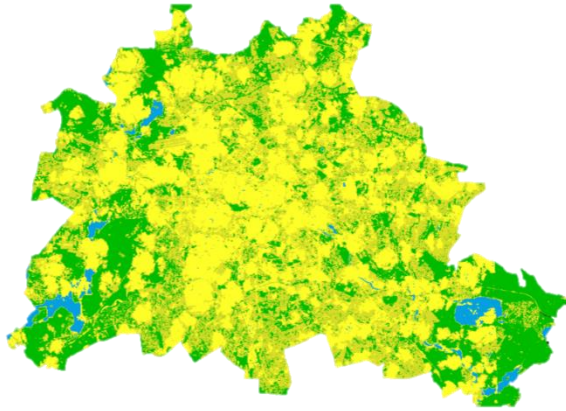


RASTER ANALYTICS IN ARCGIS NOTEBOOKS

Internship Project
Student : Frida Ruiz Mendoza
Supervisor: Thomas Paschke

CONTENT

- About the project
  - ArcGIS Notebooks
- Methods
- Showcase: Simple Change detection in Bremen
- *Tips and tricks* (Story Map)
- Recommendations and Future work

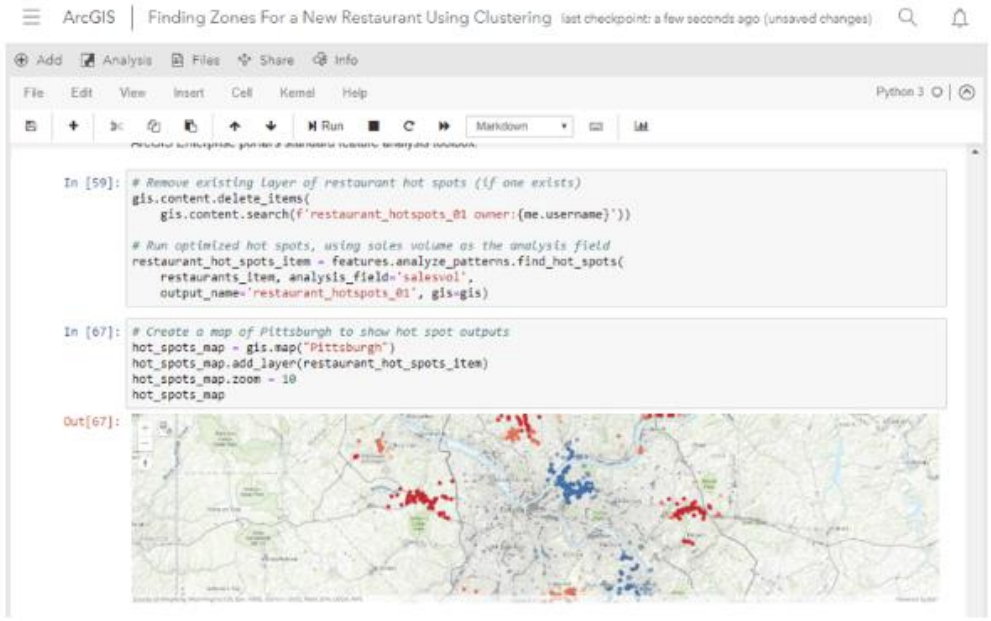
Internship Project

The **main objective** of this project was to:

"Create a showcase for **ArcGIS Notebooks** and explore the advantages it offers to create **end-to-end workflows** for **raster analysis** "

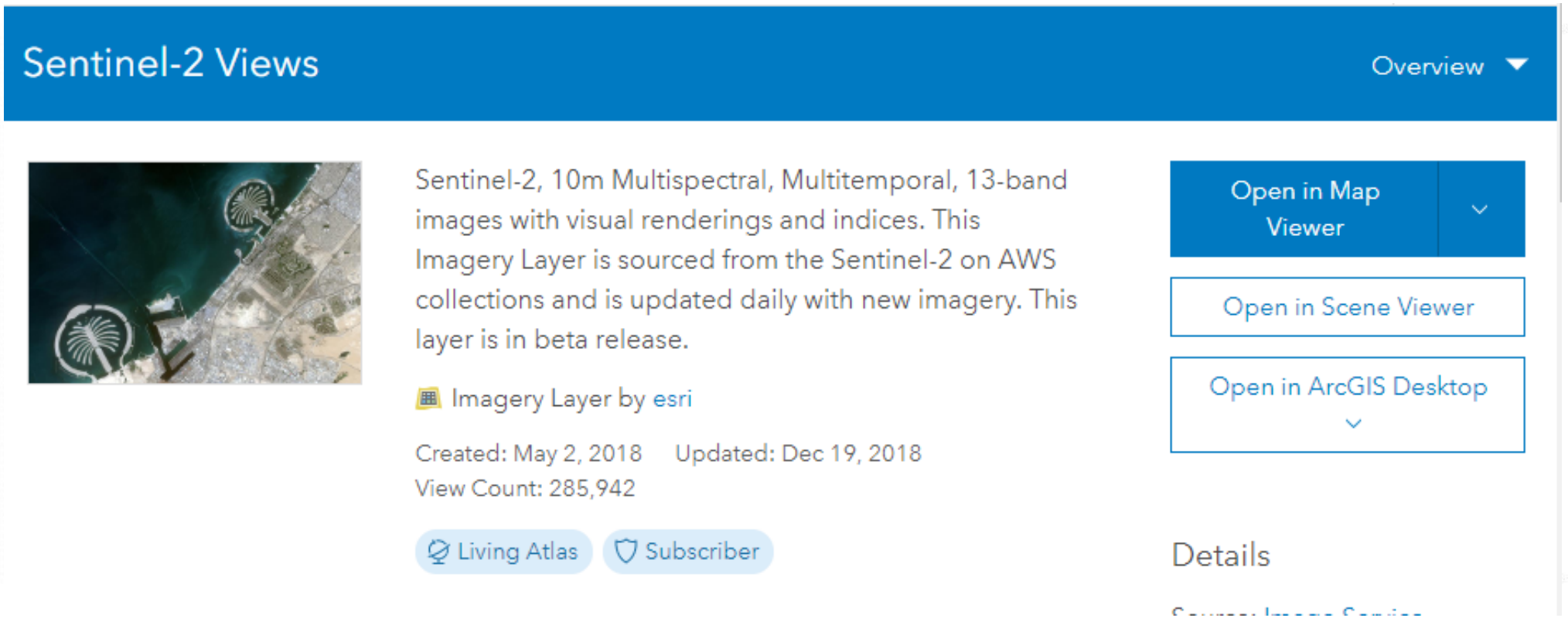
ArcGIS Notebooks

Hosted Jupyter Notebook in your ArcGIS Enterprise portal and powered by the new ArcGIS Notebook Server.

