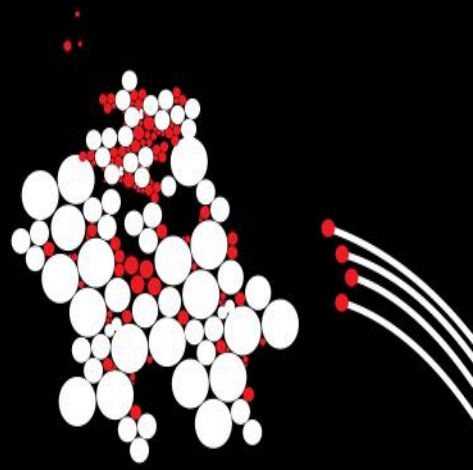
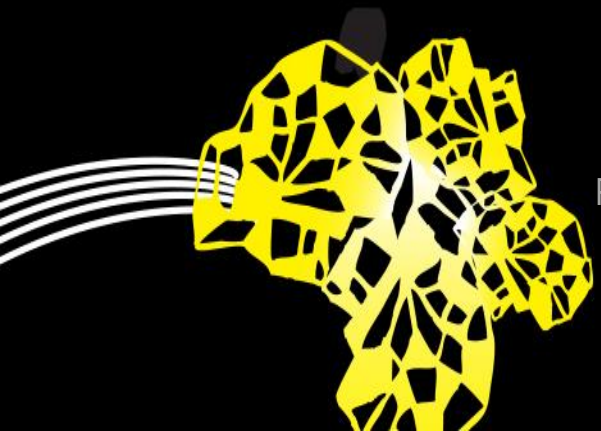
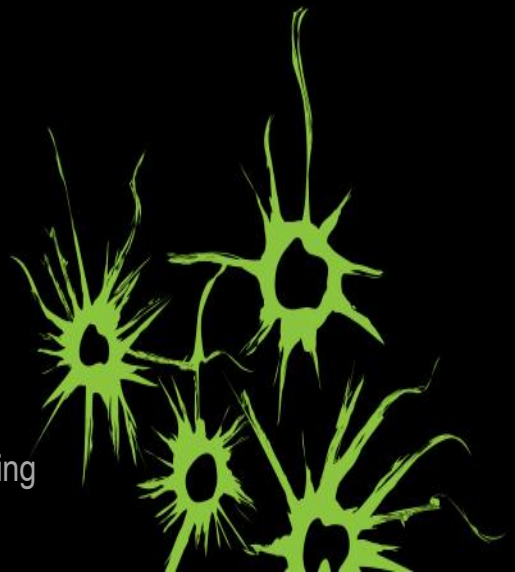
UNIVERSITY OF TWENTE.



# Integrating *Raster* and *Vector data*

***More on Python and geospatial data***





Presented by:
Mahdi Farnaghi
Assistant Professor,
Department of Geo-information Processing
ITC, The University of Twente

# Agenda

Introduction

*Warp* function

*RasterizeLayer* function

*Polygonize* function

Sample analysis using *numpy, gdal (raster)* and *ogr (vector)*

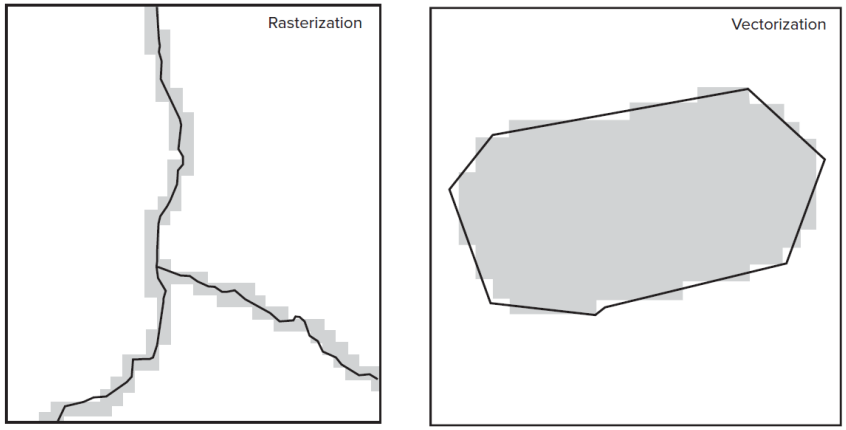
# Introduction

# Introduction

Spatial Data Analysis on Raster and Vector data

Simple solutions:

