

Package ‘iai’

January 29, 2020

Type Package

Title Interface to 'Interpretable AI' Modules

Version 1.2.0

Description An interface to the algorithms of 'Interpretable AI' <<https://www.interpretable.ai>> from the R programming language. 'Interpretable AI' provides various modules, including 'Optimal Trees' for classification, regression, prescription and survival analysis, 'Optimal Imputation' for missing data imputation and outlier detection, and 'Optimal Feature Selection' for exact sparse regression. The 'iai' package is an open-source project. The 'Interpretable AI' software modules are proprietary products, but free academic and evaluation licenses are available.

URL <https://www.interpretable.ai>

SystemRequirements Julia (>= 1.0) and Interpretable AI System Image (>= 1.0.0)

License MIT + file LICENSE

Imports JuliaCall, stringr, rlang

RoxygenNote 7.0.2

Suggests testthat, covr

NeedsCompilation yes

Author Jack Dunn [aut, cre],
Ying Zhuo [aut],
Interpretable AI LLC [cph]

Maintainer Jack Dunn <jack@interpretable.ai>

Repository CRAN

Date/Publication 2020-01-29 22:20:02 UTC

R topics documented:

apply	3
apply_nodes	4
as.mixeddata	5

clone	5
decision_path	6
delete_rich_output_param	6
fit	7
fit_cv	7
fit_transform	8
fit_transform_cv	9
get_best_params	9
get_classification_label	10
get_classification_proba	10
get_depth	11
get_grid_results	11
get_learner	12
get_lower_child	12
get_num_nodes	13
get_num_samples	13
get_params	14
get_parent	14
get_prediction_constant	15
get_prediction_weights	15
get_prescription_treatment_rank	16
get_regression_constant	16
get_regression_weights	17
get_rich_output_params	18
get_split_categories	18
get_split_feature	19
get_split_threshold	19
get_split_weights	20
get_survival_curve	20
get_survival_curve_data	21
get_upper_child	21
grid_search	22
iai_setup	22
imputation_learner	23
impute	23
impute_cv	24
is_categoric_split	24
is_hyperplane_split	25
is_leaf	25
is_mixed_ordinal_split	26
is_mixed_parallel_split	26
is_ordinal_split	27
is_parallel_split	27
mean_imputation_learner	28
missing_goes_lower	28
multi_questionnaire	29
multi_tree_plot	29
optimal_feature_selection_classifier	30

optimal_feature_selection_regressor	31
optimal_tree_classifier	31
optimal_tree_prescription_maximizer	32
optimal_tree_prescription_minimizer	32
optimal_tree_regressor	33
optimal_tree_survivor	33
opt_knn_imputation_learner	34
opt_svm_imputation_learner	34
opt_tree_imputation_learner	35
predict	35
predict_hazard	36
predict_outcomes	36
predict_proba	37
print_path	37
questionnaire	38
rand_imputation_learner	38
read_json	39
reset_display_label	39
roc_curve	40
score	40
set_display_label	41
set_julia_seed	41
set_params	42
set_rich_output_param	42
set_threshold	43
show_in_browser	43
show_questionnaire	44
single_knn_imputation_learner	44
split_data	45
transform	45
tree_plot	46
variable_importance	46
write_dot	47
write_html	47
write_json	48
write_png	48
write_questionnaire	49

Index **50**

apply	<i>Return the leaf index in a tree model into which each point in the features falls</i>
-------	--

Description

Julia Equivalent: [IAI.apply](#)

Usage

```
apply(lnr, X)
```

Arguments

lnr	The learner or grid to query.
X	The features of the data.

Examples

```
## Not run: iai::apply(lnr, X)
```

apply_nodes	<i>Return the indices of the points in the features that fall into each node of a trained tree model</i>
-------------	--

Description

Julia Equivalent: [IAI.apply_nodes](#)

Usage

```
apply_nodes(lnr, X)
```

Arguments

lnr	The learner or grid to query.
X	The features of the data.

Examples

```
## Not run: iai::apply_nodes(lnr, X)
```

as.mixeddata	<i>Convert a vector of values to IAI mixed data format</i>
--------------	--

Description

Julia Equivalent: `IAI.make_mixed_data`

Usage

```
as.mixeddata(values, categorical_levels, ordinal_levels = c())
```

Arguments

`values` The vector of values to convert
`categorical_levels` The values in values to treat as categoric levels
`ordinal_levels` (optional) The values in values to treat as ordinal levels, in the order supplied

Examples

```
df <- iris
set.seed(1)
df$mixed <- rnorm(150)
df$mixed[1:5] <- NA # Insert some missing values
df$mixed[6:10] <- "Not graded"
df$mixed <- iai::as.mixeddata(df$mixed, c("Not graded"))
```

clone	<i>Return an unfitted copy of a learner with the same parameters</i>
-------	--

Description

Julia Equivalent: `IAI.clone`

Usage

```
clone(lnr)
```

Arguments

`lnr` The learner to copy.

