ManageDatasetsMetadata_FirstApproach

February 18, 2021

[1]:
```
%config Completer.use_jedi = False
```

1 Test the proposed metadata approach

We want to have metadata expressed in text files in s3 alongside the actual datasets. In the proposed schema the yaml files will contain all the necessary information about the datasources, including its geographical region. https://gist.github.com/cronosnull/a179f1bba556e724ce58cb50d7f02b86 There are two variants of metadata-external.yml, one with independent documents for each dataset and other with a list of datasets (so you can use YAML anchoring to reuse blocks from previous documents in the same file).

We can use this script to find the metadata files, map it into a dataframe and finaly convert it into features in the arcgis online instance (so AGOL will be the portal to find the datasets based on geographical or metadata based queries).

1.0.1 sample yaml file

```
---

global_attributes:
  title: CEOS Data Cube Landsat Surface Reflectance
  summary: Landsat 8 Operational Land Imager ARD prepared by NASA on behalf of CEOS.
  source: LaSRC surface reflectance product prepared using USGS Collection 1 data.
  institution: CEOS
  instrument: OLI_TIRS
  cdm_data_type: Grid
  keywords: AU/GA,NASA/GSFC/SED/ESD/LANDSAT,REFLECTANCE,ETM+,TM,OLI,EARTH SCIENCE
  keywords_vocabulary: GCMD
  platform: LANDSAT_8
  processing_level: L2
  product_version: '2.0.0'
  product_suite: USGS Landsat Collection 1
  project: CEOS
  coverage_content_type: physicalMeasurement
  references: http://dx.doi.org/10.3334/ORNLDAAC/1146
  license: https://creativecommons.org/licenses/by/4.0/
  naming_authority: gov.usgs
  acknowledgment: Landsat data is provided by the United States Geological Survey (USGS).
```

```
  topic: Imagery and base maps
  start_date: 20210106
  end_date: 20210106
  creation_date: 20210111
  published_date: 20210112
  crs: "EPSG:4326"
bounds_wkt: MultiPolygon (((-99.01244262644466687 29.68847714169181984, -99.01260000000000616 2
index_location: s3://landsat-pds/c1/L8/027/039/LO08_L1TP_027039_20201106_20201111_01_T1/index.h
metadata_contacts:
  name: Christian Ariza
  email: carizaporras@NOTSPAMTWI.org
  role: user
measurements:
    - name: coastal_aerosol
      dtype: int16
      nodata: -9999
      resampling_method: nearest
      src_varname: 'sr_band1'
      zlib: True
      attrs:
          long_name: "Surface Reflectance 0.43-0.45 microns (Coastal Aerosol)"
          alias: "band_1"
    - name: blue
      dtype: int16
      nodata: -9999
      resampling_method: nearest
      src_varname: 'sr_band2'
      zlib: True
      attrs:
          long_name: "Surface Reflectance 0.45-0.51 microns (Blue)"
          alias: "band_2"
    - name: green
      dtype: int16
      nodata: -9999
      resampling_method: nearest
      src_varname: 'sr_band3'
      zlib: True
      attrs:
          long_name: "Surface Reflectance 0.53-0.59 microns (Green)"
          alias: "band_4"
    - name: red
      dtype: int16
      nodata: -9999
      resampling_method: nearest
      src_varname: 'sr_band4'
      zlib: True
      attrs:
          long_name: "Surface Reflectance 0.64-0.67 microns (Red)"
```

```
      alias: "band_4"
- name: nir
  dtype: int16
  nodata: -9999
  resampling_method: nearest
  src_varname: 'sr_band5'
  zlib: True
  attrs:
      long_name: "Surface Reflectance 0.85-0.88 microns (Near Infrared)"
      alias: "band_5"
- name: swir1
  dtype: int16
  nodata: -9999
  resampling_method: nearest
  src_varname: 'sr_band6'
  zlib: True
  attrs:
      long_name: "Surface Reflectance 1.57-1.65 microns (Short-wave Infrared)"
      alias: "band_6"
- name: swir2
  dtype: int16
  nodata: -9999
  resampling_method: nearest
  src_varname: 'sr_band7'
  zlib: True
  attrs:
      long_name: "Surface Reflectance 2.11-2.29 microns (Short-wave Infrared)"
      alias: "band_7"
- name: 'pixel_qa'
  dtype: int32
  nodata: 1
  resampling_method: nearest
  src_varname: 'pixel_qa'
  zlib: True
  attrs:
      long_name: "Pixel Quality Attributes Bit Index"
      alias: [pixel_qa]
- name: 'aerosol_qa'
  dtype: uint8
  nodata: 0
  resampling_method: nearest
  src_varname: 'sr_aerosol'
  zlib: True
  attrs:
      long_name: "Aerosol Quality Attributes Bit Index"
      alias: [sr_aerosol_qa, sr_aerosol]
- name: 'radsat_qa'
  dtype: int32
```

```
      nodata: 1
      resampling_method: nearest
      src_varname: 'radsat_qa'
      zlib: True
      attrs:
          long_name: "Radiometric Saturation Quality Attributes Bit Index"
          alias: [radsat_qa]
    - name: 'solar_azimuth'
      dtype: int16
      nodata: -32768
      resampling_method: nearest
      src_varname: 'solar_azimuth_band4'
      zlib: True
      attrs:
          long_name: "Solar Azimuth Angle for Band 4"
          alias: [solar_azimuth_band4]
    - name: 'solar_zenith'
      dtype: int16
      nodata: -32768
      resampling_method: nearest
      src_varname: 'solar_zenith_band4'
      zlib: True
      attrs:
          long_name: "Solar Zenith Angle for Band 4"
          alias: [solar_zenith_band4]
    - name: 'sensor_azimuth'
      dtype: int16
      nodata: -32768
      resampling_method: nearest
      src_varname: 'sensor_azimuth_band4'
      zlib: True
      attrs:
          long_name: "Sensor Azimuth Angle for Band 4"
          alias: [sensor_azimuth_band4]
    - name: 'sensor_zenith'
      dtype: int16
      nodata: -32768
      resampling_method: nearest
      src_varname: 'sensor_zenith_band4'
      zlib: True
      attrs:
          long_name: "Sensor Zenith Angle for Band 4"
          alias: [sensor_zenith_band4]
lineage:
  description: ""
  source: "<URI for the source file>"
  process: "LaSRC"
```

