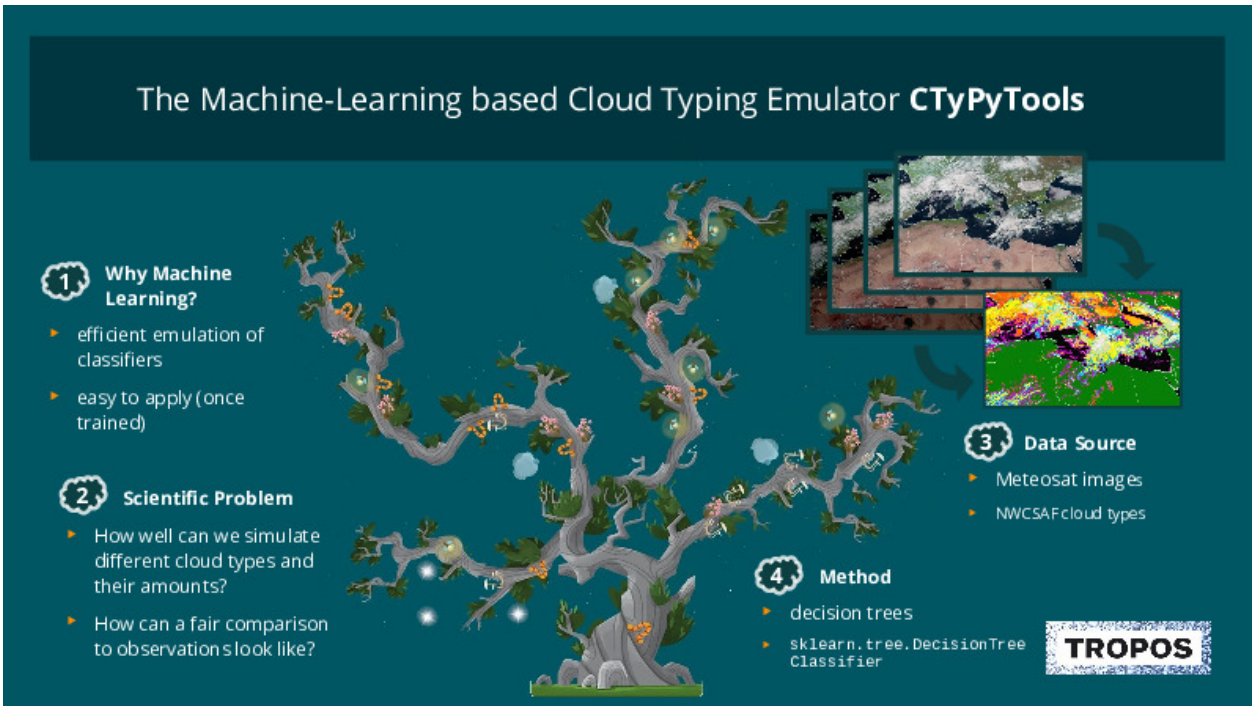
CTyPyTool

Release v0.1

Fabian Senf

Oct 26, 2022

INTRO


The Machine-Learning based Cloud Typing Emulator CTyPyTools
1 Why Machine Learning?
efficient emulation of classifiers
easy to apply (once trained)
2 Scientific Problem
How well can we simulate different cloud types and their amounts?
How can a fair comparison to observations look like?
3 Data Source
Meteosat images
NWCSAF cloud types
4 Method
decision trees
sklearn.tree.DecisionTree Classifier
TROPOS

CHAPTER

ONE

CTYPYTOOL: CLOUD TYPING PYTHON TOOL

This tools is intended to help weather forecasters in assessing the quality of their cloud forecasts.

A few facts:

- It emulates a cloud typing methodology (see https://www.nwcsaf.org/ct2) applied to Meteosat data (see https://www.eumetsat.int/meteosat-second-generation).
- It uses standard machine learning techniques, e.g. tree & random forest classifier
- It can be applied to so-called synthetic satellite data (observsation-equivalents derived from numerical forecast data).

Schematic

source/docs/images/ctypytools-slide.jpg

1.1 Installation

1.1.1 On your Local Computer

Cloning repository

Use the following command to clone the project to your local machine.

```
$ git clone https://github.com/fsenf/CTyPyTool
```

Installing Dependencies:

This project comes with a Pipfile specifying all project dependencies. When using `pipenv` first move into the project folder with:

```
$ cd cloud_classification
```

and then use the following command to install all necesarry dependencies into your virtual environment

```
$ pipenv install
```

1.1.2 On the DKRZ Servers

See here to get started with `CTyPyTools` on the DKRZ Super computer.

