

Google Earth Engine for Urban Environment

Ramadhan

November 17, 2023

Contents

1	Introducing spectral indices for built-up	1
2	Calculating built-up spectral indices in Google Earth Engine	4
3	Modelling built-up using relational operation	5
4	Modelling urban sprawl using multitemporal data	6
5	Introducing Land Surface Temperature (LST) and land cover	10
6	Compositing and preprocessing LST data	11
7	Modelling LST and urban density data	14
8	Predict future LST using machine learning	17

1 Introducing spectral indices for built-up

Spectral indices or spectral index (singular) is a term used to define a result of mathematical operation (map algebra, raster algebra, etc.) of two or more band of spectrum from a multispectral imagery [1].

The most famous one of spectral indices is NDVI (Normalized Difference Vegetation Index). It is the result from the margin of near infrared (NIR) and red band divided by the sum of both, following Equation 1. This index have many use such as to classify certain land cover: soil, built-up, water, sparse, and dense vegetation. It also can be used to monitor vegetation health and pattern over time. The result of NDVI is an image with a value ranging from -1 to 1 where value below 0 tend to be water, 0 to 0.3 is built-up, soil, and grass while above the 0.4 to be shrub and denser vegetation. Although high value can be used to identify if the vegetation is healthy. Example of the multispectral imagery and NDVI can be seen in Figure 1.

$$NDVI = \frac{NIR - RED}{NIR + RED} \quad (1)$$

NIR band is used in the formula is following the spectral reflectance curve on how the interaction between multiple electromagnetic wavelength to certain object. This relation could be understand from Figure 1. Based on the curve, it can be understood that the reflectance of vegetation at its peak in NIR band wavelength interval while it have a smaller reflectance in the red spectrum while red band also have a high reflectance in soil object.

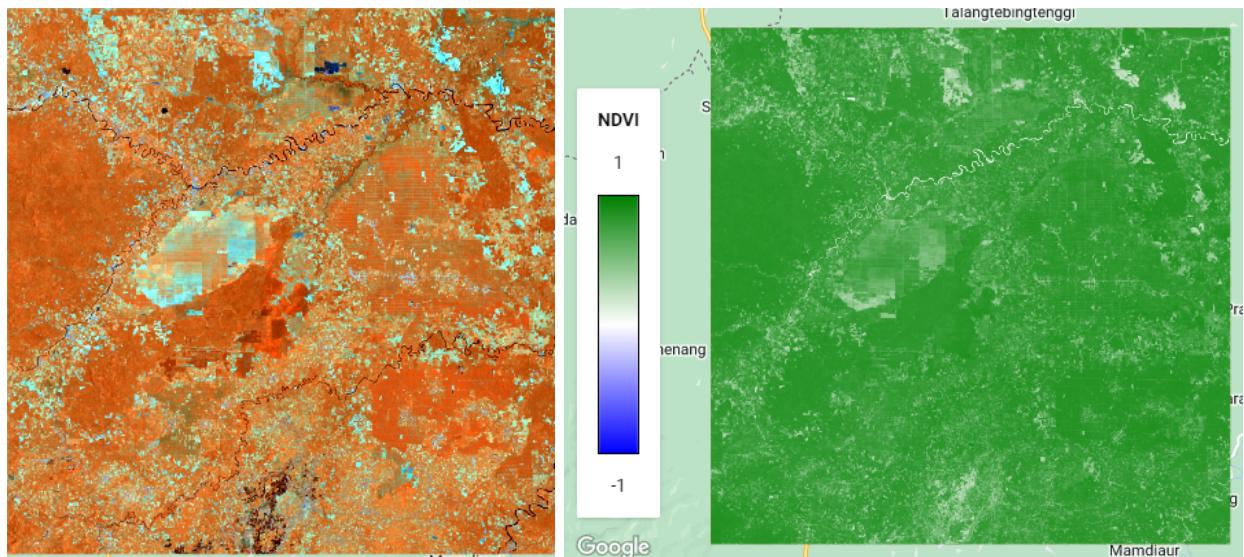


Figure 1: NIR-SWIR1-SWIR2 composite (left) and NDVI (right) image

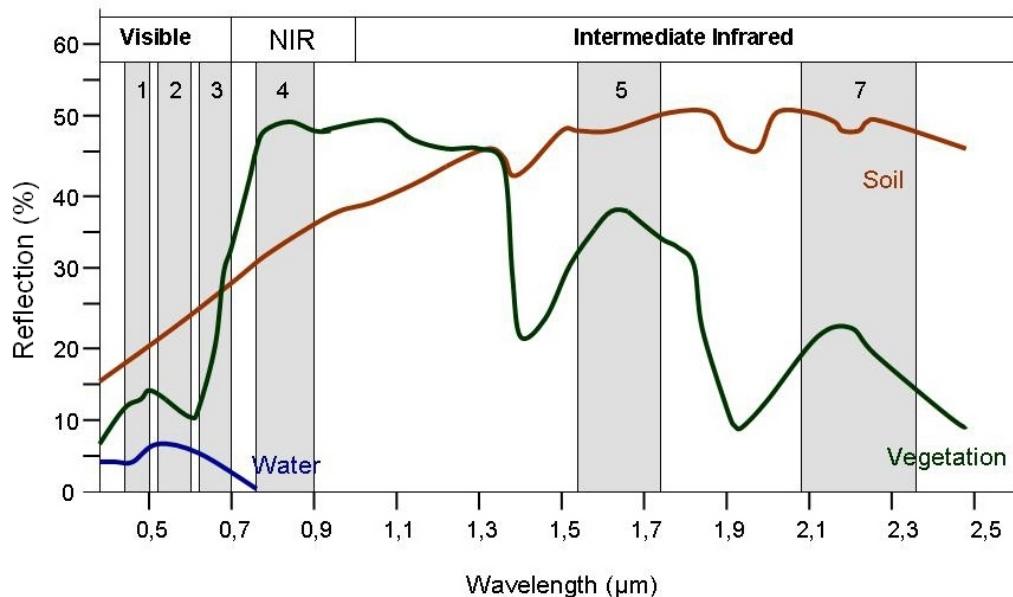


Figure 2: Spectral reflectance curve of soil, vegetation, and water on multiple electromagnetic spectrum [2]

While NDVI is quite famous, spectral indices is not only for vegetation. There are many spectral indices where it used for specific object and case. In indentifying built-up area, there is NDBI (Normalized Difference Built-up Index) [3]. NDBI is using two bands: SWIR and NIR where SWIR is good enough differentiate between built-up and bareland with vegetation. NDBI is calculated using Equation 2. While NDBI is intended for built-up, it can also be used on purpose and not on purpose on detecting bareland, sand, and soil.

$$NIR = \frac{SWIR1 - NIR}{SWIR1 + NIR} \quad (2)$$

In Landsat 8 and 9, NDBI cannot be used directly, sometime it needed additional mask. Water area usually needed to be mask first then NDBI will be applied. If that step is not covered, water could have higher NDBI value than built up like in Figure 1.

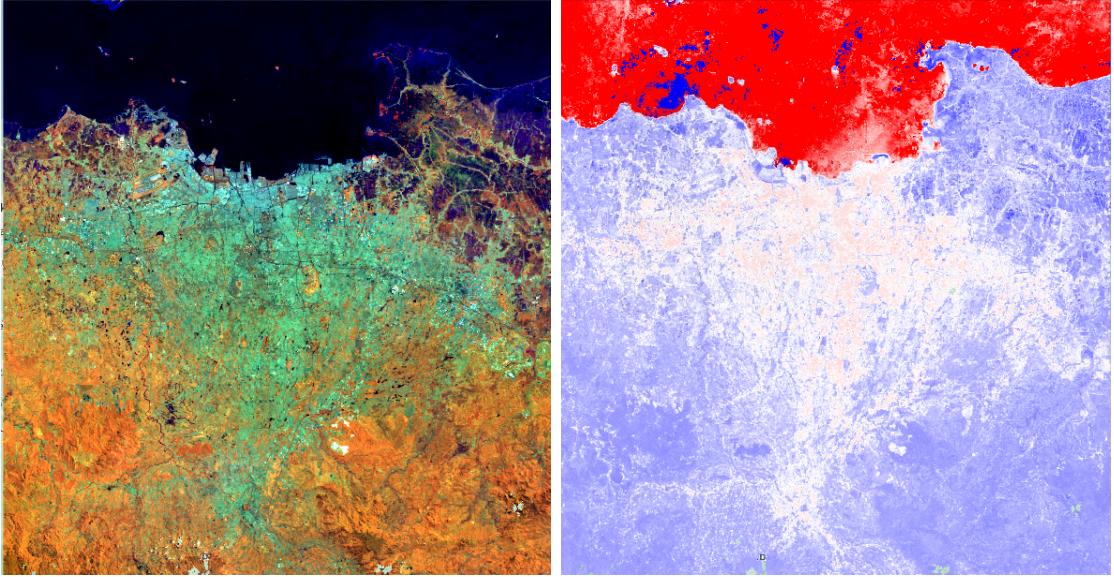


Figure 3: Comparison of Landsat NIR-SWIR1-BLUE composite (left) and NDBI (right)

Other than NDBI, there are many more spectral indices for built-up such as Index-based Built-up Index (IBI) and New Built-up Index (NBI). For more indices, it good to follow this paper [4]. The list of built-up indices mentioned in the paper can be read in Table 1.