[2]:
```
#PARAMETERS
project = "TEST" #Either GLO, LWI, OR TEST
process_all = True
buckets = {"GLO": ["glo-data"], "LWI":["lwi-common", *[f"lwi-region{x}" for x␣
↪in range(1,8)]], "TEST": ["test-bucket"]}
```

[3]:
```
import json
import glob
import yaml
import datetime
import boto3  # aws client api
from botocore.client import Config
import requests
import pandas as pd
from urllib.parse import urlparse
from hashlib import sha3_256

# to work with maps:
from shapely import wkt, ops
import geopandas
import contextily as ctx

try:
    from yaml import CLoader as Loader, CDumper as Dumper
except ImportError:
    from yaml import Loader, Dumper
boto3.setup_default_session(profile_name=project)
hucs_gdf = geopandas.read_file(r"./glo_eastregion_huc08.gdb")
```

[4]:
```
def huc2geometry(x):
    """
    If the code is huc_8 returns the geometry from the hucs_gdf.
    If it is a huc 4 or 6, it returns the union of the geometries of the␣
↪matching rows.
    params:
        x: a string or an number representing the huc.
    """
    if pd.isna(x):
        return None
    huc_s = str(int(x))
    huc_type = len(huc_s)
    if huc_type == 8:
        return hucs_gdf.loc[hucs_gdf.HUC_8 == huc_s].geometry.values[0]
    else:
        values = hucs_gdf.loc[hucs_gdf[f"HUC_{huc_type}"] == huc_s].geometry.
↪values
        return ops.unary_union(values)
```

```
def hash_doc(doc):
    return sha3_256(json.dumps(doc, default=str).encode("utf-8")).hexdigest()
```

2 Retrieve the metadata from the s3 buckets

We should query the buckets for the metadata.yaml or metadata-external.yaml files (or other patterns)

