

→ THE ESA EARTH OBSERVATION Φ -WEEK

EO Open Science and FutureEO

12–16 November 2018 | ESA–ESRIN | Frascati (Rome), Italy

EO data processing in QGIS with a python API

Andreas Rabe, Benjamin Jakimow, Fabian Thiel, Sebastian van der Linden*

**andreas.rabe@geo.hu-berlin.de*

<http://www.hu-geomatics.de>

FOSS

Free Open Source



CSS

Closed Source

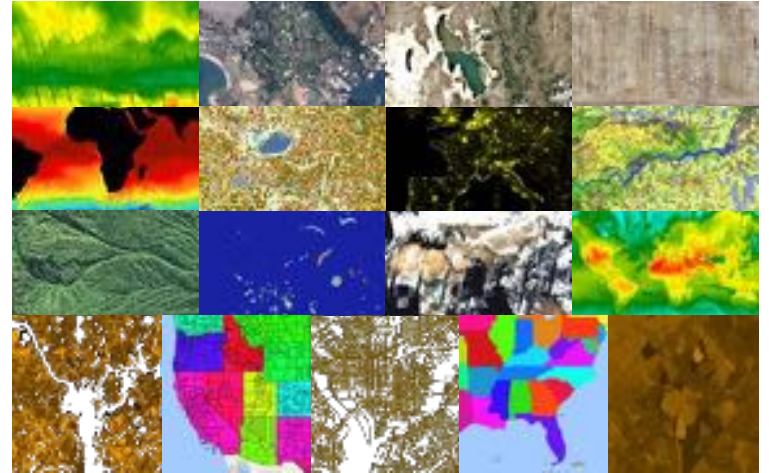


User



EO Data

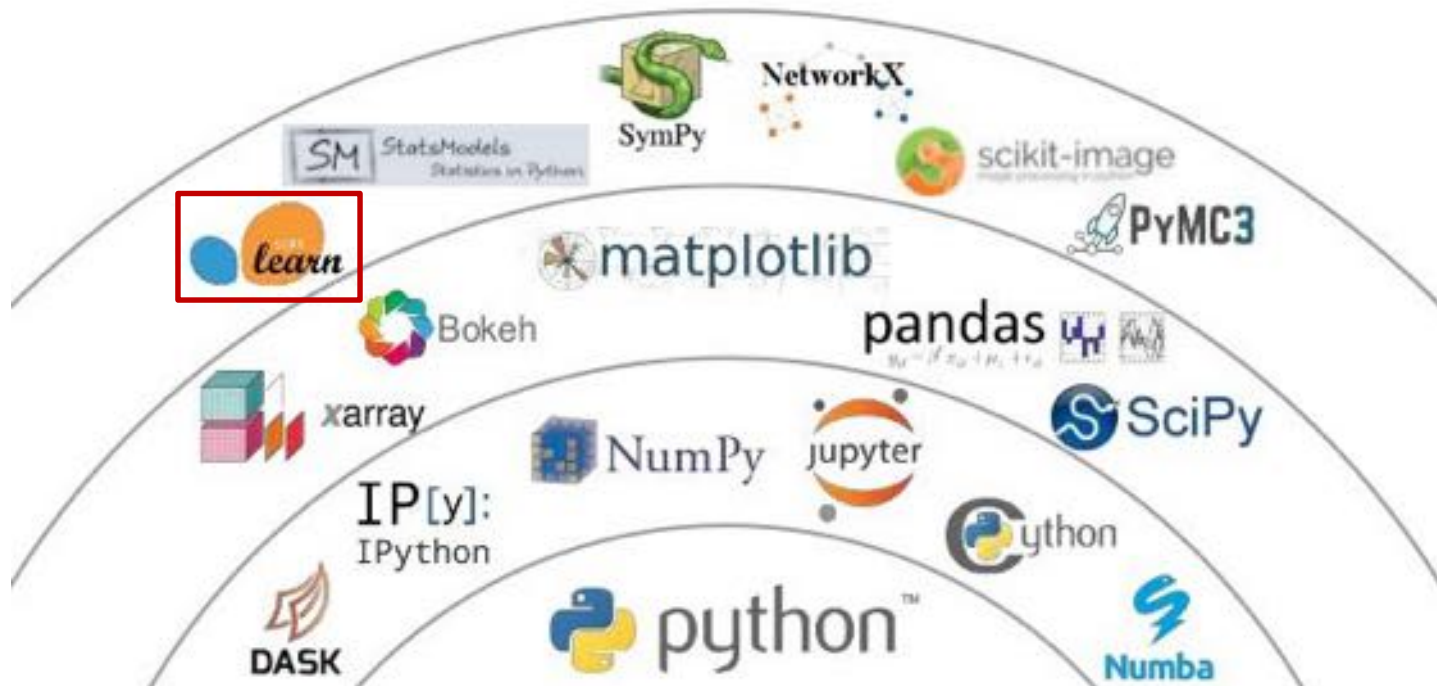
Climate, Imagery, Geophysical, Census



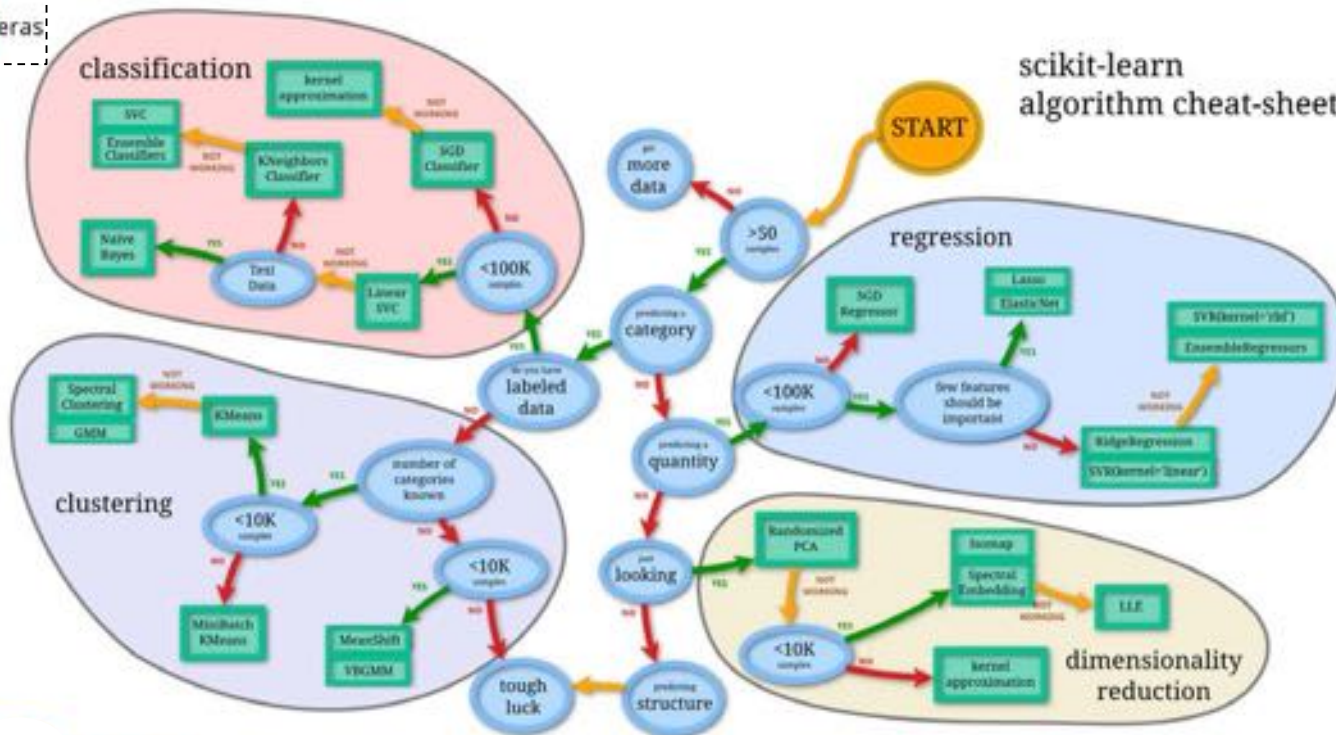
QGIS



