Eureqa Python API Documentation

Release 1.0

Dennis Oleksyuk

May 22, 2017

Contents

The Eureqa Python API provides a way to automate the entire Eureqa modeling process. Use it to:

- Connect to a data platform
- Automate model creation
- Generate summary reports
- Integrate with other modeling tools
- Deploy production models
- and much more!

The Python API integrates seamlessly with your Eureqa SaaS or Eureqa On Prem environment – access data, models, and analyses built via the API in the Eureqa UI and vice versa.

Please click on one of the links below or in the sidebar for more information.

CHAPTER 1

Getting Started

It is very easy to become productive with the Nutonian Python API - just ensure that you have the prerequisite Python version and Eureqa libraries installed, obtain an access key, and then connect. Details are below.

Installation

Prerequisites

Verify Python Installation

The Eureqa API requires Python 2.x version 2.7.11 or later. It is not tested with Python 3.x. Before getting started, ensure you have the correct version of python installed.

To verify your Python version in your terminal:

```
python --version
```

If Python is not installed, you can download it here: https://www.python.org.downloads

Install or Verify Pip

The Eureqa Python API is installed using *pip*. If you don't have pip, follow the installation instructions here: https://pip.pypa.io/en/stable/installing/

Eureqa API Installation

Installing the Eureqa API

To install the Eureqa API, use your terminal to run:

```
pip install eureqa[==version]
```

You can find the correct version by opening Eureqa and going to Settings > Eureqa API

Upgrading or Downgrading the Eureqa API

To upgrade or downgrade the API, install again with the version specified, e.g.:

```
pip install eureqa[==version]
```

Note: pip does have "-upgrade" flag which will upgrade to the latest version. We recommend *not* using this flag and instead using the command above to install the specific version that you need.

Uninstalling the Eureqa API

To uninstall the Eureqa API:

```
pip uninstall eureqa
```

Connecting to Eureqa

To get started with the API, connect to Eureqa as follows:

Eureqa SaaS:

```
from eureqa import Eureqa

e = Eureqa(url='Eureqa URL', user_name='user name', key='access key')
```

Where:

- 'Eureqa URL' is the base URL used to connect to Eureqa (e.g. "https://rds.nutonian.com")
- 'user name' is your Eureqa user name (e.g. ”bob@nutonian.com")
- 'access key' is your Eureqa API access key (see below)

Eureqa OnPrem:

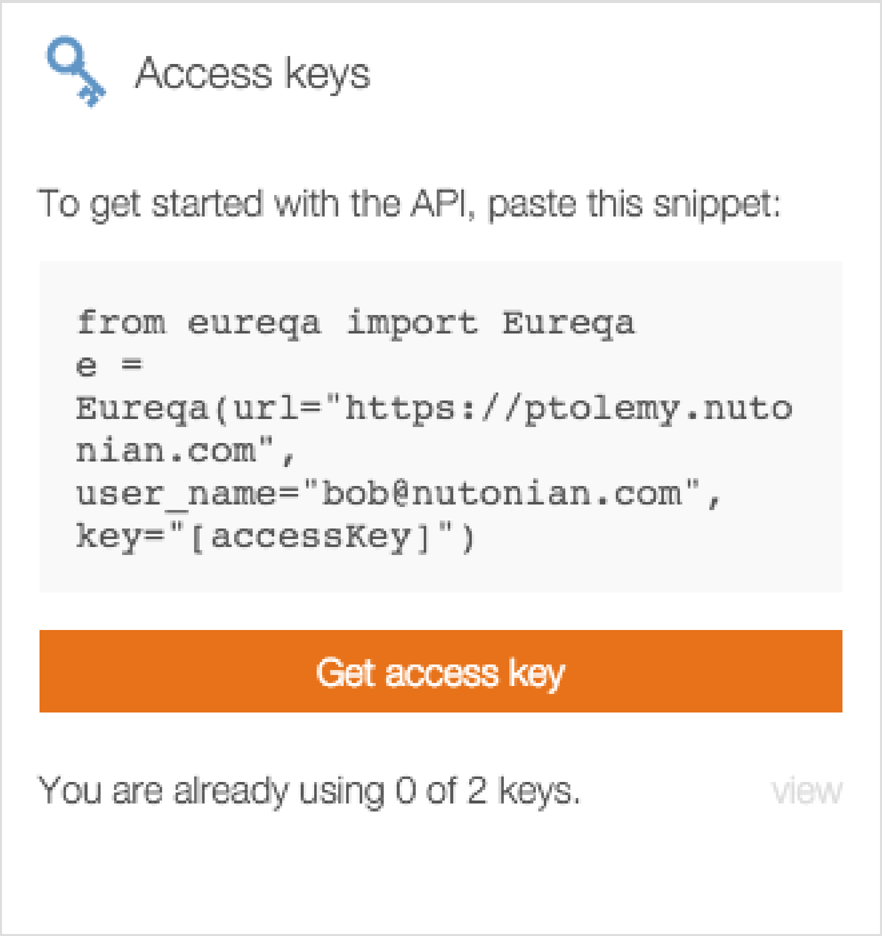
```
from eureqa import Eureqa

e = Eureqa(url='Eureqa URL', user_name='user name', key='access key', verify_ssl_certificate=False)
```

Where:

- 'Eureqa URL' is the base URL used to connect to your local installation of Eureqa (e.g. "https://eureqa.company.com:10002")
- 'user name' is your Eureqa user name (e.g. ”bob@nutonian.com")
- 'access key' is your Eureqa API access key (see below)
- 'verify_ssl_certificates=False' is needed if your local Eureqa installation hasn't been configured with SSL certificates

Access Keys

The Eureqa Python API uses key-based authentication. To generate a license key for your user account, open Eureqa and navigate to Settings > Python API and select the option to "Get access key"

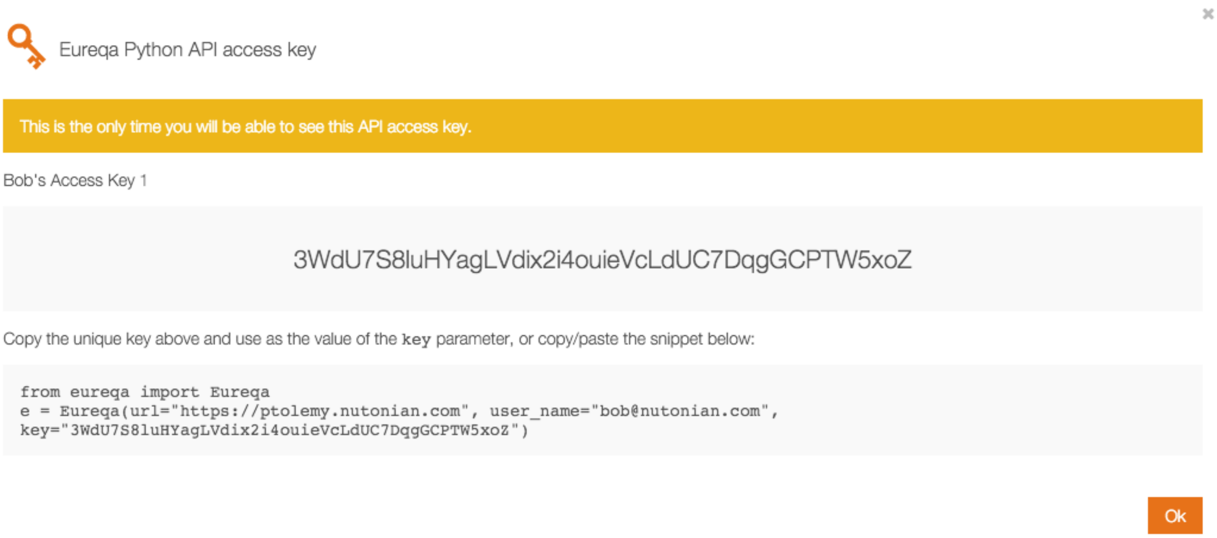


To get started with the API, paste this snippet:

```
from eureqa import Eureqa
e =
Eureqa(url="https://ptolemy.nuto
nian.com",
user_name="bob@nutonian.com",
key="[accessKey]")
```

You are already using 0 of 2 keys.

Edit the Key name / description if desired and then select Generate Key. Copy and paste the access key into your code – the key will only be shown once!



This is the only time you will be able to see this API access key.

Bob's Access Key 1

3WdU7S8luHYagLVdix2i4ouieVcLdUC7DqgGCPTW5xoZ

Copy the unique key above and use as the value of the `key` parameter, or copy/paste the snippet below:

```
from eureqa import Eureqa
e = Eureqa(url="https://ptolemy.nutonian.com", user_name="bob@nutonian.com",
key="3WdU7S8luHYagLVdix2i4ouieVcLdUC7DqgGCPTW5xoZ")
```

You may have up to 2 active keys a time. You can delete an existing key and generate a new key at any time.

Next Steps

Now that you have the Eureqa API installed and have successfully connected to Eureqa, get started uploading data, building models, generating reports and more.

Check out the *Examples* for sample code and the *Eureqa API Reference* for details on specific classes and functions.

CHAPTER 2

Examples

CreateSimpleSearch

Build a model to forecast sales and build an analysis in Eureqa that displays the results.

The example datasource includes weekly metrics for Sales along with marketing spend in various channels, website activity, and weather data.

Create a connection to Eureqa:

```
from eureqa import Eureqa

e = Eureqa(url="https://rds.nutonian.com", user_name="user@nutonian.com",
           key="JA24SZI94AKV0EJAEVPK")
```

Create a data source:

```
data_source = e.create_data_source("Sales & Marketing", "sales_marketing.csv")
```

```
<eureqa.data_source.DataSource instance at 0x00000000042F24C8>
```

`Download sales_marketing.csv.`

Create a search. The goal is to forecast Sales using all other variables in the datasource. The time series template default settings will work well for this search:

```
variables = set(data_source.get_variables())
target_variable = "Sales"
settings = e.search_templates.time_series("Sales Forecast", target_variable,
                                          variables - {target_variable})
search = data_source.create_search(settings)
```

Submit the search and wait until the search is done:

```
search.submit(30)
search.wait_until_done()
```

Get the best solution. Eureqa identifies the solution that gives the best trade-off between error and complexity:

```
solution = search.get_best_solution()
print("The best model found is:\n %s = %s" % (solution.target, solution.model))
```

```
The best model found is:
 Sales = 109569817.15572 + 1055.11588490586*delay(Emails_Sent, 1) + 1.0544595112534*wma(delay(Avg_Te
```

Get the model performance:

```
print("The %s value for this search is %.2f" % (solution.optimized_error_metric,
                                                 solution.optimized_error_metric_value))
```

```
The R^2 Goodness of Fit value for this search is 0.79
```

Create an analysis that summarizes the results:

```
analysis = e.create_analysis("Sales Forecasting Model",
                             "This model forecasts sales for next week.")
analysis.create_model_summary_card(solution)
analysis.create_model_fit_by_row_plot_card(solution, title="Model Fit with Sales Forecast",
                   description="View the predicted value for next week by positioning
                                           your mouse over the yellow predicted value.")
analysis.create_model_card(solution, title="Model Details",
                 description="Use the interactive model mode to see how predicted Sales
                           changes as the model inputs change.")
```

```
<eureqa.analysis_cards.model_card.ModelCard at 0x4629668>
```

```
Log into Eureqa to view and interact with the resulting analysis!
```

CreateCustomSearch

Create a search with customized search settings. Validate the model against a test datasource and get actual and predicted values to be used for additional analysis.

This datasource uses experimental data from a double pendulum. Variables represent the x positions (x1 and x2), velocities (v1 and v2), and accelerations (a1 and a2) of the two arms at different points in time.

Create a connection to Eureqa:

```
from eureqa import Eureqa
from eureqa import error_metric

e = Eureqa(url="https://rds.nutonian.com", user_name="user@nutonian.com",
           key="JA24SZI94AKV0EJAEVPK")
```

Create a data source:

```
data_source = e.create_data_source("Double Pendulum - Training Data",
                                   "double_pendulum_train.csv")
```

`Download double_pendulum_train.csv.`

Initialize search settings with the “numeric” template. Target variable is “a2”, the acceleration of the second arm of the double pendulum:

```
variables = set(data_source.get_variables())
target_variable = "a2"
settings = e.search_templates.numeric("Model a2", target_variable, variables - {target_variable, 't'
```

The double pendulum is a physical system. Customize the search settings to disable irrelevant building blocks like if-then-else, and to enable those that are relevant like the trigonometric operators:

```
settings.math_blocks.const.complexity = 1
settings.math_blocks.if_op.disable()   # disable if-then-else
settings.math_blocks.less.disable()   # disable less
settings.math_blocks.sin.enable(3)    # enable sine and set complexity to 3
settings.math_blocks.cos.enable(3)    # enable cosine and set complexity to 3
settings.error_metric = error_metric.mean_square_error()
```

Create a search and run for 30 seconds:

```
search = data_source.create_search(settings)
search.submit(30)
search.wait_until_done()
```

Get the best model, view the model and the error metrics:

```
solution = search.get_best_solution()
print("The best model found is:\n %s = %s" % (solution.target, solution.model))
```

```
The best model found is:
 a2 = -0.0239994769615275 - a1*cos(x2 - x1) - v1^2*sin(x2 - x1) - 9.81639183106058*sin(x2)
```

Get the model performance:

```
print("The %s value for this search is %.2f" % (solution.optimized_error_metric, solution.optimized_e
```

```
The Mean Squared Error value for this search is 0.49
```

Evaluate the model against a test datasource withheld from Eureqa:

```
test_data_source = e.create_data_source("Double Pendulum - Test Data", "double_pendulum_test.csv")
test_metrics = e.compute_error_metrics(test_data_source, target_variable, solution.model)
test_mse = test_metrics.mean_square_error
print("The %s value for the test data is %.2f" % (solution.optimized_error_metric, test_mse))
```

```
The Mean Squared Error value for the test data is 1.34
```

Download `double_pendulum_test.csv`.

Get the actual and predicted values for a test set and load into a DataFrame for future analysis:

```
predicted_and_actual = e.evaluate_expression(test_data_source, ["t", 'a2', solution.model])
import pandas as pd
df = pd.DataFrame(predicted_and_actual)
df
```

SearchAutomation

Automate a set of models and summarize model results. In this example we'll test how well we can model each variable in a datasource based on the other variables.

Create a connection to Eureqa:

```
from eureqa import Eureqa

e = Eureqa(url="https://rds.nutonian.com", user_name="user@nutonian.com",
           key="JA24SZI94AKV0EJAEVPK")
```

Get the list of variables in the datasource:

```
data_source = e.create_data_source("Sales & Marketing", "sales_marketing.csv")
variables = set(data_source.get_variables())
variables
```

```
{u'Avg_Temp',
 u'Days_Precipitation',
 u'Days_Rain',
 u'Days_Snow',
 u'Emails_Clicked',
 u'Emails_Opened',
 u'Emails_Sent',
 u'Flyer_Spend',
 u'Inches_Rain',
 u'Inches_Snow',
 u'Max_Temp',
 u'Min_Temp',
 u'Online_Ad_Spend',
 u'Sales',
 u'TV_Ad_Spend',
 u'Web_New_Visitors',
 u'Web_Returning_Visitors',
 u'Web_Total_Visits',
 u'Web_Unique_Visitors',
 u'Week'}
```

`Download sales_marketing.csv.`

Create a search for each variable, modeling the variable as the target with all other variables as inputs:

```
searches = []
for variable in variables:
    settings = e.search_templates.numeric("Model for variable: %s" % variable,
                                          variable, variables - {variable})
    search = data_source.create_search(settings)
    search.submit(2)
    searches.append(search)
searches
```

```
[<eureqa.search.Search instance at 0x00000000046A3E48>,
 <eureqa.search.Search instance at 0x00000000046A3848>,
 <eureqa.search.Search instance at 0x000000000472BDC8>,
 <eureqa.search.Search instance at 0x00000000046A39C8>,
 <eureqa.search.Search instance at 0x00000000046A3608>,
 <eureqa.search.Search instance at 0x0000000004748E48>,
 <eureqa.search.Search instance at 0x00000000046E6A48>,
 <eureqa.search.Search instance at 0x0000000004701D48>,
 <eureqa.search.Search instance at 0x0000000004748A08>,
 <eureqa.search.Search instance at 0x0000000004748A48>,
 <eureqa.search.Search instance at 0x0000000004701F48>,
 <eureqa.search.Search instance at 0x000000000480A748>,
 <eureqa.search.Search instance at 0x00000000047EA408>,
 <eureqa.search.Search instance at 0x000000000486AB08>,
 <eureqa.search.Search instance at 0x000000000480A588>,
 <eureqa.search.Search instance at 0x0000000004889A48>,
 <eureqa.search.Search instance at 0x000000000486A548>,
 <eureqa.search.Search instance at 0x00000000048FEF88>,
 <eureqa.search.Search instance at 0x0000000004911C88>,
 <eureqa.search.Search instance at 0x00000000048FE888>]
```

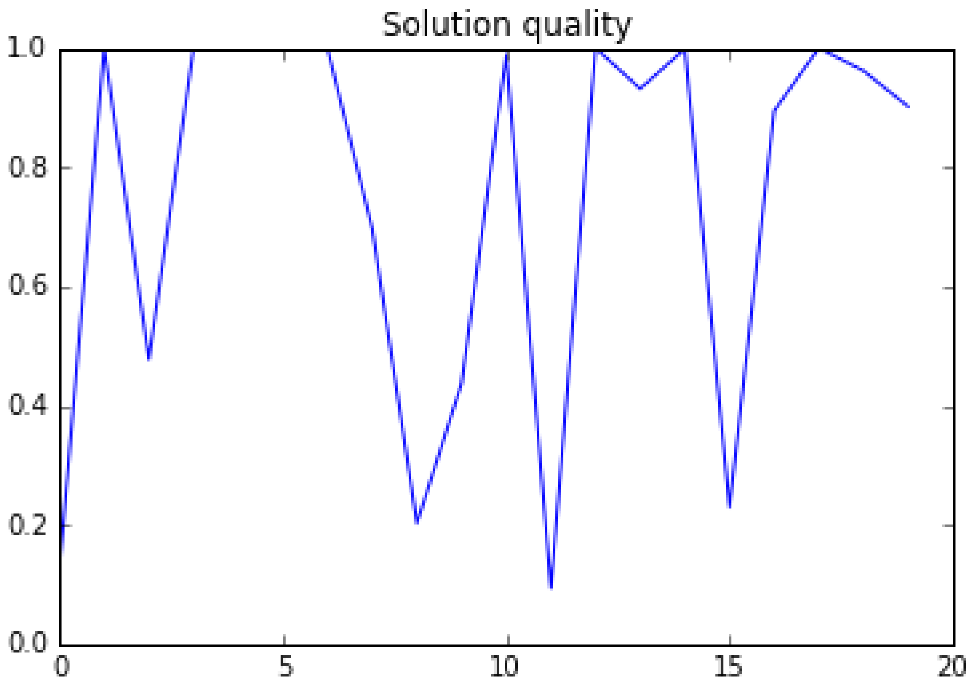
Wait for searches to complete. Print the best solution for each search as they complete:

```
for search in searches:
    search.wait_until_done()
    best_solution = search.get_best_solution()
    print 'Best model for the %s variable is %s' % (best_solution.target, best_solution.model)
```

```
Best model for the Week variable is 0.318191337724807 + 0.648759972160649*step(14.9834893404746 + 22
Best model for the Web_Unique_Visitors variable is 1.04484030337699 + 1.00003420535283*Web_New_Visito
Best model for the Inches_Rain variable is 0.14180331189039 + 0.389285765629462*Days_Rain + 0.012459
Best model for the Web_Returning_Visitors variable is 1.00001733340471*Web_Unique_Visitors - 0.94661
Best model for the Days_Snow variable is Days_Precipitation - Days_Rain
Best model for the Days_Precipitation variable is Days_Rain + Days_Snow
Best model for the Emails_Opened variable is Week + 3.58602233572101*Emails_Clicked + 1.795696851791
Best model for the Sales variable is 88547999.0622628 + 553925.992067077*Avg_Temp + 57999.4727128087
Best model for the Inches_Snow variable is Days_Snow
Best model for the Online_Ad_Spend variable is 324756.430393052 + Emails_Opened*Emails_Clicked*less(
Best model for the Emails_Sent variable is 4.88394640914473*Emails_Opened + 1.36338463476562*min(Ema
Best model for the TV_Ad_Spend variable is 230229.215764672 + 11226.1289950602*Inches_Snow + 5367.92
Best model for the Web_New_Visitors variable is 0.999992737715856*Web_Unique_Visitors - 0.97936248010
Best model for the Avg_Temp variable is 36.4457896586784 + 1.52605710485033*Max_Temp + 6.71485944864
Best model for the Emails_Clicked variable is 4.70332166064393 + 0.249648462804642*Emails_Opened - 0
Best model for the Flyer_Spend variable is 121967.639533629 + 1.27247056501962*Web_New_Visitors*Week
Best model for the Min_Temp variable is if(15.2531317411839, Avg_Temp, 15.2531317411839) - 7.4821757
Best model for the Days_Rain variable is Days_Precipitation - Days_Snow
Best model for the Web_Total_Visits variable is 15.6700031160911*Emails_Opened + 7.8982810405798*Web_
Best model for the Max_Temp variable is 8.63740515122003 + max(1 + 0.961215843860963*Avg_Temp, Min_Te
```

Plot the error metric values to show which variables are easiest and hardest to predict:

```
%matplotlib inline
import matplotlib.pyplot as plt

error_metric = searches[0].error_metric
solutions = [search.get_best_solution() for search in searches]

plt.plot(range(len(solutions)), [s.get_error_metric_value(error_metric) for s in solutions])
plt.title('Solution quality')
plt.show()
```



AnalysisCustomPlot

Build two custom plots and add them to an analysis.