1.2 Getting Started

There are severeal Jupyter Notebooks explaining the basic steps for training and applying the cloud classifier.

For using an already trained classifier check out this notebook

1.3 Contributing

Your Contribution is very welcome! Yo could either contribute with:

- providing pre-trained classifiers for a specifically defined geographical region or for certain sessions
- reporting issues, missing features or bugs
- improving code

4 Steps for source code developers:

1. fork the `devel` branch
2. update source code / software parts in your fork
3. check functionality with example notebooks
4. make a pull request onto the `devel` branch in the "official" repository under https://github.com/fsenf/CTyPyTool

CHAPTER

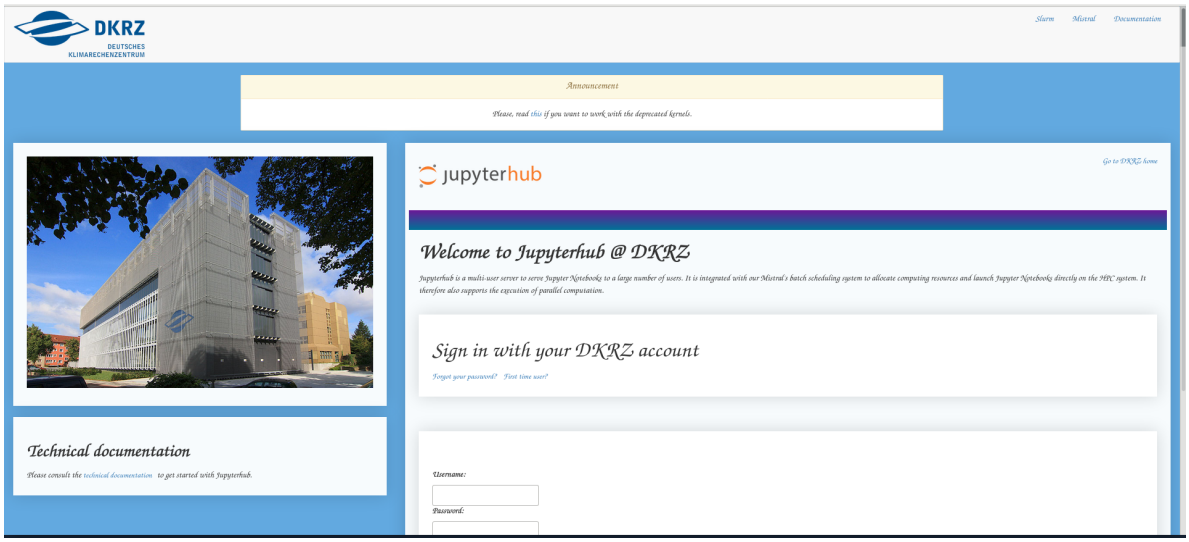
TWO

USING CLOUD CLASSIFICATION TOOLS ON JUYPTERHUB

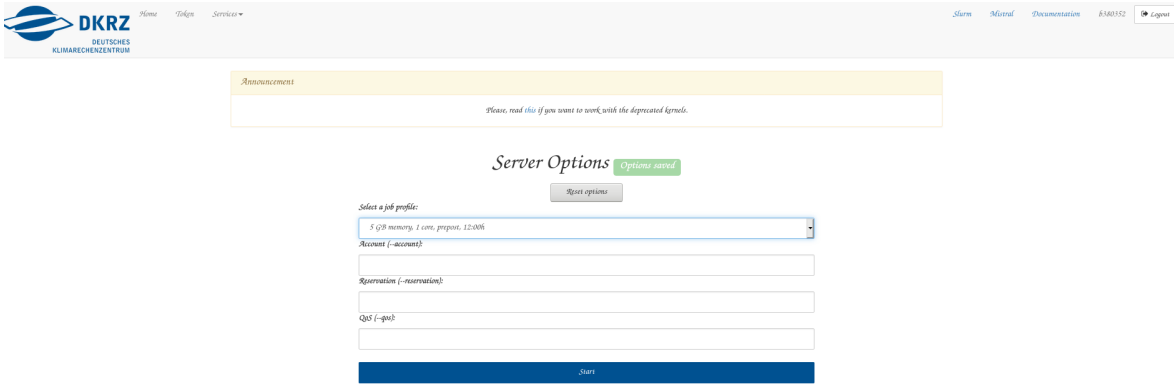
This description is developed for the application of our cloud classification tool on DKRZ JupyterHub. However, it should be applicable for other jupyterhub serivces with minor changes.