- Rasterization: Convert vector data to raster
- Vectorization: Convert raster data to vector




Chang (2019), Introduction to Geographic Information Systems

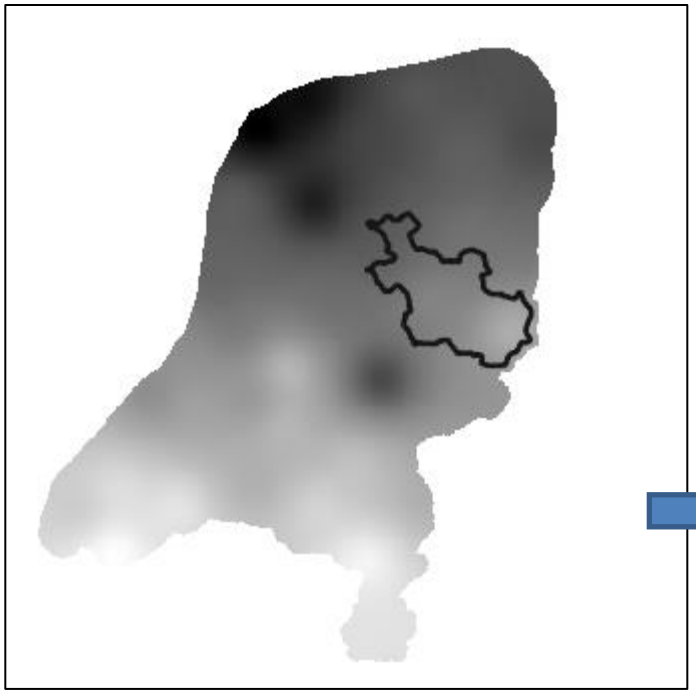
Advance analysis may require several transformations from raster to vector and vector to raster spaces

# Warp

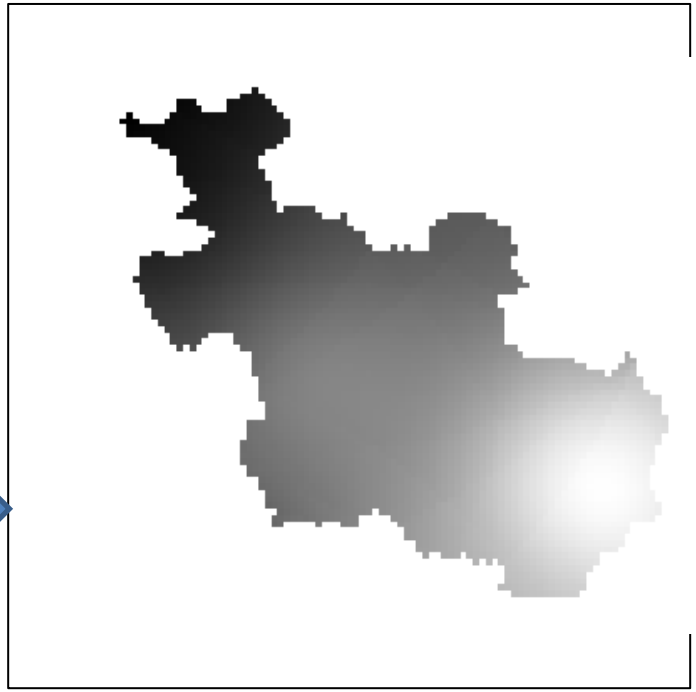
# Subsetting a raster based on a polygon

*Determine temperature pixels inside Overijssel province only.*

Temperature values



Temperature values



# Subsetting a raster based on a polygon

Use **gdal.Warp()** with the following cutline options:

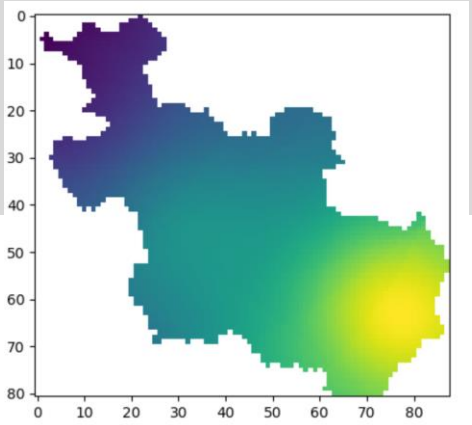
- cutlineDSName - cutline *ogr* polygon dataset name
  e.g., path to shapefile;
- cutlineLayer – cutline polygon layer;
- cutlineWhere - restrict desired cutline features based on attribute query.

