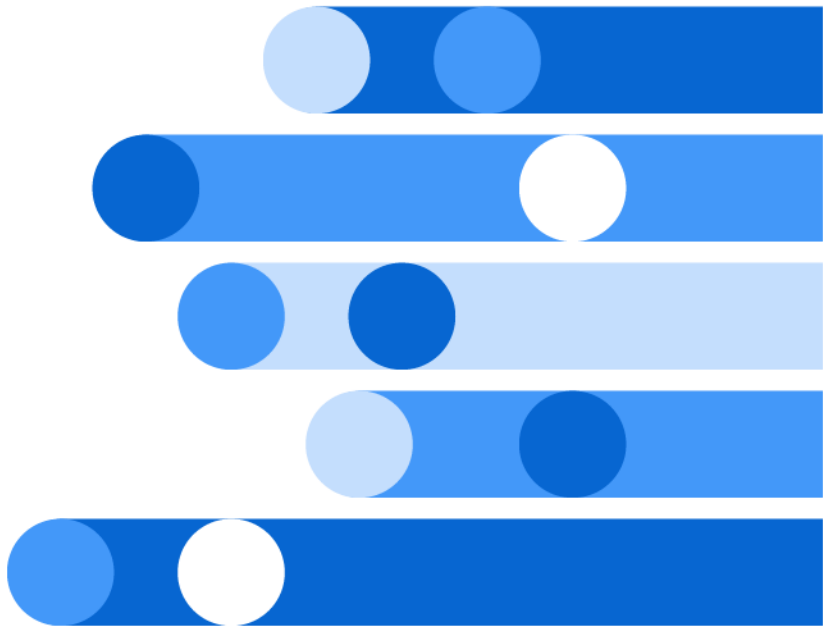
SAS® Viya® Workbench: Machine Learning Python API Guide



SAS® Documentation
July 29, 2026

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2024. *SAS® Viya® Workbench: Machine Learning Python API Guide*. Cary, NC: SAS Institute Inc.

SAS® Viya® Workbench: Machine Learning Python API Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

August 2026

v_001-P1:vwbpymlpg

Contents

1

Data Processing

Classes

Class Name	Description
Cardinality	The Cardinality class determines the cardinality or limited cardinality of a variable.
CategoricalEncoder	Represents the CategoricalEncoder class, which enables users to group and encode categorical variables by using unsupervised and supervised grouping techniques.
Discretize	Represents the Discretize class, which enables users to perform supervised and unsupervised variable discretization.
Impute	Represents the Impute class, which enables users to impute both interval and nominal variables in various ways.

Cardinality

The Cardinality class determines the cardinality or limited cardinality of a variable.

API: Cardinality

class sasviya.ml.cardinality.Cardinality(*, level_order='ASC', level_threshold=20, verbose=0)

The cardinality of a variable is the number of its distinct values, and the limited cardinality of a variable is the number of its distinct values that do not exceed a specified threshold.

Parameters

level_order : {"ASC", "DESC"}, optional
Specifies the order in which variable levels are sorted. The level order determines how variable levels are sorted in the details output table as well as in the list that is returned when the get_variable_levels() method is called. The default is "ASC".

level_threshold : int in [5,254], optional
Specifies the maximum number of levels of the variables to consider in the input data. When the summarize() method is run, only up to the specified level_threshold value of variable levels, or distinctive values, are considered. If the number of levels exceeds the level_threshold value for a variable, the _MORE_ column in the summary output table will be 'Y'. The default is 20. Allowed values: [5,254].

verbose : int in {0,1,2}, optional
Specifies the amount of information to print when the summarize() method is run. The default is 0. A value of 1 prints the cardinality summary table to the log. A value of 2 prints the cardinality summary and details tables to the log.

Attributes

details_ : Results
Specifies the details table that contains information about the model and its results.

variables_in_ : list(str)
Specifies the variables to include in the summary.

n_variables_in_ : int
Specifies the number of variables.

level_threshold_in_ : int
Specifies the maximum number of variable levels.

level_order_in_ : str
Specifies the order in which variable levels are sorted.

nominal_variables_ : list(str)
Specifies the names of nominal variables to include.

interval_variables_ : list(str)
Specifies the names of interval variables to include.

binary_variables_ : list(str)
Specifies the names of binary variables to include.

id_variables_ : list(str)
Specifies the names of ID variables to include.

Methods

Method Name	Description
customize_results	Generate plots for all actions that support graphics, as well as filter tables from the result list and create custom result orders.
get_details_table	Return the cardinality details table.
get_levels	Return the levels of a variable.
get_n_levels	Return the cardinality of a variable.
get_params	Gets parameters for this model.
get_summary_table	Return the cardinality summary table.
set_params	Updates the parameters of this summarizer.
summarize	Summarize the cardinality of the input data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generate plots for all actions that support graphics, as well as filter tables from the result list and create custom result orders.

Parameters

plots : str or str list
Request plots. This can be a single plot or list of plots. This request can also be "all" or "none". Parenthesized arguments are also allowed with each plot request. In addition, automatic plots from models can be disabled by setting the environment variable AUTO_GRAPHICS=false. However, this variable does not disable automatic plot generation through the customize_results() method. You will need to set plots="none" on the method to guaranteed that only tables are returned. For more information, see *SAS Viya: Managing Plots in Python Automatic Graphics <https://documentation.sas.com/doc/en/pyautogrplots/v_001/p11t293hhzoo85n1r73e8aring2y.htm>*.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default dots per inch of the graph. The default is 96 dpi.

theme : {"light", "dark", "opal", "midnight", "raven", "htmlencore", "highcontrast", "grayscale", monochrome"}
Overrides the default theme used in the graph. The default is "light". The session default can be set by setting the GRAPHICS_THEME environment variable.

path : str
If specified, the plot results are written to the specified path in addition to the notebook.

output_type : {"svg", "png"}
Overrides the default output type. The default type is "svg", which contains image map information for tooltips. The "png" output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the display results. This can be a keyword ("none", "all"), a single table, or list of tables. The default is "all", which keeps all tables. The "none" keyword drops all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names can seen in the top-left corner of all displayed results. If this option is specified with the "exclude_tables" option, then the "include_tables" option has precedence.

exclude_tables : str or str list
Excludes tables from the display results. This can be a keyword ("none", "graph_tables", "all"), a single table, or list of tables. The default is "none", so that all tables are shown. The "graph_tables" keyword drops all tables that were used to produce a graph. The "all" keyword drops all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names can seen in the top-left corner of all displayed results. If this option is specified with the "include_tables" option, then the "include_tables" option has precedence.

order : str or str list
Orders plot and tabular results for display. Any result not specified in the option will be excluded in the new results. The names for the results can seen in the top-left corner of all displayed results. If this option is specified along with the “include_tables” or “exclude_tables” option, those options are applied BEFORE the ordering is done. If the ordered list contains no results, then the “order” option is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, The CustomResults are typically assigned to a variable for later usage or passed passed directly into the display() method for viewing in a Jupyter notebook cell.

get_details_table

```
get_details_table()
```

Return the cardinality details table.

Returns

Returns an array-like object of the same type as the summarized data.

get_levels

```
get_levels(var)
```

Return the levels of a variable.

Parameters

variable : str
Specifies the variable name.

Returns

list(str)

get_n_levels

```
get_n_levels(var)
```

Return the cardinality of a variable.

Parameters

variable : str
Specifies the variable name.

Returns

int

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

get_summary_table

```
get_summary_table()
```

Return the cardinality summary table.

Returns

Returns an array-like object of the same type as the summarized data.

set_params

```
set_params(**params)
```

Updates the parameters of this summarizer.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self

summarize

```
summarize(X)
```

Summarize the cardinality of the input data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform cardinality analysis on.

Returns

self

CategoricalEncoder

Represents the CategoricalEncoder class, which enables users to group and encode categorical variables by using unsupervised and supervised grouping techniques.

API: CategoricalEncoder

class sasviya.ml.preprocess.CategoricalEncoder(*, strategy='group_rare', n_bins=None, max_bins=None, min_bins=None, min_bin_obs=None, min_bin_percent=5, tree_criteria='gain_ratio', bin_missing=False, preprocess_rare=False, rare_threshold=None, rare_threshold_percent=5)

Parameters

strategy : {'group_rare', 'label', 'classification_tree', 'regression_tree', 'woe'}, default='group_rare'
Specifies which of the following grouping or encoding strategies to use:

- *'group_rare'*: groups rare levels of the analysis variable. This is an unsupervised technique.
- *'label'*: uses one-hot encoding of categorical variables. This is an unsupervised technique.
- *'classification_tree'* : groups on the basis of a one-level decision tree. This is a supervised technique.
- *'regression_tree'*: groups on the basis of a one-level regression tree. This is a supervised technique.
- *'woe'*: groups on the basis of maximizing the information value (IV). This is a supervised technique. The 'event' parameter in the fit and fit_transform methods is required for this strategy.

n_bins : int or list(int), optional
Specifies a list of the number of bins to create for each variable. This parameter can override the following parameters: *max_bins*, *min_bins*, *min_bin_obs*, and *min_bin_percent*.

max_bins : int or list(int), optional
Specifies the maximum number of bins for supervised strategies. The default is computed as log2 of the number of distinct values.

min_bins : int or list(int), optional
Specifies the minimum number of bins for supervised strategies.

min_bin_obs : int, optional
Specifies the minimum number of observations to include in a bin when the *strategy* parameter value is 'classification_tree', 'regression_tree', or 'woe'. This parameter takes precedence over the *min_bin_percent* parameter.

min_bin_percent : double, optional
Specifies the minimum percentage of all observations to include in a bin when the *strategy* parameter value is 'classification_tree', 'regression_tree', or 'woe', with the range (0,50).

tree_criteria : {'gain', 'gain_ratio', 'gini', 'sse'}, default='gain_ratio'|'sse'
Specifies tree criteria for the *strategy* parameter value 'classification_tree' or 'regression_tree'. The default criterion for 'classification_tree' is 'gain_ratio', and the default criterion for 'regression_tree' is 'sse'.

bin_missing : boolean, optional
Bins missing values in a separate bin.

preprocess_rare : boolean, optional
Groups rare levels before implementing the specified binning strategy. This parameter is not applicable when the *strategy* parameter value is 'label'.

rare_threshold : int, optional
Specifies the rare frequency threshold. This parameter takes precedence over the *rare_threshold_percent* parameter. The *preprocess_rare* parameter value must be True.

rare_threshold_percent : float, optional
Specifies the rare threshold percentage, with the range (0,100). The *preprocess_rare* parameter value must be True.