2.1 Login into JupyterHub

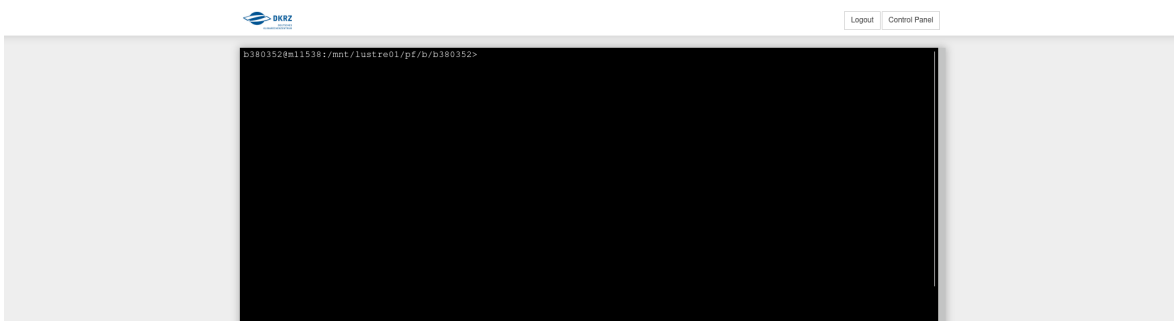
- First, you need to log into JupyterHub following the link https://jupyterhub.dkrz.de and enter your account details.



- Second, you select a preset. I like the 5 GB, prepost setting.



- Third, you open a terminal in your JuypterHub session (alternatively, you could login via `ssh` and use the ssh-terminal session)



2.2 Installation of Cloud Classification Software and Its Dependencies

- Select a project location `<cloud_type_project_directory>`. You might need to create a new one!

```
> cd <cloud_type_project_directory>
```

- **Get the git repository** (using `git clone`)

```
> git clone https://github.com/fsenf/CTyPyTool.git
Cloning into 'CTyPyTool'...
remote: Enumerating objects: 587, done.
remote: Counting objects: 100% (587/587), done.
remote: Compressing objects: 100% (270/270), done.
remote: Total 587 (delta 296), reused 581 (delta 294), pack-reused 0
Receiving objects: 100% (587/587), 12.80 MiB | 19.00 MiB/s, done.
Resolving deltas: 100% (296/296), done.
```

  Perfect! The source is there!

- **Look at dependencies**
  - The `PipFile` only names `numpy` & `request` as dependencies, see

```
> cd CTyPyTool
> cat Pipfile
...

[packages]
requests = "*"
numpy = "*"

[dev-packages]

[requires]
python_version = "3.8"
```

Both might be part of the standard anaconda env. We ignore the dependencies, here. You might need to install the packages an other platforms.

The list of dependencies also looks a bit incomplete (TODO: check all loaded modules!)

2.3 Application 1: Run Example Cases with a Pretrained Tree Classifier

2.3.1 Download Data and Classifier

- make a download folder (we assume that you are already in the `CTyPyTool` directory)

```
mkdir download
cd download
```

- start the download of zips with (currently located on swiftbrowser; later this will move on zenodo)

```
link="https://swiftbrowser.dkrz.de/tcl_objects/2023-10-14T14:41:05Z/r_
↪7d20b33512e14d8b56ea40e25aa35978bfc3921f/w_/dkrz_d7550ef1-c227-4463-a6a7-
↪29c14dc05fde/cloud_typing_project/11/classifier/"
wget -r -H -N --cut-dirs=3 --content-disposition --no-directories -I "/v1/" "${link}
↪/?show_all"
```

The `*zip` Files contain pretrained classifiers. The `index.html?show_all` also comes along but is not needed.

- let us extract the tree classifier

```
> unzip -d ../classifiers TreeClassifier.zip
Archive:  TreeClassifier.zip
   creating: ../classifiers/TreeClassifier/
   creating: ../classifiers/TreeClassifier/data/
  inflating: ../classifiers/TreeClassifier/data/classifier
  inflating: ../classifiers/TreeClassifier/data/label_reference.nc
  inflating: ../classifiers/TreeClassifier/data/training_data
   creating: ../classifiers/TreeClassifier/filelists/
  inflating: ../classifiers/TreeClassifier/filelists/evaluation_sets.json
  inflating: ../classifiers/TreeClassifier/filelists/input_files.json
  inflating: ../classifiers/TreeClassifier/filelists/label_files.json
  inflating: ../classifiers/TreeClassifier/filelists/training_sets.json
```

(continues on next page)

(continued from previous page)

```
   creating: ../classifiers/TreeClassifier/labels/
  inflating: ../classifiers/TreeClassifier/labels/nwcsaf_msevi-medi-20190317_1800_
↪predicted.nc
  inflating: ../classifiers/TreeClassifier/labels/nwcsaf_msevi-medi-20190318_1100_
↪predicted.nc
   creating: ../classifiers/TreeClassifier/settings/
  inflating: ../classifiers/TreeClassifier/settings/config.json
  inflating: ../classifiers/TreeClassifier/settings/data_structure.json
```

Nice!

