ADOPT DT-MIL Usage Guide

Getting Started:

- Make sure you have the latest version of Python 3 installed on your system
- Install the required dependencies (can be found in requirements.txt).
    - You can automatically install these dependencies by using the command “pip install -r requirements.txt” in the command line.

Formatting the Dataset:

- Run source/guis/dataset formatter.py

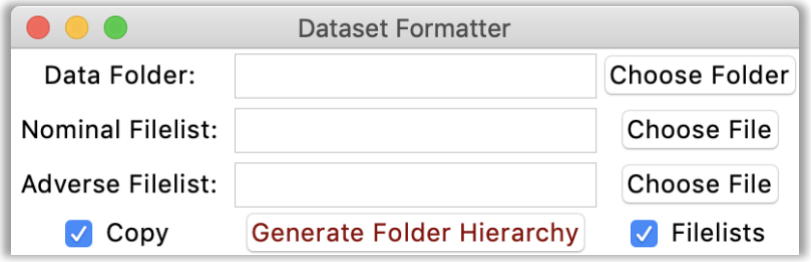

Fig. 1: Dataset Formatter default view

- Make sure the dataset contains a list of the nominal data and a list of the adverse data.
    - If you do not have the required file lists, but you have the nominal and adverse data in separate folders, uncheck the “filelists” checkbox, which will allow you to select a directory.

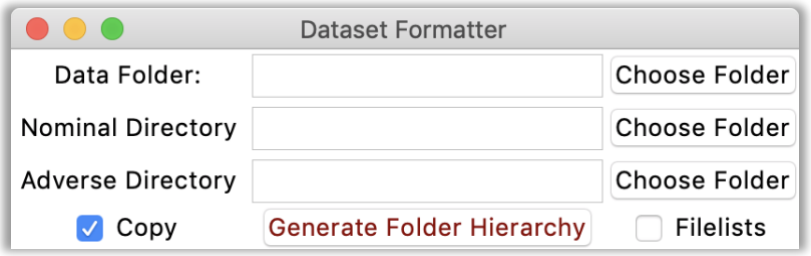

Fig. 2: Dataset Formatter directory selector

- Input the file paths for the nominal and adverse file lists (or directories) into the fields.
- Press “Generate Folder Hierarchy” and select the directory in which you want to save your dataset, i.e. [your_dataset_folder].
- Refer to readme.txt for a description of the created directory/file structure.
- For very large datasets, uncheck “copy” and the program will move the data to the created directory instead.
- There is also a command line version of this program (*dataset_formatter_cl.py*) that can be used as an alternative to the GUI.

Selecting Parameters

- Run source/guis/parameter_selector.py and select your dataset folder.
- The left column contains all the parameters in your dataset.

