

Analyzer Widget Start Guide.

version 1.0

Eric Papenhausen (epapenha@akaikaeru.com)
Klaus Mueller (mueller@akaikaeru.com)
<https://akaikaeru.com>

August 18, 2021

Introduction

The Analyzer Widget is a JupyterLab extension which contains the Akai Kaeru pattern discovery engine and visual explorer. It finds and visualizes statistically significant and temporally consistent patterns. For a data set consisting of stocks, for example, these patterns can represent patterns of stock behavior (e.g. price/book ratio < 1) that are associated with unusually high or low returns. In the case of a data set with a temporal component (e.g. stocks in the NYSE over time), the patterns found are temporally consistent (i.e. they are consistently high/low over time).

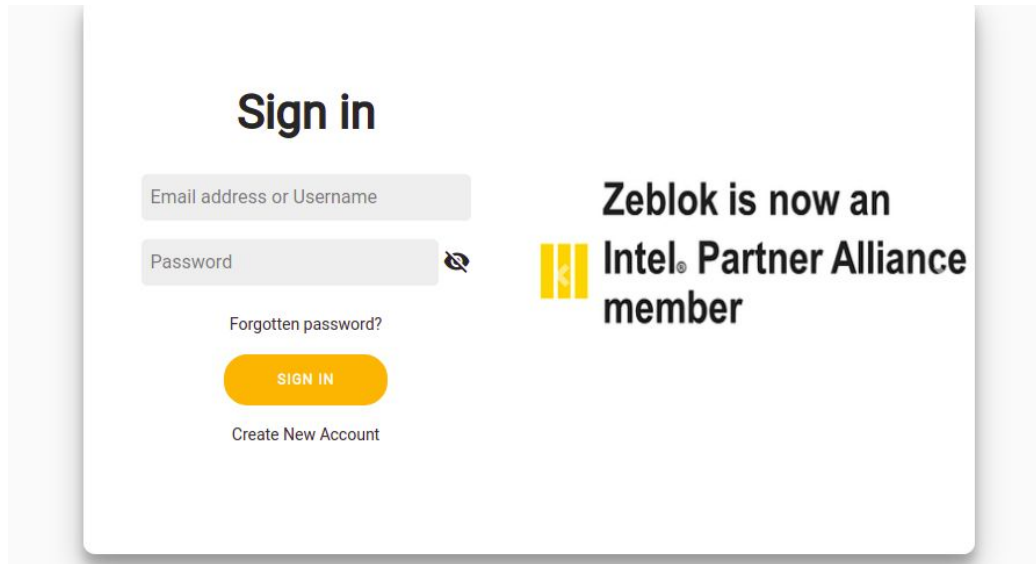
Contents

1	Launching a Workstation	2
2	Overview of Features	3
2.1	Getting it to Work	3
2.2	Pattern Mining	4
2.2.1	Numerical	4
2.2.2	Binary	4
2.3	Analyzer Widget Interface	4
3	API	10
3.1	AKMiner	10
3.2	render	11
3.3	get_pattern_json	11

I Launching a Workstation

The following instructions will help guide you in logging into the Analyzer Widget workstation through the Zeblok computational AI micro-cloud.

1. Login into the Zeblok computational AI platform at <https://app.zbl-aws.zeblok.com/>.




2. This will bring up the home page. Click on the "Start" button to start the workstation. This may take a few minutes.

Zeblok Ai-WorkStation

The Zeblok AI Workstation provides a single unified pre-configured environment for data scientists to access datasets, GPU-powered AI/ML frameworks, languages, and tools, with AI algorithms available from our AlgoStore. The Workstation includes one GPU, one vCPU and 64 GB of storage, but scales easily as required. For more demanding workloads, the Zeblok HPC Workstation adds Slurm-as-a-Service and parallel processing, using worker nodes, by enabling the deployment of 4 or more GPUs.

The Zeblok Ai-WorkStation is built upon a Jupyter notebook, an interactive coding environment that lets you combine code with equations, visualizations, rich text, and other media. We make it easy to explore data and coding concepts and collaborate with other people on data science projects.