Table 1: List of some built-up spectral indices

Indices	Formula
Index-based Built-up Index	$IBI = \frac{NDBI - (SAVI + MNDWI)/2}{NDBI + (SAVI + MNDWI)/2}$
New Built-up Index	$NBI = \frac{RED * SWIR}{NIR}$
Normalized Difference Imprevious Surface Index	$NDISI = \frac{TIR - [WI + NIR + SWIR]/3}{TIR + [WI + NIR + SWIR]/3}$
Band Ratio for Built-up Area	$BRBA = \frac{RED}{SWIR}$
Normalized Difference Built-up Area Index	$NDBAI = \frac{SWIR1 * SWIR2 / GREEN}{SWIR1 + SWIR2 / GREEN}$
Modified Built-up Index	$MBI = \frac{SWIR2 * RED - NIR^2}{RED + NIR + SWIR2}$
Built-up Extraction Index	$BAEI = \frac{RED + L}{GREEN + SWIR}$

2 Calculating built-up spectral indices in Google Earth Engine

Google Earth Engine (GEE) is cloud-based geospatial analysis software for global scale and multitemporal data [5]. It allows user to do analysis online without the need for high computation resources. It stores many remote sensing data that can be used directly. This allows professionals and academics to utilize for many projects and research.

Calculating spectral indices is one method to analyze the landscape using multispectral satellite imagery. This imagery is available in the GEE, so instead of downloading the image and calculating the indices in a GIS software, it is now possible to do it directly in the cloud.

In GEE, an imagery or stack of multispectral imagery is represented by `ee.Image` object/class. This object will have several properties such as all the bands of stacked image inside, geometry boundary, date, source, and any other properties. Using the bands available in the `ee.Image`, it is possible to calculate built-up indices. It could be done using mathematical operation such as `add`, `subtract`, `multiply`, & `divide` or using `ee.Image.expression` method where it receives two arguments: the formula in quoted text/string and band map or the definition of the formula's variables. Script 1 shows the example to calculate NDBI using Landsat imagery.

```
1 // Importing landsat imagery collection surface reflectance (level 2)
2 var landsat8 = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2");
3
4 // ee.Geometry object to filter and clip image
5 var roi = ee.Geometry({
6   "geodesic": false,
7   "type": "Polygon",
8   "coordinates": [
9     [
10       [
11         [
12           [
13             [
14               [
15                 [
16                   [
17                     [
18                       [
19                         [
20                           [
21                             [
22                               [
23                                 [
24                                   [
25                                     [
26                                       [
27                                         [
28                                           [
29                                             [
30                                               [
31                                                 [
32   });
33
34 // Cloud masking
35 function cloudMaskOli(image){
36   var qa = image.select('QA_PIXEL');
37   var dilated = 1 << 1;
38   var cirrus = 1 << 2;
```

```

39  var cloud = 1 << 3;
40  var shadow = 1 << 4;
41  var mask = qa.bitwiseAnd(dilated).eq(0)
42    .and(qa.bitwiseAnd(cirrus).eq(0))
43    .and(qa.bitwiseAnd(cloud).eq(0))
44    .and(qa.bitwiseAnd(shadow).eq(0));
45
46  return image.select(['SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7'
47    [, 'B2', 'B3', 'B4', 'B5', 'B6', 'B7']).updateMask(mask);
48}
49
50// Filter the imagery using boundary and date
51var image = landsat8.filterBounds(roi) // Filter by region
52  .filterDate('2023-01-01', '2023-12-31') // Filter by date
53  .map(cloudMaskOli) // Cloud mask
54  .median() // Get the first image from the collection
55  .clip(roi) // Clip the image
56  .multiply(0.0000275).add(-0.2);
57
58// Band map to define the band use for calculation
59var bandMap = {
60  NIR: image.select('B5'),
61  SWIR1: image.select('B6')
62};
63
64// Calculate NDBI
65var ndbi = image.expression('NDBI = (SWIR1 - NIR) / (SWIR1 + NIR)', bandMap)
66;
67// Show NDBI to map
68Map.addLayer(ndbi, { min: -1, max: 1, palette: ['blue', 'white', 'red'] }, 'NDBI');

```

Listing 1: GEE script to calculate NDBI from Landsat 8 OLI imagery

3 Modelling built-up using relational operation

NDBI is a raster value ranged -1 to 1 where the higher value mean it could have high probability to be built-up. However sometime using one indices is not enough to determine built-up, so it need another indices to mask that. Using relational operation in GEE, we could determine the condition to decide built-up area. In GEE, relational operation can be done using `lt`, `lte`, `gt`, `gte`, `eq`, `neq`, `and`, `or`, where each will state if our threshold is lower than (`lt`) or greater than (`gte`) of certain value. Script 2 show how to classify built-up area using relational operation which utilize some indices. It also using Landsat NIR band to mask cloud. The result can be seen in Figure 3.

```

1 // image is the landsat imagery
2 // Band map to define the band use for calculation
3 var bandMap = {
4   SWIR2: image.select('B7'),
5   SWIR1: image.select('B6'),
6   NIR: image.select('B5'),
7   GREEN: image.select('B3'),
8 };
9
10 // Calculate MNDWI
11 var mndwi = image.expression('MNDWI = (GREEN - SWIR1) / (GREEN + SWIR1)', bandMap);

```

```

12
13 // Calculate NBR2
14 var nbr2 = image.expression('NBR = (SWIR1 - SWIR2) / (SWIR1 + SWIR2)', bandMap);
15
16 // Calculate NDBI
17 var ndbi = image.expression('NDBI = (SWIR1 - NIR) / (SWIR1 + NIR)', bandMap);
18
19 // Built-up
20 var built = ndbi.gt(-0.1).and(mdnwi.lt(0)).and(nbr2.lte(0.2));
21
22 // Show land cover
23 Map.addLayer(built.selfMask(), { palette: 'lightcoral' }, 'Built-up');

```

Listing 2: GEE Script to Classify Built-up using Relational Operation

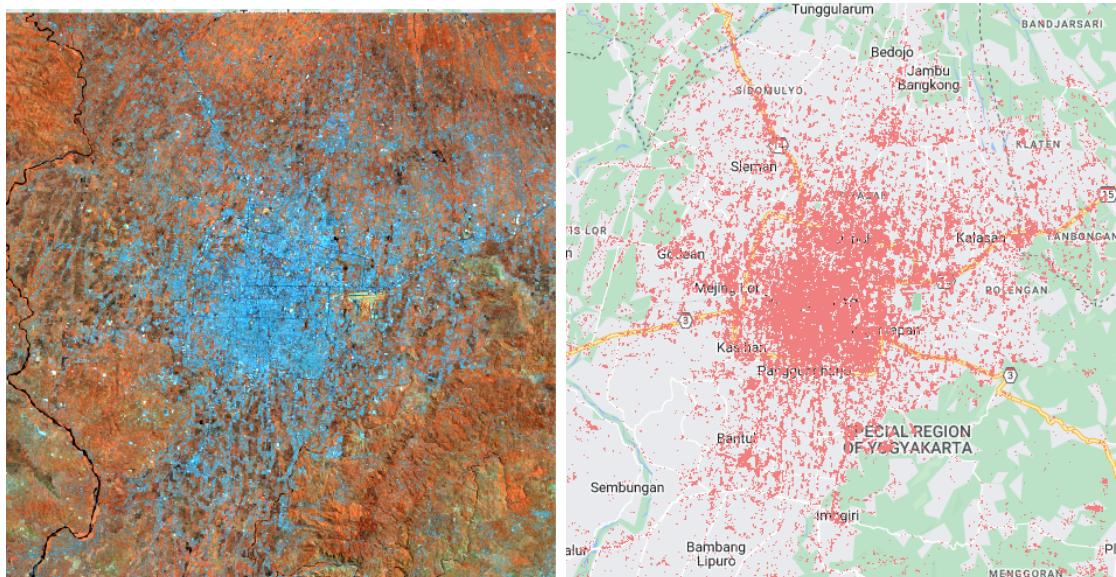


Figure 4: Image of Yogyakarta composite (NIR-SWIR1-SWIR2) (left) and classified built-up using spectral indices (right)

