Package ‘iai’

October 13, 2022

Type Package

Title Interface to 'Interpretable AI' Modules

Version 1.8.0

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 (>= 0.17.4), stringr, rlang, lifecycle, rappdirs,
ggplot2, cowplot, rjson

RoxygenNote 7.1.2

Suggests testthat, covr, xml2, withr

NeedsCompilation no

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

Maintainer Jack Dunn <jack@interpretable.ai>

Repository CRAN

Date/Publication 2022-08-29 20:50:02 UTC

R topics documented:

acquire_license .......................................................... 6
add_julia_processes ...................................................... 7
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_treatment_combinations</td>
</tr>
<tr>
<td>apply</td>
</tr>
<tr>
<td>apply_nodes</td>
</tr>
<tr>
<td>as.mixeddata</td>
</tr>
<tr>
<td>autoplot.grid_search</td>
</tr>
<tr>
<td>autoplot.roc_curve</td>
</tr>
<tr>
<td>autoplot.similarity_comparison</td>
</tr>
<tr>
<td>autoplot.stability_analysis</td>
</tr>
<tr>
<td>categorical_classification_reward_estimator</td>
</tr>
<tr>
<td>categorical_regression_reward_estimator</td>
</tr>
<tr>
<td>categorical_reward_estimator</td>
</tr>
<tr>
<td>categorical_survival_reward_estimator</td>
</tr>
<tr>
<td>cleanup_installation</td>
</tr>
<tr>
<td>clone</td>
</tr>
<tr>
<td>convert_treatments_to_numeric</td>
</tr>
<tr>
<td>copy_splits_and_refit_leaves</td>
</tr>
<tr>
<td>decision_path</td>
</tr>
<tr>
<td>delete_rich_output_param</td>
</tr>
<tr>
<td>equal_propensity_estimator</td>
</tr>
<tr>
<td>fit</td>
</tr>
<tr>
<td>fit.grid_search</td>
</tr>
<tr>
<td>fit.imputation_learner</td>
</tr>
<tr>
<td>fit.learner</td>
</tr>
<tr>
<td>fit.optimal_feature_selection_learner</td>
</tr>
<tr>
<td>fit_and_expand</td>
</tr>
<tr>
<td>fit_cv</td>
</tr>
<tr>
<td>fit_predict</td>
</tr>
<tr>
<td>fit_predict.categorical_reward_estimator</td>
</tr>
<tr>
<td>fit_predict.numeric_reward_estimator</td>
</tr>
<tr>
<td>fit_transform</td>
</tr>
<tr>
<td>fit_transform_cv</td>
</tr>
<tr>
<td>get_best_params</td>
</tr>
<tr>
<td>get_classification_label</td>
</tr>
<tr>
<td>get_classification_proba</td>
</tr>
<tr>
<td>get_cluster_assignments</td>
</tr>
<tr>
<td>get_cluster_details</td>
</tr>
<tr>
<td>get_cluster_distances</td>
</tr>
<tr>
<td>get_depth</td>
</tr>
<tr>
<td>get_estimation_densities</td>
</tr>
<tr>
<td>get_features_used</td>
</tr>
<tr>
<td>get_grid_results</td>
</tr>
<tr>
<td>get_grid_result_details</td>
</tr>
<tr>
<td>get_grid_result_summary</td>
</tr>
<tr>
<td>get_learner</td>
</tr>
<tr>
<td>get_lower_child</td>
</tr>
<tr>
<td>get_machine_id</td>
</tr>
<tr>
<td>get_num_fits</td>
</tr>
<tr>
<td>get_num_fits.glmnetcv_learner</td>
</tr>
</tbody>
</table>
R topics documented:

- get_num_fits.optimal_feature_selection_learner ........................................... 35
- get_num_nodes ......................................................................................... 36
- get_num_samples ....................................................................................... 36
- get_params ................................................................................................. 37
- get_parent ................................................................................................. 37
- get_policy_treatment_outcome .................................................................... 38
- get_policy_treatment_rank ......................................................................... 38
- get_prediction_constant ............................................................................ 39
- get_prediction_constant.glmnetcv_learner .................................................. 39
- get_prediction_constant.optimal_feature_selection_learner ....................... 40
- get_prediction_weights .............................................................................. 41
- get_prediction_weights.glmnetcv_learner .................................................... 41
- get_prediction_weights.optimal_feature_selection_learner ....................... 42
- get_prescription_treatment_rank ................................................................ 42
- get_regression_constant ............................................................................ 43
- get_regression_constant.classification_tree_learner .................................... 43
- get_regression_constant.prescription_tree_learner ...................................... 44
- get_regression_constant.regression_tree_learner ......................................... 45
- get_regression_constant.survival_tree_learner ........................................... 45
- get_regression_weights .............................................................................. 46
- get_regression_weights.classification_tree_learner .................................... 46
- get_regression_weights.prescription_tree_learner ...................................... 47
- get_regression_weights.regression_tree_learner ......................................... 48
- get_regression_weights.survival_tree_learner ........................................... 48
- get_rich_output_params ............................................................................. 49
- get_roc_curve_data ...................................................................................... 49
- get_split_categories ................................................................................... 50
- get_split_feature ....................................................................................... 50
- get_split_threshold .................................................................................... 51
- get_split_weights ...................................................................................... 51
- get_stability_results .................................................................................. 52
- get_survival_curve ...................................................................................... 52
- get_survival_curve_data ............................................................................. 53
- get_survival_expected_time ...................................................................... 53
- get_survival_hazard .................................................................................... 54
- get_train_errors ......................................................................................... 55
- get_tree ..................................................................................................... 55
- get_upper_child .......................................................................................... 56
- glmnetcv_classifier ..................................................................................... 56
- glmnetcv_regressor ...................................................................................... 57
- glmnetcv_survival_learner .......................................................................... 57
- grid_search .................................................................................................. 58
- iai_setup ...................................................................................................... 59
- imputation_learner ...................................................................................... 59
- impute ......................................................................................................... 60
- impute_cv .................................................................................................... 60
- install_julia .................................................................................................. 61
- install_system_image .................................................................................. 61
R topics documented:

is_categoric_split .................................................. 62
is_hyperplane_split .................................................. 63
is_leaf ................................................................. 63
is_mixed_ordinal_split ................................................. 64
is_mixed_parallel_split .............................................. 64
is_ordinal_split ......................................................... 65
is_parallel_split ....................................................... 65
load_graphviz .......................................................... 66
mean_imputation_learner .............................................. 66
missing_goes_lower ..................................................... 67
multi_questionnaire ................................................... 67
multi_questionnaire.default ......................................... 68
multi_questionnaire.grid_search .................................... 69
multi_tree_plot ........................................................ 69
multi_tree_plot.default .............................................. 70
multi_tree_plot.grid_search ......................................... 71
numeric_classification_reward_estimator ......................... 71
numeric_regression_reward_estimator .............................. 72
numeric_reward_estimator ............................................ 73
numeric_survival_reward_estimator ................................ 73
optimal_feature_selection_classifier ................................ 74
optimal_feature_selection_regressor ................................ 75
optimal_tree_classifier ................................................. 75
optimal_tree_policy_maximizer ...................................... 76
optimal_tree_policy_minimizer ...................................... 76
optimal_tree_prescription_maximizer ................................. 77
optimal_tree_prescription_minimizer ................................. 77
optimal_tree_regressor ................................................ 78
optimal_tree_survival_learner ....................................... 78
optimal_tree_survivor ................................................... 79
opt_knn_imputation_learner .......................................... 79
opt_svm_imputation_learner .......................................... 80
opt_tree_imputation_learner .......................................... 80
plot.grid_search ........................................................ 81
plot.roc_curve .......................................................... 81
plot.similarity_comparison ............................................ 82
plot.stability_analysis ............................................... 82
predict ................................................................. 83
predict.categorical_reward_estimator ............................... 83
predictglmnetcv_learner .............................................. 84
predict.numeric_reward_estimator .................................. 85
predict.optimal_feature_selection_learner ......................... 85
predict.supervised_learner .......................................... 86
predict.survival_learner .............................................. 87
predict_expected_survival_time ..................................... 87
predict_expected_survival_time.glmnetcv_survival_learner .... 88
predict_expected_survival_time.survival_curve ..................... 88
predict_expected_survival_time.survival_learner .................. 89
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>predict_hazard</td>
<td>90</td>
</tr>
<tr>
<td>predict_hazard.glmmnetcv_survival_learner</td>
<td>90</td>
</tr>
<tr>
<td>predict_hazard.survival_learner</td>
<td>91</td>
</tr>
<tr>
<td>predict_outcomes</td>
<td>92</td>
</tr>
<tr>
<td>predict_outcomes.policy_learner</td>
<td>92</td>
</tr>
<tr>
<td>predict_outcomes.prescription_learner</td>
<td>93</td>
</tr>
<tr>
<td>predict_proba</td>
<td>93</td>
</tr>
<tr>
<td>predict_proba.classification_learner</td>
<td>94</td>
</tr>
<tr>
<td>predict_proba.glmmnetcv_classifier</td>
<td>94</td>
</tr>
<tr>
<td>predict_reward</td>
<td>95</td>
</tr>
<tr>
<td>predict_reward.categorical_reward_estimator</td>
<td>95</td>
</tr>
<tr>
<td>predict_reward.numeric_reward_estimator</td>
<td>96</td>
</tr>
<tr>
<td>predict_shap</td>
<td>97</td>
</tr>
<tr>
<td>predict_treatment_outcome</td>
<td>97</td>
</tr>
<tr>
<td>predict_treatment_rank</td>
<td>98</td>
</tr>
<tr>
<td>print_path</td>
<td>98</td>
</tr>
<tr>
<td>prune_trees</td>
<td>99</td>
</tr>
<tr>
<td>questionnaire</td>
<td>100</td>
</tr>
<tr>
<td>questionnaire.optimal_feature_selection_learner</td>
<td>100</td>
</tr>
<tr>
<td>questionnaire.tree_learner</td>
<td>101</td>
</tr>
<tr>
<td>random_forest_classifier</td>
<td>101</td>
</tr>
<tr>
<td>random_forest_regressor</td>
<td>102</td>
</tr>
<tr>
<td>random_forest_survival_learner</td>
<td>102</td>
</tr>
<tr>
<td>rand_imputation_learner</td>
<td>103</td>
</tr>
<tr>
<td>read_json</td>
<td>103</td>
</tr>
<tr>
<td>refit_leaves</td>
<td>104</td>
</tr>
<tr>
<td>release_license</td>
<td>104</td>
</tr>
<tr>
<td>reset_display_label</td>
<td>105</td>
</tr>
<tr>
<td>resume_from_checkpoint</td>
<td>105</td>
</tr>
<tr>
<td>reward_estimator</td>
<td>106</td>
</tr>
<tr>
<td>roc_curve</td>
<td>106</td>
</tr>
<tr>
<td>roc_curve.classification_learner</td>
<td>107</td>
</tr>
<tr>
<td>roc_curve.default</td>
<td>107</td>
</tr>
<tr>
<td>roc_curve.glmmnetcv_classifier</td>
<td>108</td>
</tr>
<tr>
<td>score</td>
<td>109</td>
</tr>
<tr>
<td>score.categorical_reward_estimator</td>
<td>109</td>
</tr>
<tr>
<td>score.default</td>
<td>110</td>
</tr>
<tr>
<td>score.glmmnetcv_learner</td>
<td>110</td>
</tr>
<tr>
<td>score.numeric_reward_estimator</td>
<td>111</td>
</tr>
<tr>
<td>score.optimal_feature_selection_learner</td>
<td>112</td>
</tr>
<tr>
<td>score.supervised_learner</td>
<td>112</td>
</tr>
<tr>
<td>set_display_label</td>
<td>113</td>
</tr>
<tr>
<td>set_julia_seed</td>
<td>113</td>
</tr>
<tr>
<td>set_params</td>
<td>114</td>
</tr>
<tr>
<td>set_reward_kernel_bandwidth</td>
<td>114</td>
</tr>
<tr>
<td>set_rich_output_param</td>
<td>115</td>
</tr>
<tr>
<td>set_threshold</td>
<td>115</td>
</tr>
<tr>
<td>show_in_browser</td>
<td>116</td>
</tr>
</tbody>
</table>
acquire_license

Acquire an IAI license for the current session.

Julia Equivalent: IAI.acquire_license
add_julia_processes

Usage

acquire_license(...)

Arguments

... Refer to the Julia documentation for available parameters

IAI Compatibility

Requires IAI version 3.1 or higher.

Examples

## Not run: iai::acquire_license()

---

add_julia_processes Add additional Julia worker processes to parallelize workloads

Description

Julia Equivalent: Distributed.addprocs!

Usage

add_julia_processes(...)