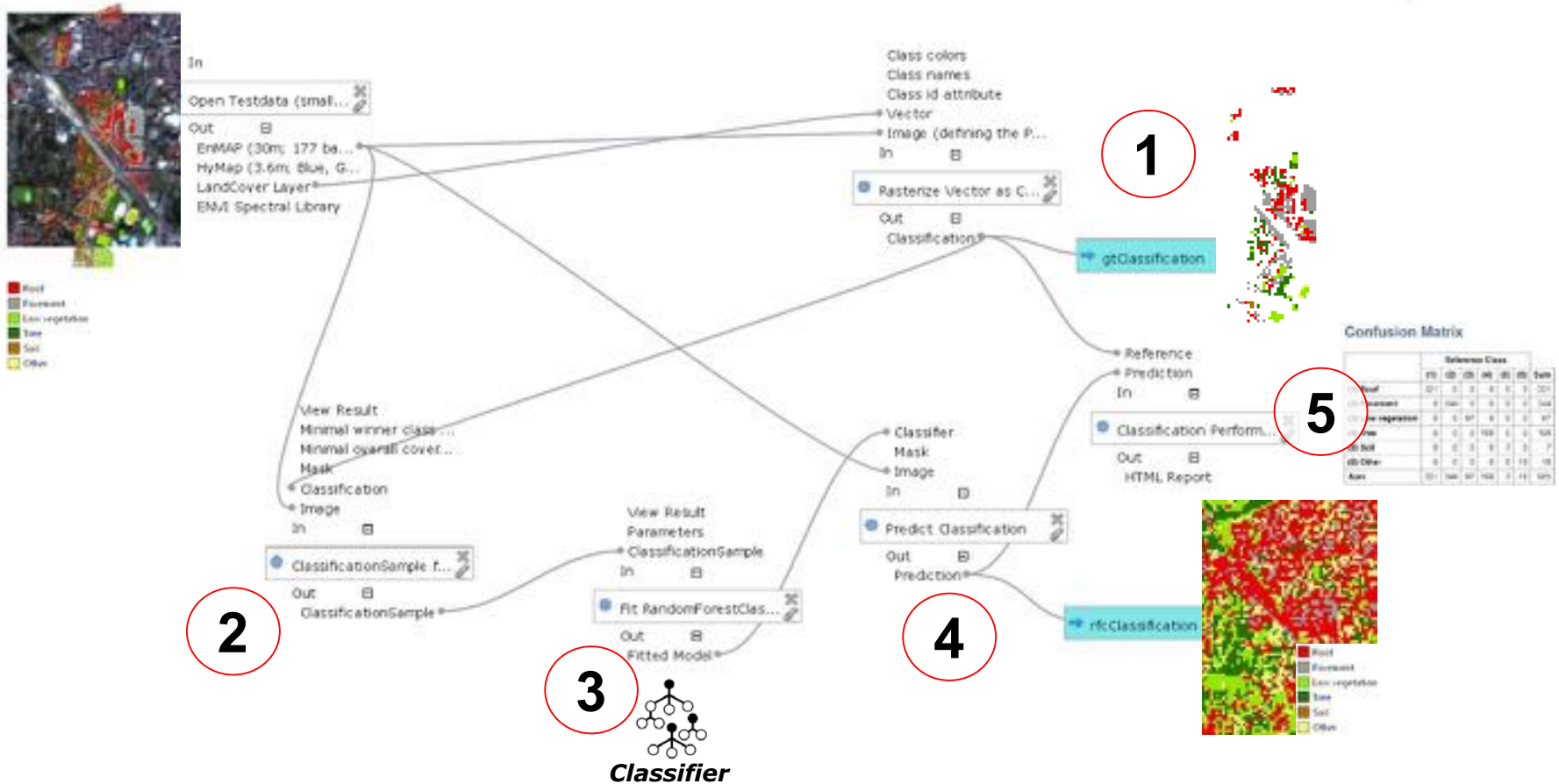
python



Scikit-Learn – Machine Learning in Python



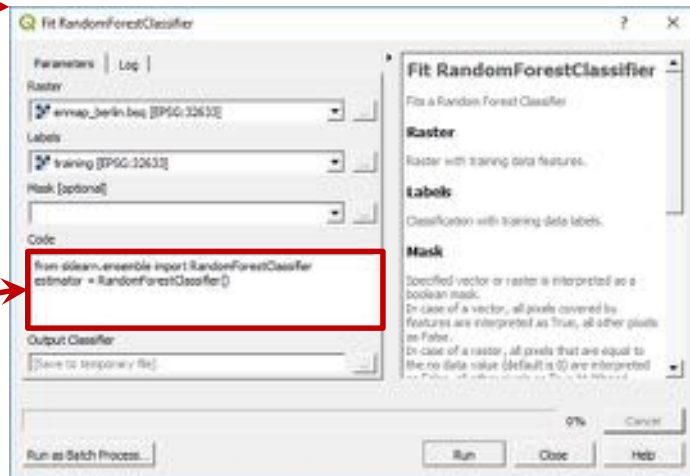
A typical Image Classification Workflow



Scikit-Learn Estimators in QGIS



- EnMAP-Box
 - Accuracy Assessment
 - Auxiliary
 - Classification
 - Fit GaussianProcessClassifier
 - Fit LinearSVC
 - Fit RandomForestClassifier**
 - Fit SVC
 - Predict Class Probability
 - Predict Classification
 - Clustering
 - Fit AffinityPropagation
 - Fit Birch
 - Fit KMeans
 - Fit MeanShift
 - Predict Clustering