4 Modelling urban sprawl using multitemporal data

It is possible to model one step further using multitemporal data so that we could model how the built-up have expanded over the years. This usually need a yearly composite which will show cloud masked built-up area. Then we just stacked the multitemporal built-up area where each year have their own number assigned as urban expansion over the years. Script 3 will help you to do that. The result will shown like Figure 4.

```

1 // Importing landsat imagery collection surface reflectance (level 2)
2 var landsat8 = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2");
3 var landsat9 = ee.ImageCollection("LANDSAT/LC09/C02/T1_L2");
4 var landsat5 = ee.ImageCollection("LANDSAT/LT05/C02/T1_L2");
5 var landsat7 = ee.ImageCollection("LANDSAT/LE07/C02/T1_L2");
6 var landsat4 = ee.ImageCollection("LANDSAT/LT04/C02/T1_L2");
7
8 // ee.Geometry object to filter and clip image
9 var roi = ee.Geometry({

```

```

10  "geodesic": false,
11  "type": "Polygon",
12  "coordinates": [
13    [
14      [
15        110.2833489688428,
16        -7.892826204719211
17      ],
18      [
19        110.4701165469678,
20        -7.892826204719211
21      ],
22      [
23        110.4701165469678,
24        -7.697580101285619
25      ],
26      [
27        110.2833489688428,
28        -7.697580101285619
29      ],
30      [
31        110.2833489688428,
32        -7.892826204719211
33    ]
34  ]
35 ]
36 });
37
38 // Year list to map
39 var yearList = [1990, 1995, 2000, 2005, 2010, 2015, 2020];
40
41 // Function to filter
42 function filterCol(col, roi, date){
43   return col.filterDate(date[0], date[1]).filterBounds(roi);
44 }
45
46 // Composite function
47 function landsat457(roi, date){
48   var col = filterCol(landsat4, roi, date).merge(filterCol(landsat5, roi,
49     date)).merge(filterCol(landsat7, roi, date));
50   var image = col.map(cloudMaskTm).median().clip(roi);
51   return image;
52 }
53
54 function landsat89(roi, date){
55   var col = filterCol(landsat8, roi, date).merge(filterCol(landsat9, roi,
56     date));
57   var image = col.map(cloudMaskOli).median().clip(roi);
58   return image;
59 }
60
61 // Cloud mask
62 function cloudMaskTm(image){
63   var qa = image.select('QA_PIXEL');
64   var dilated = 1 << 1;
65   var cloud = 1 << 3;
66   var shadow = 1 << 4;
67   var mask = qa.bitwiseAnd(dilated).eq(0)
68     .and(qa.bitwiseAnd(cloud).eq(0))
69     .and(qa.bitwiseAnd(shadow).eq(0));

```

```

68
69     return image.select(['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B7'
70     ], ['B2', 'B3', 'B4', 'B5', 'B6', 'B7']).updateMask(mask);
71 }
72
73 function cloudMaskOli(image){
74     var qa = image.select('QA_PIXEL');
75     var dilated = 1 << 1;
76     var cirrus = 1 << 2;
77     var cloud = 1 << 3;
78     var shadow = 1 << 4;
79     var mask = qa.bitwiseAnd(dilated).eq(0)
80         .and(qa.bitwiseAnd(cirrus).eq(0))
81         .and(qa.bitwiseAnd(cloud).eq(0))
82         .and(qa.bitwiseAnd(shadow).eq(0));
83
84     return image.select(['SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7'
85     ], ['B2', 'B3', 'B4', 'B5', 'B6', 'B7']).updateMask(mask);
86 }
87
88 // Generate image per year
89 var builtCol = ee.ImageCollection(yearList.map(function(year){
90     var start = ee.Date.fromYMD(year, 1, 1);
91     var end = ee.Date.fromYMD(year, 12, 31);
92     var date = [start, end];
93
94     // Conditional on landsat collection to use
95     var landsat;
96     if (year < 2014) {
97         landsat = landsat457;
98     } else {
99         landsat = landsat89;
99     }
100
101     // Create an image composite
102     var image = landsat(roi, date).multiply(0.0000275).add(-0.2);
103
104     // Show the image
105     Map.addLayer(image, { min: [0.1, 0.05, 0.025], max: [0.4, 0.3, 0.2], bands
106         : ['B5', 'B6', 'B7'] }, 'Landsat_' + year, false);
107
108     // Band map
109     var bandMap = {
110         NIR: image.select('B5'),
111         SWIR: image.select('B6'),
112         RED: image.select('B4'),
113         GREEN: image.select('B3'),
114         BLUE: image.select('B2')
115     };
116
117     // Normalized Difference Built-up Index
118     var ndbi = image.expression('(SWIR - NIR) / (SWIR + NIR)', bandMap).rename
119         ('NDBI');
120
121     // Show the NDBI
122     Map.addLayer(ndbi, { min: -1, max: 1, palette: ['blue', 'white', 'red'] },
123         'NDBI_' + year, false);
124
125     // Modified Normalized Difference Water Index
126     var mndwi = image.expression('(GREEN - SWIR) / (GREEN + SWIR)', bandMap).

```

```

123 rename('MNDWI');
124
125 // Show the MNDWI
126 Map.addLayer(mndwi, { min: -1, max: 1, palette: ['red', 'white', 'blue'] }
127   , 'MNDWI_' + year, false);
128
129 // Built up
130 var built = ee.Image(0).where(ndbi.gt(-0.1).and(mndwi.lte(0)), year)
131   .selfMask()
132   .clip(roi);
133 Map.addLayer(built, { palette: 'lightcoral' }, 'Built-up_' + year, false);
134
135 // Image area for calculation
136 var area = built.multiply(ee.Image.pixelArea().multiply(0.0001)).rename(
137   'area');
138
139 return built.toInt16().rename('built').addBands(area).set('year', year,
140   'system:time_start', start);
141 });
142
143 // Create dictionary for each year expansion for visualization
144 var dict = {
145   'built_class_values': yearList,
146   'built_class_palette': ['800080', '0000FF', '00FFFF', '008000', 'FFFF00',
147     'FFA500', 'FF0000']
148 };
149
150 // Create expansion image
151 var urbanExpansion = builtCol.select('built').min().set(dict);
152 Map.addLayer(urbanExpansion, {}, 'Urban_expansion');
153
154 // Create legend for expansion year
155 var legend = ui.Panel([ui.Label('Urban expansion')], ui.Panel.Layout.flow(
156   'vertical'), { position: 'bottom-left' });
157 yearList.map(function(year, index){
158   legend.add(ui.Panel([
159     ui.Label('', { width: '20px', height: '20px', backgroundColor: dict.
160       built_class_palette[index], border: '0.5px solid black' }),
161     ui.Label(year)
162   ], ui.Panel.Layout.flow('horizontal')));
163 });
164 Map.add(legend);
165
166 // Create table to show the urban area change
167 var areaChart = ui.Chart.image.series(builtCol.select('area'), roi, ee.
168   Reducer.sum(), 30, 'year')
169   .setChartType('AreaChart')
170   .setOptions({
171     title: 'Urban area (Ha)',
172     hAxis: { title: 'Year' },
173     vAxis: { title: 'Area (Ha)' }
174   });
175 print(areaChart);