[Spawn Ai-WorkStation](#) [Ai-Data Lake](#)

Workstation Demo	stopped	Created 6 minutes ago	Start Delete Open	Spawned from AI-Rover
	Machine Type 1 GPU / 1 vCPU			

3. Click "Open" to open the JupyterLab interface.

Zeblok Ai-WorkStation

The Zeblok AI Workstation provides a single unified pre-configured environment for data scientists to access datasets, GPU-powered AI/ML frameworks, languages, and tools, with AI algorithms available from our AlgoStore. The Workstation includes one GPU, one vCPU and 64 GB of storage, but scales easily as required. For more demanding workloads, the Zeblok HPC Workstation adds Slurm-as-a-Service and parallel processing, using worker nodes, by enabling the deployment of 4 or more GPUs.

The Zeblok AI-WorkStation is built upon a Jupyter notebook, an interactive coding environment that lets you combine code with equations, visualizations, rich text, and other media. We make it easy to explore data and coding concepts and collaborate with other people on data science projects.

Spawn AI-WorkStation AI-Data Lake

Workstation
Demo

● running

Created 10 minutes
ago

Stop

Delete

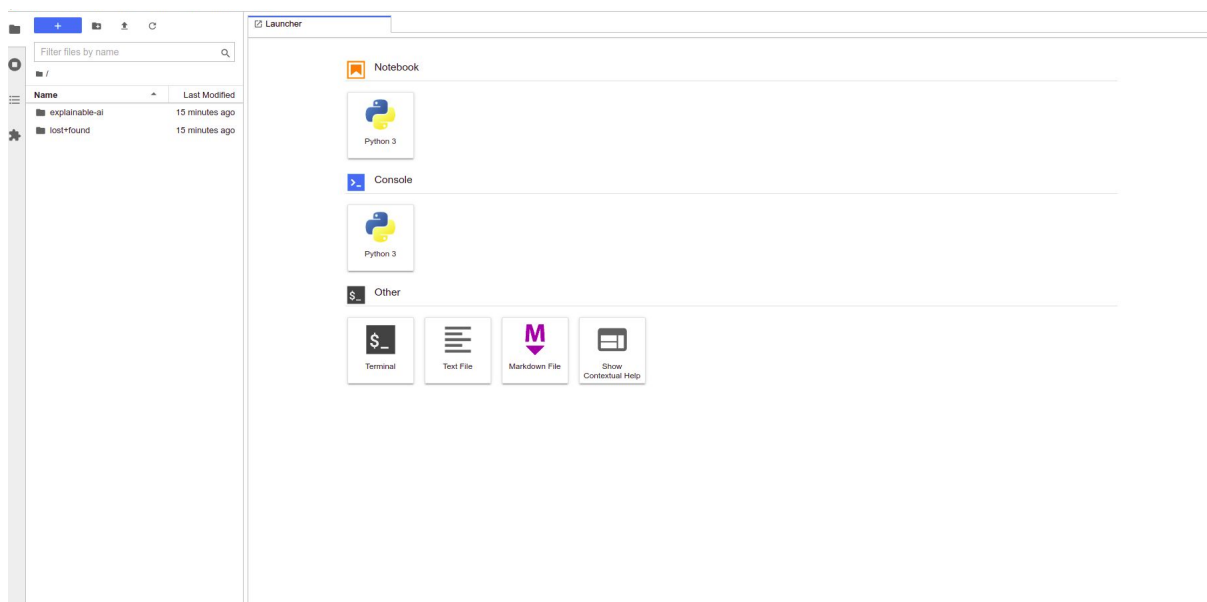
Open

Spawned from
AI-Rover

Machine Type 1 GPU / 1
vCPU



- This will bring you to the JupyterLab interface. Within the *explainable-ai* directory you will find some demo notebooks which walk through some use cases of the Analyzer Widget.



2 Overview of Features

2.1 Getting it to Work

There are two sample Jupyter notebooks that show how to operate the analyzer widget. These are in the notebooks folder in the analyzer_widget directory. The main algorithm which runs the pattern miner and visual explorer is the AKMiner algorithm. This can be imported through:

```
from analyzer_widget import AKMiner
```

This is needed for the multi-variate pattern miner. It is initiated by calling:

```
out = AKMiner(df, target)
```

where `df` is a pandas dataframe and `target` is the target attribute of interest. The analyzer widget visual interface is then rendered by calling:

```
out.render()
```

2.2 Pattern Mining

The Analyzer widget finds and visualizes groups of data points called patterns. Patterns are hypercubes of the form 'attribute' (<,>=) 'value' (e.g. price/book < 1 and sector = Financial). Two types of pattern mining are supported within the analyzer widget – numeric and binary. Numeric pattern mining is used when the target variable is continuous (e.g. returns). Binary pattern mining is used when the target variable is a 0 or 1 indicator (e.g. defaulting on a loan).

