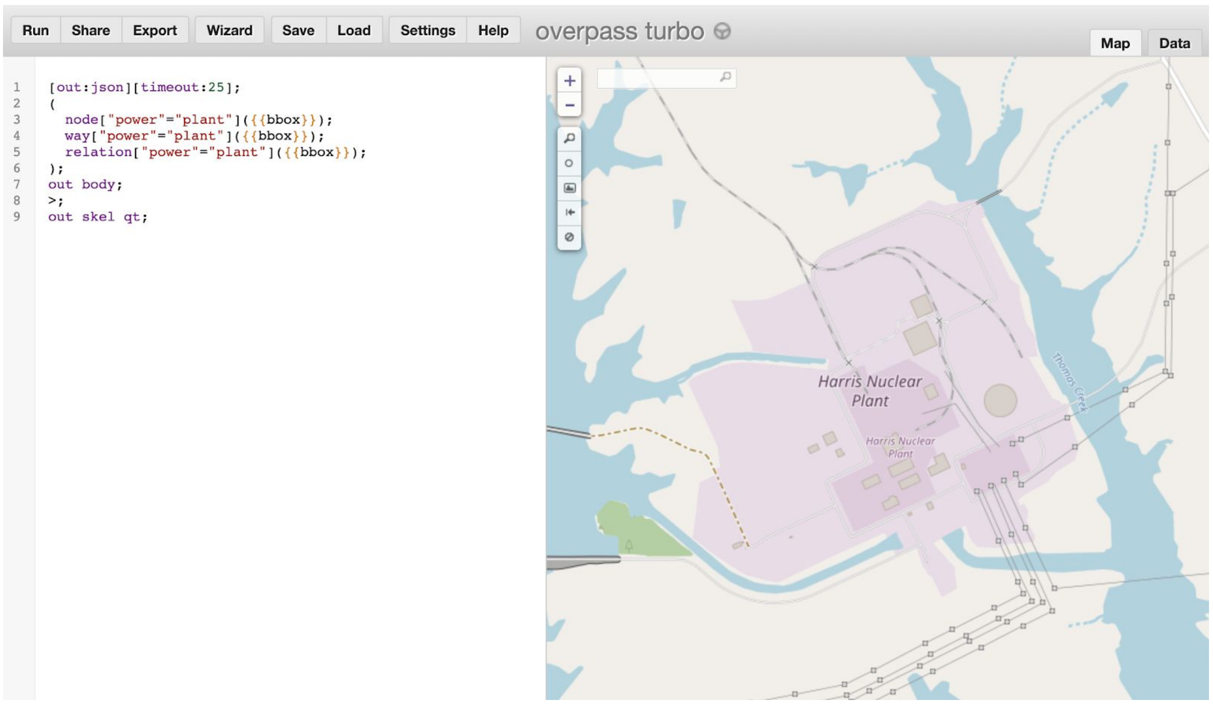
All files you need are in “mids_ei/data”

README for the pipeline

Step 1: download the geojson file

(we have an example “eastern_US.geojson” file containing power plants in eastern US so that you can skip this step for eastern US trial runs.
If you want to do trial runs, start from Step 2;
If you want to customize your region or object types, follow the steps below.)

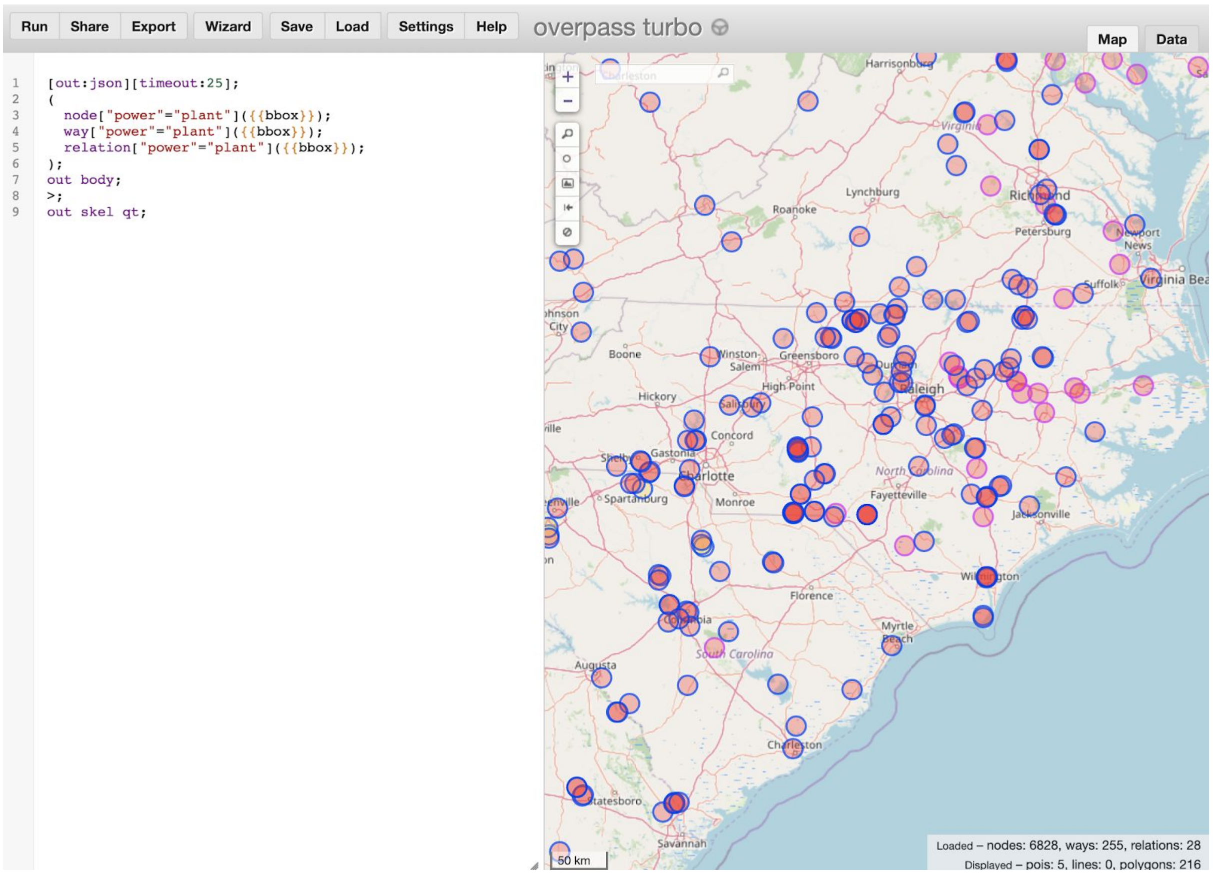
1. Enter the website:
https://overpass-turbo.eu/s/QBq



```
[out:json][timeout:25];
(
  node["power"="plant"]({{bbox}});
  way["power"="plant"]({{bbox}});
  relation["power"="plant"]({{bbox}});
);
out body;
>;
out skel qt;
```