```

Listing 3: GEE Script to Model Urban Expansion

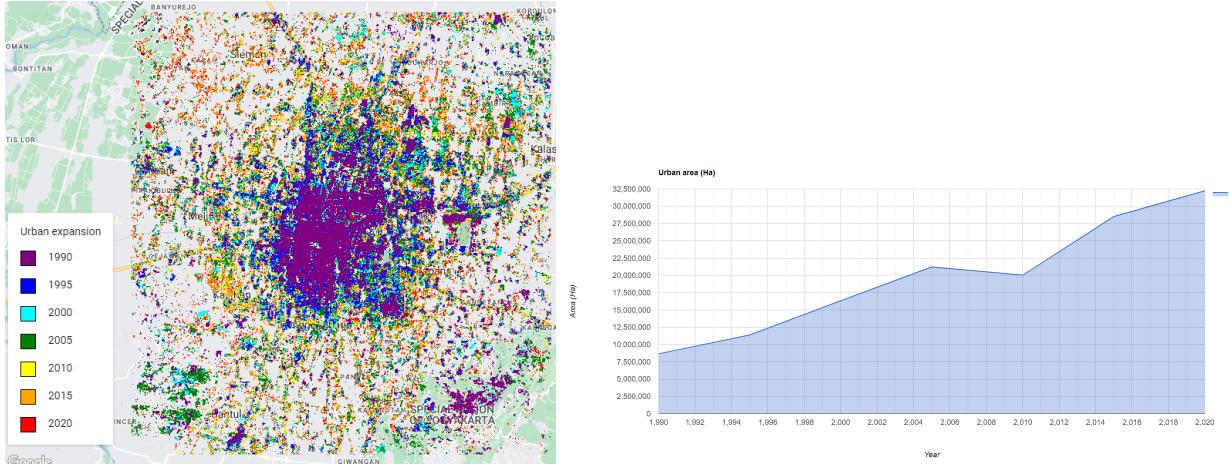


Figure 5: Urban expansion area by year (left) and the area changes (right)

5 Introducing Land Surface Temperature (LST) and land cover

Land surface temperature (LST) is the temperature generated when the energy from the sun hit the surface of the earth when touched. It is different than air temperature like in weather news. LST can be retrieved using thermal infrared wavelength in multispectral imagery like in Landsat and MODIS. LST have high correlation with the object on the surface of the Earth. Usually greener and wet object have lower LST than dry and barren land although another factor such as climate and atmospheric difference also affected it.

Usind NDVI, the LST usually high in range 0 - 0.3 and keep decreasing the further away from that range as seen in Figure 5. It is due the characteristic of vegetation and water that absorb thermal infrared more than bareland. Using this model, we can model what will happen to LST in the future.

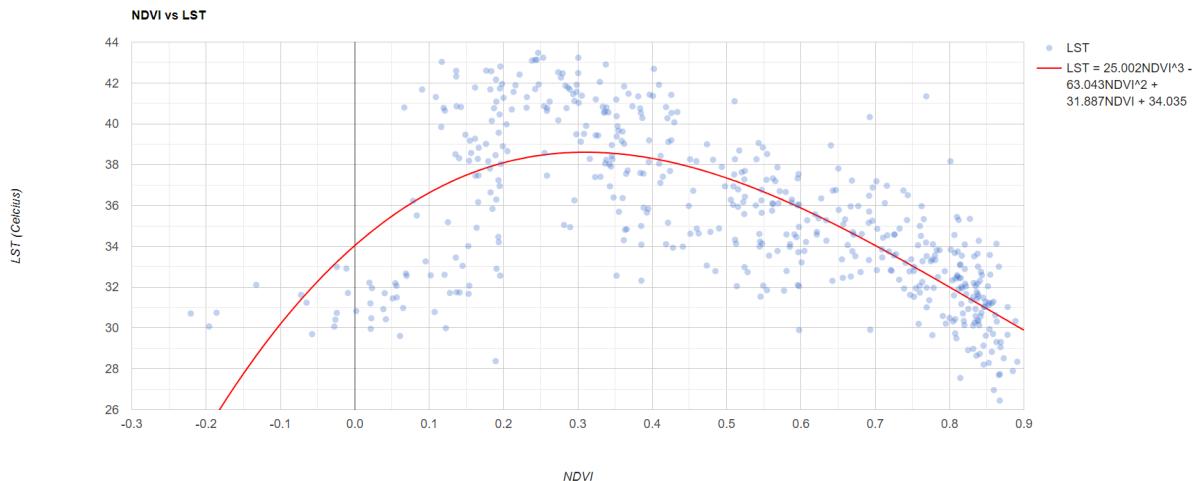


Figure 6: Relationship between NDVI and LST

6 Compositing and preprocessing LST data

There are two ways of generating LST data from Landsat: using the surface temperature ready band in level 2 collection or calculate yourself from the raw digital number data using known formula. To use ready to use data, you can select band `ST_B10` in the collection then multiply it by 0.00341802 and add 149 to turn it up into Kelvin like in Script 4. It will show LST like in Figure 6.

```
1 // Importing landsat imagery collection surface reflectance (level 2)
2 var landsat8 = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2");
3
4 // ee.Geometry object to filter and clip image
5 var roi = ee.Geometry({
6   "geodesic": false,
7   "type": "Polygon",
8   "coordinates": [
9     [
10       [
11         [
12           [
13             [
14               [
15                 [
16                   [
17                     [
18                       [
19                         [
20                           [
21                             [
22                               [
23                                 [
24                                   [
25                                     [
26                                       [
27                                         [
28                                           [
29                                             [
30                                               [
31                                                 [
32       });
33
34 // Cloud mask
35 function cloudMaskOli(image){
36   var qa = image.select('QA_PIXEL');
37   var dilated = 1 << 1;
38   var cirrus = 1 << 2;
39   var cloud = 1 << 3;
40   var shadow = 1 << 4;
41   var mask = qa.bitwiseAnd(dilated).eq(0)
42     .and(qa.bitwiseAnd(cirrus).eq(0))
43     .and(qa.bitwiseAnd(cloud).eq(0))
44     .and(qa.bitwiseAnd(shadow).eq(0));
45
46   return image.select(['SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7',
47     ['B2', 'B3', 'B4', 'B5', 'B6', 'B7'])
48     .multiply(0.0000275).add(-0.2)
49     .addBands(image.select(['ST_B10'], ['LST']).multiply(0.00341802).add
50     (149).add(-273.15))
51     .updateMask(mask);
```

```

51
52 // Image composite
53 var image = landsat8.filterBounds(roi).filterDate('2022-01-01', '2022-12-31')
    .map(cloudMaskOli).median().clip(roi);
54
55 // Show LST
56 Map.addLayer(image, { min: 30, max: 40, palette: ['black', 'purple', 'blue',
    'cyan', 'green', 'yellow', 'red'], bands: 'LST' }, 'LST');
57
58 // Calcualte NDVI
59 var bandMap = {
60     NIR: image.select('B5'),
61     RED: image.select('B4'),
62     SWIR1: image.select('B6'),
63     SWIR2: image.select('B7'),
64     GREEN: image.select('B3')
65 };
66
67 // NDVI group
68 var groupNdvi = ee.Image(0).where(ndvi.lte(0), 1)
69     .where(ndvi.gt(0).and(ndvi.lte(0.2)), 1)
70     .where(ndvi.gt(0.2).and(ndvi.lte(0.4)), 2)
71     .where(ndvi.gt(0.4).and(ndvi.lte(0.6)), 3)
72     .where(ndvi.gt(0.6).and(ndvi.lte(0.8)), 4)
73     .where(ndvi.gt(0.8), 5)
74     .selfMask()
75     .rename('group');

```

Listing 4: GEE Script get LST data

If you want to calculate LST yourself you could use Mono-window algorithm, Split-window algorithm, Planck, and many more. To do it you can follow Script 5. The result will be LST raster data like Figure 6.

```

1 // Calculate LST manually
2 // TOA to radiance
3 function toaRadiance(image){
4     var L10 = image.expression('(M * Q) + A', {
5         M: ee.Number(image.get('RADIANCE_MULT_BAND_10')),
6         Q: image.select('B10'),
7         A: ee.Number(image.get('RADIANCE_ADD_BAND_10'))
8     }).rename('B10_Rad');
9
10    var L11 = image.expression('(M * Q) + A', {
11        M: ee.Number(image.get('RADIANCE_MULT_BAND_11')),
12        Q: image.select('B11'),
13        A: ee.Number(image.get('RADIANCE_ADD_BAND_11'))
14    }).rename('B11_Rad');
15
16    return image.addBands([L10, L11]);
17}
18
19 // TOA to brightness temperature
20 function toaBrightnessTemp(image){
21     var T10 = image.expression('K2 / (log((K1 / L) + 1))', {
22         K1: ee.Number(image.get('K1_CONSTANT_BAND_10')),
23         K2: ee.Number(image.get('K2_CONSTANT_BAND_10')),
24         L: image.select('B10_Rad')
25     }).rename('B10_BT');
26
27     var T11 = image.expression('K2 / (log((K1 / L) + 1))', {

