIFF image.
    - Retrieves the transformation matrix (transform) and pixel coordinates (rows, cols) of the pixels in the mask.
    - Converts pixel coordinates to geographic coordinates (latitude and longitude) using the transformation matrix.
3. Save Latitude and Longitude Information:
    - Writes the latitude and longitude information to a CSV file ('lat_lon_coordinates.csv').
4. Clustering:

- Defines a function clustering that takes a CSV file containing latitude and longitude information as input.
- Loads the CSV data into a Pandas DataFrame.
- Extracts the 'Latitude' and 'Longitude' columns from the DataFrame.
- Defines a DBSCAN (Density-Based Spatial Clustering of Applications with Noise) model with specified parameters (epsilon and min_samples).
- Fits the DBSCAN model to the coordinates.
- Adds a 'cluster' column to the DataFrame, indicating the cluster label assigned by DBSCAN. Points classified as noise are labeled with -1.
- Filters out noise points and saves the cleaned data to a new CSV file ('cleaned_data.csv').

5. Execution:
    - Calls the clustering function with the 'lat_lon_coordinates.csv' file as input.

Note: Ensure that you have the required libraries (rasterio, pandas, scikit-learn, and imageio) installed before running the code. Additionally, the values of epsilon and min_samples in the DBSCAN model should be adjusted based on the characteristics of your geographic data.