- Get NWCSAF & Meteosat Data & Georef for Running the Examples
  - Extract the data:

```
> unzip -d .. data.zip
Archive:  data.zip
   creating: ../data/
   creating: ../data/auxilary_files/
  inflating: ../data/auxilary_files/lsm_mask_medi.nc
  inflating: ../data/auxilary_files/msevi-medi-georef.nc
  inflating: ../data/auxilary_files/msevi_georef.nc
   creating: ../data/example_data/
  inflating: ../data/example_data/msevi-medi-20190317_1800.nc
  inflating: ../data/example_data/msevi-medi-20190318_1100.nc
  inflating: ../data/example_data/nwcsaf_msevi-medi-20190317_1800.nc
  inflating: ../data/example_data/nwcsaf_msevi-medi-20190318_1100.nc
```

  - On the content:

```
> cd ../data
> tree
.
|-- auxilary_files
|   |-- lsm_mask_medi.nc
|   |-- msevi-medi-georef.nc
|   `-- msevi_georef.nc
`-- example_data
    |-- msevi-medi-20190317_1800.nc
    |-- msevi-medi-20190318_1100.nc
    |-- nwcsaf_msevi-medi-20190317_1800.nc
    `-- nwcsaf_msevi-medi-20190318_1100.nc

2 directories, 7 files
```

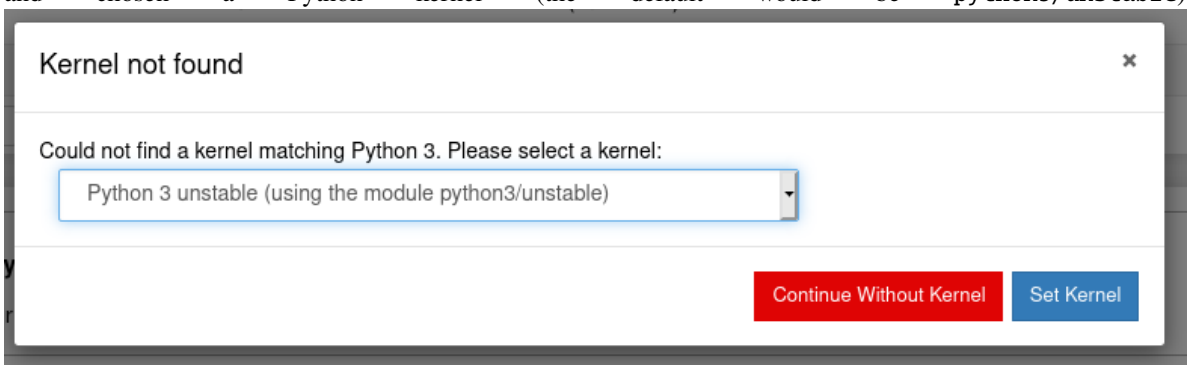
OK, the downloaded data contains a land-sea mask and a georeference in `auxilary_files/` plus two Meteosat and NWCSAF cloud typing files for the Mediterranean region in `example_data/`.

2.3.2 Run The Tests on JupyterHub

For the application 1 we will work with the notebook `Application_of_a_pretrained_classifier.ipynb` which shows how a pre-trained classifier is loaded and applied to example data.

Go through the following steps:

1. go to the JupyterHub browser tab and navigate to the `notebooks` directory
2. open the notebook `Application_of_a_pretrained_classifier.ipynb` and chosen a Python kernel (the default would be `python3/unstable`)



3. finally run the notebook and watch out for errors!

2.4 Application 2: Change to a Forest Classifier and Run the Examples Again

2.4.1 Get the Forest Classifier Data

We assume that executed all steps mentioned above and that you

- already downloaded the forest classifier
- like to apply the forest classifier on the same example data
- and now went back to your `download` directory with the content:

```
> tree
.
|-- ForestClassifier.zip
|-- TreeClassifier.zip
`-- data.zip

0 directories, 3 files
```

- For extracting the Forest Classifier, run:

```
> unzip -d ../classifiers ForestClassifier.zip
Archive:  ForestClassifier.zip
   creating: ../classifiers/ForestClassifier/
   creating: ../classifiers/ForestClassifier/data/
  inflating: ../classifiers/ForestClassifier/data/classifier
  inflating: ../classifiers/ForestClassifier/data/label_reference.nc
```

(continues on next page)

