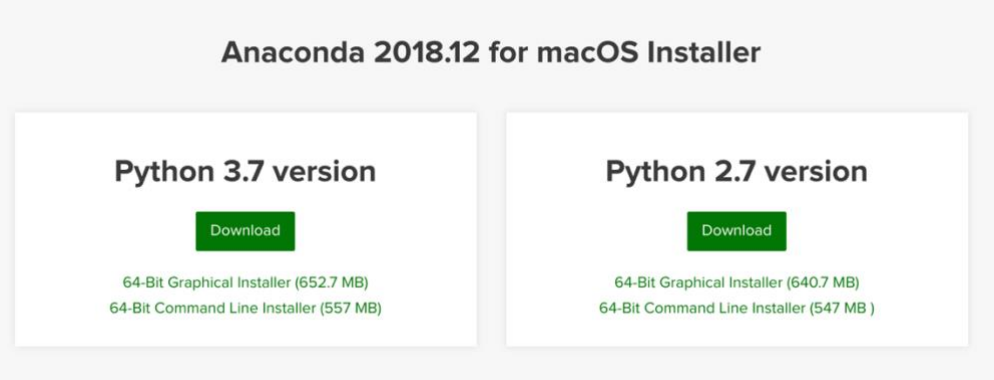
ReadMe

I wrote Python scripts for my study project (A Performance Assessment of the Random Forest Method for Land Cover Classification with Landsat Satellite Data in Mount Merapi Regions, Indonesia) in the form of functions in one file (**functions.py**). That way, the usage is flexible as any user can execute these functions using different input in any kind of combination possible. I have included the execution ***.py** files that calls these functions. I will explain how to use everything step-by-step.

I only use the terminal/shell (**iTerm**, in Mac) to execute all the codes and install all the Python packages. It will also work in Windows's Terminal (I used Windows the first time I learned Python), but the Python package library path on the disk will be different. The package library path is not difficult to find (there is a command to find the Python package library location, and I will include it).

Compiler

I run Anaconda Python 2.7 on terminal. Anaconda Python installer is available for download at https://www.anaconda.com/distribution/#macos (and then you choose the installer based on your operating system, e.g. Windows, MacOS, or Linux). There will be at least 2 Python versions available, as follows:



The reasons why I stick with Python 2.7 instead of updating it into version 3 are because, 1) it's easier to use as I don't always have to declare the variable type every time I perform a calculation, and 2) I have QGIS installed in my laptop that requires Python 2 (you can run multiple version of Python in one device but I was afraid I will mess it up).

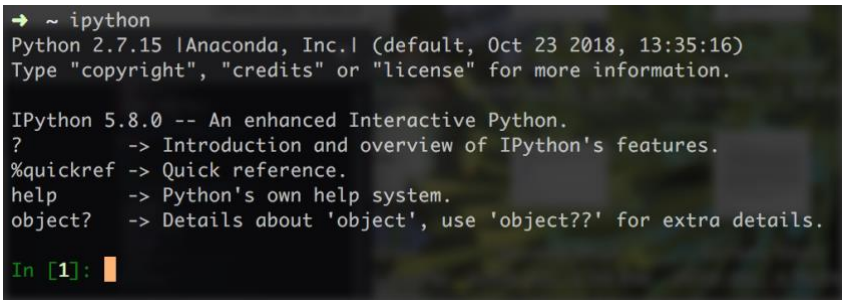
Shell command to check the Python version:

```
python --version
```

If my scripts were to be compiled using Python 3 then it will need some adjustments, e.g. importing the print function manually, declaring the variable type on every single calculation, etc.

If you're running on Mac, it is also important to acknowledge that by installing this, you will have to different Python packages in your device: 1) the Python that is required by the OS, 2) Anaconda Python. These two packages are located in different folder locations, and it is important for the library installation. I will get to this in the next section.

Instead of just calling `Python` on Terminal, I run Python scripts on Terminal using iPython (install in Terminal: `pip install ipython`). iPython can be called directly by typing `ipython` on Terminal. Then you can write and execute codes, or even execute ***.py** file using `%run [FILE NAME]`.