In both cases, patterns are found to be "interesting" when the pattern satisfies the following conditions:

1. The target variable within the pattern is statistically significantly higher or lower than the rest of the data set.
2. The effect size is large (i.e. higher than some predefined threshold)
3. The size of the pattern is large (i.e. higher than some predefined threshold)

2.2.1 Numerical

The statistical test performed with numeric pattern mining is the non-parametric Mann-Whitney U test. This is a non-parametric test and so it makes no assumption about the distribution of the target variable. The effect size used is the common language effect size. This is a measure of the probability of an item selected from the pattern being higher / lower than an item selected from outside the pattern. For example, an effect size of 0.8 would indicate that if we were to randomly select one point within the pattern and one point outside the pattern, 80% of the time the point within the pattern will be higher. Negative effect size measures indicate the opposite relationship (e.g. -0.8 indicates that 80% of the time the point within the pattern is lower).

2.2.2 Binary

For pattern mining with a binary target variable, the target attribute is assumed to consist of 1's and 0's. The statistical test performed with binary pattern mining is the chi squared test for independence. This determines if the proportion of 1's within the pattern is statistically higher than the overall data set. The effect size used is the odds ratio. An odds ratio of 2 indicates that the odds of the target being a 1 within the pattern increase by a factor of 2x compared to the overall data set.

2.3 Analyzer Widget Interface

Figure 1 shows the starting state of the Analyzer widget interface. The main plot (figure 1(a)) shows a scatter plot of volatility v.s. return. In this example, the target variable is return. The attribute of interest (AOI) is volatility. The AOI is a user selected feature and can be changed by clicking on a feature in the Feature Importance plot (b). This will change the x-axis of the main plot to the selected feature. By clicking on the switch in figure 1(e) the main plot will switch from a scatter plot to the group bubble chart.

Figure 2 shows the interface after switching to the group bubble chart. The colored circles in the main plot represent groups of data points that are similar in some way and have an unusually low/high distribution of the target variable. The green (red) circles indicate that return is higher (lower) within these groups. The position of the circles is based on the median target and AOI values within the group. The opacity of the circles indicates how important the AOI is in defining the groups. In figure 2, for example, the circles at the extreme ends of volatility are opaque; indicating that volatility is an important feature for these groups. Conversely, the