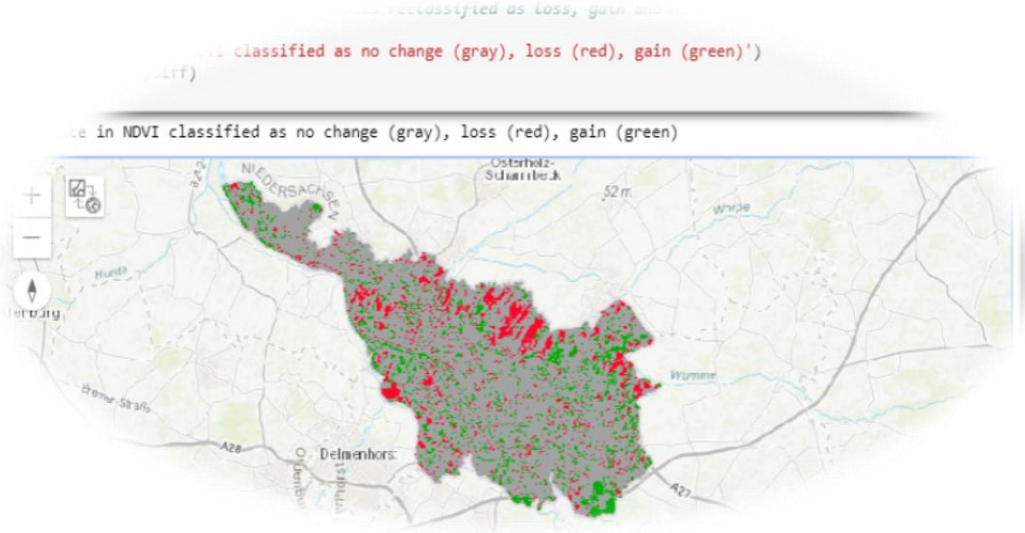


esri Deutschland
THE SCIENCE OF WHERE

Earth Observation programme by ESA that aims to provide global, continuous, autonomous, high quality.