If you want object other than power plant, change the parameter.

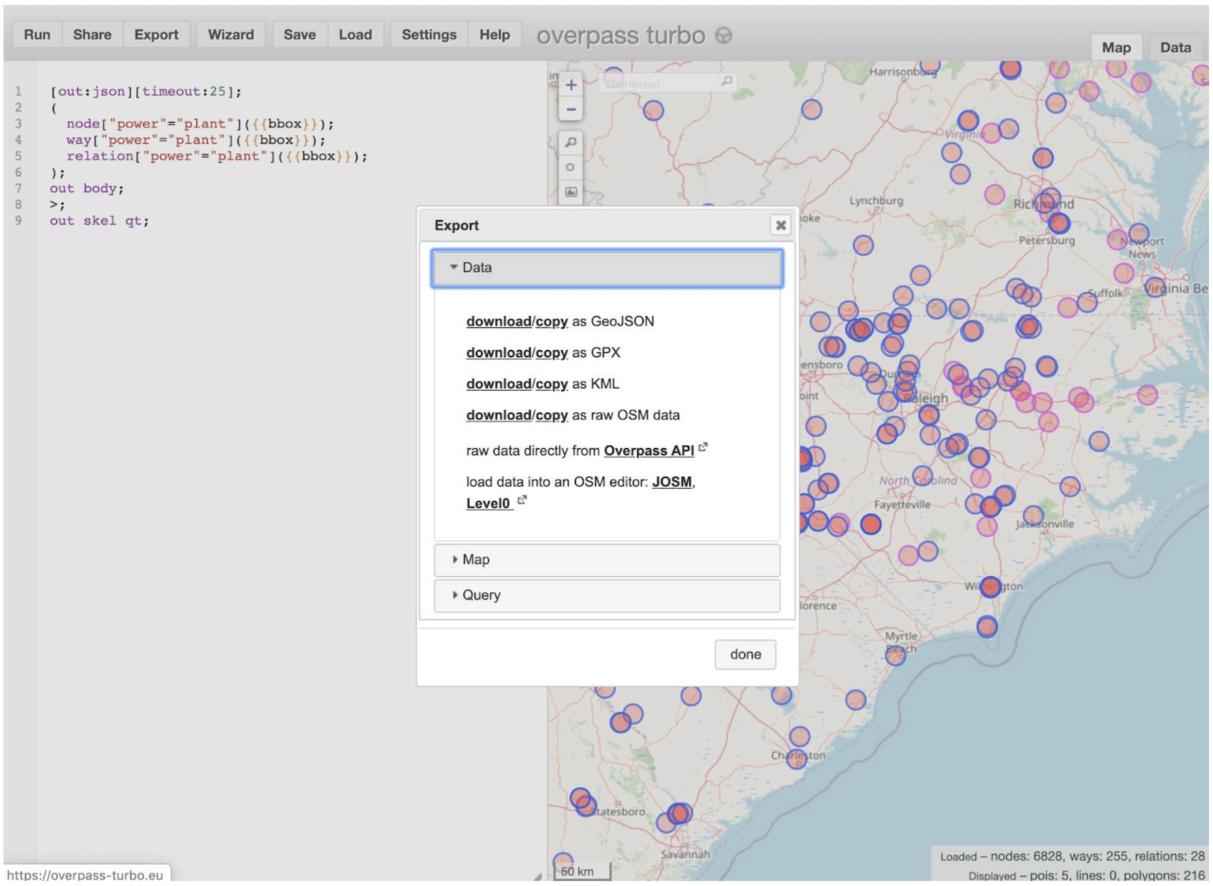
2. Zoom out to the region you want and click Run on the upper left:



```
[out:json][timeout:25];
(
  node["power"="plant"]({{bbox}});
  way["power"="plant"]({{bbox}});
  relation["power"="plant"]({{bbox}});
);
out body;
>;
out skel qt;
```