(continued from previous page)

```
  inflating: ../classifiers/ForestClassifier/data/training_data
   creating: ../classifiers/ForestClassifier/filelists/
  inflating: ../classifiers/ForestClassifier/filelists/evaluation_sets.json
  inflating: ../classifiers/ForestClassifier/filelists/input_files.json
  inflating: ../classifiers/ForestClassifier/filelists/label_files.json
  inflating: ../classifiers/ForestClassifier/filelists/training_sets.json
   creating: ../classifiers/ForestClassifier/labels/
  inflating: ../classifiers/ForestClassifier/labels/nwcsaf_msevi-medi-20190317_1800_
↪predicted.nc
  inflating: ../classifiers/ForestClassifier/labels/nwcsaf_msevi-medi-20190318_1100_
↪predicted.nc
   creating: ../classifiers/ForestClassifier/settings/
  inflating: ../classifiers/ForestClassifier/settings/config.json
  inflating: ../classifiers/ForestClassifier/settings/data_structure.json
```

- Prepations on JupyterHub:
  - Goto to your JuypterHub browser tab and select the checkbox of the notebook `Application_of_a_pretrained_classifier.ipynb`
  - Press the "Duplicate" button - you get a copy of this notebook
  - Click on the copy - it will open in a new browser tab
  - Rename the notebook e.g. to "Test_the_Forest_Classifier_Example" - click on the title on the top row, just right to the DKRZ logo
- Testing the forest classifier notebook:
  - apply the following modifications:

    1. Replace line in "In [3]:"

    from

    ```
    path = "../classifiers/TreeClassifier"
    ```

    to

    ```
    path = "../classifiers/ForestClassifier"
    ```

    2. Replace line in "In [9]:"

    from

    ```
    ls ../classifiers/TreeClassifier/labels
    ```

    to

    ```
    ls ../classifiers/ForestClassifier/labels
    ```

  - press the run button and watch out for errors.

This takes slightly longer. I might change the descriptive text in markdown for your own reference.

2.5 Application 3: Examples for Plotting Classifier Labels

Plotting of labels and some rudimetary statistics is performed in the notebook `Plotting_of_example_data.ipynb`

If both prior application examples on the application of a pre-trained tree and forest classifier went well, then you just need to start the notebook `Plotting_of_example_data.ipynb` in your JupyterHub (selecting the default kernel) and run it. That`s it …

CHAPTER

THREE

APPLICATION OF A PRETRAINED CLASSIFIER

The project aims to use maschine learning methods to emulate a cloud classification scheme. The classifer can be trained using large amounts of data and later be used to predict cloud types from satelite data. Those two steps can be run separately.

This notebook contains a short explanation how to use a pretrained classifier in order to predict labels from new input data.

3.1 Imports

At first we need to point python to the project folder. The path can be assigned as a relative path as shown below, or as an absolute system path. Than the module can be imported via the `import cloud_classifier` command.

[53]:
```
import sys
sys.path.append('../cloud_classifier')
import cloud_classifier
import importlib
importlib.reload(cloud_classifier)
```

[53]:
```
<module 'cloud_classifier' from '/home/squidy/tropos/CTyPyTool/notebooks/../cloud_
↪classifier/cloud_classifier.py'>
```

3.2 Initialization

Our first step is to create a classifier object:

[54]:
```
cc = cloud_classifier.cloud_classifier()
```

Than we need to point our classifier object to an already existing classifier. The `load_project()` method will load an existing classifier into our classifier object.

[55]:
```
path = "../classifiers/TreeClassifier"
cc.load_project(path)
```

3.3 Applying the Classifier: Prediction of Cloud Type Labels

3.3.1 Using a User-Defined File List

In order to predict labels with the now loaded classifier, we need to specify input files of satelite data. This can be done manually via in `input_files` option in the `set_project_parameters` method.

[56]:
```
file_1 = "../data/example_data/msevi-medi-20190317_1800.nc"
file_2 = "../data/example_data/msevi-medi-20190318_1100.nc"

cc.set_project_parameters(input_files = [file_1, file_2])
```

We now run the prediction pipeline (with the `run_prediction_pipeline()` method) which * applies the classifier to our input data and * stores the predicted labels.

The option `create_filelist` is set to `False` to take the user-defined input file list.

[57]:
```
cc.run_prediction_pipeline(create_filelist = False)
```

```
/home/squidy/.local/share/virtualenvs/CTyPyTool-idbccFiL/lib/python3.8/site-packages/
↪sklearn/base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeClassifier␣
↪from version 1.0 when using version 1.0.2. This might lead to breaking code or invalid␣
↪results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-
↪limitations
  warnings.warn(
```

```
Classifier loaded!
Masked indices set!
Input vectors created!
Labels saved as nwcsaf_msevi-medi-20190317_1800_predicted.nc
Input vectors created!
Labels saved as nwcsaf_msevi-medi-20190318_1100_predicted.nc
```

3.3.2 Using an Automatically Generated Input File List

Alternatively to the manual definition, the input file list can be generated automatically.

