

## Description of Parameters in PyGRF

- **class PyGRFBuilder(band\_width, n\_estimators=100, max\_features=1.0, kernel="adaptive", train\_weighted=True, predict\_weighted=True, resampled=True, n\_jobs=None, bootstrap=True, random\_seed=None):**

Python implementation of geographic random forest (PyGRF).

**Parameters:**    **band\_width: int or float**

- The number (int) of neighbors for fitting local models if kernel is “adaptive”.
- The number (int or float) of distance within which the neighbors are used for model fitting if kernel is “fixed”.

**n\_estimators: int, default=100**

The number of trees in the forest.

**max\_features: {"sqrt", "log2", None}, int or float, default=1.0**

The number of input variables to consider at each split.

More details please refer to the documentation of scikit-learn at the link: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>.

**kernel: {"adaptive", "fixed"}, default="adaptive"**

The type of kernel used for determining the neighbors of a data point. Two types are available:

- If “adaptive”, a specific number of neighbors to use for fitting local models.
- If “fixed”, neighbors within a fixed distance for fitting local models.

**train\_weighted: bool, default = True**

Whether samples are weighted based on distances for training local models. If False, samples are equally weighted.

**predict\_weighted: bool, default = True**

Whether the ensemble of local models within the bandwidth is used and spatially weighted for producing local predictions. If False, only closest local model is used for producing local predictions.

**resampled: bool, default = True**

Whether local samples are expanded. If False, the original samples are used for fitting local models.

**n\_jobs: int, default=None**

The number of jobs to execute in parallel.

More details please refer to the documentation of scikit-learn at the link: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

**bootstrap: bool, default = True**

Whether each tree is built using bootstrap sampling (with replacement) from the original dataset. If False, each tree is built using the entire dataset.

Note that this parameter should be true if out of bag (OOB) predictions are needed.

More details please refer to the documentation of scikit-learn at the link: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>.

**random\_seed: int, instance of Numpy RandomState or None, default=None**

Determine the randomness within the model fitting. This parameter has to be fixed in order to achieve reproducibility in the model fitting process.

More details please refer to the documentation of scikit-learn at the link: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>.

- **method PyGRFBuilder.fit(self, X\_train, y\_train, coords)**

Fit PyGRF model.

**Parameters:**    **X\_train: data frame**

A data frame of the independent variables of training samples.

**y\_train: data series**

A data series of the dependent variable of training samples.

**coords: data frame**

A data frame of the two-dimensional coordinates of training samples. It is recommended to use projected coordinates.

**Returns:**        **global\_oob\_prediction: list**

The OOB prediction from the global model

**local\_oob\_prediction: list**

The OOB prediction from the local models

- **method PyGRFBuilder.predict(self, X\_test, coords\_test, local\_weight)**

Make predictions for test data using fitted model.

**Parameters:**    **X\_test: data frame**

A data frame of the independent variables of test samples.

**coords\_test: data frame**

A data frame of the two-dimensional coordinates of test samples. It is recommended to use projected coordinates.

**local\_weight: float**

A number for combining global and local predictions

**Returns:**        **predict\_combined: list**

A list of predictions combined from global and local predictions.

**predict\_global: list**

A list of global predictions.

**predict\_local: list**

A list of local predictions.

- **method PyGRFBuilder.get\_local\_feature\_importance(self)**

Get the local feature importance based on local models.

**Returns:**        **feature\_importance\_df: data frame**

A data frame containing all the feature importance from local models.

- **function search\_bw\_lw\_ISA(y, coords, bw\_min=None, bw\_max=None, step=1)**

Search for bandwidth and local model weight using incremental spatial autocorrelation (ISA).

**Parameters:** **y: data series**

A data series of dependent variable of samples.

**coords: data frame**

A data frame of two-dimensional coordinates of samples. It is recommended to use projected coordinates.

**bw\_min: int, default = None**

The minimum band\_width for searching.

**bw\_max: int, default = None**

The maximum band\_width for searching.

**step: int, default = 1**

The step for iterating the band\_width between minimum and maximum values.

**Returns:** **found\_bandwidth: int**

The found bandwidth using ISA.

**found\_moran\_I: float**

The Moran's I corresponding to the found bandwidth.

**found\_p\_value: float**

The p-value corresponding to the Moran's I.

- **funciton search\_bandwidth(X, y, coords, n\_estimators, max\_features, bw\_min=None, bw\_max=None, step=1, train\_weighted=True, resampled=True, n\_jobs=None, random\_seed=None):**  
Optimize the bandwidth using OOB score.

**Parameters:** **X: data frame**

A data frame of independent variables of samples in the data used for searching the optimal bandwidth.

**y: data series**

A data series of dependent variable of samples.

**coords: data frame**

A data frame of two-dimensional coordinates of samples. It is recommended to use projected coordinates.

**n\_estimators: int**

The number of trees for the PyGRF model.

**max\_features: {"sqrt", "log2", None}, int or float**

The number of input variables to consider at each split.

**bw\_min: int, default = None**

The minimum band\_width for searching.

**bw\_max: int, default = None**

The maximum band\_width for searching.

**step: int, default = 1**

The step for iterating the band\_width between minimum and maximum values.

**train\_weighted: bool, default = True**

Whether samples are weighted based on distances in the PyGRF model.

**resampled: bool, default = True**

Whether local samples are expanded in the PyGRF model.

**n\_jobs: int, default=None**

The number of jobs to execute in parallel.

**random\_seed: int, instance of Numpy RandomState or None, default=None**

Determine the randomness within the PyGRF model fitting.

**Returns:**

**search\_result: dictionary**

The result of searching the optimal band\_width.