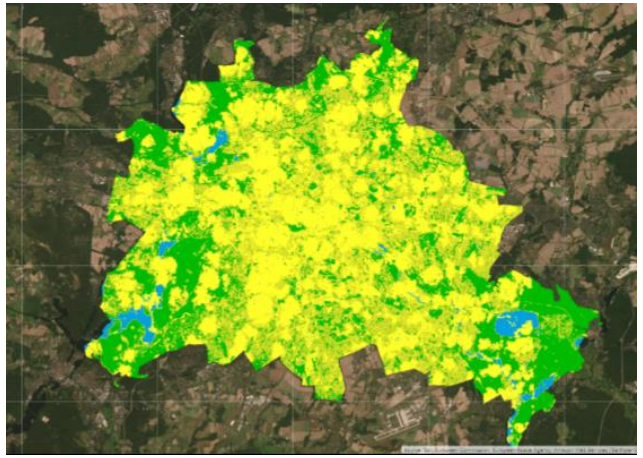


Simple change detection for vegetation monitoring

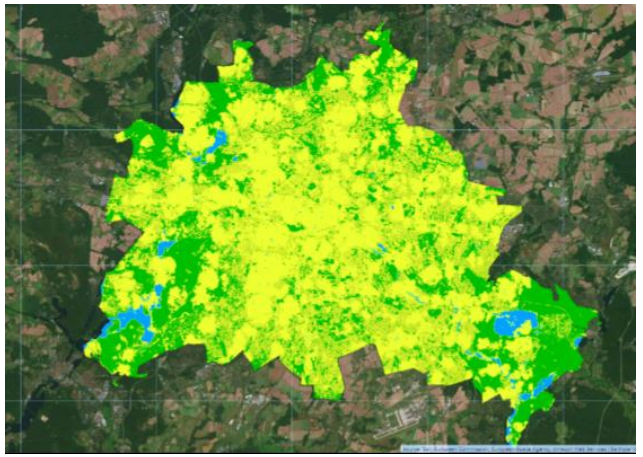


METHOD

First Date

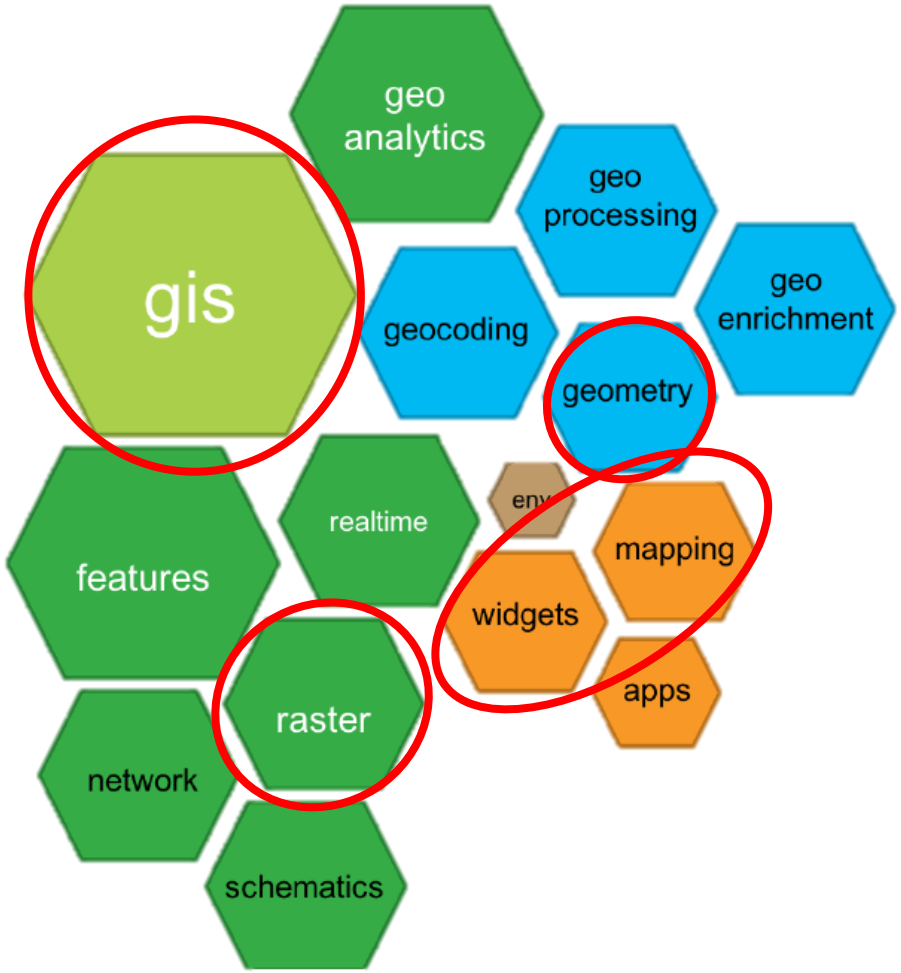


Second Date

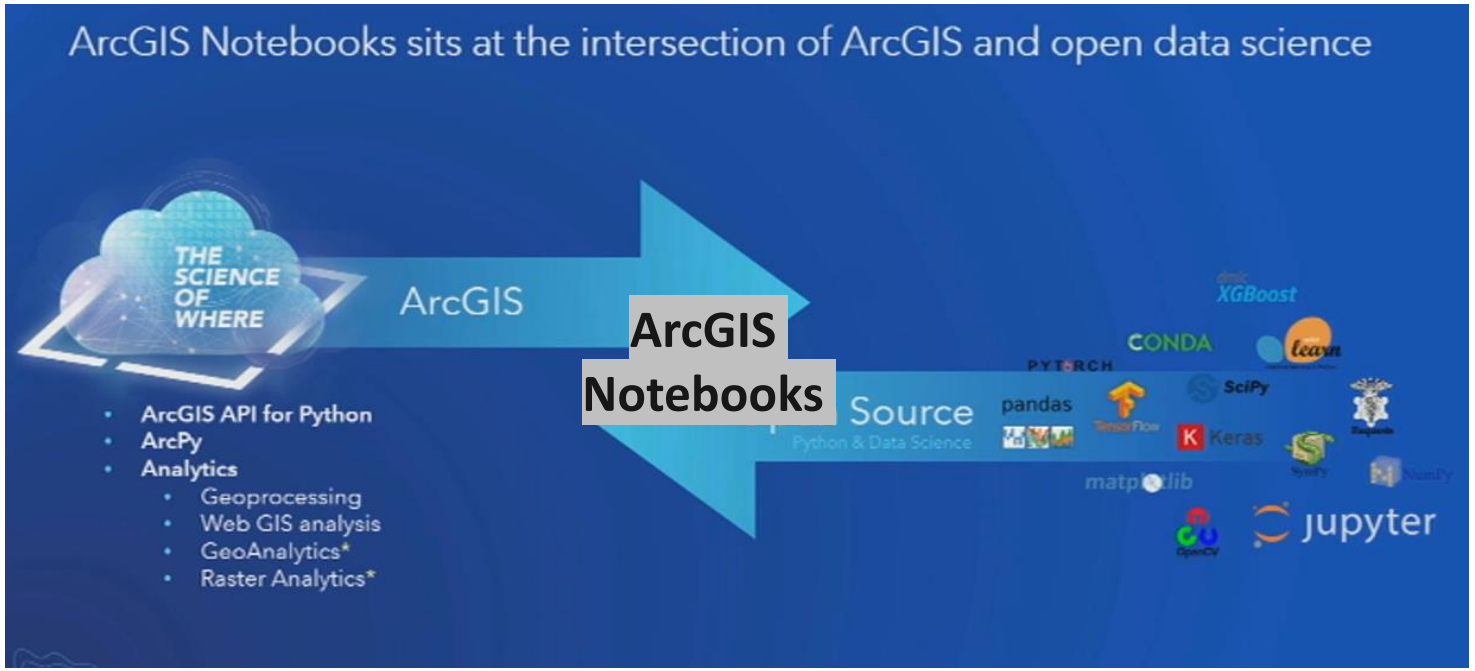


Difference map

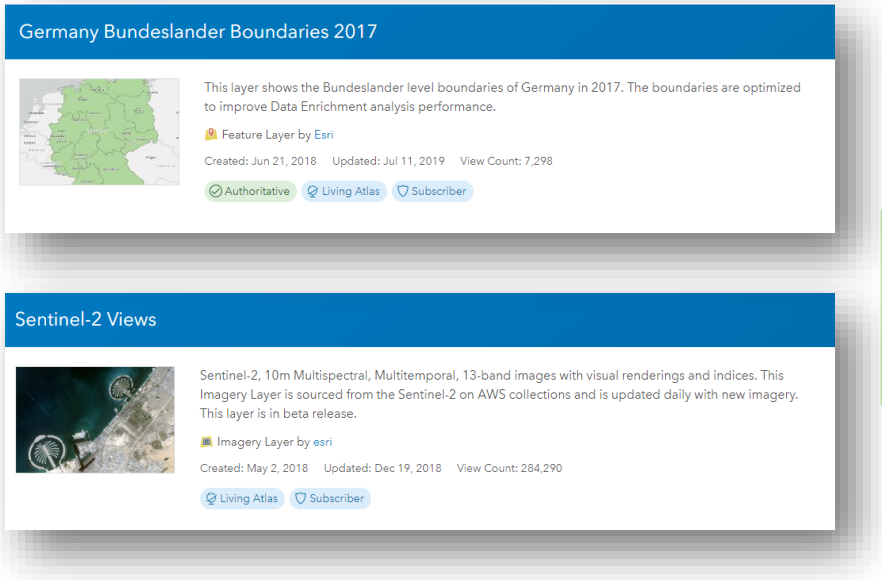
ARCGIS API FOR PYTHON



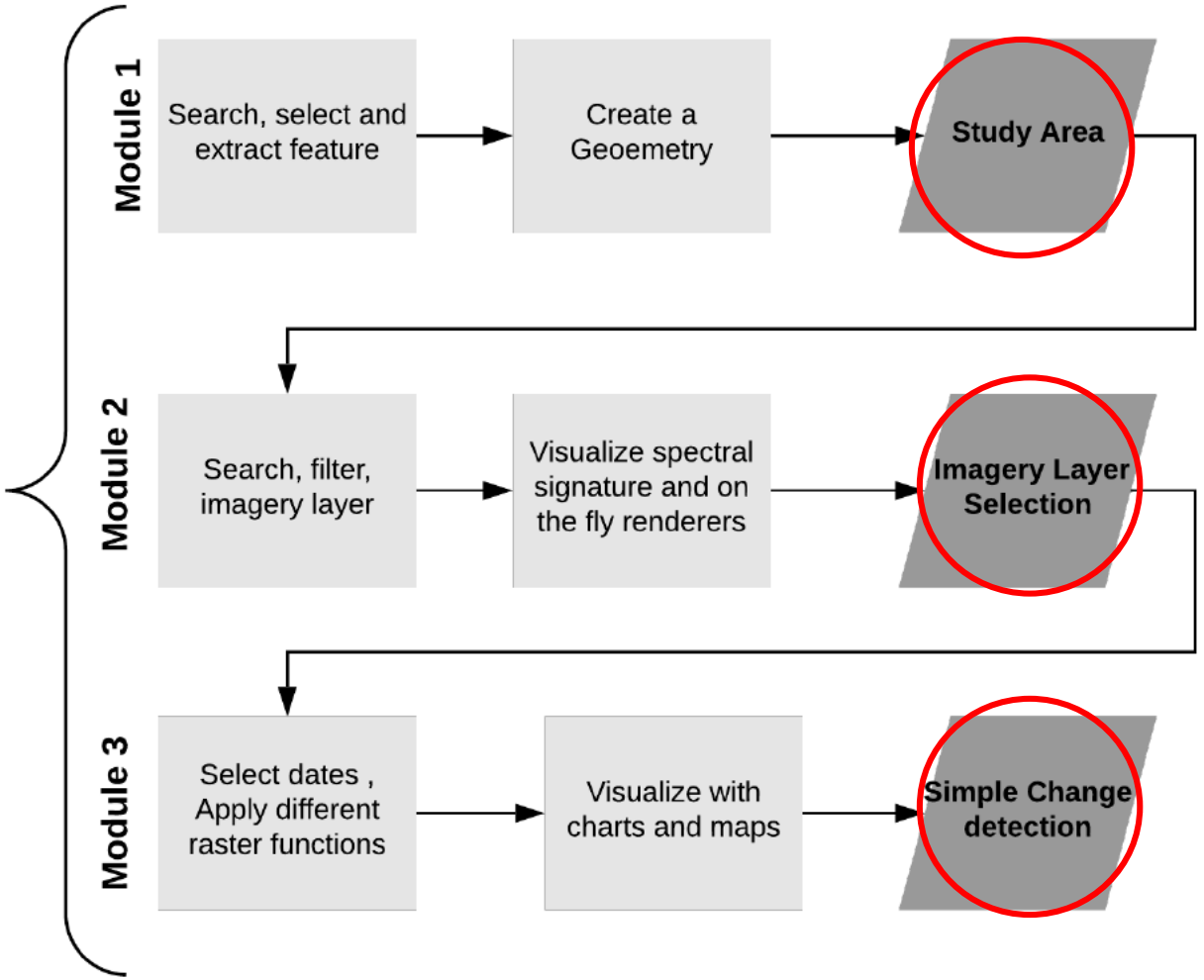
Bokeh



WORK-FLOW



Access your GIS



esri Deutschland
THE SCIENCE OF WHERE

Module 1.