Attributes

details_ : Results
Stores information about the fitting process and its results.

n_features_in_ : int
Stores the number of input features.

feature_names_in_ : list(str)
Stores the list of input features.

n_bins_ : list(int)
Store the list of the number of bins per feature. Not available when the *strategy* parameter value is 'label'.

n_levels_ : list(int)
Stores the number of levels per feature.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the data.
fit_transform	Fits and transforms the data.
get_feature_names_out	Retrieves the output feature names.
get_params	Gets parameters for this model.
save	Saves the model to a file.
set_params	Updates the parameters of the model.
transform	Transforms the data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list

Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does

not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, weight=None, y=None, event=None)
```

Fits the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a fit analysis on.

y : array-like of shape (n_samples,)
Specifies the target labels.

weight : str, optional
Specifies a column from a NumPy ndarray or a Pandas DataFrame.

event : str, optional
Specifies the event level for the target that is used in supervised strategies.

Returns

None

fit_transform

```
fit_transform(X, weight=None, y=None, event=None, transform_options=None)
```

Fits and transforms the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a fit analysis on.

y : array-like of shape (n_samples,)
Specifies the target labels.

weight : str, optional
Specifies a column from a NumPy ndarray or a Pandas DataFrame.

event : str, optional
Specifies the event level for the target that is used in supervised strategies.

transform_options : dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

result : array-like
Returns an array-like result of transformed values.

get_feature_names_out

```
get_feature_names_out()
```

Retrieves the output feature names.

Returns

list(str)
Returns a list of strings.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

transform

```
transform(X, options=None)
```

Transforms the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a transform analysis on.

options : dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

result : array-like
Returns an array-like result of transformed values.

Discretize

Represents the Discretize class, which enables users to perform supervised and unsupervised variable discretization.

API: Discretize

class sasviya.ml.preprocess.Discretize(*, strategy='uniform', n_bins=[5], cut_points=None, max_bins=None, min_bins=None, min_bin_obs=None, min_bin_percent=5, tree_criteria='gain_ratio', bin_missing=False, alpha=0.05, initial_strategy='uniform', initial_bins=100)

Parameters

strategy : {'bucket', 'uniform', 'quantile', 'custom', 'chimerge', 'mdlp', 'classification_tree', 'regression_tree', 'woe'}, default='bucket'|'uniform'
Specifies which of the following discretization strategies to use:

- *'bucket' or 'uniform*: creates bins of equal width.
- *'quantile'*: creates bins of equal frequency.
- *'custom'*: creates bins according to the user-specified cutpoints. The *cut_points* parameter is required for this strategy.
- *'chimerge'*: creates bins on the basis of chi-square merging of neighboring bins. This is a bottom-up supervised discretization technique.

- *'mdlp'*: creates bins on the basis of the minimum description length. This is a top-down supervised discretization technique.
- *'classification_tree'* : creates bins on the basis of a one-level decision tree. This is a top-down supervised discretization technique.
- *'regression_tree'*: creates bins on the basis of a one-level regression tree. This is a top-down supervised discretization technique.
- *'woe'*: creates bins on the basis of the WOE criterion. This is a top-down supervised discretization technique. The *event* parameter in the fit and fit_transform methods is required for this strategy.

n_bins : int or list(int), optional
Specifies a list of the number of bins to create for each variable. This parameter can override the following parameters: *max_bins*, *min_bins*, *min_bin_obs*, and *min_bin_percent*. The default is 5 for unsupervised strategies.

initial_strategy : {'bucket', 'quantile'}, default='bucket', optional
Specifies the initial binning method for interval inputs to start supervised strategies.

initial_bins : int, optional
Specifies the initial number of bins for interval inputs to start supervised strategies.

cut_points : list(int), optional
Specifies the user-provided cutpoints when the *strategy* parameter value is 'custom'.

max_bins : int or list(int), optional
Specifies the maximum number of bins for supervised strategies. The default is 5 for supervised strategies.

min_bins : int or list(int), optional
Specifies the minimum number of bins for supervised strategies. The default is 1 for supervised strategies.

min_bin_obs : int, optional
Specifies the minimum number of observations to include in a bin when the *strategy* parameter value is 'classification_tree', 'regression_tree', or 'woe'. This parameter takes precedence over the *min_bin_percent* parameter.

min_bin_percent : double, optional
Specifies the minimum percentage of all observations to include in a bin when the *strategy* parameter value is 'classification_tree', 'regression_tree', or 'woe', with the range (0,50).

tree_criteria : {'gain', 'gain_ratio', 'gini', 'sse'}, default='gain_ratio'|'sse'
Specifies tree criteria for the *strategy* parameter value 'classification_tree' or 'regression_tree'. The default criterion for 'classification_tree' is 'gain_ratio', and the default criterion for 'regression_tree' is 'sse'.

bin_missing : boolean, optional
Bins missing values in a separate bin.

alpha : double, optional
Specifies the significance level for the *strategy* parameter value 'chimerge', with the range (0,1).

Attributes

details_ : Results
Stores information about the fitting process and its results.

n_features_in_ : int
Stores the number of input features.

feature_names_in_ : list(str)
Stores the list of input features.

n_bins_ : list(int)
Stores the list of the number of bins per feature.

bin_edges_ : ndarray of ndarray of shape (n_features,)
Stores an array of bin edges per feature.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the data.
fit_transform	Fits and transforms the data.
get_feature_names_out	Retrieves the output feature names.
get_params	Gets parameters for this model.
save	Saves the model to a file.
set_params	Updates the parameters of the model.
transform	Transforms the data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left

corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, weight=None, y=None, event=None)
```

Fits the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a fit analysis on.

y : array-like of shape (n_samples,)
Specifies the target labels.

weight : str, optional
Specifies a column from a NumPy ndarray or a Pandas DataFrame.

event : str, optional
Specifies the event level for the target that is used in supervised strategies.

Returns

None

fit_transform

```
fit_transform(X, weight=None, y=None, event=None, transform_options=None)
```

Fits and transforms the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a fit analysis on.

y : array-like of shape (n_samples,)
Specifies the target labels.

weight : str, optional
Specifies a column from a NumPy ndarray or a Pandas DataFrame.

event : str, optional
Specifies the event level for the target that is used in supervised strategies.

transform_options : dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

result : array-like
Returns an array-like result of transformed values.

get_feature_names_out

```
get_feature_names_out()
```

Retrieves the output feature names.

Returns

list(str)
Returns a list of strings.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

transform

```
transform(X, options=None)
```

Transforms the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a transform analysis on.

options : dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

result : array-like
Returns an array-like result of transformed values.

Impute

Represents the Impute class, which enables users to impute both interval and nominal variables in various ways.

API: Impute

class sasviya.ml.preprocess.Impute(*, strategy='mean', values_numeric=None, values_string=None)

Parameters

strategy : {'mean', 'median', 'min', 'max', 'midrange', 'mode', 'most_frequent', 'value', 'constant'}, default='mean'
Specifies which of the following imputation strategies to use:

- *'mean'*: replaces missing values with the mean of the column.
- *'median'*: replaces missing values with the median of the column.
- *'min'*: replaces missing values with the minimum of the column.
- *'max'*: replaces missing values with the maximum of the column.
- *'midrange'*: replaces missing values with the midrange of the column.
- *'mode'* or *'most_frequent'*: replaces missing values with the mode of the column.
- *'value'*: replaces missing values with a specified value (see the *values_numeric* and *values_string* parameters).
- *'constant'*: replaces missing values with a constant value (see the *values_numeric* and *values_string* parameters).

values_numeric : float or list(float), optional
Specifies the value to use in numeric columns when the *strategy* parameter value is 'value' or 'constant'.

values_string : str or list(str), optional
Specifies the value to use in character columns when the *strategy* parameter value is 'value' or 'constant'.

Attributes

details_ : Results
Stores information about the fitting process and its results.

n_features_in_ : int
Stores the number of input features.

feature_names_in_ : list(str)
Stores the list of input features.

imputed_values_ : list(float) or list(str)
Stores the list of imputed values.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the data.
fit_transform	Fits and transforms the data.
get_feature_names_out	Retrieves the output feature names.
get_params	Gets parameters for this model.
save	Saves the model to a file.
set_params	Updates the parameters of the model.
transform	Transforms the data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list

Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does

not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, weight=None, y=None)
```

Fits the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a fit analysis on.

weight : str
Specifies a column from a NumPy ndarray or a Pandas DataFrame.

Returns

None

fit_transform

```
fit_transform(X, weight=None, y=None, transform_options=None)
```

Fits and transforms the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a fit analysis on.

weight : str
Specifies a column from a NumPy ndarray or a Pandas DataFrame.

transform_options : dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Returns an array-like result of transformed values.

get_feature_names_out

```
get_feature_names_out()
```

Retrieves the output feature names.

Returns

list(str)
Returns a list of strings.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

transform