Create a connection to Eureqa:

```
from eureqa import Eureqa
from eureqa.analysis_cards import Plot

e = Eureqa(url="https://rds.nutonian.com", user_name="user@nutonian.com",
           key="JA24SZI94AKV0EJAEVPK")

analysis = e.create_analysis('Custom Plot Analysis')
```

Create a plot with three components. The X and Y positions for these components are specified directly using Python lists.

```
plot = Plot()
plot.plot([1,2,3], [4,5,6], color='blue', line_width=3, legend_label='Label 1')
plot.plot([11,22,33], [44,55,66], style='o', legend_label='Label 2',
          color='red', circle_radius=7)
plot.plot([10,11,12], [13,14,15], line_width=5, color='orange')
plot_card = analysis.create_custom_plot_card(plot, 'This card contains my custom plot with data from
```

Create a second plot with two components. The X and Y positions for these components come from variables within a Eureqa data source. The “<row>” variable is the row number from the data source.

```
datasource = e.create_data_source('Nutrition Data', 'nutrition.csv')
plot2 = Plot(width='500px', height='500px', x_axis_label='x axis label', y_axis_label='y axis label')
plot2.plot('<row>', 'Calories', datasource=datasource, style='o', line_width=3,
           circle_radius=5, color='blue', legend_label='Calories')
plot2.plot('<row>', '30*(Fat (g))', datasource=datasource, style='-', line_width=3,
          circle_radius=5, color='red', legend_label='30*(Fat (g))')
plot_card_2 = analysis.create_custom_plot_card(plot2, 'This plot contains data from my Nutrition data
```

`Download nutrition.csv.`

Log into Eureqa and navigate to the Analysis area to view the custom plots.

CHAPTER 3

Eureqa API Reference

analysis

class `eureqa.analysis.`**`Analysis`**

Represents an analysis on the server.

Variables

- ***`name`*** (*`str`*) – The analysis name.
- ***`description`*** (*`str`*) – The analysis description.

`add_card`(*card*)

Add an analysis Item to this analysis

Parameters **`card`** (`Card`) – Card to add to the analysis. Must not have been previously added or retrieved from an Analysis.

`analysis_id`

The id of the analysis

`create_binned_mean_plot_card`(*datasource*, *x_var*, *y_var*, *title=None*, *description=None*, *needs_guides=False*, *axis_labels=None*, *label_format=None*, *collapse=False*)

Creates a new binned-mean-plot card. Adds the card to this analysis.

Parameters

- **`datasource`** (`~DataSource`) – Data source for the card's data
- **`x_var`** (*`str`*) – The X-axis variable for the card's plot
- **`y_var`** (*`str`*) – The Y-axis variable for the card's plot
- **`title`** (*`str`*) – The title of the card
- **`description`** (*`str`*) – The card's description.
- **`needs_guides`** (*`bool`*) – Whether the card needs guides
- **`axis_labels`** (*`list`*) – Axis labels for this card's plot
- **`label_format`** (*`list`*) – Label format for this card
- **`collapse`** (*`bool`*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type BinnedMeanPlotCard

create_box_plot_card(*datasource*, *x_var*, *y_var*, *title=None*, *description=None*, *needs_guides=False*, *axis_labels=None*, *label_format=None*, *collapse=False*)

Creates a new box-plot card. Adds the card to this analysis.

Parameters

- **datasource** (`DataSource`) – Data source for the card's data
- **x_var** (*str*) – The X-axis variable for the card's plot
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **title** (*str*) – The title of the card
- **description** (*str*) – The description of the card
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*list*) – Axis labels for this card's plot
- **label_format** (*str*) – Label format for this card
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type BoxPlotCard

create_by_row_plot_card(*datasource*, *x_var*, *plotted_vars*, *title=None*, *description=None*, *focus_variable=None*, *should_center=True*, *should_scale=False*, *collapse=False*)

Create a new by-row plot card. Adds the card to this analysis.

Parameters

- **datasource** (`DataSource`) – Data source for the card's data
- **x_var** (*str*) – Name of the variable to plot as the X axis
- **plotted_vars** (*str*) – List of string-names of variables to plot. (To modify a variable's display name, first create the card; then modify the display name directly on it.)
- **title** (*str*) – The card's title.
- **description** (*str*) – The card's description.
- **focus_variable** (*str*) – Name of the variable in 'plotted_vars' to bring to the foreground
- **should_center** (*bool*) – Should the plot be centered?
- **should_scale** (*bool*) – Should the plot scale?
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type ByRowPlotCard

create_card(*component*)

Create a new Card on this Analysis containing only the specified Component

Parameters **component** (*Component*) – instance of the class add to the created Item

Returns the created :class:Card

Example:

```
    card = e.create_card(comp)
```

create_custom_plot_card(*plot*, *title=None*, *description=None*, *collapse=False*)

Creates a new custom plot card. Adds the card to this analysis.

Parameters

- **plot** (CustomPlot) – The Plot to be displayed in the card.
- **title** (*str*) – The card title.
- **description** (*str*) – The card's description.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type *Card*

create_distribution_plot_card(*data_source*, *variable*, *title=None*, *description=None*, *collapse=False*)

Creates a new distribution plot card. Adds the card to this analysis.

Parameters

- **data_source** (DataSource) – The data source to which the variable belongs.
- **variable** (*str*) – The name of the variable that will be displayed on the card.
- **title** (*str*) – The card title.
- **description** (*str*) – The card's description.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type DistributionPlotCard

create_double_histogram_plot_card(*datasource*, *x_var*, *y_var*, *title=None*, *description=None*, *needs_guides=False*, *axis_labels=None*, *label_format=None*, *collapse=False*)

Creates a new double-histogram card. Adds the card to this analysis.

Parameters

- **datasource** (DataSource) – Data source for the card's data
- **x_var** (*str*) – The X-axis variable for the card's plot
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **title** (*str*) – The title of the card
- **description** (*str*) – The description of the card
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*list*) – Axis labels for this card's plot
- **label_format** (*str*) – Label format for this card
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type DoubleHistogramPlotCard

create_html_card(*html*, *title='HTML'*, *description=None*, *collapse=False*)
Creates a new HTML card. Adds the card to this analysis.

Parameters

- **html** (*str*) – The card's HTML body.
- **title** (*str*) – The card title.
- **description** (*str*) – Deprecated; unused. (This card doesn't have a Description.)
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type HtmlCard

create_image_card(*image_path*, *title=None*, *description=None*, *collapse=False*)
Creates a new text card containing only header text and one image.

Parameters

- **image_path** (*str*) – the filepath to the image in your filesystem.
- **title** (*str*) – The card title.
- **description** (*str*) – a description of the card, to appear above the image
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type TextCard

create_model_card(*solution*, *title=None*, *description=None*, *collapse=False*)
Creates a new model card. Adds the card to this analysis.

Parameters

- **solution** (*eureqa.Solution*) – The solution that will be displayed on the card.
- **title** (*str*) – The card title.
- **description** (*str*) – The card description.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type ModelCard

create_model_evaluator_card(*solution*, *title=None*, *description=None*, *collapse=False*)
Creates a new model evaluator card. Adds the card to this analysis.

Parameters

- **solution** (*eureqa.Solution*) – Solution to fetch results from for the primary model evaluator
- **title** (*str*) – Title of the card. Defaults to 'Evaluate Model'.
- **description** (*str*) – Description of the card.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card.

Return type *Card*

Additional models can be added to this model by calling the `add_solution_info` method on the returned object.

`create_model_fit_by_row_plot_card`(*solution*, *title=None*, *description=None*, *collapse=False*, *x_axis=None*)

Create a new model fit by-row plot card. Adds the card to this analysis. Note that by-row plots are meant for use with Numeric searches. They may not work properly if used with other types of searches.

Parameters

- **`solution`** (*`Solution`*) – The Solution object for which this card is being created
- **`title`** (*`str`*) – The card's title.
- **`description`** (*`str`*) – The card's description.
- **`collapse`** (*`bool`*) – Whether the card should default to be collapsed.
- **`x_axis`** (*`str`*) – **Ignored**

Returns Object representing the created card

Return type ModelFitByRowPlotCard

`create_model_fit_plot_card`(*solution*, *x_axis=None*, *title=None*, *description=None*, *collapse=False*)

Create a new model fit card. Adds the card to this analysis. Automatically choose the correct type of card (by-row plot or separation plot) based on the specified search. Numeric searches use by-row plots; time-series searches use separation plots.

Parameters

- **`solution`** (*`Solution`*) – The Solution object for which this card is being created
- **`x_axis`** (*`str`*) – **Ignored**
- **`title`** (*`str`*) – The card's title.
- **`description`** (*`str`*) – The card's description.
- **`collapse`** (*`bool`*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type ModelFitPlotCard

`create_model_fit_separation_plot_card`(*solution*, *title=None*, *description=None*, *collapse=False*)

Create a new model fit separation-plot card. Adds the card to this analysis. Note that separation plots are meant for use with classification searches. They may not work properly if used with other types of searches.

Parameters

- **`solution`** (*`Solution`*) – The Solution object for which this card is being created
- **`title`** (*`str`*) – The card's title.
- **`description`** (*`str`*) – The card's description.
- **`collapse`** (*`bool`*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type ModelFitSeparationPlotCard

`create_model_summary_card`(*solution*, *collapse=False*)

Creates a new model summary card. Adds the card to this analysis.

Parameters

- **solution** (*eureqa.Solution*) – The solution that will be displayed on the card.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type ModelSummaryCard

create_model_terms_plot_card(*solution*, *title=None*, *description=None*, *collapse=False*)

Create a model terms plot card. Adds the card to this analysis.

Parameters

- **solution** (*Solution*) – The solution object for which this card is being created
- **title** (*str*) – The card's title.
- **description** (*str*) – The card's description.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type *ModelTermsPlot*

create_most_frequent_variables_plot_card(*search*, *title=None*, *description=None*, *collapse=False*)

Create a new most frequent variables plot card. Adds the card to this analysis.

Parameters

- **search** (*Search*) – The Search object for which this card is being created
- **title** (*str*) – The card's title.
- **description** (*str*) – The card's description.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type *MostFrequentVariablesPlot*

create_scatter_plot_card(*datasource*, *x_var*, *y_var*, *title=None*, *description=None*, *needs_guides=False*, *axis_labels=None*, *label_format=None*, *collapse=False*)

Creates a new scatter-plot card. Adds the card to this analysis.

Parameters

- **datasource** (*DataSource*) – Data source for the card's data
- **x_var** (*str*) – The X-axis variable for the card's plot
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **title** (*str*) – The title of the card
- **description** (*str*) – The card's description
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*list*) – Axis labels for this card's plot
- **label_format** (*list*) – Label format for this card
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type ScatterPlotCard

create_text_card(*text*, *title='Text'*, *description=None*, *collapse=False*)
Creates a new text card. Adds the card to this analysis.

Parameters

- **text** (*str*) – The card text.
- **title** (*str*) – The card title.
- **description** (*str*) – Deprecated; unused. (This card doesn't have a Description.)
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns An object that represents the newly created card.

Return type TextCard

create_threshold_selection_plot_card(*solution*, *title=None*, *description=None*, *collapse=False*)
Create a threshold selection plot card. Adds the card to this analysis.

Parameters

- **solution** (*Solution*) – The Solution object for which this card is being created
- **title** (*str*) – The card's title.
- **description** (*str*) – The card's description.
- **collapse** (*bool*) – Whether the card should default to be collapsed.

Returns Object representing the created card

Return type *ThresholdSelectionPlot*

delete()
Deletes the analysis from the server.

description
The description of the analysis

get_cards()
Get all Analysis Cards associated with this Analysis

Returns A list of Card objects that are rendered as part of this Analysis

Return type list of *Card*

get_components()
Get all Analysis Components associated with this Analysis

Returns A list of Component objects attached to this Analysis, that may be rendered by Items in the Analysis

Return type list of _Component

name
The name of the analysis

upload_image(*image_path*)
Upload an image to the server, to be embedded in analysis cards.

Parameters **image_path** (*str*) – the filepath to the image on your filesystem.

Returns An object that represents the newly uploaded image.

Return type *Image*

analysis_file

class `eureqa.analysis.analysis_file.`**`AnalysisFile`**(*_analysis*, *_file_id*)

AnalysisFile object

Represents a file that is attached to an Analysis, and can be embedded in various places within the Analysis. AnalysisFiles are often either images (.jpg, .png, etc) or raw data files (.csv, .xls, etc), but any type of file is supported.

Do not construct this class directly. Use `AnalysisFile.create()`

classmethod **`create`**(*_cls*, *analysis*, *file*, *filename=None*)

Upload a new file to an Eureqa Analysis.

Parameters

- **`analysis`** (`eureqa.analysis.Analysis`) – Analysis to upload the file to
- **`file`** (*`str`*) – File to upload. Can be either a string, in which case it is treated as the binary data of a file, or an actual Python file object or file-like object.
- **`filename`** (*`str`*) – The default name of file when downloaded

Return type *AnalysisFile*

`delete`()

Delete this file from the server

`get`()

Return the raw data of this file

Return type AnalysisFile.file

`update`(*file*)

Update the contents of an existing file on the server. Overwrites the file's previous data.

Parameters **`file`** (*`str`*) – File to upload. Can be either a string, in which case it is treated as the binary data of a file, or an actual Python file object or file-like object.

`url`()

Returns URL that this image can be reached from

cards

class `eureqa.analysis.cards.`**`Card`**

The initially visible content of an Analysis is defined as a list of `Card` s. Cards are ordered and can be collapsed by default, and each has a default Component which is visble.

You should not create a `Card` manually, but rather use `create_card()`

Variables ***`collapse`*** (*`bool`*) – If the Card is currently collapsed or not

`collapse`

Whether this item should be rendered as "collapsed" (with its content hidden)

Type bool

`move_above`(*other_card*)

Moves this card above another card.

Parameters **other_card** (Card) – The other card above which to move this card.

move_below (*other_card*)

Moves this card below another card.

Parameters **other_card** (Card) – The other card object below which to move this card.

components

This module contains components used to build up analyses.

To add a component to an analysis, you use *Analysis.create_card* method:

```
h = HtmlBlock('This will render on a Card in an Analysis')
analysis.create_card(h)
```

Or the *Analysis.create_html_card* convenience methods:

```
vl = VariableLink(datasource=self._data_source, variable_name=self._variables[0])
analysis.create_html_card("Target variable is: {0}".format(analysis.html_ref(vl)))
```

class eureqa.analysis.components.**ModelFitByRowPlot** (*solution=None*, *use_all_data=None*)

A model fit by row plot card. See also *Analysis.create_model_fit_by_row_plot_card*

For example:

```
    p = ModelFitByRowPlot(s.get_best_solution())
    analysis.create_card(p)
```

Parameters

- **solution** (Solution) – The solution that will be displayed on the card.
- **use_all_data** (*bool*) – If true, uses all data int he datasource. Otherwise use training set

solution

The Solution that is being explained

Return type eureqa.Solution

use_all_data

Use all data or just validation data?

Return type bool

class eureqa.analysis.components.**Model** (*solution=None*)

Represents an interactive model explainer component on the server.

Parameters **solution** (Solution) – The solution that will be displayed on the card.

Variables ***solution*** (Solution) – The solution that will be displayed on the card.

solution

The Solution that is being explained

Return type eureqa.Solution

class eureqa.analysis.components.**ModelFitSeparationPlot** (*solution=None*, *use_all_data=None*)

This component is a model fit separation-plot. Separation plots are meant for use with time-series searches. They may not work properly if used with other types of searches.

Parameters

- **solution** (Solution) – The Solution object for this component
- **use_all_data** (*bool*) – If true, uses all data int he datasource. Otherwise use training set

solution

The Solution that is being explained

Return type eureqa.Solution

use_all_data

Use all data or just validation data?

Return type bool

class eureqa.analysis.components.**ModelEvaluator**(*solutions=None*, *solution=None*)

This component evaluates models against different datasources and compares their performance.

Additional models can be added to this model by calling *add_solution_info()*

For example:

```
c = ModelEvaluator(solutions=s.get_solutions(), datasource=d, search=s, solution=s.get_best_solu
analysis.create_card(c)
```

Parameters

- **solutions** (*list[Solution]*) – Solutions to evaluate against the data
- **solution** (Solution) – Solution to fetch results from for the primary model evaluator

class SolutionInfo(*component*, *body*)

Do not instantiate directly. Use *ModelEvaluator.add_solution_info()* instead. The solution information for a single model evaluation ("tab") on ModelEvaluatorCard.

Parameters

- **body** (*str*) – internal
- **component** (*_Component*) – internal

Variables

- **datasource_id** (*str*) – ID of the DataSource referenced by this solution-tab
- **search_id** (*str*) – ID of the Search referenced by this solution-tab
- **solution_id** (*str*) – ID of the Solution referenced by this solution-tab
- **has_target_variable** (*bool*) – Whether the datasource contains the target variable

accuracy

Accuracy of this Solution. Rendered as a human-readable pretty-printed string.

Return type str

has_target_variable

Whether the datasource contains the solution's target variable.

If the datasource contains the target variable, the standard plot of this Component may include the raw target-variable data for comparison alongside the computed value.

Returns Whether the datasource contains the target variable

solution

The Solution that is being explained

Return type *Solution*

ModelEvaluator.**add_solution_info**(*solution*)

Add a new (non-default) solution and tab to this card. Once added, this object will show up in the list returned by *solution_infos*.

Parameters **solution** (Solution) – Solution associated with the model evaluation.

ModelEvaluator.**clear_solution_infos**()

Remove all existing solution infos from the current Component

ModelEvaluator.**solution**

The Solution that is being explained

Return type *Solution*

ModelEvaluator.**solution_infos**

The set of all SolutionInfo objects associated with this card. One per solution tab displayed in the UI. Note that *solution_infos[0]* is the default card; it may be treated specially by the UI.

To add a solution, use the "add_solution_info()" method.

Returns List or tuple of *SolutionInfo* objects

class eureqa.analysis.components.**DistributionPlot**(*datasource=None*, *variable_name=None*)

Represents a distribution plot card on the server.

Parameters

- **datasource** (*eureqa.DataSource*) – The data source to which the variable belongs.
- **variable_name** (*str*) – The name of the variable that will be displayed on the card.

Variables

- **title** (*str*) – The card title.
- **datasource** (*str*) – The datasource used by the card.
- **variable** (*str*) – The variable plotted by the coard.

datasource

The data source providing data for this component

Returns data source providing data for this component

variable

(Deprecated) Name of the variable to plot. For backwards compatibility.

Return type str

variable_name

Name of the variable to plot

Return type str

class eureqa.analysis.components.**BoxPlot**(*datasource=None*, *axis_labels=None*, *label_format=None*, *needs_guides=None*, *x_var=None*, *y_var=None*)

Represents a box plot card on the server.

Example:

```
bp = BoxPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':'.3s'}, x_v
analysis.create_card(bp)
```

Parameters

- **datasource** (DataSource) – The data source containing the data to be plotted.
- **axis_labels** (*XYMap*) – Axis labels for this card's plot (XYMap). Set member fields "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".
- **x_var** (*str*) – The X-axis variable for the component's plot. (must be binary)
- **y_var** (*str*) – The Y-axis variable for the component's plot.
- **needs_guides** (*bool*) – Whether the card needs guides.

Variables

- **title** (*str*) – The title of the card
- **x_var** (*str*) – The X-axis variable for the card's plot (must be binary)
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set member fields "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

axis_labels

The axis labels for this card

defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type self.XYMap

datasource

The *DataSource* providing data for this component

Returns *DataSource* providing data for this card

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

defaults to:

```
{'x': 'g', 'y': '.2s'}
```

Return type XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card.

Returns X variable for this card. (must be binary)

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

class eureqa.analysis.components.**DoubleHistogramPlot**(*datasource=None*, *axis_labels=None*, *label_format=None*, *needs_guides=None*, *x_var=None*, *y_var=None*)

Represents a double-histogram plot component on the server.

For example:

```
p = DoubleHistogramPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':
analysis.create_card(p)
```

Parameters

- **datasource** (DataSource) – The data source containing the data to be plotted.
- **x_var** (*str*) – The X-axis variable for the card's plot. (must be binary)
- **y_var** (*str*) – The Y-axis variable for the card's plot.
- **needs_guides** (*bool*) – Whether the card needs guides.
- **axis_labels** (*dict*) – Axis labels for this card's plot. Set keys "x" and "y" to set the X and Y axis labels.
- **label_format** (*dict*) – Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

Variables

- **title** (*str*) – The title of the card
- **x_var** (*str*) – The X-axis variable for the card's plot (must be binary)
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set member fields "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

axis_labels

The axis labels for this card

defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type XYMap

datasource

The data source providing data for this component

Returns data source providing data for this card

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

defaults to:

```
{ 'x': '.3s', 'y': 'g' }
```

Return type XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card. (must be binary)

Returns X variable for this card

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

class eureqa.analysis.components.**ScatterPlot** (*datasource=None*, *axis_labels=None*, *label_format=None*, *needs_guides=None*, *x_var=None*, *y_var=None*)

Creates a new scatter-plot card.