Loaded – nodes: 6828, ways: 255, relations: 28
Displayed – pois: 5, lines: 0, polygons: 216

3. Click export and download as Geojson



```
[out:json][timeout:25];
(
  node["power"="plant"]({{bbox}});
  way["power"="plant"]({{bbox}});
  relation["power"="plant"]({{bbox}});
);
out body;
>;
out skel qt;
```

download/**copy** as GeoJSON

download/**copy** as GPX

download/**copy** as KML

download/**copy** as raw OSM data

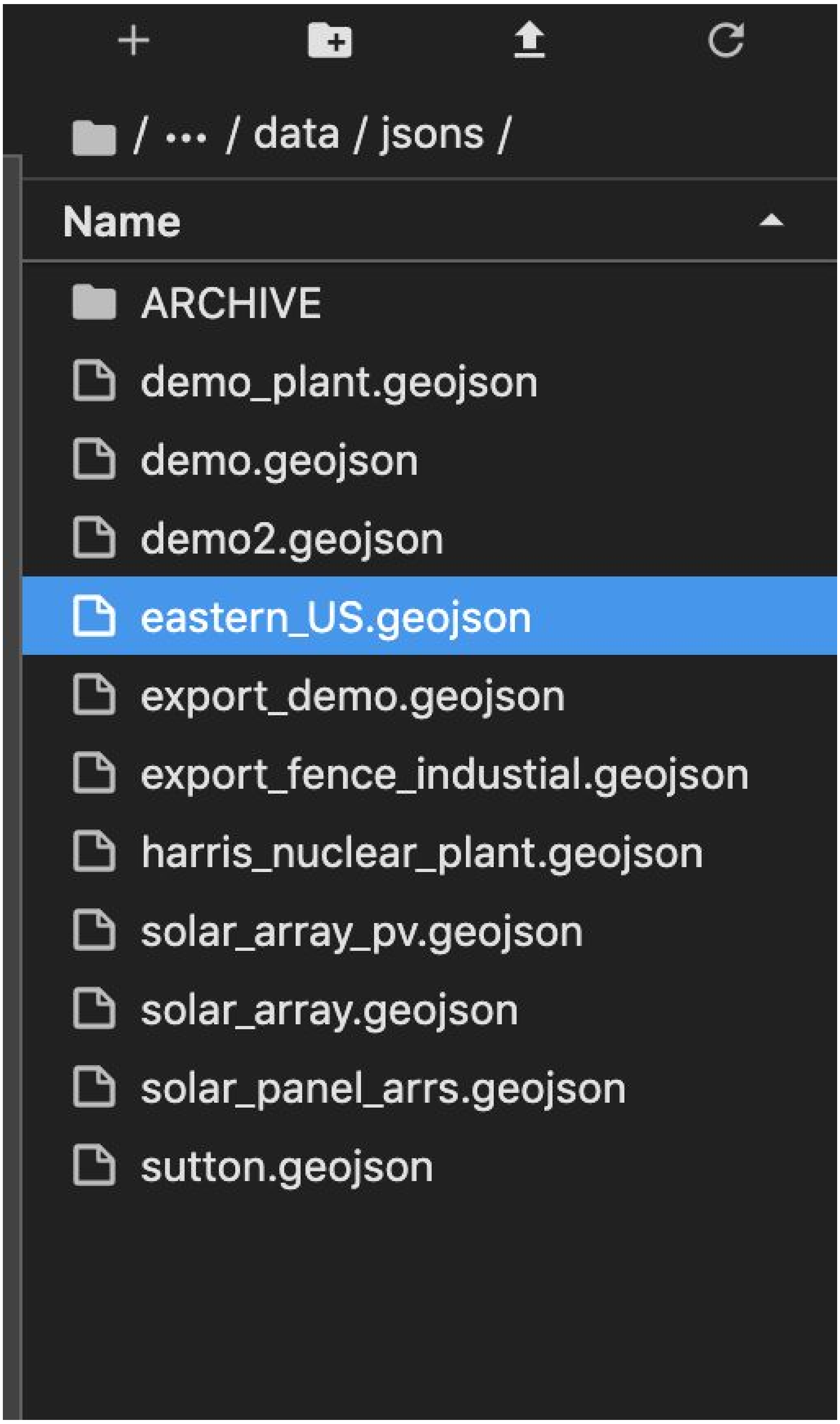
raw data directly from **Overpass API**

load data into an OSM editor: **JOSM**, **Level0**

Loaded – nodes: 6828, ways: 255, relations: 28
Displayed – pois: 5, lines: 0, polygons: 216