[5]:
```
def get_metadata_from_buckets(project, process_all=True):
    """
    get the data from the buckets of the project
    """
    kargs = {}
    if project == "TEST":
        kargs["endpoint_url"] = "http://localhost:9000"
        kargs["config"] = Config(signature_version="s3v4")
    already_seen = []
    if not process_all:
        hash_files = glob.glob("processed_hash_*")
        if hash_files:
            _pdf = pd.concat(map(pd.read_csv, hash_files))
            already_seen = _pdf.metadata_hash.to_list()
    s3 = boto3.resource("s3", **kargs)
    p_buckets = buckets[project]
    for b in p_buckets:
        s3_bucket = s3.Bucket(b)
        metadata_files = [
            f"{x.key}"
            for x in s3_bucket.objects.all()
            if any(
                map(
                    x.key.__contains__,
                    [
                        "metadata.yml",
                        "metadata-external.yml",
                        "metadata.yaml",
                        "metadata-external.yaml",
                    ],
                )
            )
        ]
        metadata_list = []
        for mfile in metadata_files:
            obj = s3_bucket.Object(mfile).get()
```

```
            content = obj["Body"].read()
            metadata_gen = yaml.load_all(content, Loader=Loader)
            folder = f"s3://{s3_bucket.name}/{mfile[:mfile.rfind('/')+1]}"
            try:
                for document in metadata_gen:
                    if isinstance(document, list):
                        for d in document:
                            d["metadata_hash"] = hash_doc(d)
                            d["metadata_folder"] = folder
                        metadata_list.extend(document)
                    else:
                        document["metadata_hash"] = hash_doc(document)
                        document["metadata_folder"] = folder
                        metadata_list.append(document)
            except yaml.error.YAMLError:
                print(
                    f"Invalid syntax on s3://{s3_bucket.name}{mfile}, the␣
↪dataset(s) defined on it will be ignored"
                )
                continue
        metadata_df = pd.json_normalize(metadata_list)
        metadata_df.drop(
            metadata_df[metadata_df.metadata_hash.isin(already_seen)].index,
            inplace=True,
        )
        # If all the datasets has been already seen, exit.
        return metadata_df
```

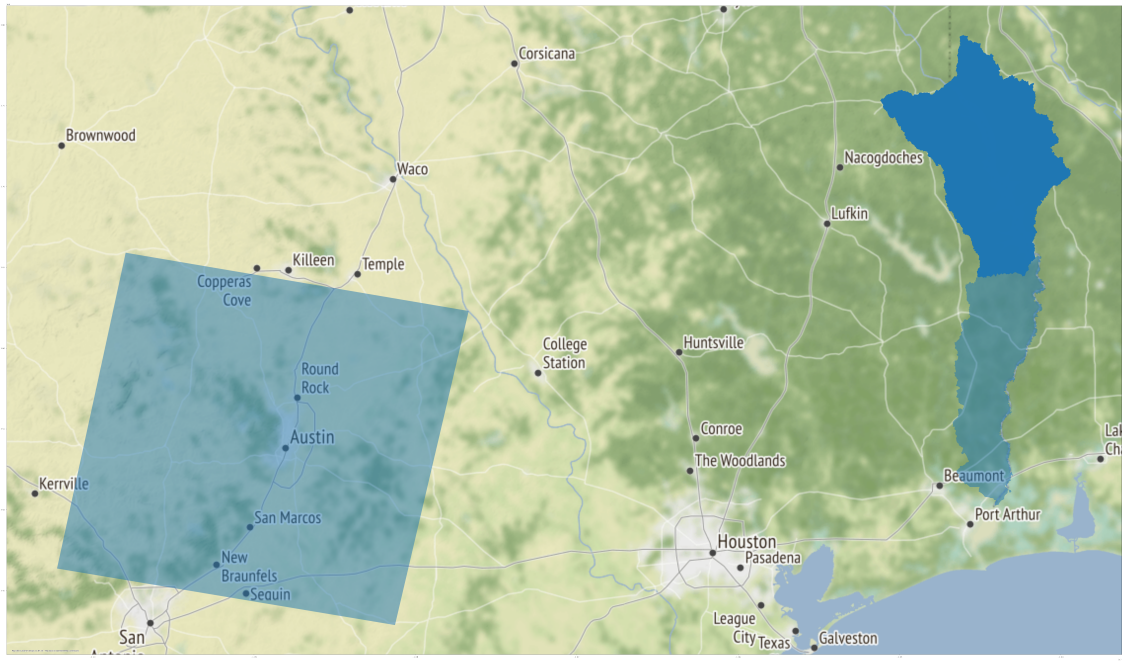
[6]:
```
def generate_geodataframe(project, process_all=True):
    """
    Generate a geodataframe for all the buckets of a given project
    """
    metadata_df = get_metadata_from_buckets(project, process_all)
    metadata_df["limits"] = metadata_df.bounds_wkt.apply(
        lambda x: wkt.loads(x) if pd.notnull(x) else None
    )
    if "bounds_huc" in metadata_df:
        metadata_df["limits"] = metadata_df.apply(
            lambda row: huc2geometry(row.bounds_huc)
            if pd.notnull(row.bounds_huc)
            else row.limits,
            axis=1,
        )
    metadata_df.index_location = metadata_df.apply(
        lambda x: x.index_location
        if bool(urlparse(x.index_location).netloc)
        else f"{x.metadata_folder}{x.index_location}",
```

```
        axis=1,
    )
    metadata_df["name"] = metadata_df["global_attributes.title"]
    metadata_gdf = geopandas.GeoDataFrame(metadata_df, geometry=metadata_df.
↪limits)
    metadata_gdf = metadata_gdf.drop(
        columns=["limits", "bounds_wkt"], errors="ignore"
    )  # ignore if they don't exists
    metadata = metadata_gdf.set_crs(crs="EPSG:4326")
    if "measurements" in metadata_gdf:
        metadata_gdf["measurements"] = metadata_gdf["measurements"].apply(
            lambda x: json.dumps(x, default=str) if x is not None else None
        )
    for col in metadata_gdf:
        metadata_gdf[col] = metadata_gdf[col].apply(
            lambda x: x if not isinstance(x, list) else json.dumps(x,␣
↪default=str)
        )
    return metadata_gdf
```