For example:

```
p = ScatterPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':'.3s'},
analysis.create_card(p)
```

Parameters

- **datasource** (DataSource) – Data source for the card's data
- **x_var** (*str*) – The X-axis variable for the card's plot
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*list*) – Axis labels for this card's plot
- **label_format** (*list*) – Label format for this card

axis_labels

The axis labels for this card.

Defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type self.XYMap

datasource

The data source providing data for this component

Return type eureqa.DataSource

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

Defaults to:

```
{ 'x': '.3s', 'y': '.3s' }
```

Return type DoubleHistogramPlot.XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card.

Returns X variable for this card

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

class eureqa.analysis.components.**BinnedMeanPlot**(*datasource=None*, *axis_labels=None*, *label_format=None*, *needs_guides=None*, *x_var=None*, *y_var=None*)

Represents a binned mean plot component on the server.

For example:

```
p = BinnedMeanPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':'.3s'
analysis.create_card(p)
```

Parameters

- **datasource** (DataSource) – The data source containing the data to be plotted.
- **x_var** (*str*) – The X-axis variable for the card's plot.

- **y_var** (*str*) – The Y-axis variable for the card's plot.
- **needs_guides** (*bool*) – Whether the card needs guides.
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set keys "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

Variables

- **title** (*str*) – The title of the card
- **x_var** (*str*) – The X-axis variable for the card's plot
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set member fields "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

axis_labels

The axis labels for this card

defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type XYMap

datasource

The data source providing data for this component

Returns data source providing data for this card

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

defaults to:

```
{ 'x': '.3s', 'y': '.2f' }
```

Return type XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card.

Returns X variable for this card

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

class eureqa.analysis.components.**ByRowPlot** (*datasource=None*, *x_var=None*, *plotted_variables=None*, *focus_variable=None*, *should_center=None*, *should_scale=None*)

Represents a line graph for plotting variables by row.

For example:

```
p = ByRowPlot(d, x_var='W', focus_variable='W', plotted_variables=['A'], should_center=True)
analysis.create_card(p)
```

Parameters

- **datasource** (*eureqa.DataSource*) – Data source for the component's data
- **x_var** (*str*) – Name of the variable to plot as the X axis
- **plotted_variables** (*list*) – List of variable names to plot.
- **focus_variable** (*str*) – Name of the variable in *plotted_variables* to bring to the foreground (required)
- **should_center** (*bool*) – Should the plot be centered?
- **should_scale** (*bool*) – Should the plot scale?

Variables

- **eureqa.analysis.components.variable_line_graph.ByRowPlot.focus_variable** (*str*) – Focused (foreground) variable for the component. Must be a member of *plotted_variables*
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.x_var** (*str*) – Name of the variable to plot as the X axis
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.plotted_variables** (*list*) – Variables to plot. (List of string variable names.)
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.should_center** (*bool*) – Should the plot be centered?
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.should_scale** (*bool*) – Should the plot scale?

datasource

The data source providing data for this component

Returns data source providing data for this component

focus_variable

The variable that is currently in focus (in the foreground) for this component. Must be a member of *plotted_variables*.

Returns focus_variable for this component

Return type str

plotted_variables

The plotted variables for this component.

Returns List of the names of the variables being plotted against the X axis

Return type list

should_center

The should_center option for this component.

Returns whether this plot should be centered

Return type bool

should_scale

The should_scale option for this component.

Returns whether this plot should be scaled

Return type bool

x_var

The X-axis variable for this component

Returns the name of the X-axis variable for this component

Return type str

class eureqa.analysis.components.**CustomPlot**(*datasource=None*, *width=None*, *height='400px'*, *x_axis_label=None*, *y_axis_label=None*, *show_legend=True*, *zero_x_line=False*, *zero_y_line=False*, *x_tick_format=None*, *y_tick_format=None*, *guides_type='XY'*)

Represents a plot to be displayed in an analysis plot card.

By calling the plot() method multiple times on one instance of this class, multiple lines and scatter plots can be superimposed on this object.

Parameters

- **datasource** (DataSource) – The data source containing the data to be plotted.
- **width** (*str*) – Set manual dimensions for the plot in "px" units (e.g. 350px). Defaults to full panel width.
- **height** (*str*) – Set manual dimensions for the plot in "px" units. Defaults to a constant height of 400px.
- **x_axis_label** (*str*) – The label to use on the x axis of the plot.
- **y_axis_label** (*str*) – The label to use on the y axis of the plot.
- **show_legend** (*bool*) – Controls whether or not a legend will be used in the plot.
- **zero_x_line** (*bool*) – Draws a horizontal line through the origin
- **zero_y_line** (*bool*) – Draws a vertical line through the origin
- **x_tick_format** (*str*) – Format for x axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format
- **y_tick_format** (*str*) – Format for y axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format

- **guides_type** (*str*) – The type of value-guides to show when hovering over a point in the graph. Valid values are "XY", "YY" or False. XY guides will show the x and y values for the point under the cursor. YY guides will show the x and y values of each component for the data point closest to the cursor. False will turn off value guides.

datasource

The data source providing data for this component

Returns data source providing data for this component

delete()

Delete the Plot and any associated data which has been uploaded.

guides_type

The type of value-guides to show when hovering over a point in the graph. Valid values are "XY", "YY" or False. XY guides will show the x and y values for the point under the cursor. YY guides will show the x and y values of each component for the data point closest to the cursor. False will turn off value guides.

Return type str

height

The height of the plot. Represented as a string containing a valid CSS "width" attribute; for example, '300px'.

Return type str

plot

Add new data to the plot with the specified options.

If a datasource is specified, x and y can be provided as expressions in terms of the variables in that datasource. Otherwise, x and y must be lists of data of identical length.

Parameters

- **x** (*list*) – X axis data. Either a list of input values, or if a datasource is specified, an expression in terms of that datasource's variables.
- **y** (*list*) – Y axis data. Either a list of input values, or if a datasource is specified, an expression in terms of that datasource's variables.
- **datasource** (DataSource) – If provided, a specific DataSource to use as a base for the variables to be plotted.
- **style** (*str*) – A matplotlib-like style string of 'o' or '-' or '-o' to indicate circle, line, or line-circle, respectively.
- **color** (*str*) – CSS color.
- **line_width** (*int*) – The width of the line, if applicable.
- **circle_radius** (*int*) – The radius of each circle, if applicable.
- **use_in_plot_range** (*bool*) – The chart auto-computes the x and y axis ranges based on the data for each component. If you want to add a component, but have its data not be used to compute the x and y axis ranges, set this value to False. For example, if you want to make a scatter plot, with a trend line going through it, then setting this field to False for the trend-line component will make the chart snugly fit the points, with the trend line extending beyond.
- **error_bars_upper_values** (*list*) – Specifies the tops of error bars. Either a list of values, or if a datasource is specified, an expression in terms of that datasource's variables. If input values is a list, then this must be a list as well.

- **error_bars_lower_values** (*list*) – Specifies the bottoms of error bars. Either a list of values, or if a datasource is specified, an expression in terms of that datasource's variables. If input values is a list, then this must be a list as well.
- **legend_label** (*str*) – A string label to be used in the legend. If unspecified, this plot will not appear in the legend.
- **tooltip_template** (*str*) – A basic template string that can be used to provide custom mouseover tooltips to plot points. It is passed the current point's x/y values as `{{x_value}}` and `{{y_value}}`. It is also passed the current tooltip as `{{tooltip}}` if a `tooltips` parameter is provided.
- **tooltips** (*list*) – A list of per-point tooltip values. If specified, adds a mouseover tooltip to each point on the plot. If a `tooltip_template` has been provided, tooltips will be passed into the template context as `{{tooltip}}`.

Tooltip examples:

- `tooltips` parameter:

```
tooltips=["A", "B", "C"]

# Point one will have a tooltip of "A"
# Point two will have a tooltip of "B"
# Point three will have a tooltip of "C"
```

- `tooltip_template` parameter:

```
tooltip_template="Point: {{x_value}}, {{y_value}}"
```

Let's say we're plotting two points: `[{x: 0, y: 0}, {x: 1, y: 1}]`. With the template above, their tooltips will render as:

```
"Point: 0, 0" and "Point: 1, 1"
```

- `tooltips` and `tooltip_template` parameters:

Given points `[{x: 0, y: 0}, {x: 1, y: 1}]`:

```
tooltip_template="Item {{tooltip}} has a value of {{x_value}}, {{y_value}}",
tooltips=["A", "B"]
```

will generate the following tooltips:

```
"Item A has a value of 0, 0"
"Item B has a value of 1, 1"
```

show_legend
Whether or not the legend should be shown

Return type bool

upload_data(*eureqa=None*)
Upload the plot data to eureqa. This is required before the plot can be viewed.

Parameters **eureqa** (`Eureqa`) – a eureqa connection

width
The width of the plot. Represented as a string containing a valid CSS "width" attribute; for example, '400px'.

Return type str

`x_axis_label`

The label of the plot's X axis

Return type str

`x_tick_format`

Format for x axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format

Returns str

`y_axis_label`

The label of the plot's Y axis

Return type str

`y_tick_format`

Format for y axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format

Returns str

`zero_x_line`

Whether an axis line should be shown for x=0

Return type bool

`zero_y_line`

Whether an axis line should be shown for y=0

Return type bool

class `eureqa.analysis.components.`**`ModelSummary`**(*solution=None*)

A model summary card. See also *`Analysis.create_model_summary_card`*

For example:

```
p = ModelSummary(solution=s.get_best_solution())
analysis.create_card(p)
```

Parameters **`solution`** (`Solution`) – The solution that will be displayed on the card.

`solution`

The Solution that is being explained

Return type *Solution*

class `eureqa.analysis.components.`**`TextBlock`**(*text=’‘*, *description=None*)

Contains free-form user-specified text, formatted using markdown. *Deprecated*. Use *`HtmlBlock`* instead.

Parameters

- **`text`** (*`str`*) – Body of the card.
- **`description`** (*`str`*) – alias for text (backwards compatible)

Variables ***`eureqa.analysis.components.TextBlock.text`*** (*`str`*) – Body of the card.

`text`

Markdown-formatted text contents of this component

Return type str

class `eureqa.analysis.components.`**`HtmlBlock`**(*html=’‘*)

Contains free-form user-specified HTML, including references to other components.

Example:

```
h = HtmlBlock('This will render on a Card in an Analysis')
analysis.create_card(h)
```

Example of using *`HtmlBlock`* with a *`VariableLink`*:

```
vl = VariableLink(d, 'X')
h = HtmlBlock('Formatted <b>Card</b> in the Analysis, with reference to {0}'.format(analysis.htm
analysis.create_card(h)
```

Parameters **`html`** (*`str`*) – Body of the card.

Variables ***`eureqa.analysis.components.html_block.HtmlBlock.html`*** (*`str`*) – Body of the card.

If you need particular styles or JS libraries, you MUST define and/or load them as part of the HtmlBlock where they are needed. Any styles or JS objects defined by Eureqa may change or be removed from release to release.

Embedding custom JavaScript (or CSS, which can contain JavaScript) from a malicious third-party source can grant that source access to your site. Only embed content from trusted sources.

`html`

The body of this card.

Returns body of this card

Return type str

class `eureqa.analysis.components.`**`TitledLayout`**(*title=None*, *description=None*, *content=None*)

A layout with space for a title and description and content within an Analysis

For example:

```
h=HtmlBlock(html="<h1>Hybrid Performance</h1>")
layout=TitledLayout(title="the title", description="You can add content specific description her
analysis.create_card(layout)
```

Parameters

- **`title`** (*`str`*) – The title of the layout
- **`description`** (*`str`*) – The description of the layout (can contain HTML)
- **`content`** (*`_Component`*) – The component to use in the main content of the layout

`content`

The Component that this TitledLayout is adding a title to. This field can’t be assigned to; use ‘create_card()’ to assign a Component to this TitledLayout.

Return type _Component

`create_card`(*component*)

Assign a Component to this TitledLayout

Parameters **`component`** (*`_Component`*) – Component to use as the content of this layout

`description`

The description of this card.

Returns description of this card

Return type str

title

The title of this card.

Returns title of this card

Return type str

class eureqa.analysis.components.**MagnitudeBar**(*value=None*, *color=None*)

Magnitude Bar: implements a visual representation of percentages

For Example:

```
pink_bar = MagnitudeBar(value=-0.22, color='#ff00ff')
analysis.create_html_card("Here is the magnitude bar: {0}".format(analysis.html_ref(pink_bar)))
```

Parameters

- **value** (*float*) – value expressed by this MagnitudeBar
- **color** (*str*) – (optional) html hex color code to use for the magnitude bar instead of default

color

The Color of this bar, expressed as an html hex color code.

For example: ‘#ff5733’ - red

Return type str

value

The value expressed by this MagnitudeBar. Must be a fractional number between 0 and 1 (positive bar) or 0 and -1 (negative bar).

Return type float

class eureqa.analysis.components.**VariableLink**(*datasource=None*, *variable_name=None*)

Create a VariableLink component

Example of using HtmlBlock with a VariableLink:

```
vl = VariableLink(d, 'X')
h = HtmlBlock('Formatted <b>Card</b> in the Analysis, with reference to {0}'.format(analysis.htm
analysis.create_card(h)
```

Parameters

- **datasource** (DataSource) – datasource from which to find the variable
- **variable_name** (*str*) – name of the variable

datasource

The data source providing data for this component

Return type *DataSource*

variable_name

The name of the variable from this datasource to link to

Return type str

class `eureqa.analysis.components.`**`SearchLink`**(*search=None*, *link_text=None*)

Create a link to a specified search

Example of using HtmlBlock with a SearchLink:

```
vl = SearchLink(search)
h = HtmlBlock('Formatted <b>Card</b> in the Analysis, with reference to search {0}'.format(analy
analysis.create_card(h)
```

Parameters

- **`search`** (`Search`) – the search to link to
- **`link_text`** (*`str`*) – (optional) the text to use for the search link is displayed

`link_text`

The text to use for the search link when displayed.

Returns the link text of this card

Return type str

`search`

The search linked to

Return type *Search*

class `eureqa.analysis.components.`**`SearchBuilderLink`**(*link_text=None*, *datasource=None*, *search_template=None*, *min_delay=None*, *max_delay=None*, *target_variable=None*, *input_variables=None*)

Create a SearchBuilderLink component

Example of using HtmlBlock with a SearchBuilderLink:

```
# include a link to the search builder with nothing pre-populated
search_builder_link = SearchBuilderLink()
h = HtmlBlock('Click this link to open the pre-populated search builder: {0}'.format(analysis.ht
layout.add_component(h)

# include a link to the search builder with settings pre-populated by a search template
datasource = e.create_data_source("data_source_1", "tests/Nutrition.csv")
search_builder_link = SearchBuilderLink(datasource=datasource,
                                        search_template=search_templates.SearchTemplates.Timeser
                                        min_delay=1,
                                        max_delay=10,
                                        target_variable='Calories',
                                        input_variables=['Steps', 'Weight', '(Protein (g))', '(F
h = HtmlBlock('Click this link to open the pre-populated search builder: {0}'.format(analysis.ht
layout.add_component(h)
```

Parameters

- **`link_text`** (*`str`*) – the visible, clickable text for the link
- **`datasource`** (`DataSource`) – datasource to pre-populate the search builder with
- **`search_template`** (*`SearchTemplate`*) – search template information to pre-populate the search builder with
- **`min_delay`** (*`int`*) – the minimum delay used in a timeseries search

- **max_delay** (*int*) – the maximum delay used in a timeseries search
- **target_variable** (*str*) – the variable to search for models of
- **input_variables** (*str[]*) – the variables to use as inputs in the models

link_text

The HTML text which will be wrapped with a link to the search builder.

Return type str

class eureqa.analysis.components.**DownloadFile**(*file_content=None*, *link_text=None*, *filename=None*)

DownloadFile component Represents a Download link for a file within an Analysis, including the contents of that file.

For example:

```
df = DownloadFile(file_content="The file content", link_text="The link Text", filename="download
analysis.create_html_card('Download file here: {0}'.format(analysis.html_ref(df)))
```

Parameters

- **file_content** (*str*) – The contents of the file as a str() or eureqa.analysis.analysis_file.AnalysisFile
- **link_text** (*str*) – The text of the HTML download link in the Analysis
- **filename** (*str*) – The name of the downloaded file

file

Return type *eureqa.analysis.analysis_file.AnalysisFile*

link_text

Text of the link as rendered by the component

class eureqa.analysis.components.**TabbedLayout**(*tab_type='default'*)

TabbedLayout component

Lets users pick to see other components based on titles (works with a few items). Similar to *DropdownLayout* but suited to smaller lists

For example:

```
t = TabbedLayout()
t.add_component(title="choice 1", component=HtmlBlock("This is item 1"))
t.add_component(title="choice 2", component=HtmlBlock("This is item 2"))
analysis.create_card(t)
```

add_component(*title*, *component*, *icon=None*)

Add a new tab containing a new component

Parameters

- **title** (*str*) – Title of the tab
- **component** (*_Component*) – Tab contents, as a Component
- **icon** (*AnalysisFile*) – Tab icon (optional) – must be a file containing an image

class eureqa.analysis.components.**DropdownLayout**(*label=None*, *padding=None*)

DropdownLayout component Lets users pick to see other components based on titles (works with “many” items). Similar to a *TabbedLayout* but more suited to larger lists of items.

For example:

```
d = DropdownLayout(label="Choose an item", padding="1.0em")
d.add_component(title="choice 1", content=HtmlBlock("This is item 1"))
d.add_component(title="choice 2", content=HtmlBlock("This is item 2"))
analysis.create_card(d)
```

Parameters

- **label** (*str*) – The overall label
- **padding** (*str*) – padding, specified in ems. Example "0.5em"

add_component (*title*, *component=None*, *content=None*)

Add a Component to the DropdownLayout

Parameters

- **title** (*str*) – Title of the Component in the dropdown list
- **component** (*_Component*) – The Component to add to the dropdown list
- **content** (*_Component*) – Deprecated, Do not use

components

Internal: Set of components represented in the dropdown

Returns list() of `_Component`

label

Text label for the Dropdown box

Return type str

padding

Padding (spacing) for this DropdownLayout, expressed in ems. Example "0.5em"

Return type int

class `eureqa.analysis.components.`**Layout** (*rows=None*, *borders=False*)

Generic layout within an Analysis

A generic grid layout composed of rows and columns sized by fractions (*1*, *1/2*, *1/4*, etc). Layouts can be nested within other layouts to create any type of grid needed.

For example:

```
layout = Layout()
layout = Layout(borders=True)                                       # create a layout where bor
layout.add_component(HtmlBlock(html="<h1>Hybrid Performance</h1>")) # add an item that is an ent
layout.add_row()                                                    # start a new row. newly ad
layout.add_component(MagnitudeBar(value=-0.22), "1/3")              # add a magnutide bar on the
layout.add_component(MagnitudeBar(value=-0.22), "1/3", True)        # add a magnutide bar with c
layout.add_component(HtmlBlock(html="Variables...."), "1/2")        # add a text panel in the ne
analysis.create_card(layout)                                        # Add this Layout to an Anal
```

Parameters

- **borders** (*bool*) – If true, render visible borders on the layout component
- **rows** (*list*) – (optional) List of `Row` objects to add to this class initially

class `eureqa.analysis.components.`**`Modal`**(*title=None*, *size=None*, *icon_file_path=None*, *component=None*)

Represents a popup window (modal) which can contain other components.

Example: # create a modal which is activated if a link is clicked text_block = TextBlock("This text is inside the modal") modal = Modal(title="This is the modal title", size='medium', icon_file_path='path/to/icon', component=text_block) h = HtmlBlock('Click this link to open a modal: {0}'.format(analysis.html_ref(modal.link('link text')))) layout.add_component(h)

Parameters

- **`title`** (*`str`*) – The title of the modal
- **`size`** (*`str`*) – (optional) modal size. Options are 'small', 'medium' or 'large'
- **`icon_file_path`** (*`str`*) – (optional) path to an icon file to display in the corner of the modal
- **`component`** (*`Component`*) – The component to display in the modal

`content_component_id`

The id of the component to be displayed in the modal

Return type str

`icon_file_id`

The id of an icon file to be displayed in the corner of the modal

Return type str

`icon_file_url`

The url of an icon file to be displayed in the corner of the modal

Return type str

`link`(*link_text=None*)

Returns a ModalLink component which represents a link to this modal

Parameters **`link_text`** (*`str`*) – (optional) link text to use in the ModalLink instead of default text

Return type *eureqa.analysis.components.modal_link.ModalLink*

`size`

The size of the modal

Return type str

`title`

The title of the modal

Return type str

class `eureqa.analysis.components.`**`ModalLink`**(*modal=None*, *link_text=None*)

Represents a link which opens a popup window (modal) which can contain other components.

The use of this class is typically hidden by the Modal classes link method.

