EnMAP-Box Algorithms Documentation

Release 0.13.0dev

Andreas Rabe

Mar 02, 2020

GeoAlgorithms:

CHAPTER 1

Processing Algorithms

1.1 Accuracy Assessment

1.1.1 Classification Performance

Assesses the performance of a classification.

Parameters

Prediction **[raster]** Specify classification raster be evaluated

Reference **[raster]** Specify reference classification raster (i.e. ground truth).

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.2 Classifier Fit/Training Performance

Assesses the fit performance of a regressor using the training data.

Parameters

Classifier **[file]** Select path to a classifier file (.pkl).

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.3 Clustering Performance

Assesses the performance of a clusterer.

Parameters

Prediction **[raster]** Specify clustering raster to be evaluated.

Reference **[raster]** Specify reference clustering raster (i.e. ground truth).

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.4 Cross-validated Classifier Performance

Assesses the performance of a classifier using n-fold cross-validation.

Parameters

Classifier **[file]** Select path to a classifier file (.pkl).

Number of folds **[number]** undocumented parameter

Default: *10*

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.5 Cross-validated Regressor Performance

Assesses the performance of a regressor using n-fold cross-validation.

Parameters

Regressor **[file]** Select path to a regressor file (.pkl).

Number of folds **[number]** undocumented parameter

Default: *10*

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.6 Regression Performance

Assesses the performance of a regression.

Parameters

Prediction **[raster]** Specify regression raster to be evaluated.

Reference **[raster]** Specify reference regression raster (i.e. ground truth).

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.7 Regressor Fit/Training Performance

Assesses the fit performance of a regressor using the training data.

Parameters

Regressor **[file]** Select path to a regressor file (.pkl).

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.1.8 ROC Curve and AUC Performance

Assesses the performance of class fractions in terms of AUC and ROC curves.

Parameters

Prediction **[raster]** Specify class fraction raster to be evaluated.

Reference **[raster]** Specify reference classification raster (i.e. ground truth).

Outputs

HTML Report **[fileDestination]** Specify output path for HTML report file (.html).

Default: *outReport.html*

1.2 Auxilliary

1.2.1 Classification Statistics

This algorithm returns class count statistics. The output will be shown in the log window and can the copied from there accordingly.

Parameters

Classification **[raster]** Specify input raster.

1.2.2 Create Test Classification Map

Create a classification map at 30 m resolution by rasterizing the landcover polygons.

Parameters

Outputs

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.2.3 Create Test Classifier (RandomForest)

Create a fitted RandomForestClassifier using enmap testdata.

Parameters

Outputs

Output Classifier **[fileDestination]** Specifiy output path for the classifier (.pkl). This file can be used for applying the classifier to an image using 'Classification -> Predict Classification' and 'Classification -> Predict ClassFraction'.

Default: *outClassifier.pkl*

1.2.4 Create Test Clusterer (KMeans)

Create a fitted KMeans clusterer using enmap testdata.

Parameters

Outputs

Output Clusterer **[fileDestination]** Specifiy output path for the clusterer (.pkl). This file can be used for applying the clusterer to an image using 'Clustering -> Predict Clustering'.

Default: *outClusterer.pkl*

1.2.5 Create Test Fraction Map

Create a fraction map at 30 m resolution by rasterizing the landcover polygons.

Parameters

Outputs

Output Fraction **[rasterDestination]** Specify output path for fraction raster.

1.2.6 Create Test Regressor (RandomForest)

Create a fitted RandomForestRegressor using enmap testdata.

Parameters

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outRegressor.pkl*

1.2.7 Create Test Transformer (PCA)

Create a fitted PCA transformer using enmap testdata.

Parameters

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outTransformer.pkl*

1.2.8 Import ARTMO lookup table

Creates a raster and a regression from the profiles and biophysical parameters in the lookup table.

Parameters

ARTMO lookup table **[file]** undocumented parameter

Reflectance scale factor **[number]** Reflectance scale factor. Keep the default to have the data in the [0, 1]. Use a value of 10000 to scale the data into the [0, 10000] range.

Default: *1.0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

Output Regression **[rasterDestination]** Specify output path for regression raster.

1.2.9 Import Library

Import Library profiles as single line Raster.

Parameters

Library **[file]** Select path to an ENVI Spectral Library file (e.g. .sli or .esl).

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.2.10 Import Library Classification Attribute

Import Library classification attribute as single line Classification.

Parameters

Library **[file]** Select path to an ENVI Spectral Library file (e.g. .sli or .esl).

Classification Attribute **[string]** Attribute name as specified in the library CSV attribute file.

Outputs

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.2.11 Open Test Maps

Opens testdata into current QGIS project (LandCov_BerlinUrbanGradient.shp, HighResolution_BerlinUrbanGradient.bsq, EnMAP_BerlinUrbanGradient.bsq, SpecLib_BerlinUrbanGradient.sli).

Parameters

Outputs

EnMAP (30m; 177 bands) **[rasterDestination]** File name: EnMAP_BerlinUrbanGradient.bsq

Simulated EnMAP data (based on 3.6m HyMap imagery) acquired in August 2009 over south eastern part of Berlin covering an area of 4.32 km^2 (2.4 x 1.8 km). It has a spectral resolution of 177 bands and a spatial resolution of 30m.