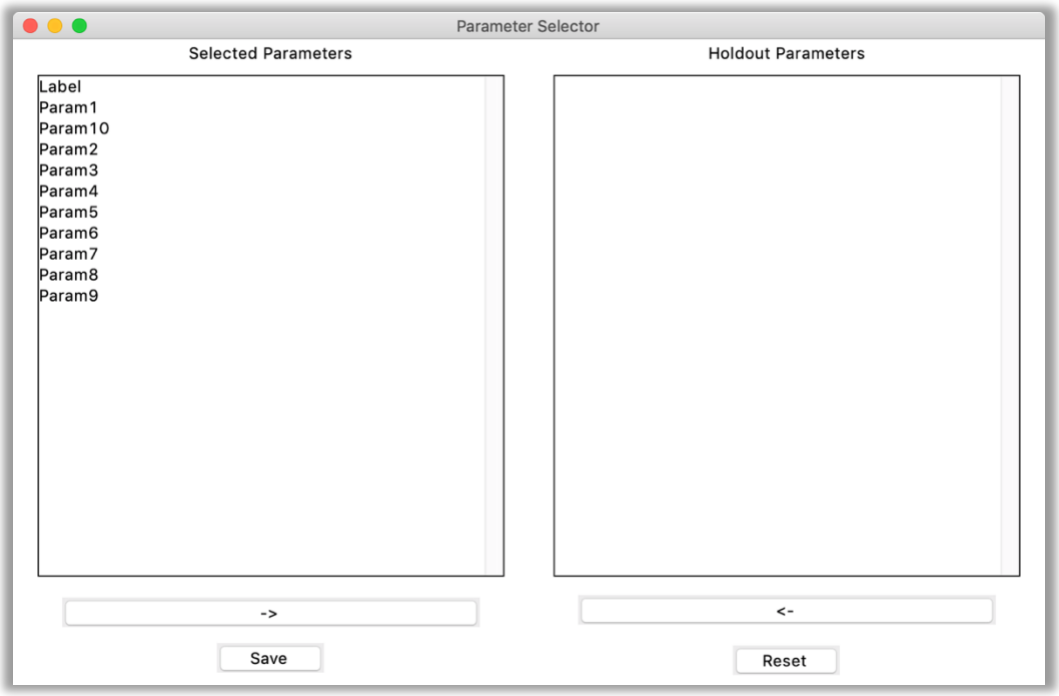

Fig. 3: Parameter Selector default view

- Highlight the parameter(s) you wish to be held out from training and press the arrow to move them over to the “holdout parameters column” These may be ground truth labels or other parameters that are known to directly trigger the event of interest.

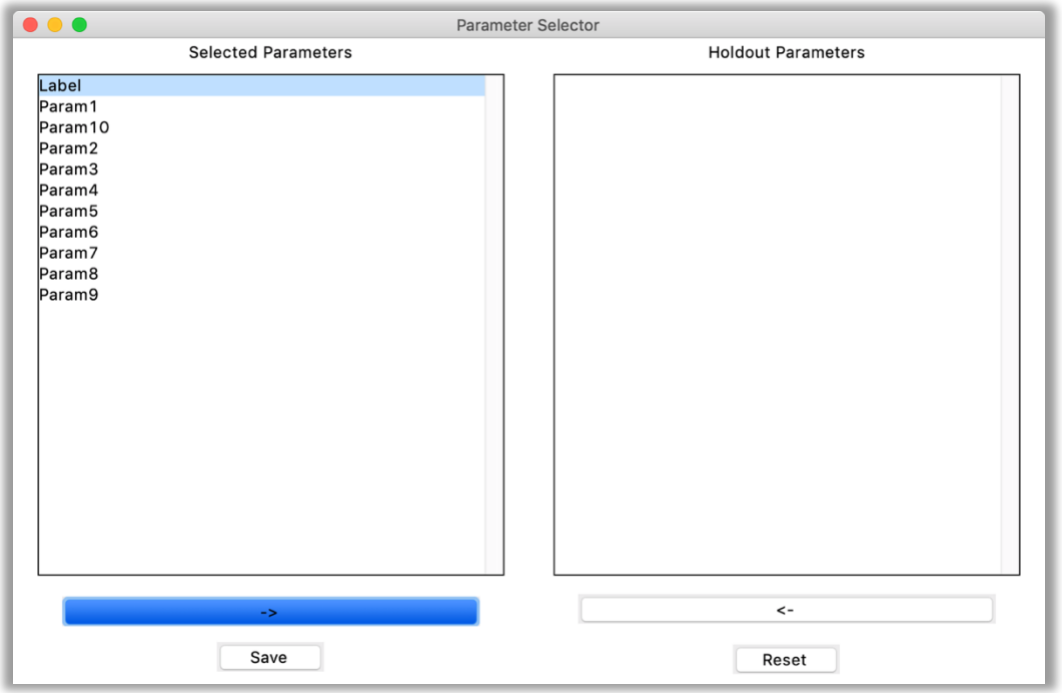

Fig. 3: Parameter Selector highlighted examples

- The same process can be repeated to move parameters from the Holdout column to the selected one.

