# process_patches_ki2

November 3, 2020

In [1]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

In [2]:
```
# Built-in modules
import pickle
import sys
import os

import datetime
import itertools
from enum import Enum

# Basics of Python data handling and visualization
import numpy as np
import geopandas as gpd
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from mpl_toolkits.axes_grid1 import make_axes_locatable
from shapely.geometry import Polygon
from tqdm import tqdm_notebook as tqdm

# Machine learning
import lightgbm as lgb
from sklearn.externals import joblib
from sklearn import metrics
from sklearn import preprocessing

# Imports from eo-learn and sentinelhub-py
from eolearn.core import EOTask, EOPatch, LinearWorkflow, Dependency,FeatureType, Overwr
    LoadFromDisk, SaveToDisk, EOExecutor
from eolearn.io import S2L1CWCSInput, ExportToTiff, S2L2AWCSInput, AddSen2CorClassificat
from eolearn.mask import AddCloudMaskTask, get_s2_pixel_cloud_detector, AddValidDataMask
from eolearn.geometry import VectorToRaster, PointSamplingTask, ErosionTask
from eolearn.features import LinearInterpolation, SimpleFilterTask
from sentinelhub import BBoxSplitter, MimeType, BBox, CRS, CustomUrlParam
from pathlib import Path
```

In [3]:
```
slo = gpd.read_file('E:/2015/V4-1811/slo_UTM.shp')
slo_crs = CRS.UTM_33N
print(slo.bounds)
slo_shape = list(slo.geometry.values)[-1]
bbox_splitter_large = BBoxSplitter([slo_shape], slo_crs, (25, 17))
bbox_splitter=bbox_splitter_large

bbox_list = np.array(bbox_splitter.get_bbox_list())
info_list = np.array(bbox_splitter.get_info_list())

patchIDs = []
for idx, [bbox, info] in enumerate(zip(bbox_list, info_list)):
    patchIDs.append(idx)
list(patchIDs)
```

```
           minx          miny           maxx          maxy
0  374874.850567  5.029783e+06  622569.653389  5.192195e+06
```

Out[3]:
```
[0,
 1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 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,
35,
36,
37,
38,
39,
40,
41,
42,
43,
44,
45,
46,
47,
48,
49,
50,
51,
52,
53,
54,
55,
56,
57,
58,
59,
60,
61,
62,
63,
64,
65,
66,
67,
68,
69,
70,
71,
72,
73,
74,
75,
76,

77,
78,
79,
80,
81,
82,
83,
84,
85,
86,
87,
88,
89,
90,
91,
92,
93,
94,
95,
96,
97,
98,
99,
100,
101,
102,
103,
104,
105,
106,
107,
108,
109,
110,
111,
112,
113,
114,
115,
116,
117,
118,
119,
120,
121,
122,
123,
124,

125,
126,
127,
128,
129,
130,
131,
132,
133,
134,
135,
136,
137,
138,
139,
140,
141,
142,
143,
144,
145,
146,
147,
148,
149,
150,
151,
152,
153,
154,
155,
156,
157,
158,
159,
160,
161,
162,
163,
164,
165,
166,
167,
168,
169,
170,
171,
172,

173,
174,
175,
176,
177,
178,
179,
180,
181,
182,
183,
184,
185,
186,
187,
188,
189,
190,
191,
192,
193,
194,
195,
196,
197,
198,
199,
200,
201,
202,
203,
204,
205,
206,
207,
208,
209,
210,
211,
212,
213,
214,
215,
216,
217,
218,
219,
220,

```
        221,
        222,
        223,
        224,
        225,
        226,
        227,
        228,
        229,
        230,
        231,
        232,
        233,
        234,
        235,
        236,
        237,
        238,
        239,
        240,
        241,
        242,
        243,
        244,
        245,
        246,
        247,
        248,
        249,
        250,
        251,
        252,
        253,
        254,
        255,
        256,
        257,
        258,
        259,
        260,
        261,
        262,
        263,
        264,
        265]
```