HyMap (3.6m; Blue, Green, Red, NIR bands) **[rasterDestination]** File name: HighResolution_BerlinUrbanGradient.bsq

HyMap image acquired in August 2009 over south eastern part of Berlin covering an area of 4.32 km^2 (2.4 x 1.8 km). This dataset was reduced to 4 bands (0.483, 0.558, 0.646 and 0.804 micrometers). The spatial resolution is 3.6m.

LandCover Layer **[vectorDestination]** File name: LandCov_BerlinUrbanGradient.shp

Polygon shapefile containing land cover information on two classification levels. Derived from very high resolution aerial imagery and cadastral datasets.

Level 1 classes: Impervious; Other; Vegetation; Soil

Level 2 classes: Roof; Low vegetation; Other; Pavement; Tree; Soil

Library as Raster **[rasterDestination]** File name: SpecLib_BerlinUrbanGradient.sli

Spectral library with 75 spectra (material level, level 2 and level 3 class information)

1.2.12 Raster Band Statistics

This algorithm returns raster band statistics. The output will be shown in the log window and can the copied from there accordingly.

Parameters

Raster **[raster]** Specify input raster.

Band **[band]** Specify input raster band.

1.2.13 Set Raster no data value

Set the raster no data value. Note that the raster has to be re-opened.

Parameters

Raster **[raster]** Specify input raster.

No data value **[number]** Value used as the new raster no data value.

Default: *0.0*

1.2.14 Unique Values from Raster Band

This algorithm returns unique values from a raster band as a list. The output will be shown in the log window and can the copied from there accordingly.

Parameters

Raster **[raster]** Specify input raster.

Band **[band]** Specify input raster band.

1.2.15 Unique Values from Vector Attribute

This algorithm returns unique values from vector attributes as a list, which is also usable as Class Definition in other algorithms. The output will be shown in the log window and can the copied from there accordingly.

Parameters

Vector **[vector]** Specify input vector.

Field **[field]** Specify field of vector layer for which unique values should be derived.

1.2.16 View Map Metadata

Prints all Map metadata to log.

Parameters

Map **[layer]** undocumented parameter

1.3 Classification

1.3.1 Fit GaussianProcessClassifier

Fits Gaussian Process Classifier. See Gaussian Processes for further information.

See the following Cookbook Recipes on how to use classifiers: Classification , Graphical Modeler

Parameters

Raster **[raster]** Raster with training data features.

Labels **[raster]** Classification with training data labels.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See GaussianProcessClassifier for information on different parameters.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF

gpc = GaussianProcessClassifier(RBF(), max_iter_predict=1)
estimator = make_pipeline(StandardScaler(), gpc)
```

Outputs

Output Classifier **[fileDestination]** Specifiy output path for the classifier (.pkl). This file can be used for applying the classifier to an image using 'Classification -> Predict Classification' and 'Classification -> Predict ClassFraction'.

Default: *outEstimator.pkl*

1.3.2 Fit LinearSVC

Fits a linear Support Vector Classification. Input data will be scaled and grid search is used for model selection.

See the following Cookbook Recipes on how to use classifiers: Classification , Graphical Modeler

Parameters

Raster **[raster]** Raster with training data features.

Labels **[raster]** Classification with training data labels.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. For information on different parameters have a look at LinearSVC. See GridSearchCV for information on grid search and StandardScaler for scaling.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

svc = LinearSVC()
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
tunedSVC = GridSearchCV(cv=3, estimator=svc, scoring='f1_macro', param_grid=param_
↪grid)
estimator = make_pipeline(StandardScaler(), tunedSVC)
```

Outputs

Output Classifier **[fileDestination]** Specifiy output path for the classifier (.pkl). This file can be used for applying the classifier to an image using 'Classification -> Predict Classification' and 'Classification -> Predict ClassFraction'.

Default: *outEstimator.pkl*

1.3.3 Fit RandomForestClassifier

Fits a Random Forest Classifier

See the following Cookbook Recipes on how to use classifiers: Classification , Graphical Modeler

Parameters

Raster **[raster]** Raster with training data features.

Labels **[raster]** Classification with training data labels.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See RandomForestClassifier for information on different parameters. If this code is not altered, 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).'

Default:

```
from sklearn.ensemble import RandomForestClassifier
estimator = RandomForestClassifier(n_estimators=100, oob_score=True)
```

Outputs

Output Classifier **[fileDestination]** Specifiy output path for the classifier (.pkl). This file can be used for applying the classifier to an image using 'Classification -> Predict Classification' and 'Classification -> Predict ClassFraction'.

Default: *outEstimator.pkl*

1.3.4 Fit SVC

Fits a Support Vector Classification. Input data will be scaled and grid search is used for model selection.

See the following Cookbook Recipes on how to use classifiers: Classification , Graphical Modeler

Parameters

Raster **[raster]** Raster with training data features.