https://overpass-turbo.eu

4. put the file into the json directory

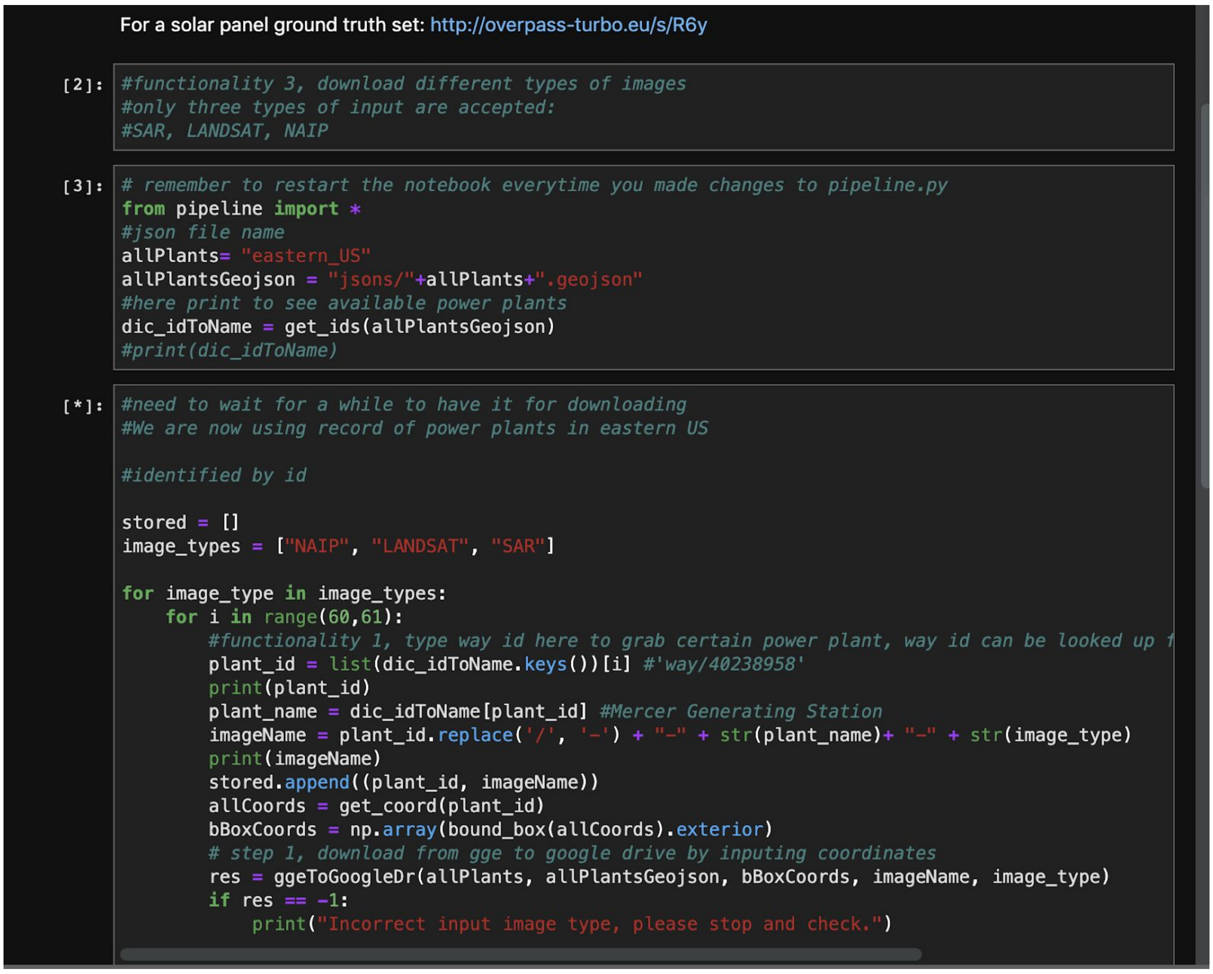


/ ... / data / jsons /

Name

ARCHIVE
demo_plant.geojson
demo.geojson
demo2.geojson
eastern_US.geojson
export_demo.geojson
export_fence_industial.geojson
harris_nuclear_plant.geojson
solar_array_pv.geojson
solar_array.geojson
solar_panel_arrs.geojson
sutton.geojson

Step 2: Start the pipeline: (All helper function codes are in pipeline.py)
2.1 Functionality 1: pick power plant(s) from exist record


For a solar panel ground truth set: http://overpass-turbo.eu/s/R6y

```
[2]: #functionality 3, download different types of images
     #only three types of input are accepted:
     #SAR, LANDSAT, NAIP
```

```
[3]: # remember to restart the notebook everytime you made changes to pipeline.py
     from pipeline import *
     #json file name
     allPlants= "eastern_US"
     allPlantsGeojson = "jsons/"+allPlants+".geojson"
     #here print to see available power plants
     dic_idToName = get_ids(allPlantsGeojson)
     #print(dic_idToName)
```

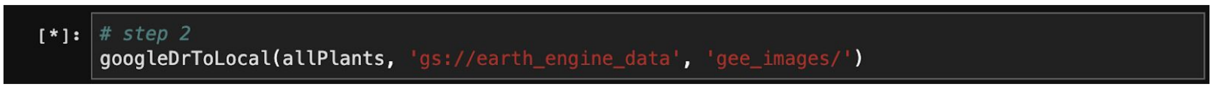
```
[*]: #need to wait for a while to have it for downloading
     #We are now using record of power plants in eastern US

     #identified by id

     stored = []
     image_types = ["NAIP", "LANDSAT", "SAR"]

     for image_type in image_types:
         for i in range(60,61):
             #functionality 1, type way id here to grab certain power plant, way id can be looked up f
             plant_id = list(dic_idToName.keys())[i] #'way/40238958'
             print(plant_id)
             plant_name = dic_idToName[plant_id] #Mercer Generating Station
             imageName = plant_id.replace('/', '-') + "-" + str(plant_name)+ "-" + str(image_type)
             print(imageName)
             stored.append((plant_id, imageName))
             allCoords = get_coord(plant_id)
             bBoxCoords = np.array(bound_box(allCoords).exterior)
             # step 1, download from gge to google drive by inputing coordinates
             res = ggeToGoogleDr(allPlants, allPlantsGeojson, bBoxCoords, imageName, image_type)
             if res == -1:
                 print("Incorrect input image type, please stop and check.")
```