4. Select single Features
and create a geometry using
Geometry module

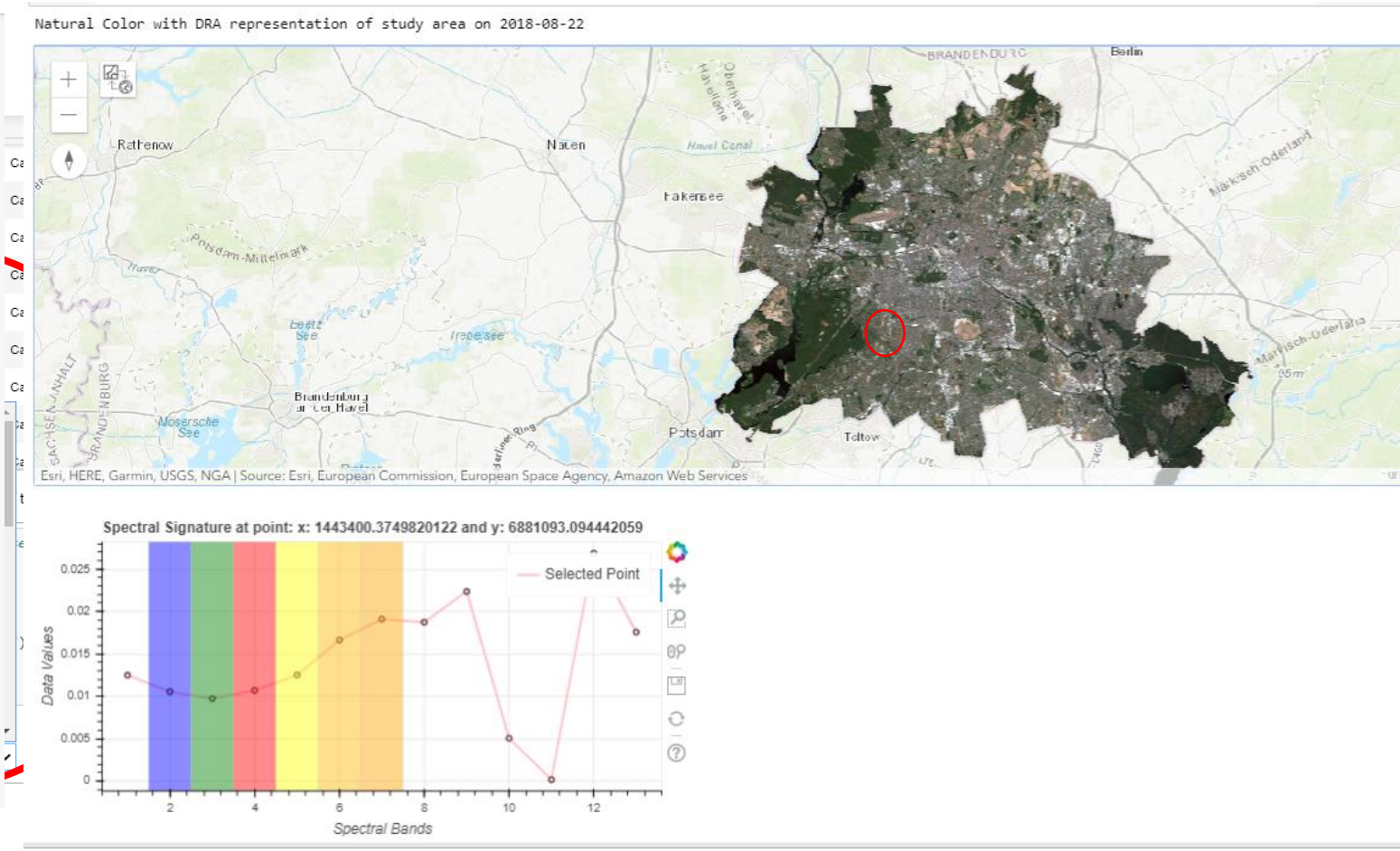
In [5]:
```
# NOTEBOOK MODULE 1
attributechoice = input("From which of the above attributes do you want to query from? ")
choice = input("enter your attribute parameter: ")
choice = attributechoice + " = "+"'"+choice+"'" # probably easier to choose from name attribute (need to improve interaction)
select_study_area = poli_lyr.query(choice, return_geometry = True,out_sr=3857)
from arcgis import geometry
study_area_geometry= geometry.Polygon(select_study_area.features[0].geometry)
study_area_geometry
```

```
From which of the above attributes do you want to query from? name
enter your attribute parameter: bremen
```

Out[5]:

Module 2.