```

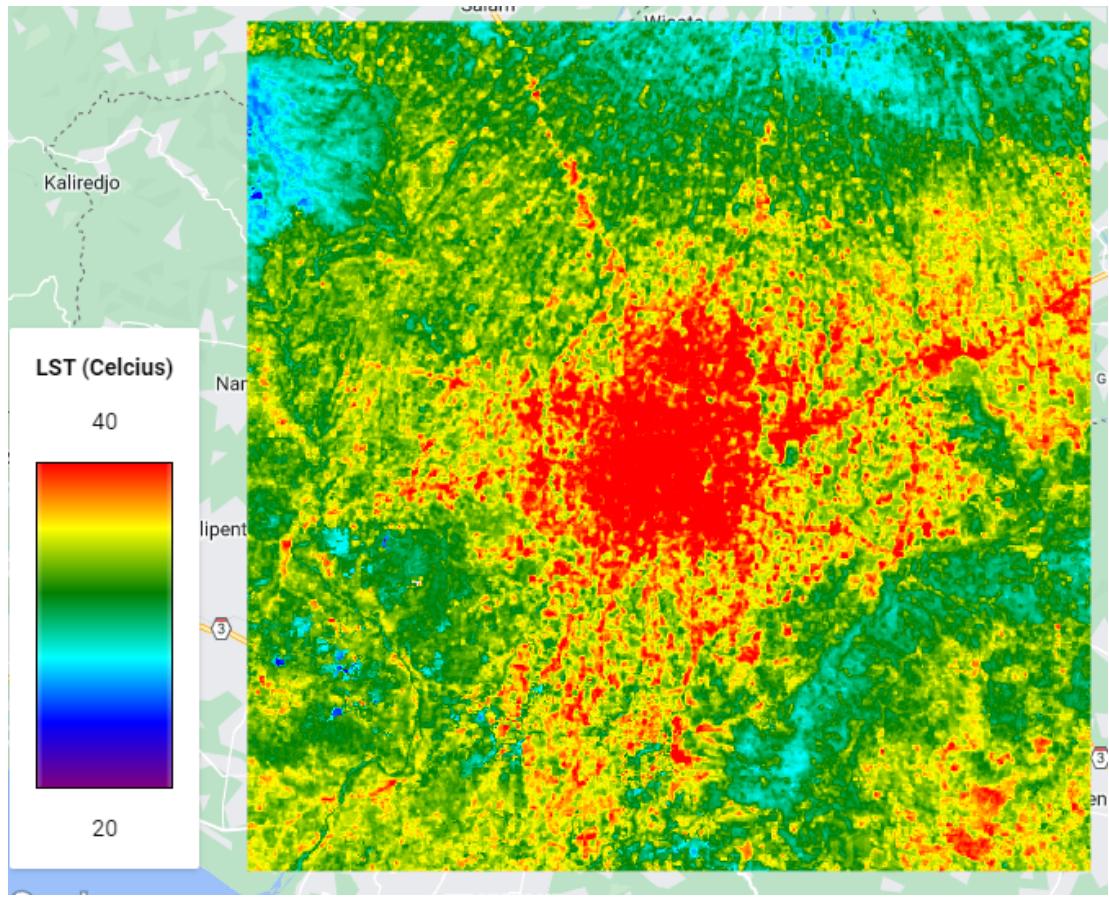


Figure 7: LST generated from Landsat 8 Level 2

```

28     K1: ee.Number(image.get('K1_CONSTANT_BAND_11')),
29     K2: ee.Number(image.get('K2_CONSTANT_BAND_11')),
30     L: image.select('B11_Rad')
31   }).rename('B11_BT');
32
33
34   return image.addBands([T10, T11]);
35 }
36
37 // Landsat 8 raw
38 var landsat8raw = ee.ImageCollection('LANDSAT/LC08/C02/T1').filterBounds(roi)
  .filterDate('2022-01-01', '2022-12-31');
39
40 // Landsat 8 LST
41 var landsat8Lst = landsat8raw.map(function(image){
42   // Cloud masking
43   var qa = image.select('QA_PIXEL');
44   var dilated = 1 << 1;
45   var cirrus = 1 << 2;
46   var cloud = 1 << 3;
47   var shadow = 1 << 4;
48   var mask = qa.bitwiseAnd(dilated).eq(0)
49     .and(qa.bitwiseAnd(cirrus).eq(0))
50     .and(qa.bitwiseAnd(cloud).eq(0))
51     .and(qa.bitwiseAnd(shadow).eq(0));
52
53   image = image.updateMask(mask);
54
55   // Toa radiance

```

```

56     image = toaRadiance(image);
57
58 // Toa brightness
59 image = toaBrightnessTemp(image);
60
61 // MWA
62 var mwa = image.expression('((ai * (1 - Ci - Di)) + (((bi * (1 - Ci - Di))
+ Ci + Di) * Ti) - (Di * Ta)) / Ci', {
63   ai: -67.355351,
64   bi: 0.458606,
65   Ci: 0.98 * 0.9,
66   Di: image.expression('(1 - ti) * (1 + ((1 + 0.98) * ti))', { ti: 0.95 })
67   ,
68   Ti: image.select('B10_BT'),
69   Ta: 288
70 }).add(-273.15).rename('MWA');
71
72 // SWA
73 var swa = image.expression('B10 + (2.946 * (B10 - B11)) - 0.038', {
74   B10: image.select('B10_BT'),
75   B11: image.select('B11_BT')
76 }).add(-273.15).rename('SWA');
77
78 // Planck
79 var planck = image.expression('BT / (1 + (((A * BT) / p) * log(EMIT)))',
80   {
81   BT: image.select('B10_BT'),
82   A: 10.895,
83   p: 1438 * 10000,
84   EMIT: 249.9
85 }).add(-273.15).rename('Planck');
86
87
88 // LST palette
89 var lstPalette = ['purple', 'blue', 'cyan', 'green', 'yellow', 'red'];
90 Map.addLayer(landsat8Lst.select('MWA'), { min: 20, max: 40, palette:
91   lstPalette }, 'LST_MWA');
92 Map.addLayer(landsat8Lst.select('SWA'), { min: 20, max: 40, palette:
93   lstPalette }, 'LST_SWA');
94 Map.addLayer(landsat8Lst.select('Planck'), { min: 20, max: 40, palette:
95   lstPalette }, 'LST_Planck');

```

Listing 5: GEE Script to calculate LST manually

7 Modelling LST and urban density data

We can see the relationship between LST and NDVI but it is also possible to model it using urban density. Urban density index is how dense a built-up area which can be calculated using built-up and its neighborhood average built-up area. Urban density can be generated using convolution or focal operation of built-up area. In GEE, we can utilize `reduceNeighborhood` or `focalMean`, `focalMode`, `focalMedian`, `focalMax`, and `focalMin` where it will use `ee.Kernel` as the window. To generate urban density index and model it with LST you can follow Script 6. Urban density will look like Figure 7.

```

1 // Sample for chart
2 var sample = image.addBands([ndvi, groupNdvi]).stratifiedSample({