Arguments

... Refer to the Julia documentation for available parameters

Details

For more information, refer to the documentation on parallelization

Examples

## Not run: iai::add_julia_processes(3)
all_treatment_combinations

Return a dataframe containing all treatment combinations of one or more treatment vectors, ready for use as treatment candidates in 'fit_predict' or 'predict'.

Description
Julia Equivalent: IAI.all_treatment_combinations

Usage
all_treatment_combinations(...)

Arguments
... A vector of possible options for each treatment

Examples
## Not run: iai::all_treatment_combinations(c(1, 2, 3))

apply

Return the leaf index in a tree model into which each point in the features falls

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**

Return the indices of the points in the features that fall into each node of a trained tree model.

**Description**

Julia Equivalent: `IAI.apply_nodes`

**Usage**

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

**Arguments**

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

**Examples**

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

---

**as.mixeddata**

Convert a vector of values to IAI mixed data format.

**Description**

Julia Equivalent: `IAI.make_mixed_data`

**Usage**

```r
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

```r
## Not run:
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"))

## End(Not run)
```

**Description**

Construct a `ggplot2::ggplot` object plotting grid search results for Optimal Feature Selection learners.

**Usage**

```r
## S3 method for class 'grid_search'
autoplot(x, type = stop("\'type\' is required"), ...)
```

**Arguments**

- `x`: The grid search to plot
- `type`: The type of plot to construct (either "validation" or "importance", for more information refer to the Julia documentation for plotting grid search results)
- `...`: Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: ggplot2::autoplot(grid)
```
## autoplot.roc_curve

**Description**

Construct a \texttt{ggplot2::ggplot} object plotting the ROC curve

**Usage**

```r
## S3 method for class 'roc_curve'
autoplot(x, ...)
```

**Arguments**

- \texttt{x}
  
  The ROC curve to plot

- \texttt{...}
  
  Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.1 or higher.

**Examples**

```r
## Not run: ggplot2::autoplot(roc)
```

## autoplot.similarity_comparison

**Description**

Construct a \texttt{ggplot2::ggplot} object plotting the results of the similarity comparison

**Usage**

```r
## S3 method for class 'similarity_comparison'
autoplot(x, ...)
```

**Arguments**

- \texttt{x}
  
  The similarity comparison to plot

- \texttt{...}
  
  Additional arguments (unused)
**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: ggplot2::autoplot(similarity)
```

---

**autoplot.stability_analysis**

*Construct a* [ggplot2::ggplot](https://ggplot2.tidyverse.org/reference/ggplot.html)* object plotting the results of the stability analysis*

**Description**

Construct a `ggplot2::ggplot` object plotting the results of the stability analysis

**Usage**

```r
## S3 method for class 'stability_analysis'
autoplot(x, ...)
```

**Arguments**

- `x` The stability analysis to plot
- `...` Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: ggplot2::autoplot(stability)
```
categorical_classification_reward_estimator

Learner for conducting reward estimation with categorical treatments and classification outcomes

Description

Julia Equivalent: IAI.CategoricalClassificationRewardEstimator

Usage

categorical_classification_reward_estimator(...) 

Arguments

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

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: lnr <- iai::categorical_classification_reward_estimator()

categorical_regression_reward_estimator

Learner for conducting reward estimation with categorical treatments and regression outcomes

Description

Julia Equivalent: IAI.CategoricalRegressionRewardEstimator

Usage

categorical_regression_reward_estimator(...) 

Arguments

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

Requires IAI version 2.2 or higher.

Examples

## Not run: lnr <- iai::categorical_regression_reward_estimator()

categorical_reward_estimator

Learner for conducting reward estimation with categorical treatments

Description

This function was deprecated in iai 1.6.0, and [categorical_classification_reward_estimator()] or [categorical_classification_reward_estimator()] should be used instead.

Usage

categorical_reward_estimator(...)

Arguments

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

Details

This deprecation is no longer supported as of the IAI v3 release.

IAI Compatibility

Requires IAI version 2.0, 2.1 or 2.2.

Examples

## Not run: lnr <- iai::categorical_reward_estimator()
categorical_survival_reward_estimator

Learner for conducting reward estimation with categorical treatments and survival outcomes

Description

Julia Equivalent: IAI.CategoricalSurvivalRewardEstimator

Usage

categorical_survival_reward_estimator(...)