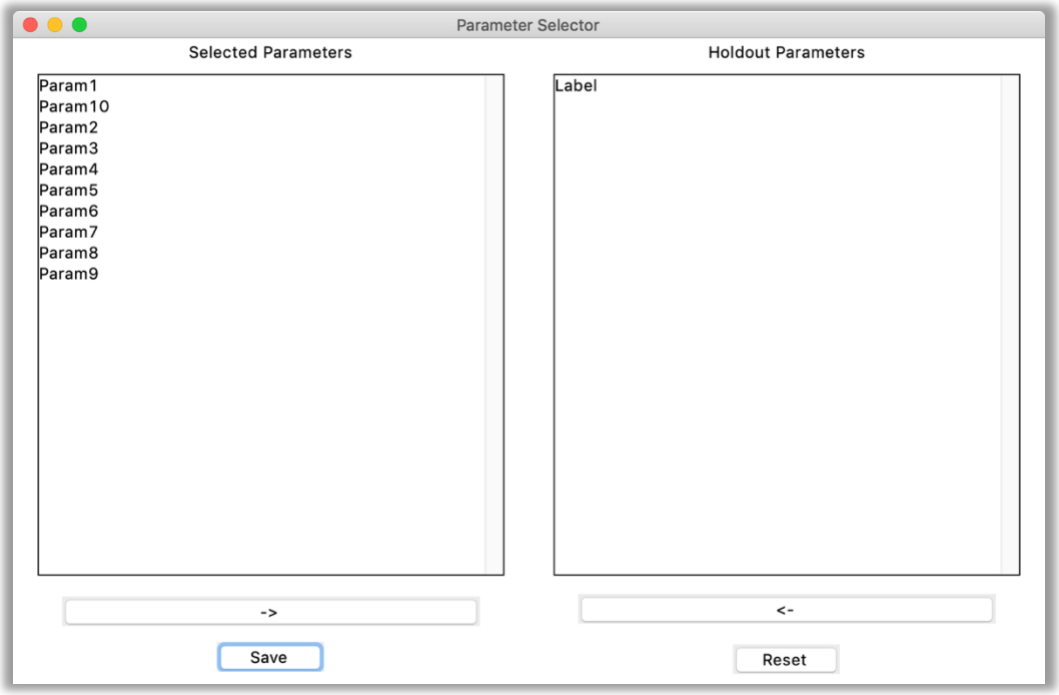

Fig. 3: Parameter Selector with both columns populated

- Once finished, press the “save” button.
- Or hit the “revert” button to revert the lists to the most recently saved version.
- The changes can be seen in [your_dataset_folder]/DTMIL_config.json.
- As an alternative to the GUI, you can also add the parameter names to the “redundant_parameters” entry in the DTMIL config file.

Editing the DTMIL Config File

- You can change parameters by opening [your_dataset_folder]/DTMIL_config.json and editing the field for the desired parameter.
- The descriptions for these parameters can be found in “config_readme.rtf” in the documentation folder.
- At the top of the file there is a parameter called “config_ID.” Changing this parameter will allow you to define the unique run identifier and naming convention of the output folder.

Running the Algorithm

- Run begin_DTMIL.py from the command line. Put the path of the dataset as the argument.
- Alternatively, if a path is not provided, the program will prompt you for one.
- You may stop the algorithm at any point during training and it will save the model of the most recent epoch.
- Once finished, a summary of the training is saved to the “output” folder and the model is saved to “[your_dataset_folder]/model_saves”. A copy of the config file used for this run can also be found there as well.

Visualizations (Parameter Graphs)

- Run source/simple_visualization.py and follow the instructions given onscreen
- Reading the graph:
    - The precursor score is in the last cell, and the parameter graphs are in the preceding cells.
    - When the precursor score crosses the threshold (marked with a thin dotted line), the parameter graphs are highlighted red at that time step.
    - Parameters that are chosen to be held out (during the parameter selection process) are marked with “(H/O)” in the graph title.
- An example of the output can be seen in: “documentation/parameter_graph_example.pdf”

