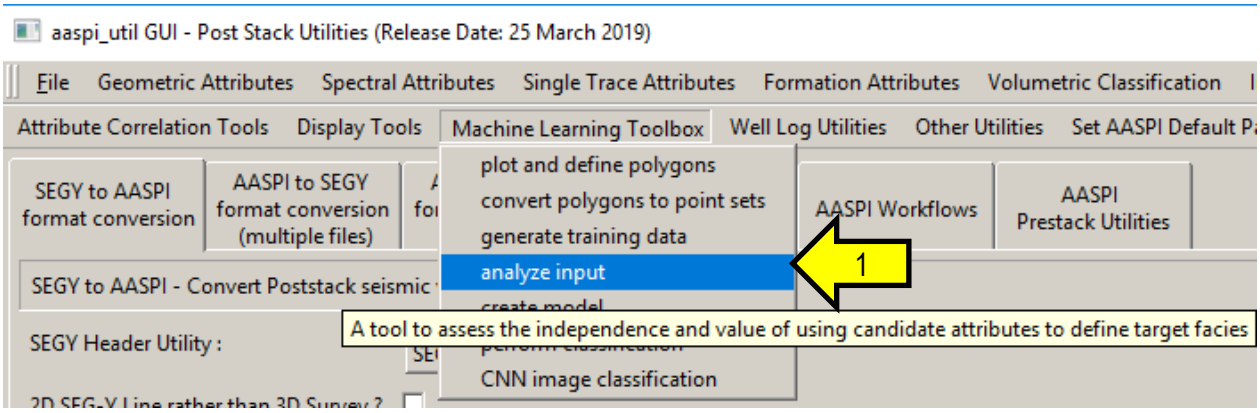
# MACHINE LEARNING TOOLBOX – Analyze Input