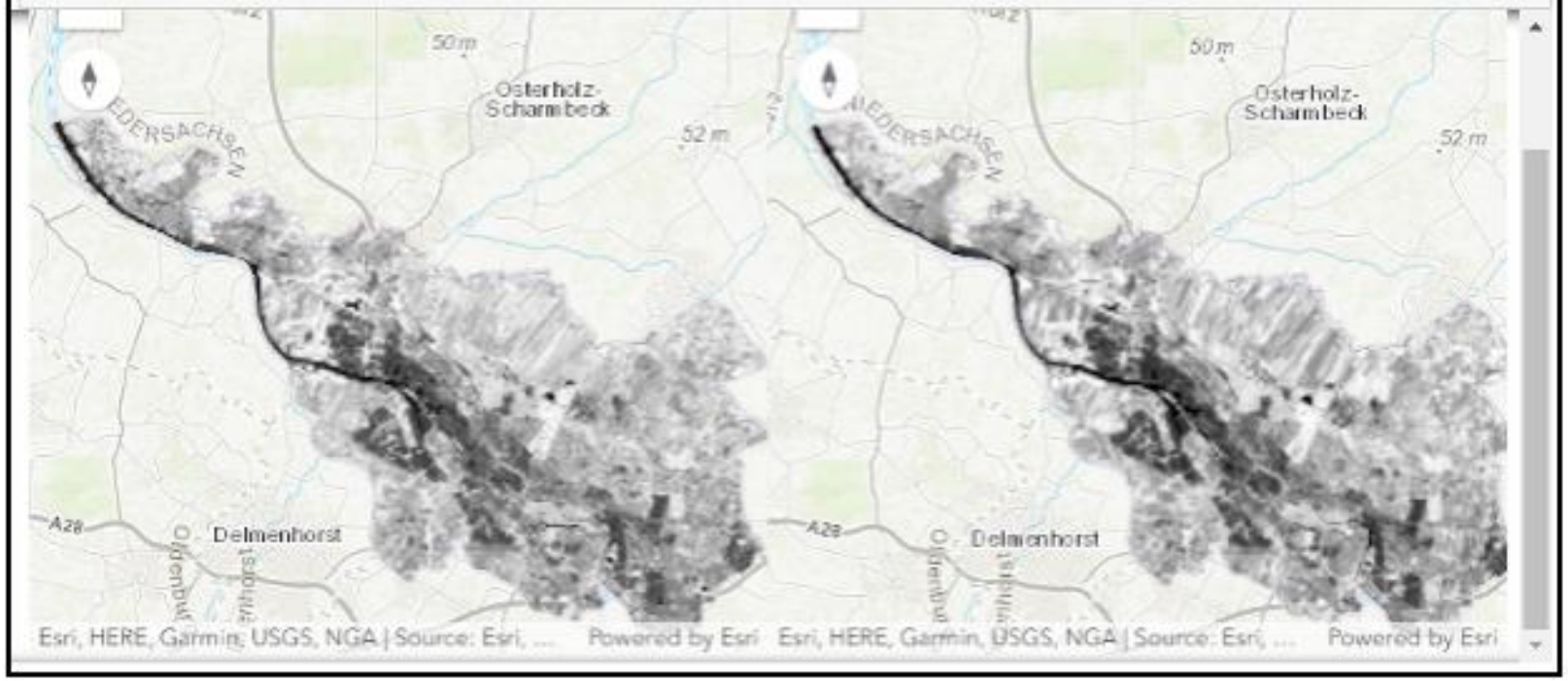
```
# use on click event on the map above to retrieve the x,y and spatial reference of whichever
# point is clicked on the map
import arcgis.raster
# define onclick function, supports two paramentes first one is map widget info, second point coordinates
def pltSpectralSig(discard, pointCoordinate):
    # draw a cirle around clicked point
    m.draw(pointCoordinate) #----->The only thing that cant be generalized so the function can be reused<-----
    #print(pointCoordinate)
    #print("You clicked ",pontCoordinate,'this is:',discard)
    #get sample from the clicked point_geoemtry
    pointSample = imageSelection.get_samples(pointCoordinate) # change imagery layer to retrieve desired values
    #access the pixel values from the sample
    values = pointSample[0]['value']
    vals = [float(int(s)/100000) for s in values.split(' ')] # is this normalizing the values otherwise erros (BAD_COLUMN_NAME)
    y = vals
    # this has now to be changed manually known number of bands
    #TBC to wavelenghts

    #idd = imageSelection.identify(geometry=pointCoordinate)
    #print(pointCoordinate)

    #[thing for thing in list_of_things]
    x=[i for i in range(1,imageSelection.band_count+1)]
    # bnds= [coasta, blue,green,red,NIR,Swir]
    #x= ['1','2','3','4','5','6','7','8','9','10','11','12','13']
    # create a plot figure with given x and y
    myPlot = figure(title="Spectral Signature at point: "+ "x: " +str(pointSample[0]['location']['x'])+" and "+
                "y: " +str(pointSample[0]['location']['y']),
                plot_width=600, plot_height=300,
                x_axis_label='Spectral Bands',
                y_axis_label='Data Values')
    #myPlot.y_range=Range1d(0, 1.0) # for visualization purposes
    #plot the line using x and y
    myPlot.line(x, y, legend="Selected Point", line_color="pink", line_width=2)
    # plot a circle on each x and y of line representing value per band
    myPlot.circle(x, y, line_color="black", fill_color="pink", size=5)

    #myPlot.xgrid.band_fill_alpha = 0.1
    #myPlot.xgrid.band_fill_color = "navy"
    #myPlot.quad(top=max(vals), bottom=0, left=1, right=2, color="#BBBBBB", alpha=0.5)
    p2 = BoxAnnotation(left=1.5, right=2.5, fill_alpha=0.45, fill_color='blue')
    p3 = BoxAnnotation(left=2.5, right=3.5, fill_alpha=0.45, fill_color='green')
    p4 = BoxAnnotation(left=3.5, right=4.5, fill_alpha=0.45, fill_color='red')
    p5 = BoxAnnotation(left=4.5, right=5.5, fill_alpha=0.45, fill_color='#ffff01')
    p6 = BoxAnnotation(left=5.5, right=6.5, fill_alpha=0.45, fill_color='#fcbb04')
    p7 = BoxAnnotation(left=6.5, right=7.5, fill_alpha=0.45, fill_color='#fda303')
    myPlot.renderers.extend([p2, p3, p4, p5, p6, p7])
    #myPlot.ygrid.grid_line_color = None

    #display
    show(myPlot)

print('Click on the map above to create a spectral profile of the point')
m.on_click(pltSpectralSig)
```