```

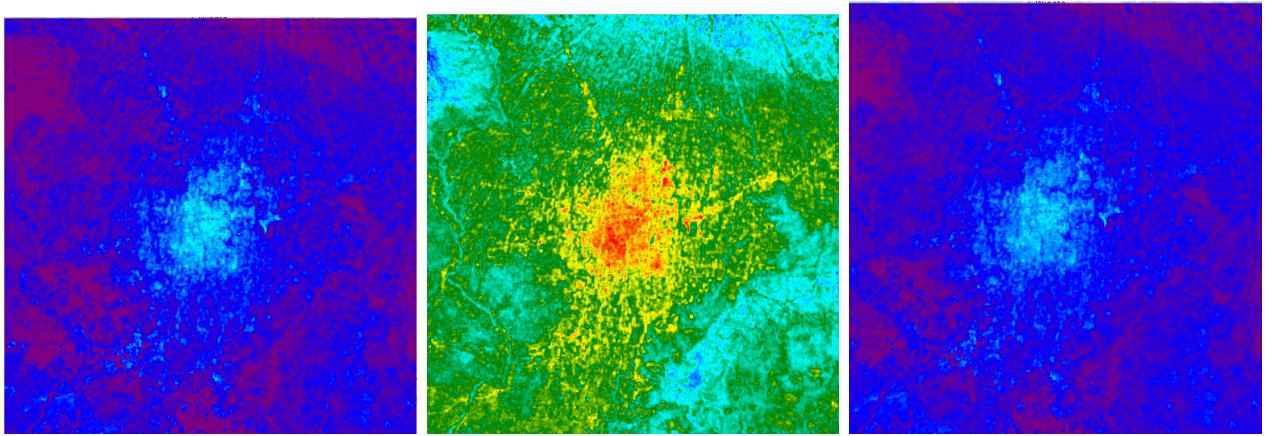


Figure 8: LST with multiple algorithms: MWA (left), SWA (center), and Planck (right)

```

3   numPoints: 100,
4   region: roi,
5   scale: 30,
6   classBand: 'group'
7 });
8
9 // Chart LST and NDVI
10 var chart = ui.Chart.feature.byFeature(sample, 'NDVI', ['LST'])
11   .setChartType('ScatterChart')
12   .setOptions({
13     title: 'NDVI vs LST',
14     dataOpacity: 0.3,
15     hAxis: {
16       title: 'NDVI',
17     },
18     vAxis: {
19       title: 'LST (Celcius)'
20     },
21     trendlines: {
22       0: {
23         type: 'polynomial',
24         opacity: 1,
25         color: 'red',
26         visibleInLegend: true,
27       }
28     }
29   );
30 print(chart);
31
32 // MDNWI
33 var mndwi = image.expression('MNDWI = (GREEN - SWIR2) / (GREEN + SWIR2)', 
34   bandMap);
35 Map.addLayer(mndwi, { min: -1, max: 1, palette: ['red', 'white', 'blue']}, ,
36   'MNDWI', false);
37
38 // NDBI
39 var ndbi = image.expression('NDBI = (SWIR2 - NIR) / (SWIR2 + NIR)', bandMap)
40 ;
41 Map.addLayer(ndbi, { min: -1, max: 1, palette: ['blue', 'white', 'red']}, ,
42   'NDBI', false);
43
44 // NBR2
45 var nbr2 = image.expression('NBR2 = (SWIR1 - SWIR2) / (SWIR1 + SWIR2)',
```

```

    bandMap);
42 Map.addLayer(nbr2, { min: -1, max: 1, palette: ['red', 'white', 'green']}, ,
  'NBR2', false);
43
44 // Built-up
45 var built = mndwi.lte(0).and(ndbi.gte(-0.5));
46 Map.addLayer(built.selfMask(), { palette: 'lightcoral' }, 'Built', false);
47
48 // Urban density
49 var urbanDensity = built.focalMean(3).reproject('EPSG:4326', null, 30).
  rename('urban_density');
50 Map.addLayer(urbanDensity, { min: 0, max: 1, palette: lstPalette }, 'Urban
  density');
51
52 // Sample
53 var sampleDensity = ee.Image([image.select('LST'), urbanDensity, groupNdvi])
  .stratifiedSample({
54   numPoints: 200,
55   classBand: 'group',
56   scale: 30,
57   region: roi
58 }).randomColumn();
59 Map.add(legendGradient('Urban density index', { min: 0, max: 1, palette:
  lstPalette}, 'bottom-right'));
60
61 // Chart
62 var chartDensity = ui.Chart.feature.byFeature(sampleDensity, 'urban_density'
  , ['LST'])
  .setChartType('ScatterChart')
  .setOptions({
63   title: 'Urban density vs LST',
64   dataOpacity: 0.3,
65   hAxis: {
66     title: 'Urban density'
67   },
68   vAxis: {
69     title: 'LST (Celcius)'
70   },
71   trendlines: {
72     0: {
73       type: 'linear',
74       opacity: 1,
75       color: 'red',
76       visibleInLegend: true,
77     }
78   }
79 })
80 );
81 });
82 print(chartDensity);
83
84 // Split sample to train and test
85 var train = sampleDensity.filter(ee.Filter.lte('random', 0.8));
86 var test = sampleDensity.filter(ee.Filter.gt('random', 0.8));
87
88 // Linear regression model
89 var model = train.reduceColumns(ee.Reducer.linearFit(), ['urban_density',
  'LST']);
90 var scale = ee.Number(model.get('scale'));
91 var offset = ee.Number(model.get('offset'));
92
93 // Test model

```

```

94 var cm = test.map(function(feat){
95   var lst = ee.Number(feat.get('urban_density'))
96     .multiply(scale)
97     .add(offset);
98   return feat.set('prediction', lst);
99 });
100
101 // Accuracy
102 var chartAccuracy = ui.Chart.feature.byFeature(cm, 'LST', ['prediction'])
103   .setChartType('ScatterChart')
104   .setOptions({
105     title: 'LST reference vs prediction (Celcius)',
106     dataOpacity: 0.3,
107     hAxis: {
108       title: 'Reference',
109     },
110     vAxis: {
111       title: 'Prediction',
112     },
113     trendlines: {
114       0: {
115         type: 'linear',
116         opacity: 1,
117         color: 'red',
118         visibleInLegend: true,
119         showR2: true
120       }
121     }
122   });
123 print(chartAccuracy);
124
125 // Apply model
126 var lstPrediction = urbanDensity.multiply(scale).add(offset);
127 Map.addLayer(lstPrediction, { min: 20, max: 40, palette: lstPalette }, 'LST
  prediction from urban density');

```

Listing 6: GEE Script to model urban density and LST

In the script, we try to do linear regression between LST and urban density index, the result shows that it have high correlation at Figure 7. If we apply the regression, we will have R^2 0.685 at Figure 7. Comparison between reference and prediction LST can be seen in 7.

8 Predict future LST using machine learning

Able to predict LST using built-up and bareland land cover mean it is also possible to predict LST in the future too. It also possible to add another variables such as elevation and other land cover density. In this part, I will show you how to predict LST in 2023 using data from 2021 and 2022. Then, we compare the result to 2023. You can follow the Script 7.

```

1 // Importing landsat imagery collection surface reflectance (level 2)
2 var landsat8 = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2");
3
4 // ee.Geometry object to filter and clip image
5 var roi = ee.Geometry({
6   "geodesic": false,
7   "type": "Polygon",
8   "coordinates": [
9     [
10       [