[7]:
```
#Main process:
metadata_gdf = generate_geodataframe(project, process_all)
if not metadata_gdf.empty:
    # Save the gdf to file
    metadata_gdf.to_file("datasources.json", driver="GeoJSON")
    # Create the checkpoint file (list of hashes)
    metadata_gdf[["metadata_hash"]].to_csv(
        f"processed_hash_{datetime.datetime.utcnow().timestamp()}", index=False
    )
```

3 Plot the datasets:

[8]:
```
#all the rows should have the same projection, so this change should be either␣
↪enforced by policy or made beforehand
tmp = metadata_gdf.set_crs(crs="EPSG:4326")
#to match with the ctx map, we should use web mercator projection:
ax = tmp.to_crs(epsg=3857).plot( figsize=(100, 100), alpha=0.5)
ctx.add_basemap(ax)
```



[9]:
```
metadata_gdf.columns
```

[9]:
```
Index(['index_location', 'metadata_hash', 'metadata_folder',
       'global_attributes.title', 'global_attributes.summary',
       'global_attributes.source', 'global_attributes.institution',
       'global_attributes.cdm_data_type', 'global_attributes.keywords',
       'global_attributes.keywords_vocabulary',
       'global_attributes.product_suite', 'global_attributes.project',
       'global_attributes.coverage_content_type',
       'global_attributes.references', 'global_attributes.license',
       'global_attributes.acknowledgment', 'metadata_contacts.name',
       'metadata_contacts.email', 'metadata_contacts.role',
       'lineage.description', 'lineage.source', 'lineage.source_list',
       'lineage.process', 'measurements', 'global_attributes.crs',
       'global_attributes.instrument', 'global_attributes.platform',
       'global_attributes.processing_level',
       'global_attributes.product_version',
       'global_attributes.naming_authority', 'global_attributes.topic',
       'global_attributes.start_date', 'global_attributes.end_date',
       'global_attributes.creation_date', 'global_attributes.published_date',
       'bounds_huc', 'name', 'geometry'],
      dtype='object')
```