1. Open the "NAIP-LANDSAT-SAR-gif-download.ipynb" file, and run the following blocks for Functionality 1:

   The first block (2) is for importing.
   The second block (3) shows all ids and names of the existing power plant we have (uncomment the print statement).
   The third block (*) starts to download file from google earth engine to google drive, several input parameters can be changed:
   image_types: the "LANDSAT" one generates a blue image now so probably need to be fixed in function "getLandsatTask(coords,name)" in pipeline.py file, the other two types of images are generated just fine.
   "allPlantsGeojson" as the current using Geojson name
   "range(60, 61)" as an example trial to download the 60th power plants in dictionary "dic_idToName", can be changed to generate more power plants (e.g. range(100)).

   this one need some time to run, which can be done in parallel later to improve its performance.

2. Run the following blocks for Functionality 1:



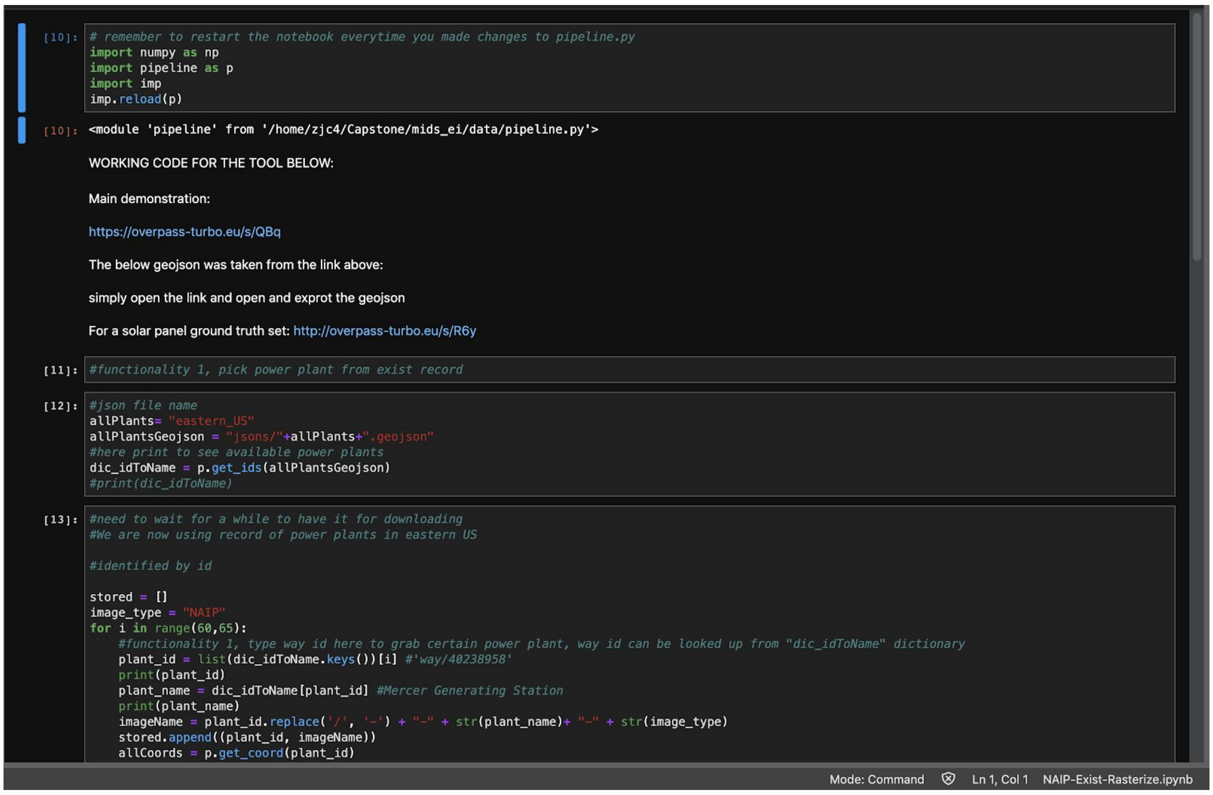
```
[*]: # step 2
     googleDrToLocal(allPlants, 'gs://earth_engine_data', 'gee_images/')
```

This block downloads all tif image from google drive to local machine. We can find all tif images downloaded from “gee_images/earth_engine_data/”. In this function, the second input is the Google Drive address, and the third is the local address.