Labels **[raster]** Classification with training data labels.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. For information on different parameters have a look at SVC. See GridSearchCV for information on grid search and StandardScaler for scaling.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

svc = SVC(probability=False)
param_grid = {'kernel': ['rbf'],
              'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
              'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
tunedSVC = GridSearchCV(cv=3, estimator=svc, scoring='f1_macro', param_grid=param_
↪grid)
estimator = make_pipeline(StandardScaler(), tunedSVC)
```

Outputs

Output Classifier **[fileDestination]** Specifiy output path for the classifier (.pkl). This file can be used for applying the classifier to an image using 'Classification -> Predict Classification' and 'Classification -> Predict ClassFraction'.

Default: *outEstimator.pkl*

1.3.5 Predict Class Probability

Applies a classifier to a raster.

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Classifier **[file]** Select path to a classifier file (.pkl).

Outputs

Probability **[rasterDestination]** Specify output path for fraction raster.

1.3.6 Predict Classification

Applies a classifier to a raster.

Used in the Cookbook Recipes: Classification , Graphical Modeler

Parameters

Raster **[raster]** Select raster file which should be classified.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Classifier **[file]** Select path to a classifier file (.pkl).

Outputs

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.4 Clustering

1.4.1 Fit AffinityPropagation

Fits a Affinity Propagation clusterer (input data will be scaled).

See the following Cookbook Recipes on how to use clusterers: Clustering

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. For information on different parameters have a look at AffinityPropagation. See StandardScaler for information on scaling

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AffinityPropagation

clusterer = AffinityPropagation()
estimator = make_pipeline(StandardScaler(), clusterer)
```

Outputs

Output Clusterer **[fileDestination]** Specifiy output path for the clusterer (.pkl). This file can be used for applying the clusterer to an image using 'Clustering -> Predict Clustering'.

Default: *outEstimator.pkl*

1.4.2 Fit Birch

Fits a Birch clusterer (input data will be scaled).

See the following Cookbook Recipes on how to use clusterers: Clustering

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. For information on different parameters have a look at Birch. See StandardScaler for information on scaling

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import Birch

clusterer = Birch()
estimator = make_pipeline(StandardScaler(), clusterer)
```

Outputs

Output Clusterer **[fileDestination]** Specifiy output path for the clusterer (.pkl). This file can be used for applying the clusterer to an image using 'Clustering -> Predict Clustering'.

Default: *outEstimator.pkl*

1.4.3 Fit KMeans

Fits a KMeans clusterer (input data will be scaled).

See the following Cookbook Recipes on how to use clusterers: Clustering

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. For information on different parameters have a look at KMeans. See StandardScaler for information on scaling

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

clusterer = KMeans()
estimator = make_pipeline(StandardScaler(), clusterer)
```

Outputs

Output Clusterer **[fileDestination]** Specifiy output path for the clusterer (.pkl). This file can be used for applying the clusterer to an image using 'Clustering -> Predict Clustering'.

Default: *outEstimator.pkl*

1.4.4 Fit MeanShift

Fits a MeanShift clusterer (input data will be scaled).

See the following Cookbook Recipes on how to use clusterers: Clustering

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. For information on different parameters have a look at MeanShift. See StandardScaler for information on scaling

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import MeanShift

clusterer = MeanShift()
estimator = make_pipeline(StandardScaler(), clusterer)
```

Outputs

Output Clusterer **[fileDestination]** Specifiy output path for the clusterer (.pkl). This file can be used for applying the clusterer to an image using 'Clustering -> Predict Clustering'.

Default: *outEstimator.pkl*

1.4.5 Predict Clustering

Applies a clusterer to a raster.

Used in the Cookbook Recipes: Clustering

Parameters

Raster **[raster]** Select raster file which should be clustered.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Clusterer **[file]** Select path to a clusterer file (.pkl).

Outputs

Clustering **[rasterDestination]** Specify output path for classification raster.

1.5 Convolution, Morphology and Filtering

1.5.1 Spatial Gaussian Gradient Magnitude

Applies gaussian_gradient_magnitude filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.gaussian_gradient_magnitude for information on different parameters.

Default:

```
from scipy.ndimage.filters import gaussian_gradient_magnitude

function = lambda array: gaussian_gradient_magnitude(array, sigma=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.2 Spatial Generic Filter

Applies generic_filter to image using a user-specifiable function. This algorithm can perform operations you might know as moving window or focal statistics from some other GIS systems. Mind that depending on the function this algorithms can take some time to process.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. The function argument can take any callable function that expects a 1D array as input and returns a single value. You should alter the preset in the code window and define your own function. See scipy.ndimage.generic_filter for information on different parameters.

Default:

```
from scipy.ndimage.filters import generic_filter
import numpy as np

def filter_function(invalues):

    # do whatever you want to create the output value, e.g. np.nansum
    # outvalue = np.nansum(invalues)
    return outvalue

function = lambda array: generic_filter(array, function=filter_function, size=3)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.3 Spatial Laplace

Applies laplace filter to image. See Wikipedia for more information on laplace filtering.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.laplace for information on different parameters.

Default:

```
from scipy.ndimage.filters import laplace

function = lambda array: laplace(array)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.4 Spatial Maximum Filter

Applies maximum filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.maximum for information on different parameters.

Default:

```
from scipy.ndimage.filters import maximum_filter

function = lambda array: maximum_filter(array, size=3)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.5 Spatial Median Filter

Applies median filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.median for information on different parameters.

Default:

```
from scipy.ndimage.filters import median_filter

function = lambda array: median_filter(array, size=3)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.6 Spatial Minimum Filter

Applies minimum filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.minimum for information on different parameters.