```
transform(X, options=None)
```

Transforms the data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform a transform analysis on.

options : dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

result : array-like
Returns an array-like result of transformed values.

2

Model Assessment

Classes

Class Name	Description
Assess	The Assess class enables you to measure and evaluate the performance of supervised learning models.
AssessBias	The AssessBias class calculates bias metrics for predictive models.

Assess

The Assess class enables you to measure and evaluate the performance of supervised learning models.

API: Assess

class sasviya.ml.assess.Assess(*, input=None, y_true=None, pos_label=None, p_var=None, p_pos_label=None, cut_step=None, user_cutoff=None, verbose=0)

The Assess class supports both regression and classification models. It provides the ROC-and-fit-statistics-related metrics.

Parameters

input : str
Specifies the predicted variable to assess. For classification models, it is the predicted probability of the specified target event. For regression models, it is the predicted target.

y_true : str
Specifies the true target variable.

pos_label : str
Specifies the target event for classification model assessment. If the pos_label is not specified, the assessment is for regression models.

p_pos_label : list(str), optional
Specifies the target levels, excluding the pos_label. This parameter is for classification models.

p_var : list(str)
Specifies the predicted probabilities of the p_pos_label. This parameter is for classification models.

cut_step : float, optional
Specifies an interval to generate a set of thresholds between 0 and 1 for evaluating metrics related to confusion matrices. This parameter is for classification models. By default, cut_step=0.01.

user_cutoff : float, optional
Specifies the threshold for evaluating metrics related to confusion matrices. The user_cutoff value is in (0, 1). This is for classification models. By default, user_cutoff=0.5.

verbose : int, optional
Specifies the amount of information to print during summarize(). Allowed values: [0,1,2]. A verbosity level of 1 prints the fit statistic table to the log. A verbosity level of 2 prints the ROC table and the fit statistic table to the log. By default, verbose=0.

Attributes

The following attributes are associated with parameters specified in the constructor:

details_ : Results
Additional information about the summarize process and the final results.

input_in_ : str
The predicted probability variable.

y_true_in_ : str
The specified target.

pos_label_in_ : str
The specified target event.

cut_step_in_ : float
The specified value of option cut_step, default is 0.01.

user_cutoff_in_ : float
The specified value of option user_cutoff, default is 0.5.

The following attributes are associated with regression models:

mean_absolute_error_ : float
The returned mean absolute error for regression models.

mean_squared_error_ : float
The returned mean square error for regression models.

The following attributes are associated with classification models:

accuracy_score_ : float
The classification score.

log_loss_ : float
The Log Loss metric for classification models.

roc_curve_ : array list
The returned array of TPR and FPR.

auc_ : float
The returned area under the curve for classification model.

precision_recall_curve_ : array list
The returned array of the precision score and recall score.

The following attributes are associated with user-defined cutoffs:

f1_score_ : float
The returned f1 score for classification models.

confusion_matrix_ : array
The returned 2x2 matrix for the specified event and threshold.

precision_score_ : float
The returned precision score for classification models.

recall_score_ : float
The returned recall score for classification models.

classification_report_ : dict
The returned f1 score, precision, and recall for the target event and overall precision score.

Methods

Method Name	Description
customize_results	Generate plots for all actions that support graphics, as well as filter tables from the result list and create custom result orders.
get_params	Gets parameters for this model.
set_params	Updates the parameters of this summarizer.
summarize	Summarize the scored data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generate plots for all actions that support graphics, as well as filter tables from the result list and create custom result orders.

Parameters

plots : str or str list
Request plots. This can be a single plot or list of plots. This request can also be “all” or “none”. Parenthesized arguments are also allowed with each plot request. In addition, automatic plots from models can be disabled by setting the environment variable AUTO_GRAPHICS=false. However, this variable does not disable automatic plot generation through the customize_results() method. You will need to set plots=”none” on the method to guaranteed that only tables are returned. For more information, see *SAS Viya: Managing Plots in Python Automatic Graphics <https://documentation.sas.com/doc/en/pyautogrplots/v_001/p11t293hhzoo85n1r73e8aring2y.htm>*.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default dots per inch of the graph. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme used in the graph. The default is “light”. The session default can be set by setting the GRAPHICS_THEME environment variable.

path : str
If specified, the plot results are written to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the display results. This can be a keyword (“none”, “all”), a single table, or list of tables. The default is “all”, which keeps all tables. The “none” keyword drops all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names can seen in the top-left corner of all displayed results. If this option is specified with the “exclude_tables” option, then the “include_tables” option has precedence.

exclude_tables : str or str list
Excludes tables from the display results. This can be a keyword (“none”, “graph_tables”, “all”), a single table, or list of tables. The default is “none”, so that all tables are shown. The “graph_tables” keyword drops all tables that were used to produce a graph. The “all” keyword drops all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names can seen in the top-left corner of all displayed results. If this option is specified with the “include_tables” option, then the “include_tables” option has precedence.

order : str or str list
Orders plot and tabular results for display. Any result not specified in the option will be excluded in the new results. The names for the results can seen in the top-left corner of all displayed results. If this option is specified along with the “include_tables” or “exclude_tables” option, those options are applied BEFORE the ordering is done. If the ordered list contains no results, then the “order” option is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, The CustomResults are typically assigned to a variable for later usage or passed passed directly into the display() method for viewing in a Jupyter notebook cell.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

set_params

```
set_params(**params)
```

Updates the parameters of this summarizer.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self

summarize

```
summarize(X)
```

Summarize the scored data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to perform assess analysis on

Returns

self

AssessBias

The AssessBias class calculates bias metrics for predictive models.

API: AssessBias

class sasviya.ml.assessbias.AssessBias(*, target, target_type, sensitive_var, pred_vars, levels=None, event=None, reference_level=None, cutoff=None, n_bins=None, depth=None, n_cuts=None, verbose=0)

In predictive modeling, bias occurs when the prediction or performance of a model differs for different values of a given variable, which is referred to here as the "sensitive" variable. Prediction bias is measured by calculating the difference in average model prediction for different values of the sensitive variable. Performance bias is measured by calculating the difference in model performance for different values of the sensitive variable.

The AssessBias class calculates performance statistics and average model predictions for each level of a nominal variable. The difference in these statistics is then calculated either by reporting the maximum pairwise difference among all levels or by comparing each level to a reference group in order to report a single number that represents model bias.

Parameters

target : str
Specifies the target variable.

target_type: {"INTERVAL", "NOMINAL"}
Specifies the target type. The default is INTERVAL.

sensitive_var : str
Specifies the sensitive variable to use in bias calculations.

pred_vars : list(str)
Specifies the list of variables that contain the model's predictions. The order of the variables must match the order that you specify in the levels parameter.

levels : list(str), optional
Specifies the list of formatted values of the target variable. The order of the variables must match the order that you specify in the pred_vars parameter.

event : str, optional
Specifies the formatted value of the target variable that represents the event of interest. For an interval target, this parameter is ignored.

reference_level : str, optional
Specifies the reference level for the sensitive variable.

cutoff : float, optional
Specifies the cutoff for the confusion matrix. The default is 0.5. The allowed values are (0, 1).

n_bins : int, optional
Specifies the number of bins to use in lift calculations. The default is 20. The allowed values are [2, 100].

depth : int, optional
Specifies the depth to use in lift calculations. The default is 10. The allowed values are (0, 100].

n_cuts : int, optional
Specifies the number of cuts to use in the receiver operating characteristic (ROC) calculation. The default is 100. The allowed values are [2, infinity].

verbose : int, optional
Specifies the amount of information to print during summarize(). The default is 0. The allowed values are [0, 1, 2].

Attributes

details_ : Results
The details table that contains information about the model and its results.

Methods

Method Name	Description
customize_results	Generate plots for all actions that support graphics, as well as filter tables from the result list and create custom result orders.
get_params	Gets parameters for this model.
set_params	Updates the parameters of this summarizer.

Method Name	Description
summarize	Assess the bias of the scored input data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generate plots for all actions that support graphics, as well as filter tables from the result list and create custom result orders.

Parameters

plots : str or str list
Request plots. This can be a single plot or list of plots. This request can also be “all” or “none”. Parenthesized arguments are also allowed with each plot request. In addition, automatic plots from models can be disabled by setting the environment variable AUTO_GRAPHICS=false. However, this variable does not disable automatic plot generation through the customize_results() method. You will need to set plots=”none” on the method to guaranteed that only tables are returned. For more information, see *SAS Viya: Managing Plots in Python Automatic Graphics <https://documentation.sas.com/doc/en/pyautogrplots/v_001/p11t293hhzoo85n1r73e8aring2y.htm>*.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default dots per inch of the graph. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme used in the graph. The default is “light”. The session default can be set by setting the GRAPHICS_THEME environment variable.

path : str
If specified, the plot results are written to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the display results. This can be a keyword (“none”, “all”), a single table, or list of tables. The default is “all”, which keeps all tables. The “none” keyword drops all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names can seen in the top-left corner of all displayed results. If this option is specified with the “exclude_tables” option, then the “include_tables” option has precedence.

exclude_tables : str or str list
Excludes tables from the display results. This can be a keyword (“none”, “graph_tables”, “all”), a single table, or list of tables. The default is “none”, so that all tables are shown. The “graph_tables” keyword drops all tables that were used to produce a graph. The “all” keyword drops all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names can seen in the top-left corner of all displayed results. If this option is specified with the “include_tables” option, then the “include_tables” option has precedence.

order : str or str list
Orders plot and tabular results for display. Any result not specified in the option will be excluded in the new results. The names for the results can seen in the top-left corner of all displayed results. If this option is specified along with the “include_tables” or “exclude_tables” option, those options are applied BEFORE the ordering is done. If the ordered list contains no results, then the “order” option is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, The CustomResults are typically assigned to a variable for later usage or passed passed directly into the display() method for viewing in a Jupyter notebook cell.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

set_params

```
set_params(**params)
```

Updates the parameters of this summarizer.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self

summarize

```
summarize(X)
```

Assess the bias of the scored input data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the input data to assess the bias of. Does not support NumPy arrays.

Returns

self

Examples

```
from sasviya.ml.tree import GradientBoostingRegressor
from sasviya.ml import AssessBias
# Train a gradient boosting model
model = GradientBoostingRegressor()
model.fit(simdata.drop(['intervalTarget', 'mySensitiveVariable'], axis=1),
          simdata['intervalTarget'])
# Score the data to get predictions
```

```
scored_data = model.predict(simdata)
# Assess bias in the model predictions
bias_assessor = AssessBias(
    target='intervalTarget',
    target_type='INTERVAL',
    sensitive_var='mySensitiveVariable',
    pred_vars=['P_intervalTarget']
)
bias_assessor.summarize(scored_data)
```

3

Bayesian Network Models

Classes

Class Name	Description
BNET	Bayesian Network Classification Model

BNET

Bayesian Network Classification Model

API: BNET

class sasviya.ml.bnet.BNET(num_bin=5, pre_screening=1, var_select=1, missing_int='ignore', missing_nom='ignore', indep_test='chigsquare', alpha=0.05, mi_alpha=0.05, structure='pc', parenting='bestset', max_parents=5, best_model=False, in_network=None, verbose=0)

Parameters

num_bin : int, optional
Specifies the number of binning levels for all continuous variables.

pre_screening : {0, 1}, optional
Specifies the initial screening for the input variables.

var_select : {0, 1, 2, 3}, optional
Specifies the selection for the input variables.

missing_int : {“ignore”, “impute”}, optional
Specifies how to handle missing values for continuous variables.

missing_nom : {“ignore”, “impute”, “level”}, optional
Specifies how to handle missing values for nominal variables.

indep_test : {“all”, “chigsquare”, “chisquare”, “gsquare”, “mi”},
Specifies the methods for independence tests.

alpha : float, optional
Specifies the significance level for independence tests by using chi-square or G-square statistics.

mi_alpha “ float, optional
Specifies the significance level for independence tests by using mutual information.

structure : {“GENERAL”, “GN”, “NAIVE”, “MB”, “TAN”, “PC”}, optional
Specifies the network structure types.

parenting : {“bestone”, “bestset”}, optional
Specifies the structure learning methods.

max_parents : int, optional
Specifies the maximum number of parents that are allowed for each node in the network.

best_model : bool, optional
Requests that the best model be selected.

verbose: int, optional
Specifies the amount of information to print during model fitting.

Attributes

classes_ : list of str
Stores class labels that are seen during model fitting.

details_ : Results
Stores additional information about the model fitting process and the final results.

feature_names_in : list of str
Stores the column names of the input features that are seen during fitting.

n_features_in_ : int
Stores the number of input features that are seen by the model.

outnetwork_ : DataFrame
Stores the parent-child connections that the fitted network contains.

target_name_in_ : str
Stores the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
decision_function	Computes the decision function given input features.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts class labels given input features.
predict_proba	Estimates probabilities for each class label.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with

the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

decision_function