 - Convolution, Morphology and Filtering
 - Create Raster
 - Create Sample
 - Masking
 - Post-Processing
 - Random
 - Regression
 - Fit GaussianProcessRegressor
 - Fit KernelRidge
 - Fit LinearRegression
 - Fit LinearSVR
 - Fit RandomForestRegressor
 - Fit SVR
 - Predict Regression
 - Resampling
 - Transformation
 - Fit FactorAnalysis
 - Fit FastICA
 - Fit FeatureAgglomeration
 - Fit Imputer
 - Fit KernelPCA
 - Fit MaxAbsScaler
 - Fit MinMaxScaler
 - Fit Normalizer
 - Fit PCA
 - Fit QuantileTransformer
 - Fit RobustScaler
 - Fit StandardScaler
 - Inverse Transform Raster
 - Transform Raster



```
RandomForestClassifier.py x  
1 from sklearn.ensemble import RandomForestClassifier  
2 estimator = RandomForestClassifier()  
3
```

- estimators
 - classifiers
 - __init__.py
 - GaussianProcessClassifier.py
 - GaussianProcessClassifierHelp.py
 - LinearSVC.py
 - LinearSVCHelp.py
 - RandomForestClassifier.py**
 - RandomForestClassifierHelp.py
 - SVC.py
 - SVCHelp.py
 - clusterers
 - regressors
 - transformers

Previous
Next
Up

scikit-learn v0.20.0

Other versions

Please [cite us](#) if you use the software.

