Capstone Project

Machine Learning Engineer Nanodegree

Lucas Pontes

May 16,2021

Land User Land Cover Classifier App

Sumário

1. Project Definition

1.1. Overview

Brazil is one of the largest producers of agricultural products in the world and, despite strict environmental legislation, there is much criticism about the preservation of important biomes such as the Amazon Forest. With the availability of high-resolution satellite images, it is possible to verify that rural producers respect the environmental laws for the conservation of natural vegetation.

The main idea of this project is to provide an app that, using machine learning models, allows to classify satellite images according to the main classes of land use and occupation: water, bare soil, forest, annual crop, perennial crop, pasture and urban areas.

One of the main difficulties in working with satellite images on a large scale is the consumption of memory and the disk space needed to store the images, in addition to the traditional process of searching and filtering images being very laborious. However, it is currently possible to use cloud services such as Google Earth Engine (GEE), which has a catalog of satellite products and a platform with geoprocessing tools and algorithms that runs on Google servers. Therefore, the main advantages provided by the GEE is to perform searches in free image catalogs and use Google processing to process and model the data.

With that in mind, I deploy a web application that displays a satellite image from a given date and space selected by user and predicts what type of land user land cover (LULC) each pixel on selected polygon is classified. Finally, plot the results over the polygon image.

The focus of the project is to use the Machine Learning Engineering knowledge acquired in Udacity Nanodegree to deploy the web app. For that I use the Google Earth Engine API to obtain the data and train the model and Heroku service to deliver the application.

1.2. Problem Statement

The aim is to provide a web app where the user can interact to choose the date and draw a polygon over a Sentinel Satellite image (10 m resolution) and to return the predicted LULC classes for that area. The geemap package provides the necessary integration with google Earth Engine API, from which the apps obtains the data. The steps to accomplish it are:

- Collect and tag pixels from Sentinel images accordingly with the main LULC in Brazil
- Transform the collected data to tabular form with the latitude, longitude, class_id, and the float value from each of image bands
- Train a machine learning model to predict LULC class given the image bands values
- Build a web page where users can interact and send the data to model
- Return and plot the results to the user visualize and compare with what his is seeing from the satellite image.

1.3. Metrics

Given that I retrieved balanced number of sample points for each class, accuracy was used as the main metric to evaluate the modeling process. In the other hand it is a multi-class problem, so it is important to analyze the confusion matrix as well to spot if the model is systematically misclassifying some of the classes.

2. Methodology

2.1. Data retrieve and Preprocessing

Google Earth Engineer (https://developers.google.com/earth-engine) is a geospatial processing service that provides a python API that was used to obtain the data used to train and evaluate the model being developed.

To retrieve the data it was used the geemap python package (Wu, 2020). This package allows interactive mapping with Google Earth Engine and made easier to choose the interest areas, draw polygons that possess as property a chosen landcover value.

For each pixel inside each polygon the band values are captured together with the landcover value on a feature collection that is a json file used on Earth Engine API. This data can also be exported as a Pandas dataframe (Table 01).

landcover	label	B1	B2	B3	B4	B5	B6	B7
1	bare_soil	673	877	1244	1830	1930	1986	2057
5	pasture	418	535	820	915	1408	2141	2409
1	bare_soil	904	1144	1562	2164	2301	2374	2419
1	bare_soil	805	1042	1552	2386	2634	2837	2983
0	water	423	426	465	307	290	261	259

Table 01. sample from the dataframe obtained to train machine learning model used to predict landcover classes given the bands (B1, B2, etc) values.

For short the process consists in:

1) Select a region of interest (user_rois), search for a Sentinel image in a given date range and select the one with the less cloud coverage.

```
image = (ee.ImageCollection('COPERNICUS/S2_SR')
         .filterBounds(Map.user_rois.geometry())
         .filterDate('2020-09-01', '2020-12-31')
         .sort('CLOUDY_PIXEL_PERCENTAGE')
         .first()
         )
```

2) Draw the polygons and visually classifies each the one of the polygons (Figure 01)

Figure 01. The second step in the retrieve data process is to set land cover and label property that is assign for the polygon you drawn over the Sentinel image in the background.

