**Data Manipulation**

**Clean Missing Data**

Data scientists often check data for missing values and then perform various operations to fix the data or insert new values. The goal of such cleaning operations is to prevent problems caused by missing data that can arise when training a model.

This module supports multiple type of operations for "cleaning" missing values, including:

* Replacing missing values with a placeholder, mean, or other value
* Completely removing rows and columns that have missing values
* Inferring values based on statistical methods

Using this module does not change your source dataset. Instead, it creates a new dataset in your workspace that you can use in the subsequent workflow. You can also save the new, cleaned dataset for reuse.

This module also defines the transformation used to clean the missing values. Using the Apply Transformation module, you can re-use this transformation on other datasets with the same schema.

**How to use Clean Missing Data**

This module lets you define a cleaning operation. You can also save the cleaning operation so that you can apply it later to new data. See the following links for a description of how to create and save a cleaning process:

* To replace missing values
* To apply a cleaning transformation to new data

**Replace missing values**

Each time that you apply the Clean Missing Data module to a set of data, the same cleaning operation is applied to all columns that you select. Therefore, if you need to clean different columns using different methods, use separate instances of the module.

1. Add the Clean Missing Data module to your experiment, and connect the dataset that has missing values.
2. For Columns to be cleaned, choose the columns that contain the missing values you want to change. You can choose multiple columns, but you must use the same replacement method in all selected columns. Therefore, typically you need to clean string columns and numeric columns separately.

For example, to check for missing values in all numeric columns:

* + Open the Column Selector, and select WITH RULES.
  + For BEGIN WITH, select NO COLUMNS.

You can also start with ALL COLUMNS and then exclude columns. Initially, rules are not shown if you first click ALL COLUMNS, but you can click NO COLUMNS and then click ALL COLUMNS again to start with all columns and then filter out (exclude) columns based on the name, data type, or columns index.

* + For Include, select Column type from the dropdown list, and then select Numeric, or a more specific numeric type.

Any cleaning or replacement method that you choose must be applicable to all columns in the selection. If the data in any column is incompatible with the specified operation, the module returns an error and stops the experiment.

1. For the Minimum missing value ratio, specify the minimum number of missing values required for the operation to be performed.

You use this option in combination with the Maximum missing value ratio to define the conditions under which a cleaning operation is performed on the dataset. If there are too many or too few rows that are missing values, the operation cannot be performed.

The number you enter represents the ratio of missing values to all values in the column. By default, the Minimum missing value ratio property is set to 0. This means that missing values are cleaned even if there is only one missing value. For an example of how to use this option, see Setting a Threshold for Cleaning Operations.

1. For the Maximum missing value ratio, specify the maximum number of missing values that can be present for the operation to be performed.

For example, you might want to perform missing value substitution only if 30% or fewer of the rows contain missing values, but leave the values as-is if more than 30% of rows have missing values.

You define the number as the ratio of missing values to all values in the column. By default, the Maximum missing value ratio is set to 1. This means that missing values are cleaned even if 100% of the values in the column are missing.

1. For Cleaning Mode, select one of the following options for replacing or removing missing values:
   * ***Replace using MICE****:* For each missing value, this option assigns a new value, which is calculated by using a method described in the statistical literature as "Multivariate Imputation using Chained Equations" or "Multiple Imputation by Chained Equations". With a multiple imputation method, each variable with missing data is modeled conditionally using the other variables in the data before filling in the missing values. In contrast, in a single imputation method (such as replacing a missing value with a column mean) a single pass is made over the data to determine the fill value.

All imputation methods introduce some error or bias, but multiple imputation better simulates the process generating the data and the probability distribution of the data.

* + **Custom substitution value:** Use this option to specify a placeholder value (such as a 0 or NA) that applies to all missing values. The value that you specify as a replacement must be compatible with the data type of the column.
  + ***Replace with mean:*** Calculates the column mean and uses the mean as the replacement value for each missing value in the column. This applies only to columns that have Integer, Double, or Boolean data types.
  + ***Replace with median***: Calculates the column median value, and uses the median value as the replacement for any missing value in the column. This applies only to columns that have Integer or Double data types.
  + ***Replace with mode:*** Calculates the mode for the column, and uses the mode as the replacement value for every missing value in the column. Applies to columns that have Integer, Double, Boolean, or Categorical data types.
  + ***Remove entire row:*** Completely remove any row in the dataset that has one or more missing values. This is useful if the missing value can be considered randomly missing.
  + ***Remove entire column:*** Completely removes any column in the dataset that has one or more missing values.
  + ***Replace using Probabilistic PCA:*** Replace the missing values by using a linear model that analyzes the correlations between the columns and estimates a low-dimensional approximation of the data, from which the full data is reconstructed. The underlying dimensionality reduction is a probabilistic form of Principal Component Analysis (PCA), and it implements a variant of the model proposed in the Journal of the Royal Statistical Society, Series B 21(3), 611–622 by Tipping and Bishop. Compared to other options, such as Multiple Imputation using Chained Equations (MICE), this option has the advantage of not requiring the application of predictors for each column. Instead, it approximates the covariance for the full dataset. Therefore, it might offer better performance for datasets that have missing values in many columns.The key limitations of this method are that it expands categorical columns into numerical indicators and computes a dense covariance matrix of the resulting data. It also is not optimized for sparse representations. For these reasons, datasets with large numbers of columns and/or large categorical domains (tens of thousands) are not supported due to prohibitive space consumption.

1. The option Replacement value is available if you have selected the option, Custom substitution value. Type a new value to use as the replacement value for all missing values in the column.Note that you can use this option only in columns that have the Integer, Double, Boolean, or Date data types. For date columns, the replacement value can also be entered as the number of 100-nanosecond ticks since 1/1/0001 12:00 A.M.
2. Generate missing value indicator column: Select this option if you want to output some indication of whether the values in the column met the criteria for missing value cleaning. This option is particularly useful when you are setting up a new cleaning operation and want to make sure it works as designed.
3. Run the experiment, or select the Clean Missing Data module and click Run selected.

**Results**

The module returns two outputs:

* **Cleaned dataset:** A dataset comprised of the selected columns, with missing values handled as specified, along with an indicator column, if you selected that option. Columns not selected for cleaning are also "passed through".
* **Cleaning transformation:** A data transformation used for cleaning, that can be saved in your workspace and applied to new data later.

**Apply a saved cleaning operation to new data**

If you need to repeat cleaning operations often, we recommend that you save your recipe for data cleansing as a transform, to reuse with the same dataset. Saving a cleaning transformation is particularly useful if you must frequently re-import and then clean data that has the same schema.

* Add the Apply Transformation module to your experiment.
* Add the dataset you want to clean, and connect the dataset to the right-hand input port.
* Expand the Transforms group in the left-hand pane of Studio (classic). Locate the saved transformation and drag it into the experiment.
* Connect the saved transformation to the left input port of Apply Transformation.

When you apply a saved transformation, you cannot select the columns to which the transformation are applied. That is because the transformation has been already defined and applies automatically to the data types specified in the original operation.

However, suppose you created a transformation on a subset of numeric columns. You can apply this transformation to a dataset of mixed column types without raising an error because the missing values are changed only in the matching numeric columns.

* Run the experiment.

**Issues**

* An error occurs if the mean or median option is used when any string columns are selected. If you need to process columns of different data types, create two instances of Clean Missing Data.
* When replacing missing values with a mean value in columns with the Boolean, Integer, DateTime, or TimeSpan data types, the column is first converted to floating point numbers, the mean is calculated, and then the result is rounded to the nearest value of the original data type.
* When you type a replacement value, the value must be compatible with the data type in the selected column.
* Values of NaN, Inf, and –Inf are allowed for columns where the data type is Double.
* When using the MICE method, the replacement value is predicted by using the trained MICE model.
* Using Clean Missing Data can reset other column types to feature. If your data contains other types of columns, such as labels, use Edit Metadata to correct the column types.

**Restrictions on use of cleaning transformations**

The following restrictions apply when you use a saved transformation (based on Clean Missing Data) to new data:

* A saved transformation cannot generate indicator values, even if this option was used in the original cleaning operation. Consider the indicator values as most useful when testing a new transformation.
* The transformation does not calculate new values based on the new dataset. In other words, if you used Clean Missing Data on Dataset A and generated a mean value of 0.5, that same value would be applied as the mean for replacing missing values in Dataset B, regardless of the actual values in Dataset B.
* The data type of the columns in the new dataset must match the data type of the columns on which the transformation was created. An error is raised if any operations are performed on the column that implicitly changes the data type.

For example, suppose you create a mean for an integer data column [Col1], and save the transformation. Now you want to apply the cleanup transformation to a copy of [Col1] that has been adjusted using a formula, such as ([Col1] /1.5). To ensure that the result is an integer, you round up the result but still get an error when you apply the transformation. However, if you adjust the value using a formula such as ([Col 1] \* 10), no error is raised!

To avoid such issues, use Edit Metadata to explicitly reset the data type to integer. In general, operations in Apply Math Operation module implicitly change numeric columns to double.

**Setting and interpreting threshold values**

When you specify a threshold for cleaning operations using the options Minimum missing value ratio or Maximum missing value ratio, the results can be unexpected or confusing. To illustrate how the options for maximum and minimum missing values work, we have provided some examples from the Automobile Prices sample dataset, which has many columns with missing values.

The following table shows the count of missing values for several columns in that dataset, together with the ratio of missing values computed on the dataset. The ratio of missing values (in the rightmost column) is the value that would be used in evaluating the dataset against the specified threshold values.

Assume that you set the Minimum missing value ratio to 0.019 and set the Maximum missing value ratio to 0.020. Given the following table of values, some columns meet the threshold criteria, and some do not:

* The columns bore and stroke meet the threshold criteria.
* The columns normalized-losses and compression-ratio do not meet the threshold criteria.

|  |  |  |
| --- | --- | --- |
| **Column name** | **Count of missing values** | **Ratio of missing values** |
| Normalized-losses | 41 | 0.2 |
| Bore | 4 | 0.019512195 |
| Stroke | 4 | 0.019512195 |
| Compression Ratio | 0 | 0 |

Because some columns in the selection did not meet the specified criteria, no cleaning operation was performed on any column. To help you figure out what happened, the module returns the value FALSE in the two indicator columns, bore\_IsMissing and stroke\_IsMissing.

However, if you change the threshold back to the default values of 0 for the Minimum missing value ratio and 1 for the Maximum missing value ratio, an indicator column is returned for all selected columns, and the specified operation is performed.

**Remove Duplicate Rows**

This describes how to use the **Remove Duplicate Rows** module in Machine Learning Studio (classic), to remove potential duplicates from a dataset.

For example, assume your data looks like the following, and represents multiple records for patients.

| **PatientID** | **Initials** | **Gender** | **Age** | **Admitted** |
| --- | --- | --- | --- | --- |
| 1 | F.M. | M | 53 | Jan |
| 2 | F.A.M. | M | 53 | Jan |
| 3 | F.A.M. | M | 24 | Jan |
| 3 | F.M. | M | 24 | Feb |
| 4 | F.M. | M | 23 | Feb |
|  | F.M. | M | 23 |  |
| 5 | F.A.M. | M | 53 |  |

Clearly, this example has multiple columns with potentially duplicate data. Whether they are actually duplicates depends on your knowledge of the data.

* For example, you might know that many patients have the same name. You wouldn't eliminate duplicates using any name columns, only the **ID** column. That way, only the rows with duplicate ID values are filtered out, regardless of whether the patients have the same name or not.
* Alternatively, you might decide to allow duplicates in the ID field, and use some other combination of fileds to find unique records, such as first name, last name, age, and gender.

To set the criteria for whether a row is duplicate or not, you specify a single column or a set of columns to use as **keys**. Two rows are considered duplicates only when the values in **all** key columns are equal.

When you run the module, it creates a candidate dataset, and returns a set of rows that have no duplicates across the set of columns you specified.

**How to use Remove Duplicate Rows**

1. Add the module to your experiment. You can find the **Remove Duplicate Rows** module under **Data Transformation**, **Manipulation**.
2. Connect the dataset that you want to check for duplicate rows.
3. In the **Properties** pane, under **Key column selection filter expression**, click **Launch column selector**, to choose columns to use in identifying duplicates.

In this context, **Key** does not mean a unique identifier. All columns that you select using the Column Selector are designated as **key columns**. All un-selected columns are considered non-key columns. The combination of columns that you select as keys determines the uniqueness of the records.

Examples:

* + "I want to ensure that IDs are unique": Choose only the ID column.
  + "I want to ensure that the combination of first name, last name, and ID is unique": Select all three columns.

1. Use the **Retain first duplicate row** checkbox to indicate which row to return when duplicates are found:
   * If selected, the first row is returned and others discarded.
   * If you uncheck this option, the last duplicate row is kept in the results, and others are discarded.
2. Run the experiment, or click the module and select **Run Selected**.
3. To review the results, right-click the module, select **Results dataset**, and click **Visualize**.

**Missing values**

The input dataset might have missing values in non-key columns and key columns. These rules apply to missing values:

* A missing value is considered a valid value in key columns. Missing values can be present in both key.
* In a sparse dataset, the missing value is considered equal only if it equals the default representation of a sparse value.
* In key columns, a missing value is considered equal to other missing values, but not equal to non-missing values.

**Convert to Indicator Values**

The purpose of this module is to convert columns that contain categorical values into a series of binary indicator columns that can more easily be used as features in a machine-learning model.

**How to configure Convert to Indicator Values**

* Add the Convert to Indicator Values module to your Machine Learning experiment, and connect it to the dataset containing the columns you want to convert. You can find this module under Data Transformations, in the Manipulation category.
* Use the Column Selector to choose one or more categorical columns.

To ensure that the columns you select are categorical, use Edit Metadata before Convert to Indicator Values in your experiment, to mark the target column as categorical.

* Select the Overwrite categorical columns option if you want to output only the new Boolean columns.

By default, this option is off, which lets you see the categorical column that is the source, together with the related indicator columns.

* Run the experiment.

**Results**

For example, suppose you have a column with scores that indicate whether a server has a high, medium or low probability of failure.

|  |  |
| --- | --- |
| Server ID | Failure score |
| 10301 | Low |
| 10302 | Medium |
| 10303 | High |

When you apply Convert to Indicator Values, the single column of labels is converted into multiple columns containing Boolean values:

|  |  |  |  |
| --- | --- | --- | --- |
| Server ID | Failure score - Low | Failure score - Medium | Failure score - High |
| 10301 | 1 | 0 | 0 |
| 10302 | 0 | 1 | 0 |
| 10303 | 0 | 0 | 1 |

Here is how the conversion works:

* In the Failure score column that describes risk, there are only three possible values (High, Medium, and Low), and no missing values. Therefore exactly three new columns are created.
* The new indicator columns are named based on the column headings and values of the source column, using this pattern: <source column>- <data value>.
* There should be a 1 in exactly one indicator column, and 0 in all other indicator columns. That is because each server can have only one risk rating.

You can now use the three indicator columns as features and analyze their correlation with other properties that are associated with different risk level

**Edit Metadata**

This describes how to use the Edit Metadata module in Machine Learning Studio (classic) to change metadata that is associated with columns in a dataset. The values and the data types in the dataset are not altered; what changes is the metadata inside Machine Learning that tells downstream components how to use the column.

Typical metadata changes might include:

* Treating Boolean or numeric columns as categorical values
* Indicating which column contains the class label, or the values you want to categorize or predict
* Marking columns as features
* Changing date/time values to a numeric value, or vice versa
* Renaming columns

Use Edit Metadata any time you need to modify the definition of a column, typically to meet requirements for a downstream module. For example, some modules can work only with specific data types, or require flags on the columns, such as IsFeature or IsCategorical.

After performing the required operation, you can reset the metadata to its original state.

**How to configure Edit Metadata**

1. In Machine Learning Studio (classic), add Edit Metadata module to your experiment and connect the dataset you want to update. You can find it under Data Transformation, in the Manipulate category.
2. Click Launch the column selector and choose the column or set of columns to work with. You can choose columns individually, by name or index, or you can choose a group of columns, by type.
3. Select the Data type option if you need to assign a different data type to the selected columns. Changing the data type might be needed for certain operations: for example, if your source dataset has numbers handled as text, you must change them to a numeric data type before using math operations.

* The data types supported are String, Integer, Floating point, Boolean, DateTime, and TimeSpan.
* If multiple columns are selected, you must apply the metadata changes to all selected columns. For example, let's say you choose 2-3 numeric columns. You could change them all to a string data type, and rename them in one operation. However, you can't change one column to a string data type and another column from a float to an integer.
* If you do not specify a new data type, the column metadata is unchanged.
* Changes of data type affect only the metadata that is associated with the dataset and how the data is handled in downstream operations. The actual column values are not altered unless you perform a different operation (such as rounding) on the column. You can recover the original data type at any time by using Edit Metadata to reset the column data type.

1. Select the Categorical option to specify that the values in the selected columns should be treated as categories.

For example, you might have a column that contains the numbers 0,1 and 2, but know that the numbers actually mean "Smoker", "Non smoker" and "Unknown". In that case, by flagging the column as categorical you can ensure that the values are not used in numeric calculations, only to group data.

1. Use the Fields option if you want to change the way that Machine Learning uses the data in a model.

* Feature: Use this option to flag a column as a feature, for use with modules that operate only on feature columns. By default, all columns are initially treated as features.
* Label: Use this option to mark the label (also known as the predictable attribute, or target variable). Many modules requires that at least one (and only one) label column be present in the dataset.

In many cases, Machine Learning can infer that a column contains a class label, but by setting this metadata you can ensure that the column is identified correctly. Setting this option does not change data values, only the way that some machine learning algorithms handle the data.

* Weight: Use this option with numeric data to indicate that column values represents weights for use in machine learning scoring or training operations. Only one weight column can be present in a dataset, and the column must be numeric. This option works only in these models: Two-Class Logistic Regression, Two-Class Support Vector Machine, and Two-Class Neural Network.

1. Use the following options to clear previous selections and restore metadata to the default values.

* Clear feature: Use this option to remove the feature flag.

Because all columns are initially treated as features, for modules that perform mathematical operations, you might need to use this option to prevent numeric columns from being treated as variables.

* Clear label: Use this option to remove the label metadata from the specified column.
* Clear score: Use this option to remove the score metadata from the specified column.

Currently the ability to explicitly mark a column as a score is not available in Machine Learning. However, some operations result in a column being flagged as a score internally. Also, a custom R module might output score values.

* Clear weight: Use this option to remove the weight metadata from the specified column.

1. For New column names, type the new name of the selected column or columns.

* Column names can use only characters that are supported by the UTF-8 encoding. Empty strings, nulls, or names consisting entirely of spaces are not allowed.
* To rename multiple columns, type the names as a comma-separated list in order of the column indices.
* All selected columns must be renamed. You cannot omit or skip columns.

1. Run the experiment.

**Group Categorical Values**

This describes how to use the Group Categorical Values module in Machine Learning Studio (classic), to create an in-place lookup table.The typical use for grouping categorical values is to merge multiple string values into a single new level. For example, you might assign individual postal codes in a region to a single regional code, or group multiple products under one category.

To use this module, you type the lookup values you want to use, and map existing values to the replacement values. You can create groupings only for categorical columns, not to columns of numeric type or columns designated as labels or features.Any column values that are not explicitly mapped to a new level are assigned to a default level. For example, if you did not map all the individual postal codes, they would be grouped in a level for unmapped values, which you might name Unknown.

**How to use Group Categorical Values**

We recommend that you prepare the list of existing values, and the new categories, beforehand. For each category, you should prepare a new category name, and a comma-separated list of values to include in the category.

1. Add the Group Categorical Values module to your experiment. You can find the module under Data Transformation, Manipulation.
2. Connect a dataset that has the values you want to transform.
3. In the Properties pane of Group Categorical Values, use the Column Selector to choose the column that has the levels you want to reduce.

* We recommend that you click BEGIN WITH and NO COLUMNS to start, and then add columns by name. Otherwise, too many columns might be added as candidates, leading to an error.
* The column must be a categorical column. If it is not, add Edit Metadata upstream, and change the column type.
* Be sure to remove from the input any columns to which string replacement should not be applied.

1. For Output mode, indicate whether you want to output just the new levels, or append the changes to see the original column, with the replacements side by side. The default, ResultOnly, shows only the new values. The Inplace option replaces the existing column values with the new levels.
2. For the Default level name, type a string value to use as the replacement for all values that are not explicitly mapped. You might use something such as "Unknown" or "Default".
3. For a New number of levels, type a number that indicates the total number of new categories (levels), including the default level for unmapped values.
4. For the Name of new level 1, provide the new group name for the first category.
5. In the text box that immediately follows, a Comma-separated list of old levels to map to new level 1, type or paste an exhaustive list of all values to map to the new level. Wildcard characters and regular expressions are not allowed.
6. Continue to type new level names and type or paste values that should be mapped to the new level.

We recommend that you save your list of values in a separate file as you are working. If you change the number of levels, any strings that you previously typed are removed, and you must start over. However, if you are editing a module that was previously saved, you can revert to the original settings.

1. Run the experiment.

**Results**

To view the results, right-click the Group Categorical Values module, select Results dataset, and click Visualize.

**SMOTE**

This article describes how to use the SMOTE module in Machine Learning Studio (classic) to increase the number of underrepresented cases in a dataset used for machine learning. SMOTE is a better way of increasing the number of rare cases than simply duplicating existing cases.

You connect the SMOTE module to a dataset that is imbalanced. There are many reasons why a dataset might be imbalanced: the category you are targeting might be very rare in the population, or the data might simply be difficult to collect. Typically, you use SMOTE when the class you want to analyze is under-represented.

The module returns a dataset that contains the original samples, plus an additional number of synthetic minority samples, depending on the percentage you specify.

***SMOTE:***

SMOTE stands for Synthetic Minority Oversampling Technique. This is a statistical technique for increasing the number of cases in your dataset in a balanced way. The module works by generating new instances from existing minority cases that you supply as input. This implementation of SMOTE does not change the number of majority cases.

The new instances are not just copies of existing minority cases; instead, the algorithm takes samples of the feature space for each target class and its nearest neighbors and generates new examples that combine features of the target case with features of its neighbors. This approach increases the features available to each class and makes the samples more general.

SMOTE takes the entire dataset as input, but it increases the percentage of only the minority cases. For example, suppose you have an imbalanced dataset where just 1% of the cases have the target value A (the minority class), and 99% of the cases have the value B. To increase the percentage of minority cases to twice the previous percentage, you would enter 200 for SMOTE percentage in the module's properties.

**How to configure SMOTE**

1. Add the SMOTE module to your experiment. You can find the module under Data Transformation modules, in the manipulation category.
2. Connect the dataset you want to boost. If you want to specify the feature space for building the new cases, either by using only specific columns or by excluding some, use the Select Columns in the Dataset module to isolate the columns you want to use before using SMOTE. Otherwise, the creation of new cases using SMOTE is based on all the columns that you provide as inputs.
3. Ensure that the column containing the label, or target class, is marked as such. If there is no label column, use the Edit Metadata module to select the column that contains the class labels, and select Label from the Fields dropdown list.
4. The SMOTE module automatically identifies the minority class in the label column and then gets all examples for the minority class.
5. In the SMOTE percentage option, type a whole number that indicates the target percentage of minority cases in the output dataset. For example:

* You type 0 (%). The SMOTE module returns exactly the same dataset that you provided as input, adding no new minority cases. In this dataset, the class proportion has not changed.
* You type 100 (%). The SMOTE module generates new minority cases, adding the same number of minority cases that were in the original dataset. Because SMOTE does not increase the number of majority cases, the proportion of cases of each class has now changed.
* You type 200 (%). The module doubles the percentage of minority cases compared to the original dataset. This does not result in having twice as many minority cases as before. Rather, the size of the dataset is increased in such a way that the number of majority cases stays the same, and the number of minority cases is increased till it matches the desired percentage value.

1. Use the Number of nearest neighbors option to determine the size of the feature space that the SMOTE algorithm uses when in building new cases. A nearest neighbor is a row of data (a case) that is very similar to some target case. The distance between any two cases is measured by combining the weighted vectors of all features.

* By increasing the number of nearest neighbors, you get features from more cases.
* By keeping the number of nearest neighbors low, you use features that are more like those in the original sample.

1. Type a value in the Random seed textbox if you want to ensure the same results overruns of the same experiment, with the same data. Otherwise, the module generates a random seed based on processor clock values when the experiment is deployed, which can cause slightly different results over runs.
2. Run the experiment.

The output of the module is a dataset containing the original rows plus some number of added rows with minority cases.