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_state=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_state: 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 recommend ed 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.

- **method PyGRFBuilder.get_local_R2(self)**
  Retrieve the local R-squared scores of the local models.

  Returns: **R2_df: data frame**
  A data frame containing the R-squared scores of 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-dimentional 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_state=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-dimentional 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_state: 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.