3.2.4.3.1.

sklearn.ensemble.RandomForestClassifier

3.2.4.3.1.1. Examples using

sklearn.ensemble.RandomForestClassifier

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators='warn', criterion='gini', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,
n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None) [source]
```

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the User Guide.

Parameters: `n_estimators` : integer, optional (default=10)

The number of trees in the forest.

Changed in version 0.20: The default value of `n_estimators` will change from 10 in version 0.20 to 100 in version 0.22.

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_depth : integer or None, optional (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.

min_samples_split : int, float, optional (default=2)

The minimum number of samples required to split an internal node:

- if int, then consider `min_samples_split` as the minimum number.
- if float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : int, float, optional (default=1)

Fit RandomForestClassifier

Parameters | Log

Router

Labels

Mask (optional)

Code

```
from sklearn.ensemble import RandomForestClassifier
estimator = RandomForestClassifier()
```

Output Classifier

[Save to temporary file]

Run as Batch Process...

Run Close Help

In case of a vector, all pixels covered by features are interpreted as True, all other pixels as False.
In case of a raster, all pixels that are equal to the no-data value (default is 1) are interpreted as false, all other pixels as True. Multiband rasters are first evaluated band-wise. The final mask for a given pixel is True, if all band-wise masks for that pixel are True.

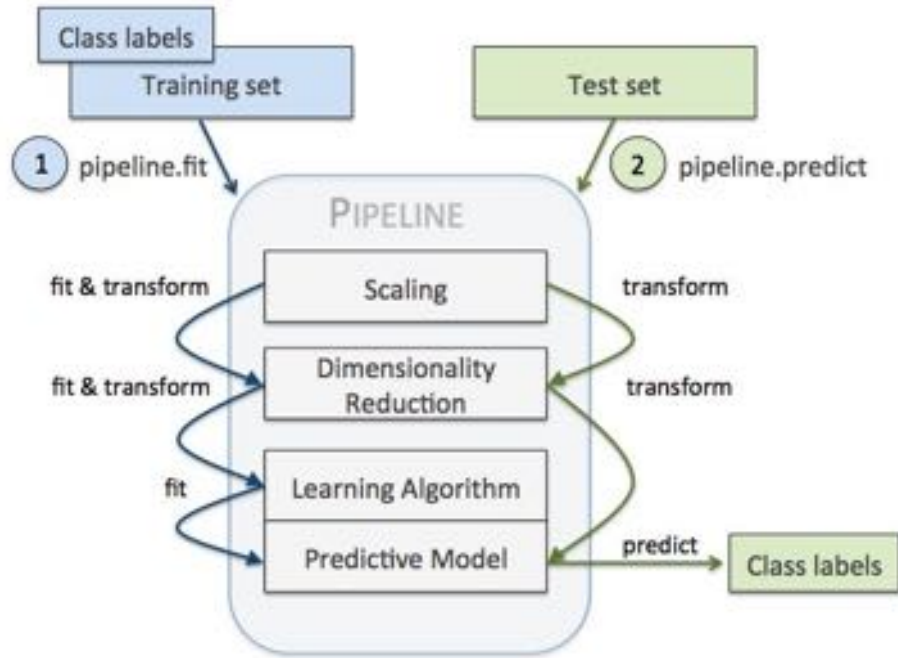
Code