Examples

```
## Not run: new_lnr <- iai::clone(lnr)
```

decision_path	<i>Return a matrix where entry (i, j) is true if the i-th point in the features passes through the j-th node in a trained tree model.</i>
---------------	---

Description

Julia Equivalent: `IAI.decision_path`

Usage

```
decision_path(lnr, X)
```

Arguments

lnr	The learner or grid to query.
X	The features of the data.

Examples

```
## Not run: iai::decision_path(lnr, X)
```

delete_rich_output_param	<i>Delete a global rich output parameter</i>
--------------------------	--

Description

Julia Equivalent: `IAI.delete_rich_output_param!`

Usage

```
delete_rich_output_param(key)
```

Arguments

key	The parameter to delete.
-----	--------------------------

Examples

```
## Not run: iai::delete_rich_output_param("simple_layout")
```

fit *Fits a model to the training data*

Description

Julia Equivalent: `IAI.fit!`

Usage

```
fit(lnr, X, ...)
```

Arguments

lnr	The learner or grid to fit.
X	The features of the data.
...	Other parameters, including zero or more target vectors as required by the problem type. Refer to the Julia documentation for available parameters.

Examples

```
X <- iris[, 1:4]
y <- iris$Species
grid <- iai::grid_search(
  iai::optimal_tree_classifier(max_depth = 1),
)
iai::fit(grid, X, y)
```

fit_cv *Fits a grid search to the training data with cross-validation*

Description

Julia Equivalent: `IAI.fit_cv!`

Usage

```
fit_cv(grid, X, ...)
```

Arguments

grid	The grid to fit.
X	The features of the data.
...	Other parameters, including zero or more target vectors as required by the problem type. Refer to the Julia documentation for available parameters.

Examples

```
X <- iris[, 1:4]
y <- iris$Species
grid <- iai::grid_search(
  iai::optimal_tree_classifier(max_depth = 1),
)
iai::fit_cv(grid, X, y)
```

fit_transform	<i>Fit an imputation model using the given features and impute the missing values in these features</i>
---------------	---

Description

Similar to calling `fit` followed by `transform`

Usage

```
fit_transform(lnr, X, ...)
```

Arguments

lnr	The learner or grid to use for imputation
X	The features of the data.
...	Refer to the Julia documentation for available parameters.

Details

Julia Equivalent: `IAI.fit_transform!`

Examples

```
X <- iris
X[1, 1] <- NA
grid <- iai::grid_search(
  iai::imputation_learner(),
  method = c("opt_knn", "opt_tree"),
)
iai::fit_transform(grid, X)
```

fit_transform_cv	<i>Train a grid using cross-validation with features and impute all missing values in these features</i>
------------------	--

Description

Julia Equivalent: `IAI.fit_transform_cv!`

Usage

```
fit_transform_cv(grid, X, ...)
```

Arguments

grid	The grid to use for imputation
X	The features of the data.
...	Refer to the Julia documentation for available parameters.

Examples

```
X <- iris
X[1, 1] <- NA
grid <- iai::grid_search(
  iai::imputation_learner(),
  method = c("opt_knn", "opt_tree"),
)
iai::fit_transform_cv(grid, X)
```

get_best_params	<i>Return the best parameter combination from a grid</i>
-----------------	--

Description

Julia Equivalent: `IAI.get_best_params`

Usage

```
get_best_params(grid)
```

Arguments

grid	The grid search to query.
------	---------------------------

Examples

```
## Not run: iai::get_best_params(grid)
```

```
get_classification_label
```

Return the predicted label at a node of a tree

Description

Julia Equivalent: `IAI.get_classification_label`

Usage

```
get_classification_label(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::get_classification_label(lnr, 1)
```

```
get_classification_proba
```

Return the predicted probabilities of class membership at a node of a tree

Description

Julia Equivalent: `IAI.get_classification_proba`

Usage

```
get_classification_proba(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::get_classification_proba(lnr, 1)
```

get_depth	<i>Get the depth of a node of a tree</i>
-----------	--

Description

Julia Equivalent: `IAI.get_depth`

Usage

```
get_depth(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::get_depth(lnr, 1)
```

get_grid_results	<i>Return a summary of the results from the grid search</i>
------------------	---