3) Extracts the image pixels, band values and the land cover value.

```
# Use these bands for prediction.
bands = ['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7']

# This property of the table stores the land cover labels.
label = 'landcover'

# Overlay the points on the imagery to get training.
training = image.select(bands).sampleRegions(**{
  'collection': aoi,
  'properties': [label],
  'scale': 10,
  'tileScale':16
})
```

3) Repeat this process for different areas in Brazil in a manner that we have a balanced data set.

So, for each class of LULC I run this process in areas that I know that I would find samples for that class, e.g. when I was looking for forest samples I took an image from Amazon forest.

With the tabular data retrieved from image pixels it was possible to train a classification model.

2.2. Implementation

With the retrieved data I split it to train and test datasets and used it to model selection. I tested some of classification algorithms available on Earth Engine API, such as Classification and Regression Tree (CART), Gradient Boosting and Random Forest. The CART algorithm presented the best results while the others seems to overfit once that the train accuracy was much higher than the accuracy obtained on test dataset. The trained CART model was saved on a pickle file and is used on the app.

To deploy the model and app I wrote a notebook and used Voila to turn it in to html that is displayed on the web app page. This notebook load geemap package and uses it to plot a

interactive map and instructs the user to draw a polygon were the Sentinel image and classified image is plotted in. User can also select date range and, after click on submit button, the app selects the image with less cloud coverage inside the date range, captures each pixel data inside the polygon, uses it as input for the trained model and return the predicted values for each pixel.

To host the app it was necessary to create an account on Heroku, install Heroku Command Line Interface (CLI) that clones the git repository into Heroku. On Heroku there is a simple process to create the app that will read the files from repository and prepare the url to access the web app.

3. Analysis

3.1. Data Exploration

Sentinel-2 is a wide-swath, high-resolution, multi-spectral imaging mission that acquire 100x100 km^2 ortho-images from all Earth surface each 5-10 days. The Sentinel-2 data contain 13 UINT16 spectral bands representing surface reflectance (Table 02).

Name	Description	Resolution (meters)	Wavelength (nm)
B1	Aerosols	60	443.9
B2	Blue	10	496.6
B3	Green	10	560
B4	Red	10	664.5
B5	Red Edge 1	20	703.9
B6	Red Edge 2	20	740.2
B7	Red Edge 3	20	782.5
B8	NIR	10	835
B8A	Red Edge 4	20	864.8
B9	Water vapor	60	945
B10	Cirrus	60	1373.5
B11	SWIR 1	20	1613.7
B12	SWIR 2	20	2202.4
QA60	Cloud mask	60	

Table 02. Brief description of the spectral bands from a Sentinel-2 image.

The B2, B3 and B4 spectral bands of Sentinel-2 corresponds to the classical RGB bands (Visible Range) and band B8 is the Near Infra-Red (NIR), all with 10 meters resolution. There are 6 bands with 20 meters resolution: 4 narrow bands in the VNIR vegetation red edge spectral domain (B5, B6, B7 and B8A) and 2 wider Short Wave Infra-Red (SWIR) bands (B11 and B12). It also have 3 bands with 60 meters focused towards cloud screening and atmospheric correction (https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/resolutions/radiometric).

For this project, I used B1 to B7 bands. I also took care to select images with the lowest cloud coverage available. Table 03 describe a summary of the bands from obtained the obtained dataset.

	B1	B2	B3	B4	B5	B6	B7
count	96820.000000	96820.000000	96820.000000	96820.000000	96820.000000	96820.000000	96820.000000
mean	510.722124	647.841458	906.806176	1085.325057	1345.081347	1794.111981	2020.281822
std	297.080105	366.067829	472.036585	861.388814	894.079007	938.323915	1060.846597
min	106.000000	103.000000	137.000000	82.000000	103.000000	47.000000	56.000000
25%	260.000000	332.000000	458.000000	184.000000	565.000000	1594.000000	1710.000000
50%	454.000000	575.000000	881.000000	946.000000	1418.000000	2091.000000	2448.000000
75%	808.250000	988.000000	1298.000000	1874.000000	2179.000000	2454.000000	2720.000000
max	1326.000000	3496.000000	3880.000000	3618.000000	3796.000000	3994.000000	4833.000000

Table 03. Summary for the train and test data set.

3.2. Visualization

Figure 02 shows a True Color (RGB) plot of the Sentinel-2 image for a Southeast Brazilian region.