```
scikit-learn python code, see RandomForestClassifier for more information on different parameters. If no parameters are provided, scikit-learn default settings will be used. Hint: you might want to alter e.g. the n_estimators value (number of trees), as the default is 10. So the line of code might be altered to 'estimator = RandomForestClassifier(n_estimators=100)'.
```

Output Classifier

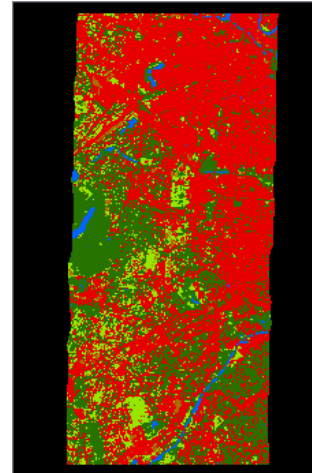
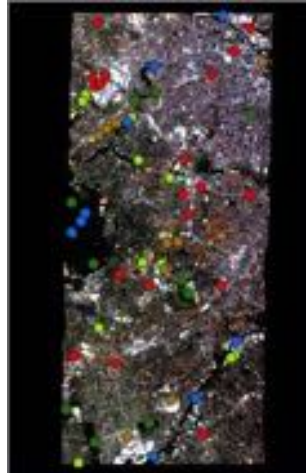
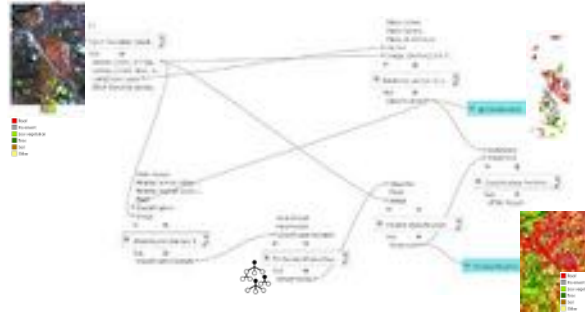
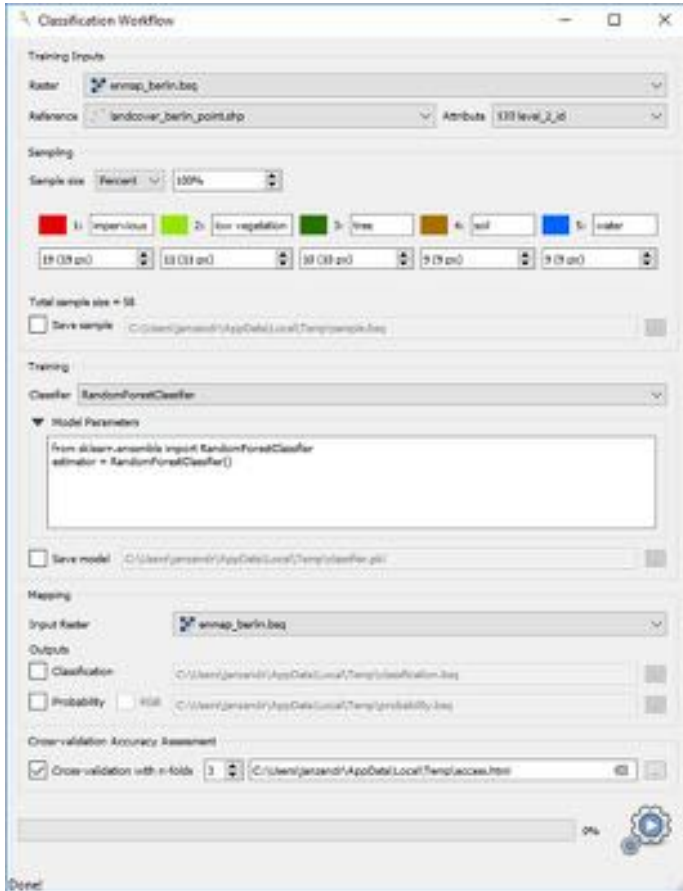
Specify output path for the classifier (.pic). This file can be used for applying the classifier to an image using 'Classification -> Predict Classifier' and 'Classification -> Predict Classification'.

Pipeline Estimators



```
SVC.py x
1 from sklearn.pipeline import make_pipeline
2 from sklearn.model_selection import GridSearchCV
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.svm import SVC
5
6 svc = SVC(probability=False)
7
8 param_grid = {'kernel': ['rbf'],
9              'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
10             'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
11
12 tunedSVC = GridSearchCV(cv=3, estimator=svc, scoring='f1_macro',
13                        param_grid=param_grid)
14
15 estimator = make_pipeline(StandardScaler(), tunedSVC)
16
```


Interactive Applications



Confusion Matrix

	Reference Class					Sum
	(1)	(2)	(3)	(4)	(5)	
(1) impervious	15	2	0	1	0	18
(2) low vegetation	1	9	0	0	0	10
(3) tree	0	0	10	0	0	10
(4) soil	3	0	0	8	0	11
(5) water	0	0	0	0	9	9
Sum	19	11	10	9	9	58

Scripting the Workflow