This solution only works with **polygons**.

# Subsetting a raster based on a polygon

```
from osgeo import ogr, osr, gdal, gdal_array as gdarr
ds = gdal.Open("2014.tif")
nlprovPath=r'NL_provinces.shp'
overTemperatureDs =gdal.Warp('temOverijssel.tif', ds,
format="GTiff", dstSRS='EPSG:28992',
cutlineDSName=nlprovPath, cutlineWhere="NAME_1 =
'Overijssel'", cropToCutline = True)

#To confirm lets show the temperature for the first day
overTempArray=gdarr.DatasetReadAsArray(overTemperatureDs, 0,
0, overTemperatureDs.RasterXSize,
overTemperatureDs.RasterYSize)
overTempArray[overTempArray == -9999] = None
plt.imshow(overTempArray[0,:,:])
plt.show()
print("Finished.")
```



# Rasterize

# Rasterize a vector layer

**gdal.RasterizeLayer(dataset, bands, layer, burn_values, options)**

- dataset : the output raster dataset
- bands : a **list** of bands to be updated
- layer : the input vector layer
- burn_values: a **list** of values to burn into the raster (optional)
- options : a **list** of special options controlling rasterization

"ATTRIBUTE": Identifies an attribute field on the features to be used for a burn-in value. The value will be burnt into all output bands. **If specified, burn_values will not be used**.

- *E.g., options=['ATTRIBUTE=FCode']*

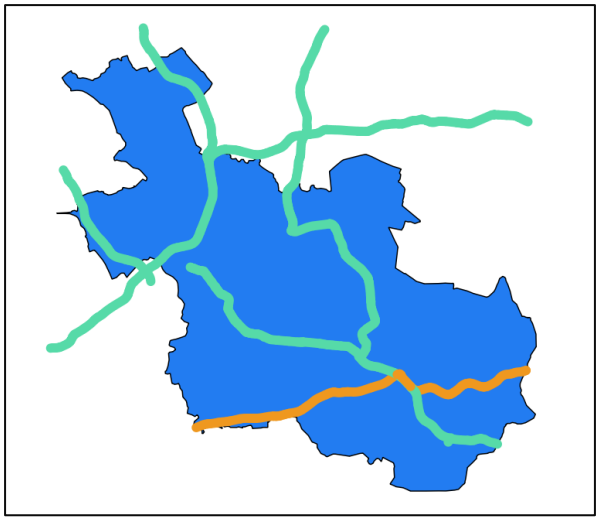
"ALL_TOUCHED": May be set to TRUE to set all pixels touched by the line or polygons, not just those whose center is within the polygon or that are selected by **brezenham's line algorithm**. Defaults to FALSE.

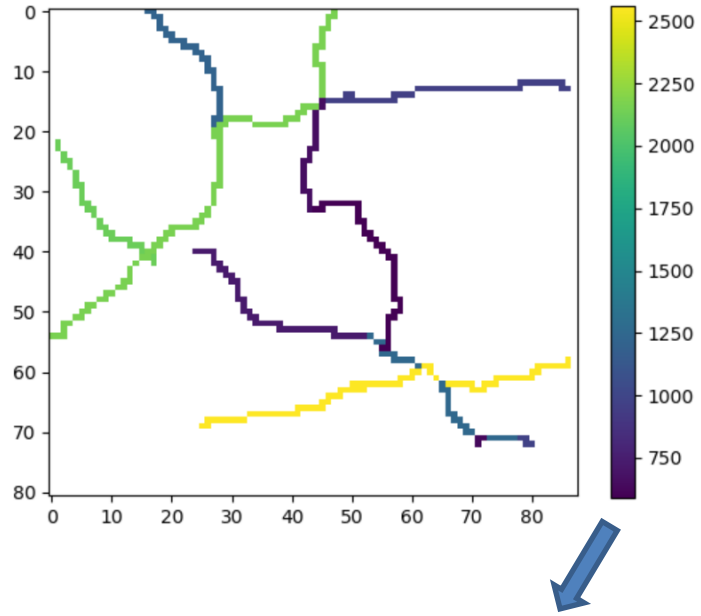
"BURN_VALUE_FROM": May be set to "Z" to use the Z values of the geometries. The value from burn_values or the attribute field value is added to this before burning. In default case a burn value is burned as it is. This is implemented properly only for points and lines for now. Polygons will be burned using the Z value from the first point. The M value may be supported in the future.