Example: # create a modal which is activated if a link is clicked text_block = TextBlock("This text is inside the modal") modal = Modal(title="This is the modal title", size='medium', icon_file_path='path/to/icon', component=text_block) h = HtmlBlock('Click this link to open a modal: {0}'.format(analysis.html_ref(modal.link('link text')))) layout.add_component(h)

# create another link to the above modal h = HtmlBlock('OR click this other link to open the same modal: {0}'.format(analysis.html_ref(modal.link('different link text')))) layout.add_component(h)

Parameters

- **modal** (eureqa.analysis.components.modal.Modal) – the Modal which this component will link to
- **link_text** (*str*) – The HTML which the link to open the modal will be wrapped around

link_text
Text of the link as rendered by the component

modal_component_id
The ID of the modal_component to be linked to

class eureqa.analysis.components.**Tooltip**(*html=None*, *tooltip=None*)
Tooltip Component: implements a tooltip

For example:

```
tt = Tooltip(html="This", tooltip="Text to show when hovering")
card = analysis.create_html_card("This is a component with a tooltip: {0}".format(analysis.html_
```

Parameters

- **html** (*str*) – Text to display normally
- **tooltip** (*str*) – The tooltip text to show when the cursor hovers over the component

html
The text to show normally for this component.

Return type str

tooltip
The tooltip text to show when the cursor hovers over the component

Return type str

class eureqa.analysis.components.**TableBuilder**(*data*, *title*, *default_rows_per_page=20*, *column_names=None*, *striped=True*, *search_box_place_holder='Search'*, *page_controls_and_search_visible=True*, *default_sort_order=None*)
High level API to build a table in analysis

Parameters

- **data** (*object*) – The data to store in the table. Accepts *pandas.DataFrame*, *numpy.ndarray*, a list of floats, or a list of strings
- **title** (*str*) – table title
- **default_rows_per_page** (*int*) – the default number of rows per page when the table is initially rendered
- **column_names** (*list*) – names of each column specified as a list of strings
- **striped** (*bool*) – whether or not the table rows should be rendered in alternating grey and white stripes
- **search_box_place_holder** (*str*) – the default text shown in the search box when user hasn't typed any search keyword
- **page_controls_and_search_visible** (*bool*) – if true show a table header with next/previous page and search controls enabled

- **`default_sort_order`** (*`str/tuple`*) – If specified, name of the column to sort the table by default. Sort order can be specified by using a tuple with the first element being the name of the column and the second element being either 'ASC' or 'DESC' to specify ascending or descending order.

class `eureqa.analysis.components.`**`FormattedText`**(*format_str*)

Creates a formatted string, substituing format specifications with component references.

For example:

```
FormattedText("Tooltip {0}", Tooltip(html='Here', tooltip='this is the tooltip'))
```

Parameters **`format_str`** (*`str`*) – The string to format

class `eureqa.analysis.components.`**`Image`**(*image_path=None*, *width=None*, *height=None*)

Image component Represents an image which can be used in an Analysis.

For example:

```
i = Image('camel.png')
analysis.create_html_card('Image here: {0}'.format(analysis.html_ref(i)))
```

Parameters

- **`image_path`** (*`str`*) – path to the image file
- **`width`** (*`int`*) – (optional) a specific width to display the image with
- **`height`** (*`int`*) – (optional) a specific height to display the image with

`file`

Return type *eureqa.analysis.analysis_file.AnalysisFile*

class `eureqa.analysis.components.`**`MostFrequentVariablesPlot`**(*search=None*)

A most frequent variables plot card. See also *`Analysis.create_most_frequent_variables_plot_card`*

For example:

```
p = MostFrequentVariablesPlot(search=s)
analysis.create_card(p)
```

Parameters **`search`** (`Search`) – The search whose variable frequencies will be displayed on the card.

`search`

The Search the variables belong to

Return type *Search*

class `eureqa.analysis.components.`**`ThresholdSelectionPlot`**(*solution=None*)

A threshold selection plot card. See also *`Analysis.create_threshold_selection_plot_card`*

For example:

```
p = ThresholdSelectionPlot(solution=s)
analysis.create_card(p)
```

Parameters **`solution`** (`Solution`) – The solution that will be displayed in the plot

solution
The Solution to be displayed in the plot

Return type *Solution*

class eureqa.analysis.components.**ModelTermsPlot**(*solution=None*)
A model terms plot card. See also *Analysis.create_model_terms_plot_card*

For example:

```
p = ModelTermsPlot(solution=s)
analysis.create_card(p)
```

Parameters **solution** (Solution) – The solution that will be displayed in the plot

solution
The Solution to be displayed in the plot

Return type *Solution*

base

binned_mean_plot

class eureqa.analysis.components.binned_mean_plot.**BinnedMeanPlot**(*datasource=None*, *axis_labels=None*, *label_format=None*, *needs_guides=None*, *x_var=None*, *y_var=None*)
Represents a binned mean plot component on the server.

For example:

```
p = BinnedMeanPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':'.3s'
analysis.create_card(p)
```

Parameters

- **datasource** (DataSource) – The data source containing the data to be plotted.
- **x_var** (*str*) – The X-axis variable for the card's plot.
- **y_var** (*str*) – The Y-axis variable for the card's plot.
- **needs_guides** (*bool*) – Whether the card needs guides.
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set keys "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

Variables

- **title** (*str*) – The title of the card
- **x_var** (*str*) – The X-axis variable for the card's plot
- **y_var** (*str*) – The Y-axis variable for the card's plot

- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set member fields "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

axis_labels

The axis labels for this card

defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type XYMap

datasource

The data source providing data for this component

Returns data source providing data for this card

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

defaults to:

```
{ 'x': '.3s', 'y': '.2f' }
```

Return type XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card.

Returns X variable for this card

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

box_plot

class `eureqa.analysis.components.box_plot.`**`BoxPlot`**(*datasource=None*, *axis_labels=None*, *label_format=None*, *needs_guides=None*, *x_var=None*, *y_var=None*)

Represents a box plot card on the server.

Example:

```
bp = BoxPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':'.3s'}, x_v
analysis.create_card(bp)
```

Parameters

- **`datasource`** (`DataSource`) – The data source containing the data to be plotted.
- **`axis_labels`** (*`XYMap`*) – Axis labels for this card's plot (XYMap). Set member fields "x" and "y" to set the X and Y axis labels.
- **`label_format`** (*`XYMap`*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".
- **`x_var`** (*`str`*) – The X-axis variable for the component's plot. (must be binary)
- **`y_var`** (*`str`*) – The Y-axis variable for the component's plot.
- **`needs_guides`** (*`bool`*) – Whether the card needs guides.

Variables

- **`title`** (*`str`*) – The title of the card
- **`x_var`** (*`str`*) – The X-axis variable for the card's plot (must be binary)
- **`y_var`** (*`str`*) – The Y-axis variable for the card's plot
- **`needs_guides`** (*`bool`*) – Whether the card needs guides
- **`axis_labels`** (*`XYMap`*) – Axis labels for this card's plot. Set member fields "x" and "y" to set the X and Y axis labels.
- **`label_format`** (*`XYMap`*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

`axis_labels`

The axis labels for this card

defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type self.XYMap

`datasource`

The *`DataSource`* providing data for this component

Returns *`DataSource`* providing data for this card

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

defaults to:

```
{'x': 'g', 'y': '.2s'}
```

Return type XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card.

Returns X variable for this card. (must be binary)

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

by_row_plot

class eureqa.analysis.components.by_row_plot.**ByRowPlot**(*datasource=None*, *x_var=None*, *plotted_variables=None*, *focus_variable=None*, *should_center=None*, *should_scale=None*)

Represents a line graph for plotting variables by row.

For example:

```
p = ByRowPlot(d, x_var='W', focus_variable='W', plotted_variables=['A'], should_center=True)
analysis.create_card(p)
```

Parameters

- **datasource** (*eureqa.DataSource*) – Data source for the component's data
- **x_var** (*str*) – Name of the variable to plot as the X axis
- **plotted_variables** (*list*) – List of variable names to plot.
- **focus_variable** (*str*) – Name of the variable in *plotted_variables* to bring to the foreground (required)
- **should_center** (*bool*) – Should the plot be centered?
- **should_scale** (*bool*) – Should the plot scale?

Variables

- **eureqa.analysis.components.variable_line_graph.ByRowPlot.focus_variable** (*str*) – Focused (foreground) variable for the component. Must be a member of *plotted_variables*
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.x_var** (*str*) – Name of the variable to plot as the X axis
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.plotted_variables** (*list*) – Variables to plot. (List of string variable names.)
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.should_center** (*bool*) – Should the plot be centered?
- **eureqa.analysis.components.variable_line_graph.ByRowPlot.should_scale** (*bool*) – Should the plot scale?

datasource

The data source providing data for this component

Returns data source providing data for this component

focus_variable

The variable that is currently in focus (in the foreground) for this component. Must be a member of *plotted_variables*.

Returns focus_variable for this component

Return type str

plotted_variables

The plotted variables for this component.

Returns List of the names of the variables being plotted against the X axis

Return type list

should_center

The should_center option for this component.

Returns whether this plot should be centered

Return type bool

should_scale

The should_scale option for this component.

Returns whether this plot should be scaled

Return type bool

x_var

The X-axis variable for this component

Returns the name of the X-axis variable for this component

Return type str

custom_plot

class `eureqa.analysis.components.custom_plot.`**`CustomPlot`**(*datasource=None*, *width=None*, *height='400px'*, *x_axis_label=None*, *y_axis_label=None*, *show_legend=True*, *zero_x_line=False*, *zero_y_line=False*, *x_tick_format=None*, *y_tick_format=None*, *guides_type='XY'*)

Represents a plot to be displayed in an analysis plot card.

By calling the plot() method multiple times on one instance of this class, multiple lines and scatter plots can be superimposed on this object.

Parameters

- **`datasource`** (`DataSource`) – The data source containing the data to be plotted.
- **`width`** (*`str`*) – Set manual dimensions for the plot in "px" units (e.g. 350px). Defaults to full panel width.
- **`height`** (*`str`*) – Set manual dimensions for the plot in "px" units. Defaults to a constant height of 400px.
- **`x_axis_label`** (*`str`*) – The label to use on the x axis of the plot.
- **`y_axis_label`** (*`str`*) – The label to use on the y axis of the plot.
- **`show_legend`** (*`bool`*) – Controls whether or not a legend will be used in the plot.
- **`zero_x_line`** (*`bool`*) – Draws a horizontal line through the origin
- **`zero_y_line`** (*`bool`*) – Draws a vertical line through the origin
- **`x_tick_format`** (*`str`*) – Format for x axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format
- **`y_tick_format`** (*`str`*) – Format for y axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format
- **`guides_type`** (*`str`*) – The type of value-guides to show when hovering over a point in the graph. Valid values are "XY", "YY" or False. XY guides will show the x and y values for the point under the cursor. YY guides will show the x and y values of each component for the data point closest to the cursor. False will turn off value guides.

`datasource`

The data source providing data for this component

Returns data source providing data for this component

`delete`()

Delete the Plot and any associated data which has been uploaded.

`guides_type`

The type of value-guides to show when hovering over a point in the graph. Valid values are "XY", "YY" or False. XY guides will show the x and y values for the point under the cursor. YY guides will show the x and y values of each component for the data point closest to the cursor. False will turn off value guides.

Return type str

height

The height of the plot. Represented as a string containing a valid CSS "width" attribute; for example, '300px'.

Return type str

plot

Add new data to the plot with the specified options.

If a datasource is specified, x and y can be provided as expressions in terms of the variables in that datasource. Otherwise, x and y must be lists of data of identical length.

Parameters

- **x** (*list*) – X axis data. Either a list of input values, or if a datasource is specified, an expression in terms of that datasource's variables.
- **y** (*list*) – Y axis data. Either a list of input values, or if a datasource is specified, an expression in terms of that datasource's variables.
- **datasource** (DataSource) – If provided, a specific DataSource to use as a base for the variables to be plotted.
- **style** (*str*) – A matplotlib-like style string of 'o' or '-' or '-o' to indicate circle, line, or line-circle, respectively.
- **color** (*str*) – CSS color.
- **line_width** (*int*) – The width of the line, if applicable.
- **circle_radius** (*int*) – The radius of each circle, if applicable.
- **use_in_plot_range** (*bool*) – The chart auto-computes the x and y axis ranges based on the data for each component. If you want to add a component, but have its data not be used to compute the x and y axis ranges, set this value to False. For example, if you want to make a scatter plot, with a trend line going through it, then setting this field to False for the trend-line component will make the chart snugly fit the points, with the trend line extending beyond.
- **error_bars_upper_values** (*list*) – Specifies the tops of error bars. Either a list of values, or if a datasource is specified, an expression in terms of that datasource's variables. If input values is a list, then this must be a list as well.
- **error_bars_lower_values** (*list*) – Specifies the bottoms of error bars. Either a list of values, or if a datasource is specified, an expression in terms of that datasource's variables. If input values is a list, then this must be a list as well.
- **legend_label** (*str*) – A string label to be used in the legend. If unspecified, this plot will not appear in the legend.
- **tooltip_template** (*str*) – A basic template string that can be used to provide custom mouseover tooltips to plot points. It is passed the current point's x/y values as `{{x_value}}` and `{{y_value}}`. It is also passed the current tooltip as `{{tooltip}}` if a `tooltips` parameter is provided.
- **tooltips** (*list*) – A list of per-point tooltip values. If specified, adds a mouseover tooltip to each point on the plot. If a `tooltip_template` has been provided, tooltips will be passed into the template context as `{{tooltip}}`.

Tooltip examples:

- `tooltips` parameter:

```
tooltips=["A", "B", "C"]

# Point one will have a tooltip of "A"
# Point two will have a tooltip of "B"
# Point three will have a tooltip of "C"
```

- `tooltip_template` parameter:

```
tooltip_template="Point: {{x_value}}, {{y_value}}"
```

Let's say we're plotting two points: `[{x: 0, y: 0}, {x: 1, y: 1}]`. With the template above, their tooltips will render as:

```
"Point: 0, 0" and "Point: 1, 1"
```

- `tooltips` and `tooltip_template` parameters:

Given points `[{x: 0, y: 0}, {x: 1, y: 1}]`:

```
tooltip_template="Item {{tooltip}} has a value of {{x_value}}, {{y_value}}",
tooltips=["A", "B"]
```

will generate the following tooltips:

```
"Item A has a value of 0, 0"
"Item B has a value of 1, 1"
```

`show_legend`

Whether or not the legend should be shown

Return type bool

`upload_data`(*eureqa=None*)

Upload the plot data to eureqa. This is required before the plot can be viewed.

Parameters **eureqa** (`Eureqa`) – a eureqa connection

`width`

The width of the plot. Represented as a string containing a valid CSS "width" attribute; for example, '400px'.

Return type str

`x_axis_label`

The label of the plot's X axis

Return type str

`x_tick_format`

Format for x axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format

Returns str

`y_axis_label`

The label of the plot's Y axis

Return type str

y_tick_format
Format for y axis tick labels. Valid values are "date" or anything supported by D3: https://github.com/mbostock/d3/wiki/Formatting#d3_format. Defaults to our internal number format

Returns str

zero_x_line
Whether an axis line should be shown for x=0

Return type bool

zero_y_line
Whether an axis line should be shown for y=0

Return type bool

distribution_plot

class `eureqa.analysis.components.distribution_plot.`**`DistributionPlot`**(*datasource=None, variable_name=None*)

Represents a distribution plot card on the server.

Parameters

- **datasource** (*eureqa.DataSource*) – The data source to which the variable belongs.
- **variable_name** (*str*) – The name of the variable that will be displayed on the card.

Variables

- **title** (*str*) – The card title.
- **datasource** (*str*) – The datasource used by the card.
- **variable** (*str*) – The variable plotted by the coard.

datasource
The data source providing data for this component

Returns data source providing data for this component

variable
(Deprecated) Name of the variable to plot. For backwards compatibility.

Return type str

variable_name
Name of the variable to plot

Return type str

double_histogram_plot

class `eureqa.analysis.components.double_histogram_plot.`**`DoubleHistogramPlot`**(*datasource=None, axis_labels=None, label_format=None, needs_guides=None, x_var=None, y_var=None*)

Represents a double-histogram plot component on the server.

For example:

```
p = DoubleHistogramPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':
analysis.create_card(p)
```

Parameters

- **datasource** (DataSource) – The data source containing the data to be plotted.
- **x_var** (*str*) – The X-axis variable for the card's plot. (must be binary)
- **y_var** (*str*) – The Y-axis variable for the card's plot.
- **needs_guides** (*bool*) – Whether the card needs guides.
- **axis_labels** (*dict*) – Axis labels for this card's plot. Set keys "x" and "y" to set the X and Y axis labels.
- **label_format** (*dict*) – Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

Variables

- **title** (*str*) – The title of the card
- **x_var** (*str*) – The X-axis variable for the card's plot (must be binary)
- **y_var** (*str*) – The Y-axis variable for the card's plot
- **needs_guides** (*bool*) – Whether the card needs guides
- **axis_labels** (*XYMap*) – Axis labels for this card's plot. Set member fields "x" and "y" to set the X and Y axis labels.
- **label_format** (*XYMap*) – Label format for this card. Set member fields "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

axis_labels

The axis labels for this card

defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type XYMap

datasource

The data source providing data for this component

Returns data source providing data for this card

label_format

Label format for this card. Set keys "x" and "y" to set the X and Y axis printf-style format-strings; for example, ".3s".

defaults to:

```
{ 'x': '.3s', 'y': 'g' }
```

Return type XYMap

needs_guides
Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var
The X variable for this card. (must be binary)

Returns X variable for this card

Return type str

y_var
The Y variable for this card.

Returns Y variable for this card

Return type str

download_file

class eureqa.analysis.components.download_file.**DownloadFile**(*file_content=None*, *link_text=None*, *filename=None*)

DownloadFile component Represents a Download link for a file within an Analysis, including the contents of that file.

For example:

```
df = DownloadFile(file_content="The file content", link_text="The link Text", filename="download
analysis.create_html_card('Download file here: {0}'.format(analysis.html_ref(df)))
```

Parameters

- **file_content** (*str*) – The contents of the file as a str() or eureqa.analysis.analysis_file.AnalysisFile
- **link_text** (*str*) – The text of the HTML download link in the Analysis
- **filename** (*str*) – The name of the downloaded file

file

Return type *eureqa.analysis.analysis_file.AnalysisFile*

link_text
Text of the link as rendered by the component

dropdown_layout

class eureqa.analysis.components.dropdown_layout.**DropdownLayout**(*label=None*, *padding=None*)

DropdownLayout component Lets users pick to see other components based on titles (works with "many" items). Similar to a *TabbedLayout* but more suited to larger lists of items.

For example:

```
d = DropdownLayout(label="Choose an item", padding="1.0em")
d.add_component(title="choice 1", content=HtmlBlock("This is item 1"))
d.add_component(title="choice 2", content=HtmlBlock("This is item 2"))
analysis.create_card(d)
```

Parameters

- **`label`** (*`str`*) – The overall label
- **`padding`** (*`str`*) – padding, specified in ems. Example “0.5em”

`add_component`(*title*, *component=None*, *content=None*)

Add a Component to the DropdownLayout

Parameters

- **`title`** (*`str`*) – Title of the Component in the dropdown list
- **`component`** (*`_Component`*) – The Component to add to the dropdown list
- **`content`** (*`_Component`*) – Deprecated, Do not use

`components`

Internal: Set of components represented in the dropdown

Returns list() of `_Component`

`label`

Text label for the Dropdown box

Return type str

`padding`

Padding (spacing) for this DropdownLayout, expressed in ems. Example “0.5em”

Return type int

formatted_text

class `eureqa.analysis.components.formatted_text.`**`FormattedText`**(*format_str*)

Creates a formatted string, substituting format specifications with component references.

For example:

```
FormattedText("Tooltip {0}", Tooltip(html='Here', tooltip='this is the tooltip'))
```

Parameters **`format_str`** (*`str`*) – The string to format

html_block

class `eureqa.analysis.components.html_block.`**`HtmlBlock`**(*html=’‘*)

Contains free-form user-specified HTML, including references to other components.

Example:

```
h = HtmlBlock('This will render on a Card in an Analysis')
analysis.create_card(h)
```

Example of using *`HtmlBlock`* with a *`VariableLink`*:

```
vl = VariableLink(d, 'X')
h = HtmlBlock('Formatted <b>Card</b> in the Analysis, with reference to {0}'.format(analysis.htm
analysis.create_card(h)
```

Parameters **`html`** (*str*) – Body of the card.

Variables ***`eureqa.analysis.components.html_block.HtmlBlock.html`*** (*str*) – Body of the card.

If you need particular styles or JS libraries, you MUST define and/or load them as part of the HtmlBlock where they are needed. Any styles or JS objects defined by Eureqa may change or be removed from release to release.

Embedding custom JavaScript (or CSS, which can contain JavaScript) from a malicious third-party source can grant that source access to your site. Only embed content from trusted sources.

`html`

The body of this card.

Returns body of this card

Return type str

image

class `eureqa.analysis.components.image.`**`Image`**(*image_path=None*, *width=None*, *height=None*)

Image component Represents an image which can be used in an Analysis.