Figure 02. True color image from Sentinel-2 satellites taken in the coordinates -22.95 latitude and -46.63 longitude.

It is also possible change to a False Color (infra-red/G/B) representation of the same image by choosing which bands to plot. On Figure 03 the Near infra-red band (B8) is shown as red color while green and blue steels with their respective bands.

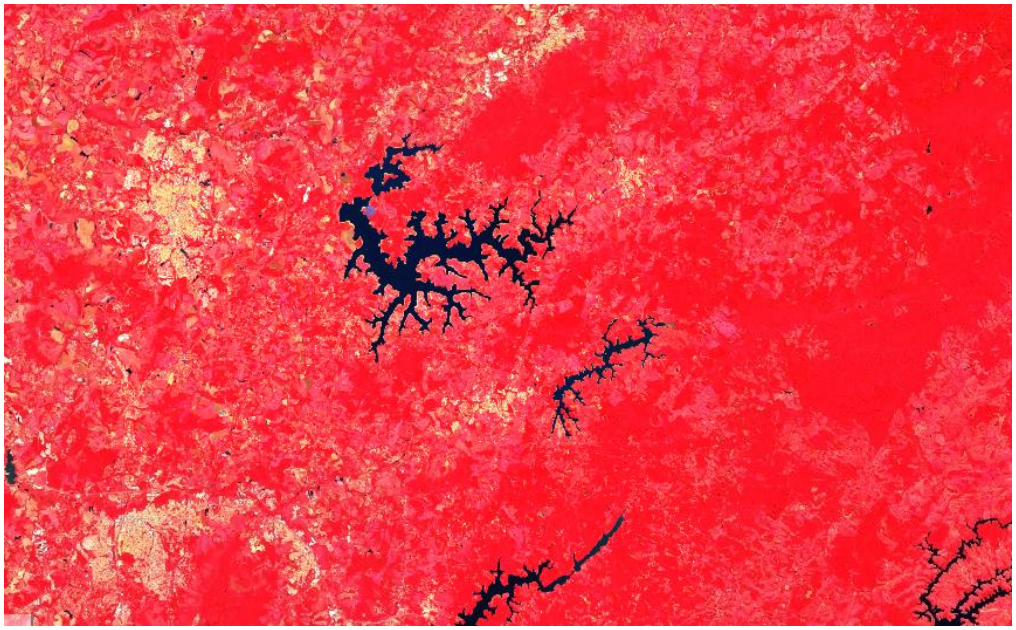


Figure 03. False color image ploted from the same Sentinel-2 data from Figure 02.

The human eye is trained to see patterns that are very difficult for a machine to identify. For example when we see in the images above as urban patches and how forested areas are easily identified by our vision. However, to identify these patterns we do not look at just the color, which corresponds to the values of the plotted spectral bands, but broader patterns in the images such as the roughness of the forested area or the right angles in urban areas.

There are more complex techniques that allow passing this type of information to the models, such as segment the image before classification (Xiong et al., 2016; Brinkhoff et al., 2020), however, in the present work, only the reflectance values of each band were used as information to the model. Therefore, it is important that the values of each spectral band are separated as different classes.

In Figure 04 are the average reflectance values of each band for each class of land use.

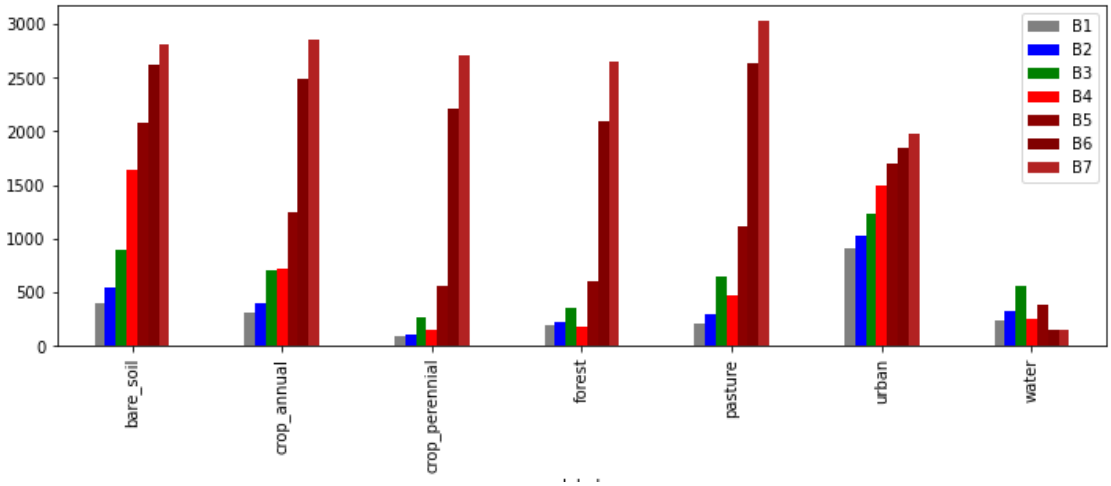


Figure 04. The average value of reflectance from the spectral bands for each landcover classes in the data that was used to train the model.

It is easy to see how the water presents a pattern so different from the other covers, this is because the water bodies absorb a good part of the wavelengths in all bands. On the other hand, we can see that some of the classes have a very similar spectral signature as perennial crops and forests, or pasture and annual crops, and represents a challenge to the classifying algorithm.

Figure 05 shows the boxplots for each band and land use, which allows to better verify the similarities and differences between the classes.

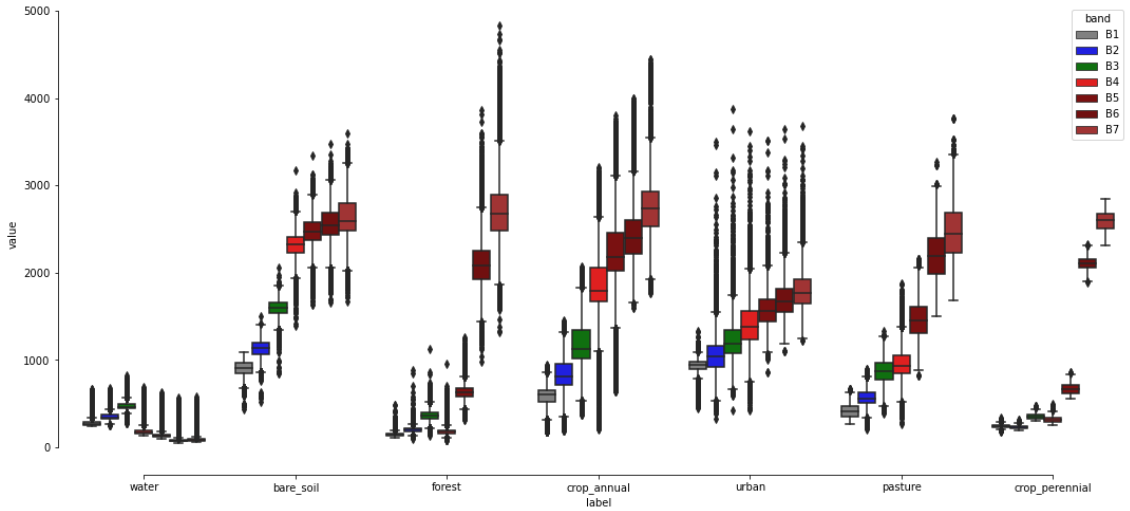


Figure 06. Boxplots for each band and land use

We see that although the average values of some classes are very similar, there are still differences that the algorithm can exploit. Like the case of forests and perennial crops that have low reflectance of bands B1 to B4 and high values for bands B6 and B7.

4. Results

4.1. Model Evaluation and Validation

With traditional train test split the model achieve very high accuracy of 0.9999612 and an almost perfect prediction as shown in the confusion matrix (Figure 06).

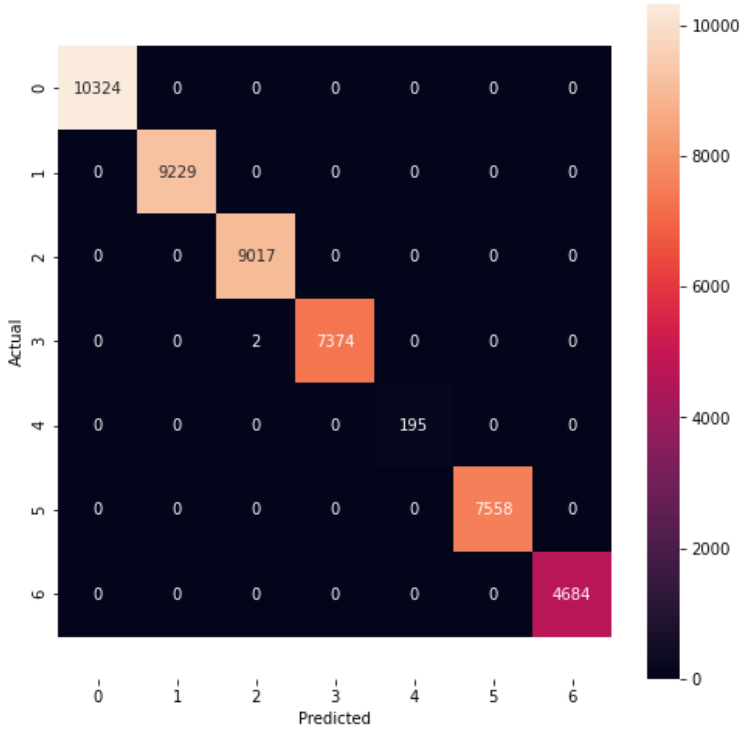
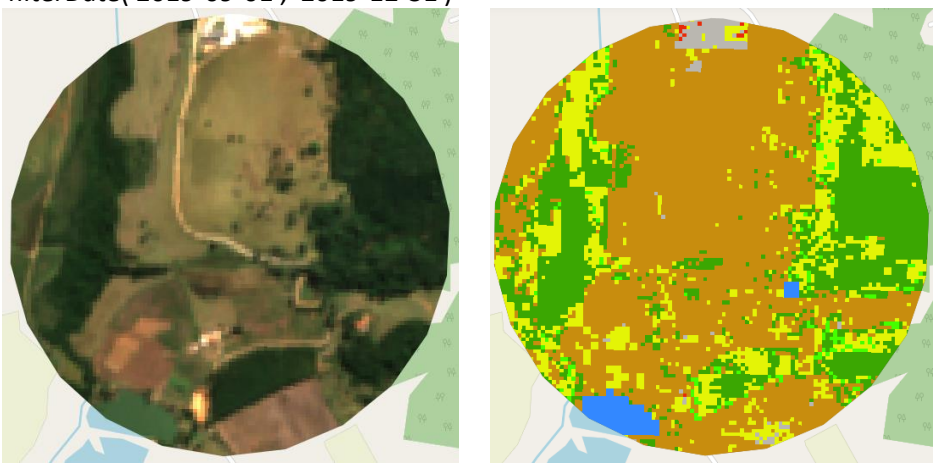


Figure 06. Confusion matrix for the results of the CART model with the test data.

We need to take in account that the way the data was collected does not allow us for a correct model evaluation. For that reason, I made a more rigid test getting data from other areas, very different from the ones I generate for train and test process, which causes a severe impact on model accuracy (0.5924). Xiong et al. (2016) presented a user's accuracy of 68.5% using a combination of random forest and SVM algorithms to classify LULC in Africa. Venter & Sydenham (2021) also used RF and obtained higher accuracy score of 90%. However, those authors had ground-truth data for the entire Europe, which is an advantage.

The main reason for the lower performance on this new data I attribute mainly due to the date. The training data was acquired from images taken between October and December 2019, that corresponds to wet season and the external validation data from dry season. During wet seasons the annual croplands had very different spectral response in comparison with dry season, when the crop lands where cropped and the soil is exposed (Figure 07).

filterDate('2019-09-01', '2019-12-31')



filterDate('2020-01-01', '2020-09-30')