"MERGE_ALG": May be REPLACE (the default) or ADD. REPLACE results in overwriting of value, while ADD adds the new value to the existing raster, suitable for heatmaps for instance.

# Rasterize a vector layer

How to rasterize all the roads using number of cars per hour?






Pixel value = Number of cars per hour

# Rasterize a vector layer

*4 steps*

## Subset only Overijssel Temperatures

- *in memory*
- to use later as a reference

## Create an empty raster

- in memory
- to carry the rasterized version of the road layer

## Rasterization

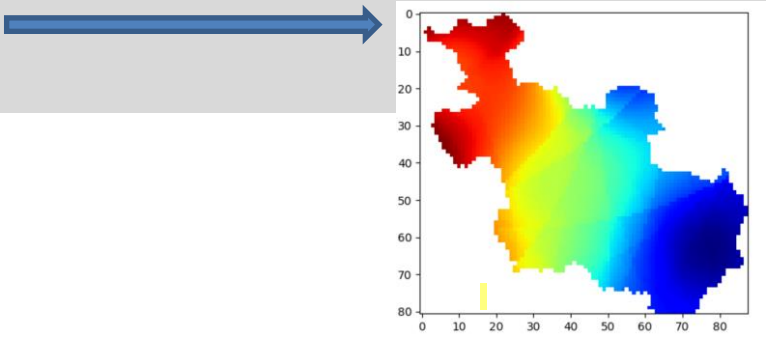
## Plot the output raster

# Rasterize a vector layer, I

```
from osgeo import ogr,gdal, gdal_array as gdarr
from matplotlib import pyplot as plt
import numpy as np
import os

dataDirectory=r'C:\data'
os.chdir(dataDirectory)

# subset only Overijssel Temperatures to use later as a reference
overTemperatureDs=gdal.Warp('',
"2014.tif",format="Mem",dstSRS='EPSG:28992',cutlineDSName='NL_provinces.s
hp',cutlineWhere="NAME_1 = 'Overijssel'", dstNodata=-9999,cropToCutline =
True, outputType=gdal.GDT_Float32)
overTempArray=gdarr.DatasetReadAsArray(overTemperatureDs, 0, 0,
overTemperatureDs.RasterXSize, overTemperatureDs.RasterYSize)
```



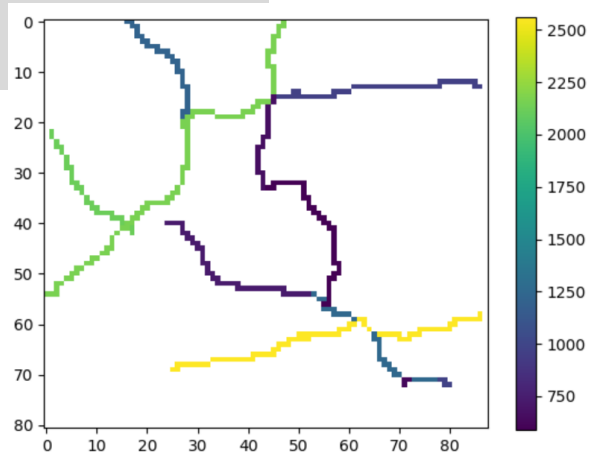
# Rasterize a vector layer, II

```
#Prepare for Rasterize
memDriver = gdal.GetDriverByName('Mem')
roadsRasterDs =
memDriver.Create('',overTemperatureDs.RasterXSize,overTemperatureDs.RasterYS
ize,1,gdal.GDT_Float32)
# set projection
roadsRasterDs.SetProjection(overTemperatureDs.GetProjection())
roadsRasterDs.SetGeoTransform(overTemperatureDs.GetGeoTransform())

    # set geotransform
    #  create 1 band and set the nodata value
outband1 = roadsRasterDs.GetRasterBand(1)
outband1.SetNoDataValue(0)
```