[10]:
```
metadata_gdf.measurements
```

[10]:
```
0                                                    NaN
1    [{"name": "DTL_STUD_AR", "dtype": "dataset", "…
2                                                    NaN
3        [{"name": "BLE_Dep01PCT", "dtype": "double"}]
4       [{"name": "BLE_Dep002PCT", "dtype": "double"}]
5       [{"name": "BLE_WSE002PCT", "dtype": "double"}]
6        [{"name": "BLE_WSE01PCT", "dtype": "double"}]
7    [{"name": "coastal_aerosol", "dtype": "int16",…
8                                                    NaN
Name: measurements, dtype: object
```

[11]:
```
metadata_gdf
```

[11]:
```
                                    index_location  \
0  https://ebfedata.s3.amazonaws.com/12010004_Tol…
1  https://ebfedata.s3.amazonaws.com/12010004_Tol…
2  https://ebfedata.s3.amazonaws.com/12010004_Tol…
3  https://ebfedata.s3.amazonaws.com/12010004_Tol…
4  https://ebfedata.s3.amazonaws.com/12010004_Tol…
5  https://ebfedata.s3.amazonaws.com/12010004_Tol…
6  https://ebfedata.s3.amazonaws.com/12010004_Tol…
7  s3://landsat-pds/c1/L8/027/039/LO08_L1TP_02703…
8  s3://test-bucket/test_downloads/urls_12010005_…

                                     metadata_hash  \
0  4022c1a9c2f9ccddd57aa5371ed5514e363048abf82da5…
1  3f60dcaadd439383dfcc0dfd7e5c9d6754e12fdc4a3753…
2  578bdb2d7c7246a877d945342d6bc9e933ae3c457b8eed…
3  51319fa64d620e028154fb25513857751f39f8b5efda51…
4  a1d37352c57ea0f6a1332d146085511d387b6b54d90cd6…
5  f9297513759c87e34d22918601d4529f7788bc6cc5991c…
6  a4e248c5102d25edc63d739456eed4a5d5a999ea3a7cf3…
7  cdf4ac6a18685da4bfd7a60c353cc3d10effbb0a0bb056…
8  8074d7077cfbb4c64f6572c2fe5592aad6ae7879c1f752…

                    metadata_folder  \
0             s3://test-bucket/ble/
1             s3://test-bucket/ble/
2             s3://test-bucket/ble/
3             s3://test-bucket/ble/
4             s3://test-bucket/ble/
5             s3://test-bucket/ble/
6             s3://test-bucket/ble/
7    s3://test-bucket/landsattexas/
8  s3://test-bucket/test_downloads/

                           global_attributes.title  \
```

```
0  Toledo Bend Watershed, LA/TX Base Level Engine…
1  Toledo Bend Watershed, LA/TX Base Level Engine…
2  Toledo Bend Watershed, LA/TX Base Level Engine…
3  Toledo Bend Watershed, LA/TX Base Level Engine…
4  Toledo Bend Watershed, LA/TX Base Level Engine…
5  Toledo Bend Watershed, LA/TX Base Level Engine…
6  Toledo Bend Watershed, LA/TX Base Level Engine…
7         CEOS Data Cube Landsat Surface Reflectance
8                             Lidar for Huc 12010005

                           global_attributes.summary  \
0  FEMA Region VI contracted Compass to complete …
1  FEMA Region VI contracted Compass to complete …
2     A folder containing HECRAS models for streams.
3  A raster representing the estimated depth of f…
4  A raster representing the estimated depth of f…
5  A raster representing the estimated elevation …
6  A raster representing the estimated elevation …
7  Landsat 8 Operational Land Imager ARD prepared…
8                                             Lidar.

                            global_attributes.source  \
0                  BLE prepared by Compass for FEMA.
1                  BLE prepared by Compass for FEMA.
2                  BLE prepared by Compass for FEMA.
3                  BLE prepared by Compass for FEMA.
4                  BLE prepared by Compass for FEMA.
5                  BLE prepared by Compass for FEMA.
6                  BLE prepared by Compass for FEMA.
7  LaSRC surface reflectance product prepared usi…
8                                             tinris

                     global_attributes.institution  \
0  DHS/FEMA (Federal Emergency Management Agency)
1  DHS/FEMA (Federal Emergency Management Agency)
2  DHS/FEMA (Federal Emergency Management Agency)
3  DHS/FEMA (Federal Emergency Management Agency)
4  DHS/FEMA (Federal Emergency Management Agency)
5  DHS/FEMA (Federal Emergency Management Agency)
6  DHS/FEMA (Federal Emergency Management Agency)
7                                           CEOS
8                                           CEOS

  global_attributes.cdm_data_type  \
0          DATA COLLECTION BUNDLE
1                          Vector
2          DATA COLLECTION BUNDLE
```

```
3                            Grid
4                            Grid
5                            Grid
6                            Grid
7                            Grid
8                            Grid

                          global_attributes.keywords  \
0  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
1  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
2  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
