ImbalancedLearningRegression

Release 0.0.1

Wenglei Wu

Mar 19, 2023

INTRODUCTION

CHAPTER

ONE

LAYOUT

1.1 Imbalanced Learning Regression

1.1.1 Description

A Python implementation of sampling techniques for Regression. Conducts different sampling techniques for Regression. Useful for prediction problems where regression is applicable, but the values in the interest of predicting are rare or uncommon. This can also serve as a useful alternative to log transforming a skewed response variable, especially if generating synthetic data is also of interest.

1.1.2 Features

1. An open-source Python supported version of sampling techniques for Regression, a variation of Nick Kunz's package SMOGN.
2. Supports Pandas DataFrame inputs containing mixed data types.
3. Flexible inputs available to control the areas of interest within a continuous response variable and friendly parameters for re-sampling data.
4. Purely Pythonic, developed for consistency, maintainability, and future improvement, no foreign function calls to C or Fortran, as contained in original R implementation.

1.1.3 Requirements

1. Python 3
2. NumPy
3. Pandas
4. Scikit-learn

1.1.4 Installation

Install pypi release

```
$ pip install ImbalancedLearningRegression
```

Install developer version

```
$ pip install git+https://github.com/paobranco/ImbalancedLearningRegression.git
```

1.1.5 Usage

```
>>> ## load libraries
>>> import pandas
>>> import ImbalancedLearningRegression as iblr

>>> ## load data
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")

>>> ## conduct Random Over-sampling
>>> housing_ro = iblr.ro(data = housing, y = "SalePrice")

>>> ## conduct Introduction of Gaussian Noise
>>> housing_gn = iblr.gn(data = housing, y = "SalePrice")
```

1.1.6 Examples

1. Random Over-sampling
2. Introduction of Gaussian Noise
3. Condensed Nearest Neighbor
4. Edited Nearest Neighbor

For the examples of other techniques, please refer to here.

1.1.7 License

1.1.8 Contributions

ImbalancedLearningRegression is open for improvements and maintenance. Your help is valued to make the package better for everyone.

1.1.9 References

Branco, P., Torgo, L., Ribeiro, R. (2017). SMOGN: A Pre-Processing Approach for Imbalanced Regression. Proceedings of Machine Learning Research, 74:36-50. http://proceedings.mlr.press/v74/branco17a/branco17a.pdf

Branco, P., Torgo, L., & Ribeiro, R. P. (2019). Pre-processing approaches for imbalanced distributions in regression. Neurocomputing, 343, 76-99. https://www.sciencedirect.com/science/article/abs/pii/S0925231219301638

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357. https://www.jair.org/index.php/jair/article/view/10302

Elhassan, T., & Aljurf, M. (2016). Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. Global J Technol Optim S, 1. https://www.researchgate.net/profile/Mohamed-Shoukri-2/publication/326590590_Classification_of_Imbalance_Data_using_Tomek_Link_T-Link_Combined_with_Random_Under-sampling_RUS_as_a_Data_Reduction_Method/links/5b96a6a0a6fdccfd543cbc40/Classification-of-Imbalance-Data-using-Tomek-Link-T-Link-Combined-with-Random-Under-sampling-RUS-as-a-Data-Reduction-M pdf

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). IEEE transactions on information theory, 14(3), 515-516. https://ieeexplore.ieee.org/document/1054155

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008, June). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence) (pp. 1322-1328). IEEE. https://www.ele.uri.edu/faculty/he/PDFfiles/adasyn.pdf

Kunz, N., (2019). SMOGN. https://github.com/nickkunz/smogn

Menardi, G., & Torelli, N. (2014). Training and assessing classification rules with imbalanced data. Data mining and knowledge discovery, 28(1), 92-122. https://link.springer.com/article/10.1007/s10618-012-0295-5

Tomek, I. (1976). Two modifications of CNN. IEEE Trans. Systems, Man and Cybernetics, 6, 769-772. https://ieeexplore.ieee.org/document/4309452

