Crop Monitoring using Multi-Spectral Satellite Data

UC RIVERSIDE

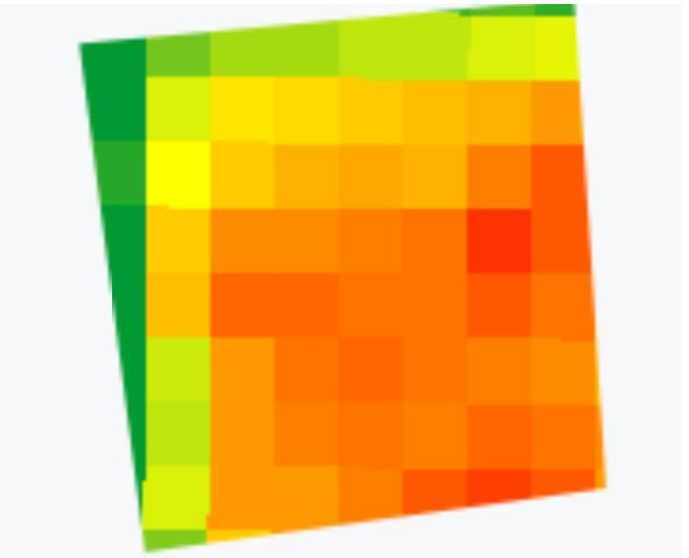
Adding NDVI band to the Collection:

```
def addNDVI100(image):
    ndvi = (image.expression("((b8 - b4) / (b8 + b4)) * (100)",{
    "b8": image.select("B8"),
    "b4": image.select("B4")
    })).toInt().rename('NDVI100')
    return image.addBands(ndvi)


collection = collection.map(addNDVI100)
```

```
Map = geemap.Map(center=[10.9925428,76.9159401], zoom=18)
Map.addLayer(first,{'min':10,'max':50,'palette': ['ff0000','ffff00','009933']}, 'ndvi')
Map.addLayerControl() # This line is not needed for ipyleaflet-based Map.
Map
```

Output Map Layer after adding NDVI:



UC RIVERSIDE

Algorithm for Calculating Stress-Zones :

Image 1

42	40
38	30
32	46

Image 2

40	40
40	32
30	44

Image 3

40	40
42	30
32	43

Image 4

40	40
44	34
34	40

Image 5

40	40
46	32
38	38

E1

0	1
0	0
0	0

E2

1	1
0	0
0	0

E3

1	1
0	0
0	0

E4

1	1
0	0
0	0

LRC1

2	0
0	0
2	2

LRC2

0	0
0	2
0	1

LRC3

0	0
0	0
0	3

LRC4

0	0
0	2
0	2

UC RIVERSIDE

Algorithm for Calculating Stress-Zones : (contd.)

F1 = LRC1+E1
x1

2	1
0	0
2	2

F2 = LRC2 + E2
x2

2	2
0	4
0	2

F3 = LRC3 + E3
x3

3	3
0	0
0	9

F4= LRC4 + E4
x4

4	4
0	8
0	8

Sum_F = F1+F2+F3+F4

11	10
0	12
2	21

Output = min-max normalization (Sum_F)

5.23	4.16
0	5.11
0.95	10

UC RIVERSIDE

Code for Calculating Stress-Zones:

```
def StressZoneImage(img1,img2,fmult):
    fmult = ee.Number(fmult)
    img1 = ee.Image(img1)
    img2 = ee.Image(img2)
    eImg = img1.eq(img2)
    rcImg = img1.subtract(img2)
    collectionFromImages = ee.ImageCollection.fromImages([img1, img2])
    maxImg = collectionFromImages.select('NDVI100').max()
    lImg = maxImg.neq(img2)
    lrc = rcImg.multiply(lImg)
    lrc = lrc.add(eImg)
    constImg = ee.Image(fmult).rename("NDVI100").clip(geometry)
    mlrc = lrc.multiply(constImg).toInt()
    return mlrc
```

UC RIVERSIDE

Code for Normalization:

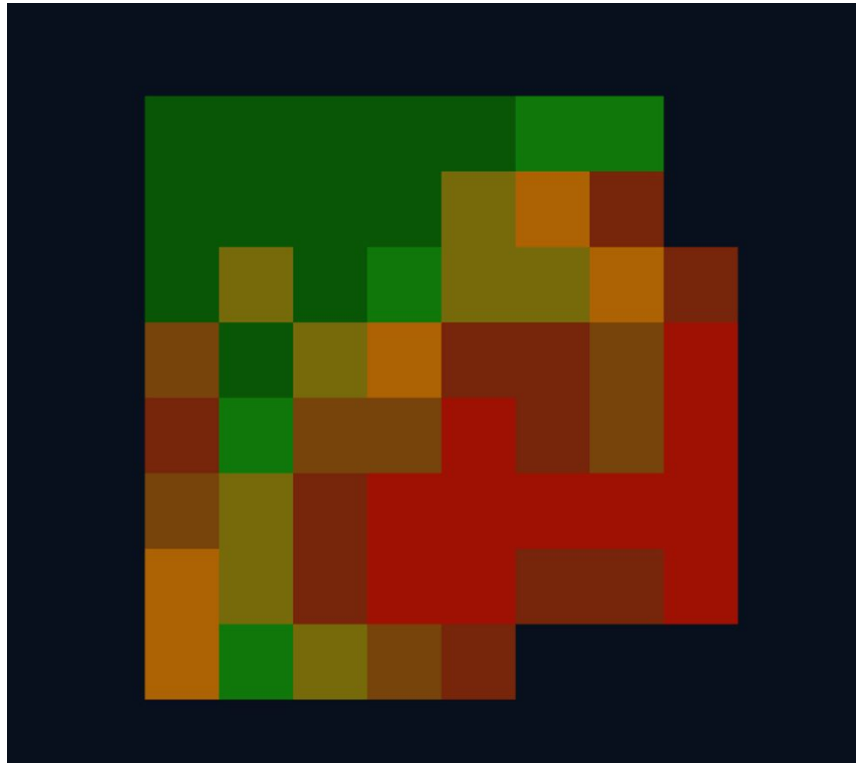
```
[ ]  minNdvi = ee.Number(finalImg.select("NDVI100").reduceRegion(**{
          'reducer': ee.Reducer.min(),
         'geometry':geometry,
          'scale': 10,
          'maxPixels': 1e9
          }).values().get(0));
     finalImg=finalImg.set({"minNdvi":minNdvi})

     maxNdvi = ee.Number(finalImg.select("NDVI100").reduceRegion(**{
          'reducer': ee.Reducer.max(),
         'geometry':geometry,
          'scale': 10,
          'maxPixels': 1e9
          }).values().get(0));
     finalImg=finalImg.set({"maxNdvi":maxNdvi})
```

Code for Normalization:

```
def addnorm(image):
    norm = (image.expression("((ndvi-min)/(max-min))*10",{
    "ndvi": image.select("NDVI100"),
    "min":minNdvi,
    "max":maxNdvi
    })).toInt().rename('norm')
    return image.addBands(norm)
finalImg = addnorm(finalImg)
```

Final Output:



UC RIVERSIDE

Existing vs Proposed Approach:

Existing Approaches:
Existing approaches gives intuition on crop's health only by calculating vegetation indices over a single image acquisition, this does not give us a solid insight on the crop's health throughout its vegetative cycle.

Proposed Approach:
The proposed approach considers all the previous image acquisitions within the vegetative cycle of the crop to compute the result for indicating the crop strength. This will provide better insights on precision agriculture.

Conclusion and Future Work:

Conclusion:
Thus, we have designed a model that works on multiple satellite image acquisitions to generate a better crop monitoring system for paddy crops using multi-spectral satellite images.

Future Work:
Before the final submission, we will work on a method to do factor multiplication by taking the missing images into account.

For future work, we will research more into compatible vegetation indices for paddy other than NDVI and compare their performances