Default:

```
from scipy.ndimage.filters import minimum_filter

function = lambda array: minimum_filter(array, size=3)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.7 Spatial Percentile Filter

Applies percentile filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.percentile_filter for information on different parameters.

Default:

```
from scipy.ndimage.filters import percentile_filter

function = lambda array: percentile_filter(array, percentile=50, size=3)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.8 Spatial Prewitt

Applies prewitt filter to image. See Wikipedia for further information on prewitt operators.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.prewitt for information on different parameters.

Default:

```
from scipy.ndimage.filters import prewitt

function = lambda array: prewitt(array, axis=0)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.9 Spatial Sobel

Applies sobel filter to image. See Wikipedia for further information on sobel operators

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.sobel for information on different parameters.

Default:

```
from scipy.ndimage.filters import sobel

function = lambda array: sobel(array, axis=0)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.10 Spatial Convolution AiryDisk2DKernel

Applies AiryDisk2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.AiryDisk2DKernel for information on different parameters.

Default:

```
from astropy.convolution import AiryDisk2DKernel

kernel = AiryDisk2DKernel(radius=5)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.11 Spatial Convolution Box2DKernel

Applies Box2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Box2DKernel for information on different parameters.

Default:

```
from astropy.convolution import Box2DKernel

kernel = Box2DKernel(width=5)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.12 Spatial Convolution Gaussian2DKernel

Applies Gaussian2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Gaussian2DKernel for information on different parameters.

Default:

```
from astropy.convolution import Gaussian2DKernel

kernel = Gaussian2DKernel(x_stddev=1, y_stddev=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.13 Spatial Convolution HighPass2DKernel

Applies a 3x3 High-Pass kernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code.

Default:

```
from astropy.convolution import Kernel2D

kernel = Kernel2D(array=
  [[-1, -1, -1],
   [-1,  8, -1],
   [-1, -1, -1]])
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.14 Spatial Convolution MexicanHat2DKernel

Applies MexicanHat2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.MexicanHat2DKernel for information on different parameters.

Default:

```
from astropy.convolution import MexicanHat2DKernel

kernel = MexicanHat2DKernel(width=5)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.15 Spatial Convolution Moffat2DKernel

Applies Moffat2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Moffat2DKernel for information on different parameters.

Default:

```
from astropy.convolution import Moffat2DKernel

kernel = Moffat2DKernel(gamma=3, alpha=2)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.16 Spatial Convolution Ring2DKernel

Applies Ring2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Ring2DKernel for information on different parameters.

Default:

```
from astropy.convolution import Ring2DKernel

kernel = Ring2DKernel(radius_in=3, width=2)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.17 Spatial Convolution Tophat2DKernel

Applies Tophat2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Tophat2DKernel for information on different parameters.

Default:

```
from astropy.convolution import Tophat2DKernel

kernel = Tophat2DKernel(radius=5)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.18 Spatial Convolution TrapezoidDisk2DKernel

Applies TrapezoidDisk2DKernel to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.TrapezoidDisk2DKernel for information on different parameters.

Default:

```
from astropy.convolution import TrapezoidDisk2DKernel

kernel = TrapezoidDisk2DKernel(radius=3, slope=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.19 Spatial Morphological Binary Closing

Applies binary_closing morphology filter to image. See Wikipedia for general information about closing morphology

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.binary_closing for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import binary_closing, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: binary_closing(array, structure=structure, iterations=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.20 Spatial Morphological Binary Dilation

Applies binary_dilation morphology filter to image. See Wikipedia for general information about dilation morphology

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.binary_dilation for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import binary_dilation, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: binary_dilation(array, structure=structure, iterations=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.21 Spatial Morphological Binary Erosion

Applies binary_erosion morphology filter to image. See Wikipedia for general information about erosion morphology

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.binary_erosion for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import binary_erosion, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: binary_erosion(array, structure=structure, iterations=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.22 Spatial Morphological Binary Fill Holes

Applies binary_fill_holes morphology filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.binary_fill_holes for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import binary_fill_holes, generate_binary_structure,
↪ iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: binary_fill_holes(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.23 Spatial Morphological Binary Opening

Applies binary_opening morphology filter to image. See Wikipedia for general information about opening morphology

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.binary_opening for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import binary_opening, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: binary_opening(array, structure=structure, iterations=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.24 Spatial Morphological Binary Propagation

Applies binary_propagation morphology filter to image.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.binary_propagation for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import binary_propagation, generate_binary_
↪structure, iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: binary_propagation(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.25 Spatial Morphological Black Tophat

Applies black_tophat morphology filter to image. See Wikipedia for more information on top-hat transformation.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.black_tophat for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import black_tophat, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: black_tophat(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.26 Spatial Morphological Gradient

Applies morphological_gradient morphology filter to image. See Wikipedia for more information about morphological gradients.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.morphological_gradient for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import morphological_gradient, generate_binary_
↪structure, iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: morphological_gradient(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.27 Spatial Morphological Grey Closing

Applies grey_closing morphology filter to image. See Wikipedia for general information about closing morphology.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.grey_closing for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import grey_closing, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: grey_closing(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.28 Spatial Morphological Grey Dilation

Applies grey_dilation morphology filter to image. See Wikipedia for general information about closing morphology.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.grey_dilation for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import grey_dilation, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: grey_dilation(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.29 Spatial Morphological Grey Erosion

Applies grey_erosion morphology filter to image. See Wikipedia for general information about erosion morphology

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.grey_erosion for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import grey_erosion, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: grey_erosion(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.30 Spatial Morphological Grey Opening

Applies grey_opening morphology filter to image. See Wikipedia for general information about opening morphology

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.grey_opening for information on different parameters. At first, the structuring element will be defined (scipy.ndimage.generate_binary_structure). By default, its dimensions are always equal to 3. The connectivity parameter defines the type of neighborhood. In order create a bigger structuring element, the parameters in iterate_structure have to be altered (e.g. iterations=2 will increase the size to 5). Alternatively, a custom numpy array can be used as structural element.

Default:

```
from scipy.ndimage.morphology import grey_opening, generate_binary_structure,␣
↪iterate_structure

structure = generate_binary_structure(rank=2, connectivity=1)
structure = iterate_structure(structure=structure, iterations=1)
function = lambda array: grey_opening(array, structure=structure)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.31 Spatial Morphological Laplace

Applies morphological_laplace filter to image. See Wikipedia for more information on laplace filtering.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.morphological_laplace for information on different parameters.

Default:

```
from scipy.ndimage.morphology import morphological_laplace
function = lambda array: morphological_laplace(array, size=(3,3))
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.32 Spatial Morphological White Tophat

Applies white_tophat morphology filter to image. See Wikipedia for more information on top-hat transformation.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.ndimage.white_tophat for information on different parameters.

Default:

```
from scipy.ndimage.morphology import white_tophat
function = lambda array: white_tophat(array, size=(3,3))
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.33 Spectral Convolution Box1DKernel

Applies Box1DKernel.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Box1DKernel for information on different parameters.

Default:

```
from astropy.convolution import Box1DKernel

kernel = Box1DKernel(width=5)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.34 Spectral Convolution Gaussian1DKernel

Applies Gaussian1DKernel.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Gaussian1DKernel for information on different parameters.

Default:

```
from astropy.convolution import Gaussian1DKernel

kernel = Gaussian1DKernel(stddev=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.35 Spectral Convolution MexicanHat1DKernel

Applies MexicanHat1DKernel.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.MexicanHat1DKernel for information on different parameters.

Default:

```
from astropy.convolution import MexicanHat1DKernel

kernel = MexicanHat1DKernel(width=10)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.36 Spectral Convolution SavitzkyGolay1DKernel

Applies Savitzki Golay Filter.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See scipy.signal.savgol_coeffs for information on different parameters.

Default:

```
from astropy.convolution import Kernel1D
from scipy.signal import savgol_coeffs

kernel = Kernel1D(array=savgol_coeffs(window_length=11, polyorder=3, deriv=0))
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.5.37 Spectral Convolution Trapezoid1DKernel

Applies Trapezoid1DKernel.

See the following Cookbook Recipes on how to apply filters: Filtering , Generic Filter

Parameters

Raster **[raster]** Specify input raster.

Code **[string]** Python code. See astropy.convolution.Trapezoid1DKernel for information on different parameters.

Default:

```
from astropy.convolution import Trapezoid1DKernel

kernel = Trapezoid1DKernel(width=5, slope=1)
```

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.6 Create Raster

1.6.1 Classification from Fraction

Creates classification from class fraction. Winner class is equal to the class with maximum class fraction.

Parameters

ClassFraction **[raster]** Specify input raster.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Default: *0.5*

Minimal dominant coverage **[number]** Mask out all pixels that have a coverage of the predominant class less than the specified value. This controls pixel purity.

Default: *0.5*

Outputs

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.6.2 Classification from Vector

Creates a classification from a vector field with class ids.

Used in the Cookbook Recipes: Classification , Graphical Modeler

Parameters

Pixel Grid **[raster]** Specify input raster.

Vector **[vector]** Specify input vector.

Class id attribute **[field]** Vector field specifying the class ids.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Minimal dominant coverage **[number]** Mask out all pixels that have a coverage of the predominant class less than the specified value. This controls pixel purity.

Oversampling factor **[number]** Defines the degree of detail by which the class information given by the vector is rasterized. An oversampling factor of 1 (default) simply rasterizes the vector on the target pixel grid.An oversampling factor of 2 will rasterize the vector on a target pixel grid with resolution twice as fine.An oversampling factor of 3 will rasterize the vector on a target pixel grid with resolution three times as fine, ... and so on.

Mind that larger values are always better (more accurate), but depending on the inputs, this process can be quite computationally intensive, when a higher factor than 1 is used.

Default: *1*

Outputs

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.6.3 Fraction from Classification

Derive (binarized) class fractions from a classification.

Parameters

Classification **[raster]** Specify input raster.

Outputs

Output Fraction **[rasterDestination]** Specify output path for fraction raster.

1.6.4 Fraction from Vector

Derives class fraction raster from a vector file with sufficient class information. Note: rasterization of complex multipart vector geometries can be very slow, use "QGIS > Vector > Geometry Tools > Multiparts to Singleparts..." in this case beforehand.

Parameters

Pixel Grid **[raster]** Specify input raster.

Vector **[vector]** Specify input vector.

Class id attribute **[field]** Vector field specifying the class ids.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Default: *0.5*

Oversampling factor **[number]** Defines the degree of detail by which the class information given by the vector is rasterized. An oversampling factor of 1 (default) simply rasterizes the vector on the target pixel grid.An oversampling factor of 2 will rasterize the vector on a target pixel grid with resolution twice as fine.An oversampling factor of 3 will rasterize the vector on a target pixel grid with resolution three times as fine, ... and so on.