```
In [4]: print(patchIDs)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 2
```

```
In [5]: geometry = [Polygon(bbox.get_polygon()) for bbox in bbox_list[patchIDs]]
        idxs_x = [info['index_x'] for info in info_list[patchIDs]]
        idxs_y = [info['index_y'] for info in info_list[patchIDs]]

        gdf = gpd.GeoDataFrame({'index_x': idxs_x, 'index_y': idxs_y}, geometry=geometry)

In [6]: shapefile_name = 'E:/2019/slobox.shp'
        gdf.to_file(shapefile_name)

In [7]: import sys
        import os
        sys.path.append(os.path.abspath('H:/2020/'))

In [8]: INSTANCE_ID='e50ce20d-d6ba-41b4-a4f1-67044a7ccfde'
```

### 0.1 Custom tasks

```
In [9]: class Sen2CorValidData:
            """
            Combine Sen2Cor's classification map with `IS_DATA` to define a `VALID_DATA_S2C` mas

            The Sen2Cor's classification map is asumed to be found in eopatch.mask['SCL']
            """
            def __call__(self, eopatch):
                sen2cor_valid = np.zeros_like(eopatch.mask['SCL'], dtype=np.bool)

                valid_classes = [2, 4, 5, 6, 7, 11]
                for valid in valid_classes:
                    sen2cor_valid = np.logical_or(sen2cor_valid, (eopatch.mask['SCL']==valid).as

                return np.logical_and(eopatch.mask['IS_DATA'].astype(np.bool), sen2cor_valid)


        class CountValid(EOTask):
            """
            The task counts number of valid observations in time-series and stores the results i
            """
            def __init__(self, count_what, feature_name):
                self.what = count_what
                self.name = feature_name

            def execute(self, eopatch):
                eopatch.add_feature(FeatureType.MASK_TIMELESS, self.name, np.count_nonzero(eopat

                return eopatch
```

In [10]:
```
# Missing tasks
class ValidDataFractionPredicate:
    """ Predicate that defines if a frame from EOPatch's time-series is valid or not. F
    valid data fraction is above the specified threshold.
    """
    def __init__(self, threshold):
        self.threshold = threshold

    def __call__(self, array):
        coverage = np.sum(array.astype(np.uint8)) / np.prod(array.shape)
        return coverage > self.threshold
```

In [11]:
```
# TASK FOR BAND DATA
#                    0        1      2       3         4              5
# add a request for B(B02), G(B03), R(B04), NIR (B08), SWIR1(B11), SWIR2(B12)
# from default layer 'ALL_BANDS' at 10m resolution
# Here we also do a simple filter of cloudy scenes. A detailed cloud cover
# detection is performed in the next step
custom_script = 'return [B02, B03, B04, B05, B06, B07, B08, B8A];'
add_data = S2L2AWCSInput(
    layer='BANDS-S2-L2A',
    feature=(FeatureType.DATA, 'BANDS'), # save under name 'BANDS'
    custom_url_params={CustomUrlParam.EVALSCRIPT: custom_script}, # custom url for 6 sp
    resx='10m', # resolution x
    resy='10m', # resolution y
    maxcc=0.8, # maximum allowed cloud cover of original ESA tiles
    instance_id=INSTANCE_ID,
)

# 2. Add Sen2Cor's scene classification map
add_SCL = AddSen2CorClassificationFeature(sen2cor_classification='SCL',layer='BANDS-S2-
add_CLD = AddSen2CorClassificationFeature(sen2cor_classification='CLD',layer='BANDS-S2-

# TASK FOR VALID MASK
# validate pixels using SentinelHub's cloud detection mask and region of acquisition
add_s2c_valmask = AddValidDataMaskTask(Sen2CorValidData(), 'VALID_DATA_S2C')
valid_data_predicate = ValidDataFractionPredicate(0.85)
filter_task = SimpleFilterTask((FeatureType.MASK, 'VALID_DATA_S2C'), valid_data_predica