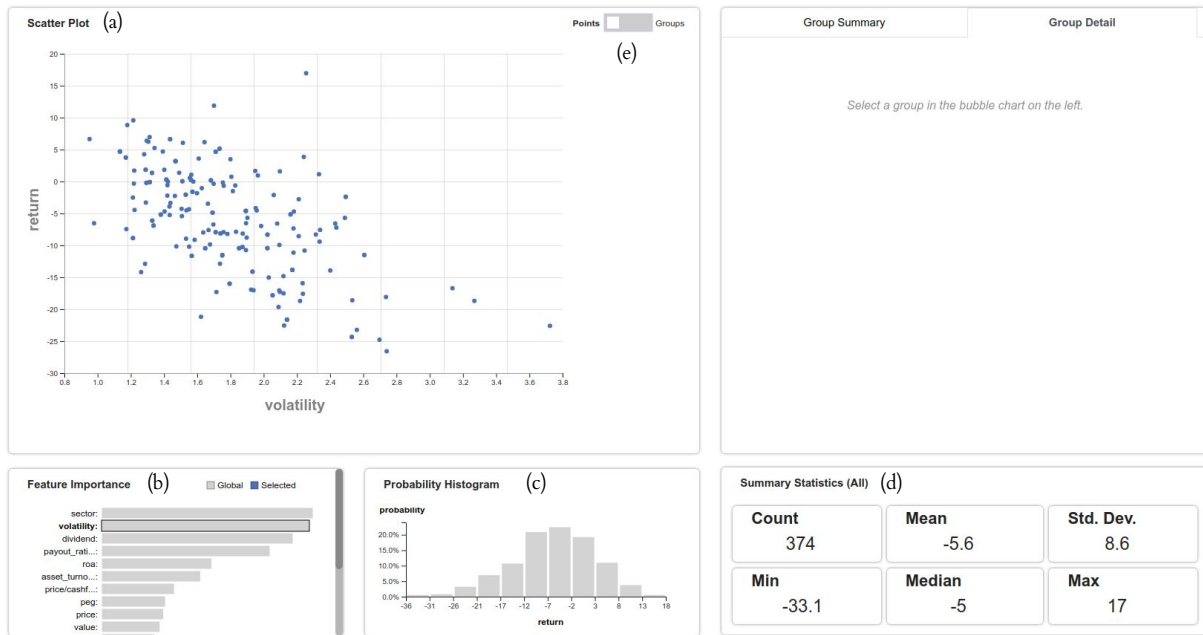


Figure 1: The initial view of the Analyzer widget interface. (a) Scatter plot plotting a selected attribute of interest (i.e. volatility) v.s. the target attribute (i.e. return). (b) Feature importance plot showing the relative predictive value of each of the features. (c) Probability histogram showing the distribution of the target variable (i.e. return). (d) Summary statistics for the target. (e) A switch for toggling between the scatter plot and the group bubble chart.

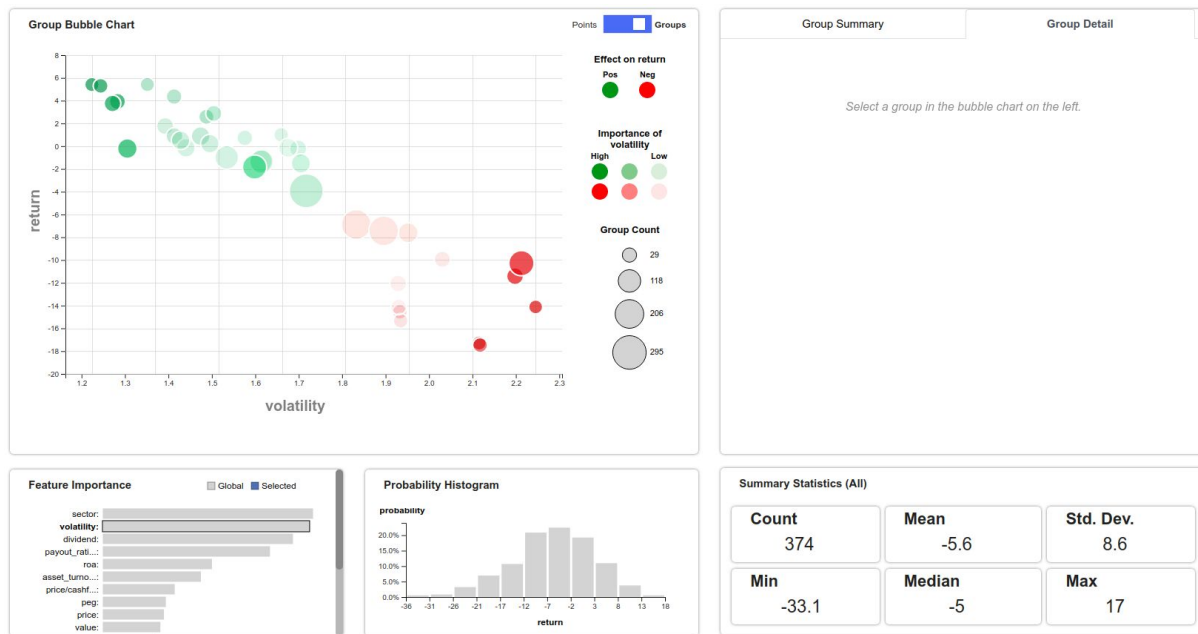


Figure 2: The Analyzer widget after a user switches to the group view.

circles in the mid-range of volatility are more transparent. This indicates that volatility is not an important attribute for these groups (i.e. these groups are defined by other features). Finally, the size of the circle is based on the number of data points within the group.

Clicking on a circle in the main plot will update the interface to show more detail about the clicked group (see figure 3). The clicked group (figure 3(a)) is indicated by a black border. This allows the user to easily track its new position if she changes the AOI. Figure 3(b) shows the updated feature importance plot. The blue bars show how much the selected group contributes to each feature's global feature importance. In this example, we can see that the selected group contributes the most to volatility and roa. This also indicates that these two attributes are the most important for this group.