For example:

```
i = Image('camel.png')
analysis.create_html_card('Image here: {0}'.format(analysis.html_ref(i)))
```

Parameters

- **`image_path`** (*str*) – path to the image file
- **`width`** (*int*) – (optional) a specific width to display the image with
- **`height`** (*int*) – (optional) a specific height to display the image with

`file`

Return type *eureqa.analysis.analysis_file.AnalysisFile*

layout

class `eureqa.analysis.components.layout.`**`Layout`**(*rows=None*, *borders=False*)

Generic layout within an Analysis

A generic grid layout composed of rows and columns sized by fractions (*1*, *1/2*, *1/4*, etc). Layouts can be nested within other layouts to create any type of grid needed.

For example:

```
layout = Layout()
layout = Layout(borders=True)                                      # create a layout where borc
layout.add_component(HtmlBlock(html="<h1>Hybrid Performance</h1>")) # add an item that is an ent
layout.add_row()                                                    # start a new row. newly adc
layout.add_component(MagnitudeBar(value=-0.22), "1/3")              # add a magnutide bar on the
layout.add_component(MagnitudeBar(value=-0.22), "1/3", True)        # add a magnutide bar with c
layout.add_component(HtmlBlock(html="Variables...."), "1/2")        # add a text panel in the ne
analysis.create_card(layout)                                        # Add this Layout to an Anal
```

Parameters

- **`borders`** (*`bool`*) – If true, render visible borders on the layout component
- **`rows`** (*`list`*) – (optional) List of `Row` objects to add to this class initially

magnitude_bar

class `eureqa.analysis.components.magnitude_bar.`**`MagnitudeBar`**(*value=None*, *color=None*)

Magnitude Bar: implements a visual representation of percentages

For Example:

```
pink_bar = MagnitudeBar(value=-0.22, color='#ff00ff')
analysis.create_html_card("Here is the magnitude bar: {0}".format(analysis.html_ref(pink_bar)))
```

Parameters

- **`value`** (*`float`*) – value expressed by this MagnitudeBar
- **`color`** (*`str`*) – (optional) html hex color code to use for the magnitude bar instead of default

`color`

The Color of this bar, expressed as an html hex color code.

For example: ‘#ff5733’ - red

Return type str

`value`

The value expressed by this MagnitudeBar. Must be a fractional number between 0 and 1 (positive bar) or 0 and -1 (negative bar).

Return type float

modal

class `eureqa.analysis.components.modal.`**`Modal`**(*title=None*, *size=None*, *icon_file_path=None*, *component=None*)

Represents a popup window (modal) which can contain other components.

Example: # create a modal which is activated if a link is clicked text_block = TextBlock(“This text is inside the modal”) modal = Modal(title=”This is the modal title”, size=’medium’, icon_file_path=’path/to/icon’, component=text_block) h = HtmlBlock(‘Click this link to open a modal: {0}’.format(analysis.html_ref(modal.link(‘link text’)))) layout.add_component(h)

Parameters

- **`title`** (*`str`*) – The title of the modal
- **`size`** (*`str`*) – (optional) modal size. Options are ‘small’, ‘medium’ or ‘large’

- **icon_file_path** (*str*) – (optional) path to an icon file to display in the corner of the modal
- **component** (*Component*) – The component to display in the modal

content_component_id
The id of the component to be displayed in the modal

Return type str

icon_file_id
The id of an icon file to be displayed in the corner of the modal

Return type str

icon_file_url
The url of an icon file to be displayed in the corner of the modal

Return type str

link(*link_text=None*)
Returns a ModalLink component which represents a link to this modal

Parameters **link_text** (*str*) – (optional) link text to use in the ModalLink instead of default text

Return type *eureqa.analysis.components.modal_link.ModalLink*

size
The size of the modal

Return type str

title
The title of the modal

Return type str

modal_link

class eureqa.analysis.components.modal_link.**ModalLink**(*modal=None*, *link_text=None*)
Represents a link which opens a popup window (modal) which can contain other components.

The use of this class is typically hidden by the Modal classes link method.

Example: # create a modal which is activated if a link is clicked text_block = TextBlock(“This text is inside the modal”) modal = Modal(title=”This is the modal title”, size=’medium’, icon_file_path=’path/to/icon’, component=text_block) h = HtmlBlock(‘Click this link to open a modal: {0}’.format(analysis.html_ref(modal.link(‘link text’)))) layout.add_component(h)

# create another link to the above modal h = HtmlBlock(‘OR click this other link to open the same modal: {0}’.format(analysis.html_ref(modal.link(‘different link text’)))) layout.add_component(h)

Parameters

- **modal** (eureqa.analysis.components.modal.Modal) – the Modal which this component will link to
- **link_text** (*str*) – The HTML which the link to open the modal will be wrapped around

link_text
Text of the link as rendered by the component

modal_component_id
The ID of the modal_component to be linked to

model

class eureqa.analysis.components.model.**Model**(*solution=None*)
Represents an interactive model explainer component on the server.

Parameters **solution** (Solution) – The solution that will be displayed on the card.

Variables ***solution*** (Solution) – The solution that will be displayed on the card.

solution
The Solution that is being explained

Return type eureqa.Solution

model_evaluator

class eureqa.analysis.components.model_evaluator.**ModelEvaluator**(*solutions=None*, *solution=None*)
This component evaluates models against different datasources and compares their performance.

Additional models can be added to this model by calling *add_solution_info()*

For example:

```
c = ModelEvaluator(solutions=s.get_solutions(), datasource=d, search=s, solution=s.get_best_solu
analysis.create_card(c)
```

Parameters

- **solutions** (*list[Solution]*) – Solutions to evaluate against the data
- **solution** (Solution) – Solution to fetch results from for the primary model evaluator

class SolutionInfo(*component*, *body*)
Do not instantiate directly. Use *ModelEvaluator.add_solution_info()* instead. The solution information for a single model evaluation ("tab") on ModelEvaluatorCard.

Parameters

- **body** (*str*) – internal
- **component** (*_Component*) – internal

Variables

- **datasource_id** (*str*) – ID of the DataSource referenced by this solution-tab
- **search_id** (*str*) – ID of the Search referenced by this solution-tab
- **solution_id** (*str*) – ID of the Solution referenced by this solution-tab
- **has_target_variable** (*bool*) – Whether the datasource contains the target variable

accuracy
Accuracy of this Solution. Rendered as a human-readable pretty-printed string.
Return type str

has_target_variable

Whether the datasource contains the solution's target variable.

If the datasource contains the target variable, the standard plot of this Component may include the raw target-variable data for comparison alongside the computed value.

Returns Whether the datasource contains the target variable

solution

The Solution that is being explained

Return type *Solution*

ModelEvaluator.**add_solution_info**(*solution*)

Add a new (non-default) solution and tab to this card. Once added, this object will show up in the list returned by *solution_infos*.

Parameters **solution** (Solution) – Solution associated with the model evaluation.

ModelEvaluator.**clear_solution_infos**()

Remove all existing solution infos from the current Component

ModelEvaluator.**solution**

The Solution that is being explained

Return type *Solution*

ModelEvaluator.**solution_infos**

The set of all SolutionInfo objects associated with this card. One per solution tab displayed in the UI. Note that *solution_infos[0]* is the default card; it may be treated specially by the UI.

To add a solution, use the "add_solution_info()" method.

Returns List or tuple of *SolutionInfo* objects

model_fit_by_row_plot

class eureqa.analysis.components.model_fit_by_row_plot.**ModelFitByRowPlot**(*solution=None*, *use_all_data=None*)

A model fit by row plot card. See also *Analysis.create_model_fit_by_row_plot_card*

For example:

```
p = ModelFitByRowPlot(s.get_best_solution())
analysis.create_card(p)
```

Parameters

- **solution** (Solution) – The solution that will be displayed on the card.
- **use_all_data** (*bool*) – If true, uses all data int he datasource. Otherwise use training set

solution

The Solution that is being explained

Return type eureqa.Solution

use_all_data

Use all data or just validation data?

Return type bool

model_fit_separation_plot

class `eureqa.analysis.components.model_fit_separation_plot.`**`ModelFitSeparationPlot`**(*solution=None*, *use_all_data=N*

This component is a model fit separation-plot. Separation plots are meant for use with time-series searches. They may not work properly if used with other types of searches.

Parameters

- **`solution`** (`Solution`) – The Solution object for this component
- **`use_all_data`** (*`bool`*) – If true, uses all data int he datasource. Otherwise use training set

`solution`

The Solution that is being explained

Return type eureqa.Solution

`use_all_data`

Use all data or just validation data?

Return type bool

model_summary

class `eureqa.analysis.components.model_summary.`**`ModelSummary`**(*solution=None*)

A model summary card. See also *`Analysis.create_model_summary_card`*

For example:

```
p = ModelSummary(solution=s.get_best_solution())
analysis.create_card(p)
```

Parameters **`solution`** (`Solution`) – The solution that will be displayed on the card.

`solution`

The Solution that is being explained

Return type *Solution*

model_terms_plot

class `eureqa.analysis.components.model_terms_plot.`**`ModelTermsPlot`**(*solution=None*)

A model terms plot card. See also *`Analysis.create_model_terms_plot_card`*

For example:

```
p = ModelTermsPlot(solution=s)
analysis.create_card(p)
```

Parameters **`solution`** (`Solution`) – The solution that will be displayed in the plot

`solution`

The Solution to be displayed in the plot

Return type *Solution*

most_frequent_variables_plot

class `eureqa.analysis.components.most_frequent_variables_plot.`**`MostFrequentVariablesPlot`**(*search=*

A most frequent variables plot card. See also *`Analysis.create_most_frequent_variables_plot_card`*

For example:

```
p = MostFrequentVariablesPlot(search=s)
analysis.create_card(p)
```

Parameters **`search`** (`Search`) – The search whose variable frequencies will be displayed on the card.

`search`

The Search the variables belong to

Return type *Search*

scatter_plot

class `eureqa.analysis.components.scatter_plot.`**`ScatterPlot`**(*datasource=None*,
axis_labels=None,
label_format=None,
needs_guides=None,
x_var=None,
y_var=None)

Creates a new scatter-plot card.

For example:

```
p = ScatterPlot(d, axis_labels={'x': 'the x var', 'y' : 'the y var'}, label_format={'y':'.3s'},
analysis.create_card(p)
```

Parameters

- **`datasource`** (`DataSource`) – Data source for the card's data
- **`x_var`** (*`str`*) – The X-axis variable for the card's plot
- **`y_var`** (*`str`*) – The Y-axis variable for the card's plot
- **`needs_guides`** (*`bool`*) – Whether the card needs guides
- **`axis_labels`** (*`list`*) – Axis labels for this card's plot
- **`label_format`** (*`list`*) – Label format for this card

`axis_labels`

The axis labels for this card.

Defaults to:

```
{ 'x': x_var, 'y': y_var }
```

Returns Axis labels for this card

Return type self.XYMap

datasource

The data source providing data for this component

Return type eureqa.DataSource

label_format

Label format for this card. Set keys “x” and “y” to set the X and Y axis printf-style format-strings; for example, ”.3s”.

Defaults to:

```
{ 'x': '.3s', 'y': '.3s' }
```

Return type DoubleHistogramPlot.XYMap

needs_guides

Does this card need guides?

Returns Whether this card needs guides

Return type bool

x_var

The X variable for this card.

Returns X variable for this card

Return type str

y_var

The Y variable for this card.

Returns Y variable for this card

Return type str

search_builder_link

class `eureqa.analysis.components.search_builder_link.`**`SearchBuilderLink`**(*link_text=None*, *datasource=None*, *search_template=None*, *min_delay=None*, *max_delay=None*, *target_variable=None*, *input_variables=None*)

Create a SearchBuilderLink component

Example of using HtmlBlock with a SearchBuilderLink:

```
# include a link to the search builder with nothing pre-populated
search_builder_link = SearchBuilderLink()
h = HtmlBlock('Click this link to open the pre-populated search builder: {0}'.format(analysis.ht
layout.add_component(h)

# include a link to the search builder with settings pre-populated by a search template
datasource = e.create_data_source("data_source_1", "tests/Nutrition.csv")
search_builder_link = SearchBuilderLink(datasource=datasource,
                                        search_template=search_templates.SearchTemplates.Timeser
                                        min_delay=1,
                                        max_delay=10,
                                        target_variable='Calories',
                                        input_variables=['Steps', 'Weight', '(Protein (g))', '(F
h = HtmlBlock('Click this link to open the pre-populated search builder: {0}'.format(analysis.ht
layout.add_component(h)
```

Parameters

- **`link_text`** (*`str`*) – the visible, clickable text for the link
- **`datasource`** (`DataSource`) – datasource to pre-populate the search builder with
- **`search_template`** (*`SearchTemplate`*) – search template information to pre-populate the search builder with
- **`min_delay`** (*`int`*) – the minimum delay used in a timeseries search
- **`max_delay`** (*`int`*) – the maximum delay used in a timeseries search
- **`target_variable`** (*`str`*) – the variable to search for models of
- **`input_variables`** (*`str[]`*) – the variables to use as inputs in the models

`link_text`

The HTML text which will be wrapped with a link to the search builder.

Return type str

search_link

class `eureqa.analysis.components.search_link.`**`SearchLink`**(*search=None*, *link_text=None*)

Create a link to a specified search

Example of using HtmlBlock with a SearchLink:

```
    vl = SearchLink(search)
    h = HtmlBlock('Formatted <b>Card</b> in the Analysis, with reference to search {0}'.format(analy
    analysis.create_card(h)
```

Parameters

- **`search`** (`Search`) – the search to link to
- **`link_text`** (*`str`*) – (optional) the text to use for the search link is displayed

`link_text`

The text to use for the search link when displayed.

Returns the link text of this card

Return type str

`search`

The search linked to

Return type *Search*

tabbed_layout

class `eureqa.analysis.components.tabbed_layout.`**`TabbedLayout`**`(`*tab_type='default'*`)`

TabbedLayout component

Lets users pick to see other components based on titles (works with a few items). Similar to `DropdownLayout` but suited to smaller lists

For example:

```
t = TabbedLayout()
t.add_component(title="choice 1", component=HtmlBlock("This is item 1"))
t.add_component(title="choice 2", component=HtmlBlock("This is item 2"))
analysis.create_card(t)
```

`add_component``(`*title*, *component*, *icon=None*`)`

Add a new tab containing a new component

Parameters

- **`title`** (*`str`*) – Title of the tab
- **`component`** (*`_Component`*) – Tab contents, as a Component
- **`icon`** (`AnalysisFile`) – Tab icon (optional) – must be a file containing an image

table

table_builder

class `eureqa.analysis.components.table_builder.`**`TableBuilder`**`(`*data*, *title*, *default_rows_per_page=20*, *column_names=None*, *striped=True*, *search_box_place_holder='Search'*, *page_controls_and_search_visible=True*, *default_sort_order=None*`)`

High level API to build a table in analysis

Parameters

- **`data`** (*`object`*) – The data to store in the table. Accepts *pandas.DataFrame*, *numpy.ndarray*, a list of floats, or a list of strings
- **`title`** (*`str`*) – table title
- **`default_rows_per_page`** (*`int`*) – the default number of rows per page when the table is initially rendered
- **`column_names`** (*`list`*) – names of each column specified as a list of strings
- **`striped`** (*`bool`*) – whether or not the table rows should be rendered in alternating grey and white stripes
- **`search_box_place_holder`** (*`str`*) – the default text shown in the search box when user hasn't typed any search keyword
- **`page_controls_and_search_visible`** (*`bool`*) – if true show a table header with next/previous page and search controls enabled

- **default_sort_order** (*str/tuple*) – If specified, name of the column to sort the table by default. Sort order can be specified by using a tuple with the first element being the name of the column and the second element being either 'ASC' or 'DESC' to specify ascending or descending order.

table_column

class eureqa.analysis.components.table_column.**TableColumn**(*parent_table*, *col_data*, *col_name*)

Represent a column in table

Parameters

- **parent_table** (TableBuilder) – the containing table
- **col_data** (*list*) – data for this column. Can be either a list of str or a list of float
- **col_name** (*str*) – name of this column

column_name

The name of this column

filter_only

Whether this column is only for filtering, if True this column doesn't appear in the table

rendered_values

A list of values specifying how the column is rendered

sort_values

A list of values specifying how the column is sorted

width

The width of this column

text_block

class eureqa.analysis.components.text_block.**TextBlock**(*text=''*, *description=None*)

Contains free-form user-specified text, formatted using markdown. *Deprecated*. Use *HtmlBlock* instead.

Parameters

- **text** (*str*) – Body of the card.
- **description** (*str*) – alias for text (backwards compatible)

Variables ***eureqa.analysis.components.TextBlock.text*** (*str*) – Body of the card.

text

Markdown-formatted text contents of this component

Return type str

threshold_selection_plot

class eureqa.analysis.components.threshold_selection_plot.**ThresholdSelectionPlot**(*solution=None*)

A threshold selection plot card. See also *Analysis.create_threshold_selection_plot_card*

For example:

```
p = ThresholdSelectionPlot(solution=s)
analysis.create_card(p)
```

Parameters **solution** (Solution) – The solution that will be displayed in the plot

solution

The Solution to be displayed in the plot

Return type *Solution*

titled_layout

class eureqa.analysis.components.titled_layout.**TitledLayout**(*title=None*, *description=None*, *content=None*)

A layout with space for a title and description and content within an Analysis

For example:

```
h=HtmlBlock(html="<h1>Hybrid Performance</h1>")
layout=TitledLayout(title="the title", description="You can add content specific description her
analysis.create_card(layout)
```

Parameters

- **title** (*str*) – The title of the layout
- **description** (*str*) – The description of the layout (can contain HTML)
- **content** (*_Component*) – The component to use in the main content of the layout

content

The Component that this TitledLayout is adding a title to. This field can't be assigned to; use 'create_card()' to assign a Component to this TitledLayout.

Return type _Component

create_card(*component*)

Assign a Component to this TitledLayout

Parameters **component** (*_Component*) – Component to use as the content of this layout

description

The description of this card.

Returns description of this card

Return type str

title

The title of this card.

Returns title of this card

Return type str

tooltip

class eureqa.analysis.components.tooltip.**Tooltip**(*html=None*, *tooltip=None*)

Tooltip Component: implements a tooltip

For example:

```
tt = Tooltip(html="This", tooltip="Text to show when hovering")
card = analysis.create_html_card("This is a component with a tooltip: {0}".format(analysis.html_
```

Parameters

- **html** (*str*) – Text to display normally
- **tooltip** (*str*) – The tooltip text to show when the cursor hovers over the component

html

The text to show normally for this component.

Return type str

tooltip

The tooltip text to show when the cursor hovers over the component

Return type str

variable_link

class eureqa.analysis.components.variable_link.**VariableLink**(*datasource=None*, *variable_name=None*)

Create a VariableLink component

Example of using HtmlBlock with a VariableLink:

```
vl = VariableLink(d, 'X')
h = HtmlBlock('Formatted <b>Card</b> in the Analysis, with reference to {0}'.format(analysis.htm
analysis.create_card(h)
```

Parameters

- **datasource** (DataSource) – datasource from which to find the variable
- **variable_name** (*str*) – name of the variable

datasource

The data source providing data for this component

Return type *DataSource*

variable_name

The name of the variable from this datasource to link to

Return type str

analysis_templates

class eureqa.analysis_templates.**AnalysisTemplate**

Represents an analysis template on the server.

Variables

- **name** (*str*) – The name of the analysis template.
- **description** (*str*) – The description of the analysis template.
- **icon_url** (*str*) – The url of the icon for the analysis template.
- **icon_url_fill_custom** (*str*) – If true, the icon_url will be filled with the analysis template's custom icon URL.
- **parameters** (*list*) – The list of parameters for the analysis template.

exception ValidationException(*results*)

Represents a template validation failure. :var list results: Validation results

AnalysisTemplate.**delete**()

Delete the analysis template.

AnalysisTemplate.**description**

Analysis Template's extended description

AnalysisTemplate.**execute**(*values*)

Start execution of an analysis template with given parameters.

Wait for the template to validate successfully. If it does not validate successfully, throw a ~eureqa.analysis_templates.analysis_template.AnalysisTemplate.ValidationException containing the template's validation results.

Parameters values (ParametersValues) – The parameter values to pass to the template.

For example:

```
ParametersValues(
    [TextParameterValue(eq, 'example_text_param','This is the text input value.'),
     DataSourceParameterValue(eq, '123', 'SimpleDataSource',
        {VariableParameterValue(eq, '345', 'value 345')})],
)
```

AnalysisTemplate.**get_executions**()

Get all executions of the analysis template.

AnalysisTemplate.**get_module**(*output_filename*)

Download the Python code which implementes this analysis template, as a .zip package, to the specified file on the local filesystem. Returns the main_module's name

Parameters output_filename (*str*) – The filename to be used for the resulting zip file containing the analysis template code.

AnalysisTemplate.**name**

Analysis Template's name

AnalysisTemplate.**parameters**

Parameters object representing this template

AnalysisTemplate.**set_icon**(*icon_filepath*)

Set the icon used for this analysis template

Parameters icon_filepath (*str*) – path to a local file with a custom icon for the template.

AnalysisTemplate.**set_module**(*main_module_name*, *module_fs_path*, *ignore_files=None*, *additional_modules_paths=[]*, *icon_path=None*)

Set the the python module and function that specifies the execution of this analysis template.