## Contents

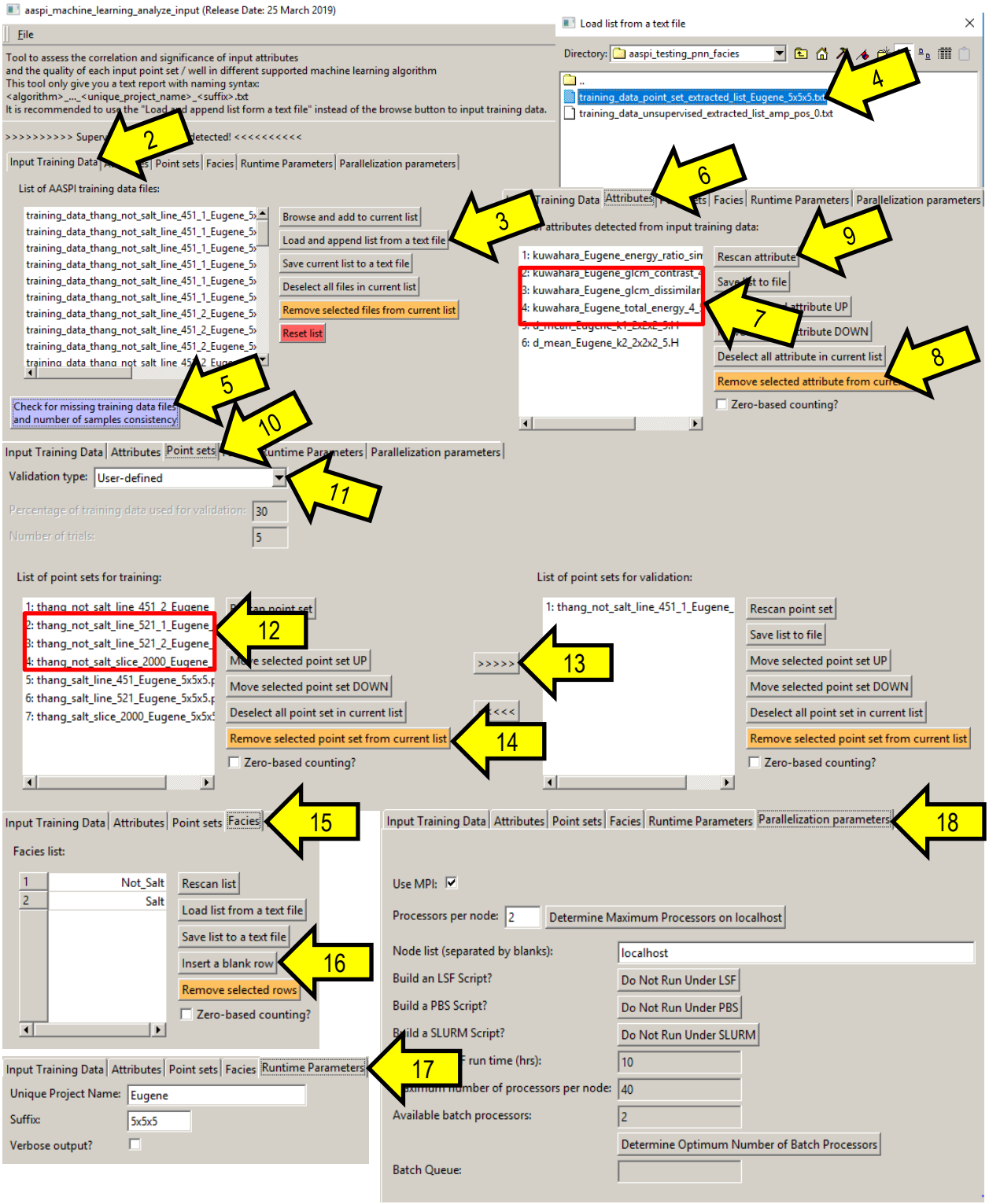


## Running aaspi_machine_learning_analyze_input

In order to understand the significance of each input attribute, point set, and modeling parameter toward the final result, it is essential to analyze input training data and test different parameter values so that users can have the best selection of attributes, point sets, and modeling parameters for classification. To analyze input training data, go to *Machine Learning Toolbox -> analyze input* (1).

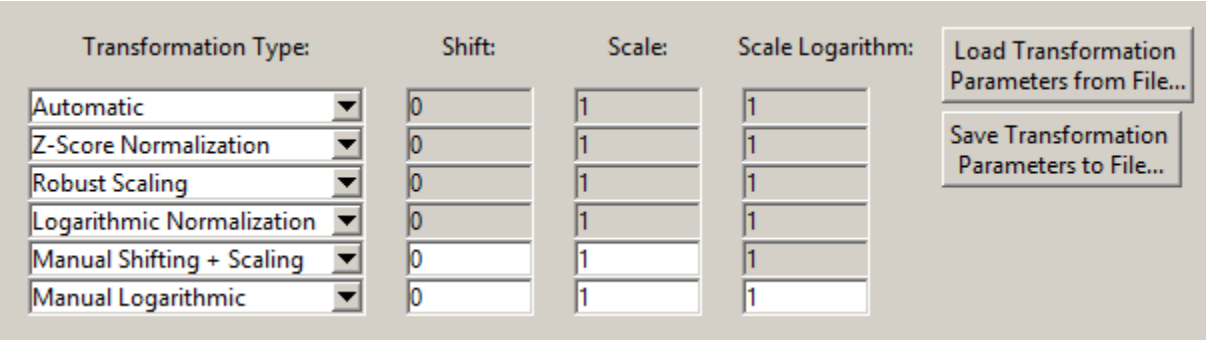



Click on *Input Training Data* tab (2). To quickly browse all training data generated by **aaspi_training_data**, click "*Load and append a list from a text file*" button (3), then select the extracted list text file (4). This list contains all extracted training data files from generate training data step. If you decide to remove some training data files from the list, make sure to click "*Check for missing training data files and number of samples consistency*" button (5). If there is no error, you can proceed. It is not recommended to remove training data files directly this way, but rather removing all training data belonging to a specific attribute or point set. More on this later.




Click on "*Attributes*" tab (6) to view the automatically detected attribute list associated with the input training data. You can move attribute up/down in the list. To remove some attributes and all of theirs associated training data files, first highlight them in the list (7) by clicking on them,

and then click "*Remove selected attributes from current list*" button (8). If you want to reset to the default attribute list, click on "*Rescan Attribute*" button (9).
To the right of attribute list, you will find transformation parameter table:




Each row of the transformation parameter table is for the corresponding attribute in the list. There are 6 options for transformation type:

- **Automatic**: The program will automatically determine if the input training attribute need logarithmic normalization or just a simple Z-Score normalization, based on how asymmetric it is. The symmetry of a data distribution is determined by the following ratio:

$$r = \frac{peak - p16}{p84 - peak}$$

  Where:
  + peak is the peak (or the mode) of a data distribution (basically the point of densest probability).
  + p16: 16% percentile of the data distribution
  + p84: 84% percentile of the data distribution
  If 0.5<r<2.0, then z-score normalization will be applied. Otherwise, logarithmic normalization will be applied.