Text Editor

My preferred text editor is SublimeText because the interface is customizable and aesthetically pleasing. SublimeText can be downloaded at https://www.sublimetext.com/3. SublimeText can color-code Python scripts seamlessly, exhibit A:

```
#extract raster values of each sample point and save it into numpy array
#this extract 1 raster band at a time
def extract_values(shp, raster):

    """
    extract the FID with the Reflectance, sort it based on FID, then take only the Reflectance
    """

    #open raster
    src_ds = gdal.Open(raster)
    gt = src_ds.GetGeoTransform()
    rb = src_ds.GetRasterBand(1)

    #open shapefile
    ds = ogr.Open(shp)
    lyr=ds.GetLayer()
    li_values = list()
    for feat in lyr:
        geom = feat.GetGeometryRef()
        feat_id = feat.GetField('FID')
        mx,my=geom.GetX(), geom.GetY()  #coordinate in map units

        #Convert from map to pixel coordinates
        #Only works for geotransforms with no rotation
        px = int((mx - gt[0]) / gt[1]) #x pixel
        py = int((my - gt[3]) / gt[5]) #y pixel

        intval=rb.ReadAsArray(px,py,1,1)
        li_values.append([feat_id, intval[0]]) #this results in a list
        #convert the list into numpy array
        a = np.array(li_values).astype(np.float64)
        #sort the array by FID
        a = a[a[:,0].argsort(kind='mergesort')]
        #take out only the class ID (eliminate the FID)
        b = a[:,1]
        b = b.ravel()

    band_array = b.astype(np.float64)

    return(band_array)
```

While debugging, the code can be copy-pasted line by line into the Terminal. The full source code then can be saved as a ***.py** file that can be called directly from the Terminal.

Required Python Packages

In this section I will list the package versions that work for me (usually each

1. Pandas (install in **shell**: `pip install pandas`)
2. SimpleDBF (install in **shell**: `pip install simpledbf`)
3. Struct (usually already included in the default Python package)
4. Numpy (usually already included in the default Python package)
5. Scikit-learn (install in **shell**: `pip install scikit-learn`)

> **!!! personal advice!!!**
> In order to make sure that the packages are interoperable, you can also opt to install it together with Numpy and Scipy as a whell package with:
> `pip install numpy scipy scikit-learn`

6. Rasterio (install in **shell**: `pip install rasterio`)
7. GDAL (install with an installer from http://www.kyngchaos.com/software/frameworks/)

> **!!! installing GDAL might be tricky!!!**
> This might vary, but based on my experience, the installer will automatically extract the package on the MacOS Python library, instead of the Anaconda Python library. Therefore, I needed to copy and paste the package into the Anaconda Python library. I will create a whole section for GDAL Python binding installation in MacOS.
>
> The Python command to check the Anaconda Python library path (on any OS):

```
import sys
sys.path
```