Arguments

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

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: lnr <- iai::categorical_survival_reward_estimator()

cleanup_installation  Remove all traces of automatic Julia/IAI installation

Description

Removes files created by install_julia and install_system_image

Usage

cleanup_installation()

Examples

## Not run: iai::cleanup_installation()
clone

Return an unfitted copy of a learner with the same parameters

Description

Julia Equivalent: `IAI.clone`

Usage

clonelnr)

Arguments

lnr The learner to copy.

Examples

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

convert_treatments_to_numeric

Convert 'treatments' from symbol/string format into numeric values.

Description

Julia Equivalent: `IAI.convert_treatments_to_numeric`

Usage

convert_treatments_to_numeric(treatments)

Arguments

treatments The treatments to convert

Examples

## Not run: iai::convert_treatments_to_numeric(c("1", "2", "3"))
copy_splits_and_refit_leaves

*Copy the tree split structure from one learner into another and refit the models in each leaf of the tree using the supplied data*

**Description**

Julia Equivalent: `IAI.copy_splits_and_refit_leaves!`

**Usage**

```julia
copy_splits_and_refit_leaves(new_lnr, orig_lnr, ...)
```

**Arguments**

- `new_lnr` The learner to modify and refit
- `orig_lnr` The learner from which to copy the tree split structure
- `...` Refer to the Julia documentation for available parameters

**IAI Compatibility**

Requires IAI version 3.0 or higher.

**Examples**

```julia
## Not run: iai::copy_splits_and_refit_leaves(new_lnr, orig_lnr, ...)
```

decision_path

*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.*

**Description**

Julia Equivalent: `IAI.decision_path`

**Usage**

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

**Arguments**

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

Delete a global rich output parameter

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")

equal_propensity_estimator

Learner that estimates equal propensity for all treatments.

Description

For use with data from randomized experiments where treatments are known to be randomly assigned.

Usage

equal_propensity_estimator(...)  

Arguments

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

Details

Julia Equivalent: `IAI.EqualPropensityEstimator`
IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: lnr <- iai::equal_propensity_estimator()
```

---

**fit**

*Generic function for fitting a learner.*

**Description**

Generic function for fitting a learner.

**Usage**

```r
fit(obj, ...)
```

**Arguments**

- `obj`: The object controlling which method is used
- `...`: Arguments depending on the specific method used

**fit.grid_search**

*Fits a grid_search to the training data*

**Description**

Julia Equivalent: `IAI.fit!`

**Usage**

```r
## S3 method for class 'grid_search'
fit(obj, X, ...)
```

**Arguments**

- `obj`: The grid search 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.
fit.imputation_learner

Fits an imputation learner to the training data.

Description

Additional keyword arguments are available for fitting imputation learners - please refer to the Julia documentation.

Usage

## S3 method for class 'imputation_learner'
fit(obj, X, ...)

Arguments

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

Details

Julia Equivalent: IAI.fit!

Examples

## Not run: iai::fit(lnr, X)
**fit.learner**  
*Fits a model to the training data*

### Description

Julia Equivalent: `IAI.fit!`

### Usage

```r
## S3 method for class 'learner'
fit(obj, X, ...)
```

### Arguments

- `obj`: The learner 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

```r
## Not run:
X <- iris[, 1:4]
y <- iris$Species
lnr <- iai::random_forest_classifier()
iai::fit(lnr, X, y)
## End(Not run)
```

---

**fit.optimal_feature_selection_learner**  
*Fits an Optimal Feature Selection learner to the training data*

### Description

When the `coordinated_sparsity` parameter of the learner is `TRUE`, additional keyword arguments are required - please refer to the Julia documentation.

### Usage

```r
## S3 method for class 'optimal_feature_selection_learner'
fit(obj, X, ...)
```
Arguments

- `obj` 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.

Details

Julia Equivalent: `IAI.fit!`

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```r
## Not run: iai::fit(lnr, X)
```

---

**fit_and_expand**

Fit an imputation learner with training features and create adaptive indicator features to encode the missing pattern

Description

Julia Equivalent: `IAI.fit_and_expand!`

Usage

```r
fit_and_expand(lnr, X, ...)
```

Arguments

- `lnr` The learner to use for imputation.
- `X` The dataframe in which to impute missing values.
- `...` Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```r
## Not run: lnr <- iai::fit_and_expand(lnr, X, type = "finite")
```
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