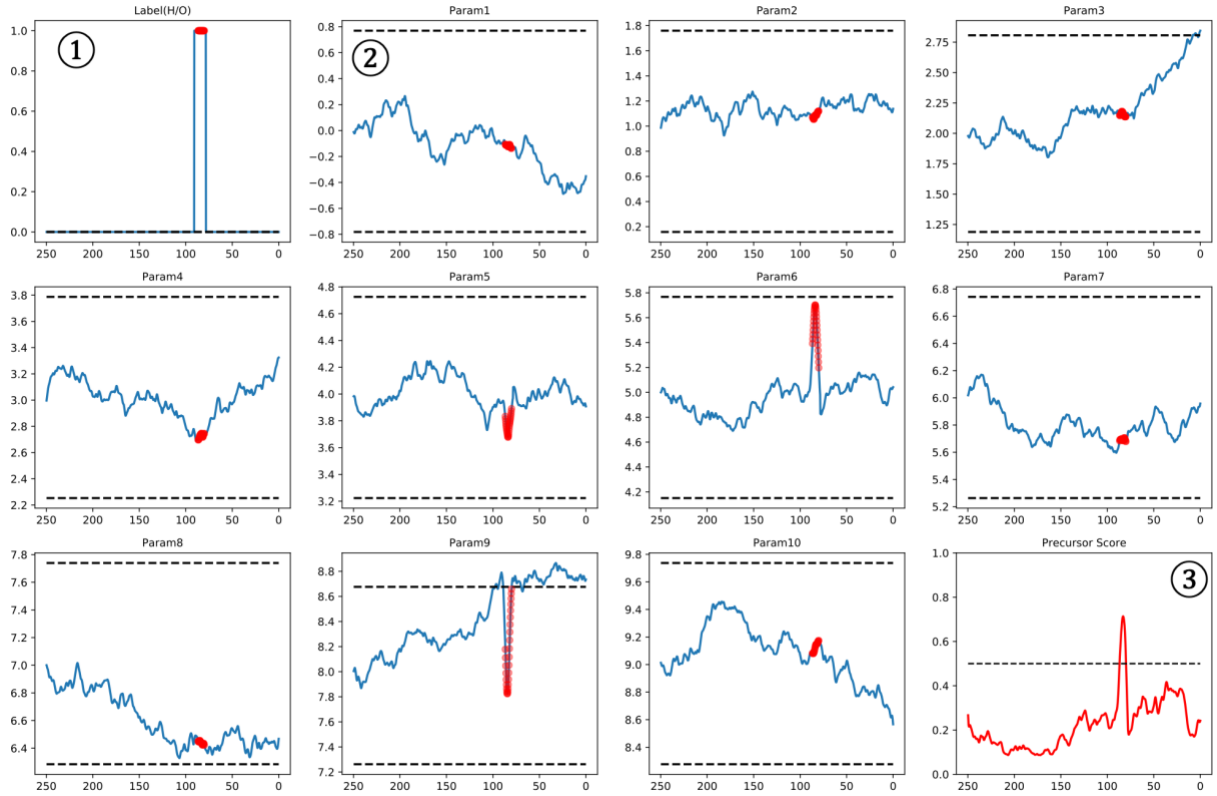

Fig. 4: Parameter graph output

1. Holdout (H/O) Window
2. Parameter Window
3. Precursor Score Window

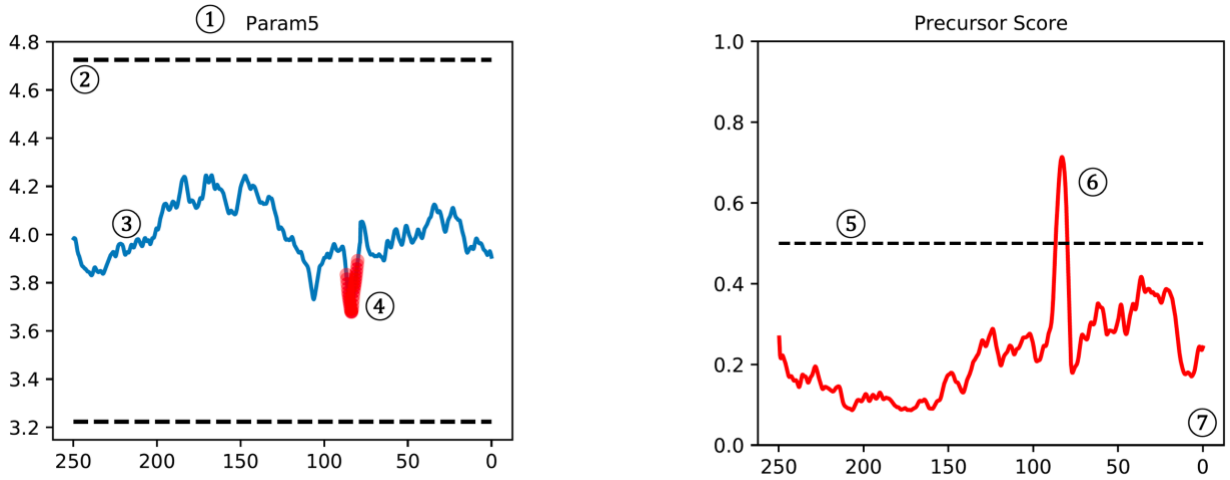

Fig. 5: Parameter graph output (zoomed in)

1. *Parameter name*
2. *Standard deviation line*
3. *Parameter graph*

4. *Precursor score highlight region*
5. *Precursor event detected*
6. *Precursor score window region (everything above the dotted line)*
7. *Timeline(sample before adverse event)*

Feature Ranking

- Run source/example_feature_ranking.py and follow the instructions onscreen.
- Up to 8 different standard deviation responses can be used for ranking (the example below uses two).
- Results can be found in [your_dataset_folder]/Output/Feature_Ranking.
- Reading the graph:
    - The ranking score metric is measured by measuring the area under the curve for the precursor score window, then measuring the area under the curve of the precursor score with a ± *σ* response applied to one of the parameters.
    - The final score is the percent difference between the two and is used to rank the sensitivity of the features for each precursor event. (e.g. a score of 90 means that the response suppressed the precursor score by 90%).
    - Each window is defined by the portions of the graph that are above the precursor threshold. If the graph rises above and falls below the precursor threshold multiple times, there will be multiple outputs and potentially separate precursor events.
    - The window for the current output is indicated by the shaded regions on the graph.
- An example of the output can be seen in "*documentation/feature_ranking_example.pdf*"

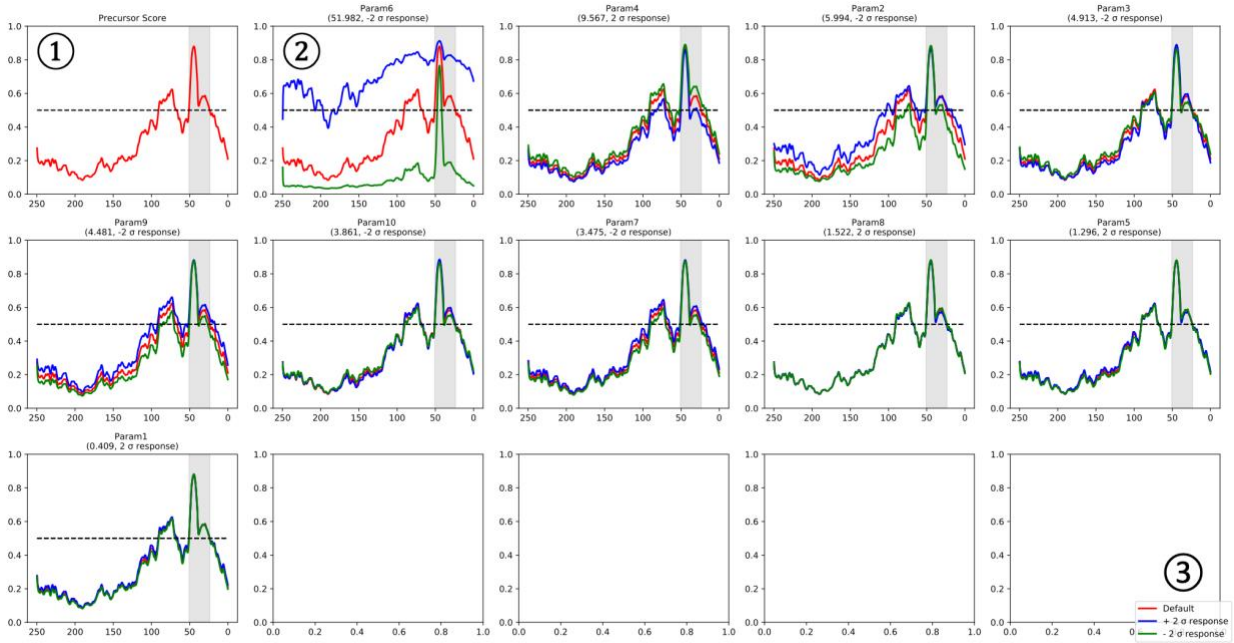

Fig. 6: Feature Ranking Output

1. Precursor Score Window
2. Response window
3. Legend

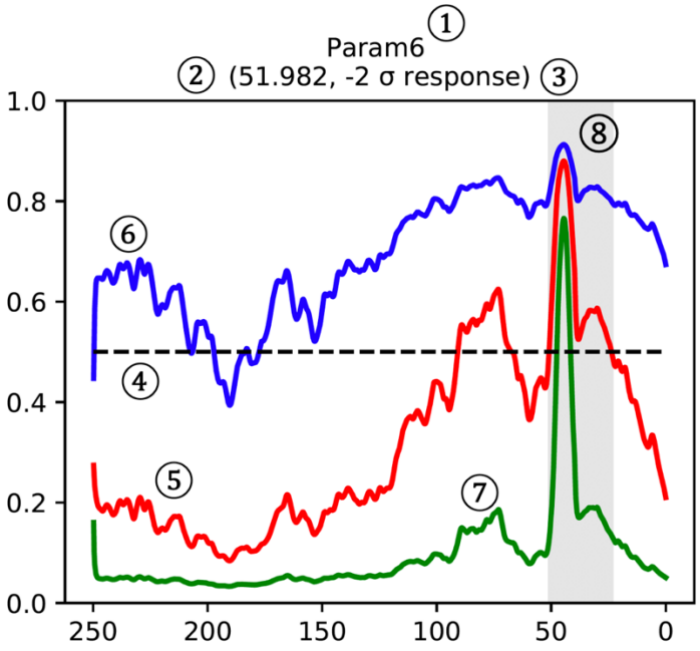

Fig. 7: Feature Ranking Output (Zoomed in)

1. *Parameter name*
2. *Percent difference from response*
3. *Most important σ response (+ means increasing the parameter lowers the precursor, - means reducing the parameter will lower the precursor score).*
4. *Precursor score threshold*
5. *Default precursor score (red)*
6. *+σ response (blue)*
7. *-σ response (green)*
8. *Shaded precursor window region*