3. We can find the gif image results in “gee_images/earth_engine_data/”.

2.2 Functionality 2: pick power plant(s) from exist record

1. Open the “NAIP-Exist-Rasterize.ipynb” file, and run the following blocks for Functionality 2:



```
[10]: # remember to restart the notebook everytime you made changes to pipeline.py
      import numpy as np
      import pipeline as p
      import imp
      imp.reload(p)

[10]: <module 'pipeline' from '/home/zjc4/Capstone/mids_ei/data/pipeline.py'>
```

WORKING CODE FOR THE TOOL BELOW:

Main demonstration:

https://overpass-turbo.eu/s/QBq

The below geojson was taken from the link above:

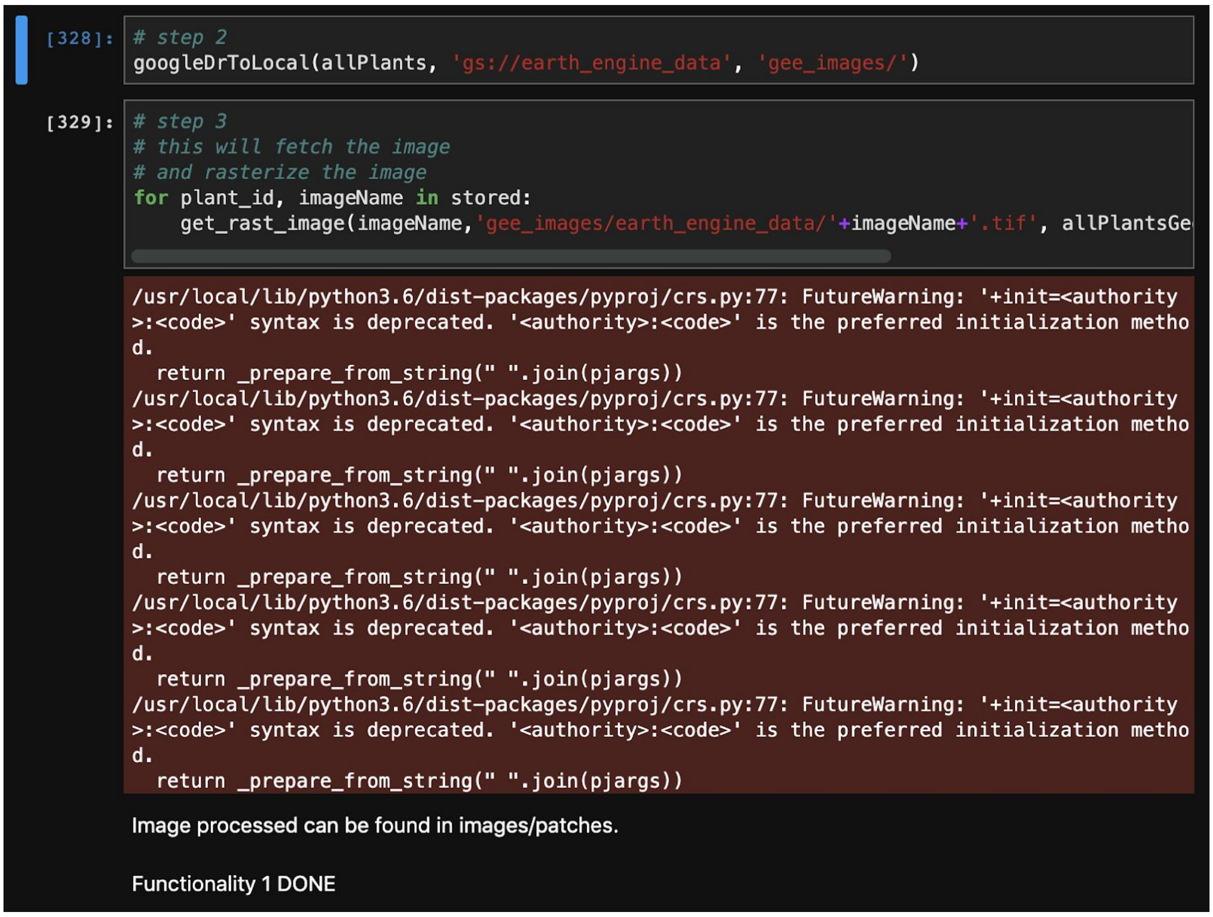
simply open the link and open and exprot the geojson

For a solar panel ground truth set: http://overpass-turbo.eu/s/R6y

```
[11]: #functionality 1, pick power plant from exist record

[12]: #json file name
      allPlants= "eastern_US"
      allPlantsGeojson = "jsons/"+allPlants+".geojson"
      #here print to see available power plants
      dic_idToName = p.get_ids(allPlantsGeojson)
      #print(dic_idToName)

[13]: #need to wait for a while to have it for downloading
      #We are now using record of power plants in eastern US

      #identified by id

      stored = []
      image_type = "NAIP"
      for i in range(60,65):
          #functionality 1, type way id here to grab certain power plant, way id can be looked up from "dic_idToName" dictionary
          plant_id = list(dic_idToName.keys())[i] #'way/40238958'
          print(plant_id)
          plant_name = dic_idToName[plant_id] #Mercer Generating Station
          print(plant_name)
          imageName = plant_id.replace('/', '-') + "-" + str(plant_name)+ "-" + str(image_type)
          stored.append((plant_id, imageName))
          allCoords = p.get_coord(plant_id)
```

The first block (10) is for importing.
The second block (12) shows all ids and names of the exist power plant we have (uncomment the print statement).
The third block (13) starts to download file from google earth engine to google drive, several input parameters can be changed:
“allPlantsGeojson” as the current using Geojson name
“range(60, 65)” as an example trial to download the 60th to 64th power plants in dictionary “dic_idToName”
(13) need some time to run, which can be done in parallel later to improve its performance.