```
decision_function(X, options=None)
```

Computes the decision function given input features.

Parameters

X : array-like
Specifies training features in the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the decision function. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the estimated posterior probability for the target class. This is (n_samples, n_classes) for the multiclass case and (n_samples,) with just self.classes_[1] returned for the binary case.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,). The target is assumed to be categorical.

nominals: list of str, optional
Specifies the names of columns in *X* to treat as categorical.

Returns

self : object
Returns the instance itself.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts class labels given input features.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted class labels in the shape (n_samples,).

predict_proba

```
predict_proba(X, copy_var=None, options=None)
```

Estimates probabilities for each class label.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns estimated class label probabilities in the shape (n_samples, n_classes).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

4

Decomposition Models

Classes

Class Name	Description
PCA	Principal component analysis.

PCA

Principal component analysis.

API: PCA

class sasviya.ml.decomposition.PCA(n_components=None, *, svd_solver='auto', iterated_power=1, random_state=None, scale=True)

A class that applies a principal component analysis technique for examining relationships among several quantitative variables. The class provides an optimal way to reduce dimensionality by projecting the data onto a lower-dimensional orthogonal subspace that explains as much variation as possible in those variables.

You can use principal component analysis to reduce the number of variables in regression, clustering, and so on.

Parameters

n_components : int or None, optional
Specifies the number of principal components to extract. The value must be less than min(n_samples, n_features). The default is min(n_samples, n_features).

svd_solver : {"auto", "full", "randomized"}, optional
Specifies the solver to be used to calculate the principal components. If the specified value is "auto", then "full" is used unless the input data is larger than 500 x 500 and the *n_components* parameter value is < 80% of the smallest dimension of the data.

iterated_power : int, optional
Specifies the number of solver iterations when the *svd_solver* parameter value is "randomized". Otherwise this parameter is ignored.

random_state : int, optional
Specifies the seed to use for random number generation. This parameter is ignored if the *svd_solver* parameter value is not "randomized".

scale : bool, optional
When set to True, scales variables. Interval inputs are always centered by their corresponding means. When this parameter is set to True, the method divides the inputs by their corresponding standard deviations.

Attributes

details_ : Results
Specifies additional information about the model fitting process and the final results.

components_ : ndarray of shape (n_components, n_features)
Specifies the principal axes in feature space, which represent the directions of maximum variance.

explained_variance_ : ndarray of shape (n_components,)
Specifies the variance explained by each of the selected components.

explained_variance_ratio_ : ndarray of shape (n_components,)
Specifies the percentage of variance explained by each of the selected components.

feature_names_in_ : list(str)
Specifies the number of features to use in the model. This parameter value is equal to len(feature_names_in_).

mean_ : ndarray of shape (n_features,)
Specifies the per-feature empirical mean, which is estimated from the training data.

n_components_ : int
Specifies the number of principal components to extract.

n_features_in_ : int
Specifies the number of features to use in the model. This parameter value is equal to len(feature_names_in_).

n_samples_ : int
Specifies the number of samples in the training data.

singular_values_ : ndarray of shape (n_components,)
Specifies the singular values that correspond to each of the selected components.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model with X.
fit_transform	Fit the model and apply dimensionality reduction to X.
get_covariance	Get the covariance of the data.
get_feature_names_out	Get column names for the transformer's output.
get_params	Gets parameters for this model.
inverse_transform	Transform data back to its original space.
save	Saves the model to a file.
set_params	Updates the parameters of the model.
transform	Reduce the dimensionality of input data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left

corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y=None)
```

Fit the model with X.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the training data, where *n_samples* is the number of samples and *n_features* is the number of features.

y : any
This parameter is ignored (for API consistency only).

Returns

self : object
Returns the instance itself.

fit_transform

```
fit_transform(X, y=None, transform_options=None)
```

Fit the model and apply dimensionality reduction to X.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the training data, where *n_samples* is the number of samples and *n_features* is the number of features.

y : any
This parameter is ignored (for API consistency only).

transform_options: dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns *X* reduced to (n_samples, n_components).

get_covariance

```
get_covariance()
```

Get the covariance of the data.

Returns

ndarray of shape (n_features, n_features).
Returns the covariance matrix.

get_feature_names_out

```
get_feature_names_out()
```

Get column names for the transformer's output.

Returns

ndarray of str
Returns the column names.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

inverse_transform

```
inverse_transform(X, copy_var=None)
```

Transform data back to its original space.

In other words, return an input *X_original* whose transform would be X.

Parameters

X : array-like of shape (n_samples, n_components)
Specifies the transformed data, where *n_samples* is the number of samples and *n_components* is the number of components.

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

Returns

X_original : array-like of shape (n_samples, n_features)
Returns the original data, where *n_samples* is the number of samples and *n_features* is the number of features.

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

transform

```
transform(X, copy_var=None, options=None)
```

Reduce the dimensionality of input data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the data to project onto the first principal components.

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns *X* reduced to (n_samples, n_components).

5

Gaussian Mixture Models

Classes

Class Name	Description
GaussianMixtureModel	The GaussianMixtureModel class fits a probabilistic GMM clustering model to data and predicts cluster assignments for new values.

GaussianMixtureModel

The GaussianMixtureModel class fits a probabilistic GMM clustering model to data and predicts cluster assignments for new values.

API: GaussianMixtureModel

class
sasviya.ml.gaussianmixturemodel.GaussianMixtureModel.GaussianMixtureModel(*, n_components=100, covariance_type='diagonal', max_iter=100, weight_concentration_prior=1, tol=0.01, random_state=None, verbose=0)

Parameters

n_components : int, optional
Specifies the number of mixture components. The default is 100.

covariance_type : {“diagonal|diag”, “full”}, optional
Specifies the covariance matrix type of the Gaussian mixtures. The default is “diagonal”.

max_iter : int, optional
Specifies the maximum number of variational Bayesian (VB) iterations to perform. The default is 100.

weight_concentration_prior : float, optional
Specifies the Dirichlet concentration of each component on the weight distribution. The default is 1.

tol : float, optional
Specifies the convergence threshold of the VB algorithm. The default is 0.01.

random_state : float, optional
Specifies the seed to use for random number generation for initialization.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

details_ : Results
Specifies additional information about the model fitting process and the final results.

cluster_summary_ : ndarray of shape (n_components, n_features+3)
Specifies the cluster size, neighbor, and center of each mixture component.

cluster_info_ : ndarray of shape (n_components+2, n_features*2 + 2)
Specifies the cluster weight, mean, and variance of each mixture component.

cluster_covariance_ : ndarray of shape (n_components*n_features, n_features + 1)
Specifies the covariance of each mixture component.

labels_ : ndarray of shape (n_samples,)
Specifies the labels of each observation.

converged_ : bool
When set to True, convergence is reached in fit().

n_iter_ : int
Specifies the number of model fitting iterations to perform.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during fitting.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
fit_predict	Fit the model to the provided training data and predict cluster assignments.
get_params	Gets parameters for this model.
predict	Predict the closest cluster for each given input feature.
predict_proba	Predict the cluster probabilities for each given input feature.
save	Saves the model to a file.
score	Score clustering results on data X, based on the learned model.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y=None)
```

Fit the model to the training data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the training data, where *n_samples* is the number of samples and *n_features* is the number of features.

y : any
This parameter is ignored (included for API consistency only).

Returns

self : object
Returns the instance itself.

fit_predict

```
fit_predict(X, y=None, predict_options=None)
```

Fit the model to the provided training data and predict cluster assignments.

Equivalent to calling fit(X) followed by predict(X).

Parameters

X : array-like
Specifies the samples to fit and predict the closest cluster for, with the shape (n_samples, n_features).

y : any
This parameter is ignored (included for API consistency only).

predict_options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

type(X)
Returns the predicted index of the cluster for each input in the shape (n_samples,).

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, options=None)
```

Predict the closest cluster for each given input feature.

Parameters

X : array-like
Specifies the samples to predict, with the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

type(X)
Returns the predicted index of the cluster for each input in the shape (n_samples,).

predict_proba

```
predict_proba(X, options=None)
```

Predict the cluster probabilities for each given input feature.

Parameters

X : array-like
Specifies the samples to predict, with the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

type(X)
Returns the predicted probabilities of the cluster for each input in the shape (n_samples, n_components).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y=None, options=None)
```

Score clustering results on data X, based on the learned model.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies new data, where *n_samples* is the number of samples and *n_features* is the number of features.

y : any
This parameter is ignored (included for API consistency only).

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

type(X)
Returns the index of the cluster that each observation belongs to, along with the probability that it belongs to the cluster.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

6

K-Means Clustering

Classes

Class Name	Description
KMeans	K-Means clustering.

KMeans

K-Means clustering.

API: KMeans

class sasviya.ml.kmeans.KMeans(*, n_clusters=6, init='random_centroids', max_iter=10, random_state=None, distance='Euclidean', distance_nom='Binary', impute=None, impute_nom=None, verbose=0, standardize=None, estimate_n_clusters=None, stop_criterion=None, gamma=None)

Parameters

n_clusters : int, optional
Specifies the number of clusters to form and centroids to generate.

init : {“forgy”, “rand|random”}, optional
Specifies the method for obtaining the initial estimate of cluster centers. The “forgy” method selects n_cluster data points at random to use as the cluster centroids. The “rand” method assigns observations to a cluster at random. By default, init = “forgy”.

max_iter : int, optional
Specifies the maximum number of iterations for the algorithm to perform. By default, the value is 10.

random_state : int, optional
Specifies the seed to use for random number generation for initialization.

distance : {“euclidean”, “manhattan”}, optional
Specifies the distance measure that is used for the similarity of the interval inputs. By default, the value is “euclidean”.

distance_nom : {“binary”, “globalfreq”, “relativefreq”}, optional
Specifies the distance measure that is used for the similarity of the nominal inputs. By default, the value is “binary”.

impute : “mean”, optional
Specifies the imputation method for interval inputs. By default, the value is None.

impute_nom : “mode”, optional
Specifies the imputation method for nominal inputs. By default, the value is None.

verbose : int, optional
Specifies the amount of information to send to the log. By default, the value is 0.

standardize : {“range”, “std”}, optional
Specifies the method to use for standardizing interval input variables. By default, the value is None.

estimate_n_clusters : dict, optional
Specifies whether to use the aligned box criterion (ABC) method and the values for that method to estimate the number of clusters. You can specify the following parameters:

- **“noc”: “abc”, optional**
  Specifies whether to use the ABC method to infer the number of clusters. By default, the value is None, and the algorithm uses the value that you specify in the n_clusters parameter.
- **“align”: “pca”, optional**
  Specifies the method to use for aligning the reference data on the basis of the input data. This parameter applies only when the noc value is “abc”. By default, the value of align is None.

- **“B”: int or numpy.int32, optional**
  Specifies the amount of reference data to create for each cluster candidate. This parameter applies only when the noc value is “abc”. By default, the value of B is 1.
- **“criterion”: {“all”, “firstmaxwithstd”, “firstpeak”, “globalpeak”}, optional**
  Specifies the criterion to be used to estimate the number of clusters that use the statistics that are obtained by the ABC method. This parameter applies only when the noc value is “abc”. By default, the value of criterion is “globalpeak”.
- **“min_clusters”: int or numpy.int32, optional**
  Specifies the minimum number of clusters to search in order to find the best number of clusters. This parameter applies only when the noc value is “abc”. By default, the value of min_clusters is 2.

stop_criterion : dict, optional
Specifies the stop criterion for the clustering.

- **“change”: {“cluster_change”, “wcss_change”}, optional**
  Specifies the method to use for convergence. If you omit this parameter, the algorithm stops after it reaches the maximum number of iterations (which is specified in the max_iter parameter).
- **“value”: float, optional**
  Specifies the threshold value for the change.

gamma : float, optional
Specifies the gamma value for the kprototypes algorithm.

Attributes

details_ : Results
Stores additional information about the model fitting process and the final results.

cluster_centers_ : ndarray of shape (n_clusters, n_features)
Stores the coordinates of cluster centers.

labels_ : ndarray of shape (n_samples,)
Stores the labels of each observation.

inertia_ : float
Stores the sum of squared distances of observations to their closest cluster center.

n_iter_ : int
Stores the number of model fit iterations that are performed.

n_features_in_ : int
Stores the number of input features that are seen by the model.

feature_names_in_ : list(str)
Stores the names of the input feature columns that are seen during model fitting.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model on the training data.
fit_predict	Fits the model on the provided training data and predicts cluster assignments
get_params	Gets parameters for this model.
predict	Predicts the closest cluster for each given input feature.
save	Saves the model to a file.
score	Estimates model fitness by calculating the negative of the K-Means objective.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, nominals=None, y=None)
```