Description

Julia Equivalent: `IAI.get_grid_results`

Usage

```
get_grid_results(grid)
```

Arguments

grid	The grid search to query.
------	---------------------------

Examples

```
## Not run: iai::get_grid_results(grid)
```

get_learner	<i>Return the fitted learner using the best parameter combination from a grid</i>
-------------	---

Description

Julia Equivalent: `IAI.get_learner`

Usage

```
get_learner(grid)
```

Arguments

grid	The grid to query.
------	--------------------

Examples

```
## Not run: lnr <- iai::get_learner(grid)
```

get_lower_child	<i>Get the index of the lower child at a split node of a tree</i>
-----------------	---

Description

Julia Equivalent: `IAI.get_lower_child`

Usage

```
get_lower_child(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::get_lower_child(lnr, 1)
```

get_num_nodes	<i>Return the number of nodes in a trained learner</i>
---------------	--

Description

Julia Equivalent: `IAI.get_num_nodes`

Usage

```
get_num_nodes(lnr)
```

Arguments

lnr The learner to query.

Examples

```
## Not run: iai::get_num_nodes(lnr)
```

get_num_samples	<i>Get the number of training points contained in a node of a tree</i>
-----------------	--

Description

Julia Equivalent: `IAI.get_num_samples`

Usage

```
get_num_samples(lnr, node_index)
```

Arguments

lnr The learner to query.
node_index The node in the tree to query.

Examples

```
## Not run: iai::get_num_samples(lnr, 1)
```

get_params	<i>Return the value of all parameters on a learner</i>
------------	--

Description

Julia Equivalent: `IAI.get_params`

Usage

```
get_params(lnr)
```

Arguments

lnr	The learner to query.
-----	-----------------------

Examples

```
## Not run: iai::get_params(lnr)
```

get_parent	<i>Get the index of the parent node at a node of a tree</i>
------------	---

Description

Julia Equivalent: `IAI.get_parent`

Usage

```
get_parent(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::get_parent(lnr, 2)
```

`get_prediction_constant`*Return the constant term in the prediction in the trained learner*

Description

Julia Equivalent: `IAI.get_prediction_constant`

Usage

```
get_prediction_constant(lnr)
```

Arguments

`lnr` The learner to query.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run: iai::get_prediction_constant(lnr)
```

`get_prediction_weights`*Return the weights for numeric and categoric features used for prediction in the trained learner*

Description

Julia Equivalent: `IAI.get_prediction_weights`

Usage

```
get_prediction_weights(lnr)
```

Arguments

`lnr` The learner to query.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run: iai::get_prediction_weights(lnr)
```

```
get_prescription_treatment_rank
```

Return the treatments ordered from most effective to least effective at a node of a tree

Description

Julia Equivalent: `IAI.get_prescription_treatment_rank`

Usage

```
get_prescription_treatment_rank(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::get_prescription_treatment_rank(lnr, 1)
```

```
get_regression_constant
```

Return the constant term in the regression prediction at a node of a tree

Description

Julia Equivalent: `IAI.get_regression_constant` (for regression or prescription tree learners as appropriate)

Usage

```
get_regression_constant(lnr, node_index, ...)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.
...	If a prescription problem, the treatment to query.

Examples

```
## Not run:  
iai::get_regression_constant(lnr, 1)  
iai::get_regression_constant(lnr, 1, "A")  
  
## End(Not run)
```

get_regression_weights

Return the weights for each feature in the regression prediction at a node of a tree

Description

Julia Equivalent: `IAI.get_regression_weights` (for regression or prescription tree learners as appropriate)

Usage

```
get_regression_weights(lnr, node_index, ...)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.
...	If a prescription problem, the treatment to query.

Examples

```
## Not run:  
iai::get_regression_weights(lnr, 1)  
iai::get_regression_weights(lnr, 1, "A")  
  
## End(Not run)
```

`get_rich_output_params`*Return the current global rich output parameter settings*

Description

Julia Equivalent: `IAI.get_rich_output_params`

Usage

```
get_rich_output_params()
```

Examples

```
## Not run: iai::get_rich_output_params()
```

`get_split_categories` *Return the categoric/ordinal information used in the split at a node of a tree*

Description

Julia Equivalent: `IAI.get_split_categories`

Usage

```
get_split_categories(lnr, node_index)
```

Arguments

<code>lnr</code>	The learner to query.
<code>node_index</code>	The node in the tree to query.

Examples

```
## Not run: iai::get_split_categories(lnr, 1)
```

get_split_feature *Return the feature used in the split at a node of a tree*

Description

Julia Equivalent: `IAI.get_split_feature`

Usage

```
get_split_feature(lnr, node_index)
```

Arguments

lnr The learner to query.
node_index The node in the tree to query.