Parameters

- **main_module_name** (*str*) – Absolute Python-import name of the main module to run. For example, "example_module".
- **module_fs_path** (*str*) – Filesystem path where the module containing the function lives. For example, "C:Userseureqaexample_module". If omitted, this is inferred by importing the function above and taking the parent directory of the file that contains it.
- **ignore_files** (*list[str]*) – List of names of files that, if encountered, will not be uploaded to the Eureqa server.
- **additional_modules_paths** (*list*) – List of other directory names, that, if encountered, will be uploaded to the Eureqa server as well
- **icon_path** (*str*) – (Optional) path to an icon to use for the analysis

class `eureqa.analysis_templates.`**ComboBoxParameter**(*id*, *label*, *items*)

Combo box parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **items** (*list[str]*) – The items to populate the combo box with.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **ComboBoxParameter.label** (*str*) – The parameter label that will be shown in UI.
- **items** (*list[str]*) – The items to populate the combo box with.

class `eureqa.analysis_templates.`**ComboBoxParameterValue**(*eureqa*, *id*, *value*)

Combo box parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **eureqa** (Eureqa) – A eureqa connection
- **value** (*str*) – The parameter value.

Variables

- **id** (*str*) – The id of the parameter.
- **ComboBoxParameterValue.value** (*str*) – The parameter value.

class `eureqa.analysis_templates.`**DataFileParameter**(*id*, *label*, *description*, *filetypes*)

Data file parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **description** (*str*) – A description that will be shown in UI.
- **filetypes** (*list[str]*) – The list of accepted filetypes.

Variables

- **eureqa.analysis_templates.DataFileParameter.id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **eureqa.analysis_templates.DataFileParameter.label** (*str*) – The parameter label that will be shown in UI.
- **eureqa.analysis_templates.DataFileParameter.description** (*str*) – A description that will be shown in UI.
- **filetypes** (*list[str]*) – The list of accepted filetypes.

class eureqa.analysis_templates.**DataFileParameterValue**(*eureqa*, *id*, *file_path*)

Combo box parameter description for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **file_path** (*str*) – The path to the file that should be uploaded to the server. If running locally, this path will be used to retreive the file directly.

Variables

- **id** (*str*) – The id of the parameter.
- **file** (*str*) – the contents of the uploaded file
- **value** (*str*) – the file id of the file on the server

class eureqa.analysis_templates.**DataSourceParameter**(*id*, *label*, *variables*)

Data source parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **variables** (*list[eureqa.analysis_templates.VariableParameter]*) – The parameters for variables that belong to the datasource described by this parameter.

class eureqa.analysis_templates.**DataSourceParameterValue**(*eureqa*, *id*, *value*, *variables*)

Data source parameter description for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*str*) – The parameter value.
- **variables** (*list[eureqa.analysis_templates.VariableParameterValue]*) – The parameters values for variables that belong to this data source.

class eureqa.analysis_templates.**Execution**(*body*, *template_id*, *eureqa*)

Represents an analysis template execution on the server.

Parameters

- **body** (*dict*) – Class metadata as dictionary
- **template_id** (*str*) – The id of the analysis_template the execution belongs to.

- **eureqa** (Eureqa) – A eureqa connection.

Variables

- **template_id** (*str*) – The id of the analysis_template the execution belongs to.
- **analysis_id** (*str*) – The id of the analysis the execution belongs to.
- **state** (*str*) – The current state of the execution.
- **parameters** (*list*) – The list of parameter values for the execution.
- ***progress_updates*** (*list*) – The list of updates for the execution.

get_analysis()
Retrieves the analysis that belongs to the execution.

get_analysis_template()
Retrieves the analysis that belongs to the execution.

progress_updates
Get all progress updates for an execution of an analysis template.

report_fatal_error(*error*)
Notifies the server that an error occurred during the execution and terminates the script execution.

Parameters **error** (*str*) – The error that occurred during the execution.

report_validation_result(*type*, *message=None*, *details=None*, *parameter_id=None*)
Report an info/warning/error message about the specified parameter to be shown to the user in validation review

Parameters

- **type** (*str*) – If this result is INFO, WARNING, or ERROR
- **message** (*str*) – The result message
- **details** (*str*) – (optional) Detailed message about the result
- **parameter_id** (*str*) – (optional) the analysis template parameter that this progress update refers to

update_progress(*message*)
Create a progress update for an execution of an analysis template.

Parameters **message** (*str*) – The progress message

validation_results
Get all validation results for the execution of an analysis template.

class eureqa.analysis_templates.**NumericParameter**(*id*, *label*, *min=None*, *max=None*, *require_int=False*)
Numeric parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **min** (*float*) – The minimum value to allow.
- **max** (*float*) – The maximum value to allow.
- **require_int** (*bool*) – Indicates that the number should be an integer.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **NumericParameter.label** (*str*) – The parameter label that will be shown in UI.
- **min** (*float*) – The minimum value to allow.
- **max** (*float*) – The maximum value to allow.
- **reqire_int** (*bool*) – Indicates that the number should be an integer.

class eureqa.analysis_templates.**NumericParameterValue**(*eureqa*, *id*, *value*)

Numeric parameter value description for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*float*) – The parameter value.

Variables

- **id** (*str*) – The id of the parameter.
- **NumericParameterValue.value** (*float*) – The parameter value.

class eureqa.analysis_templates.**Parameters**(*parameters=None*)

Analysis template parameters definition

Parameters **parameters** (*list[Parameter]*) – The list of parameters for the template, whether text, variable or datasource.

class eureqa.analysis_templates.**ParametersValues**(*parameters=None*)

Analysis template parameters values

Parameters **parameters** (*list[Parameter]*) – The list of parameter values for the template, whether text, variable or datasource value.

class eureqa.analysis_templates.**ProgressUpdate**(*body*)

Represents an analysis template execution progress update on the server.

Parameters **body** (*dict*) – Class metadata as dictionary

Variables

- **message** (*str*) – The the message about the execution's status.
- **time_stamp** (*datetime*) – The time the progress update was created.

class eureqa.analysis_templates.**TextParameter**(*id*, *label*, *text_multiline=False*)

Text parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **text_multiline** (*bool*) – Indicates that the text should be split across multiple lines.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **TextParameter.label** (*str*) – The parameter label that will be shown in UI.

class eureqa.analysis_templates.**TextParameterValue**(*eureqa*, *id*, *value*, *text_multiline=False*)

Text parameter description for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*str*) – The parameter value.
- **text_multiline** (*bool*) – Whether to display as multiline text

Variables

- **id** (*str*) – The id of the parameter.
- **TextParameterValue.value** (*str*) – The parameter value.

class eureqa.analysis_templates.**TopLevelModelParameter**(*id*, *label*, *custom_disabled*)

TopLevelModel parameter description for analysis template. For selecting a single model (either an existing one within a datasource and search, or a custom one).

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **custom_disabled** (*bool*) – Whether to block the user from setting a custom expression

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **TopLevelModelParameter.label** (*str*) – The parameter label that will be shown in UI.
- **custom_disabled** (*bool*) – Whether to block the user from setting a custom expression

class eureqa.analysis_templates.**TopLevelModelParameterValue**(*eureqa*, *id*, *value*, *datasource_id*, *search_id*, *solution_id*)

TopLevelModel parameter value for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*str*) – The parameter value.
- **datasource_id** (*str*) – The parameter value for the datasource the expression belongs to.
- **search_id** (*str*) – The parameter value for the search the expression belongs to.
- **solution_id** (*str*) – The parameter value for the solution the expression belongs to.

Variables

- **id** (*str*) – The id of the parameter.
- **TopLevelModelParameterValue.value** (*str*) – The parameter value.

class eureqa.analysis_templates.**VariableParameter**(*id*, *label*)

Variable parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **VariableParameter.label** (*str*) – The parameter label that will be shown in UI.

class eureqa.analysis_templates.**VariableParameterValue**(*eureqa*, *id*, *value*)

Variable parameter value for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*str*) – The parameter value.

Variables

- **id** (*str*) – The id of the parameter.
- **eureqa.analysis_templates.VariableParameterValue.value** (*str*) – The parameter value.

class eureqa.analysis_templates.**ParameterValidationResult**(*type*, *message=None*, *details=None*, *parameter_id=None*)

Represents an analysis template execution validation result to the server

Parameters

- **type** (*str*) – If this result is INFO, WARNING, or ERROR
- **message** (*str*) – The the message
- **details** (*str*) – (optional) Detailed message about the result
- **parameter_id** (*str*) – (optional) the analysis template parameter that this progress update refers to

Variables

- **type** (*str*) – If this result is INFO, WARNING, or ERROR
- **message** (*str*) – The the message
- **details** (*str*) – (optional) Detailed message about the result
- **parameter_id** (*str*) – (optional) the analysis template parameter that this progress update refers to

analysis_template

class `eureqa.analysis_templates.analysis_template.`**`AnalysisTemplate`**

Represents an analysis template on the server.

Variables

- **`name`** (*`str`*) – The name of the analysis template.
- **`description`** (*`str`*) – The description of the analysis template.
- **`icon_url`** (*`str`*) – The url of the icon for the analysis template.
- **`icon_url_fill_custom`** (*`str`*) – If true, the icon_url will be filled with the analysis template's custom icon URL.
- **`parameters`** (*`list`*) – The list of parameters for the analysis template.

exception **`ValidationException`**(*results*)

Represents a template validation failure. :var list results: Validation results

`AnalysisTemplate.`**`delete`**()

Delete the analysis template.

`AnalysisTemplate.`**`description`**

Analysis Template's extended description

`AnalysisTemplate.`**`execute`**(*values*)

Start execution of an analysis template with given parameters.

Wait for the template to validate successfully. If it does not validate successfully, throw a ~eureqa.analysis_templates.analysis_template.AnalysisTemplate.ValidationException containing the template's validation results.

Parameters **`values`** (`ParametersValues`) – The parameter values to pass to the template.

For example:

```
ParametersValues(
    [TextParameterValue(eq, 'example_text_param','This is the text input value.'),
     DataSourceParameterValue(eq, '123', 'SimpleDataSource',
        {VariableParameterValue(eq, '345', 'value 345')})],
)
```

`AnalysisTemplate.`**`get_executions`**()

Get all executions of the analysis template.

`AnalysisTemplate.`**`get_module`**(*output_filename*)

Download the Python code which implementes this analysis template, as a .zip package, to the specified file on the local filesystem. Returns the main_module's name

Parameters **`output_filename`** (*`str`*) – The filename to be used for the resulting zip file containing the analysis template code.

`AnalysisTemplate.`**`name`**

Analysis Template's name

`AnalysisTemplate.`**`parameters`**

Parameters object representing this template

`AnalysisTemplate.`**`set_icon`**(*icon_filepath*)

Set the icon used for this analysis template

Parameters **`icon_filepath`** (*`str`*) – path to a local file with a custom icon for the template.

AnalysisTemplate.**set_module**(*main_module_name*, *module_fs_path*, *ignore_files=None*, *additional_modules_paths=[]*, *icon_path=None*)

Set the the python module and function that specifies the execution of this analysis template.

Parameters

- **main_module_name** (*str*) – Absolute Python-import name of the main module to run. For example, "example_module".
- **module_fs_path** (*str*) – Filesystem path where the module containing the function lives. For example, "C:Userseureqaexample_module". If omitted, this is inferred by importing the function above and taking the parent directory of the file that contains it.
- **ignore_files** (*list[str]*) – List of names of files that, if encountered, will not be uploaded to the Eureqa server.
- **additional_modules_paths** (*list*) – List of other directory names, that, if encountered, will be uploaded to the Eureqa server as well
- **icon_path** (*str*) – (Optional) path to an icon to use for the analysis

combo_box_parameter

class eureqa.analysis_templates.combo_box_parameter.**ComboBoxParameter**(*id*, *label*, *items*)

Combo box parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **items** (*list[str]*) – The items to populate the combo box with.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **ComboBoxParameter.label** (*str*) – The parameter label that will be shown in UI.
- **items** (*list[str]*) – The items to populate the combo box with.

combo_box_parameter_value

class eureqa.analysis_templates.combo_box_parameter_value.**ComboBoxParameterValue**(*eureqa*, *id*, *value*)

Combo box parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **eureqa** (Eureqa) – A eureqa connection
- **value** (*str*) – The parameter value.

Variables

- **id** (*str*) – The id of the parameter.
- **ComboBoxParameterValue.value** (*str*) – The parameter value.

data_file_parameter

class eureqa.analysis_templates.data_file_parameter.**DataFileParameter**(*id*, *label*, *description*, *filetypes*)

Data file parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **description** (*str*) – A description that will be shown in UI.
- **filetypes** (*list[str]*) – The list of accepted filetypes.

Variables

- **eureqa.analysis_templates.DataFileParameter.id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **eureqa.analysis_templates.DataFileParameter.label** (*str*) – The parameter label that will be shown in UI.
- **eureqa.analysis_templates.DataFileParameter.description** (*str*) – A description that will be shown in UI.
- **filetypes** (*list[str]*) – The list of accepted filetypes.

data_file_parameter_value

class eureqa.analysis_templates.data_file_parameter_value.**DataFileParameterValue**(*eureqa*, *id*, *file_path*)

Combo box parameter description for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **file_path** (*str*) – The path to the file that should be uploaded to the server. If running locally, this path will be used to retreive the file directly.

Variables

- **id** (*str*) – The id of the parameter.
- **file** (*str*) – the contents of the uploaded file
- **value** (*str*) – the file id of the file on the server

data_source_parameter

class `eureqa.analysis_templates.data_source_parameter.`**`DataSourceParameter`**(*id*, *label*, *variables*)

Data source parameter description for analysis template

Parameters

- **`id`** (*`str`*) – The id of the parameter that will be passed together with its value to an analysis script.
- **`label`** (*`str`*) – The parameter label that will be shown in UI.
- **`variables`** (*`list[eureqa.analysis_templates.VariableParameter]`*) – The parameters for variables that belong to the datasource described by this parameter.

data_source_parameter_value

class `eureqa.analysis_templates.data_source_parameter_value.`**`DataSourceParameterValue`**(*eureqa*, *id*, *value*, *variables*)

Data source parameter description for analysis template

Parameters

- **`eureqa`** (`Eureqa`) – A eureqa connection.
- **`id`** (*`str`*) – The id of the parameter.
- **`value`** (*`str`*) – The parameter value.
- **`variables`** (*`list[eureqa.analysis_templates.VariableParameterValue]`*) – The parameters values for variables that belong to this data source.

execution

class `eureqa.analysis_templates.execution.`**`Execution`**(*body*, *template_id*, *eureqa*)

Represents an analysis template execution on the server.

Parameters

- **`body`** (*`dict`*) – Class metadata as dictionary
- **`template_id`** (*`str`*) – The id of the analysis_template the execution belongs to.
- **`eureqa`** (`Eureqa`) – A eureqa connection.

Variables

- **`template_id`** (*`str`*) – The id of the analysis_template the execution belongs to.
- **`analysis_id`** (*`str`*) – The id of the analysis the execution belongs to.
- **`state`** (*`str`*) – The current state of the execution.
- **`parameters`** (*`list`*) – The list of parameter values for the execution.

- **progress_updates** (*list*) – The list of updates for the execution.

get_analysis()

Retrieves the analysis that belongs to the execution.

get_analysis_template()

Retrieves the analysis that belongs to the execution.

progress_updates

Get all progress updates for an execution of an analysis template.

report_fatal_error(*error*)

Notifies the server that an error occurred during the execution and terminates the script execution.

Parameters error (*str*) – The error that occurred during the execution.

report_validation_result(*type*, *message=None*, *details=None*, *parameter_id=None*)

Report an info/warning/error message about the specified parameter to be shown to the user in validation review

Parameters

- **type** (*str*) – If this result is INFO, WARNING, or ERROR
- **message** (*str*) – The result message
- **details** (*str*) – (optional) Detailed message about the result
- **parameter_id** (*str*) – (optional) the analysis template parameter that this progress update refers to

update_progress(*message*)

Create a progress update for an execution of an analysis template.

Parameters message (*str*) – The progress message

validation_results

Get all validation results for the execution of an analysis template.

numeric_parameter

class eureqa.analysis_templates.numeric_parameter.**NumericParameter**(*id*, *label*, *min=None*, *max=None*, *require_int=False*)

Numeric parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **min** (*float*) – The minimum value to allow.
- **max** (*float*) – The maximum value to allow.
- **require_int** (*bool*) – Indicates that the number should be an integer.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **NumericParameter.label** (*str*) – The parameter label that will be shown in UI.
- ***min*** (*float*) – The minimum value to allow.
- ***max*** (*float*) – The maximum value to allow.
- **reqire_int** (*bool*) – Indicates that the number should be an integer.

numeric_parameter_value

class `eureqa.analysis_templates.numeric_parameter_value.`**`NumericParameterValue`**(*eureqa*, *id*, *value*)

Numeric parameter value description for analysis template

Parameters

- **eureqa** (Eureqa) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*float*) – The parameter value.

Variables

- **id** (*str*) – The id of the parameter.
- **NumericParameterValue.value** (*float*) – The parameter value.

parameter

class `eureqa.analysis_templates.parameter.`**`Parameter`**(*id*, *label*, *_type*)

Base class for all analysis templates parameters

Parameters

- **id** (*str*) – The unique identifier for this Parameter.
- **label** (*str*) – The human-readable label to be associated with this Parameter.

parameters

class `eureqa.analysis_templates.parameters.`**`Parameters`**(*parameters=None*)

Analysis template parameters definition

Parameters **parameters** (*list[Parameter]*) – The list of parameters for the template, whether text, variable or datasource.

parameters_values

class `eureqa.analysis_templates.parameters_values.`**`ParametersValues`**(*parameters=None*)

Analysis template parameters values

Parameters **parameters** (*list[Parameter]*) – The list of parameter values for the template, whether text, variable or datasource value.

parameter_validation_result

class `eureqa.analysis_templates.parameter_validation_result.`**`ParameterValidationResult`**(*type*, *message=Non* *details=Non* *parameter_id=No*

Represents an analysis template execution validation result to the server

Parameters

- **`type`** (*`str`*) – If this result is INFO, WARNING, or ERROR
- **`message`** (*`str`*) – The the message
- **`details`** (*`str`*) – (optional) Detailed message about the result
- **`parameter_id`** (*`str`*) – (optional) the analysis template parameter that this progress update refers to

Variables

- **`type`** (*`str`*) – If this result is INFO, WARNING, or ERROR
- **`message`** (*`str`*) – The the message
- **`details`** (*`str`*) – (optional) Detailed message about the result
- **`parameter_id`** (*`str`*) – (optional) the analysis template parameter that this progress update refers to

parameter_value

class `eureqa.analysis_templates.parameter_value.`**`ParameterValue`**(*id*, *value*, *_type*)

Base class for all analysis templates parameters values

Parameters

- **`id`** (*`str`*) – The unique identifier for this parameter.
- **`value`** (*`str`*) – The value of this parameter.

progress_update

class `eureqa.analysis_templates.progress_update.`**`ProgressUpdate`**(*body*)

Represents an analysis template execution progress update on the server.

Parameters **`body`** (*`dict`*) – Class metadata as dictionary

Variables

- **`message`** (*`str`*) – The the message about the execution's status.
- **`time_stamp`** (*`datetime`*) – The time the progress update was created.

runner

class `eureqa.analysis_templates.runner.`**`analysis_template_runner`**

Bootstrapper for analysis templates. Sets up the environment required to invoke the analysis template and then invokes it.

`get_analysis_module_from_template`(*eureqa*, *template*)

Retrieves the bytes that represent an analysis template from the eureqa server (as base64-encoded .zip file module); then function, raising an Exception if an error occurs.

Parameters

- **`eureqa`** (`Eureqa`) – A eureqa connection.
- **`template`** (*`Template`*) – The Template object containing the desired analysis.

`run_args`(*arguments*)

Run a template with the provided arguments.

Parameters **`arguments`** (*`str`*) – Command line-style argument string.

analysis_template_runner

exception `eureqa.analysis_templates.runner.analysis_template_runner.`**`LocalFatalException`**

Class that represents fatal exceptions from the local runner

class `eureqa.analysis_templates.runner.analysis_template_runner.`**`Local_analysis_template_execu`**

Stubs out the analysis_template_execution to provide the same interface while running locally.

Parameters

- **`eureqa`** (`Eureqa`) – A eureqa connection.
- **`parameters`** (*`str`*) – JSON string of parameters.

`report_fatal_error`(*error*)

Report a fatal error of the running analysis.

Parameters **`error`** (*`str`*) – Message to print indicating progress.

`throw_if_fatal_exception`()

Throw any error encountered as a fatal exception

`update_progress`(*message*)

Update progress of the running analysis.

Parameters **`message`** (*`str`*) – Message to print indicating progress.

class `eureqa.analysis_templates.runner.analysis_template_runner.`**`analysis_template_runner`**

Bootstrapper for analysis templates. Sets up the environment required to invoke the analysis template and then invokes it.

`get_analysis_module_from_template`(*eureqa*, *template*)

Retrieves the bytes that represent an analysis template from the eureqa server (as base64-encoded .zip file module); then function, raising an Exception if an error occurs.

Parameters

- **`eureqa`** (`Eureqa`) – A eureqa connection.

- **template** (*Template*) – The Template object containing the desired analysis.

run_args (*arguments*)
Run a template with the provided arguments.