```r
## Not run:
X <- iris[, 1:4]
y <- iris$Species
grid <- iai::grid_search(
    iai::optimal_tree_classifier(max_depth = 1),
)
aii::fit_cv(grid, X, y)
## End(Not run)
```

fit_predict

Generic function for fitting a reward estimator on features, treatments and returning predicted counterfactual rewards and scores of the internal estimators.

Description

Julia Equivalent: `IAI.fit_predict`!

Usage

`fit_predict(obj, ...)`

Arguments

- `obj` The object controlling which method is used
- `...` Arguments depending on the specific method used
fit_predict.categorical_reward_estimator

Fit a categorical reward estimator on features, treatments and outcomes and return predicted counterfactual rewards for each observation, under each treatment observed in the data, as well as the scores of the internal estimators.

Description

Julia Equivalent: `IAI.fit_predict!`

Usage

```r
## S3 method for class 'categorical_reward_estimator'
fit_predict(obj, X, treatments, ...)
```

Arguments

- `obj` The learner or grid to use for estimation
- `X` The features of the data.
- `treatments` The treatment applied to each point in the data.
- `...` Additional arguments depending on the treatment and outcome types. Refer to the Julia documentation for more information.

IAI Compatibility

Requires IAI version 2.0 or higher.

Examples

```r
## Not run: iai::fit_predict(obj, X, treatments, outcomes)
```

fit_predict.numeric_reward_estimator

Fit a numeric reward estimator on features, treatments and outcomes and return predicted counterfactual rewards for each observation, under each treatment candidate, as well as the scores of the internal estimators.

Description

Julia Equivalent: `IAI.fit_predict!`
Usage

```r
## S3 method for class 'numeric_reward_estimator'
fit_predict(obj, X, treatments, ...)
```

Arguments

- `obj` : The learner or grid to use for estimation
- `X` : The features of the data.
- `treatments` : The treatment applied to each point in the data.
- `...` : Additional arguments depending on the treatment and outcome types. Refer to the Julia documentation for more information.

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: iai::fit_predict(obj, X, treatments, outcomes)
```

---

**fit_transform**

*Fit an imputation model using the given features and impute the missing values in these features*

Description

Similar to calling `fit.imputation_learner` followed by `transform`.

Usage

```r
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

```r
## Not run:
X <- iris
X[, 1] <- NA
grid <- iai::grid_search(
    iai::imputation_learner(),
    method = c("opt_knn", "opt_tree"),
)
iai::fit_transform(grid, X)
## End(Not run)
```

---

**fit_transform_cv**

Train a grid using cross-validation with features and impute all missing values in these features

---

**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**

```r
## Not run:
X <- iris
X[, 1] <- NA
grid <- iai::grid_search(
    iai::imputation_learner(),
    method = c("opt_knn", "opt_tree"),
)
iai::fit_transform_cv(grid, X)
## End(Not run)
```
**get_best_params**

*Return the best parameter combination from a grid*

**Description**

Julia Equivalent: `IAI.get_best_params`

**Usage**

```julia
get_best_params(grid)
```

**Arguments**

- `grid`: The grid search to query.

**Examples**

```julia
## 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**

```julia
get_classification_label(lnr, node_index, ...)
```

**Arguments**

- `lnr`: The learner to query.
- `node_index`: The node in the tree to query.
- `...`: Refer to the Julia documentation for available parameters.

**Examples**

```julia
## 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.
- `...`  
Refer to the Julia documentation for available parameters.

**Examples**

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

---

**get_cluster_assignments**

*Return the indices of the trees assigned to each cluster, under the clustering of a given number of trees*

**Description**

Julia Equivalent: `IAI.get_cluster_assignments`

**Usage**

```
get_cluster_assignments(stability, num_trees)
```

**Arguments**

- `stability`  
The stability analysis to query.
- `num_trees`  
The number of trees to include in the clustering.

**IAI Compatibility**

Requires IAI version 2.2 or higher.
get_cluster_details

Examples

## Not run: iai::get_cluster_assignments(stability, num_trees)

---

get_cluster_details  *Return the centroid information for each cluster, under the clustering of a given number of trees*

Description

Julia Equivalent: `IAI.get_cluster_details`

Usage

get_cluster_details(stability, num_trees)

Arguments

- **stability**: The stability analysis to query
- **num_trees**: The number of trees to include in the clustering

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: iai::get_cluster_details(stability, num_trees)

---

get_cluster_distances  *Return the distances between the centroids of each pair of clusters, under the clustering of a given number of trees*

Description

Julia Equivalent: `IAI.get_cluster_distances`

Usage

get_cluster_distances(stability, num_trees)

Arguments

- **stability**: The stability analysis to query
- **num_trees**: The number of trees to include in the clustering
IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: iai::get_cluster_distances(stability, num_trees)
```

---

**get_depth**

Get the depth of a node of a tree

**Description**

Julia Equivalent: `IAI.get_depth`

**Usage**

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

**Arguments**

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

**Examples**

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

---

**get_estimation_densities**

Return the total kernel density surrounding each treatment candidate for the propensity/outcome estimation problems in a fitted learner.