# TASK FOR SAVING TO OUTPUT (if needed)
path_out = Path('H:/2020/')
if not os.path.isdir(path_out):
    os.makedirs(path_out)
save = SaveToDisk(str(path_out), overwrite_permission=OverwritePermission.OVERWRITE_PAT
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\eolearn\core\core_tasks.
  warnings.warn('This task is deprecated, use SaveTask instead', DeprecationWarning)
```

```
In [12]: class AddVectorFeature(EOTask):
             """
             Add vector data from dataframe.
             """
             def __init__(self, feature):
                 self.feature_type, self.feature_name = next(self._parse_features(feature)())

             def execute(self, eopatch, dataframe):
                 eopatch[self.feature_type][self.feature_name] = dataframe[
                     dataframe.geometry.intersects(eopatch.bbox.geometry)]

                 return eopatch

In [13]: subv2020 = gpd.read_file ('E:2015/v4-1811/2020/ARSKTRP/Prijava2020_UTM.shp')

In [14]: add_subvencije = AddVectorFeature(feature=(FeatureType.VECTOR_TIMELESS, 'subvencije2020
         add_subv2020= VectorToRaster( subv2020, (FeatureType.MASK_TIMELESS,'subv2020'), values=
                                       raster_shape=((FeatureType.MASK, 'VALID_DATA_S2C')))

In [15]: workflow = LinearWorkflow(
             add_data,
             add_SCL,
             add_CLD,
             add_s2c_valmask,
             filter_task,
             add_subvencije,
             add_subv2020,
             save
         )

In [16]: import re
         from eolearn.core import EOPatch
         pat = re.compile(r"eopatch_inter_(?P<patch_id>\d*)")
         taken_ids = set()
         files = sorted(os.listdir(path_out))
         for ind, filename in enumerate(files):
             m = pat.match(filename)
             print(filename, ind, "/", len(files))
             if not m:
                 print("invalid file name")
                 continue
             try:
                 EOPatch.load(os.path.join(path_out, filename))
                 taken_ids.add(int(m["patch_id"]))
             except KeyboardInterrupt:
                 break
             except Exception as e:
                 print("Invalid", filename, e)
                 break
```

```
        to_take = [j for j in patchIDs if j not in taken_ids]
```

```
eopatch_inter_0 0 / 15
eopatch_inter_1 1 / 15
eopatch_inter_10 2 / 15
eopatch_inter_11 3 / 15
eopatch_inter_12 4 / 15
eopatch_inter_13 5 / 15
eopatch_inter_14 6 / 15
eopatch_inter_2 7 / 15
eopatch_inter_3 8 / 15
eopatch_inter_4 9 / 15
eopatch_inter_5 10 / 15
eopatch_inter_6 11 / 15
eopatch_inter_7 12 / 15
eopatch_inter_8 13 / 15
eopatch_inter_9 14 / 15
```

In [17]:
```
print(to_take)
```

```
[15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
```

In [18]:
```
%%time

        # Execute the workflow
        time_interval = ['2020-01-01', '2020-08-31'] # time interval for the SH request

        # define additional parameters of the workflow
        execution_args = []
        for  bbox_ind in to_take:

            execution_args.append({
                add_data:{'bbox': bbox_list[bbox_ind], 'time_interval': time_interval},
                add_subvencije: {'dataframe': subv2020},
                save: {'eopatch_folder': 'eopatch_inter_{}'.format(bbox_ind)}
            })

        executor = EOExecutor(workflow, execution_args, save_logs=True)
        executor.run(workers=5, multiprocess=False)

        executor.make_report()
```

```
HBox(children=(IntProgress(value=0, max=251), HTML(value='')))
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\sentinelhub\data_request
  category=SHDeprecationWarning)
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\eolearn\geometry\transfo
  warnings.warn('Given vector polygons contain some invalid geometries, they will be fixed', Run
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\eolearn\core\eoexecution
Please install the system package 'graphviz' (in addition to the python package) to have the dep
  return EOExecutorVisualization(self).make_report()
```

```
Wall time: 11h 27min 33s
```

In [ ]: