Colab Instruction for Classifying Double-Cropped Fields

Hossein Noorazar

Contents

1 Prerequisite

First we need to create a Google Earth Engine (GEE) account. This is needed since we fetch the data from GEE. The data can be fetched from GEE either via https://code.earthengine.google.com/ which is a platform that uses JavaScript language or via Colab which is Python run on Google cloud. In this demonstration we use Colab.

To create a GEE account one must have a gmail. Then you go to https://earthengine.google.com/new_signup/ to sign in to your Google account and register as a new user. That page looks like Fig. 1 and you just need to follow the steps.

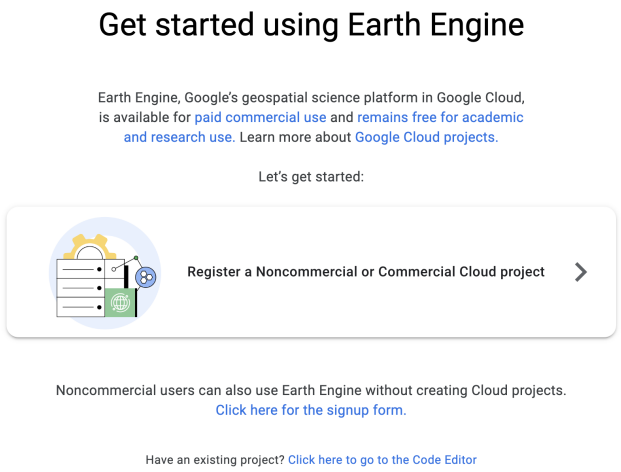

Figure 1: GEE sign up

2 Colab

When using Colab we need to have a Google Earth Engine account set up. The first step would be to *authenticate* ourselves and our account.

To run each cell on Jupyternotebook we need to hit command+return (or control+enter on windows machine) or

The Jupyternotebook that we use on Colab already has a cell that loads needed libraries and the Authentication command. That cell looks like Fig. 2a after it is run. A link is appeared there. We need to click on it and follow the steps. Once we click on the link we will be taken to another page (Fig. 2b) where we need to click on "Generate Token" after which we need to go through a couple of more clicks and then Google will give us an Authentication code. We need to copy that code and paste i in the empty box in Fig. 2a and hit enter.

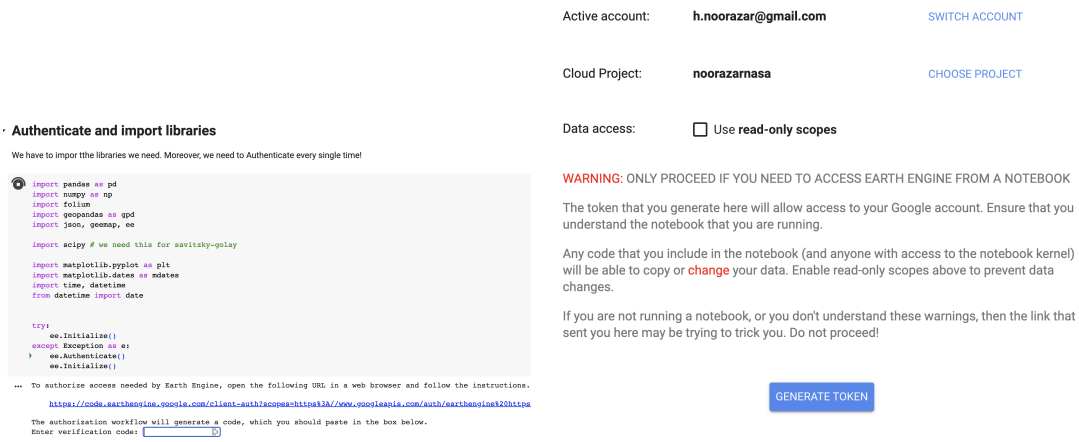

Figure 2: Authenticate on Colab.

After Authentication, we have a cell that connects the Colab Notebook to our Google Drive. We need to do this so that we can read shapefiles from our google drive and export the outputs to Google Drive as well. Once we run this cell (Fig. 3) another page pops up. We need to click on the Google Account and then click "Allow".

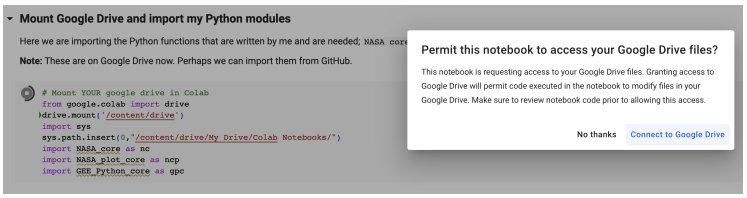

Figure 3: Mounting Google Drive.

Now, we have access to a few things. Let us go through them.

```
# Mount YOUR google drive in Colab
from google.colab import drive
drive.mount('/content/drive')
import sys
sys.path.insert(0, "/content/drive/My Drive/Colab Notebooks/")
import NASA_core as nc
import NASA_plot_core as ncp
import GEE_Python_core as gpc
```

The first line above are telling Colab to import a library that understands Google Drive. The second line is connecting the Notebook to Google Drive. And then we import `sys` which understands how to treat paths and such. Then we are telling the Notebook to look inside a folder called `Colab Notebooks`. This folder is inside Google Drive. If your files is in a folder called differently, then you need to change this lines accordingly.

The last three lines are importing three modules where they includes some functions that we need to use later in the Notebook.

Next, we need to tell the Notebook where the shapefile is located at. In this cell (Fig. 4).

First part of the path (`''/content/drive/MyDrive/''`) is in common among all users. The rest of this line is the folder names that I have created in my google drive and the name of the shapefile I am using. I have a folder called `NASA_trends` that has a subfolder called `shapefiles`. The subfolder `shapefiles` includes another folder called `Grant_4Fields_poly_wCentroids` which contains a shapefile called `Grant_4Fields_poly_wCentroids.shp`

Yes!!! I like to name the name of folder the same as the shapefile it contains!

Please tell me where to look for the shapefile!

```
shp_path = "/content/drive/MyDrive/NASA_trends/shapefiles/" + \
           "Grant_4Fields_poly_wCentroids/Grant_4Fields_poly_wCentroids.shp"

# we read our shapefile in to a geopandas data frame using the geopandas.read_file method
# we'll make sure it's initiated in the EPSG 4326 CRS
SF = gpd.read_file(shp_path, crs='EPSG:4326')

### for possible future use grab the data part of the shapefile
SF_data = SF[["ID", "Acres", "county", "CropTyp", \
              "DataSrc", "Irrigtn", "LstSrvD"]].copy()
SF_data.drop_duplicates(inplace=True)
print (SF_data.shape)

"""
   Drop extra useless columns. Saves space.**
   Also, GEE behaves strangely. It has problem with Notes column before
"""
# SF = SF.drop(columns=["Notes", "TRS", "IntlSrD", "RtCrpTy", "Shp_Lng", "Shap_Ar","CropGrp","Cro
SF = SF.drop(columns=["CropTyp","Acres","Irrigtn",
                      "LstSrvD","DataSrc","county","ExctAcr",
                      "cntrd_ln","cntrd_lt"])

SF.head(2)
```

Figure 4: Shapefile path in google drive.

In the same cell (Fig. 4) we read the shapefile and select some columns that might come handy. Also, by dropping unnecessary columns we save time and space; later when we have a time-series of vegetation indices (VI), and we merge shapefile into it, then for each field we will have repetition of unnecessary information if we do not drop those extra columns here.

The rest of the cells are pretty much simple and we only need to run them; carefully of course.

For example, there is a cell in which we have `start_date` and `end_date`. If we are using a 2025 shapefile and we are interested in classifying those fields, we need to set these parameters correctly.

Remark. In this Jupyter Notebook we are using Sentinel data. Moreover, This notebook is classifying the fields using Support Vector Machine (SVM). We have trained other methods such as Random Forest (RF), k-Nearest Neighbors (kNN) and Deep Learning (DL). We need to modify the Notebook to use DL. This may need a little work since traditionally we plot the data first, then read the figures off the disk and apply DL to it. Here we need to do it on the fly to save time.