```
from hubflow.core import *
from sklearn.ensemble import RandomForestClassifier
import enmapboxtestdata

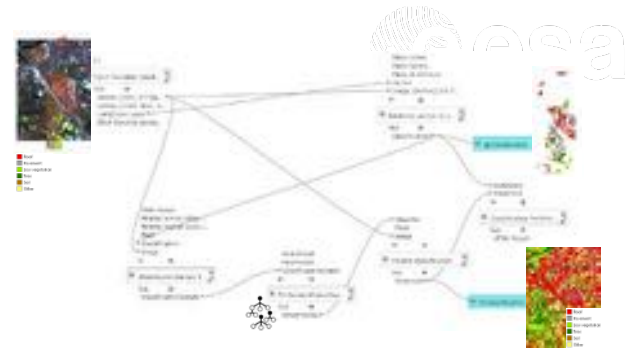
enmap = Raster(filename=enmapboxtestdata.enmap)

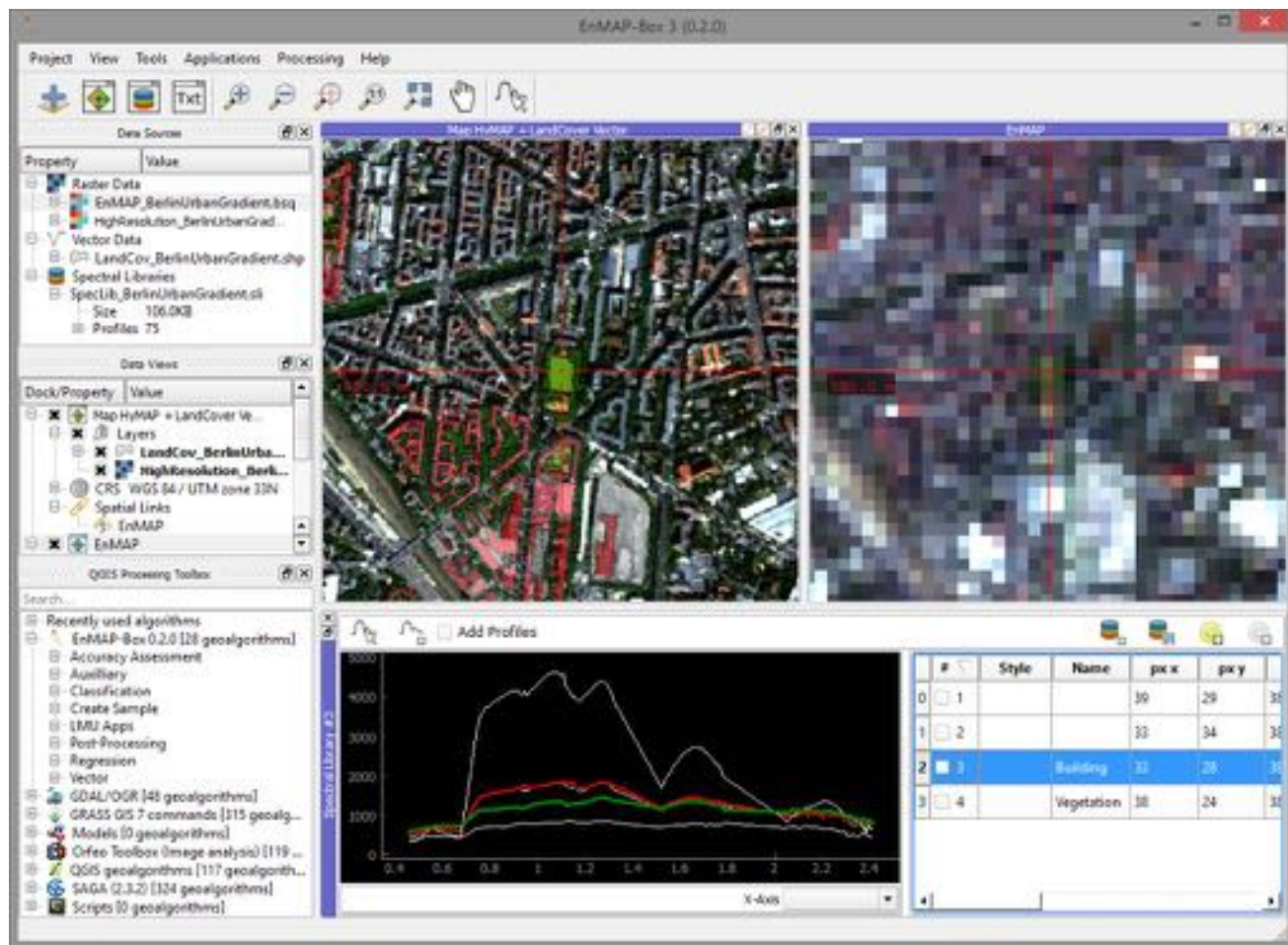
vectorClassification = VectorClassification(filename=enmapboxtestdata.landcover,
                                           classAttribute='Level_2_ID')