Torgo, L., Ribeiro, R. P., Pfahringer, B., & Branco, P. (2013, September). Smote for regression. In Portuguese conference on artificial intelligence (pp. 378-389). Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/978-3-642-40669-0_33

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics, (3), 408-421. https://ieeexplore.ieee.org/abstract/document/4309137

1.2 Random Over-sampling

Random Over-sampling is an over-sampling method that synthesizes new samples by randomly copying minority samples.

ro(*data*, *y*, *samp_method='balance'*, *drop_na_col=True*, *drop_na_row=True*, *replace=True*, *manual_perc=False*, *perc_o=-1*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.

- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **replace** (*bool*) – Randomly select sample to duplicate: with or without replacement.
- **manual_perc** (*bool*) – Keep the same percentage of re-sampling for all bins. If `True`, `perc_o` is required to be a positive real number.
- **perc_o** (*float*) – User-specified fixed percentage of over-sampling for all bins. Must be a positive real number if `manual_perc = True`.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.

Returns
Re-sampled dataset.

Return type
Pandas dataframe

Raises
ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or synthetic data contains missing values.

1.2.1 References

[1] G Menardi, N. Torelli, "Training and assessing classification rules with imbalanced data," Data Mining and Knowledge Discovery, 28(1), pp.92-122, 2014.

[2] P. Branco, L. Torgo, R. P. Ribeiro, "Pre-processing approaches for imbalanced distributions in regression," Neurocomputing, 343, pp. 76-99, 2019.

1.2.2 Examples

```
>>> from ImbalancedLearningRegression import ro
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_ro = ro(data = housing, y = "SalePrice")
```

1.3 SMOTE

Synthetic Minority Oversampling Technique. SMOTE is an over-sampling method that synthesizes new samples by using one of the neighbors of a seed sample.

smote(*data*, *y*, *k=5*, *samp_method='balance'*, *drop_na_col=True*, *drop_na_row=True*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **k** (*int*) – The number of neighbors considered. Must be a positive integer.
- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.

Returns

Re-sampled dataset.

Return type

Pandas dataframe

Raises

ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or synthetic data contains missing values.

1.3.1 References

[1] L. Torgo, R. P. Ribeiro, B. Pfahringer, P. Branco, "Smote for regression," In Portuguese conference on artificial intelligence, pp. 378-389, 2013. Springer, Berlin, Heidelberg.

[2] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 321-357, 2002.

1.3.2 Examples

```
>>> from ImbalancedLearningRegression import smote
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_smote = smote(data = housing, y = "SalePrice")
```

1.4 Introduction of Gaussian Noise

Introduction of Gaussian Noise is an over-sampling method that synthesizes new samples by introducing small perturbations on the numeric attributes and target variables of the seed samples. This over-sampling method has an optional choice of random under-sampling.

gn(*data*, *y*, *pert=0.02*, *samp_method='balance'*, *under_samp=True*, *drop_na_col=True*, *drop_na_row=True*, *replace=False*, *manual_perc=False*, *perc_u=-1*, *perc_o=-1*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **pert** (*float*) – Perturbation amplitude. Must be a real number between 0 and 1 (0, 1].
- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **under_samp** (*bool*) – If `True`, random under-sampling will be conducted on the normal bins.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **replace** (*bool*) – For decimal part of the over-sampling percentage, a subset of original dataset will be choosed as base samples to introduce noise, the selection can be with or without replacement.
- **manual_perc** (*bool*) – Keep the same percentage of re-sampling for all bins. If `True`, `perc_u` is required to be a real number between 0 and 1 (0, 1), and `perc_o` is required to be a positive real number.
- **perc_u** (*float*) – User-specified fixed percentage of under-sampling for all bins. Must be a real number between 0 and 1 (0, 1) if `manual_perc = True`.

- **perc_o** (*float*) – User-specified fixed percentage of over-sampling for all bins. Must be a positive real number if `manual_perc = True`.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.

Returns
Re-sampled dataset.