Installing GDAL Python binding on MacOS

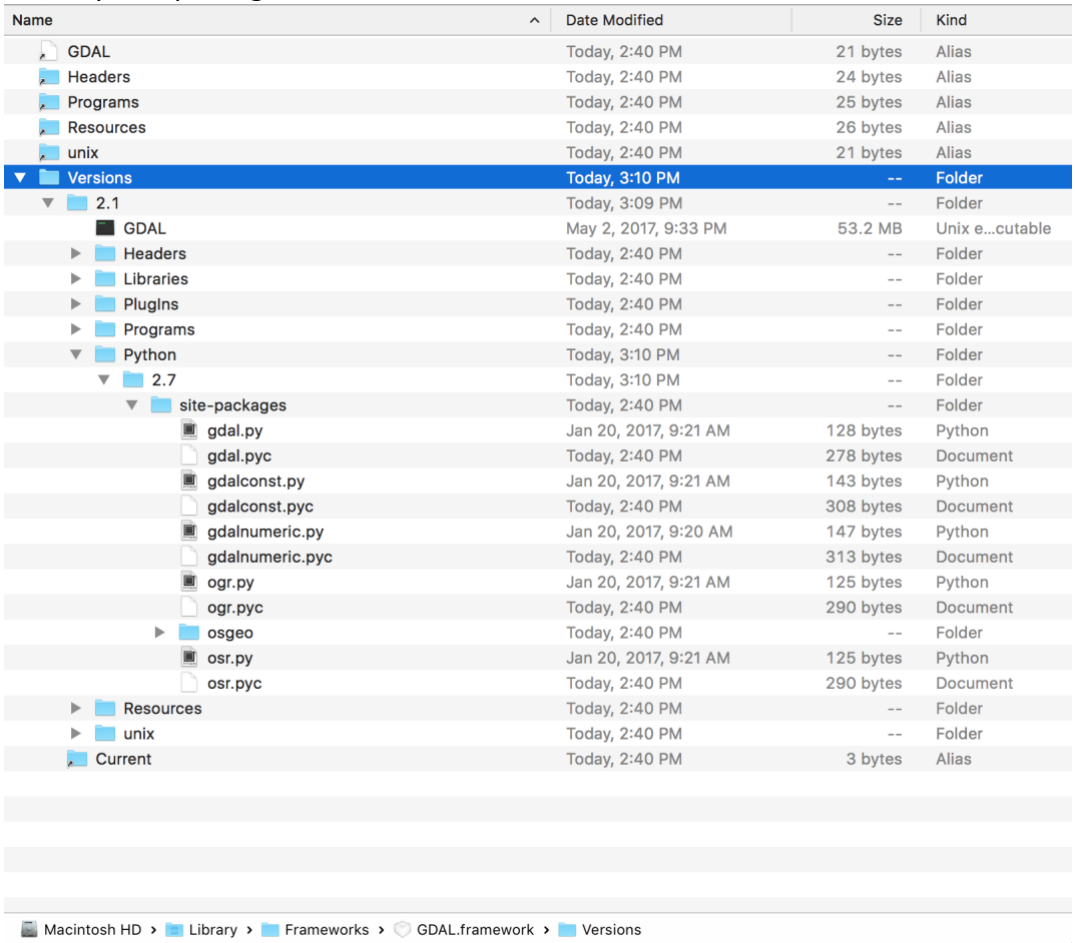
1. Download the package from http://www.kyngchaos.com/software/frameworks/ (select GDAL 2.1 if you want to keep working with 2.7 and not the newer version 3.6)
2. Install using the installer
3. Open iTerm, call the path using:

```
Export
PATH=/Library/Frameworks/GDAL.framework/Programs:$PATH
```

4. Check if it's accessible from iTerm using:

```
gdalinfo -version
```

5. Actually, the package is installed here



6. Meanwhile, that folder is not yet listed in the Python sys.path (or check first by going into iPython mode in iTerm, then: `import sys; sys.path`).

```
import sys
sys.path
```

```
In [1]: import sys

In [2]: sys.path
Out[2]:
['',
 '/Users/aviputripertiwi/.conda/bin',
 '/Users/aviputripertiwi/.conda/lib/python27.zip',
 '/Users/aviputripertiwi/.conda/lib/python2.7',
 '/Users/aviputripertiwi/.conda/lib/python2.7/plat-darwin',
 '/Users/aviputripertiwi/.conda/lib/python2.7/plat-mac',
 '/Users/aviputripertiwi/.conda/lib/python2.7/plat-mac/lib-scriptpackages',
 '/Users/aviputripertiwi/.conda/lib/python2.7/lib-tk',
 '/Users/aviputripertiwi/.conda/lib/python2.7/lib-old',
 '/Users/aviputripertiwi/.conda/lib/python2.7/lib-dynload',
 '/Users/aviputripertiwi/.conda/lib/python2.7/site-packages/setuptools-23.0.0-py2.7.egg',
 '/Users/aviputripertiwi/.conda/lib/python2.7/site-packages',
 '/Users/aviputripertiwi/.conda/lib/python2.7/site-packages/IPython/extensions',
 '/Users/aviputripertiwi/.ipython']
```