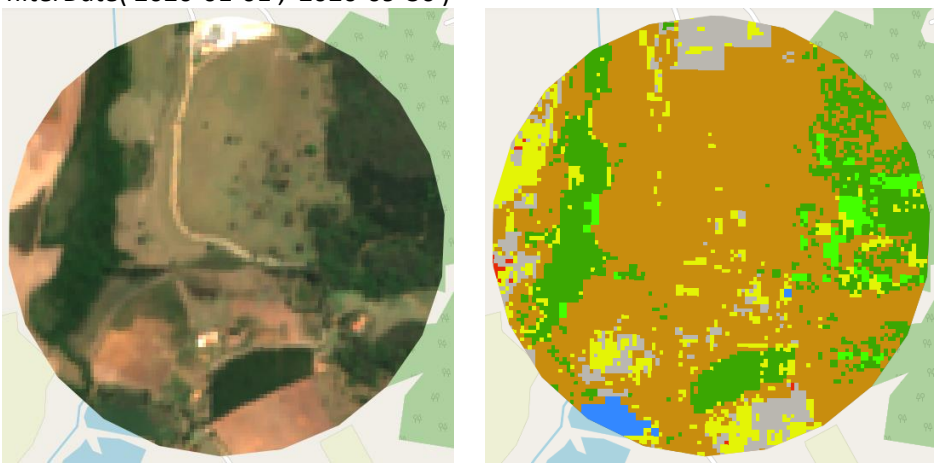


Figure 07. Comparison between the period from when the training data was taken (upper) and the external validation (lower images).

Figures 08 and 09 shows the results of classification for a larger area

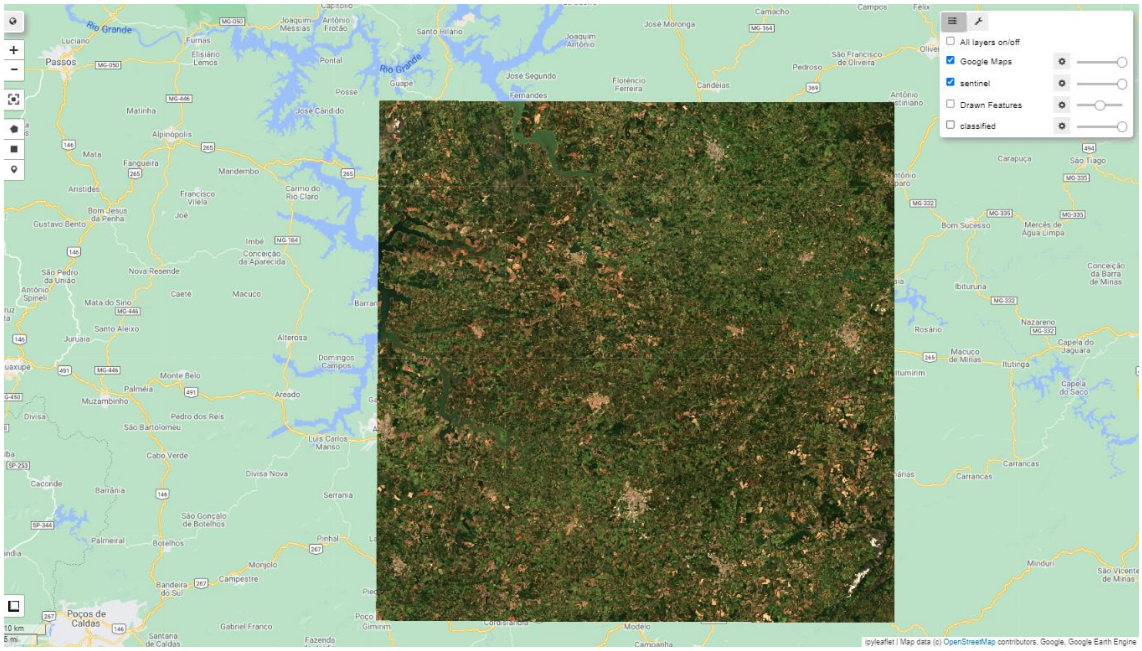


Figure 08. True color image of the Sentinel-2 data from the coordinates: -21.37 latitude and -45.30 longitude in 2019.