Parameters arguments (*str*) – Command line-style argument string.

client

eureqa.analysis_templates.runner.client.**main** (*argv*)
Handle executing analysis template code without actually uploading it to the server. Mostly a wrapper around *analysis_template_runner*. The latter is called directly by the server; this one is a little more user-friendly and respects eureqa_config.json.

Parameters argv (*list*) – Arguments

text_parameter

class eureqa.analysis_templates.text_parameter.**TextParameter** (*id*, *label*, *text_multiline=False*)
Text parameter description for analysis template

Parameters

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **label** (*str*) – The parameter label that will be shown in UI.
- **text_multiline** (*bool*) – Indicates that the text should be split across multiple lines.

Variables

- **id** (*str*) – The id of the parameter that will be passed together with its value to an analysis script.
- **TextParameter.label** (*str*) – The parameter label that will be shown in UI.

text_parameter_value

class eureqa.analysis_templates.text_parameter_value.**TextParameterValue** (*eureqa*, *id*, *value*, *text_multiline=False*)
Text parameter description for analysis template

Parameters

- **eureqa** (*Eureqa*) – A eureqa connection.
- **id** (*str*) – The id of the parameter.
- **value** (*str*) – The parameter value.
- **text_multiline** (*bool*) – Whether to display as multiline text

Variables

- **id** (*str*) – The id of the parameter.
- **TextParameterValue.value** (*str*) – The parameter value.

top_level_model_parameter

class `eureqa.analysis_templates.top_level_model_parameter.`**`TopLevelModelParameter`**(*id*, *label*, *custom_disabled*)

TopLevelModel parameter description for analysis template. For selecting a single model (either an existing one within a datasource and search, or a custom one).

Parameters

- **`id`** (*`str`*) – The id of the parameter that will be passed together with its value to an analysis script.
- **`label`** (*`str`*) – The parameter label that will be shown in UI.
- **`custom_disabled`** (*`bool`*) – Whether to block the user from setting a custom expression

Variables

- **`id`** (*`str`*) – The id of the parameter that will be passed together with its value to an analysis script.
- **`TopLevelModelParameter.label`** (*`str`*) – The parameter label that will be shown in UI.
- **`custom_disabled`** (*`bool`*) – Whether to block the user from setting a custom expression

top_level_model_parameter_value

class `eureqa.analysis_templates.top_level_model_parameter_value.`**`TopLevelModelParameterValue`**(*e*
i
v
d
s
s
s
l
t

TopLevelModel parameter value for analysis template

Parameters

- **`eureqa`** (`Eureqa`) – A eureqa connection.
- **`id`** (*`str`*) – The id of the parameter.
- **`value`** (*`str`*) – The parameter value.
- **`datasource_id`** (*`str`*) – The parameter value for the datasource the expression belongs to.
- **`search_id`** (*`str`*) – The parameter value for the search the expression belongs to.
- **`solution_id`** (*`str`*) – The parameter value for the solution the expression belongs to.

Variables

- **`id`** (*`str`*) – The id of the parameter.
- **`TopLevelModelParameterValue.value`** (*`str`*) – The parameter value.

variable_parameter

class `eureqa.analysis_templates.variable_parameter.`**`VariableParameter`**(*id*, *label*)

Variable parameter description for analysis template

Parameters

- **`id`** (*`str`*) – The id of the parameter that will be passed together with its value to an analysis script.
- **`label`** (*`str`*) – The parameter label that will be shown in UI.

Variables

- **`id`** (*`str`*) – The id of the parameter that will be passed together with its value to an analysis script.
- **`VariableParameter.label`** (*`str`*) – The parameter label that will be shown in UI.

variable_parameter_value

class `eureqa.analysis_templates.variable_parameter_value.`**`VariableParameterValue`**(*eureqa*, *id*, *value*)

Variable parameter value for analysis template

Parameters

- **`eureqa`** (`Eureqa`) – A eureqa connection.
- **`id`** (*`str`*) – The id of the parameter.
- **`value`** (*`str`*) – The parameter value.

Variables

- **`id`** (*`str`*) – The id of the parameter.
- **`eureqa.analysis_templates.VariableParameterValue.value`** (*`str`*) – The parameter value.

apply_solution_result

data_source

class `eureqa.data_source.`**`DataSource`**

Acts as an interface to a data source on the server.

DataSources can be created by calling *`create_data_source()`* or an existing one can be retreived with *`get_data_source()`*

Variables

- **`name`** (*`str`*) – The data source name.
- ***`series_id_column_name`*** (*`str`*) – The name of the column that splits data into series based on its values.

- ***series_order_column_name*** (*str*) – The name of the column that defines the order of the data. This indicates the data is timeseries data and will sort the rows based on this column.
- **number_variables** (*int*) – The number of variables in the data source.
- **number_rows** (*int*) – The number of rows in the data source.
- **number_series** (*int*) – The number of series (chunks of rows) in the data source.

create_search(*search_settings*)
Creates a new search with settings from a *SearchSettings* object.

Parameters search_settings (SearchSettings) – the settings for creating a new search.

Returns A *Search* object which represents a newly create search on the server.

Return type *Search*

create_seasonality_variable(*seasonal_target_variable*, *seasonal_period*)
Adds a seasonality variable to the data_source with specified target variable and seasonal period

Parameters

- **seasonal_target_variable** (*str*) – the variable name to calculate the seasonal trend for for
- **seasonal_period** (*str*) – period of the seasonal effect, supported options are 'yearly', 'weekly' or 'daily'

Returns name of the newly created variable

Return type str

create_variable(*expression*, *variable_name*)
Adds a new variable to the DataSource with values from evaluating the given expression.

Parameters

- **expression** (*str*) – the expression to evaluate to fill in the values
- **variable_name** (*str*) – what to name the new variable

create_variable_from_template(*template*)
Create a new derived variable on this data source with the same properties as the template variable.

Parameters template (VariableDetails) – the template to use for creating the new variable

delete()
Deletes the data source from the server.

Raises Exception – If the data source is already deleted.

download_data_file(*file_path*)
Downloads the originally uploaded data from the server.

Parameters file_path (*str*) – the filepath at which to save the data

get_searches()
Retrieves from the server a list of searches associated with the data source.

Returns The list of all searches associated with the data source.

Return type list of *Search*

get_series_id_values()

Get all available series id values for this dataset. Note the series id for a single series dataset is reported to be ‘’

Returns a list of all available series_id values

Return type list

get_variable_details(*variable_name*)

Retrieves the details for the requested variable from the `DataSource`.

Parameters **variable_name** (*str*) – the name of the variable to get the details for

Returns The object representing the variable details

Return type *VariableDetails*

get_variables(*derivation_type='all'*)

Retrieves from the server a list of variables in a datasource.

Parameters **derivation_type** (*str*) – If specified, returns only variables with the specified derivation type. Valid options are ‘all’ (default), ‘custom’, ‘original’ or ‘seasonal’. See `VariableDetails` for more information.

Returns A list of the same variables as visible in Eureqa UI, including all derived variables.

Return type list of str

series_id_column_name

The name of the column that splits data into series based on its values.

series_order_column_name

The name of the column that defines the order of the data. This indicates the data is timeseries data and will sort the rows based on this column. Use DataSource.EXISTING_ROW_ORDER to use the current row number as the series order.

series_order_variable_name

The name of the variable that defines the order of the data. Same as *series_order_column_name* except properly escaped for use as a variable name.

data_splitting

class eureqa.data_splitting.**DataSplitting**(*shuffle=None, training_data_percentage=None, validation_data_percentage=None, training_selection_expression=None, validation_selection_expression=None, training_selection_expression_type=None, validation_selection_expression_type=None*)

Represents data splitting settings for the genetic algorithm. Each `Search` contains a DataSplitting object that describes how the genetic algorithm will run.

Parameters

- **shuffle** (*bool*) – Indicates whether data are shuffled or not before the training.
- **training_data_percentage** (*float*) – The percentage of data used for training.
- **validation_data_percentage** (*float*) – The percentage of data used for validation.

- **training_selection_expression** (*str*) – The expression used for selecting the training set.
- **validation_selection_expression** (*str*) – The expression used for selecting the validation set.
- **training_selection_expression_type** (*str*) – The type of expression used for selecting the training set. (‘EXPRESSION’,’VARIABLE’,None)
- **validation_selection_expression_type** (*str*) – The type of expression used for selecting the validation set. (‘EXPRESSION’,’VARIABLE’,None)

Variables

- **shuffle** (*bool*) – Indicates whether data are shuffled or not before the training.
- **training_data_percentage** (*float*) – The percentage of data used for training.
- **validation_data_percentage** (*float*) – The percentage of data used for validation.
- **training_selection_expression** (*str*) – The expression used for selecting the training set.
- **validation_selection_expression** (*str*) – The expression used for selecting the validation set.
- **training_selection_expression_type** (*str*) – The type of expression used for selecting the training set. (‘EXPRESSION’,’VARIABLE’,None)
- **validation_selection_expression_type** (*str*) – The type of expression used for selecting the validation set. (‘EXPRESSION’,’VARIABLE’,None)

error_metric

class `eureqa.error_metric.`**`ErrorMetrics`**(*mean_absolute_error=None, r2_goodness_of_fit=None, correlation_coefficient=None, maximum_absolute_error=None, signed_difference_between_lhs_and_rhs=None, area_under_roc_error=None, log_loss_error=None, rank_correlation_1_minus_r=None, mean_square_error=None, mean_squared_error_auc_hybrid=None, mean_absolute_percentage_error=None*)

Stores the value of the error of an expression as compared to a model by a variety of different types of error-evaluation methods.

Parameters

- **mean_absolute_error** (*float*) – Mean Absolute Error
- **mean_absolute_percentage_error** (*float*) – Mean Absolute Percentage Error
- **r2_goodness_of_fit** (*float*) – R^2 Goodness of Fit
- **correlation_coefficient** (*float*) – Correlation Coefficient
- **maximum_absolute_error** (*float*) – Maximum Absolute Error

- **signed_difference_between_lhs_and_rhs** (*float*) – Signed Difference Between LHS and RHS
- **area_under_roc_error** (*float*) – Area Under ROC Curve
- **log_loss_error** (*float*) – Log Loss Error
- **rank_correlation_1_minus_r** (*float*) – Rank Correlation
- **mean_square_error** (*float*) – Mean Squared Error
- **mean_squared_error_auc_hybrid** (*float*) – Mean Squared Error for Classification
- **mean_absolute_percentage_error** – Mean Absolute Percentage Error

Variables

- ***mean_absolute_error*** (*float*) – Mean Absolute Error
- ***mean_absolute_percentage_error*** (*float*) – Mean Absolute Percentage Error
- ***r2_goodness_of_fit*** (*float*) – R^2 Goodness of Fit
- ***correlation_coefficient*** (*float*) – Correlation Coefficient
- ***maximum_absolute_error*** (*float*) – Maximum Absolute Error
- ***signed_difference_between_lhs_and_rhs*** (*float*) – Signed Difference Between LHS and RHS
- ***area_under_roc_error*** (*float*) – Area Under ROC Curve
- ***log_loss_error*** (*float*) – Log Loss Error
- ***rank_correlation_1_minus_r*** (*float*) – Rank Correlation
- ***mean_square_error*** (*float*) – Mean Squared Error
- ***mean_squared_error_auc_hybrid*** (*float*) – Mean Squared Error for Classification
- ***mean_absolute_percentage_error*** – Mean Absolute Percentage Error

eureqa.error_metric.**area_under_roc_error**()

Area Under ROC Curve error metric

eureqa.error_metric.**correlation_coefficient**()

Correlation Coefficient error metric

eureqa.error_metric.**log_loss_error**()

Log Loss Error error metric

eureqa.error_metric.**maximum_absolute_error**()

Maximum Absolute Error error metric

eureqa.error_metric.**mean_absolute_error**()

Mean Absolute Error error metric

eureqa.error_metric.**mean_absolute_percentage_error**()

Mean Absolute Percentage Error error metric

eureqa.error_metric.**mean_square_error**()

Mean Squared Error error metric

eureqa.error_metric.**mean_squared_error_auc_hybrid**()

Mean Squared Error for Classification error metric `eureqa.error_metric.`**`r2_goodness_of_fit`**`()`

R^2 Goodness of Fit error metric

`eureqa.error_metric.`**`rank_correlation_1_minus_r`**`()`

Rank Correlation error metric

`eureqa.error_metric.`**`signed_difference_between_lhs_and_rhs`**`()`

Signed Difference Between LHS and RHS error metric

eureqa

class `eureqa.eureqa.`**`Eureqa`**(*url='https://rds.nutonian.com'*, *user_name=None*, *password=None*, *organization=None*, *interactive_mode=False*, *save_credentials=False*, *verify_ssl_certificate=True*, *verify_version=True*, *verbose=False*, *retries=5*, *session_key=None*, *timeout_seconds=None*, *key=None*)

Represents an interface to the Eureqa server. All interactions with the server should start from this class.

Variables ***search_templates*** (`SearchTemplates`) – Provides access to predefined search templates.

Parameters

- **url** (*str*) – The URL of the eureqa server. It should be the same URL as used to access the web UI.
- **user_name** (*str*) – The user name to login into the server. It should be the same user name as used to login into the web UI. If the user name is not provided and the interactive mode is enabled, the user name will be requested during the script execution.
- **password** (*str*) – The password. If the password is not provided and the interactive mode is enabled, the password will be requested during the script execution.
- **organization** (*str*) – The name of the organization. All request to API will be executed in the context of this organization. If the organization name is not provided and the user is assigned to only one organization, then that organization will be used by default.
- **interactive_mode** (*bool*) – If set to True, enables interactive mode. In the interactive mode the script will request user name, password, and potentially other information if it is not provided or incorrect. If set to False (default), throws an exception if a login or password is incorrect, or if the two factor authentication is enabled. This is the default behaviour which prevents scripts from indefinitely waiting for the user input if they are executed automatically by a CRON job or a Windows Scheduler task.
- **save_credentials** (*bool*) – If set to True, saves user name and password to the '.eureqa_passwd' in the user directory. If after that any script on the same machine is trying to connect to the same server and does not provide credentials, it reuses the saved credentials. It does not save a temporary password when the two-factor authentication is enabled. When used with the two-factor authentication, the interactive_mode parameter should also be enabled.
- **verify_ssl_certificate** (*bool*) – If set to False will not verify SSL certificate authenticity while connecting to Eureqa.
- **verify_version** (*bool*) – If set to False will allow to use Python API library with incompatible version of Eureqa. Should only be used for the diagnostic purpose.
- **verbose** (*bool*) – If set to True will print to the console the detailed information for each request to the server. Should only be used for the diagnostic purpose.

- **retries** (*int*) – The number of attempts to establish a session before an Exception is raised
- **session_key** (*str*) – The session identifier.
- **timeout_seconds** (*int*) – The HTTP connection and transmission timeout. Any request to Eureqa API will abort with an exception if it takes more time, than set in the timeout, to either connect to the server or to receive a next data package.
- **key** (*str*) – Authentication key. Provide either this field or *password*.

Raises Exception – If the authentication fails or cannot be completed.

compute_error_metrics(*datasource*, *target_variable*, *model_expression*, *template_search=None*, *variable_options=[]*, *row_weight=None*, *row_weight_type=None*)

Compute the *ErrorMetrics* for the specified model, against the specified target_variable

Parameters

- **datasource** (DataSource) – DataSource to compute error against
- **target_variable** (*str*) – Variable (or expression) to compare the model to
- **model_expression** (*str*) – Model whose error is to be computed
- **template_search** (Search) – If specified, inherit variable options from the specified search. Values specified in :variable_options: take precedence over values in this search; use it for finer-grained control instead of or on top of this argument.
- **variable_options** (VariableOptionsDict) – Override any default behavior for the specified variables. If the data contains nulls and no null-handling policy is specified, this method will return an error. A list of *VariableOptions* may also be provided.
- **row_weight** (*str*) – Expression to compute the weight of a row (how much that row contributes to the computed error)
- **row_weight_type** (*str*) – The type of expression to use to compute row weight (uniform, target_frequency, variable, or custom_expr)

Returns The computed error metrics

Return type *ErrorMetrics*

create_analysis(*name*, *description=None*)

Creates an analysis.

Parameters

- **name** (*str*) – The analysis name. It will be used as the Analysis title.
- **description** (*str*) – The analysis description.

Returns An *Analysis* object that represents a newly created analysis on the server.

Return type *Analysis*

create_analysis_template(*name*, *description*, *parameters*, *icon=None*, *icon_filepath=None*)

Create a new Analysis Template on the Eureqa server.

Parameters

- **name** (*str*) – The analysis template's name. Will be used to identify the template.
- **description** (*str*) – The analysis template's description. Will be used where more space is available for an expanded description.

- **parameters** (Parameters) – Object describing the parameters that a user must fill in via the UI in order to specify the template's behavior
- **icon** (*str*) – The url of an icon to use in the UI for this analysis.
- **icon_filepath** (*str*) – The local path to an icon to upload for use in the UI for this analysis. Overrides icon argument.

Returns An *AnalysisTemplate* object representing the template on the server

Return type *AnalysisTemplate*

create_data_source (*name*, *file_or_path*, *series_id_column_name=None*, *series_order_column_name=None*)

Creates a new data source on the server.

Uploads raw data and and creates a new datasource.

Parameters

- **name** (*str*) – A name for the new data source.
- **file_or_path** (*str*) – A path to a local CSV file with data for the data source. It can be either an absolute path or path relative to the current working directory. Alternatively, a Python file-like object.
- **series_id_column_name** (*str*) – The name of the column that splits data into series based on its values.
- **series_order_column_name** (*str*) – The name of the column that defines the order of the data. This indicates the data is timeseries data and will sort the rows based on this column. Use DataSource.EXISTING_ROW_ORDER to use the current row number as the series order

Returns A *DataSource* object that represents a newly created data source on the server.

Return type *DataSource*

evaluate_expression (*datasource*, *expressions*, *template_search=None*, *variable_options=[]*)

Evaluates the provided expression against the specified datasource. Returns the value of the evaluated computation.

Example:

values = eureqa.evaluate_expression(['x','y','x^2']) # where

values['x'] –> [1,2,3,4]

values['y'] –> [5,6,7,8]

values['x^2'] –> [1,4,9,16]

data = pandas.DataFrame(values) # convert to pandas.DataFrame

Parameters

- **datasource** (DataSource) – DataSource to perform the computation against
- **expressions** (*str*) – If only one expression is to be evaluated, that expression. If multiple expressions are to be evaluated, a list of those expressions.
- **template_search** (Search) – If specified, inherit variable options from the specified search. Values specified in :variable_options: take precedence over values in this search; use it for finer-grained control instead of or on top of this argument.

- **variable_options** (VariableOptionsDict) – Override default variable options directly for particular variables. Set interpretation of NaN values, outliers, etc. default behavior is to make no changes to the original data. By default, missing values are not filled; missing values in input data may result in missing values in corresponding computed values. A list of *VariableOptions* may also be provided.

evaluate_models(*data*, *solutions*, *expressions=[]*, *include_data=False*, *calculate_error_metrics=False*, *calculate_confidence_intervals=False*, *num_future_rows=None*)

Evaluates the provided solutions and model strings on the specified data source.

Parameters

- **data** (*str*) – A path to the file, Python file-like object, or an existing Eureqa datasource on which the models have to be evaluated.
- **solutions** (*list*) – The list of *Solution* objects. which have to be evaluated against the datasource.
- **expressions** (*list*) – The optional list of expressions (could be variables) to include in the output data (optional).
- **include_data** (*bool*) – The boolean flag which will indicate whether all data columns will be included into the response.
- **calculate_error_metrics** (*bool*) – The boolean flag which will indicate whether the error metrics have to be calculated.
- **calculate_confidence_intervals** (*bool*) – The boolean flag which will indicate whether the confidence intervals have to be calculated
- **num_future_rows** (*int*) – The parameter which allows override the number of future rows on which the solution and expressions are evaluated. If no value provided for this parameter the models are evaluated into the future as far as possible.

:rtype *ModelEvaluation*

get_all_analysis_templates()

Get a list of all Analysis Template objects currently available to this connection

Return type list of *AnalysisTemplate*

get_all_data_sources()

Get all data sources from the server

Returns A list of *DataSource* objects for all data sources within the organization.

Return type list of *DataSource*

get_analyses()

Return the list of all analyses from the server.

Return type list of *Analysis*

get_analysis(*analysis_id*)

Return a specific analysis from the server, by id

Parameters **analysis_id** (*str*) – The id of the analysis to return

Return type *Analysis*

get_data_source(*data_source_name*)

Get a data source by its name.

Searches on the server for a data source given its name.

Parameters **`data_source_name`** (*`str`*) – The name of the data source.

Returns A *`DataSource`* object if such data source exists, otherwise None.

Return type *DataSource*

`get_data_source_by_id`(*data_source_id*)

Get a data source by its id.

Searches on the server for a data source given its id.

Parameters **`data_source_id`** (*`str`*) – The ID of the data source.

Returns A *`DataSource`* object if such data source exists, otherwise None.