Figure 3(c) shows the updated probability histogram. The red bars show the distribution of return within the group while the gray bars show the overall distribution of return. The summary statistics view (d) are updated to show the summary statistics of the selected group. This also includes colored text which shows the difference between the overall dataset and the selected group (e.g. the difference between mean return of the selected group and overall mean is -9.7).

Figure 3(e) shows the group detail panel. This view contains specific information about how the selected group is defined as well as its effect on the target attribute. This includes a description of the criteria for this group (i.e. the group is defined by data points with low roa and high volatility). Under the description is text which describes the effect of this group. Under the text are two white boxes which contain a quantitative description of the selected group. This group is defined by data points where $roa \leq 1.2$ and $volatility \geq 1.89$. The red bars associated with this description show the individual effect (i.e. using shapely values) of each of these constraints. In this case, the low roa constraint reduces the return by -5.6, while the high volatility constraint reduces the return by -4.1.

A scatter plot is associated with each constraint in the group detail view. Clicking on one of the white boxes will bring up the scatter plot (see figure 4). The scatter plot for a clicked box only contains points that satisfy all of the



Figure 3: The Analyzer widget interface after clicking on a group (a).

constraints above it in the group detail view. In figure 4, for example, the scatter plot only shows the points whose $roa \leq 1.20$. Conversely, the scatter plot associated with the first white box (i.e. the roa constraint) will contain all points in the data set. More generally, the first constraint in the group detail view will always contain the full scatter plot.

The group summary tab in figure 5 and figure 6 shows all feature's distributions for the selected group. Each feature is divided into 3 bins – low, medium and high. Each bin is then colored from white to blue based on the number of points within the bin. The first row in figure 6, for example, shows that the selected group has a low roa. The second row indicates that the selected group has high volatility. This makes sense since we know from the group detail view, that these are the defining characteristics of this group.

On the right we see a column labeled "Effect on return". This column provides the ability to do a counter-factual analysis. For example, if we were to add sector as a defining characteristic for this subgroup, it would only reduce the average return in this group by -1.1. This allows us to answer the question, "What if this group were defined in part by sector." In this case, once we know roa and volatility, the other features do not tell us much.

Description (Low roa + High volatility)

The probability that a randomly selected point within this group has a **lower return** than any point outside this group is **0.83**. This finding is statistically highly significant.

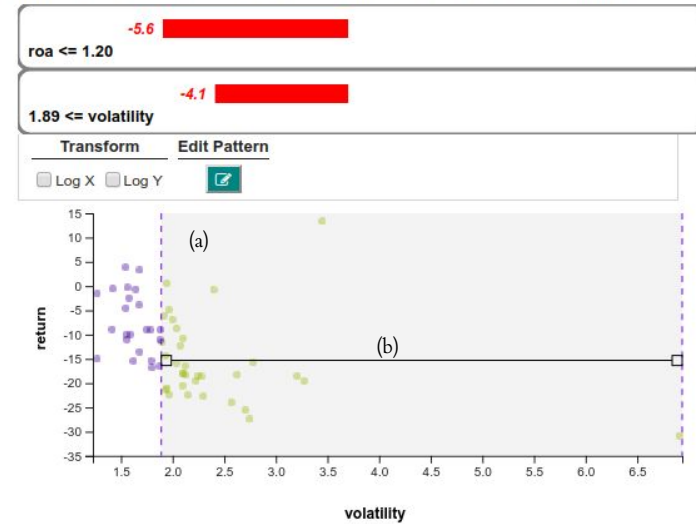


Figure 4: The scatter plot after clicking on volatility. The blue points represent the data points that fall outside the group, while the yellow points indicate the points inside the group. The transparent box (a) marks all the points that satisfy the constraint volatility ≥ 1.89 . The black line (b) shows the average return for this group.



Figure 5: The Analyzer widget interface after clicking on the "Group Summary" tab.