Fits the model on the training data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the training data, where *n_samples* is the number of samples and *n_features* is the number of features.

nominals: iterable of str, optional
Specifies the nominal features in the input data.

y : any
Ignored (included for API consistency only).

Returns

self : object
Returns the instance itself.

fit_predict

```
fit_predict(X, nominals=None, y=None, predict_options=None)
```

Fits the model on the provided training data and predicts cluster assignments

Equivalent to calling fit(X) followed by predict(X).

Parameters

X : array-like
Specifies the samples to fit and predict the closest cluster for, with shape (n_samples, n_features).

y : any
Ignored (included for API consistency only).

predict_options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Predicted index of the cluster for each input in the shape (n_samples,).

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts the closest cluster for each given input feature.

Parameters

X : array-like
Specifies the samples to predict, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns a predicted index of the cluster for each input in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X=None)
```

Estimates model fitness by calculating the negative of the K-Means objective.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

7

Linear Models

Classes

Class Name	Description
ElasticNet	Linear regression with a mix of L1 and L2 regularization.
Lasso	Linear regression with L1 regularization.
LinearRegression	Ordinary least squares linear regression.
LogisticRegression	Logistic Regression classifier.
Ridge	Linear Regression with L2 regularization.

ElasticNet

Linear regression with a mix of L1 and L2 regularization.

API: ElasticNet

class sasviya.ml.linear_model.ElasticNet(alpha=None, *, l1_ratio=0.5, fit_intercept=True, tol=1e-08, solver='lbfgs', verbose=0)

Parameters

alpha : float, optional
Specifies the strength of the regularization that is applied. A constant value is used to scale the regularization term of the cost function. If None is specified, the regularization term is computed from the data. "Alpha" is also referred to as "lambda" in fit results.

l1_ratio : float, optional
Specifies the elastic net mixing parameter. Specifies the ratio of L1:L2 regularization. If the specified value is 0, only L2 regularization is used. If the specified value is 1, only L1 regularization is used.

fit_intercept : bool, optional
Specifies whether or not to include a bias term in the model.

solver : {"admm", "bfgs", "lbfgs", "nlp"}, optional
Specifies the optimization algorithm to use.

tol : float, optional
Specifies the tolerance threshold for early stopping. Model fitting stops when the size of the gradient is less than the value of *tol*.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

coef_ : ndarray of shape (n_parameters,)
Specifies the model coefficients. The number of parameters can be greater than the number of features if additional parameters are introduced to handle missing values, or less than the number of features if variable selection is used.

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

intercept_ : ndarray of shape (, n_intercepts)
Specifies the bias term(s) to be added to the model.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

Lasso

Linear regression with L1 regularization.

API: Lasso

class sasviya.ml.linear_model.Lasso(*, fit_intercept=True, verbose=0)

Parameters

fit_intercept : bool, optional
Specifies whether or not to include a bias term in the model.

verbose : int
Specifies the amount of information to print about the fitted model.

Attributes

coef_ : ndarray of shape (n_parameters,)
Specifies the model coefficients. The number of parameters can be greater than the number of features if additional parameters are introduced to handle missing values, or less than the number of features if variable selection is used.

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

intercept_ : ndarray of shape (, n_intercepts)
Specifies the bias term(s) to be added to the model.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list

Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method.

You need to specify plots="none" in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {"light", "dark", "opal", "midnight", "raven", "htmlencore", "highcontrast", "grayscale", monochrome"}
Overrides the default theme that the graph uses. The default is "light". You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {"svg", "png"}
Overrides the default output type. The default type is "svg", which contains image map information for tooltips. The "png" output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword ("none", "all"), a single table, or a list of tables. The default is "all", which keeps all tables. The keyword "none" removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword ("none", "graph_tables", "all"), a single table, or a list of tables. The default is "none", which displays all tables. The keyword "graph_tables" removes all tables that were used to produce a graph. The keyword "all" removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

LinearRegression

Ordinary least squares linear regression.

API: LinearRegression

class sasviya.ml.linear_model.LinearRegression(*, fit_intercept=True, verbose=0)

Parameters

fit_intercept : bool, optional
Specifies whether or not to include a bias term in the model.

verbose : int
Specifies the amount of information to print about the fitted model.

Attributes

coef_ : pandas.DataFrame of shape (n_parameters,)
Specifies the model coefficients. The number of parameters can be greater than the number of features if additional parameters are introduced to handle missing values, or less than the number of features if variable selection is used.

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

intercept_ : pandas.DataFrame of shape (, n_intercepts)
Specifies the bias term(s) to be added to the model.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Method Name	Description
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

LogisticRegression

Logistic Regression classifier.

API: LogisticRegression

```
class sasviya.ml.linear_model.LogisticRegression(*, tol=1e-08,
fit_intercept=True, solver='nrridg', selection=None, max_iter=50,
max_time=None, verbose=0)
```

Parameters

tol : float, optional
Specifies the relative gradient convergence criterion.

fit_intercept : bool, optional
Specifies whether to include a bias term in the model.

max_iter : float, optional
Specifies the maximum number of training iterations.

max_time : float, optional
Specifies the maximum training time, in seconds.

solver : {“congra”, “dbldog”, “lbfgs”, “newrap”, “nmsimp”, “nrridg”, “quanew”, “duquanew”}, optional
Specifies the optimization algorithm to use. Valid options are:

- “congra”: conjugate gradient method
- “dbldog”: double-dogleg method
- “lbfgs”: limited-memory BFGS solver
- “newrap”: Newton-Raphson method with line search and ridging
- “nmsimp”: Nelder-Mead simplex method
- “nrridg”: Newton-Raphson method with ridging
- “quanew”: dual quasi-Newton optimization
- “duquanew”: dual quasi-Newton optimization

selection : {“backward”, “forward”, “lasso”, “stepwise”}, optional
Specifies the variable selection method to use.

verbose : int, optional
Specifies the information to print about the fitted model:

0: None 1: Iteration history 2: Iteration history and fit statistics

Attributes

classes_ : ndarray of shape (n_classes,)
Class labels seen during model fitting.

coef_ : pandas.DataFrame of shape (n_parameters,)
Model coefficients. The number of parameters may be greater than the number of features if additional parameters are introduced to handle missing values, or less than the number of features if variable selection was used.

details_ : Results
Additional information about the model fitting process and the final results.

intercept_ : pandas.DataFrame of shape (n_classes-1,)
Bias term(s) added to the model.

n_iter_ : int
The number of model fit iterations performed.

n_features_in_ : int
The number of input features seen by the model.

feature_names_in_ : list(str)
The column names of the input features seen during fitting.

target_name_in_ : str
Column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
decision_function	Computes the decision function given input features.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model on the provided training data.
get_params	Gets parameters for this model.
predict	Predicts class labels given input features.
predict_proba	Estimates probabilities for each class label.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left

corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

decision_function

```
decision_function(X, options=None)
```

Computes the decision function given input features.

Parameters

X : array-like
Specifies training features in the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the decision function. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the estimated posterior probability for the target class. This is (n_samples, n_classes) for the multiclass case and (n_samples,) with just self.classes_[1] returned for the binary case.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model on the provided training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Names of columns in *X* that should be treated as categorical. The target is always treated as categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts class labels given input features.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted class labels in the shape (n_samples,).

predict_proba

```
predict_proba(X, copy_var=None, options=None)
```

Estimates probabilities for each class label.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns estimated class label probabilities in the shape (n_samples, n_classes).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

Ridge

Linear Regression with L2 regularization.

API: Ridge