# Rasterize a vector layer III

```
roadsVectorDs = ogr.Open("ovRoads.geojson")
roadsLayer = roadsVectorDs.GetLayer()
 #  rasterize A1 road in Overijssel
gdal.RasterizeLayer(roadsRasterDs, [1], roadsLayer,
options=['ATTRIBUTE=vehic_p_hour'])
roadsArray=gdarr.DatasetReadAsArray(roadsRasterDs, 0, 0,
roadsRasterDs.RasterXSize, roadsRasterDs.RasterYSize)
roadsArray[roadsArray == 0] = None
plt.imshow(roadsArray)
plt.colorbar()
plt.show()
```



# Polygonize

# Polygonize a raster layer

Convert raster into vector:

- *creates vector polygons for all connected regions of pixels sharing a common pixel value*
- *Each polygon is created with an attribute indicating the pixel value of that polygon*

**gdal.Polygonize(source band, mask band, output layer, value field, options)**

More *information* [https://www.gdal.org/gdal_polygonize.html](https://www.gdal.org/gdal_polygonize.html)

# Sample analysis using numpy, gdal (raster) and ogr (vector)

# Example of integration

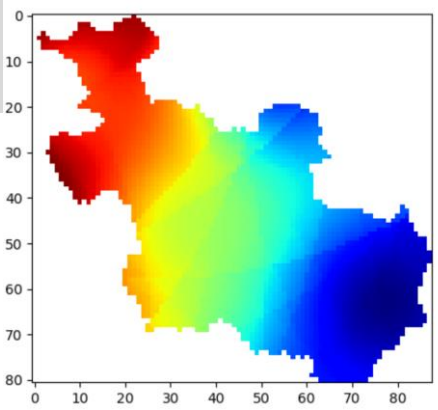
What were the lowest temperatures felt in 2014 on motorway A1 in the province of Overijssel?

- Subset the temperature dataset for Overijssel province
- Rasterize roads dataset (only A1 motorway)
- Convert rasterized dataset to numpy
- Convert the temperatures dataset to numpy
- Multiply A1 numpy array by the temperatures array
- Compute the lowest temperature using numpy functions

# Example of integration

```
from osgeo import ogr,gdal, gdal_array as gdarr
import numpy as np
import os

dataDirectory=r'C:\data'
os.chdir(dataDirectory)
#Subset only Overijssel Temperatures
overTemperatureDs=gdal.Warp('',
"2014.tif",format="Mem",dstSRS='EPSG:28992',cutlineDSName
='NL_provinces.shp',cutlineWhere="NAME_1 = 'Overijssel'",
dstNodata=-9999,cropToCutline = True,
outputType=gdal.GDT_Float32)
overTempArray=gdarr.DatasetReadAsArray(overTemperatureDs,
0, 0, overTemperatureDs.RasterXSize,
overTemperatureDs.RasterYSize)
```



# Example of integration

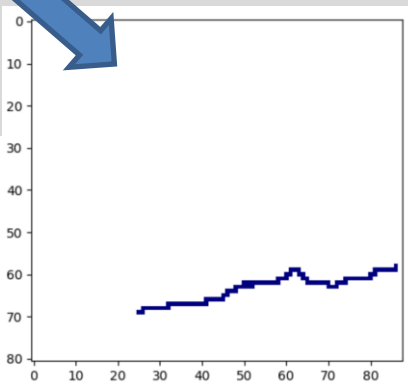
```
#  prepare for Rasterize Road A1
memDriver = gdal.GetDriverByName('Mem')
roadsRasterDs =
memDriver.Create('',overTemperatureDs.RasterXSize,overTemperatureDs.RasterYS
ize,1,gdal.GDT_Float32)
roadsRasterDs.SetProjection(overTemperatureDs.GetProjection())    # set projection
roadsRasterDs.SetGeoTransform(overTemperatureDs.GetGeoTransform())
#  set geotransform
#  create 1 band and set the nodata value
outband1 = roadsRasterDs.GetRasterBand(1)
outband1.SetNoDataValue(0)

roadsVectorDs = ogr.Open("ovRoads.geojson")
roadsLayer = roadsVectorDs.GetLayer()
roadsLayer.SetAttributeFilter("id = 'A1'")
```