Examples

```
## Not run: iai::get_split_feature(lnr, 1)
```

get_split_threshold *Return the threshold used in the split at a node of a tree*

Description

Julia Equivalent: `IAI.get_split_threshold`

Usage

```
get_split_threshold(lnr, node_index)
```

Arguments

lnr The learner to query.
node_index The node in the tree to query.

Examples

```
## Not run: iai::get_split_threshold(lnr, 1)
```

`get_split_weights` *Return the weights for numeric and categoric features used in the hyperplane split at a node of a tree*

Description

Julia Equivalent: `IAI.get_split_weights`

Usage

```
get_split_weights(lnr, node_index)
```

Arguments

`lnr` The learner to query.
`node_index` The node in the tree to query.

Examples

```
## Not run: iai::get_split_weights(lnr, 1)
```

`get_survival_curve` *Return the survival curve at a node of a tree*

Description

Julia Equivalent: `IAI.get_survival_curve`

Usage

```
get_survival_curve(lnr, node_index)
```

Arguments

`lnr` The learner to query.
`node_index` The node in the tree to query.

Examples

```
## Not run: iai::get_survival_curve(lnr, 1)
```

```
get_survival_curve_data
```

Extract the underlying data from a survival curve (as returned by [R](https://docs.interpretable.ai/v1.2.0/IAI-R/reference/#iai::predict) or [R](https://docs.interpretable.ai/v1.2.0/IAI-R/reference/#iai::get_survival_curve))

Description

The data is returned as a list with two keys: `times` containing the time for each breakpoint on the curve, and `coefs` containing the probability for each breakpoint on the curve.

Usage

```
get_survival_curve_data(curve)
```

Arguments

`curve` The curve to query.

Details

Julia Equivalent: `IAI.get_survival_curve_data`

Examples

```
## Not run: iai::get_survival_curve_data(curve)
```

```
get_upper_child
```

Get the index of the upper child at a split node of a tree

Description

Julia Equivalent: `IAI.get_upper_child`

Usage

```
get_upper_child(lnr, node_index)
```

Arguments

`lnr` The learner to query.
`node_index` The node in the tree to query.

Examples

```
## Not run: iai::get_upper_child(lnr, 1)
```

grid_search	<i>Controls grid search over parameter combinations</i>
-------------	---

Description

Julia Equivalent: [IAI.GridSearch](#)

Usage

```
grid_search(lnr, ...)
```

Arguments

lnr	The learner to use when validating.
...	The parameters to validate over.

Examples

```
grid <- iai::grid_search(
  iai::optimal_tree_classifier(
    random_seed = 1,
  ),
  max_depth = 1:5,
)
```

iai_setup	<i>Initialize Julia and the IAI package.</i>
-----------	--

Description

This needs to be done in every R session before calling 'iai' functions

Usage

```
iai_setup(...)
```

Arguments

...	All parameters are passed through to JuliaCall::julia_setup
-----	---

Examples

```
## Not run: iai::iai_setup()
```

imputation_learner *Generic learner for imputing missing values*

Description

Julia Equivalent: `IAI.ImputationLearner`

Usage

```
imputation_learner(method = "opt_knn", ...)
```

Arguments

method (optional) Specifies the imputation method to use.
... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::imputation_learner(method = "opt_tree")
```

impute *Impute missing values using either a specified method or through validation*

Description

Julia Equivalent: `IAI.impute`

Usage

```
impute(X, ...)
```

Arguments

X The dataframe in which to impute missing values.
... Refer to the Julia documentation for available parameters.

Examples

```
X <- iris
X[1, 1] <- NA
iai::impute(X)
```

impute_cv	<i>Impute missing values using cross validation</i>
-----------	---

Description

Julia Equivalent: `IAI.impute_cv`

Usage

```
impute_cv(X, ...)
```

Arguments

X	The dataframe in which to impute missing values.
...	Refer to the Julia documentation for available parameters.

Examples

```
X <- iris
X[1, 1] <- NA
iai::impute_cv(X, list(method = c("opt_knn", "opt_tree")))
```

is_categoric_split	<i>Check if a node of a tree applies a categoric split</i>
--------------------	--

Description

Julia Equivalent: `IAI.is_categoric_split`

Usage

```
is_categoric_split(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::is_categoric_split(lnr, 1)
```

is_hyperplane_split *Check if a node of a tree applies a hyperplane split*

Description

Julia Equivalent: [IAI.is_hyperplane_split](#)

Usage

```
is_hyperplane_split(lnr, node_index)
```

Arguments

lnr The learner to query.
node_index The node in the tree to query.

Examples