- **Z-Score normalization**: The most commonly used normalization scheme:

$$T(x) = \frac{x - \mu}{\sigma}$$

  Where:
  + μ is the mean (arithmetic average) of the data
  + σ is the standard deviation of the data

- **Robust scaling**: Another shift and scaling scheme used widely in machine learning that is supposed to be more robust toward outliers:

$$T(x) = \frac{x - q2}{q3 - q1}$$

  Where:
  + q1 is the value at the first quarter of the data distribution (i.e. same as p25)
  + q2 is the median of the data distribution (i.e. same as p50)
  + q3 is the value at the third quarter of the data distribution (i.e. same as p75)

- **Logarithmic normalization**: AASPI-implementation of logarithmic transformation:

$$T(x) = c * \ln\big((x + a) * b\big)$$

With:

$$a = \frac{P^2 - L * R}{L + R - 2P}$$

$$b = \frac{e^{-\mu_{log}}}{P + a}$$

$$c = \frac{1}{\sigma_{log}}$$

Where:
+ P: the peak of the input data distribution
+ L: Left endpoint of the input data distribution (usually 1% percentile, but could be the minimum of the input data distribution)
+ R: Right endpoint of the input data distribution (usually 99% percentile, but could be the maximum of the input data distribution)
+ $\mu_{log}$: the mean of the data distribution just after applying logarithm function.
+ $\sigma_{log}$: the standard deviation of the data distribution just after applying logarithm function.

Because the peak of the logarithmically transformed data distribution is not the same as the peak of the original data distribution, P, a, and b have to be computed in an iterative manner. Currently we found that 100 iterations are sufficiently precise enough. Also, if the distribution is flipped during the logarithmic normalization (i.e. having negative stretch value b), then the log_scale value c would have its sign reversed as well. This is to ensure the order of transformed data distribution is the same as that of the original data distribution.

- **Manual shift and scale**: User-defined shift and scale factor of the transformation. This option is usually for advanced user who wants to experiment with different shifting and scaling, or if the input attribute has an absolute physical mean and standard deviation (which is very Unlikely).

$$T(x) = (x + shift) * scale$$

- **Manual shift, scale, and log scale:** User-defined shift, scale, and log scale of the transformation:

$$T(x) = \text{logscale} * \ln\big((x + shift) * scale\big)$$

Note: because ln() function cannot accept non-positive value (i.e. ln(0) = -∞), it is highly advised that the shift and scale factor would make the data distribution entirely positive. Otherwise, you will encounter NaN (Not-a-Number) error. This option is usually for advanced user only.

For supervised classification scheme, click on "*Point sets*" tab (10) to view and define training point sets and validation point sets. Currently AASPI support two methods of training/validation data distribution (11):

- User-defined: the user will decide which point sets to be used for training and which for validation. To transfer some training point sets to the validation list, first select the training point sets you want to move by clicking on them (12), then click on the transfer button (13) to swap them to the validation list (and vice versa). The analysis is then performed just on that particular training-validation setting.
- One-point-set cross-validation: The program will loop through all training point sets and choose one for validation. If there are N training point sets, there will be N analyses. Transfer buttons and validation list will be disabled for this method.

To remove a point set and all associated training data files, highlight them and click on "Remove selected point sets from current list" button (14).

For supervised classification scheme, click on "*Facies*" tab (15) to view and edit the facies list associated with the input training data. **CAUTION! DO NOT modify facies names, unless you are trying to merge multiple training data sets with different facies lists. The analysis depends on facies names to match a training data file to a facies, and if it cannot find a matching facies name, it will discard the training data file from the analysis! In case you do need to modify the facies list, you can insert a blank row (16) and then double click on the blank row to define the new facies name. DO NOT modify existing facies!**

Click on "*Run time parameters*" tab (17) to define unique project name and suffix for an analysis. These should be automatically loaded after user browse input training data.
Finally, click on "*Parallelization parameters*" tab (18) to define MPI parameters (i.e. number of parallel processors).

**In the lower section, click on the desired analysis. Please refer each analysis's documentation to see how to set up the parameters for each of them. The documentation file name of individual analysis is as follow:**
**Machine_Learning_Toolbox-analyze_input-<analysis_or_algorithm_name>.pdf**

## Output file naming convention

Program **aaspi_machine_learning_analyze_input** will always generate the following output files:

| **Output file description** | **File name syntax** |
|---|---|
| program log information | machine_learning_analyze_input_analysis_name_*unique_project_name_suffix*.log |
| program error/completion information | machine_learning_analyze_input_analysis_name_*unique_project_name_suffix*.err |

where the values in red are defined by the program GUI. The errors we anticipated will be written to the **.err* file and be displayed in a pop-up window upon program termination. These errors,

much of the input information, a description of intermediate variables, and any software trace-back errors will be contained in the **.log* file.