Contents

1. [add\_noise(n, mean=0, sd=1): 1](#_Toc142049950)
2. [add\_na(vec, n=None, share=0.05) 2](#_Toc142049951)
3. [add\_outlier(vec, n=None, share=0.01) 2](#_Toc142049952)
4. [binarize\_column(self, column\_name, limit) 3](#_Toc142049953)
5. [generate\_name(self) 3](#_Toc142049954)
6. [categorize\_column(self, column\_name, limits) 3](#_Toc142049955)
7. [add\_var(self, var\_name: str, data\_generation\_function: Callable, features: list or dict = None, noise=True, outliers=True, nas=True) 4](#_Toc142049956)
8. [extract\_dependencies(self) 5](#_Toc142049957)
9. [draw\_graph(self) 5](#_Toc142049958)
10. [check\_inputs(self, n: int = None, var\_name: str = None) 5](#_Toc142049959)
11. [add\_nominal(self, n: int = None, var\_name: str = None, topic: Literal["gender", "random"] = "random") 6](#_Toc142049960)
12. [add\_ordinal(self, n: int = None, var\_name: str = None, topic: Literal["grades", "random"] = "random") 6](#_Toc142049961)
13. [add\_interval(self, n: int = None, var\_name: str = None, topic: Literal["IQ", "random"] = "random") 7](#_Toc142049962)
14. [add\_ratio(self, n: int = None, var\_name: str = None, topic: Literal["revenue", "random"] = "random") 7](#_Toc142049963)
15. [gen\_target(self, var\_name: str = "target", dependency\_rate: float = 0.1, mandatory\_features: list = [], bias: list = []) 8](#_Toc142049964)
16. [add\_target(self, var\_name: str = None, dependency\_rate: float = 0.2, target\_type: Literal["numerical", "binary", "categorical"] = "numerical", topic: Literal["random", "loan"] = "random", n\_classes: int = 2) 9](#_Toc142049965)
17. [get\_ground\_truth(self) 9](#_Toc142049966)
18. [create\_df(self) 10](#_Toc142049967)
19. [gen\_multi(self, n:int=None, num\_cols:int=0, cat\_cols:int=0, n\_layers:int=None) 10](#_Toc142049968)

# add\_noise(n, mean=0, sd=1):

**Description**: This function generates and returns an array of random numbers with normal distribution that can be used to add noise to data. The level of noise is controlled by the standard deviation (sd) parameter, while the mean parameter determines the center of the distribution.

**Parameters**:

* **n** (int): Number of random numbers to generate for adding noise.
* **mean** (float): optional Mean (center) of the normal distribution. Default is 0.
* **Sd (**float):optional Standard deviation (strength) of the normal distribution. Default is 1.

**Returns:**

ndarray Array of random numbers with normal distribution, used for adding noise to data.

# add\_na(vec, n=None, share=0.05)

**Description:** The **add\_na** function is used to introduce **None** (or **null**) values into a given series of values. The number of **None** entries can be controlled either by specifying an absolute number **n** or by specifying a relative share **share** of **None** values.

**Parameters**:

* **vec** (array-like): The input series of values that need to be transformed by adding **None** values.
* **n** (int, optional): The fixed number of **None** values to be added. If not provided, the function will calculate the number of **None** values based on the relative share parameter **share**.
* **share** (float, optional): The relative share of **None** values to be added. This parameter is used when **n** is not provided. The default value is **0.05**.

**Returns:**

* **result** (array-like): A new array with **None** values introduced at randomly selected positions in the input series **vec**.

# add\_outlier(vec, n=None, share=0.01)

**Description:** The **add\_outlier** function is designed to introduce outlier values into a given series of values. The number of outlier entries can be controlled either by specifying an absolute number **n** or by specifying a relative share **share** of outlier values. Outliers are generated based on a random distribution and can fall outside the typical range of data values.