```
class sasviya.ml.linear_model.Ridge(alpha=None, *, fit_intercept=True,
tol=1e-08, solver='lbfgs', verbose=0)
```

Parameters

alpha : float, optional
Specifies the strength of the regularization that is applied. A constant value is used to scale the regularization term of the cost function. If None is specified, the regularization term is computed from the data. “Alpha” is also referred to as “lambda” in fit results.

fit_intercept : bool, optional
Specifies whether or not to include a bias term in the model.

solver : {“admm”, “bfgs”, “lbfgs”, “nlp”}, optional
Specifies the optimization algorithm to use.

tol : float, optional
Specifies the tolerance threshold for early stopping. Model fitting stops when the size of the gradient is less than the value of *tol*.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

coef_ : ndarray of shape (n_parameters,)
Specifies the model coefficients. The number of parameters can be greater than the number of features if additional parameters are introduced to handle missing values, or less than the number of features if variable selection is used.

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

intercept_ : ndarray of shape (, n_intercepts)
Specifies the bias term(s) to be added to the model.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
get_params	Gets parameters for this model.

Method Name	Description
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

8

Nominal Dimension Reduction

Classes

Class Name	Description
LPCA	Nominal Variables Dimension Reduction with Logistic Principal Component Analysis
MCA	Nominal Variables Dimension Reduction with Multiple Correspondence Analysis

LPCA

Nominal Variables Dimension Reduction with Logistic Principal Component Analysis

API: LPCA

class sasviya.ml.nominaldr.LPCA(dimensions, prefix='rv', epsilon=0.0001, max_iter=100, m=4, verbose=0)

This object performs nominal variables dimension reduction via the logistic principal component analysis (LPCA) method.

Parameters

dimensions : int
Specifies the number of reduced variables.

prefix : str, optional
Specifies a prefix to apply to the names of the reduced variables. The default is 'rv'.

epsilon : float, optional
Specifies the tolerance to use in determining the convergence of the iterative algorithm. The default is 0.0001.

max_iter : int, optional
Specifies the maximum number of iterations for the iterative algorithm. The default is 100.

m : int, optional
Specifies a finite positive value to approximate the logit function's infinite limits. logit(1) is approximated by m, and logit(0) is approximated by -m. The default is 4.

verbose : int, optional
Specifies the amount of information to print about the fitted model. The default is 0. A value of 0 prints nothing. A value of 1 prints the model information table to the log. A value of 2 prints the model information and number of observations tables to the log. A value of 3 prints the model information, number of observations, and variable information tables to the log. A value of 4 prints the model information, number of observations, variable information, and iteration history tables to the log. A value of 5 prints the model information, number of observations, variable information, iteration history, and explained variance summary tables to the log.

Attributes

details_ : Results
Stores additional information about the model fitting process and the final results.

dimensions_ : int
Stores the number of reduced variables (which can be less than the specified parameter dimensions).

n_features_ : int
Stores the number of features that appear in the training data.

n_features_in_ : int
Stores the number of input nominal features that the model uses.

n_samples_ : int
Stores the number of samples that appear in the training data.

n_samples_in_ : int
Stores the number of samples that the model uses.

n_all_categories_in_ : int
Stores the total number of categories for all the input nominal features.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
fit_transform	Fits a transformation model to data set X and then runs the transformation model on that data set.
get_params	Gets parameters for this model.
save	Saves the model to a file.
set_params	Updates the parameters of the model.
transform	Applies the model to new input data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y=None, nominals=None, freq=None, id=None)
```

Fits the model to the training data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the training data, where *n_samples* is the number of samples and *n_features* is the number of features.

y : any
This parameter is ignored (included for API consistency only).

nominals: str | iterable of str | None, optional
Specifies the columns in the input to the X parameter to treat as nominal data.

freq : str | None, optional
Specifies the column in the input to the X parameter that contains the frequency of each observation.

id: str | iterable of str | None, optional
Specifies the columns in the input to the X parameter that contain the ID of each observation.

Returns

self : object
Returns the instance itself.

fit_transform

```
fit_transform(X, y=None, transform_options=None, **kwargs)
```

Fits a transformation model to data set X and then runs the transformation model on that data set.

Parameters

X : array-like
Specifies the input data set that is used to train the model.

y : array-like, optional
Specifies the target data set that is used to train the model. This parameter is ignored if the transformation model does not use a target data set.

transform_options: dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the X data set after it has been transformed.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

transform

```
transform(X, options=None)
```

Applies the model to new input data.

Parameters

X : array-like
Specifies new input data.

options: dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns a table that contains the reduced data for the input to the X parameter.

MCA

Nominal Variables Dimension Reduction with Multiple Correspondence Analysis

API: MCA

class sasviya.ml.nominaldr.MCA(dimensions, prefix='rv', verbose=0)

This object performs nominal variables dimension reduction via the multiple correspondence analysis (MCA) method.

Parameters

dimensions : int
Specifies the number of reduced variables.

prefix : str, optional
Specifies a prefix to apply to the names of the reduced variables. The default is 'rv'.

verbose : int , optional
Specifies the amount of information to print about the fitted model. The default is 0. A value of 0 prints nothing. A value of 1 prints the model information table to the log. A value of 2 prints the model information and number of observations tables to the log. A value of 3 prints the model information, number of observations, and variable information tables to the log. A value of 4

prints the model information, number of observations, variable information, and explained variance summary tables to the log.

Attributes

details_ : Results
Stores additional information about the model fitting process and the final results.

dimensions_ : int
Stores the number of reduced variables (which can be less than the specified parameter dimensions).

n_features_ : int
Stores the number of features that appear in the training data.

n_features_in_ : int
Stores the number of input nominal features that the model uses.

n_samples_ : int
Stores the number of samples that appear in the training data.

n_samples_in_ : int
Stores the number of samples that the model uses.

n_all_categories_in_ : int
Stores the total number of categories for all the input nominal features.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
fit_transform	Fits a transformation model to data set X and then runs the transformation model on that data set.
get_params	Gets parameters for this model.
save	Saves the model to a file.

Method Name	Description
set_params	Updates the parameters of the model.
transform	Applies the model to new input data.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y=None, nominals=None, freq=None, id=None)
```

Fits the model to the training data.

Parameters

X : array-like of shape (n_samples, n_features)
Specifies the training data, where *n_samples* is the number of samples and *n_features* is the number of features.

y : any
This parameter is ignored (included for API consistency only).

nominals: str | iterable of str | None, optional
Specifies the columns in the input to the X parameter to treat as nominal data.

freq : str | None, optional
Specifies the column in the input to the X parameter that contains the frequency of each observation.

id: str | iterable of str | None, optional
Specifies the columns in the input to the X parameter that contain the ID of each observation.

Returns

self : object
Returns the instance itself.

fit_transform

```
fit_transform(X, y=None, transform_options=None, **kwargs)
```

Fits a transformation model to data set X and then runs the transformation model on that data set.

Parameters

X : array-like
Specifies the input data set that is used to train the model.

y : array-like, optional
Specifies the target data set that is used to train the model. This parameter is ignored if the transformation model does not use a target data set.

transform_options: dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the X data set after it has been transformed.

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

transform

```
transform(X, options=None)
```

Applies the model to new input data.

Parameters

X : array-like
Specifies new input data.

options: dict, optional
Specifies additional runtime options to pass to the transformation. The keys should be option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns a table that contains the reduced data for the input to the X parameter.

9

Support Vector Models

Classes

Class Name	Description
SVC	Support vector classification model.
SVR	Support vector regression model.

SVC

Support vector classification model.

API: SVC

class sasviya.ml.svm.SVC(*, max_iter=25, C=1.0, kernel='linear', degree=2, sigma='auto', coef0=-1, method=None, tol=None, random_state=1, scale=True, verbose=0)

Parameters

max_iter : int, optional
Specifies the maximum number of training iterations.

C : float, optional
Specifies the cost regularization parameter.

kernel : {“linear”, “poly”, “polynomial”, “rbf”, “sigmoid”}, optional
Specifies the type of kernel to use.

degree : int, optional
Specifies the degree of the polynomial kernel. This parameter is ignored if the *kernel* parameter value is not “polynomial”.

sigma : “auto”, “scale” or float, optional
Specifies the kernel parameter for the RBF and Sigmoid kernels.

coef0 : float, optional
Specifies the second parameter for the Sigmoid kernel.

method : {“activeset”, “cd”, “ipoint”} or None, optional
Specifies the optimization method to be used to fit the model.

- The “activeset” method can be used with all kernels and is the default for the RBF and Sigmoid kernels, as well as for polynomial kernels of more than degree 3.
- The “ipoint” (interior point) method can be used with the linear and polynomial kernels and is the default for linear kernels and polynomial kernels of degree 3 or less.
- Coordinate descent can be used only with the linear kernel.

tol : float, optional
Specifies the tolerance threshold for early stopping.

random_state : int, optional
Specifies the seed to use for random number generation.

scale : bool, optional
When set to True, scales the numeric variables prior to model fitting.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

classes_ : ndarray of shape (2,)
Specifies the class labels that are seen during model fitting.

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

n_iter_ : int
Specifies the number of training iterations. This attribute is present only when the ipoint method is used.

n_support_ : int
Specifies the number of support vectors. This attribute is not present if the *method* parameter value is “cd”.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
decision_function	Computes the decision function given input features.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts class labels given input features.
predict_proba	Estimates probabilities for each class label.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left

corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

decision_function

```
decision_function(X, options=None)
```

Computes the decision function given input features.

Parameters

X : array-like
Specifies training features in the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the decision function. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the estimated posterior probability for the target class. This is (n_samples, n_classes) for the multiclass case and (n_samples,) with just self.classes_[1] returned for the binary case.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as nominal values. For classifier models, the target is assumed to be categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts class labels given input features.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted class labels in the shape (n_samples,).

predict_proba

```
predict_proba(X, copy_var=None, options=None)
```

Estimates probabilities for each class label.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns estimated class label probabilities in the shape (n_samples, n_classes).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

SVR

Support vector regression model.

API: SVR

```
class sasviya.ml.svm.SVR(*, max_iter=25, C=1.0, kernel='linear', degree=2,
tol=None, epsilon=0.01, random_state=1, scale=True, verbose=0)
```

Parameters

max_iter : int, optional
Specifies the maximum number of training iterations.

C : float, optional
Specifies the cost regularization parameter.

kernel : {“linear”, “poly”, “polynomial”}, optional
Specifies the type of kernel to use.

degree : int, optional
Specifies the degree of the polynomial kernel. This parameter is used only when the *kernel* parameter value is “polynomial”.

tol : float, optional
Specifies the tolerance threshold for early stopping.

epsilon : float, optional
Specifies the epsilon-tube distance. Points that have a predicted value within this distance from their actual value are not counted toward the training loss.

random_state : int, optional
Specifies the seed to use for random number generation.

scale : bool, optional
When set to True, scales the numeric variables prior to model fitting.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fits the model to the training data.

Method Name	Description
get_params	Gets parameters for this model.
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fits the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as nominal values. For classifier models, the target is assumed to be categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

10

Tree Models

Classes

Class Name	Description
DecisionTreeClassifier	A decision tree classification model.
DecisionTreeRegressor	A decision tree regression model.
ForestClassifier	A random forest classification model.
ForestRegressor	A random forest regression model.
GradientBoostingClassifier	Gradient boosting classifier.
GradientBoostingRegressor	Gradient boosting regression model.

DecisionTreeClassifier

A decision tree classification model.

API: DecisionTreeClassifier

class sasviya.ml.tree.DecisionTreeClassifier(criterion='IGR', max_depth=10, min_samples_leaf=5, n_bins=50, ccp_alpha=0, verbose=0)

Parameters

criterion : {"chaid", "chisquare", "entropy|gain", "igr|gainratio", "gini"}
Specifies the split criterion for each tree node.

max_depth : int, optional
Specifies the maximum depth of the tree.

min_samples_leaf : int, optional
Specifies the minimum number of observations on each node.

n_bins : int, optional
Specifies the number of bins to use for numeric variables.

ccp_alpha : float, optional
Specifies the value to use for minimal cost-complexity pruning for regression trees.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

details_ : Results
Specifies additional information about the model fitting process and the final results.

n_features_in_ : int
Specifies the number of input features that are seen by the model. This can be a subset of the input columns if variable selection is used.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during fitting.

n_features_used_ : int
Specifies the number of input features to be used by the model.

target_name_in_ : str
Specifies the column name of the target variable.

n_nodes_ : int
Specifies the number of tree nodes.

max_n_branches_ : int
Specifies the maximum number of branches of the tree.

depth_ : int
Specifies the depth of the tree.

n_leaves_ : int
Specifies the number of leaves of the tree.

n_bins_ : int
Specifies the number of bins for numeric variables.

min_n_leaves_ : int
Specifies the minimum size of leaves.

max_n_leaves_ : int
Specifies the maximum size of leaves.

n_obs_ : int
Specifies the number of observations to be used by the model.

classes_ : list(str)
Specifies the class labels.

n_classes_ : int
Specifies the number of classes.

feature_importances_ : pandas.DataFrame()
Specifies the ranking of features by importance.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
decision_function	Computes the decision function given input features.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts class labels given input features.
predict_proba	Estimates probabilities for each class label.
save	Saves the model to a file.

Method Name	Description
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

decision_function

```
decision_function(X, options=None)
```

Computes the decision function given input features.

Parameters

X : array-like
Specifies training features in the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the decision function. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the estimated posterior probability for the target class. This is (n_samples, n_classes) for the multiclass case and (n_samples,) with just self.classes_[1] returned for the binary case.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None, weight=None)
```

Fit the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical. For classifier models, the target is assumed to be categorical.

weight : str, optional
Specifies the name of a column in *X* that contains the weight of each observation.