2. Run the following blocks for Functionality 2:



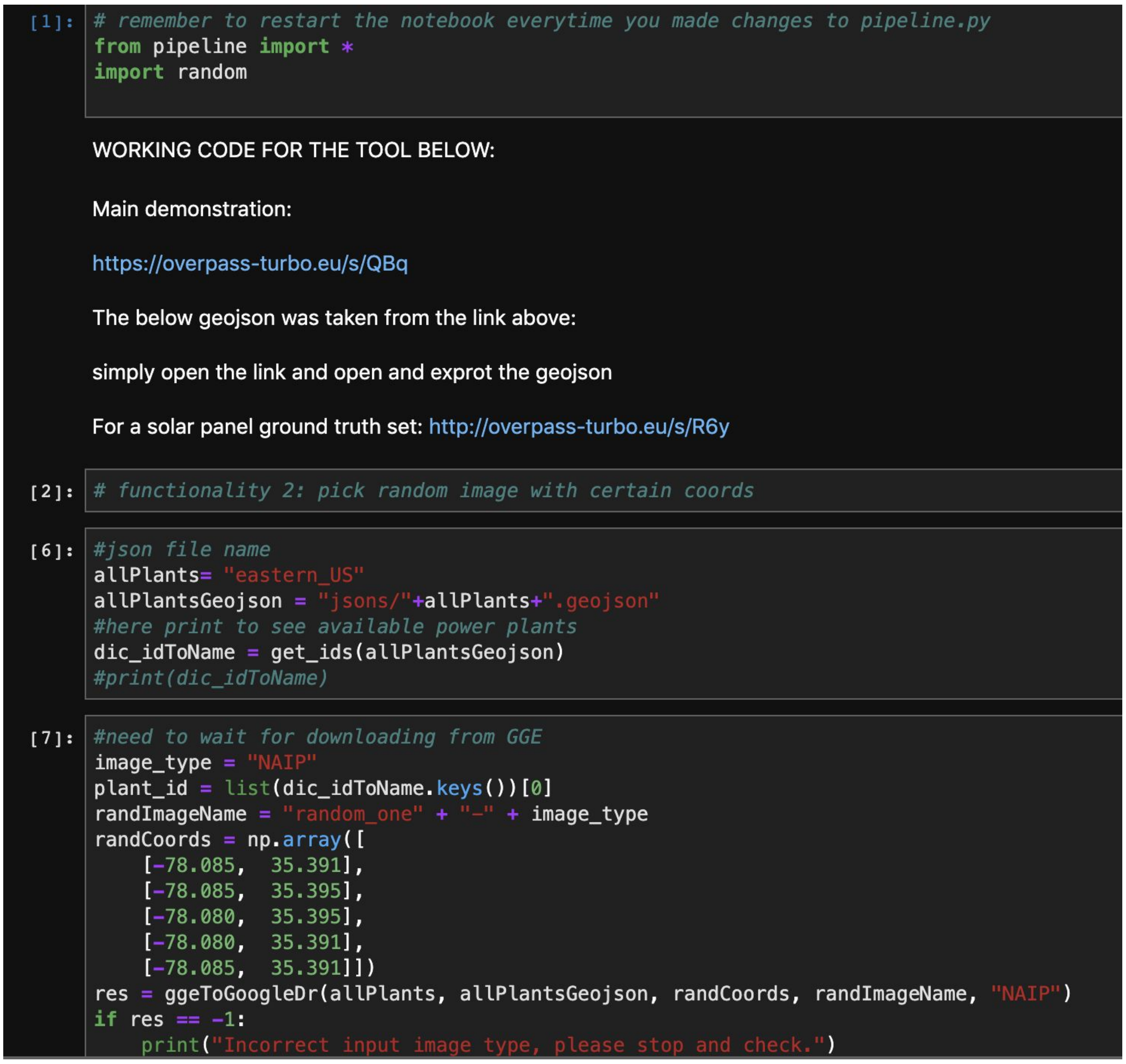
```
[328]: # step 2
       googleDrToLocal(allPlants, 'gs://earth_engine_data', 'gee_images/')

[329]: # step 3
       # this will fetch the image
       # and rasterize the image
       for plant_id, imageName in stored:
           get_rast_image(imageName,'gee_images/earth_engine_data/'+imageName+'.tif', allPlantsGe

/usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning: '+init=<authority
>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization metho
d.
  return _prepare_from_string(" ".join(pjargs))
/usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning: '+init=<authority
>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization metho
d.
  return _prepare_from_string(" ".join(pjargs))
/usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning: '+init=<authority
>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization metho
d.
  return _prepare_from_string(" ".join(pjargs))
/usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning: '+init=<authority
>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization metho
d.
  return _prepare_from_string(" ".join(pjargs))
/usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning: '+init=<authority
>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization metho
d.
  return _prepare_from_string(" ".join(pjargs))

Image processed can be found in images/patches.

Functionality 1 DONE
```

The first block (328) downloads all tif image from google drive to local machine. We can find all tif images downloaded from “gee_images/earth_engine_data”. In this function, the second input is the Google Drive address, and the third is the local address.
The second block (329) rasterize all tif images in the “gee_images/earth_engine_data” (all tif images we downloaded from the previous steps).