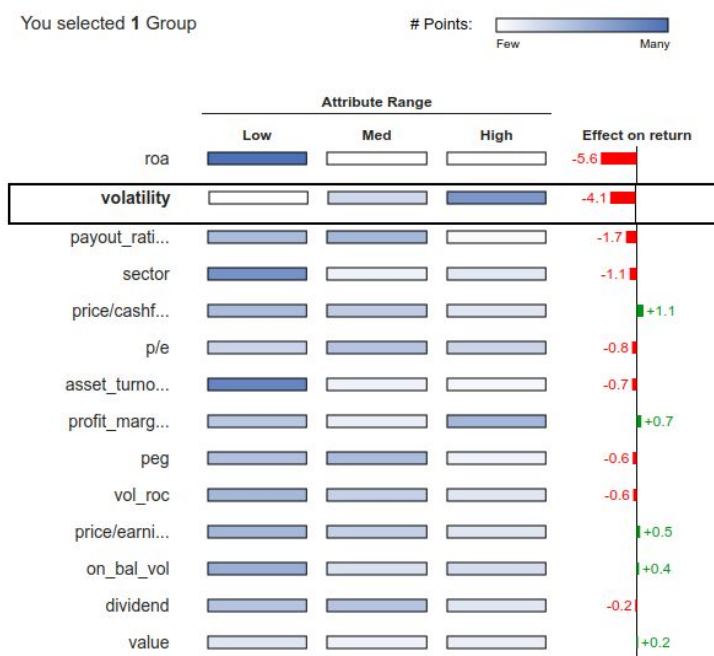


Figure 6: The group summary tab.

3 API

There are several analyzer widget specific functions that a user calls within the Jupyter notebook interface. These are explained in more detail in this section.

3.1 AKMiner

```
analyzer_widget.AKMiner(df, dependent, temporal=None, pattern_json=None, mine_type='numeric',
minsup=0.01, es_thresh='auto', max_pattern=None, max_depth=None, min_stable=None, ts_width=None,
train_range=None, lib_path=None, opt_bound=False, fdr='fast', holdout=3)
```

Description:

Performs pattern mining and prepares the visualization to be rendered.

Parameters:

- **df** (*pandas dataframe*): The data set to mine.
- **dependent** (*str*): The column of the dataframe that acts as the dependent variable. This is the attribute to predict.
- **temporal** (*str*): The attribute to be treated as a time variable.
- **pattern_json** (*list*): A list of pre-computed patterns of the form {attribute:{'lb': v1, 'ub': v2}}. When this parameter is set, pattern mining will not occur and instead this list of patterns will be visualized.
- **mine_type** (*str*): The data type of the dependent attribute. Valid mine types are 'numeric' and 'binary'.
- **minsup** (*float*): The minimum size threshold of a pattern as a percentage of the dataset size. Default is 1%.
- **es_thresh** (*float or dict*): The minimum effect size for a pattern to be considered 'interesting'. The effect size is the common language effect size for numeric mine_type (default is 0.6) and the odds ratio for binary or multiclass mine_types (default is 2). For multiclass mine_types, it can also be a dictionary mapping class ids (i.e. 0, 1, or 2) to its corresponding minimum effect size.
- **max_pattern** (*int*): Maximum number of patterns to mine.
- **max_depth** (*int*): Maximum complexity of a pattern. e.g. max_depth=2 indicates that no pattern containing more than 2 attributes will be returned.
- **min_stable** (*int*): The number of consecutive time steps for which a pattern must be 'interesting' to be considered temporally consistent.
- **ts_width** (*int*): Width of the timestep. If ts_width = 1, then each unique value of the temporal attribute is a timestep. If ts_width > 1, then each time step is a range.
- **train_range** (*list*): List defining [start, end] of a training range along the temporal dimension under which to mine the patterns. If None, then patterns are mined from the full data set.
- **lib_path** (*str*): File path to the libdcm.so (libdcm.dll) shared library.
- **opt_bound** (*bool*): If true, the bounds of the patterns are optimized throughout each level of the mining process.
- **fdr** (*str*): Method for controlling the false discovery rate during pattern mining ('fast' or 'exhaustive'). The 'fast' method is a greedy strategy, whereas the 'exhaustive' looks at all significant patterns and removes false discoveries afterwards.
- **holdout** (*int*): Number of holdout sets to validate the patterns on.

Returns:

An AKMiner object.

3.2 render

`AKMiner.render()`

Description:

Draws the data context map list view to the screen.

3.3 get_pattern_json

`AKMiner.get_pattern_json()`

Description:

Returns the mined patterns as list of python dictionaries of the form `{attribute:{'lb': v1, 'ub': v2}}`.