7. Therefore, copy and paste the GDAL package manually into the sys.path.
   The MacOS Python path:

   `/Library/Frameworks/GDAL.framework/Versions/2.1/Python/2.7/site-packages`

   Copy the whole GDAL-related content into

   `/Users/[username]/.conda/lib/python2.7/site-packages`

8. Restart iPython. Now the module can be imported in iPython

```
In [6]: import gdal

In [7]: from osgeo import gdal
```

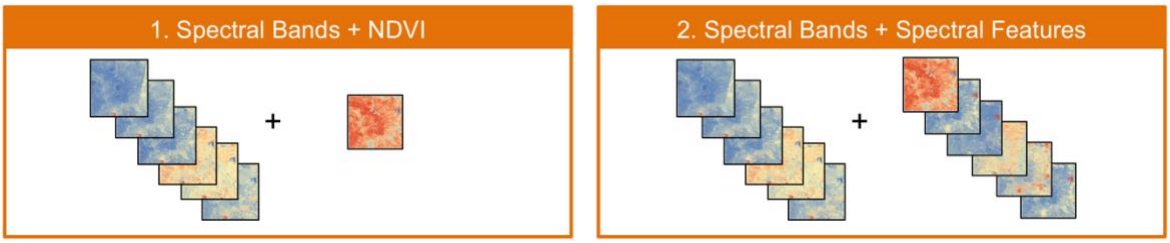
Model Scenarios

Land cover classification in Remote Sensing is performed by selecting sample points from the ROI to train and be predicted by the classifier. There are three cases of the number of sample points in order to verify that the number of samples is proportionate to the accuracy. Case A consists of 0.5% of the number of raster pixels, case B 1%, and case C 2%. Note that only the pixels with the SLC-off cell values were taken into account. The 2009 and 2011 datasets have different pixel count due to the mosaic results, hence the resulting number of sample points also differ. Then, the sample points were split into the training (by 80%) and test sample points (by 20%).

		Sample points	
		2009	**2011**
Case A	0.5% raster pixels*	21,933	19,221
Case B	1% raster pixels*	43,867	38,442
Case C	2% raster pixels*	87,734	76,884

*the number of raster pixels of the SLC-off mask

These cases were run on these two scenarios of input raster bands: 1) Spectral Bands + NDVI (SB+NDVI), 2) Spectral Bands + Spectral Features (SB+SF) in order to check if the Spectral Features can actually increase the accuracy of the classification.



List of *.py files

Containing all the used functions:

- functions.py

Calling functions to predict test samples of the Spectral Band+NDVI scenario:

- process_case_A_testonly.py
- process_case_B_testonly.py
- process_case_C_testonly.py

Calling functions to predict test samples of the Spectral Band+Spectral Features scenario:

- extract_img_with_SF_case_A.py
- extract_img_with_SF_case_B.py
- extract_img_with_SF_case_C.py
- process_case_A_sf_testonly.py
- process_case_B_sf_testonly.py
- process_case_C_sf_testonly.py

Calling functions to predict the whole map for the result with the highest accuracy (also resulting in a raster file of predicted land cover map):

- run_CaseC_300trees_2009.py
- run_CaseC_300trees_2011.py

Calling functions to calculate band importance (without prediction):

- calculate_vim.py

Input files

Training sample points (SHP)

- 2009 classification
  - Case A: 'Samples/SLC off/0_5percent/s09train.shp'
  - Case B: 'Samples/SLC off/1percent/s09train.shp'
  - Case C: 'Samples/SLC off/2percent/s09train.shp'
- 2011 classification
  - Case A: 'Samples/SLC off/0_5percent/s11train.shp'
  - Case B: 'Samples/SLC off/1percent/s11train.shp'
  - Case C: 'Samples/SLC off/2percent/s11train.shp'

Test sample points (SHP)

- 2009 classification
  - Case A: 'Samples/SLC off/0_5percent/s09test.shp'
  - Case B: 'Samples/SLC off/1percent/s09test.shp'
  - Case C: 'Samples/SLC off/2percent/s09test.shp'
- 2011 classification
  - Case A: 'Samples/SLC off/0_5percent/s11test.shp'
  - Case B: 'Samples/SLC off/1percent/s11test.shp'
  - Case C: 'Samples/SLC off/2percent/s11test.shp'