```
## Not run: iai::is_hyperplane_split(lnr, 1)
```

is_leaf *Check if a node of a tree is a leaf*

Description

Julia Equivalent: [IAI.is_leaf](#)

Usage

```
is_leaf(lnr, node_index)
```

Arguments

lnr The learner to query.
node_index The node in the tree to query.

Examples

```
## Not run: iai::is_leaf(lnr, 1)
```

```
is_mixed_ordinal_split
```

Check if a node of a tree applies a mixed ordinal/categoric split

Description

Julia Equivalent: `IAI.is_mixed_ordinal_split`

Usage

```
is_mixed_ordinal_split(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::is_mixed_ordinal_split(lnr, 1)
```

```
is_mixed_parallel_split
```

Check if a node of a tree applies a mixed parallel/categoric split

Description

Julia Equivalent: `IAI.is_mixed_parallel_split`

Usage

```
is_mixed_parallel_split(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::is_mixed_parallel_split(lnr, 1)
```

is_ordinal_split *Check if a node of a tree applies a ordinal split*

Description

Julia Equivalent: `IAI.is_ordinal_split`

Usage

```
is_ordinal_split(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::is_ordinal_split(lnr, 1)
```

is_parallel_split *Check if a node of a tree applies a parallel split*

Description

Julia Equivalent: `IAI.is_parallel_split`

Usage

```
is_parallel_split(lnr, node_index)
```

Arguments

lnr	The learner to query.
node_index	The node in the tree to query.

Examples

```
## Not run: iai::is_parallel_split(lnr, 1)
```

mean_imputation_learner
Learner for conducting mean imputation

Description

Julia Equivalent: `IAI.MeanImputationLearner`

Usage

```
mean_imputation_learner(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::mean_imputation_learner()
```

missing_goes_lower *Check if points with missing values go to the lower child at a split node of a tree*

Description

Julia Equivalent: `IAI.missing_goes_lower`

Usage

```
missing_goes_lower(lnr, node_index)
```

Arguments

lnr The learner to query.
node_index The node in the tree to query.

Examples

```
## Not run: iai::missing_goes_lower(lnr, 1)
```

multi_questionnaire	<i>Construct an interactive questionnaire using multiple tree learners as specified by questions</i>
---------------------	--

Description

Julia Equivalent: `IAI.MultiQuestionnaire`

Usage

```
multi_questionnaire(questions)
```

Arguments

questions The questions to visualize. Refer to the [Julia documentation on multi-learner visualizations](#) for more information.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run:
iai::multi_questionnaire(list("Questionnaire for" = list(
  "first learner" = lnr1,
  "second learner" = lnr2
)))

## End(Not run)
```

multi_tree_plot	<i>Construct an interactive tree visualization of multiple tree learners as specified by questions</i>
-----------------	--

Description

Julia Equivalent: `IAI.MultiTreePlot`

Usage

```
multi_tree_plot(questions)
```

Arguments

questions The questions to visualize. Refer to the [Julia documentation on multi-learner visualizations](#) for more information.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run:
iai::multi_tree_plot(list("Visualizing" = list(
  "first learner" = lnr1,
  "second learner" = lnr2
)))

## End(Not run)
```

optimal_feature_selection_classifier

Learner for conducting Optimal Feature Selection on classification problems

Description

Julia Equivalent: `IAI.OptimalFeatureSelectionClassifier`

Usage

```
optimal_feature_selection_classifier(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run: lnr <- iai::optimal_feature_selection_classifier()
```

optimal_feature_selection_regressor

Learner for conducting Optimal Feature Selection on regression problems

Description

Julia Equivalent: `IAI.OptimalFeatureSelectionRegressor`

Usage

```
optimal_feature_selection_regressor(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run: lnr <- iai::optimal_feature_selection_regressor()
```

optimal_tree_classifier

Learner for training Optimal Classification Trees

Description

Julia Equivalent: `IAI.OptimalTreeClassifier`

Usage

```
optimal_tree_classifier(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::optimal_tree_classifier()
```

optimal_tree_prescription_maximizer

Learner for training Optimal Prescriptive Trees where the prescriptions should aim to maximize outcomes

Description

Julia Equivalent: `IAI.OptimalTreePrescriptionMaximizer`

Usage

```
optimal_tree_prescription_maximizer(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::optimal_tree_prescription_maximizer()
```

optimal_tree_prescription_minimizer

Learner for training Optimal Prescriptive Trees where the prescriptions should aim to minimize outcomes

Description

Julia Equivalent: `IAI.OptimalTreePrescriptionMinimizer`

Usage