```

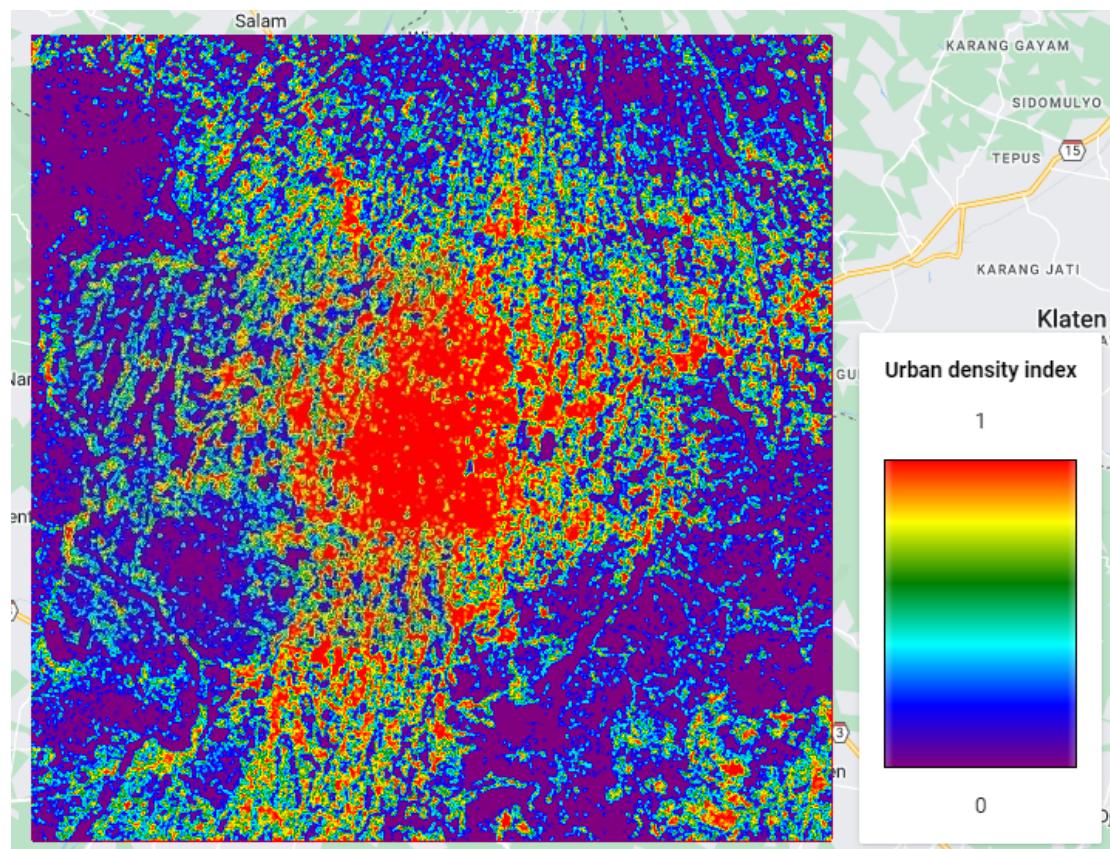


Figure 9: Urban density index

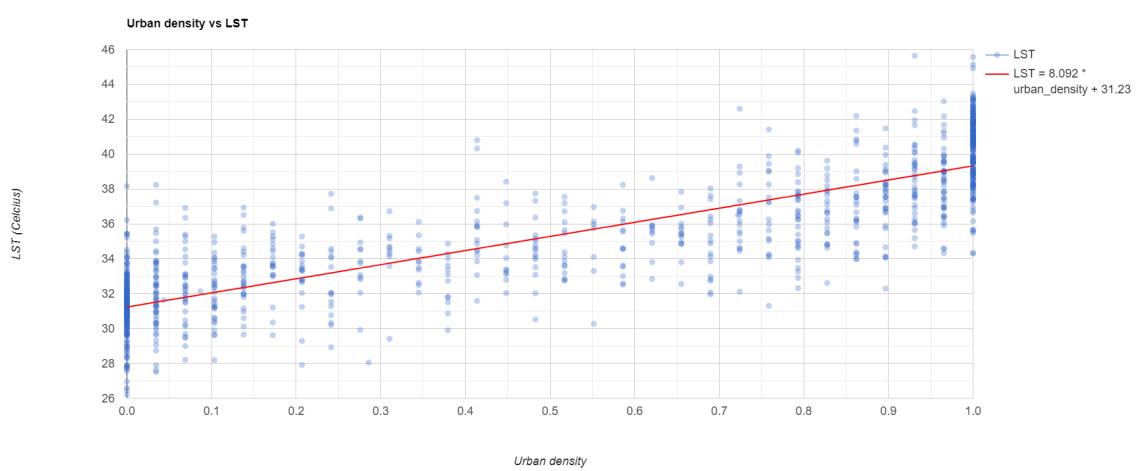


Figure 10: Correlation between LST and urban density index

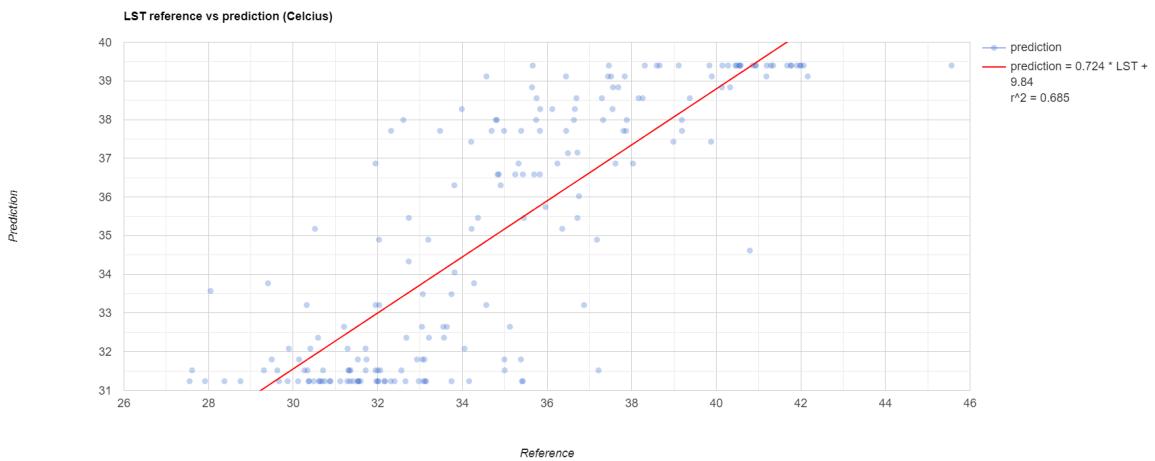


Figure 11: Regression accuracy using urban density index to predict LST

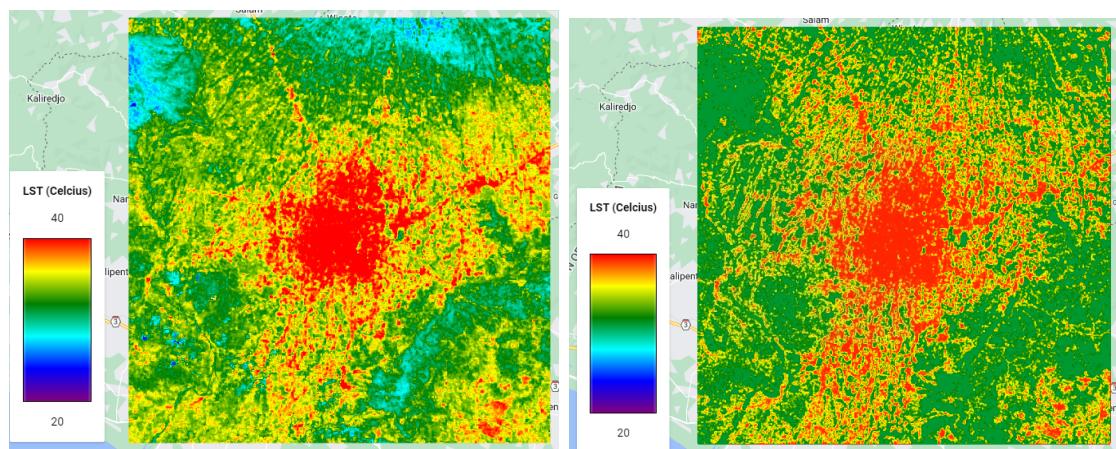


Figure 12: LST reference (left) vs prediction (right)

```

11         110.2833489688428 ,
12         -7.892826204719211
13     ] ,
14     [
15         110.4701165469678 ,
16         -7.892826204719211
17     ] ,
18     [
19         110.4701165469678 ,
20         -7.697580101285619
21     ] ,
22     [
23         110.2833489688428 ,
24         -7.697580101285619
25     ] ,
26     [
27         110.2833489688428 ,
28         -7.892826204719211
29     ]
30   ]
31 ]
32 });
33
34 // Predict future built-up bareland
35 // Years data for prediction
36 var years = [2021, 2022, 2023];
37 var yearToPredict = 2023;
38
39 // Elevation data
40 var srtm = ee.Image("USGS/SRTMGL1_003").clip(roi);
41
42 // Years image
43 var images = years.map(function(year){
44   var image = landsat8.filterBounds(roi).filterDate(year + '-01-01', year +
45     '-12-31').map(cloudMaskOli).median().clip(roi);
46
47   // Band map
48   var bandMap = {
49     NIR: image.select('B5'),
50     RED: image.select('B4'),
51     SWIR1: image.select('B6'),
52     SWIR2: image.select('B7'),
53     GREEN: image.select('B3')
54   };
55
56   // MNDWI
57   var mndwi = image.expression('MNDWI = (GREEN - SWIR2) / (GREEN + SWIR2)', {
58     bandMap: bandMap
59   });
60   //Map.addLayer(mndwi, { min: -1, max: 1, palette: ['red', 'white', 'blue'] }, 'MNDWI_' + year, false);
61
62   // NDBI
63   var ndbi = image.expression('NDBI = (SWIR2 - NIR) / (SWIR2 + NIR)', {
64     bandMap: bandMap
65   });
66   //Map.addLayer(ndbi, { min: -1, max: 1, palette: ['blue', 'white', 'red'] }, 'NDBI_' + year, false);
67
68   // NBR2
69   var nbr2 = image.expression('NBR2 = (SWIR1 - SWIR2) / (SWIR1 + SWIR2)', {
70     bandMap: bandMap
71   });

