Python Developer Entry Task

Sinergise

The Earth Observation (EO) research team at Sinergise is working on satellite image classification. They are developing machine learning algorithms which can classify each pixel of the image into one of the land cover classes (i.e. cultivated land, forest, artificial surface, water, etc.).

The team has put together a small dataset of ground truth reference data. They are asking you to write a Python program which reads the dataset, uses Sentinel Hub services to download satellite data and performs some analysis of the data. The program can use imports from any external Python package available out there.

Instructions are divided into multiple parts. They also contain some hints to help you work with geographical data.

a) The reference dataset, `land_polygons.shp`, is in ESRI Shapefile format [1]. First, your program has to read the data from the file. In the dataset there is a list 10000 of polygons, each of them representing a land parcel somewhere in Slovenia. Besides the geometry each polygon will have an *ID* and a *LAND_TYPE* attribute. The LAND_TYPE attribute specifies the land cover class to which a polygon belongs to.
`Hint 1:` Using `geopandas` Python package might be a good choice for this.
`Hint 2:` To better understand the data it is a good idea to visualize it. You can do that by opening the shapefile in QGIS application [2].

b) Next your program should collect satellite image data for the region where polygons are located. You will be working with Sentinel-2 satellite [3] and its standard L1C products. The data should be downloaded from Sentinel Hub Process API [4]. Fortunately, we have `sentinelhub-py` Python package which is an interface for the service. By exploring its documentation examples [5] you should be able to understand how to obtain satellite images. In the process you will have to create a trial Sentinel Hub account which will be valid for a month. If your trial period ends before you finish the task, please ask us to extend it.

Once you understand how to get the data, here is what you need:

- You should request a single image of a region which contains all polygons from a).
- For this task you only need *normalized difference vegetation index* (NDVI, [7]) values. For each pixel of the image NDVI value is defined with formula

$$NDVI = \frac{B08 - B04}{B08 + B04}$$

where B04 and B08 are values of red and near infrared Sentinel-2 bands of the pixel. You can use the following evalscript:

```
// VERSION=3
function setup() {
  return {
    input: [
      {
        bands: ["B04", "B08"],
        units: ["REFLECTANCE", "REFLECTANCE"]
      }
    ],
    output: {
      bands: 1,
      sampleType: "FLOAT32"
    }
  }
}
```

```
function evaluatePixel(sample) {
  let ndvi = index(sample.B08, sample.B04);
  return [ndvi];
}
```

- The requested image should be from 2018-09-28. On that day the sky over Slovenia was cloudless.
- To get the most accurate information back, the requested image should be at 10 meter resolution.
- The coordinate reference system (CRS) of your polygons is UTM 33 north (EPSG:32633) [8]. Therefore the requested image should also be in the same CRS (i.e. `CRS.UTM_33N`).
- To obtain correct values make sure to request data in `MimeType.TIFF` format.
- To avoid re-downloading data each time your program is executed, you can specify `data_folder` parameter and achieve that downloaded data is being cached locally.

At the end of this step you should have a 2D `numpy` array containing NDVI values for each pixel of the image.

c) In this step you have to transform the list of polygons, which is currently in vector form, into a raster mask, which should be geographically aligned with the NDVI image. By doing so you will be able to tell which pixels from NDVI image belong to which polygon.
`Hint 1:` For rasterization use `rasterio.features.rasterize` function from `rasterio` Python package. For the function's parameter `transform` use an instance of `affine.Affine` class which is returned by `rasterio.transform.from_bounds` function.
`Hint 2:` This process is also used somewhere in between the lines of `eo-learn` package. You can take a sneak peek into the code: [9].
Note: In the rasterization process a pixel will be mapped to a specific polygon if and only if the majority of the pixel's area is covered by the polygon.

You have reached the stage where you have all the data you need. Now it's time to implement 2 functions required by EO research team.

d) Implement a function `get_statistics(query)`. For parameter `query` the function should accept either an integer, which should be an ID of one of the polygons, or a string, which should be a name of one of the land cover classes from LAND_TYPE attribute.
The function should return 4 values which are minimum, maximum, mean and standard deviation of NDVI values over pixels belonging either to a polygon with specified ID or to a specified land cover class. The values should be rounded to 6 decimals after the decimal point.

Examples:

```
> get_statistics(42)
(0.588741, 0.794158, 0.719222, 0.062071)
```

The function returned minimum, maximum, mean and standard deviation for NDVI values of all 15 pixels that belong to polygon with ID 42.

```
> get_statistics('forest')
(-0.335347, 0.843855, 0.662613, 0.079952)
```

The function returned statistics for NDVI values of all 209063 pixels that belong to polygons whose LAND_TYPE attribute is "forest".

e) Implement a function `get_closest_pair(id_list, *criteria)`. The parameter `id_list` should receive a list of polygon IDs. The list can contain from 2 and up to all 10000 IDs. The parameter `criteria` should get either one or two out of the following 4 strings: `'min'`, `'max'`, `'mean'` or `'std'`. These strings represent criteria functions minimum, maximum, mean and standard deviation.

The function should return 2 IDs from the given list for which the 2 polygons are most similar according to the given criteria.

- If a single criteria function is given, we are looking for those IDs i, j for which polygons P_i, P_j minimize

$$|f(P_i) - f(P_j)|,$$

where f represents the given criteria function over NDVI values of polygon's pixels.

- If 2 criteria functions are given, we are looking for those IDs i, j for which polygons P_i, P_j minimize

$$\sqrt{(f(P_i) - f(P_j))^2 + (g(P_i) - g(P_j))^2},$$

where f and g represent given criteria functions over NDVI values of polygon's pixels.

Examples:

```
> get_closest_pair([2, 3, 5, 7, 11], 'min')
(3, 5)
```

Difference between minimums over NDVI values for polygons with ID 3 and 5 is only about 0.0178. That is the least over all pairs of polygons with IDs from the given list.

```
> complete_id_list = list(range(10 ** 4))
> get_closest_pair(complete_id_list, 'mean', 'std')
(167, 4466)
```

Out of all polygons, those with ID 167 and 4466 have the most similar mean and standard deviation of their NDVI values.

f) Bonus subtask: Try to optimize the performance of both functions and the program in general. You can make it run in under a minute for any test case. Optimizing `get_closest_pair` for large ID lists and 2 criteria functions might require some more work.

We hope you will enjoy solving this task and in the process learn some cool new stuff. The task will test your proficiency in Python, familiarity with commonly used libraries (e.g. `numpy`), adaptiveness to unknown tools, algorithmic skills and code style. At the same time it should give you some insight into day to day work of a Python developer at Sinergise.

If you have any questions about the task, feel free to send them to EO research team. For questions about Sentinel Hub service and `sentinelhub-py` you can check our forum [10]. You are also allowed to use any other literature and online resources.

If you successfully solve the entire task, you are probably a good fit for the team. If not, the team would still like to check your progress and decide to invite you to the interview.

At the end put your solution in a ZIP file and send it back to your Sinergise contact.

Good luck!

References

[1] https://en.wikipedia.org/wiki/Shapefile
[2] https://qgis.org/en/site/forusers/download.html
[3] https://en.wikipedia.org/wiki/Sentinel-2
[4] https://docs.sentinel-hub.com/api/latest/api/process
[5] https://sentinelhub-py.readthedocs.io/en/latest/examples/process_request.html
[6] https://github.com/sentinel-hub/sentinelhub-py
[7] https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index
[8] https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
[9] https://github.com/sentinel-hub/eo-learn/blob/v0.5.1/geometry/eolearn/geometry/transformations.py#L212
[10] https://forum.sentinel-hub.com/