Mind that larger values are always better (more accurate), but depending on the inputs, this process can be quite computationally intensive, when a higher factor than 1 is used.

Default: *5*

Outputs

Output Fraction **[rasterDestination]** Specify output path for fraction raster.

1.6.5 Raster from Vector

Converts vector to raster (using gdal rasterize).

Parameters

Pixel Grid **[raster]** Specify input raster.

Vector **[vector]** Specify input vector.

Init Value **[number]** Pre-initialization value for the output raster before burning. Note that this value is not marked as the nodata value in the output raster.

Default: *0*

Burn Value **[number]** Fixed value to burn into each pixel, which is covered by a feature (point, line or polygon).

Default: *1*

Burn Attribute **[field]** Specify numeric vector field to use as burn values.

All touched **[boolean]** Enables the ALL_TOUCHED rasterization option so that all pixels touched by lines or polygons will be updated, not just those on the line render path, or whose center point is within the polygon.

Default: *False*

Filter SQL **[string]** Create SQL based feature selection, so that only selected features will be used for burning.

Example: Level_2 = 'Roof' will only burn geometries where the Level_2 attribute value is equal to 'Roof', others will be ignored. This allows you to subset the vector dataset on-the-fly.

Default: **

Data Type **[enum]** Specify output datatype.

Default: *7*

No Data Value **[string]** Specify output no data value.

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.6.6 Regression from Vector

Creates a regression from a vector field with target values.

Parameters

Pixel Grid **[raster]** Specify input raster.

Vector **[vector]** Specify input vector.

Regression value attribute **[field]** Vector field specifying the regression values.

No Data Value **[string]** Specify output no data value.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Oversampling factor **[number]** Defines the degree of detail by which the class information given by the vector is rasterized. An oversampling factor of 1 (default) simply rasterizes the vector on the target pixel grid.An oversampling factor of 2 will rasterize the vector on a target pixel grid with resolution twice as fine.An oversampling factor of 3 will rasterize the vector on a target pixel grid with resolution three times as fine, ... and so on.

Mind that larger values are always better (more accurate), but depending on the inputs, this process can be quite computationally intensive, when a higher factor than 1 is used.

Default: *1*

Outputs

Output Regression **[rasterDestination]** Specify output path for regression raster.

1.7 Create Sample

1.7.1 Extract classification samples from raster and classification

Extract classification samples from raster and classification.

Parameters

Raster **[raster]** Specify input raster.

Classification **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.7.2 Extract fraction samples from raster and fraction

Extract fraction samples from raster and fraction.

Parameters

Raster **[raster]** Specify input raster.

ClassFraction **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

Output Fraction **[rasterDestination]** Specify output path for fraction raster.

1.7.3 Extract ordination sample

Extract a regression samples where the regression labels are ordinated. See http://dx.doi.org/10.1111/avsc.12115 for details.

Parameters

Raster **[raster]** Specify input raster.

Vector **[vector]** Specify input vector.

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

Output Regression **[rasterDestination]** Specify output path for regression raster.

Vector for DataPlotly **[vectorDestination]** Specify output path for the vector.

1.7.4 Extract regression samples from raster and regression

Extract regression samples from raster and regression.

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

Output Regression **[rasterDestination]** Specify output path for regression raster.

1.7.5 Extract samples from raster and mask

Extract samples from raster and mask.

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.8 Masking

1.8.1 Apply Mask to Raster

Pixels that are masked out are set to the raster no data value.

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Outputs

Masked Raster **[rasterDestination]** Specify output path for raster.

1.8.2 Build Mask from Raster

Builds a mask from a raster based on user defined values and value ranges.

Parameters

Raster **[raster]** Specify input raster.

Foreground values **[string]** List of values and ranges that are mapped to True, e.g. [1, 2, 5, range(5, 10)].

Default: *[]*

Background values **[string]** List of values and ranges that are mapped to False, e.g. [-9999, range(-10, 0)].

Default: *[]*

Outputs

Output Mask **[rasterDestination]** Specify output path for mask raster.

1.9 Post-Processing

1.9.1 Fraction as RGB Raster

Creates a RGB representation from given class fractions. The RGB color of a specific pixel is the weighted mean value of the original class colors, where the weights are given by the corresponding class propability.

Parameters

ClassFraction **[raster]** Specify input raster.

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.10 Random

1.10.1 Random Points from Classification

Randomly samples a user defined amount of points/pixels from a classification raster and returns them as a vector dataset.

Parameters

Classification **[raster]** Specify input raster.

Number of Points per Class **[string]** List of number of points, given as integers or fractions between 0 and 1, to sample from each class. If a scalar is specified, the value is broadcasted to all classes.

Default: *100*

Outputs

Output Vector **[vectorDestination]** Specify output path for the vector.

1.10.2 Random Points from Mask

Randomly draws defined number of points from Mask and returns them as vector dataset.

Parameters

Mask **[raster]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Invert Mask **[boolean]** Whether or not to invert the selected mask.

Default: *0*

Number of Points **[string]** Number of points, given as integer or fraction between 0 and 1, to sample from the mask.

Default: *100*

Outputs

Output Vector **[vectorDestination]** Specify output path for the vector.