3. We can find both our rgb and ground truth (gt) image result in “images/patches/”.

2.3 Functionality 3: pick power plant(s) from input coordinates

1. Open the “NAIP-Coord-Rasterize.ipynb” file, and run the following blocks for Functionality 3:



```
[1]: # remember to restart the notebook everytime you made changes to pipeline.py
     from pipeline import *
     import random
```

WORKING CODE FOR THE TOOL BELOW:

Main demonstration:

https://overpass-turbo.eu/s/QBq

The below geojson was taken from the link above:

simply open the link and open and exprot the geojson

For a solar panel ground truth set: http://overpass-turbo.eu/s/R6y

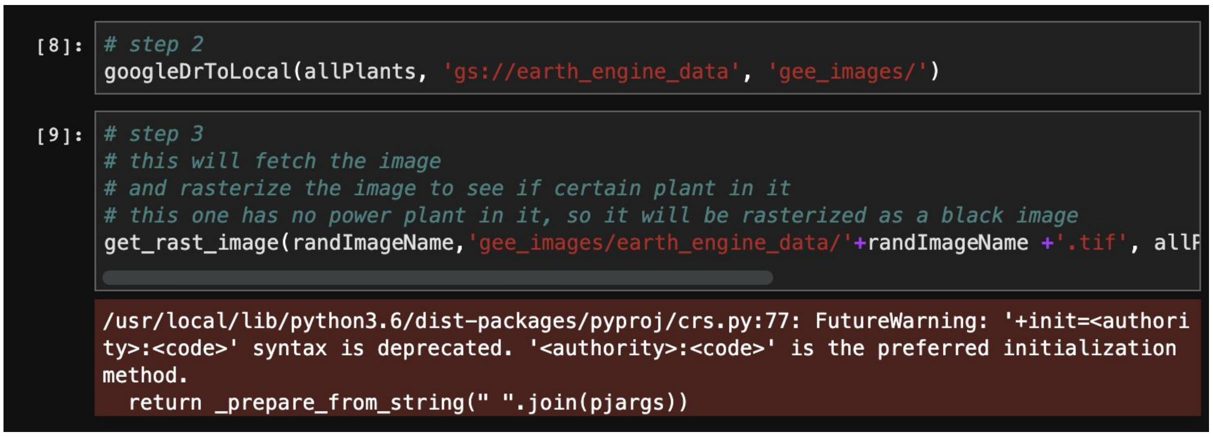
```
[2]: # functionality 2: pick random image with certain coords
```

```
[6]: #json file name
     allPlants= "eastern_US"
     allPlantsGeojson = "jsons/"+allPlants+".geojson"
     #here print to see available power plants
     dic_idToName = get_ids(allPlantsGeojson)
     #print(dic_idToName)
```

```
[7]: #need to wait for downloading from GGE
     image_type = "NAIP"
     plant_id = list(dic_idToName.keys())[0]
     randImageName = "random_one" + "-" + image_type
     randCoords = np.array([
         [-78.085,  35.391],
         [-78.085,  35.395],
         [-78.080,  35.395],
         [-78.080,  35.391],
         [-78.085,  35.391]])
     res = ggeToGoogleDr(allPlants, allPlantsGeojson, randCoords, randImageName, "NAIP")
     if res == -1:
         print("Incorrect input image type, please stop and check.")
```

The first block (10) is for importing.
The second block (12) shows all ids and names of the exist power plant we have (uncomment the print statement).
The third block (13) starts to download file from google earth engine to google drive, several input parameters can be changed:
“allPlantsGeojson” as the current using Geojson name
“range(60, 65)” as an example trial to download the 60th to 64th power plants in dictionary “dic_idToName”
(13) need some time to run, which can be done in parallel later to improve its performance.

2. Run the following blocks for Functionality 2:



```
[8]: # step 2
     googleDrToLocal(allPlants, 'gs://earth_engine_data', 'gee_images/')

[9]: # step 3
     # this will fetch the image
     # and rasterize the image to see if certain plant in it
     # this one has no power plant in it, so it will be rasterized as a black image
     get_rast_image(randImageName,'gee_images/earth_engine_data/'+randImageName +'.tif', allP

     /usr/local/lib/python3.6/dist-packages/pyproj/crs.py:77: FutureWarning: '+init=<authori
     ty>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization
     method.
       return _prepare_from_string(" ".join(pjargs))
```

The first block (328) downloads all tif image from google drive to local machine. We can find all tif images downloaded from “gee_images/earth_engine_data”. In this function, the second input is the Google Drive address, and the third is the local address.
The second block (329) rasterize all tif images in the “gee_images/earth_engine_data” (all tif images we downloaded from the previous steps).

3. We can find both our rgb and ground truth (gt) image result in “images/patches/”.

Limitation and suggest fix:

The “LANDSAT” one generates a blue image now so probably need to be fixed in function “getLandsatTask(coords,name)” in pipeline.py file
We only have eastern US power plants downloaded, download western part if needed.
Efficiency and scalability.