imputena

Jun 08, 2020

Contents:

CHAPTER 1

Public functions

1.1 Listwise deletion

imputena.**delete_listwise**(*data=None*, *threshold=None*, *inplace=False*)

Performs listwise deletion on the data: Drops any rows that contain missing values. If a threshold is given, the function drops those rows which have less non-NA values. If the operation should only affect certain columns, user delete_pairwise instead.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the listwise deletion of missing values.
- **threshold** (*int, optional*) – If the data is a DataFrame, require that many non-NA values
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with all rows containing NA eliminated, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.2 Pairwise deletion

imputena.**delete_pairwise**(*data=None*, *columns=None*, *threshold=None*, *inplace=False*)

Performs pairwise deletion on the data: Drops any rows that contain NA values in any of the specified columns. If a threshold is given, the function drops those rows which have less non-NA values in the specified columns.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the pairwise deletion of missing values.

- **columns** (*array-like*) – rows will be dropped if any of their value in any of those columns is NA.
- **threshold** (*int, optional*) – Require that many non-NA values in the specified columns
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The dataframe with all rows containing NA in one or more of the specified columns eliminated or None if inplace=True.

Return type pandas.DataFrame or None

Raises TypeError, ValueError

1.3 Variable deletion

imputena.**delete_columns**(*data=None*, *columns=None*, *threshold=None*, *inplace=False*)

Drops variables that contain NA values from the data. If a list of column names is passed, all other columns will be ignored, otherwise all the columns will be considered. If a threshold is given, the function drops those rows which have less non-NA values in the specified columns.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the pairwise dropping of variables.
- **columns** (*array-like, optional*) – The columns which should be considered. If not passed or None, all columns will be considered.
- **threshold** (*int, optional*) – Require that many non-NA values in order to not drop a column. If not passed or None, all columns with any NA value will be dropped.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The dataframe with columns that contain NA dropped or None if inplace=True.

Return type pandas.DataFrame or None

Raises TypeError, ValueError

1.4 Random sample imputation

imputena.**random_sample_imputation**(*data=None*, *columns=None*, *inplace=False*)

Performs random sample imputation on the data. Missing values in each column are replaced by a randomly selected observed values of the same column, if available. The operation can be applied to a series, a whole dataframe, or a selection of columns of a dataframe.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the random sample imputation
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.5 Random hot deck imputation

imputena.**random_hot_deck_imputation**(*data=None*, *incomplete_variable=None*, *deck_variables=None*, *inplace=False*)

Performs random hot deck imputation on the data. Missing values receive a valid value from a donor randomly chosen from a pool. The pool is different for each row containing a missing value in incomplete_variable and consists of all rows which coincide in value with the incomplete row for all of the columns in deck_variables.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the random hot deck imputation.
- **incomplete_variable** (*String*) – The variable in which the missing values should be imputed.
- **deck_variables** (*array-like*) – The donor has to have the same value as the row for these variables.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The dataframe with random hot deck imputation performed for the incomplete variable or None if inplace=True.

Return type pandas.DataFrame or None

Raises TypeError, ValueError

1.6 Last observation carried forward

imputena.**locf**(*data=None*, *fill_leading=False*, *columns=None*, *inplace=False*)

Fills in NA values with the last observation in the same column. If fill_leading is true, leading values are filled in with the first observation. The operation can be applied to a series, a whole dataframe, or a selection of columns of a dataframe.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the LOCF operation.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **fill_leading** (*bool, default False*) – Whether to fill in leading NA values with the first observation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.7 Next observation carried backward

imputena.**nocb**(*data=None*, *fill_trailing=False*, *columns=None*, *inplace=False*)

Fills in NA values with the next observation in the same column. If fill_trailing is true, trailing values are filled in with the last observation. The operation can be applied to a series, a whole dataframe, or a selection of columns of a dataframe.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the NOCB operation.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **fill_trailing** (*bool, default False*) – Whether to fill in trailing NA values with the last observation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.8 Substitution by most frequent value

imputena.**most_frequent**(*data=None*, *columns=None*, *inplace=False*)

Fills in missing values with the most frequent value (mode) in the same column, in case of a dataframe, or in the series as a whole in case of a series. If the data is passed as a dataframe, the operation can be applied to all columns, by leaving the parameter columns empty; or to selected columns, passed as an array of strings.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the most frequent imputation.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.9 Substitution by mean or median value

imputena.**mean_substitution**(*data=None*, *method='mean'*, *columns=None*, *inplace=False*)

Fills in missing values with the average value of the same column, in case of a dataframe, or of the series as a whole in case of a series. If the data is passed as a dataframe, the operation can be applied to all columns, by leaving the parameter columns empty, or to selected columns, passed as an array of strings.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the mean substitution.

- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **method** (*{'mean', 'median'}, default 'mean'*) – Method to use to calculate the average.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.10 Imputation by a constant value

imputena.**constant_value_imputation**(*data=None*, *value=0*, *columns=None*, *inplace=False*)

Fills in missing values with the constant value given. If the data is passed as a dataframe, the operation can be applied to all columns, by leaving the parameter columns empty, or to selected columns, passed as an array of strings.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the constant value imputation
- **value** (*scalar, dict, Series, or DataFrame, default 0*) – The value with which to fill in missing values. If columns is not set, the value can be a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). Values not in the dict/Series/DataFrame will not be filled.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.11 Imputation by a random value

imputena.**random_value_imputation**(*data=None*, *distribution='uniform'*, *vmin=0*, *vmax=1*, *sigma=1*, *mu=0*, *columns=None*, *inplace=False*)

Fills in missing values with a randomly generated number. If distribution is uniform, a float between vmin (inclusive) and vmax (exclusive) will be generated. If distribution is normal, a float from a normal distribution specified by sigma and int will be generated. If distribution is integer, an integer between vmin (inclusive) and vmax (exclusive) will be drawn from a uniform distribution. If the data is passed as a dataframe, the operation can be applied to all columns, by leaving the parameter columns empty, or to selected columns, passed as an array of strings.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the constant value imputation
- **distribution** (*{'uniform', 'normal', 'integer'}, default 'uniform'*) – The distribution from which to draw the random values.

- **vmin** (*scalar, default 0*) – The lowest value to be drawn
- **vmax** (*scalar, default 1*) – One above the highest value to be drawn
- **sigma** (*scalar, default 1*) – The sigma value tu be used when drawing from a normal distribution.
- **mu** (*scalar, default 0*) – The mu value tu be used when drawing from a normal distribution.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values filled in, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.12 Interpolation

imputena.**interpolation**(*data=None*, *method='linear'*, *direction='both'*, *columns=None*, *inplace=False*)

Performs linear, quadratic, or cubic interpolation on a series or a data frame. If the data is passed as a dataframe, the operation can be applied to all columns, by leaving the parameter columns empty, or to selected columns, passed as an array of strings.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the interpolation.
- **method** (*{'linear', 'quadratic', 'cubic'}, default 'linear'*) – The interpolation model to use.
- **direction** (*{'forward', 'backward', 'both'}, default 'both'*) – Direction in which to interpolate values when the interpolation method is linear.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values interpolated, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.13 Interpolation with seasonal adjustment

imputena.**seasonal_interpolation**(*data=None*, *dec_model='multiplicative'*, *int_method='linear'*, *int_direction='both'*, *columns=None*, *inplace=False*)

Performs interpolation with seasonal adjustment on a time series or a data frame containing time series. First, the time series gets decomposed according to the decomposition model (additive or multiplicative). Then, the missing values are interpolated using the interpolation method (linear, cubic or quadratic) on a series consisting of only the trend and irregular components. Finally, the seasonality is added back to the series.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data on which to perform the seasonal interpolation.
- **dec_model** (*{'multiplicative', 'additive'}, default 'multiplicative'*) – The decomposition model to use.
- **int_method** (*{'linear', 'quadratic', 'cubic'}, default 'linear'*) – The interpolation model to use.
- **int_direction** (*{'forward', 'backward', 'both'}, default 'both'*) – Direction in which to interpolate values when the interpolation method is linear.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values interpolated, or None if inplace=True.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

1.14 Linear and stochastic regression imputation

imputena.**linear_regression**(*data=None*, *dependent=None*, *predictors=None*, *regressions='available'*, *noise=False*, *inplace=False*)

Performs simple or multiple linear regression imputation on the data. First, the regression equation for the dependent variable given the predictor variables is computed. For this step, all rows that contain a missing value in either the dependent variable or any of the predictor variable is ignored via pairwise deletion. Then, missing valued in the dependent column in imputed using the regression equation. If, in the same row as a missing value in the dependent variable the value for any predictor variable is missing, a regression model based on all available predictors in calculated just to impute those values where the predictor(s) are missing. This behavior can be changed by assigning to the parameter regressions the value 'complete'. In this case, rows in which a predictor variable is missing do not get imputed. If stochastic regression imputation should be performed, set noise=True. In this case, a random value is chosen from a normal distribution with the width of the standard error of the regression model and added to the imputed value. If the parameter predictors is omitted, all variables other than the dependent are used as predictors. If the parameter dependent is omitted, the operation is performed on all columns that contain missing values.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the linear regression imputation.
- **dependent** (*String, optional*) – The dependent variable in which the missing values should be imputed.
- **predictors** (*array-like, optional*) – The predictor variables on which the dependent variable is dependent.
- **regressions** (*{'available', 'complete'}, default 'available'*) – If 'available': Impute missing values by modeling a regression based on all available predictors if some predictors have missing values themselves. If 'complete': Only impute with a regression model based on all predictors and leave missing values in rows in which some predictor value is missing itself unimputed.
- **noise** (*bool, default False*) – Whether to add noise to the imputed values (stochastic regression imputation)

- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The dataframe with linear regression imputation performed for the incomplete variable(s) or None if inplace=True.

Return type pandas.DataFrame or None

Raises TypeError, ValueError

1.15 Logistic regression imputation

imputena.**logistic_regression**(*data=None*, *dependent=None*, *predictors=None*, *regressions='available'*, *inplace=False*)

Performs logistic regression imputation on the data. First, the regression equation for the dependent variable given the predictor variables is computed. For this step, all rows that contain a missing value in either the dependent variable or any of the predictor variable is ignored via pairwise deletion. Then, missing valued in the dependent column in imputed using the regression equation. If, in the same row as a missing value in the dependent variable the value for any predictor variable is missing, a regression model based on all available predictors in calculated just to impute those values where the predictor(s) are missing. This behavior can be changed by assigning to the parameter regressions the value 'complete'. In this case, rows in which a predictor variable is missing do not get imputed. If the parameter predictors is omitted, all variables other than the dependent are used as predictors.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the logistic regression imputation.
- **dependent** (*String*) – The dependent variable in which the missing values should be imputed.
- **predictors** (*array-like, optional*) – The predictor variables on which the dependent variable is dependent.
- **regressions** (*{'available', 'complete'}, default 'available'*) – If 'available': Impute missing values by modeling a regression based on all available predictors if some predictors have missing values themselves. If 'complete': Only impute with a regression model based on all predictors and leave missing values in rows in which some predictor value is missing itself unimputed.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The dataframe with logistic regression imputation performed for the incomplete variable or None if inplace=True.

Return type pandas.DataFrame or None

Raises TypeError, ValueError

1.16 K-nearest neighbors imputation

imputena.**knn**(*data=None*, *columns=None*, *k=3*, *inplace=False*)

Performs k-nearest neighbors imputation on the data. The k nearest neighbors or each subject with missing data are chosen and the average of their values is used to impute the missing value. The operation can be applied to all columns, by leaving the parameter columns empty, or to selected columns, passed as an array of strings.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the k-nearest neighbors imputation.
- **columns** (*array-like, optional*) – Columns on which to apply the operation.
- **k** (*int, default 3*) – The number of neighbors to which the subject with missing values should be compared
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Returns The series or dataframe with NA values imputed, or None if inplace=True.

Return type pandas.DataFrame or None

Raises TypeError, ValueError

1.17 Sequential regression multiple imputation

imputena.**srmi**(*data=None*, *sample_size=10*, *imputations=3*, *regressions='available'*)

Performs sequential regression multiple imputation on the data. Several (parameter imputations) imputations are performed and the resulting dataframes returned as a list. For each one, a regression model is created based on a sample of the available rows. The size of these samples is fixed by the parameter sample_size. If, in the same row as a missing value in the dependent variable the value for any predictor variable is missing, a regression model based on all available predictors in calculated just to impute those values where the predictor(s) are missing. This behavior can be changed by assigning to the parameter regressions the value 'complete'. In this case, rows in which a predictor variable is missing do not get imputed.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the SRMI.
- **sample_size** (*scalar, default 10*) – Maximum size of the set of rows used to compute the regression model. Has to be at least 2.
- **imputations** (*scalar, default 3*) – Number of imputations to perform
- **regressions** (*{'available', 'complete'}, default 'available'*) – If 'available': Impute missing values by modeling a regression based on all available predictors if some predictors have missing values themselves. If 'complete': Only impute with a regression model based on all predictors and leave missing values in rows in which some predictor value is missing itself unimputed.

Returns A list of linear regression imputations performed based on regression models calculated from different samples.

Return type list of pandas.DataFrame

Raises TypeError, ValueError

1.18 Multiple imputation by chained equations

imputena.**mice**(*data=None*, *imputations=3*)

Performs multiple imputation by chained equations (MICE) on the data. Several (parameter imputations) linear regression imputations are performed on the dataset. For each one, the a random order of imputation of columns in generated. Then the dataset is imputed with mean substitution. For each column with missing data, and in the previously generated order, (1) the missing values imputed with the mean are set missing again, (2) a linear regression model is calculated based on the available data and (3) the predictions from the model are used to impute the missing values.

Parameters

- **data** (*pandas.DataFrame*) – The data on which to perform the MICE imputation.
- **imputations** (*scalar, default 3*) – Number of imputations to perform

Returns A list of MICE imputations performed with randomly chosen orders of column imputations.

Return type list of pandas.DataFrame

Raises TypeError, ValueError

1.19 Get applicable methods

imputena.**get_applicable_methods**(*data=None*)

Informs about the imputation methods that are applicable to a given data frame or series, based on the number of variables (one or multiple), type of data (categorical, numerical, o both), and whether the data is of temporal nature.

Parameters **data** (*pandas.Series or pandas.DataFrame*) – The data for which an the applicable imputation method should be returned.

Returns The imputation methods that are applicable to the data

Return type set of strings

Raises TypeError

1.20 Recommend method

imputena.**recommend_method**(*data=None*, *column=None*, *title_only=False*)

Recommends an imputation method to use on a series, data frame, or particular column of a data frame. If data_only is True, only the title of the recommended method is returned, otherwise a description of the decision process is provided as well.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data for which an imputation method should be recommended.
- **column** (*string, optional*) – If data is a data frame, the column for which an imputation method should be recommended
- **title_only** (*bool, default False*) – If true, return only the title of the imputation method, otherwise provide a description of the decision process as well.

Returns The title of the recommended imputation method and a description of the decision model if title_only is False.

Return type string

1.21 Impute by recommended

imputena.**impute_by_recommended**(*data=None*, *column=None*, *inplace=False*)

Imputes a series, dataframe or particular column of a dataframe with the best imputation method for the given data.

Parameters

- **data** (*pandas.Series or pandas.DataFrame*) – The data that should be imputed.
- **column** (*string, optional*) – If data is a data frame, the column that should be imputed.
- **inplace** (*bool, default False*) – If True, do operation inplace and return None.

Return type pandas.Series, pandas.DataFrame, or None

Raises TypeError, ValueError

CHAPTER 2

Indices and tables

- genindex
- search

Index