The easiest way to do so is to put all input files into an input data folder (here it is set to `../data_example_data`) and just tell the classifier where to look via the `input_source_folder` option.

[58]:
```
%%bash

ls -l ../data/example_data
```

```
total 30120
-rw-rw-r-- 1 squidy squidy 14946418 Jun  4  2021 msevi-medi-20190317_1800.nc
-rw-rw-r-- 1 squidy squidy 15552552 Jun  4  2021 msevi-medi-20190318_1100.nc
-rw-rw-r-- 1 squidy squidy   155069 Jun  4  2021 nwcsaf_msevi-medi-20190317_1800.nc
-rw-rw-r-- 1 squidy squidy   178946 Jun  4  2021 nwcsaf_msevi-medi-20190318_1100.nc
```

[59]:
```
cc.set_project_parameters(input_source_folder = "../data/example_data")
```

In a next step, we can let the classifier predict labels from the input files we have specified. This is again done with the `run_prediction_pipeline()` method.

If we want the classifier to automatically generate a list of input files and therefore set the option `create_filelist` to `True`.

[60]:
```
cc.run_prediction_pipeline(create_filelist = True)
```

```
Input filelist created!
Classifier loaded!
Masked indices set!
```

```
/home/squidy/.local/share/virtualenvs/CTyPyTool-idbccFiL/lib/python3.8/site-packages/
↪sklearn/base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeClassifier␣
↪from version 1.0 when using version 1.0.2. This might lead to breaking code or invalid␣
↪results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-
↪limitations
  warnings.warn(
```

```
Input vectors created!
Labels saved as nwcsaf_msevi-medi-20190318_1100_predicted.nc
Input vectors created!
Labels saved as nwcsaf_msevi-medi-20190317_1800_predicted.nc
```

3.4 Accessing predicted labels

The predicted labels are stored in the folder of the classifier we are using. They are located in the subfolder `labels`.

[61]:
```
%%bash

ls ../classifiers/TreeClassifier/labels
```

```
nwcsaf_msevi-medi-20190317_1800_predicted.nc
nwcsaf_msevi-medi-20190318_1100_predicted.nc
```

[ ]:
```

```

CHAPTER

FOUR

PLOTTING OF EXAMPLE DATA

4.1 Imports

At first we need to point python to the project folder. The path can be assigned as a relative path as shown below, or as an absolute system path. For plotting of data the `cloud_plotter` modle is used, which can be imported via the `import cloud_plotter` command.

[32]:
```
import sys
sys.path.append('../cloud_classifier')
import cloud_plotter
import importlib
importlib.reload(cloud_plotter)
```

[32]:
```
<module 'cloud_plotter' from '/home/squidy/tropos/CTyPyTool/notebooks/../cloud_
↪classifier/cloud_plotter.py'>
```

4.2 Initialization

Our first step is to create a plotter object and to load a previously created cloud classifier project. Loading the project is neccesarry in order to import all project settings like the location of auxilary files.

[33]:
```
cp = cloud_plotter.cloud_plotter()

path = "../classifiers/ForestClassifier"
cp.load_project(path)
```

Next we specify some label files we want to plot. In this example the data consists of one original label file and two files of predicted labels, one from the Decision Tree and the other from the Random Forest Classifier.

[41]:
```
orig_file = "../data/example_data/nwcsaf_msevi-medi-20190317_1800.nc"
tree_prediction = "../classifiers/TreeClassifier/labels/nwcsaf_msevi-medi-20190317_1800_
↪predicted.nc"
forest_prediction = "../classifiers/ForestClassifier/labels/nwcsaf_msevi-medi-20190317_
↪1800_predicted.nc"
```