Tabular data of the training sample points (DBF)

- 2009 classification
  - Case A: 'Samples/SLC off/0_5percent/s09train.dbf'
  - Case B: 'Samples/SLC off/1percent/s09train.dbf'
  - Case C: 'Samples/SLC off/2percent/s09train.dbf'
- 2011 classification
  - Case A: 'Samples/SLC off/0_5percent/s11train.dbf'
  - Case B: 'Samples/SLC off/1percent/s11train.dbf'
  - Case C: 'Samples/SLC off/2percent/s11train.dbf'

Tabular data of the test sample points (DBF)

- 2009 classification
  - Case A: 'Samples/SLC off/0_5percent/s09test.dbf'
  - Case B: 'Samples/SLC off/1percent/s09test.dbf'
  - Case C: 'Samples/SLC off/2percent/s09test.dbf'
- 2011 classification
  - Case A: 'Samples/SLC off/0_5percent/s11test.dbf'
  - Case B: 'Samples/SLC off/1percent/s11test.dbf'
  - Case C: 'Samples/SLC off/2percent/s11test.dbf'

Spectral Band + Spectral Features 2009 (TIF)

Band 1 = 'L2 imagery/2009/clip_b1r.tif'
Band 2 = 'L2 imagery/2009/clip_b2r.tif'
Band 3 = 'L2 imagery/2009/clip_b3r.tif'
Band 4 = 'L2 imagery/2009/clip_b4r.tif'

Band 5 = 'L2 imagery/2009/clip_b5r.tif'
Band 7 = 'L2 imagery/2009/clip_b7r.tif'
NDVI = 'L2 imagery/2009/ndvi.tif'
NDWI = 'L2 imagery/2009/ndwi.tif'
MNDWI1 = 'L2 imagery/2009/mndwi1.tif'
MNDWI2 = 'L2 imagery/2009/mndwi2.tif'
NDBI = 'L2 imagery/2009/ndbi.tif'
MNDBI = 'L2 imagery/2009/mndbi.tif'

Spectral Band + Spectral Features 2011 (TIF)

Band 1 = 'L2 imagery/2011/clip_b1r.tif'
Band 2 = 'L2 imagery/2011/clip_b2r.tif'
Band 3 = 'L2 imagery/2011/clip_b3r.tif'
Band 4 = 'L2 imagery/2011/clip_b4r.tif'
Band 5 = 'L2 imagery/2011/clip_b5r.tif'
Band 7 = 'L2 imagery/2011/clip_b7r.tif'
NDVI = 'L2 imagery/2011/ndvi.tif'
NDWI = 'L2 imagery/2011/ndwi.tif'
MNDWI1 = 'L2 imagery/2011/mndwi1.tif'
MNDWI2 = 'L2 imagery/2011/mndwi2.tif'
NDBI = 'L2 imagery/2011/ndbi.tif'
MNDBI = 'L2 imagery/2011/mndbi.tif'

Ground Truth Data

!!! These files are not used in the script, but are provided in the DVD.
The sample points were created based on these shapefiles (SHP) using ArcGIS.

- 2009 land cover map of the ROI path: 'Ground Truth Data/2009/'
- 2011 land cover map of the ROI path: 'Ground Truth Data/2011/'

The functions.py file

This file contains all used functions for this project:

- `read_class` to read the class ID of the reference data and save it into Numpy array
- `extract_values` to extract raster values of each sample point and save it into Numpy array
- `combine_bands` to stack the resulting Numpy array of Band 1, 2, 3, 4, 5, 7, and NDVI.
- `combine_bands_sf` to stack the resulting Numpy array of Band 1, 2, 3, 4, 5, 7, NDVI, NDWI, MNDWI1, MNDWI2, NDBI, and MNDBI.
- `create_band` to read the whole raster file and save it into Numpy array
- `train_rf` to train the Random Forest and predict image/test sample (Numpy array)
- `vim` to calculate the band importance
- `rasterize` to write a raster file from a Numpy array

- `test_accuracy` to calculate overall accuracy and confusion matrix

Calling Python functions

I have provided all the ***.py** files that call the functions in the **functions.py** file depending on the targeted result. These ***.py** files can be executed directly on Terminal.