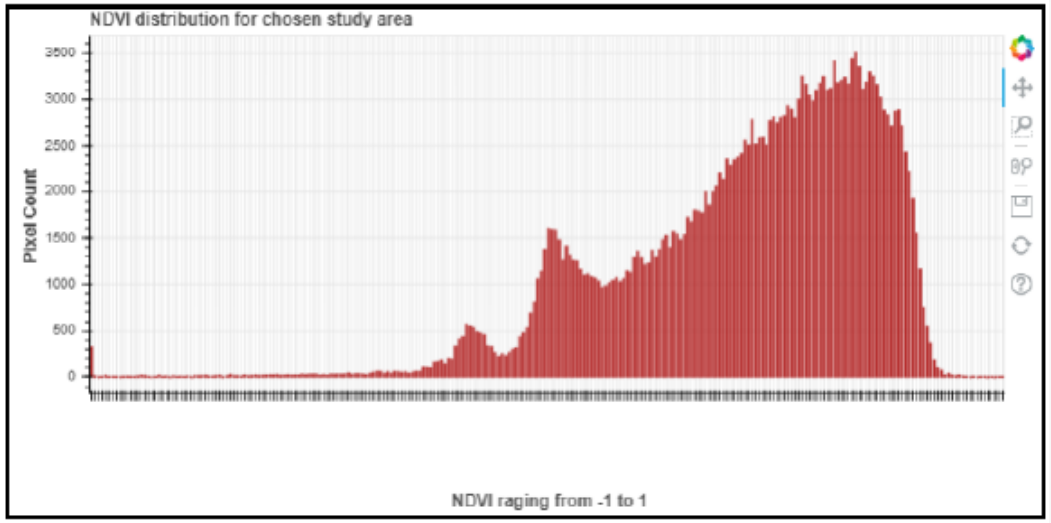
Module 3

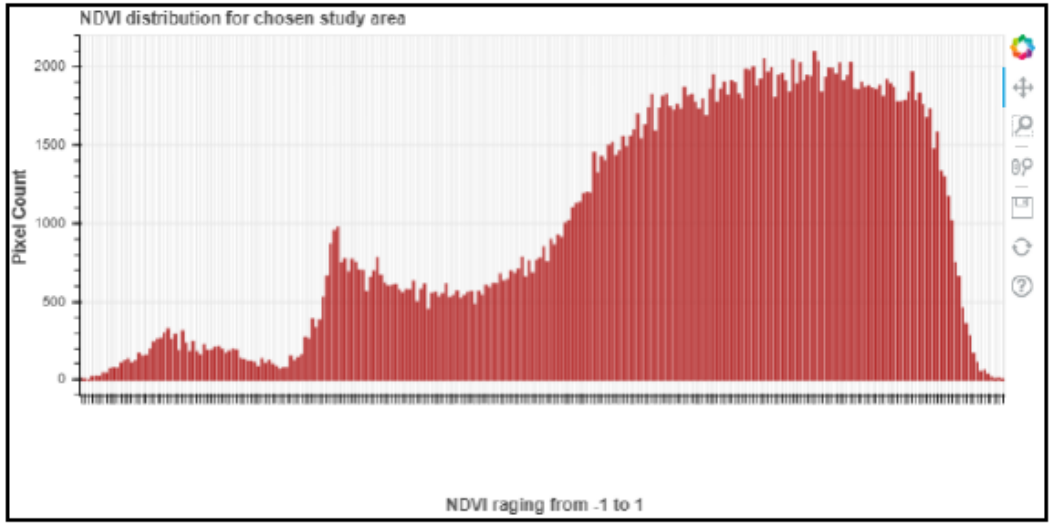
Example : Simple change detection in Bremen

```
# display items on map
m1 = agol_gis.map('bremen')
m1.add_layer(old)
m2 = agol_gis.map('bremen')
m2.add_layer(new)
#m1.layout= Layout(flex='1 1', padding='6px', height='450px')
#m2.layout= Layout(flex='1 1', padding='6px', height='450px')
m1label = widgets.HTML('Study area NDVI on: '+ oldDate.value)
m2label = widgets.HTML('Study area NDVI on:'+ newDate.value)
btxt = widgets.HBox([m1label,m2label])
#btxt
b=widgets.HBox([m1,m2])
b
```



esri Deutschland
THE SCIENCE OF WHERE

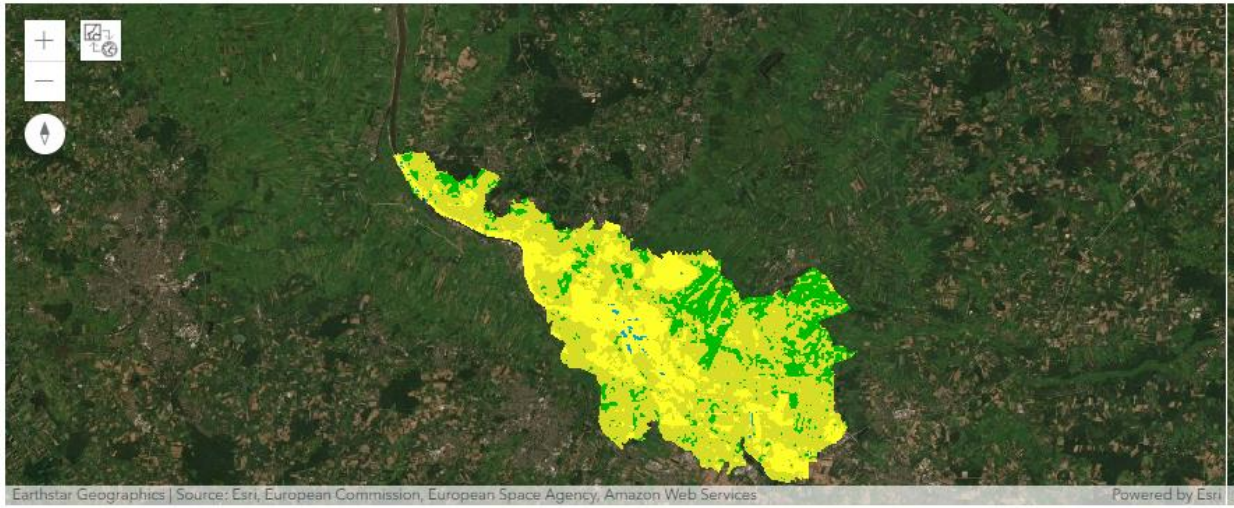


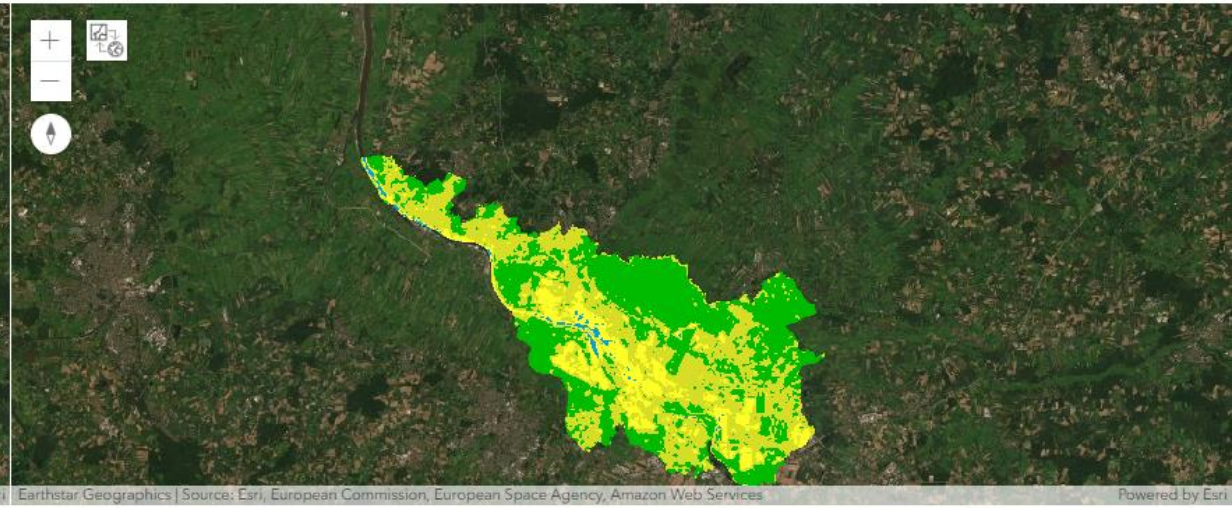



```
###### from bokeh.plotting import figure, show, output_file
from bokeh.models import NumeralTickFormatter

#####Function to plot based on group size and data (counts), returns bokeh histogram.######
def plotHistograms(counts, nGroups):
    #counts is a variable (topValUngrouped = statsHistold['histograms'][0]['counts']) statsHistold will change eveytime a
    #nGroups is a number divisible by 256 of our choosing

    assert (256 % nGroups == 0), "Number of groups must be a divisor of 256" #This is making our custom error message: if

    #Divied list up in nGroups sublists, taken from https://stackoverflow.com/questions/9671224/split-a-python-list-into-c
    chunks = [counts[x:x+int(256/nGroups)] for x in range(0, len(counts), int(256/nGroups))]

    # Sum all items in each sublist and store these in a new list.
    topValGrouped=[]
    i=0
    while i < len(chunks):
        topValGrouped.append(sum(chunks[i]))
        i+=1

    # Create names for each bar.
    start = -1.0
    groupIncrement = 2/nGroups
    xLabel = [] # creating empty list to input str from count plus each increment
    while start < 1:
        xLabel.append(str(round(start, 2)) + " to " + str(round(start+groupIncrement, 2)))
        start+=groupIncrement

    #print(len(xLabel), groupIncrement, len(chunks))

    #Plotting, dynamic width depending on amount of groups.
    p = figure(title='NDVI distribution for chosen study area',plot_width=800, plot_height=400, x_range=xLabel)
    p.vbar(x=xLabel, width=0.5, bottom=0,
           top=topValGrouped, color="firebrick")
    p.yaxis.formatter=NumeralTickFormatter(format="0")
    p.xaxis.major_label_orientation = 1 # change label x orientation
    p.yaxis.axis_label = "Pixel Count"
    p.xaxis.axis_label_text_font_style = "bold"
    p.xaxis.axis_label =" NDVI raging from -1 to 1" # labeling axis
    p.yaxis.axis_label_text_font_style = "bold" # styling

    #If we have too many bars, remove labels on the x-axis:
    if nGroups > 32:
        p.xaxis.major_label_text_color = None

    return p
```