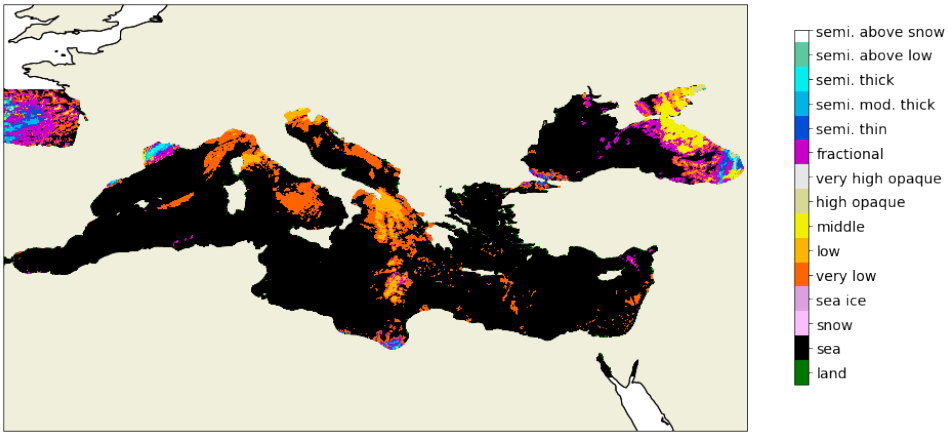
4.3 Plotting

4.3.1 Individual Plots

Using the `plot_data` method we can plot each of those datasets individually

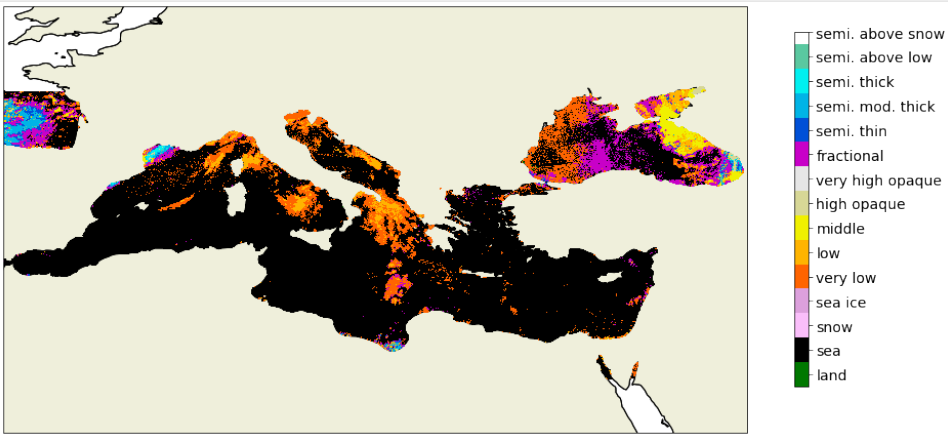
[35]:
```
cp.plot_data(label_file = orig_file, colorbar = True)
```

nbsphinx-code-borderwhite



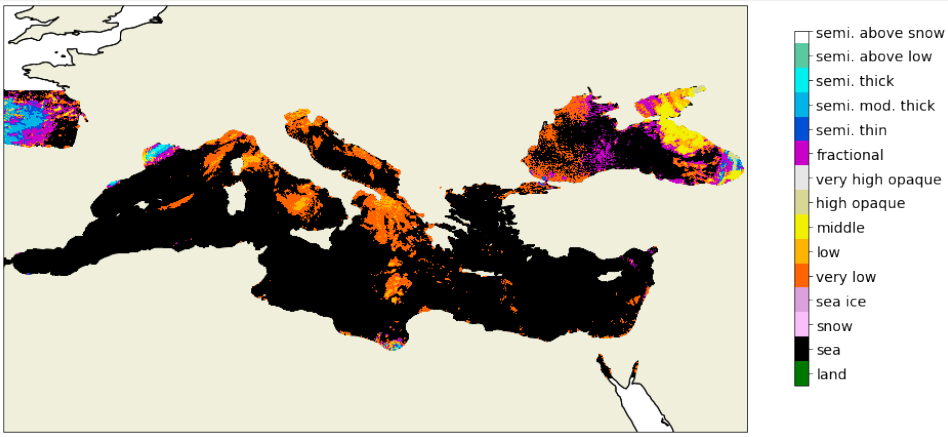
[36]:
```
cp.plot_data(label_file = tree_prediction, colorbar = True)
```

nbsphinx-code-borderwhite



[37]:
```
cp.plot_data(label_file = forest_prediction, colorbar = True)
```

nbsphinx-code-borderwhite

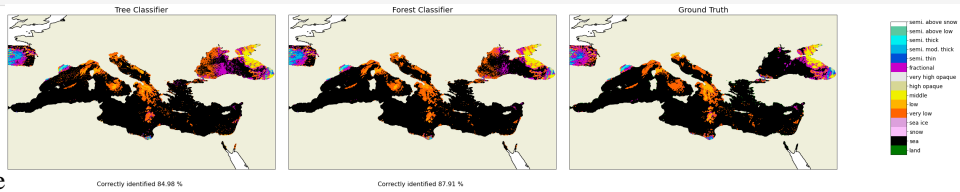


4.3.2 Combined Plots

Using the `plot_multiple` method we can plot multile datasets next to each other and evaluate the predcition performance in respect to the original labels.