**Parameters:**

* **vec** (array-like): The input series of values that need to be transformed by adding outlier values.
* **n** (int, optional): The fixed number of outlier values to be added. If not provided, the function will calculate the number of outlier values based on the relative share parameter **share**.
* **share** (float, optional): The relative share of outlier values to be added. This parameter is used when **n** is not provided. The default value is **0.01**.

**Returns:**

* **result** (array-like): A new array with outlier values introduced at randomly selected positions in the input series **vec**.

# binarize\_column(self, column\_name, limit)

**Description:** The **binarize\_column** method is used to binarize a specific column of the dataframe associated with the object. Binarization involves converting values in the column into binary values (0 or 1) based on whether they are below the specified limit.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **column\_name** (str): The name of the column in the dataframe that needs to be binarized.
* **limit** (numeric): The threshold value. Values in the column that are less than this limit will be assigned a binary value of 1, while values greater than or equal to the limit will be assigned a binary value of 0.

**Returns:**

* **self** (object): The modified object (class instance) with the specified column binarized.

# generate\_name(self)

**Description:** The **generate\_name** method is used to create generic column names that are not yet present in the dataframe associated with the object. It generates column names in the format "colX," where X is a number that ensures the generated name is unique among the existing column names.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.

**Returns:**

* **temp** (str): A generated column name that is not already used in the dataframe.

# categorize\_column(self, column\_name, limits)

**Description:** The **categorize\_column** method replaces numerical values in a specified column of the dataframe associated with the class instance with generic categorical labels. This function is used to discretize continuous data into predefined categories based on specified limits. The categories are determined by the ranges defined by the limits.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **column\_name** (str): The name of the column in the dataframe that needs to be categorized.
* **limits** (list or array-like): The limits that define the boundaries of the categories. The upper limit of one category corresponds to the lower limit of the next category. The values in the column will be mapped to these categories based on these limits.

**Returns:**

* **self** (object): The modified object (class instance) with the specified column categorized.

# add\_var(self, var\_name: str, data\_generation\_function: Callable, features: list or dict = None, noise=True, outliers=True, nas=True)

**Description:** The **add\_var** method adds a new variable as a column to the dataframe associated with the class instance. The method provides flexibility to generate data for the new column using a callable function. It also allows adding various types of synthetic data like noise, outliers, and missing values to the new column.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **var\_name** (str): The name of the new variable (column) to be added to the dataframe.
* **data\_generation\_function** (Callable): A callable function that generates the data for the new column. This function should accept arguments based on the features specified.
* **features** (list or dict, optional): The column names that the **data\_generation\_function** relies on for generating the new variable. If using a dictionary, keys represent argument names in the function, and values are columns in the dataframe. If using a list, elements are columns in the dataframe.
* **noise** (bool, optional): Whether to add noise to the new variable. Noise is added if the column's data type is numeric. Default is **True**.
* **outliers** (bool, optional): Whether to add outliers to the new variable. Outliers are added if the column's data type is numeric. Default is **True**.
* **nas** (bool, optional): Whether to add missing values (None values) to the new variable. Default is **True**.

**Returns:**

* **self** (object): The modified object (class instance) with the new variable added to the dataframe.

# extract\_dependencies(self)

**Description:** The **extract\_dependencies** method extracts the dependencies of various columns in the dataframe and assigns respective levels to them. It prepares the dependency information for graph visualization, where columns that depend on others are visually represented in different levels.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.

**Returns:**

* **features** (list): A list of dictionaries representing the extracted features and their dependencies, along with assigned levels for visualization.

# draw\_graph(self)

**Description:** The **draw\_graph** method exports a graph file and a visualization in PNG format that illustrates the dependencies between different columns in the dataframe associated with the class instance. It leverages graph visualization to visually represent how columns depend on one another.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.

**Returns:**

* Exports a graph file and a .png visualization of the dependencies between different columns in the dataframe of the class instance

# check\_inputs(self, n: int = None, var\_name: str = None)

**Description:** The **check\_inputs** method replaces the **None** values of the **n** and **var\_name** parameters with default values. This method is used to ensure that appropriate values are assigned to these parameters, considering the context of the dataframe and the class instance.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **n** (int, optional): The row number. If not provided, it defaults to the current length of the dataframe's index. If the dataframe is empty, it defaults to **10000**.
* **var\_name** (str, optional): The variable name. If not provided, it defaults to a generic column name generated using the **generate\_name** method.

**Returns:**

* **n** (int): The row number after replacing the **None** value with a default value.
* **var\_name** (str): The variable name after replacing the **None** value with a default value.

# add\_nominal(self, n: int = None, var\_name: str = None, topic: Literal["gender", "random"] = "random")

**Description:** The **add\_nominal** method adds a nominal column to the dataframe associated with the class instance. Nominal columns are categorical columns that represent qualitative data with distinct categories. The method currently implements a distribution for gender, but it can be extended to include other nominal topics like races.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **n** (int, optional): The number of rows (entries) in the new column. If not provided, it defaults to the current length of the dataframe's index. If the dataframe is empty, it defaults to **10000**.
* **var\_name** (str, optional): The name of the new nominal column. If not provided, it defaults to a generic column name generated using the **generate\_name** method.
* **topic** (Literal["gender", "random"], optional): The topic of the nominal column to be added. It can be either "gender" (to add a gender distribution) or "random" (to add a random categorical distribution). Default is "random".

**Returns:**

* **self** (object): The modified object (class instance) with the new nominal column added to the dataframe.

# add\_ordinal(self, n: int = None, var\_name: str = None, topic: Literal["grades", "random"] = "random")

**Description:** The **add\_ordinal** method adds an ordinal column to the dataframe associated with the class instance. Ordinal columns represent categorical data with a clear ordering or ranking of categories. The method currently implements a distribution for grades, but it can be extended to include other ordinal topics.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **n** (int, optional): The number of rows (entries) in the new column. If not provided, it defaults to the current length of the dataframe's index. If the dataframe is empty, it defaults to **10000**.
* **var\_name** (str, optional): The name of the new ordinal column. If not provided, it defaults to a generic column name generated using the **generate\_name** method.
* **topic** (Literal["grades", "random"], optional): The topic of the ordinal column to be added. It can be either "grades" (to add a distribution for grades) or "random" (to add a random ordinal distribution). Default is "random".

**Returns:**

* **self** (object): The modified object (class instance) with the new ordinal column added to the dataframe.

# add\_interval(self, n: int = None, var\_name: str = None, topic: Literal["IQ", "random"] = "random")

**Description:** The **add\_interval** method adds an interval column to the dataframe associated with the class instance. Interval columns represent continuous data with a clear ordering and consistent measurement increments. The method currently implements a distribution for IQ, but it can be extended to include other interval topics.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **n** (int, optional): The number of rows (entries) in the new column. If not provided, it defaults to the current length of the dataframe's index. If the dataframe is empty, it defaults to **10000**.
* **var\_name** (str, optional): The name of the new interval column. If not provided, it defaults to a generic column name generated using the **generate\_name** method.
* **topic** (Literal["IQ", "random"], optional): The topic of the interval column to be added. It can be either "IQ" (to add a distribution for IQ values) or "random" (to add a random interval distribution). Default is "random".

**Returns:**

* **self** (object): The modified object (class instance) with the new interval column added to the dataframe.

# add\_ratio(self, n: int = None, var\_name: str = None, topic: Literal["revenue", "random"] = "random")

**Description:** The **add\_ratio** method adds a ratio column to the dataframe associated with the class instance. Ratio columns represent continuous data with meaningful ratios between values. The method currently implements a distribution for revenue, but it can be extended to include other ratio topics.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **n** (int, optional): The number of rows (entries) in the new column. If not provided, it defaults to the current length of the dataframe's index. If the dataframe is empty, it defaults to **10000**.
* **var\_name** (str, optional): The name of the new ratio column. If not provided, it defaults to a generic column name generated using the **generate\_name** method.
* **topic** (Literal["revenue", "random"], optional): The topic of the ratio column to be added. It can be either "revenue" (to add a distribution for revenue values) or "random" (to add a random ratio distribution). Default is "random".

**Returns:**

* **self** (object): The modified object (class instance) with the new ratio column added to the dataframe.

# gen\_target(self, var\_name: str = "target", dependency\_rate: float = 0.1, mandatory\_features: list = [], bias: list = [])

**Description:** The **gen\_target** method generates a target column and a biased target column for the dataframe associated with the class instance. The target column represents the dependent variable to be predicted, and the biased target column is a version of the target column with added bias. The method allows you to specify mandatory features, features for bias, and other parameters to influence the generation of the target columns.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **var\_name** (str, optional): The name of the target column. Default is "target".
* **dependency\_rate** (float, optional): The rate of dependencies between the target and other features. Default is 0.1.
* **mandatory\_features** (list, optional): A list of feature names that must be included in the generation of the target. Default is an empty list.
* **bias** (list, optional): A list of feature names that contribute to bias in the target generation. Default is an empty list.

**Returns:**

* **self** (object): The modified object (class instance) with the generated target and biased target columns added to the dataframe.

# add\_target(self, var\_name: str = None, dependency\_rate: float = 0.2, target\_type: Literal["numerical", "binary", "categorical"] = "numerical", topic: Literal["random", "loan"] = "random", n\_classes: int = 2)

**Description:** The **add\_target** method adds a target variable to the dataframe associated with the class instance. The method allows you to specify the type of target variable, the generation topic, and the number of classes (for categorical targets).

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has a dataframe associated with it.
* **var\_name** (str, optional): The name of the target variable to be added. If not provided, it defaults to "target".
* **dependency\_rate** (float, optional): The rate of dependencies between the target and other features. Default is 0.2.
* **target\_type** (Literal["numerical", "binary", "categorical"], optional): The type of target variable to be added. It can be "numerical", "binary", or "categorical". Default is "numerical”.
* **topic** (Literal["random", "loan"], optional): The topic for generating the target. It can be "random" (randomly generated target) or "loan" (loan target for debugging). Default is "random".
* **n\_classes** (int, optional): The number of classes for categorical target variables. Applicable only when **target\_type** is "categorical". Default is 2.

**Returns:**

* **self** (object): The modified object (class instance) with the generated target variable added to the dataframe.

# get\_ground\_truth(self)

**Description:** The **get\_ground\_truth** method prints the ground truth functions for the target variables associated with the class instance. It displays both the biased and unbiased ground truth functions.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that has ground truth functions stored within it.

**Returns:**

* Prints the ground truth function for the biased and unbiased dataset

# create\_df(self)

**Description:** The **create\_df** method returns a copy of the generated dataframe associated with the class instance. This method allows you to obtain a duplicate of the dataframe with all the modifications and additions made through the various methods of the class.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that holds the generated dataframe.

**Returns:**

* **df\_copy** (DataFrame): A copy of the generated dataframe with all modifications.

# gen\_multi(self, n:int=None, num\_cols:int=0, cat\_cols:int=0, n\_layers:int=None)

**Description:** The **gen\_multi** method adds columns to the dataframe based on the specified number of numerical and categorical columns requested. This method provides a convenient way to generate multiple columns of different types at once.

**Parameters:**

* **self** (object): The object (instance of a class) that calls the method. Typically, this refers to an instance of a class that manages the dataframe and column generation.
* **n** (int, optional): The number of rows for the generated columns. If not provided, it defaults to the length of the existing dataframe or 10,000 on initialization.
* **num\_cols** (int, optional): The number of numerical columns to add. Defaults to 0 if not provided.
* **cat\_cols** (int, optional): The number of categorical columns to add. Defaults to 0 if not provided.
* **n\_layers** (int, optional): The number of layers for hierarchical generation. Defaults to None.

**Returns:**

* **self** (object): The modified object instance with the newly added columns.