Returns

——-

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts class labels given input features.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted class labels in the shape (n_samples,).

predict_proba

```
predict_proba(X, copy_var=None, options=None)
```

Estimates probabilities for each class label.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns estimated class label probabilities in the shape (n_samples, n_classes).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

DecisionTreeRegressor

A decision tree regression model.

API: DecisionTreeRegressor

class sasviya.ml.tree.DecisionTreeRegressor(criterion='RSS', max_depth=10, min_samples_leaf=5, n_bins=50, ccp_alpha=0, verbose=0)

Parameters

criterion : {"chaid", "ftest", "variance|rss"}
Specifies the criterion to be used to measure the quality of a split in each tree node.

max_depth : int, optional
Specifies the maximum depth of any tree.

min_samples_leaf : int, optional
Specifies the minimum number of observations on each node.

n_bins : int, optional
Specifies the number of bins to use for numeric variables.

ccp_alpha : float, optional
Specifies the alpha value to use for minimal cost-complexity pruning for regression trees.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

details_ : Results
Specifies additional information about the model fitting process and the final results.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during fitting.

n_features_used_ : int
Specifies the number of input features to be used by the model.

target_name_in_ : str
Specifies the column name of the target variable.

n_nodes_ : int
Specifies the number of tree nodes.

max_n_branches_ : int
Specifies the maximum number of branches of the tree.

depth_ : int
Specifies the depth of the tree.

n_leaves_ : int
Specifies the number of leaves of the tree.

n_bins_ : int
Specifies the number of bins for numeric variables.

min_n_leaves_ : int
Specifies the minimum size of leaves.

max_n_leaves_ : int
Specifies the maximum size of leaves.

n_obs_ : int
Specifies the number of observations to be used by the model.

feature_importances_ : pandas.DataFrame()
Specifies the ranking of features by importance.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left

corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None, weight=None)
```

Fit the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical. For classifier models, the target is assumed to be categorical.

weight : str, optional
Specifies the name of a column in *X* that contains the weight of each observation.

Returns

—-

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

ForestClassifier

A random forest classification model.

API: ForestClassifier

```
class sasviya.ml.tree.ForestClassifier(n_estimators=100, *, bootstrap=0.6,
criterion='IGR', max_depth=20, max_features=None, min_samples_leaf=5,
n_bins=50, oob_score=True, random_state=None, verbose=0)
```

Parameters

n_estimators : int, optional
Specifies the number of trees to create.

bootstrap : float, optional
Specifies the fraction of the data to use for the bootstrap sample.

criterion : {“chaid”, “chisquare”, “entropy|gain”, “igr|gainratio”, “gini”}
Specifies the criterion to be used to measure the quality of a split in each tree node.

max_depth : int, optional
Specifies the maximum depth of any tree.

max_features : int, optional
Specifies the number of input variables to consider in splitting a node (also referred to as “M” in model output). The variables are selected at random from the input variables for each tree. By default, the value of this parameter is equal to the square root of the number of input variables.

min_samples_leaf : int, optional
Specifies the minimum number of observations at each node.

n_bins : int, optional
Specifies the number of bins to use for numeric variables.

oob_score : bool, optional
Specifies whether or not to compute the out-of-bag error during model training.

random_state : int, optional
Specifies the seed to use for random number generation.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

classes_ : list(str) with length n_classes
Specifies the class labels that are seen during model fitting.

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_importances_ : DataFrame
Specifies the feature importances to calculate during model fitting, scaled to the range [0, 1].

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

n_classes_ : int
Specifies the number of target classes.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

oob_score_ : float
Specifies the out-of-bag error that is observed during model fitting. This value is computed only if the *oob_score* parameter is set to True.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
decision_function	Computes the decision function given input features.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts class labels given input features.
predict_proba	Estimates probabilities for each class label.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

decision_function

```
decision_function(X, options=None)
```

Computes the decision function given input features.

Parameters

X : array-like
Specifies training features in the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the decision function. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the estimated posterior probability for the target class. This is (n_samples, n_classes) for the multiclass case and (n_samples,) with just self.classes_[1] returned for the binary case.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fit the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical. For classifier models, the target is assumed to be categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts class labels given input features.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted class labels in the shape (n_samples,).

predict_proba

```
predict_proba(X, copy_var=None, options=None)
```

Estimates probabilities for each class label.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns estimated class label probabilities in the shape (n_samples, n_classes).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

ForestRegressor

A random forest regression model.

API: ForestRegressor

class sasviya.ml.tree.ForestRegressor(n_estimators=100, *, bootstrap=0.6, criterion='RSS', max_depth=20, max_features=None, min_samples_leaf=5, n_bins=50, oob_score=True, random_state=None, verbose=0)

Parameters

n_estimators : int, optional
Specifies the number of trees to create.

bootstrap : float, optional
Specifies the fraction of the data to use for the bootstrap sample.

criterion : {"chaid", "ftest", "variance|rss"}
Specifies the criterion to be used to measure the quality of a split in each tree node.

max_depth : int, optional
Specifies the maximum depth of any tree.

max_features : int, optional
Specifies the number of input variables to consider in splitting a node (also referred to as "M" in model output). The variables are selected at random from the input variables for each tree. By default, the value of this parameter is equal to the square root of the number of input variables.

min_samples_leaf : int, optional
Specifies the minimum number of observations on each node.

n_bins : int, optional
Specifies the number of bins to use for numeric variables.

oob_score : bool, optional
Specifies whether or not to compute the out-of-bag error during model training.

random_state : int, optional
Specifies the seed to use for random number generation.

verbose : int, optional
Specifies the amount of information to print about the fitted model.

Attributes

details_ : Results
Specifies additional information about the model fitting process and the final results.

feature_importances_ : DataFrame
Specifies the feature importances to calculate during model fitting, scaled to the range [0, 1].

feature_names_in_ : list(str)
Specifies the column names of the input features that are seen during model fitting.

n_features_in_ : int
Specifies the number of input features that are seen by the model.

oob_score_ : float
Specifies the out-of-bag error that is observed during model fitting. This value is computed only if the *oob_score* parameter is set to True.

target_name_in_ : str
Specifies the column name of the target variable.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
get_params	Gets parameters for this model.

Method Name	Description
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fit the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical. For classifier models, the target is assumed to be categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

GradientBoostingClassifier

Gradient boosting classifier.

API: GradientBoostingClassifier

```
class sasviya.ml.tree.GradientBoostingClassifier(*, min_samples_leaf=5,
n_bins=50, n_estimators=100, max_depth=4, max_features=None,
learning_rate=0.1, subsample=0.5, l1_penalty=0, l2_penalty=1,
random_state=None, verbose=0, n_iter_no_change=0, tol=0,
calc_feature_importances=False, validation_fraction=0)
```

Parameters

n_bins: int, optional
Specifies the number of bins to use for numeric variables. The default is 50.

min_samples_leaf : int, optional
Specifies the minimum number of observations on each node. The default is 5.

n_estimators : int, optional
Specifies the number of trees to create. The default is 100.

max_depth : int, optional
Specifies the maximum depth of the tree. The default is 4.

max_features : float, optional
Specifies the number of input variables to consider for splitting on a node. The default is the number of input variables.

learning_rate : float, optional
Specifies the learning rate of each tree. The default is 0.1.

subsample : float, optional
Specifies the fraction of the data to use for building each tree. The default is 0.5.

l1_penalty : float, optional
Specifies the L1 norm regularization on prediction. The default is 0.

l2_penalty : float, optional
Specifies the L2 norm regularization on prediction. The default is 1.

random_state : float, optional
Specifies the seed for the random number generator. By default, the random number stream is based on the computer clock. If you want a reproducible random number sequence between runs, specify a value greater than 0.

verbose : int, optional
Specifies the amount of information to display about the fitted model.

calc_feature_importances : bool, optional
Specifies whether or not to calculate feature importances.

validation_fraction: float, optional
Specifies the fraction of training samples to use for validation. A value greater than 0 enables early stopping mode.

n_iter_no_change : int, optional
Specifies the maximum number of training iterations for early stopping. This parameter is ignored unless a validation fraction is specified.

tol : float, optional
Specifies the tolerance value for early stopping as the change relative to the previous iteration. This parameter is ignored unless a validation fraction is specified.

Attributes

details_ : Results
Specifies information about the model fitting process and the final results.

n_estimators_ : int
Specifies the number of trees in the fitted model. The number could be less than the specified value because of early stopping.

avg_n_leaves_ : float
Specifies the average number of leaves among trees in the fitted model.

min_n_leaves_ : int
Specifies the minimum number of leaves among trees in the fitted model.

max_n_leaves_ : int
Specifies the maximum number of leaves among trees in the fitted model.

max_n_nodes_ : int
Specifies the maximum number of nodes among trees in the fitted model.

min_n_nodes_ : int
Specifies the minimum number of nodes among trees in the fitted model.

max_depth_ : int
Specifies the maximum depth among trees in the fitted model.

min_depth_ : int
Specifies the minimum depth among trees in the fitted model.

min_samples_leaf_ : int
Specifies the minimum leaf size among trees in the fitted model.

max_samples_leaf_ : int
Specifies the maximum leaf size among trees in the fitted model.

n_features_in_ : int
Specifies the number of input features.

feature_names_in_ : list(str)
Specifies the column names of the input features.

n_features_used_ : int
Specifies the number of features to be used by the fitted model.

target_name_in_ : str
Specifies the column name of the target variable.

classes_ : list(str)
Specifies the class labels.

n_classes_ : int
Specifies the number of classes of the target variable.

feature_importances_ : pandas.DataFrame()
Specifies the ranking of features by importance.

distribution_ : str
Specifies the distribution function of the fitted model.

validation_scores_ : pandas.DataFrame()
Specifies the misclassification rate for the validation sample by iteration.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
decision_function	Computes the decision function given input features.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts class labels given input features.
predict_proba	Estimates probabilities for each class label.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

decision_function