Python commands to call the functions manually:

1. Go to the folder containing the **functions.py** file on Terminal
2. Call iPython
3. Import the **functions.py** file

```
import functions
```

4. For example, to call the `extract_`values function as follows:

```
extract_values(shp, raster)
```

where,
shp = the path of the sample shapefile
raster = the path of a band raster file

use the command:

```
band1 = functions.extract_values(shp = 'Samples/SLC
off/0_5percent/s09train.shp', raster = 'L2
imagery/2009/clip_b1r.tif')
```

where,
band1 will be the variable of the returning result from

```
return(band_array)
```

in the function.

Functions without return statement doesn't need declaring variable when called.

The descriptions regarding the input of each function were already commented on the script.

Executing "Function Caller" Files

1. Spectral Band + NDVI Scenario
    - process_case_A_testonly.py
    - process_case_B_testonly.py
    - process_case_C_testonly.py

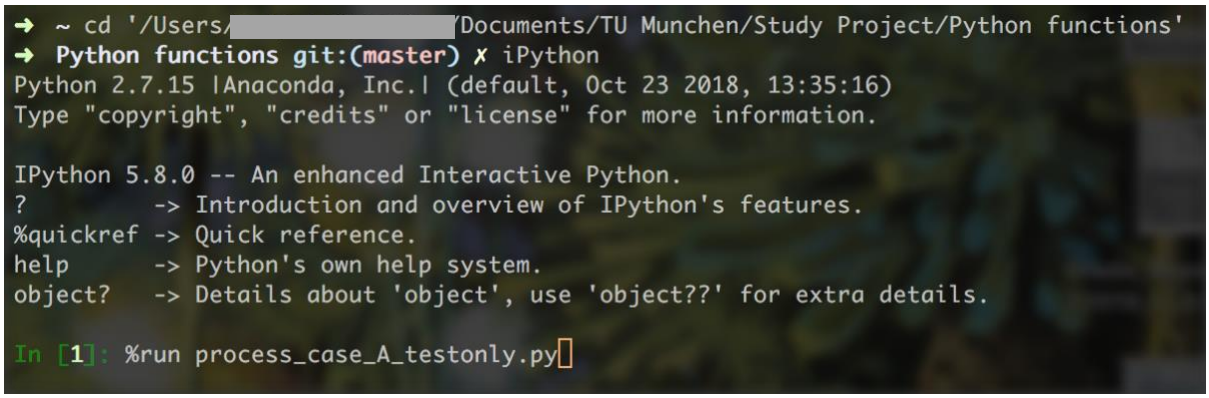
2. Spectral Band + Spectral Feature Scenario
   For some reason I split the "function caller" into two:
    a. Extracting the Training Samples
        - extract_img_with_SF_case_A.py
        - extract_img_with_SF_case_B.py
        - extract_img_with_SF_case_C.py

    b. Extracting and predicting the Test Samples
        - process_case_A_sf_testonly.py
        - process_case_B_sf_testonly.py
        - process_case_C_sf_testonly.py

   Execute this on the following order: a first, and then b

3. Predicting the whole map of the scenarios with the highest accuracy
   * the scenario was selected manually, I didn't automate the case with the highest overall accuracy to be predicted
    - run_CaseC_300trees_2009.py
    - run_CaseC_300trees_2011.py

In order to run this file (one by one, of course), go to Terminal, go to the intended path (where all these ***.py** files are located), open iPython, then call the intended ***.py** by using the command `%run`



Result files

***.npy files**

Many of the outputs from the functions I wrote were in the form of Numpy array files (***.npy**). I already included all these *.npy files in the DVD. If you execute the codes that I wrote, these files will be overwritten automatically.

All result files are included in the DVD in the following path: 'L2 imagery/Results/'