```

```

65  //Map.addLayer(nbr2, { min: -1, max: 1, palette: ['red', 'white', 'green']}, 'NBR2_' + year, false);
66
67 // NBR
68 var nbr = image.expression('NBR = (NIR - SWIR2) / (NIR + SWIR2)', bandMap)
;
69 //Map.addLayer(nbr, { min: -1, max: 1, palette: ['red', 'white', 'green']}, 'NBR_' + year, false);
70
71 // Built-up
72 var built = mndwi.lte(0).and(ndbi.gte(-0.5)).and(nbr.lte(0.2)).rename('built');
73 //Map.addLayer(built, { palette: 'lightcoral' }, 'Built_' + year, false);
74
75 // Water
76 var water = mndwi.gt(0.1).rename('water');
77 //Map.addLayer(water, { palette: 'lightskyblue' }, 'Water_' + year, false)
;
78
79 // Sparse vegetation
80 var sparseVegetation = nbr.gt(0.3).and(nbr.lt(0.6)).rename('sparse_vegetation');
81 //Map.addLayer(sparseVegetation, { palette: 'lightgreen' }, 'SparseVegetation_' + year, false);
82
83 // Sparse vegetation
84 var denseVegetation = nbr.gt(0.6).rename('dense_vegetation');
85 //Map.addLayer(denseVegetation, { palette: 'darkgreen' }, 'DenseVegetation_' + year, false);
86
87 // Bareland
88 var bareland = mndwi.lte(0).and(ndbi.gte(-0.5)).and(nbr2.gt(0.2)).and(nbr.lt(0.3)).rename('bareland');
89 //Map.addLayer(bareland, { palette: 'burlywood' }, 'Bareland_' + year, false);
90
91
92 return ee.Image([built, water, sparseVegetation, denseVegetation, bareland, image.select('LST')]);
93 });
94
95 // Images
96 var image2021 = images[0];
97 var image2022 = images[1];
98
99 // predictors
100 var predictors = ['built', 'bareland', 'water', 'sparse_vegetation', 'dense_vegetation', 'elevation', 'LST_first'];
101 var label = 'LST_last';
102
103 // Sample
104 var samplePredictionLst = ee.Image([
105   image2021.addBands(srtm).select(
106     ['built', 'bareland', 'water', 'sparse_vegetation', 'dense_vegetation', 'elevation', 'LST'],
107     predictors
108   ),
109   image2022.select(['LST'], ['LST_last']),
110   groupNdvi
111 ]).stratifiedSample({

```

```

112     numPoints: 200,
113     scale: 30,
114     classBand: 'group',
115     region: roi
116 }).randomColumn();
117
118 // Split into train and testing
119 var trainLst = samplePredictionLst.filter(ee.Filter.lte('random', 0.8));
120 var testLst = samplePredictionLst.filter(ee.Filter.gt('random', 0.8));
121
122 // Model
123 var modelLst = ee.Classifier.smileRandomForest(50).train(trainLst, label,
124   predictors)
125   .setOutputMode('REGRESSION');
126 print(modelLst.explain());
127
128 // Apply amodel to 2022 to predict 2023
129 var lst2023Prediction = image2022.addBands(srtm)
130   .select(
131     ['built', 'bareland', 'water', 'sparse_vegetation', 'dense_vegetation',
132      'elevation', 'LST'],
133     predictors
134   ).classify(modelLst, 'LST_prediction_2023');
135 Map.addLayer(lst2023Prediction, { min: 20, max: 40, palette: lstPalette }, ,
136   'LST prediction 2023', false);
137
138 // LST truth 2023
139 var lstTruth2023 = images[2].select(['LST'], ['LST_truth_2023']);
140 Map.addLayer(lstTruth2023, { min: 20, max: 40, palette: lstPalette }, 'LST
141   reference 2023', false);
142
143 // Accuracy assessment
144 var lstAssessment = ee.Image([lstTruth2023, lst2023Prediction, groupNdvi]).stratifiedSample({
145   region: roi,
146   scale: 30,
147   classBand: 'group',
148   numPoints: 200
149 });
150
151 // Chart
152 var lstChartPrediction = ui.Chart.feature.byFeature(lstAssessment, ,
153   'LST_truth_2023', ['LST_prediction_2023'])
154   .setChartType('ScatterChart')
155   .setOptions({
156     title: 'LST reference vs prediction (Celcius) in 2023',
157     dataOpacity: 0.3,
158     hAxis: {
159       title: 'Reference'
160     },
161     vAxis: {
162       title: 'Prediction'
163     },
164     trendlines: {
165       0: {
166         type: 'linear',
167         opacity: 1,
168         color: 'red',
169         visibleInLegend: true,
170         showR2: true

```

```

166     }
167   }
168 });
169 print(1stChartPrediction);

```

Listing 7: GEE Script to predict future LST

After we predict the future LST in 2023, we can compare it with truth in 2023. Figure 8 show 1 on 1 comparison while 8 show how accurate our prediction.

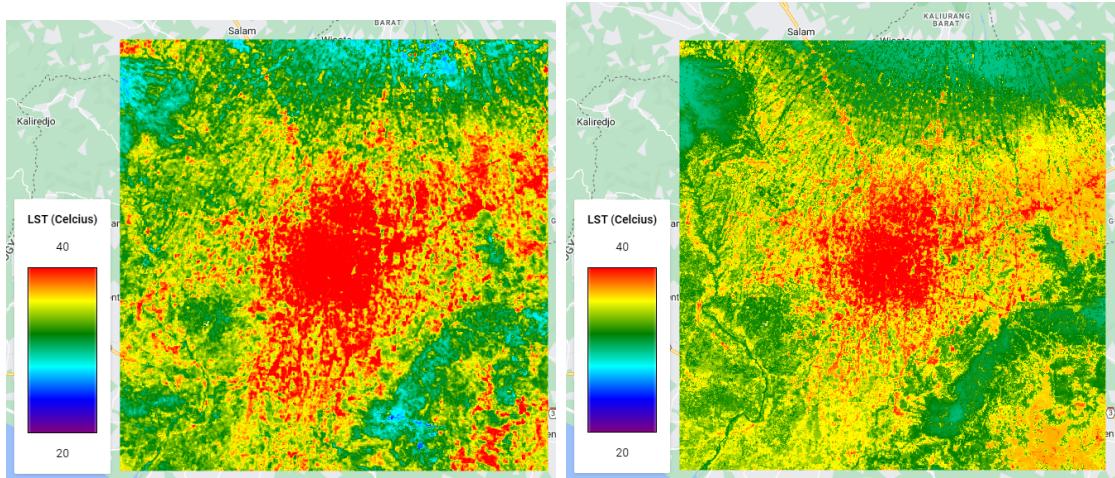


Figure 13: Reference (left) and predicted (right) LST in 2023

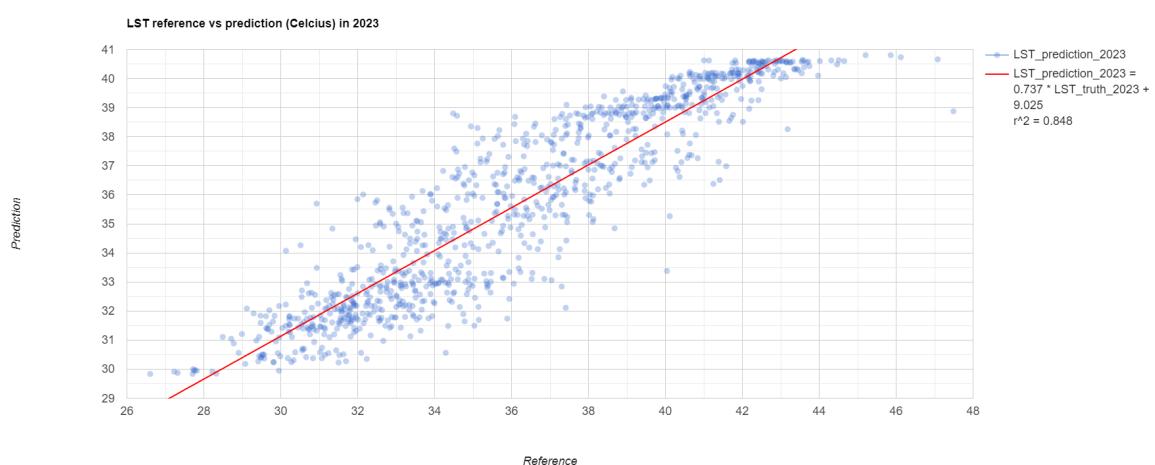


Figure 14: Accuracy assessment prediction for LST 2023

References

- [1] Jinru Xue, Baofeng Su, et al. "Significant remote sensing vegetation indices: A review of developments and applications". In: *Journal of sensors* 2017 (2017).
- [2] Alexander Siegmund and G Menz. "Fernes nah gebracht–Satelliten-und Luftbildeinsatz zur Analyse von Umweltveränderungen im Geographieunterricht". In: *Geographie und Schule* 154.4 (2005), pp. 2–10.

- [3] Youshui Zhang, Inakwu OA Odeh, and Chunfeng Han. “Bi-temporal characterization of land surface temperature in relation to impervious surface area, NDVI and NDBI, using a sub-pixel image analysis”. In: *International Journal of Applied Earth Observation and Geoinformation* 11.4 (2009), pp. 256–264.
- [4] Juan C Valdiviezo-N et al. “Built-up index methods and their applications for urban extraction from Sentinel 2A satellite data: discussion”. In: *JOSA A* 35.1 (2018), pp. 35–44.
- [5] Noel Gorelick et al. “Google Earth Engine: Planetary-scale geospatial analysis for everyone”. In: *Remote sensing of Environment* 202 (2017), pp. 18–27.