```
optimal_tree_prescription_minimizer(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::optimal_tree_prescription_minimizer()
```

`optimal_tree_regressor`*Learner for training Optimal Regression Trees*

Description

Julia Equivalent: `IAI.OptimalTreeRegressor`

Usage

```
optimal_tree_regressor(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::optimal_tree_regressor()
```

`optimal_tree_survivor` *Learner for training Optimal Survival Trees*

Description

Julia Equivalent: `IAI.OptimalTreeSurvivor`

Usage

```
optimal_tree_survivor(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::optimal_tree_survivor()
```

`opt_knn_imputation_learner`*Learner for conducting optimal k-NN imputation*

Description

Julia Equivalent: `IAI.OptKNNImputationLearner`

Usage`opt_knn_imputation_learner(...)`**Arguments**

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::opt_knn_imputation_learner()
```

`opt_svm_imputation_learner`*Learner for conducting optimal SVM imputation*

Description

Julia Equivalent: `IAI.OptSVMImputationLearner`

Usage`opt_svm_imputation_learner(...)`**Arguments**

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::opt_svm_imputation_learner()
```

opt_tree_imputation_learner
Learner for conducting optimal tree-based imputation

Description

Julia Equivalent: `IAI.OptTreeImputationLearner`

Usage

```
opt_tree_imputation_learner(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::opt_tree_imputation_learner()
```

predict *Return the predictions made by the model for each point in the features*

Description

Julia Equivalent: `IAI.predict`

Usage

```
predict(lnr, X)
```

Arguments

lnr The learner or grid to use for prediction.
X The features of the data.

Examples

```
## Not run: iai::predict(lnr, X)
```

predict_hazard	<i>Return the fitted hazard coefficient estimate made by a model for each point in the features.</i>
----------------	--

Description

A higher hazard coefficient estimate corresponds to a smaller predicted survival time.

Usage

```
predict_hazard(lnr, X)
```

Arguments

lnr	The learner or grid to use for prediction.
X	The features of the data.

Details

Julia Equivalent: `IAI.predict_hazard`

IAI Compatibility

Requires IAI version 1.2 or higher.

Examples

```
## Not run: iai::predict_hazard(lnr, X)
```

predict_outcomes	<i>Return the the predicted outcome for each treatment made by a model for each point in the features</i>
------------------	---

Description

Julia Equivalent: `IAI.predict_outcomes`

Usage

```
predict_outcomes(lnr, X)
```

Arguments

lnr	The learner or grid to use for prediction.
X	The features of the data.

Examples

```
## Not run: iai::predict_outcomes(lnr, X)
```

predict_proba	<i>Return the probabilities of class membership predicted by a model for each point in the features</i>
---------------	---

Description

Julia Equivalent: `IAI.predict_proba`

Usage

```
predict_proba(lnr, X)
```

Arguments

lnr	The learner or grid to use for prediction.
X	The features of the data.

Examples

```
## Not run: iai::predict_proba(lnr, X)
```

print_path	<i>Print the decision path through the learner for each sample in the features</i>
------------	--

Description

Julia Equivalent: `IAI.print_path`

Usage

```
print_path(lnr, X, ...)
```

Arguments

lnr	The learner or grid to query.
X	The features of the data.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run:  
iai::print_path(lnr, X)  
iai::print_path(lnr, X, 1)  
  
## End(Not run)
```

questionnaire

Specify an interactive questionnaire of a tree learner

Description

Julia Equivalent: [IAI.Questionnaire](#)

Usage

```
questionnaire(lnr, ...)
```

Arguments

lnr The learner to visualize.
... Refer to the [Julia documentation](#) for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run: iai::questionnaire(lnr)
```

rand_imputation_learner

Learner for conducting random imputation

Description

Julia Equivalent: [IAI.RandImputationLearner](#)

Usage

```
rand_imputation_learner(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::rand_imputation_learner()
```

read_json	<i>Read in a learner or grid saved in JSON format</i>
-----------	---

Description

Julia Equivalent: `IAI.read_json`

Usage

```
read_json(filename)
```

Arguments

filename The location of the JSON file.

Examples

```
## Not run: obj <- iai::read_json("out.json")
```

reset_display_label	<i>Reset the predicted probability displayed to be that of the predicted label when visualizing a learner</i>
---------------------	---

Description

Julia Equivalent: `IAI.reset_display_label!`

Usage

```
reset_display_label(lnr)
```

Arguments

lnr The learner to modify.

Examples