[38]:
```
titles = ["Tree Classifier", "Forest Classifier"]
cp.plot_multiple(label_files = [tree_prediction, forest_prediction], truth_file = orig_
↪file, plot_titles = titles)
```

nbsphinx-code-borderwhite



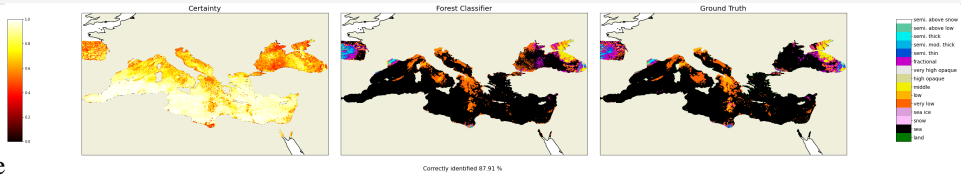
4.3.3 Probabilites Plots

The labels predicted with the Random Forest Classifier come with a probability score. That is, for each data point there also is a measure of how certain the classifier is about its choice of label. Those certainties are also stored in the label files and can be plooted using the `plot_probas` method.

[39]:
```
titles = ["Certainty", "Forest Classifier"]

cp.plot_probas(label_file = forest_prediction, truth_file = orig_file, plot_titles =␣
↪titles)
```

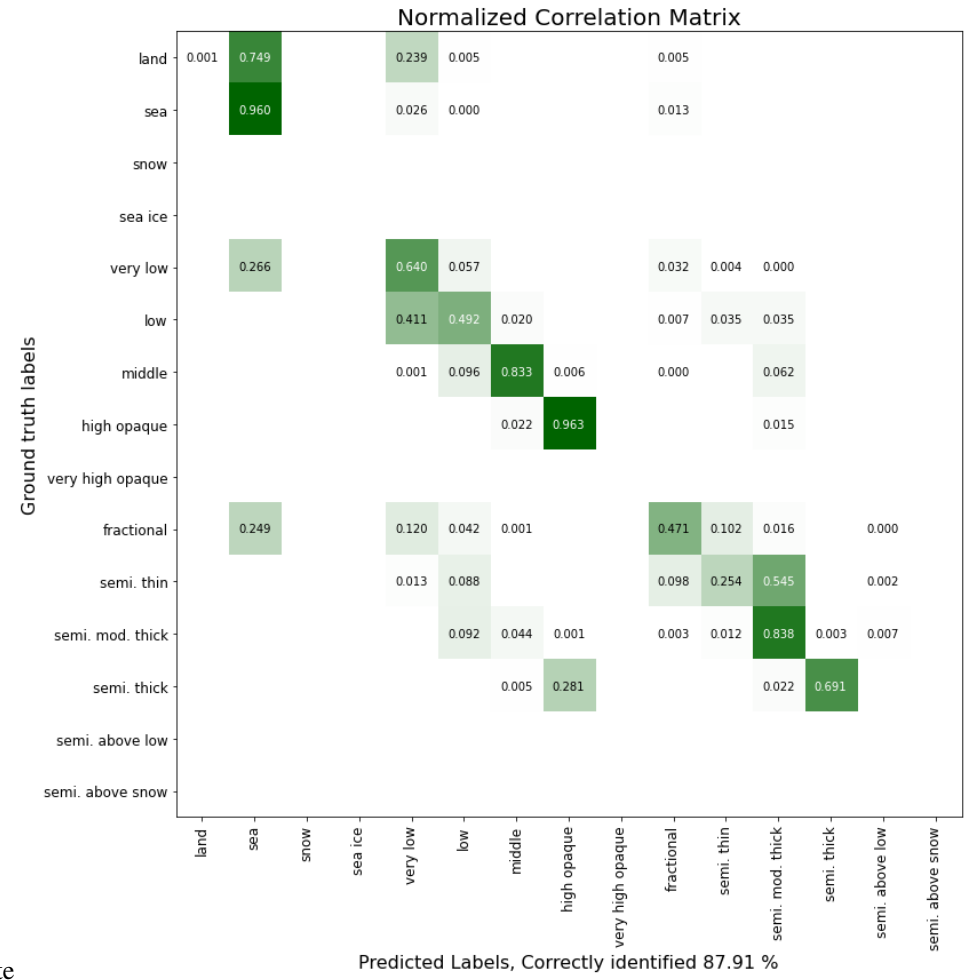
nbsphinx-code-borderwhite



4.3.4 Correlation Matrix Plots

Given predicted labels and the original file we can also compute and plot a correlation matrix via the `plot_coocurrence_matrix` method

[40]:
```
cp.plot_coocurrence_matrix(forest_prediction, orig_file)
```



nbsphinx-code-borderwhite

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search