classification = Classification.fromClassification(classification=vectorClassification,
                                                grid=enmap.grid(),
                                                filename='/vsimem/classification.bsq')

sample = ClassificationSample(raster=enmap, classification=classification)

rfc = Classifier(sklearnEstimator=RandomForestClassifier())
rfc.fit(sample=sample)
rfc.predict(raster=enmap, filename='/vsimem/rfcClassification.bsq')
```





About EnMAP-Box | About

EnMAP-Box 3

Version 3.3.20181109T1242.devlop

Website, Source Code and more
<https://bitbucket.org/hu-geomatix/enmap-box>

Licensed under the GNU General Public Licence
<http://www.gnu.org/licenses/>

Environmental Mapping and Analysis Program (EnMAP)
<http://www.enmap.org>

The EnMAP-Box is developed at Humboldt-Universität zu Berlin under contract by the Helmholtz Centre Potsdam GFZ and is part of the EnMAP Core Science Team activities. It is funded by the German Aerospace Centre (DLR) - Project Management Agency, granted by the Federal Ministry of Economic Affairs and Energy (BMWi; grant no. 50E1529).



Read the Docs

<https://enmap-box.readthedocs.io>

Docs - User Guide

User Guide

Image Classification

In this guide you will learn how to perform a supervised image classification and calibration using the EnMAP-Box and the test dataset that comes with it. The input image is a simulated EnMAP image acquired over the city of Berlin and a detailed vector dataset with the classes Road, Pavement, Low vegetation, Tree, Soil and Other will be used as reference.

Graphical model of the classification workflow presented in the steps below. See the [Section](#) for more details on using the graphical modeler for automating EnMAP-Box workflows.

Load data / preprocessing

Bitbucket

<https://bitbucket.org>

enmap-box

The EnMAP Box is a freely available, platform-independent software designed to process hyperspectral remote sensing data, and particularly developed to handle data from the EnMAP sensor. It is distributed as Free and Open Source Software (FOSS).

Name	Size	Last commit	Message
doc		16 hours ago	avoided .Qt imports
enmapbox		2 hours ago	Std
enmapboxtesting		18 hours ago	changed Qt imports (Bitfootools)
examples		18 hours ago	changed Qt imports (Bitfootools)
make		16 hours ago	avoided .Qt imports
site-packages		4 days ago	update external
snippets		2018-10-31	gui/classification added resources
test		2018-06-22	removed pngs, icons are now ic
gitattributes	1.66 KB	2017-08-24	updated gitattributes
gitignore	385 B	2018-09-28	update gitignore
readthedocs.yml	231 B	2018-06-22	RTD
CHANGES.txt	218 B	2018-04-20	removed 'spectral' from depen