```
## Not run: iai::reset_display_label(lnr)
```

`roc_curve`*Construct an ROC curve using a trained model on the given data*

DescriptionJulia Equivalent: `IAI.ROCCurve`**Usage**`roc_curve(lnr, X, y)`**Arguments**

<code>lnr</code>	The learner or grid to use for prediction.
<code>X</code>	The features of the data.
<code>y</code>	The labels of the data.

Examples

```
## Not run: iai::roc_curve(lnr, X, y)
```

`score`*Calculate the score for a model on the given data*

DescriptionJulia Equivalent: `IAI.score`**Usage**`score(lnr, X, ...)`**Arguments**

<code>lnr</code>	The learner or grid to evaluate.
<code>X</code>	The features of the data.
<code>...</code>	Other parameters, including zero or more target vectors as required by the problem type. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::score(lnr, X, y)
```

set_display_label	<i>Show the probability of a specified label when visualizing a learner</i>
-------------------	---

Description

Julia Equivalent: `IAI.set_display_label!`

Usage

```
set_display_label(lnr, display_label)
```

Arguments

lnr The learner to modify.
display_label The label for which to show probabilities.

Examples

```
## Not run: iai::set_display_label(lnr, "A")
```

set_julia_seed	<i>Set the random seed in Julia</i>
----------------	-------------------------------------

Description

Julia Equivalent: `Random.seed!`

Usage

```
set_julia_seed(seed)
```

Arguments

seed The seed to set

Examples

```
## Not run: iai::set_julia_seed(1)
```

set_params *Set all supplied parameters on a learner*

Description

Julia Equivalent: `IAI.set_params!`

Usage

```
set_params(lnr, ...)
```

Arguments

lnr	The learner to modify.
...	The parameters to set on the learner.

Examples

```
## Not run: iai::set_params(lnr, random_seed = 1)
```

set_rich_output_param *Sets a global rich output parameter*

Description

Julia Equivalent: `IAI.set_rich_output_param!`

Usage

```
set_rich_output_param(key, value)
```

Arguments

key	The parameter to set.
value	The value to set

Examples

```
## Not run: iai::set_rich_output_param("simple_layout", TRUE)
```

set_threshold	<i>For a binary classification problem, update the the predicted labels in the leaves of the learner to predict a label only if the predicted probability is at least the specified threshold.</i>
---------------	--

Description

Julia Equivalent: `IAI.set_threshold!`

Usage

```
set_threshold(lnr, label, threshold, ...)
```

Arguments

lnr	The learner to modify.
label	The referenced label.
threshold	The probability threshold above which label will be be predicted.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::set_threshold(lnr, "A", 0.4)
```

show_in_browser	<i>Show interactive visualization of an object (such as a learner or curve) in the default browser</i>
-----------------	--

Description

Julia Equivalent: `IAI.show_in_browser`

Usage

```
show_in_browser(obj, ...)
```

Arguments

obj	The object to visualize.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::show_in_browser(lnr)
```

show_questionnaire *Show an interactive questionnaire based on a learner in default browser*

Description

Julia Equivalent: `IAI.show_questionnaire`

Usage

```
show_questionnaire(lnr, ...)
```

Arguments

lnr The learner to visualize.
... Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::show_questionnaire(lnr)
```

single_knn_imputation_learner
Learner for conducting heuristic k-NN imputation

Description

Julia Equivalent: `IAI.SingleKNNImputationLearner`

Usage

```
single_knn_imputation_learner(...)
```

Arguments

... Use keyword arguments to set parameters on the resulting learner. Refer to the Julia documentation for available parameters.

Examples

```
## Not run: lnr <- iai::single_knn_imputation_learner()
```

split_data	<i>Split the data into training and test datasets</i>
------------	---

Description

Julia Equivalent: `IAI.split_data`

Usage

```
split_data(task, X, ...)
```

Arguments

task	The type of problem.
X	The features of the data.
...	Other parameters, including zero or more target vectors as required by the problem type. Refer to the Julia documentation for available parameters.

Examples

```
X <- iris[, 1:4]
y <- iris$Species
split <- iai::split_data("classification", X, y, train_proportion = 0.75)
train_X <- split$train$X
train_y <- split$train$y
test_X <- split$test$X
test_y <- split$test$y
```

transform	<i>Impute missing values in a dataframe using a fitted imputation model</i>
-----------	---

Description

Julia Equivalent: `IAI.transform`

Usage

```
transform(lnr, X)
```

Arguments

lnr	The learner or grid to use for imputation
X	The features of the data.

Examples

```
## Not run: iai::transform(lnr, X)
```

tree_plot	<i>Specify an interactive tree visualization of a tree learner</i>
-----------	--

Description

Julia Equivalent: `IAI.TreePlot`

Usage