```
decision_function(X, options=None)
```

Computes the decision function given input features.

Parameters

X : array-like
Specifies training features in the shape (n_samples, n_features).

options: dict, optional
Specifies additional runtime options to pass to the decision function. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns the estimated posterior probability for the target class. This is (n_samples, n_classes) for the multiclass case and (n_samples,) with just self.classes_[1] returned for the binary case.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fit the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical. For classifier models, the target is assumed to be categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts class labels given input features.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted class labels in the shape (n_samples,).

predict_proba

```
predict_proba(X, copy_var=None, options=None)
```

Estimates probabilities for each class label.

Parameters

X : array-like
Specifies the samples to predict class labels for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns estimated class label probabilities in the shape (n_samples, n_classes).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.

GradientBoostingRegressor

Gradient boosting regression model.

API: GradientBoostingRegressor

class sasviya.ml.tree.GradientBoostingRegressor(*, min_samples_leaf=5, n_bins=50, n_estimators=100, max_depth=4, max_features=None, learning_rate=0.1, subsample=0.5, l1_penalty=0, l2_penalty=1, random_state=None, verbose=0, n_iter_no_change=0, tol=0, calc_feature_importances=False, validation_fraction=0)

Parameters

n_bins: int, optional
Specifies the number of bins to use for numeric variables. The default is 50.

min_samples_leaf : int, optional
Specifies the minimum number of observations on each node. The default is 5.

n_estimators : int, optional
Specifies the number of trees to create. The default is 100.

max_depth : int, optional
Specifies the maximum depth of the tree. The default is 4.

max_features : float, optional
Specifies the number of input variables to consider for splitting on a node. The default is the number of input variables.

learning_rate : float, optional
Specifies the learning rate of each tree. The default is 0.1.

subsample : float, optional
Specifies the fraction of the data to use for building each tree. The default is 0.5.

l1_penalty : float, optional
Specifies the L1 norm regularization on prediction. The default is 0.

l2_penalty : float, optional
Specifies the L2 norm regularization on prediction. The default is 1.

random_state : float, optional
Specifies the seed for the random number generator. By default, the random number stream is based on the computer clock. If you want a reproducible random number sequence between runs, specify a value greater than 0.

verbose : int, optional
Specifies the amount of information to display about the fitted model.

calc_feature_importances : bool, optional
Specifies whether or not to calculate feature importances.

validation_fraction: float, optional
Specifies the fraction of training sample to use for validation. A value greater than 0 enables early stopping mode.

n_iter_no_change : int, optional
Specifies the maximum number of training iterations for early stopping. This parameter is ignored unless a validation fraction is specified.

tol : float, optional
Specifies the tolerance value for early stopping as the change relative to the previous iteration. This parameter is ignored unless a validation fraction is specified.

Attributes

details_ : Results
Specifies information about the model fitting process and the final results.

n_estimators_ : int
Specifies the number of trees in the fitted model. The number could be less than the specified value because of early stopping.

avg_n_leaves_ : float
Specifies the average number of leaves among trees in the fitted model.

min_n_leaves_ : int
Specifies the minimum number of leaves among trees in the fitted model.

max_n_leaves_ : int
Specifies the maximum number of leaves among trees in the fitted model.

max_n_nodes_ : int
Specifies the maximum number of nodes among trees in the fitted model.

min_n_nodes_ : int
Specifies the minimum number of nodes among trees in the fitted model.

max_depth_ : int
Specifies the maximum depth among trees in the fitted model.

min_depth_ : int
Specifies the minimum depth among trees in the fitted model.

min_samples_leaf_ : int
Specifies the minimum leaf size among trees in the fitted model.

max_samples_leaf_ : int
Specifies the maximum leaf size among trees in the fitted model.

n_features_in_ : int
Specifies the number of input features.

feature_names_in_ : list(str)
Specifies the column names of the input features.

n_features_used_ : int
Specifies the number of features to be used by the fitted model.

target_name_in_ : str
Specifies the column name of the target variable.

feature_importances_ : pandas.DataFrame()
Specifies the ranking of features by importance.

distribution_ : str
Specifies the distribution function of the fitted model.

validation_scores_ : pandas.DataFrame()
Specifies the average square error for the validation sample by iteration.

Methods

Method Name	Description
customize_results	Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.
describe	Describes the type of model, its inputs, and its output.
export	Exports the internal SAS analytic store from a fitted model.
fit	Fit the model to the training data.
get_params	Gets parameters for this model.
predict	Predicts continuous values given input features.
save	Saves the model to a file.
score	Determines model accuracy by using the data set X and the targets y.
set_params	Updates the parameters of the model.

customize_results

```
customize_results(plots=None, width=None, height=None, dpi=None,
theme=None, path=None, output_type=None, include_tables=None,
exclude_tables=None, order=None)
```

Generates plots for all actions that support graphics, as well as filter tables from the results list, and creates custom result orders.

Parameters

plots : str or str list
Requests plots. You can request a single plot or a list of plots. The request can also be “all” or “none”. Arguments in parentheses are also allowed with each plot request. In addition, you can disable automatic plots from models by setting the environment variable AUTO_GRAPHICS to false. However, this variable does not disable automatic plot generation through the customize_results() method. You need to specify plots=”none” in the method to guarantee that only tables are returned. For more information, see SAS Viya: Managing Plots in Python Automatic Graphics.

width : int
Overrides the default width of the graph in pixels.

height : int
Overrides the default height of the graph in pixels.

dpi : int
Overrides the default resolution of the graph in dots per inch. The default is 96 dpi.

theme : {“light”, “dark”, “opal”, “midnight”, “raven”, “htmlencore”, “highcontrast”, “grayscale”, monochrome”}
Overrides the default theme that the graph uses. The default is “light”. You can specify the session default by setting the GRAPHICS_THEME environment variable.

path : str
Writes the plot results to the specified path in addition to the notebook.

output_type : {“svg”, “png”}
Overrides the default output type. The default type is “svg”, which contains image map information for tooltips. The “png” output type does not support tooltip information.

include_tables : str or str list
Includes only the specified tables from the displayed results. You can specify a keyword (“none”, “all”), a single table, or a list of tables. The default is “all”, which keeps all tables. The keyword “none” removes all tables, so that the results contain only plots that were generated. You can include tables by name (case-sensitive), and wildcarding is supported. These names appear in the top left

corner of all displayed results. If you specify this parameter along with the *exclude_tables* parameter, then the *include_tables* parameter takes precedence.

exclude_tables : str or str list
Excludes tables from the displayed results. You can specify a keyword (“none”, “graph_tables”, “all”), a single table, or a list of tables. The default is “none”, which displays all tables. The keyword “graph_tables” removes all tables that were used to produce a graph. The keyword “all” removes all tables, so that the results contain only plots that were generated. You can also exclude tables by name (case-sensitive), and wildcarding is supported. Those names appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* parameter, then the *include_tables* parameter takes precedence.

order : str or str list
Orders plot and tabular results for display. Any result that you do not specify in this parameter is excluded from the new results. The names for the results appear in the top left corner of all displayed results. If you specify this parameter along with the *include_tables* or *exclude_tables* parameter, those parameters are applied before the ordering is done. If the ordered list contains no results, then the *order* parameter is ignored.

Returns

CustomResults
Returns CustomResults that are a combination of both model results and plot results. These results do not replace the original model results. Instead, the CustomResults are typically assigned to a variable for later use or passed directly to the display() method for viewing in a Jupyter notebook cell.

describe

```
describe(options=None)
```

Describes the type of model, its inputs, and its output.

Parameters

options: dict, optional
Specifies additional runtime options to pass to the description. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

Results
Returns a mapping of various properties that describe aspects of the model.

export

```
export(file=None, replace=False)
```

Exports the internal SAS analytic store from a fitted model.

Generates an analytic store file that is common to other SAS products or procedures.

Parameters

file : str, pathlib.Path, or file-like, optional
Specifies the location where the exported model is saved. The location can be a path that specifies a file or an existing file-like object. It must be writable in binary format.

replace : bool, optional
Specifies whether or not to overwrite the *file* parameter if it already exists. This parameter is ignored unless the *file* parameter value is a file path.

Returns

bytes or None
Returns the analytic store in binary form if the *file* parameter is omitted. Otherwise None is returned.

fit

```
fit(X, y, nominals=None)
```

Fit the model to the training data.

Parameters

X : array-like
Specifies the training features in the shape (n_samples, n_features).

y : array-like
Specifies the target labels in the shape (n_samples,).

nominals : list(str), optional
Specifies the names of columns in *X* that should be treated as categorical. For classifier models, the target is assumed to be categorical.

Returns

self

get_params

```
get_params(deep=True)
```

Gets parameters for this model.

Parameters

deep : bool, default=True
This parameter is unused but is provided for compatibility with scikit-learn.

Returns

dict
Returns parameter names and current values.

predict

```
predict(X, copy_var=None, options=None)
```

Predicts continuous values given input features.

Parameters

X : array-like
Specifies the samples to predict values for, with the shape (n_samples, n_features).

copy_var : str or list of str, optional
Specifies variable(s) to be copied from the data. Can be a single variable name as a string or a list of variable names. For DuckDB queries, these variables will be added to the query if they don't already exist in X.

options: dict, optional
Specifies additional runtime options to pass to the prediction. The keys should be the option names as strings, and the values should be the corresponding option values.

Returns

array-like
Returns predicted continuous values in the shape (n_samples,).

save

```
save(path)
```

Saves the model to a file.

Save uses pickle and can be loaded with the sasviya.load_model.

Parameters

path : str or pathlib.Path
Specifies the location where the model is to be saved.

Returns

None

score

```
score(X, y, options=None)
```

Determines model accuracy by using the data set X and the targets y.

Parameters

X : array-like
Specifies the data set to score on.

y : array-like
Specifies the target values to compare the predictions to.

options: dict, optional
Specifies additional runtime options to pass to the scoring task. The keys should be the option names as strings, and the values should be the corresponding option values.

set_params

```
set_params(**params)
```

Updates the parameters of the model.

Parameters

****params : dict**
Specifies name:value pairs of the parameters to update.

Returns

self
Returns itself.