NDVI Classification

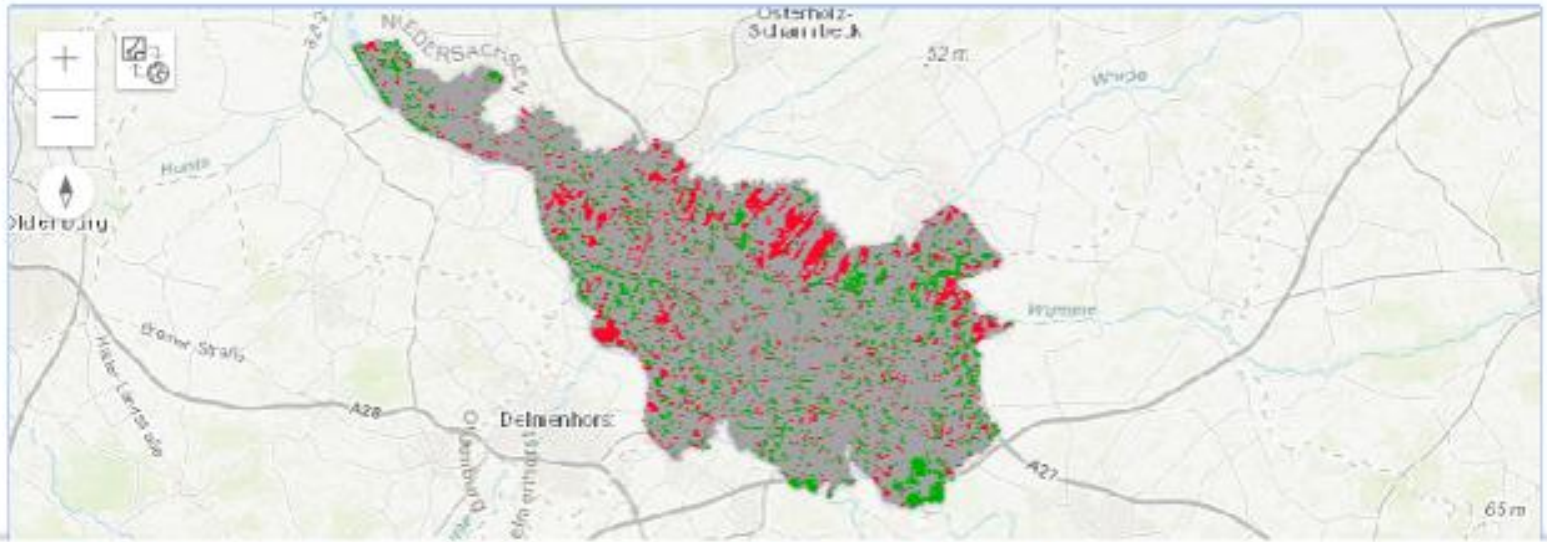
In [73]:
```
mm = agol_gis.map(choice.split( )[2])
mm.add_layer(oldremap)
mm.basemap='satellite'
mm1 = agol_gis.map(choice.split( )[2])
mm1.add_layer(newremap)
mm1.basemap='satellite'
b = widgets.HBox([mm,mm1])
b
```

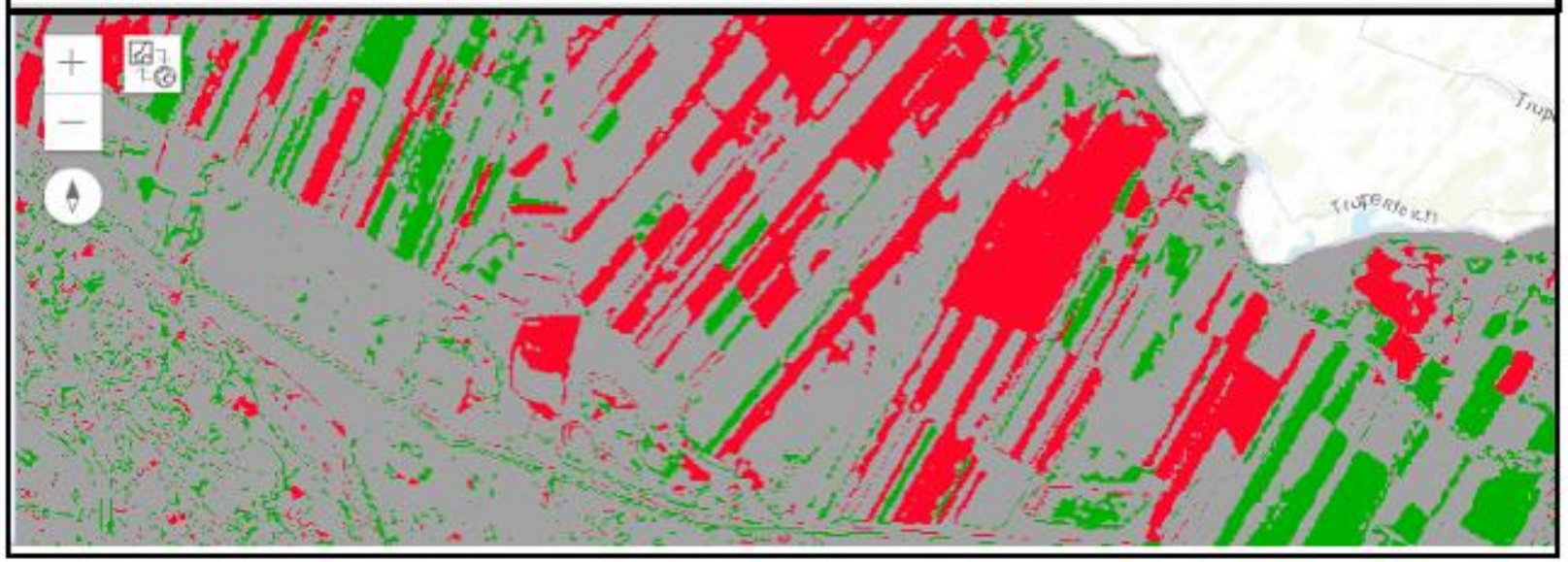




```
# Displa difference in NDVI from two dates reclassified as loss, gain and no change
m =egol_gis.map('bremen')
print('Difference in NDVI classified as no change (gray), loss (red), gain (green)')
m.add_layer(remapdiff)
m
```

```
Difference in NDVI classified as no change (gray), loss (red), gain (green)
```