1.11 Regression

1.11.1 Fit GaussianProcessRegressor

Fits Gaussian Process Regression. See Gaussian Processes for further information.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See GaussianProcessRegressor for information on different parameters.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF

gpr = GaussianProcessRegressor(RBF())
scaledGPR = make_pipeline(StandardScaler(), gpr)
estimator = scaledGPR
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.2 Fit KernelRidge

Fits a KernelRidge Regression. Click here for additional information.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See KernelRidge for information on different parameters. See GridSearchCV for information on grid search and StandardScaler for scaling.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.kernel_ridge import KernelRidge

krr = KernelRidge()
param_grid = {'kernel': ['rbf'],
              'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
              'alpha': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
tunedKRR = GridSearchCV(cv=3, estimator=krr, scoring='neg_mean_absolute_error',␣
↪param_grid=param_grid)
scaledAndTunedKRR = make_pipeline(StandardScaler(), tunedKRR)
estimator = scaledAndTunedKRR
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.3 Fit LinearRegression

Fits a Linear Regression.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See LinearRegression for information on different parameters. See StandardScaler for information on scaling.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

linearRegression = LinearRegression()
estimator = make_pipeline(StandardScaler(), linearRegression)
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.4 Fit LinearSVR

Fits a Linear Support Vector Regression.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See LinearSVR for information on different parameters. See GridSearchCV for information on grid search and StandardScaler for scaling.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVR

svr = LinearSVR()
param_grid = {'epsilon' : [0.], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
tunedSVR = GridSearchCV(cv=3, estimator=svr, scoring='neg_mean_absolute_error',␣
↪param_grid=param_grid)
scaledAndTunedSVR = make_pipeline(StandardScaler(), tunedSVR)
estimator = MultiOutputRegressor(scaledAndTunedSVR)
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.5 Fit PLSRegression

Fits a Partial Least Squares Regression.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See PLSRegression for information on different parameters.

Default:

```
from sklearn.cross_decomposition import PLSRegression
estimator = PLSRegression(n_components=3, scale=True)
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.6 Fit RandomForestRegressor

Fits a Random Forest Regression.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See RandomForestRegressor for information on different parameters.

Default:

```
from sklearn.ensemble import RandomForestRegressor
estimator = RandomForestRegressor(n_estimators=100, oob_score=True)
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.7 Fit SVR

Fits a Support Vector Regression.

See the following Cookbook Recipes on how to use regressors: Regression

Parameters

Raster **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See SVR for information on different parameters. See GridSearchCV for information on grid search and StandardScaler for scaling.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR

svr = SVR()
param_grid = {'epsilon' : [0.], 'kernel' : ['rbf'],
              'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
              'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
tunedSVR = GridSearchCV(cv=3, estimator=svr, scoring='neg_mean_absolute_error',␣
↪param_grid=param_grid)
scaledAndTunedSVR = make_pipeline(StandardScaler(), tunedSVR)
estimator = MultiOutputRegressor(scaledAndTunedSVR)
```

Outputs

Output Regressor **[fileDestination]** Specifiy output path for the regressor (.pkl). This file can be used for applying the regressor to an image using 'Regression -> Predict Regression'.

Default: *outEstimator.pkl*

1.11.8 Predict Regression

Applies a regressor to an raster.

Used in the Cookbook Recipes: Regression

Parameters

Raster **[raster]** Select raster file which should be regressed.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Regressor **[file]** Select path to a regressor file (.pkl).

Outputs

Output Regression **[rasterDestination]** Specify output path for regression raster.

1.12 Resampling and Subsetting

1.12.1 Spatial Resampling (Classification)

Resamples a Classification into a target grid.

Parameters

Pixel Grid **[raster]** Specify input raster.

Classification **[raster]** Specify input raster.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Default: *0.5*

Minimal dominant coverage **[number]** Mask out all pixels that have a coverage of the predominant class less than the specified value. This controls pixel purity.

Default: *0.5*

Outputs

Output Classification **[rasterDestination]** Specify output path for classification raster.

1.12.2 Spatial Resampling (Fraction)

Resamples a Fraction into a target grid.

Parameters

Pixel Grid **[raster]** Specify input raster.

ClassFraction **[raster]** Specify input raster.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Default: *0.5*

Outputs

Output Fraction **[rasterDestination]** Specify output path for fraction raster.

1.12.3 Spatial Resampling (Mask)

Resamples a Mask into a target grid.

Parameters

Pixel Grid **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Default: *0.5*

Outputs

Output Mask **[rasterDestination]** Specify output path for mask raster.

1.12.4 Spatial Resampling (Raster)

Resamples a Raster into a target grid.

Parameters

Pixel Grid **[raster]** Specify input raster.

Raster **[raster]** Specify input raster.

Resampling Algorithm **[enum]** Specify resampling algorithm.

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.12.5 Spatial Resampling (Regression)

Resamples a Regression into a target grid.

Parameters

Pixel Grid **[raster]** Specify input raster.

Regression **[raster]** Specify input raster.

Minimal overall coverage **[number]** Mask out all pixels that have an overall coverage less than the specified value. This controls how edges between labeled and no data regions are treated.

Default: *0.5*

Outputs

Output Regression **[rasterDestination]** Specify output path for regression raster.

1.12.6 Spectral Resampling

Spectrally resample a raster.

Parameters

Raster **[raster]** Select raster file which should be resampled.

[Options 1] Spectral characteristic from predefined sensor **[enum]** undocumented parameter

[Option 2] Spectral characteristic from Raster **[raster]** Raster with defined wavelength and fwhm

[Option 3] Spectral characteristic from response function files. **[file]** Select path to an ENVI Spectral Library file (e.g. .sli or .esl).

Resampling Algorithm **[enum]** undocumented parameter

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.12.7 Subset Raster Bands

Subset raster bands by the given list of band numbers.

Parameters

Raster **[raster]** Specify input raster.

Band subset **[string]** List of bands to subset. E.g. 1, 2, -1 will select the first, the second and the last band.

Invert list **[boolean]** Wether to invert the list of bands. E.g. 1, 2, -1 will selecting all bands but the first, the second and the last.

Default: *0*

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.12.8 Subset Raster Wavebands

Subset raster bands that best match the given wavelength.

Parameters

Raster **[raster]** Specify input raster.

Wavelength **[raster]** Specify input raster.

Outputs

Output Raster **[rasterDestination]** Specify output path for raster.

1.13 Transformation

1.13.1 Fit FactorAnalysis

Fits a Factor Analysis.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See FactorAnalysis for information on different parameters.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import FactorAnalysis

factorAnalysis = FactorAnalysis(n_components=3)
estimator = make_pipeline(StandardScaler(), factorAnalysis)
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.2 Fit FastICA

Fits a FastICA (Independent Component Analysis).

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See FastICA for information on different parameters.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import FastICA

fastICA = FastICA(n_components=3)
estimator = make_pipeline(StandardScaler(), fastICA)
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.3 Fit FeatureAgglomeration

Fits a Feature Agglomeration.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See FeatureAgglomeration for information on different parameters.

Default:

```
from sklearn.cluster import FeatureAgglomeration

estimator = FeatureAgglomeration(n_clusters=3)
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.4 Fit Imputer

Fits an Imputer (Imputation transformer for completing missing values).

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See Imputer for information on different parameters.

Default:

```
from sklearn.preprocessing import Imputer

estimator = Imputer()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.5 Fit KernelPCA

Fits a Kernel PCA (Principal Component Analysis).

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See KernelPCA for information on different parameters.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import KernelPCA

kernelPCA = KernelPCA(n_components=3, fit_inverse_transform=True)
estimator = make_pipeline(StandardScaler(), kernelPCA)
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.6 Fit MaxAbsScaler

Fits a MaxAbsScaler (scale each feature by its maximum absolute value). See also examples for different scaling methods.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See MaxAbsScaler for information on different parameters.

Default:

```
from sklearn.preprocessing import MaxAbsScaler

estimator = MaxAbsScaler()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.7 Fit MinMaxScaler

Fits a MinMaxScaler (transforms features by scaling each feature to a given range). See also examples for different scaling methods.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See MinMaxScaler for information on different parameters.

Default:

```
from sklearn.preprocessing import MinMaxScaler

estimator = MinMaxScaler()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.8 Fit Normalizer

Fits a Normalizer (normalizes samples individually to unit norm). See also examples for different scaling methods.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See Normalizer for information on different parameters.

Default:

```
from sklearn.preprocessing import Normalizer

estimator = Normalizer()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.9 Fit PCA

Fits a PCA (Principal Component Analysis).

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See PCA for information on different parameters.

Default:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

pca = PCA()
estimator = make_pipeline(StandardScaler(), pca)
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.10 Fit QuantileTransformer

Fits a Quantile Transformer (transforms features using quantiles information). See also examples for different scaling methods

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See quantile_transform for information on different parameters.

Default:

```
from sklearn.preprocessing import QuantileTransformer

estimator = QuantileTransformer()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.11 Fit RobustScaler

Fits a Robust Scaler (scales features using statistics that are robust to outliers). Click here for example. See also examples for different scaling methods.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See RobustScaler for information on different parameters.

Default:

```
from sklearn.preprocessing import RobustScaler

estimator = RobustScaler()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.12 Fit StandardScaler

Fits a Standard Scaler (standardizes features by removing the mean and scaling to unit variance). See also examples for different scaling methods.

See the following Cookbook Recipes on how to use transformers: Transformation

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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 **[string]** Scikit-learn python code. See StandardScaler for information on different parameters.

Default:

```
from sklearn.preprocessing import StandardScaler

estimator = StandardScaler()
```

Outputs

Output Transformer **[fileDestination]** Specifiy output path for the transformer (.pkl). This file can be used for applying the transformer to an image using 'Transformation -> Transform Raster' and 'Transformation -> InverseTransform Raster'.

Default: *outEstimator.pkl*

1.13.13 InverseTransform Raster

Performs an inverse transformation on an previously transformed raster (i.e. output of 'Transformation -> Transform Raster'). Works only for transformers that have an 'inverse_transform(X)' method. See scikit-learn documentations.

Parameters

Raster **[raster]** Specify input raster.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Transformer **[file]** Select path to a transformer file (.pkl).

Outputs

Inverse Transformation **[rasterDestination]** Specify output path for raster.

1.13.14 Transform Raster

Applies a transformer to an raster.

Used in the Cookbook Recipes: Transformation

Parameters

Raster **[raster]** Select raster file which should be regressed.

Mask **[layer]** Specified vector or raster is interpreted as a boolean mask.

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 0) 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.

Transformer **[file]** Select path to a transformer file (.pkl).

Outputs

Transformation **[rasterDestination]** Specify output path for raster.