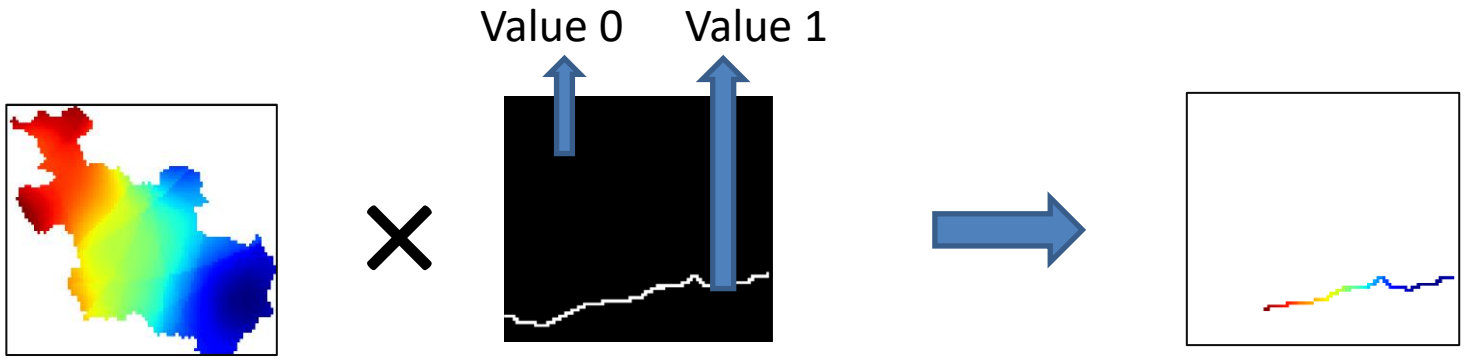
# Example of integration

```
#  rasterizing A1 road in Overijssel
gdal.RasterizeLayer(roadsRasterDs, [1], roadsLayer, burn_values=[1],
options=['ALL_TOUCHED=TRUE'])
roadsArray=gdarr.DatasetReadAsArray(roadsRasterDs, 0, 0,
roadsRasterDs.RasterXSize, roadsRasterDs.RasterYSize)

#  clean no needed GDAL datasets from memory
overTemperatureDs=None
roadsVectorDs=None
roadsRasterDs=None
```



# Multiplying two numpy arrays




Temeprature array

A1 motorway array

# Example of integration

```
print('A1 road shape:',roadsArray.shape)
print('Overijssel temperature shape:',overTempArray.shape)
#Multiply roads by temperature. The output will be temperatures.
roadsTemperature=roadsArray*overTempArray
print('A1 road temperatures shape:',roadsTemperature.shape)
#Compute lower temperatures
roadsMinTemperature=np.min(roadsTemperature, axis=0)
print('A1 lower temperatures shape',roadsMinTemperature.shape)
#Removing -9999 values
roadsMinTemperature[roadsMinTemperature == -9999] = None

minLowerTemp=np.nanmin(roadsMinTemperature)
maxLowerTemp=np.nanmax(roadsMinTemperature)
print('Lowest temperatures recorded in 2014 in A1 in Overijssel
are between:',minLowerTemp,' and ',maxLowerTemp)
```

Lowest temperatures recorded in 2014 in A1 in Overijssel are between: -0.33159244 and 0.4753607

# Conclusion

# Summary

- Integration is important for solving real life problems
- *Rasterize* – vector to raster
- *Polygonize* – raster to vector
- *Numpy* is usually the package with which results are computed and it can produce data products such as vegetation indices.
- There are more libraries available.

# Thank you