# Planner requirements in restriction file

Chen Ling (Author)

This document explains the schema in restriction file of human-guided machine learning (HGML), in which the user provide their restriction requirement to our system.

Human-guided machine learning (HGML) is a hybrid approach where a user interacts with an AutoML system and tasks it to explore different problem settings that reflect the user’s knowledge about the data available.

**Schema documentation**

* **include\_model** [status: implemented]
  + Definition: this flag allows you to customize the model being included in the solution pipelines.
  + assumption
  + Parameters: Array with the ids of the primitive models to be included. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will explore all possible pipelines.
  + Example: "include\_model":["LinearSVC","LogisticRegression"],
* **exclude\_model** [status: implemented]
  + Definition: this flag allows you to customize the model being excluded in the solution pipelines.
  + Parameters: Array with the ids of the primitive models to be excluded. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will explore all possible pipelines.
  + If include\_model and exclude\_model has the same models, the TA2 system will explore all possible pipelines.
  + Example: "exclude\_model":["LinearSVC","LogisticRegression"]
* **include\_feature\_generation** [status: implemented]
  + Definition: this flag allows you to provide a rule that generate the features
  + Parameters: Array with the names of feature generation primitive. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will explore all possible pipelines.
  + Example: "include\_feature\_generation":[""]
* **use\_imputation\_method** [status: implemented]
  + Definition: this flag allows you to use specific imputation method for missing values
  + Parameters: Array with the imputation method names. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will use “mean” to impute missing values.
  + Example: "use\_imputation\_method":["median","most frequent"]
* **include\_variables** [status: not implemented]
  + Definition: this flag allows you to customize the variables (columns) being included in the solution.
  + Parameters: Array with the ids of the variables to be included.
  + Default value: if empty, the TA2 system will use all variables.
  + Example: "include\_variables":[]
* **exclude\_variables** [status: not implemented]
  + Definition: this flag allows you to customize the variables being excluded in the solution.
  + Parameters: Array with the ids of the variables to be included.
  + Default value: if empty, the TA2 system will use all variables.
  + Example: "exclude\_variables":[]
* **include\_instances** [status: not implemented]
  + Definition: this flag allows you to customize the instances (rows) being included in the solution.
  + Parameters: Array with the ids of the instances to be included.
  + Default value: if empty, the TA2 system will use all instances.
  + Example: "include\_instances":[]
* **exclude\_instances** [status: not implemented]
  + Definition: this flag allows you to customize the instances being included in the solution.
  + Parameters: Array with the ids of the instances to be excluded.
  + Default value: if empty, the TA2 system will use all instances.
  + Example: "exclude\_instances":[]
* **define\_variable\_weight** [status: not implemented]
  + Definition: this flag allows you to specify the priority/ relative weight of variables.
  + Parameters: dictionary includes the ids of the variables and their priority .
  + Default value: if empty, the TA2 system will set each variable the same priority.
  + Example: "define\_variable\_weight":{“variable\_id“: ,”priority”:1}
* **select\_training\_and\_test\_data** [status: not implemented]
  + Definition: this flag allows you to select training and testing data, optionally with cross-validation specifications.
  + Parameters: dictionary includes the ids of instances to be the training and testing data, with optional cross-validation method .
  + Default value: if empty, the TA2 system will use the default method to get training and testing data.
  + Example: "select\_training\_and\_testing\_data":{“training\_data“:(1,1000) ,”testing\_data”:(1001,1200), “cross\_validation”:”k fold”}
* **replace\_model** [status: not implemented]
  + Definition: this flag allows you to have two or more solutions with different models but same other steps.
  + Parameters: Array contains the model you want to replace.
  + Default value: if empty, the TA2 system will explore all possible pipelines
  + Example: "replace\_model": {“replace\_model”: [“LogisticRegression”],”new\_model”:[“RandomForestClassifier”]}
* **use\_specific\_parameter\_for\_model** [status: not implemented]
  + Definition: this flag allows you to specify a model and the parameter values desired
  + Parameters: An array of dictionaries, including the model and its parameters, the parameters are stored in a dictionary
  + Default value: if empty, the TA2 system will use the default parameters
  + Example: "use\_specific\_parameter\_for\_model": [{"model":"DecisionTreeClassifier","parameter":{},”parameter\_value:}]
* **include\_class\_of\_model** [status: not implemented]
  + Definition: this flag allows you to specify the class of model desired
  + Parameters: An array of class of model. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will explore all possible pipelines
  + Example: "include\_class\_of\_model": []
* **include\_class\_of\_data\_preparation\_method** [status: not implemented]
  + Definition: this flag allows you to specify the class of data preparation method
  + Parameters: An array of class of data preparation method. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will explore all possible pipelines
  + Example: "include\_class\_of\_data\_preparation\_method": []
* **include\_statistical\_test** [status: not implemented]
  + Definition: this flag allows you to request a particular statistic test and parameters
  + Parameters: An array of statistical test. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will explore the default statistical test
  + Example: "include\_statistical\_test": []
* **results\_after\_step** [status: not implemented]
  + Definition: this flag allows you to request results after any step in a solution
  + Parameters: step’s name. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will show the final result
  + Example: "results\_after\_step": “DataPreprocessing”
* **compare\_solutions** [status: not implemented]
  + Definition: this flag allows you to generate comparative explanations for two given solutions
  + Parameters: dictionary includes model name and different solutions. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will show the final result
  + Example: "compare\_solutions": {"model":,"data\_preparation\_method":[]}
* **compare\_models** [status: not implemented]
  + Definition: this flag allows you to generate comparative models.
  + Parameters: array includes model names. All primitives have to be registered as part of the primitive library of the TA2 system.
  + Default value: if empty, the TA2 system will only show the final result
  + Example: "compare\_models": []