Return type
Pandas dataframe

Raises
ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or synthetic data contains missing values.

1.4.1 References

[1] P. Branco, L. Torgo, R. P. Ribeiro, "Pre-processing approaches for imbalanced distributions in regression," Neurocomputing, 343, pp. 76-99, 2019.

1.4.2 Examples

```
>>> from ImbalancedLearningRegression import gn
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_gn = gn(data = housing, y = "SalePrice")
```

1.5 ADASYN

ADASYN is an over-sampling method that is similar to SMOTE, but more samples are generated for seed samples that are difficult to be learned.

adasyn(*data*, *y*, *k=5*, *samp_method='balance'*, *drop_na_col=True*, *drop_na_row=True*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **k** (*int*) – The number of neighbors considered. Must be a positive integer.

- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.

Returns
Re-sampled dataset.

Return type
Pandas dataframe

Raises

- **ValueError** – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or synthetic data contains missing values.
- **AssertionError** – If normalized ratio ri does not sum up to one.

1.5.1 References

[1] He, Haibo, Yang Bai, Edwardo A. Garcia, and Shutao Li. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," In IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322-1328, 2008.

1.5.2 Examples

```
>>> from ImbalancedLearningRegression import adasyn
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_adasyn = adasyn(data = housing, y = "SalePrice")
```

1.6 Random Under-sampling

Random Under-sampling is an under-sampling method that randomly select a subset of majority samples.

random_under(*data*, *y*, *samp_method='balance'*, *drop_na_col=True*, *drop_na_row=True*, *replacement=False*, *manual_perc=False*, *perc_u=-1*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **replacement** (*bool*) – Randomly select sample to duplicate: with or without replacement.
- **manual_perc** (*bool*) – Keep the same percentage of re-sampling for all bins. If `True`, `perc_u` is required to be a real number between 0 and 1 (0, 1).
- **perc_u** (*float*) – User-specified fixed percentage of under-sampling for all bins. Must be a real number between 0 and 1 (0, 1) if `manual_perc = True`.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.

Returns

Re-sampled dataset.

Return type

Pandas dataframe

Raises

ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or under_sampled data contains missing values.

1.6.1 Examples

```
>>> from ImbalancedLearningRegression import random_under
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_ru = random_under(data = housing, y = "SalePrice")
```

1.7 Condensed Nearest Neighbor

Condensed Nearest Neighbor is an under-sampling method that condenses the majority set by selecting a subset of majority samples from the original majority set.

cnn(*data*, *y*, *samp_method='balance'*, *drop_na_col=True*, *drop_na_row=True*, *n_seed=1*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*, *k=1*, *n_jobs=1*, *k_neighbors_classifier=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **n_seed** (*int*) – Number of majority samples put into STORE at the beginning of under-sampling each normal bin. Must be a positive integer.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.
- **k** (*int*) – The number of neighbors considered. Must be a positive integer.
- **n_jobs** (*int*) – The number of parallel jobs to run for neighbors search. Must be an integer. See sklearn.neighbors.KNeighborsClassifier for more details.
- **k_neighbors_classifier** (*KNeighborsClassifier*) – If users want to define more parameters of KNeighborsClassifier, such as `weights`, `algorithm`, `leaf_size`, and `metric`, they can create an instance of KNeighborsClassifier and pass it to this method. In that case, setting `k` and `n_jobs` will have no effect.

Returns
Re-sampled dataset.

Return type
Pandas dataframe

Raises
ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or once the index of a sample exists in both STORE and GRABBAG.

1.7.1 References

[1] P. Hart, "The condensed nearest neighbor rule (corresp.).," IEEE transactions on information theory, 14(3), pp. 515-516, 1968.

1.7.2 Examples

```
>>> from ImbalancedLearningRegression import cnn
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_cnn = cnn(data = housing, y = "SalePrice")
```

1.8 TomekLinks

TomekLinks is an under-sampling method that under-samples the majority/minority/both class(es) by removing Tomek-Links.

tomeklinks(*data*, *y*, *option='majority'*, *drop_na_col=True*, *drop_na_row=True*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **option** (*str*) – Sampling information to sample the data set. If `majority`, resample only the majority class; if `minority`, resample only the minority class; if `both`, resample both majority and minority class.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.

- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.

Returns
Re-sampled dataset.

Return type
Pandas dataframe

Raises
ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one, or synthetic data contains missing values.

1.8.1 References

[1] I. Tomek, "Two modifications of CNN," In Systems, Man, and Cybernetics, IEEE Transactions on, vol. 6, pp 769-772, 1976.

[2] T. Elhassan, M. Aljurf, "Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method," Global J Technol Optim S, 1, 2016.

1.8.2 Examples

```
>>> from ImbalancedLearningRegression import tomeklinks
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_tomeklinks = tomeklinks(data = housing, y = "SalePrice")
```

1.9 Edited Nearest Neighbor

Edited Nearest Neighbor is an under-sampling method that edits the majority set by removing some of the majority samples from the original majority set.

enn(*data*, *y*, *samp_method='balance'*, *drop_na_col=True*, *drop_na_row=True*, *rel_thres=0.5*, *rel_method='auto'*, *rel_xtrm_type='both'*, *rel_coef=1.5*, *rel_ctrl_pts_rg=None*, *k=3*, *n_jobs=1*, *k_neighbors_classifier=None*)

Parameters

- **data** (*Pandas dataframe*) – Pandas dataframe, the dataset to re-sample.
- **y** (*str*) – Column name of the target variable in the Pandas dataframe.
- **samp_method** (*str*) – Method to determine re-sampling percentage. Either `balance` or `extreme`.
- **drop_na_col** (*bool*) – Determine whether or not automatically drop columns containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.

- **drop_na_row** (*bool*) – Determine whether or not automatically drop rows containing NaN values. The data frame should not contain any missing values, so it is suggested to keep it as default.
- **rel_thres** (*float*) – Relevance threshold, above which a sample is considered rare. Must be a real number between 0 and 1 (0, 1].
- **rel_method** (*str*) – Method to define the relevance function, either `auto` or `manual`. If `manual`, must specify `rel_ctrl_pts_rg`.
- **rel_xtrm_type** (*str*) – Distribution focus, `high`, `low`, or `both`. If `high`, rare cases having small y values will be considerd as normal, and vise versa.
- **rel_coef** (*float*) – Coefficient for box plot.
- **rel_ctrl_pts_rg** (*2D array*) – Manually specify the regions of interest. See SMOGN advanced example for more details.
- **k** (*int*) – The number of neighbors considered. Must be a positive integer.
- **n_jobs** (*int*) – The number of parallel jobs to run for neighbors search. Must be an integer. See sklearn.neighbors.KNeighborsClassifier for more details.
- **k_neighbors_classifier** (*KNeighborsClassifier*) – If users want to define more parameters of KNeighborsClassifier, such as `weights`, `algorithm`, `leaf_size`, and `metric`, they can create an instance of KNeighborsClassifier and pass it to this method. In that case, setting `k` and `n_jobs` will have no effect.

Returns
Re-sampled dataset.

Return type
Pandas dataframe

Raises
ValueError – If an input attribute has wrong data type or invalid value, or relevance values are all zero or all one.

1.9.1 References

[1] D. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," In IEEE Transactions on Systems, Man, and Cybernetrics, vol. 2 (3), pp. 408-421, 1972.

1.9.2 Examples

```
>>> from ImbalancedLearningRegression import enn
>>> housing = pandas.read_csv("https://raw.githubusercontent.com/paobranco/
↪ImbalancedLearningRegression/master/data/housing.csv")
>>> housing_enn = enn(data = housing, y = "SalePrice")
```

1.10 Glossary

2D array
A Matrix. A list of lists having the same length.

KNeighborsClassifier
A K-nearest-neighbor classifier class from Scikit-learn.

Pandas dataframe
Data frme of Package Pandas.

INDEX