```
tree_plot(lnr, ...)
```

Arguments

lnr	The learner to visualize.
...	Refer to the Julia documentation on advanced tree visualization for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```
## Not run: iai::tree_plot(lnr)
```

variable_importance	<i>Generate a ranking of the variables in the learner according to their importance when training the trees</i>
---------------------	---

Description

Julia Equivalent: `IAI.variable_importance`

Usage

```
variable_importance(lnr)
```

Arguments

lnr	The learner to query.
-----	-----------------------

Examples

```
## Not run: iai::variable_importance(lnr)
```

write_dot	<i>Output a learner in R dot format</i>
-----------	---

Description

Julia Equivalent: `IAI.write_dot`

Usage

```
write_dot(filename, lnr, ...)
```

Arguments

filename	Where to save the output.
lnr	The learner to output.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::write_dot(file.path(tempdir(), "tree.dot"), lnr)
```

write_html	<i>Output a learner as an interactive browser visualization in HTML format</i>
------------	--

Description

Julia Equivalent: `IAI.write_html`

Usage

```
write_html(filename, lnr, ...)
```

Arguments

filename	Where to save the output.
lnr	The learner to output.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::write_html(file.path(tempdir(), "tree.html"), lnr)
```

write_json *Output a learner or grid in JSON format*

Description

Julia Equivalent: `IAI.write_json`

Usage

```
write_json(filename, obj, ...)
```

Arguments

filename	Where to save the output.
obj	The learner or grid to output.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::write_json(file.path(tempdir(), "out.json"), obj)
```

write_png *Output a learner as a PNG image*

Description

Julia Equivalent: `IAI.write_png`

Usage

```
write_png(filename, lnr, ...)
```

Arguments

filename	Where to save the output.
lnr	The learner to output.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::write_png(file.path(tempdir(), "tree.png"), lnr)
```

write_questionnaire *Output a learner as an interactive questionnaire in HTML format*

Description

Julia Equivalent: `IAI.write_questionnaire`

Usage

```
write_questionnaire(filename, lnr, ...)
```

Arguments

filename	Where to save the output.
lnr	The learner to output.
...	Refer to the Julia documentation for available parameters.

Examples

```
## Not run: iai::write_questionnaire(file.path(tempdir(), "questionnaire.html"), lnr)
```

Index

apply, 3
apply_nodes, 4
as.mixeddata, 5

clone, 5

decision_path, 6
delete_rich_output_param, 6

fit, 7
fit_cv, 7
fit_transform, 8
fit_transform_cv, 9

get_best_params, 9
get_classification_label, 10
get_classification_proba, 10
get_depth, 11
get_grid_results, 11
get_learner, 12
get_lower_child, 12
get_num_nodes, 13
get_num_samples, 13
get_params, 14
get_parent, 14
get_prediction_constant, 15
get_prediction_weights, 15
get_prescription_treatment_rank, 16
get_regression_constant, 16
get_regression_weights, 17
get_rich_output_params, 18
get_split_categories, 18
get_split_feature, 19
get_split_threshold, 19
get_split_weights, 20
get_survival_curve, 20
get_survival_curve_data, 21
get_upper_child, 21
grid_search, 22

iaai_setup, 22

imputation_learner, 23
impute, 23
impute_cv, 24
is_categoric_split, 24
is_hyperplane_split, 25
is_leaf, 25
is_mixed_ordinal_split, 26
is_mixed_parallel_split, 26
is_ordinal_split, 27
is_parallel_split, 27

mean_imputation_learner, 28
missing_goes_lower, 28
multi_questionnaire, 29
multi_tree_plot, 29

opt_knn_imputation_learner, 34
opt_svm_imputation_learner, 34
opt_tree_imputation_learner, 35
optimal_feature_selection_classifier, 30
optimal_feature_selection_regressor, 31
optimal_tree_classifier, 31
optimal_tree_prescription_maximizer, 32
optimal_tree_prescription_minimizer, 32
optimal_tree_regressor, 33
optimal_tree_survivor, 33

predict, 35
predict_hazard, 36
predict_outcomes, 36
predict_proba, 37
print_path, 37

questionnaire, 38

rand_imputation_learner, 38
read_json, 39

reset_display_label, 39
roc_curve, 40

score, 40
set_display_label, 41
set_julia_seed, 41
set_params, 42
set_rich_output_param, 42
set_threshold, 43
show_in_browser, 43
show_questionnaire, 44
single_knn_imputation_learner, 44
split_data, 45

transform, 45
tree_plot, 46

variable_importance, 46

write_dot, 47
write_html, 47
write_json, 48
write_png, 48
write_questionnaire, 49