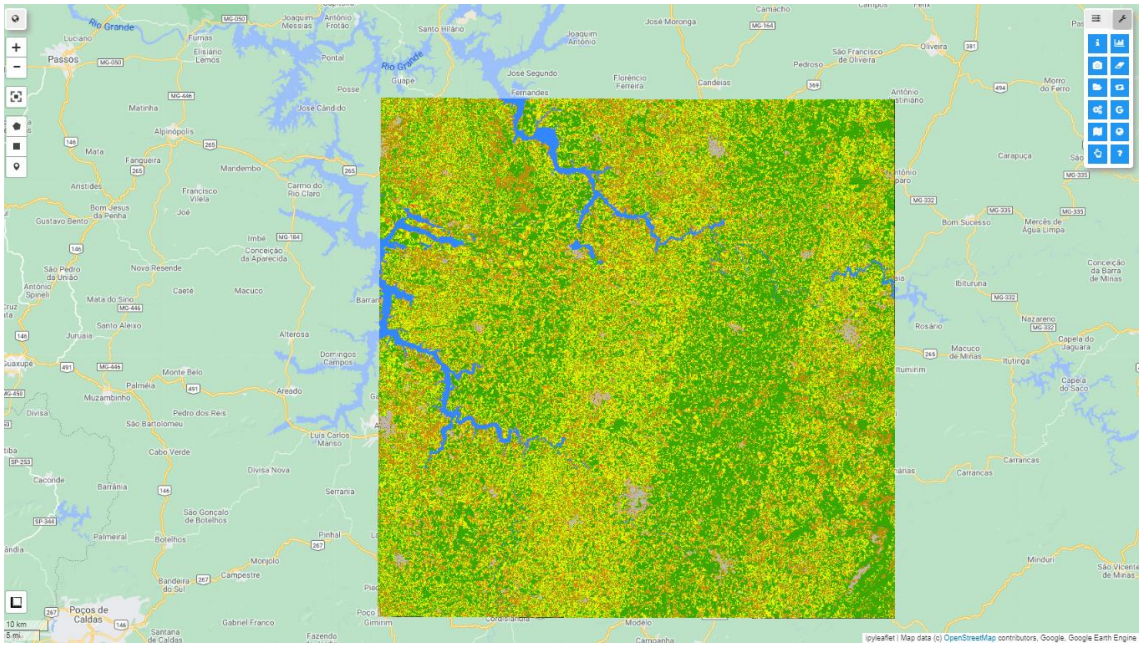


Figure 09. Classification obtained for the same área of Figure 08.

It is interesting to notice how the water areas are very well defined and classified correctly by the model, the same happens with urban areas. On the other hand, it is possible to see that there is problems in classifying others classes such as forest, pastures and annual crop areas.

5. Conclusion

5.1. Reflection

I initially thought it would be much more simple to obtain the data, however, I expend a huge time to make it work, and when it worked I realize that I should have done it a little differently, or at least have time to test other ways. I would like to have tested average values

instead of just taking the image with the lowest cloud coverage and to have data from different seasons and years.

On the other hand, I am very satisfied with the result and, above all, with the learning obtained. I believe I have gone far beyond the basic projects that are given as an option for the course since I had to develop geoprocessing skills and use the Earth Engine. In addition, I was able to host the app on a free platform that can be accessed and used freely.

There is still a lot to improve before it is an application that delivers value and could even be used in some business. However, what I learned gave me much more clarity on the way forward and I have many ideas to improve the predictive model.

5.2. Improvement

The most important improvement that needs to be done is in the accuracy of the model. To do this, it is necessary to rethink sampling and obtain more data. The use of a single image within a period causes a bias, so an alternative would be to obtain the median response of each band per pixel, or else to obtain data at different times of the year.

Use a for feature engineering process to combine bands and obtain vegetative indices such as NDVI is also an interest approach to improve the quality of the predictions. I am also interested in testing more advanced techniques such as image segmentation and custom models, that can be deployed on Earth Engine trought tensorflow.

6. Deliverables

6.1. Application

The app can be accessed through the link, were you can find instructions to run it. https://predict-land-cover-br.herokuapp.com/

6.2. Github Repository

The code used to deploy the app can be found on: https://github.com/LPontes/LULC_app

7. References

Brinkhoff et al. 2020. Land Cover Classification of Nine Perennial Crops Using Sentinel-1 and -2 Data. Remote Sensing. doi:10.3390/rs12010096.

Venter & Sydenham. 2021. Continental-scale land cover mapping at 10 m resolution over Europe (ELC10). Computer Vision and Pattern Recognition. https://arxiv.org/abs/2104.10922

Wu. 2020. geemap: A Python package for interactive mapping with Google Earth Engine. The Journal of Open Source Software. doi:10.21105/joss.02305

Xiong et al. 2016. Nominal 30-m Cropland Extent Map of Continental Africa by Integrating Pixel-Based and Object-Based Algorithms Using Sentinel-2 and Landsat-8 Data on Google Earth Engine. Remote Sensing. doi:10.3390/rs9101065.