3  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
4  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
5  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
6  EARTH SCIENCE SERVICES,"ENVIRONMENTAL ADVISORI…
7  AU/GA,NASA/GSFC/SED/ESD/LANDSAT,REFLECTANCE,ET…
8  AU/GA,NASA/GSFC/SED/ESD/LANDSAT,REFLECTANCE,ET…

  global_attributes.keywords_vocabulary  …  \
0                                  GCMD  …
1                                  GCMD  …
2                                  GCMD  …
3                                  GCMD  …
4                                  GCMD  …
5                                  GCMD  …
6                                  GCMD  …
7                                  GCMD  …
8                                  GCMD  …

  global_attributes.product_version global_attributes.naming_authority  \
0                               NaN                                NaN
1                               NaN                                NaN
2                               NaN                                NaN
3                               NaN                                NaN
4                               NaN                                NaN
5                               NaN                                NaN
6                               NaN                                NaN
7                             2.0.0                           gov.usgs
8                               NaN                           gov.usgs

  global_attributes.topic global_attributes.start_date  \
0                     NaN                          NaN
1                     NaN                          NaN
2                     NaN                          NaN
3                     NaN                          NaN
4                     NaN                          NaN
5                     NaN                          NaN
```

```
6                     NaN                           NaN
7   Imagery and base maps                  20210106.0
8                     NaN                           NaN

  global_attributes.end_date global_attributes.creation_date  \
0                        NaN                             NaN
1                        NaN                             NaN
2                        NaN                             NaN
3                        NaN                             NaN
4                        NaN                             NaN
5                        NaN                             NaN
6                        NaN                             NaN
7                20210106.0                     20210111.0
8                        NaN                             NaN

  global_attributes.published_date  bounds_huc  \
0                              NaN         NaN
1                              NaN         NaN
2                              NaN         NaN
3                              NaN         NaN
4                              NaN         NaN
5                              NaN         NaN
6                              NaN         NaN
7                      20210112.0         NaN
8                              NaN  12010005.0

                                              name  \
0  Toledo Bend Watershed, LA/TX Base Level Engine…
1  Toledo Bend Watershed, LA/TX Base Level Engine…
2  Toledo Bend Watershed, LA/TX Base Level Engine…
3  Toledo Bend Watershed, LA/TX Base Level Engine…
4  Toledo Bend Watershed, LA/TX Base Level Engine…
5  Toledo Bend Watershed, LA/TX Base Level Engine…
6  Toledo Bend Watershed, LA/TX Base Level Engine…
7         CEOS Data Cube Landsat Surface Reflectance
8                            Lidar for Huc 12010005

                                           geometry
0  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
1  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
2  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
3  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
4  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
5  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
6  MULTIPOLYGON (((-93.96285 32.22747, -93.96155 …
7  MULTIPOLYGON (((-99.01244 29.68848, -99.01260 …
8  MULTIPOLYGON (((-93.78500 29.99408, -93.78491 …
```

```
[9 rows x 38 columns]
```

4 ARCGIS TEST

Publish the data to arcgis online (it requires a valid user with publish permissions)

[12]:
```
from arcgis import GeoAccessor, GIS
```

[13]:
```
arcgis_ga = GeoAccessor.from_geodataframe(metadata_gdf, column_name="geometry")
```

[ ]:
```
gis = GIS("https://twiotg.maps.arcgis.com", "<username>", "<password>")
arcgis_ga.spatial.to_featurelayer(f'datasets_{project}',tags=[project,␣
↪"datasets", "metadata"], folder=project, gis=gis)
```

[ ]:
```

```