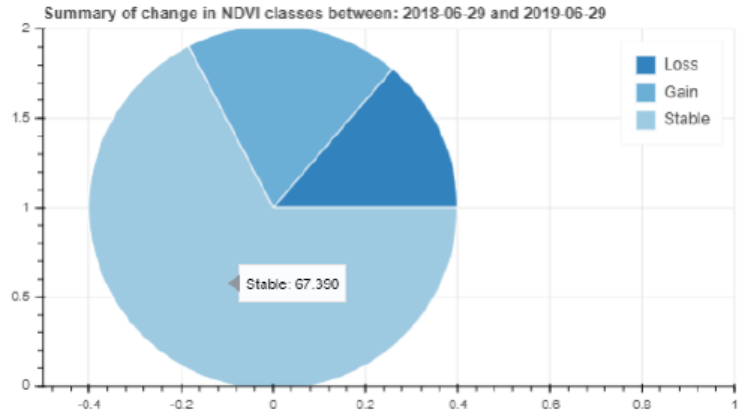




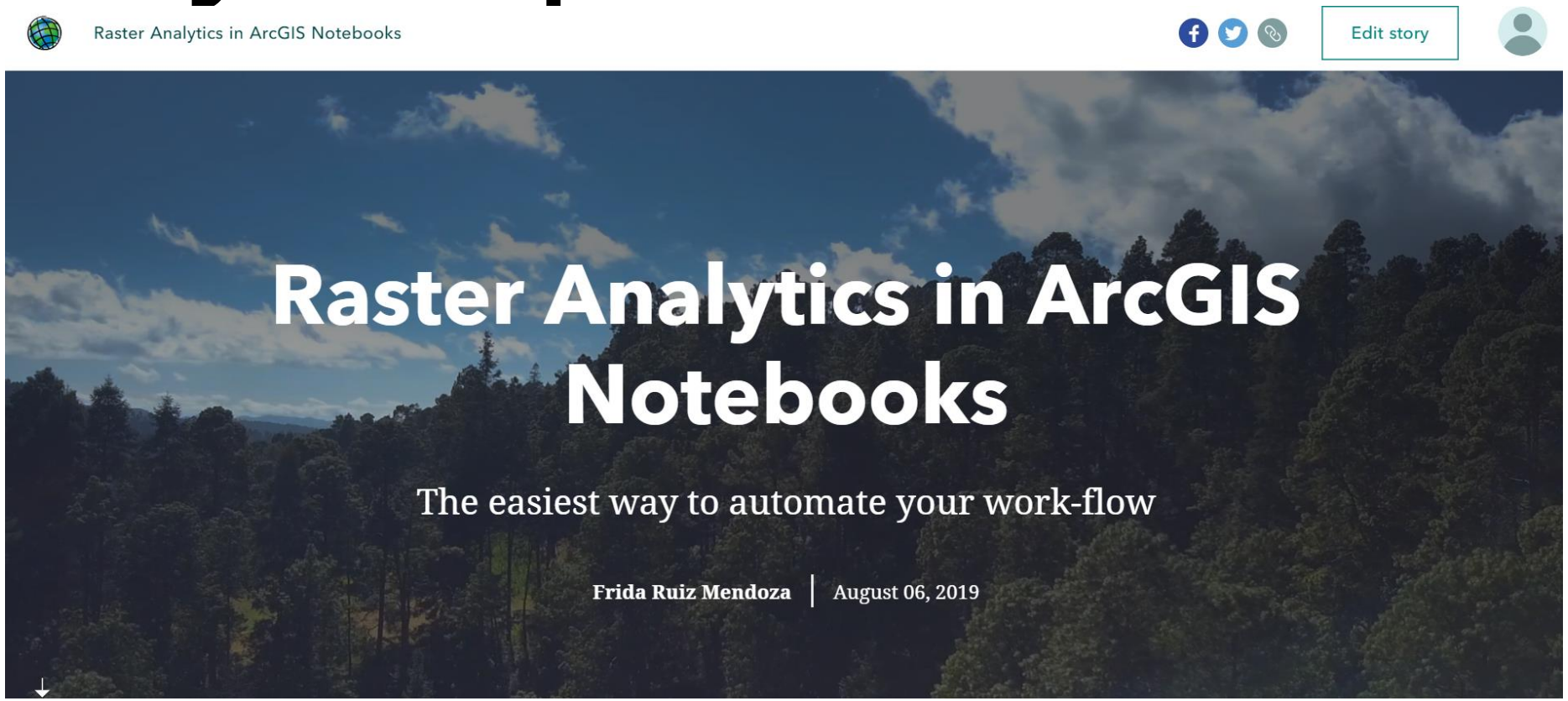
```
# Summary : What is the percentage of area that
x = percentChange
data =pd.Series(x)# making our dictionary a pan
#data# show df
data.reset_index(name='Change class').rename(co
```

	Change Class	Change class
0	Loss	13.80
1	Gain	18.82
2	Stable	67.39

```
x= percentChange# input data which is the reclass histogram ,changed manually
data = pd.Series(x).reset_index(name='value').rename(columns={'index':'class'})# change
data['angle'] = data['value']/data['value'].sum() * 2*pi
data['color'] = Category20c[len(x)]

p = figure(plot_height=350, title="Summary of change in NDVI classes between: "+oldDate.
           tools="hover", tooltips="@class: @value", x_range=(-0.5, 1.0))# x_range is wh

p.wedge(x=0, y=1, radius=0.4,
        start_angle=cumsum('angle', include_zero=True), end_angle=cumsum('angle'),
        line_color="white", fill_color='color', legend='class', source=data)

show(p)
```



Story Map



https://storymaps.arcgis.com/stories/41ede17ec6754aae81ea87d072902535

Future work and lessons learned

Lessons learned:

- Working with ONE item (image) has no restrictions to calculate pixel values

- Working with MORE than one item requires understanding of **Mosaic_rule** and works best when using chained raster functions.

- Imagery Layers have **maximum Height and Width** -> # of pixels you can export or make calculations on -> modify pixel_size

- Way of reaching pixel values done by calculating statistics and deriving pixel count per value -> explore working with arrays

CONCLUSIONS

- Straightforward to use the service and quick to filter images
- Convenient to publish Web applications through the Esri Platforms
- API updated often (documentation is tricky)
- Sharing and reproducing your workflow easy
- Integration of both Esri and open source libraries
- User interface from Arcigs Enterprise constantly updated and linked to Notebooks

THANK YOU !

esri Deutschland
THE SCIENCE OF WHERE

esri Deutschland