Return type *DataSource*

`search_templates`

Return all Search Templates available to the current connection

Return type *SearchTemplates*

html

button

class `eureqa.html.button.`**`Button`**(*title*)

BETA An HTML button inside an HtmlCard

Parameters **`title`** (*`str`*) – Title of the card

class **`Events`**

Constants that identify the types of events that can trigger an action

`Button.`**`add_action`**(*card_action*, *args=None*, *kwargs=None*, *event='click'*)

Add an action to this button, to be performed when the specified event happens

Parameters

- **`card_action`** (*`instancemethod`*) – Method on an analysis-card instance to invoke. For example, *card.replace*.
- **`args`** (*`tuple`*) – Arguments to the *card_action* method
- **`kwargs`** (*`dict`*) – Keyword arguments to the *card_action* method
- **`event`** (*`str`*) – Event that triggers this action

`Button.`**`add_replace_action`**(*target_card*, *replacement_card*, *event='click'*)

Add a 'Replace' action to the specified card. Equivalent to *add_action(target_card.replace, (replacement_card,), event=event)*.

Parameters

- **`target_card`** (*`eureqa.analysis_card.AnalysisCard`*) – Card to be replaced
- **`replacement_card`** (*`eureqa.analysis_card.AnalysisCard`*) – Card to replace *target_card* with
- **`event`** (`Events`) – Event to trigger the replacement

Button.**to_html**(*html_tag='a'*)

Render this button as an HTML tag

For example: to_html('Go!', html_tag='button', style='color: blue;') would return "<button style='color: blue;' (...)>Go!</button>"

Parameters

- **html_tag** (*str*) – Tag name. For example, 'a', 'input', 'button', 'div'
- ****kwargs** – Tag parameters. For example, "style='color: blue;'"

Returns HTML string representation of this button

math_block

class eureqa.math_block.**MathBlock**

Represents a building block of a mathematical model. Created automatically during the construction of a *SearchSettings* object.

MathBlocks can be enabled or disabled to specify whether their corresponding mathematical operations are allowed in a *Search*'s solutions. They should be accessed through the *eureqa.search.Search.math_blocks* or *eureqa.search_settings.SearchSettings.math_blocks* properties. For a list of available Math Blocks, view the properties of *MathBlockSet*.

complexity

MathBlock's complexity (settable)

disable()

Don't allow this MathBlock to be used for modeling

enable(*complexity=None*)

Allow this MathBlock to be used for modeling

Parameters **complexity** (*int*) – The level of complexity of the MathBlock. If you do not specify a complexity, the default complexity will be used.

enabled

Can this MathBlock be used for modeling?

name

MathBlock's name (read-only)

math_block_set

class eureqa.math_block_set.**MathBlockSet**

Contains a set of *MathBlock* objects that represents a set of all available mathematical operations. Created automatically during the construction of a *SearchSettings*.

Every *Search* and *SearchSettings* contains a MathBlockSet in *eureqa.search.Search.math_blocks* or *eureqa.search_settings.SearchSettings.math_blocks*. Use the properties of this variable to access individual *MathBlock* objects. The enabled *MathBlock* objects of *eureqa.search.Search.math_blocks* determine the operations allowed in solutions to the *Search*.

abs

Absolute Value math block

Return type *MathBlock*

acos

Arccosine math block

Return type *MathBlock*

acosh

Inverse Hyperbolic Cosine math block

Return type *MathBlock*

add

Addition math block

Return type *MathBlock*

and_op

Logical And math block

Return type *MathBlock*

asin

Arcsine math block

Return type *MathBlock*

asinh

Inverse Hyperbolic Sine math block

Return type *MathBlock*

atan

Arctangent math block

Return type *MathBlock*

atanh

Inverse Hyperbolic Tangent math block

Return type *MathBlock*

ceiling

Ceiling math block

Return type *MathBlock*

complementary_error

Complementary Error Function math block

Return type *MathBlock*

const

Returns math block for Constant

Return type *MathBlock*

cos

Cosine math block

Return type *MathBlock*

cosh

Hyperbolic Cosine math block

Return type *MathBlock*

`delay`
Delayed Variable math block

Return type *MathBlock*

`div`
Division math block

Return type *MathBlock*

`equal`
Equal-To math block

Return type *MathBlock*

`error`
Error Function math block

Return type *MathBlock*

`exp`
Exponential math block

Return type *MathBlock*

`fact`
Factorial math block

Return type *MathBlock*

`floor`
Floor math block

Return type *MathBlock*

`gauss`
Gaussian Function math block

Return type *MathBlock*

`greater`
Greater-Than math block

Return type *MathBlock*

`greater_equal`
Greater-Than-Or-Equal math block

Return type *MathBlock*

`if_op`
If-Then-Else math block

Return type *MathBlock*

`int_const`
Integer Constant math block

Return type *MathBlock*

`less`
Less-Than math block

Return type *MathBlock*

`less_equal`
Less-Than-Or-Equal math block

Return type *MathBlock*

log
Natural Logarithm math block

Return type *MathBlock*

logistic
Logistic Function math block

Return type *MathBlock*

max
Maximum math block

Return type *MathBlock*

min
Minimum math block

Return type *MathBlock*

mod
Modulo math block

Return type *MathBlock*

mult
Multiplication math block

Return type *MathBlock*

neg
Negation math block

Return type *MathBlock*

not_op
Logical Not math block

Return type *MathBlock*

or_op
Logical Or math block

Return type *MathBlock*

pow
Power math block

Return type *MathBlock*

round
Round math block

Return type *MathBlock*

sign
Sign Function math block

Return type *MathBlock*

simple_moving_average
Simple Moving Average math block

Return type *MathBlock*

simple_moving_median
Simple Moving Median math block

Return type *MathBlock*

sin
Sine math block

Return type *MathBlock*

sinh
Hyperbolic Sine math block

Return type *MathBlock*

sqrt
Square Root math block

Return type *MathBlock*

step
Step Function math block

Return type *MathBlock*

sub
Subtraction math block

Return type *MathBlock*

tan
Tangent math block

Return type *MathBlock*

tanh
Hyperbolic Tangent math block

Return type *MathBlock*

two_args_atan
Two-Argument Arctangent math block

Return type *MathBlock*

var
Input Variable math block

Return type *MathBlock*

weighted_moving_average
Weighted Moving Average math block

Return type *MathBlock*

xor
Logical Xor math block

Return type *MathBlock*

model_evaluation

class `eureqa.model_evaluation.`**`ModelEvaluation`**
Represents the result of evaluting solutions and expressions.

Variables

- **data** (*dict*) – The dictionary, where keys are expressions and variable names, and values are columns of data.
- **num_future_rows** (*int*) – The number of future rows that are included in each data series.
- **frame** (*pandas.PandFrame*) – The Pandas frame, where column names are expressions and variable names.
- **error_metrics** (*dict*) – The dictionary where the keys are model strings for the provided solutions and expressions, and values are *ErrorMetrics* objects.

search

class eureqa.search.**Search**

Represents a search on the server. It should be created using the *create_search()* method. A search can be run using the *submit()* method. Once a search has ran, solutions can be retreived from one of several methods below that return a *Solution*, including *get_best_solution()* to get the best solution and *get_solutions()* to get a list of all solutions.

Many of the variables will be set based on the *SearchSettings* used when *create_search()* is called. However, these variables can still be changed after the Search's creation.

Variables

- **name** (*str*) – The name of the search.
- ***math_blocks*** (MathBlockSet) – The MathBlockSet which represents mathematical operations allowed to be used in the search's solutions. See the usage section of *SearchSettings* to see examples of how to change it.
- ***data_splitting*** (DataSplitting) – A DataSplitting object that holds data splitting settings for the search algorithm.
- ***error_metric*** (*str*) – The error metric that will be used for this search. Choose from the error metrics in *error_metric*.
- **maximum_history_absolute_rows** (*int*) – The maximum number of rows that can be used in range based functions.
- **prior_solutions** (*list*) – The list of prior solutions.
- ***row_weight*** (*str*) – The row weight expression.
- ***target_expression*** (*str*) – The target expression.
- ***variable_options*** (VariableOptionsDict) – Override default behavior for the specified variables.

create_solution(*solution_string*, *use_all_data=False*)

Creates a custom solution for the search. Use this if you want to compute error metrics and other statistics of a specified expression. It is also useful to compare how well a known model does against one found by Eureqa

Parameters

- **solution_string** (*string*) – the right hand side of the expression.
- **use_all_data** (*bool*) – whether to use all data or just validation data when calculating the metrics for the solution.

Return type *Solution*

delete()

Deletes the search from the server.

Raises Exception – search is already deleted.

evaluate_expression(*expressions*)

Deprecated. Use `evaluate_expression()` instead

Parameters expressions (`str`) – Deprecated. Do not use.

get_best_solution()

Retrieves from the server the best solution found so far.

Return type *Solution*

get_data_source()

Retrieves from the server the data source information for this search.

Return type *DataSource*

get_most_accurate_solution()

Retrieves from the server the most accurate solution found so far.

Return type *Solution*

get_solutions()

Retrieves from the server the list of solutions found so far.

This method can be called while the search is running to check what searches are already found and make a decision whether continue the search.

Return type list of `Solution` objects.

is_running

Indicates if the search currently running.

Return type bool

math_blocks

The MathBlockSet which represents mathematical operations allowed in solutions to the Search.

rename(*new_search_name*)

Change search display name.

Parameters new_search_name (`std`) – New search name.

stop()

Stops running the search.

submit(*time_seconds*)

Submit the search to the server to run for the specified amount of time.

This method does not guarantee to start the search immediately. The search can be queued for some time before it will start producing any results.

Parameters time_seconds (`int`) – The maximum amount of time to run the search. The server will stop running the search once the running time will reach this limit.

wait_until_done(*show_progress=False*, *poll_seconds=5*, *print_callback=None*)

Waits until the search stops running.

Parameters

- **show_progress** (`bool`) – whether to print the search progress while waiting.

- **poll_seconds** (*int*) – number of seconds to wait between checking progress.
- **print_callback** (*function*) – method to invoke to print the progress (sys.stdout.write by default).

search_settings

class eureqa.search_settings.**SearchSettings**

A set of settings that describe how a *Search* will run. To create an instance of this class, call a method from *search_templates()*. After than, the settings can then be changed from their default values. Once the desired settings are attained, pass the SearchSettings instance into *create_search()* to create a *Search* with those settings.

Variables

- **name** (*str*) – The name of the search.
- **target_variable** (*str*) – The target variable.
- **input_variables** (*list*) – The list of input variables.
- ***math_blocks*** (MathBlockSet) – The set of mathematical operations allowed in the search's solution.
- ***data_splitting*** (DataSplitting) – The data splitting settings for the search algorithm.
- ***error_metric*** (*str*) – The error metric that will be used for this search. Choose from the error metrics in *error_metric*.
- **maximum_history_absolute_rows** (*int*) – The maximum number of rows that can be used in range based functions.
- **prior_solutions** (*list*) – The list of prior solutions.
- ***row_weight*** (*str*) – The row weight expression.
- ***row_weight_type*** (*str*) – The row weight type expression.
- ***target_expression*** (*str*) – The target expression.
- ***variable_options*** (VariableOptionsDict) – Override default behavior for the specified variables.
- ***default_min_delay*** (*int*) – the default minimum number of rows each variable will be delayed in the model, if not overridden by a specific variable option.

Example:

```
# create a new SearchSettings object with the default settings for a numeric search
settings = e.search_templates.numeric("Sample Model", target_variable, variables - {target_varia

# enable the pow building block and set its complexity to 3
settings.math_blocks.pow.enable(complexity=3)
# disable the pow building block in searches
settings.math_blocks.pow.disable()
# set the complexity of log to 5 (do not change enabled or disabled)
settings.math_blocks.log.complexity = 5

# enable a list of blocks with default complexity
for block in [settings.math_blocks.pow, settings.math_blocks.exp]:
    block.enable()

# print a list of enabled blocks w/ complexity
for block in settings.math_blocks:
    if block.enabled:
         print block.name + ": " + str(block.complexity)

# select a different error metric
settings.error_metric = error_metric.mean_square_error()

# create a new Search with these settings
search = my_data_source.create_search(settings)
```

`default_min_delay`
The default minimum number of rows each variable will be delayed in the model, if not overridden by a specific variable option. This value is the number of rows the resulting model can predict into the future.

`math_blocks`
The *`MathBlockSet`* which represents mathematical operations allowed in solutions to the search.

`row_weight`
The relative weight that each row is given when computing the error metric values for a model.

See description of *`row_weight_type`* for restrictions on the value of row_weight.

`row_weight_type`
The method to use for weighting the error metric calculation.

Options are 'uniform' (default), 'target_frequency', 'variable', and 'custom_expr'. If set to 'variable', *`row_weight`* must be the name of a variable. If set to 'custom_expr', *`row_weight`* may be an arbitrary expression.

`target_expression`
The target expression to optimize.

Only set if a custom expression is required. If set, the resulting search will be treated as an "Advanced" search in the UI; the variable chooser will be replaced with an expression editor.

search_templates

class `eureqa.search_templates.`**`SearchTemplates`**
Provides a set of search settings for well known search scenarios. Created automatically when you create a *`Eureqa`*. Can be accessed through *`search_templates`*.

Each function of this class represents a type of search that there is a template for. The function will return a *`SearchSettings`* containing the suggested settings for that type of search.

`classification`(*name*, *target_variable*, *input_variables*, *variable_options=None*, *datasource=None*)
The classification search settings template.

Parameters

- **`name`** (*`str`*) – The search name.
- **`target_variable`** (*`str`*) – The target variable.
- **`input_variables`** (*`list`*) – The list (str) of input variables.

- **variable_options** (*list*) – An optional list of *VariableOptions* to include in the search
- **datasource** (DataSource) – The datasource this template will be applied in.

Return type *SearchSettings*

numeric (*name*, *target_variable*, *input_variables*, *variable_options=None*, *datasource=None*)

The numeric search settings template.

Parameters

- **name** (*str*) – The search name.
- **target_variable** (*str*) – The target variable.
- **input_variables** (*list*) – The list (str) of input variables.
- **variable_options** (*list*) – An optional list of *VariableOptions* to include in the search
- **datasource** (DataSource) – The datasource this template will be applied in.

Return type *SearchSettings*

time_series (*name*, *target_variable*, *input_variables*, *min_delay=1*, *data_custom_history_fraction=0.1*, *max_delays_per_variable=None*, *variable_options=None*, *datasource=None*)

The time series search settings template.

Parameters

- **name** (*str*) – The search name.
- **target_variable** (*str*) – The target variable.
- **input_variables** (*list*) – The list (str) of input variables.
- **min_delay** (*int*) – Optionally specify the minimum number of rows used in the range functions.
- **data_custom_history_fraction** (*float*) – Optionally specify the percentage of the data to be withheld from history blocks. Specifies the maximum possible delay for a history block.
- **max_delays_per_variable** (*int*) – Optionally overrides data_custom_history_fraction to directly set the maximum possible delay for a history block.
- **variable_options** (*list*) – An optional list of *VariableOptions* to include in the search
- **datasource** (DataSource) – The datasource this template will be applied in.

Return type *SearchSettings*

time_series_classification (*name*, *target_variable*, *input_variables*, *min_delay=1*, *data_custom_history_fraction=0.1*, *max_delays_per_variable=None*, *variable_options=None*, *datasource=None*)

The time series classification search settings template.

Parameters

- **name** (*str*) – The search name.
- **target_variable** (*str*) – The target variable.

- **input_variables** (*list*) – The list (str) of input variables.
- **min_delay** (*int*) – Optionally specify the number of rows the model should predict into the future (i.e. the default minimum number of rows each variable will be delayed in the model).
- **data_custom_history_fraction** (*float*) – Optionally specify the percentage of the data to be withheld from history blocks. Specifies the maximum possible delay for a history block.
- **max_delays_per_variable** (*int*) – Optionally overrides data_custom_history_fraction to directly set the maximum possible delay for a history block.
- **variable_options** (*list*) – An optional list of *VariableOptions* to include in the search
- **datasource** (DataSource) – The datasource this template will be applied in.

Return type *SearchSettings*

solution

class eureqa.solution.**Solution**

Represents one of the solutions found by the server for a particular search. Can be obtained from one of the methods in *Search*. Some solutions may be available even if the search has not finished running.

Variables

- **target** (*str*) – The target variable.
- **model** (*str*) – The model expression.
- **complexity** (*int*) – The model complexity based on the complexity weights.
- **is_best** (*bool*) – An indicator whether the solution is considered to be the best based on its complexity and precision. The system can pick only one solution as best.
- **is_most_accurate** (*bool*) – An indicator whether the solution is the most accurate for the optimized error metric. The system can pick only one solution as most accurate.
- **optimized_error_metric** (*str*) – The error metric for which the solution was optimized. It is one of the error metrics from *error_metric*.
- **optimized_error_metric_value** (*float*) – The value of the optimized error metric.
- **search** (Search) – The search object to which this solution belongs.

get_all_series_error_metrics(*data_split='all'*)

Returns the metric values for each series in the datasource.

Parameters **data_split** (*str*) – For internal use only.

Returns list of *ErrorMetrics* objects.

Return type list

get_error_metric_value(*name*)

Returns the value for the specified error metric.

Parameters **name** (*str*) – One of the error metrics from *error_metric*.

Return type float

get_single_series_error_metrics (*series=None*, *data_split='all'*, *series_index=None*)

Returns the metric values for the specified series in the datasource.

Parameters

- **series** (*int, str*) – The series to compute the metrics on. If series is an integer, returns the error metrics for the series with that series index. Otherwise, returns the error metrics for the series with the specified series id value.
- **data_split** (*str*) – For internal use only.
- **series_index** (*int*) – Deprecated. Only kept for backwards compatibility. Do not use.

Return type *ErrorMetrics*

variable_details

class eureqa.variable_details.**VariableDetails**

Holds information about a variable. Obtained through the *get_variable_details()* method of a *DataSource* . Use this class to find mathematic properties of a variable and to change a variable's name.

Variables

- **name** (*str*) – The name of the variable.
- **expression** (*str*) – The expression used to create the variable if it is a custom or seasonal variable
- **min_value** (*float/datetime*) – The smallest value. Will be a datetime if datatype is 'datetime'.
- **max_value** (*float/datetime*) – The largest value. Will be a datetime if datatype is 'datetime'.
- **mean_value** (*float/datetime*) – The mean value. Will be a datetime if datatype is 'datetime'.
- **standard_deviation** (*float*) – The standard deviation of the values.
- **distinct_values** (*int*) – The number of distinct values.
- **missing_values** (*int*) – The number of missing values.
- **total_rows** (*int*) – The total rows occupied by the variable.
- **datatype** (*str*) – Whether it is a binary or numeric variable
- **num_zeroes** (*int*) – The number of zero values.
- **num_ones** (*int*) – The number of one values.
- **derivation_type** (*str*) – From where this variable originated. Valid values are 'original' for variables from the original dataset, 'custom' for custom derived variables, or 'seasonal' for seasonal variables.
- **seasonal_target_variable** (*str*) – For seasonal variables, what variable was the target used to derive the seasonal trend. None for other types of variables
- **seasonal_period** (*str*) – For seasonal variables, over what period was the seasonal trend derived. None for other types of variables

- **data_source** (`DataSource`) – The DataSource the VariableDetails belongs to

variable_options

class `eureqa.variable_options.`**`VariableOptions`**(*name=None*, *missing_value_policy='column_iqm'*, *remove_outliers_enabled=False*, *outlier_threshold=None*, *normalize_enabled=False*, *normalization_offset=None*, *normalization_scale=None*, *min_delay=None*)

Represents a set of settings for a variable that can be included into a search. Overrides any default behavior for the specified variable.

Parameters

- **name** (*str*) – The name of the variable. It is the same as a column name in the list of data source columns.
- **missing_value_policy** (*str*) – A policy name for the processing of missing values. One of the values from missing_value_policies module can be used.
- **remove_outliers_enabled** (*bool*) – Enables removal of rows which contain outlier.
- **outlier_threshold** (*float*) – Controls the threshold for whether a point is considered an outlier. A higher value means points must be further from the mean to be considered outliers. Default value is 2.0.
- **normalize_enabled** (*bool*) – Enables normalization of the variable. If True, each value tranformed via the expression (value - normalization_offset) / normalization_scale
- **normalization_offset** (*int*) – A constant offset value to subtract from each value. Defaults to 0 if not specified.
- **normalization_scale** (*int*) – A constant factor to divide each value after subtracting the normalization offset. Defaults to 1 if not specified.
- **min_delay** (*int*) – The number of time increments into the future to predict with a time series search.

Variables

- **name** (*str*) – The name of the variable. It is the same as a column name in the list of data source columns.
- **missing_values_enabled** (*bool*) – Enables additional processing of missing values.
- **missing_value_policy** (*str*) – A policy name for the processing of missing values. One of the values from missing_value_policies module can be used. missing_value_policies.column_mean is used if the additional processing of missing values is enabled but no policy name provided.
- **outlier_threshold** (*float*) – The threshold at which a point is considered an outlier and will be removed.

variable_options_dict

class `eureqa.variable_options_dict.`**`VariableOptionsDict`**`(...)`

Create a dictionary of *`VariableOptions`* objects in which the key is the *`VariableOptions`* name.

This class is used primarily to hold variable parameter overrides for *`Eureqa`* and *`Search`* .

`add``(`*variable_options*`)`

Add an existing *`VariableOptions`* to this dictionary.

Parameters **`variable_options`** (`VariableOptions`) – *`VariableOptions`* object to add

versions

Python Module Index

Index