**Description**

Julia Equivalent: `IAI.get_estimation_densities`

**Usage**

```r
get_estimation_densities(lnr, ...)
```

**Arguments**

- `lnr`: The learner from which to extract densities
- `...`: Refer to the Julia documentation for other parameters
get_features_used

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: iai::get_estimation_densities(lnr, ...)
```

---

get_features_used  
*Return the names of the features used by the learner*

Description

Julia Equivalent: `IAI.get_features_used`

Usage

```r
get_features_used(lnr)
```

Arguments

- `lnr`  
The learner to query.

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: iai::get_features_used(lnr)
```

---

get_grid_results  
*Return a summary of the results from the grid search*

Description

This function was deprecated and renamed to `get_grid_result_summary()` in iai 1.5.0. This is for consistency with the IAI v2.2.0 Julia release.

Usage

```r
get_grid_results(grid)
```

Arguments

- `grid`  
The grid search to query.
Examples

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

---

get_grid_result_details

*Return a vector of lists detailing the results of the grid search*

Description

Julia Equivalent: `IAI.get_grid_result_details`

Usage

```r
get_grid_result_details(grid)
```

Arguments

- `grid` The grid search to query.

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: iai::get_grid_result_details(grid)

---

get_grid_result_summary

*Return a summary of the results from the grid search*

Description

Julia Equivalent: `IAI.get_grid_result_summary`

Usage

```r
get_grid_result_summary(grid)
```

Arguments

- `grid` The grid search to query.

Examples

## Not run: iai::get_grid_result_summary(grid)
**get_learner**

Return the fitted learner using the best parameter combination from a grid

**Description**

Julia Equivalent: `IAI.get_learner`

**Usage**

```r
get_learner(grid)
```

**Arguments**

- `grid`: The grid to query.

**Examples**

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

---

**get_lower_child**

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

**Description**

Julia Equivalent: `IAI.get_lower_child`

**Usage**

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

**Arguments**

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

**Examples**

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

Return the machine ID for the current computer.

Description

This ID ties the IAI license file to your machine.

Usage

get_machine_id()

Examples

## Not run: iai::get_machine_id()

---

get_num_fits

Generic function for returning the number of fits in a trained learner

Description

Generic function for returning the number of fits in a trained learner

Usage

get_num_fits(obj, ...)

Arguments

- **obj** The object controlling which method is used
- **...** Arguments depending on the specific method used
Description

Julia Equivalent: **IAI.get_num_fits**

Usage

```r
## S3 method for class 'glmnetcv_learner'
get_num_fits(obj, ...)
```

Arguments

- `obj` The GLMNet learner to query.
- `...` Additional arguments (unused)

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: lnr <- iai::get_num_fits(lnr)
```

Description

Julia Equivalent: **IAI.get_num_fits**

Usage

```r
## S3 method for class 'optimal_feature_selection_learner'
get_num_fits(obj, ...)
```

Arguments

- `obj` The Optimal Feature Selection learner to query.
- `...` Additional arguments (unused)
IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```r
# Not run: iai::get_num_fits(lnr)
```

### get_num_nodes

Return the number of nodes in a trained learner

**Description**

Julia Equivalent: `IAI.get_num_nodes`

**Usage**

```r
get_num_nodes(lnr)
```

**Arguments**

- `lnr` The learner to query.

**Examples**

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

### get_num_samples

Get the number of training points contained in a node of a tree

**Description**

Julia Equivalent: `IAI.get_num_samples`

**Usage**

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

**Arguments**

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

**Examples**

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

Return the value of all parameters on a learner

Description
Julia Equivalent: `IAI.get_params`

Usage
`get_params(lnr)`

Arguments
`lnr` The learner to query.

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

get_parent

Get the index of the parent node at a node of a tree

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

Return the quality of the treatments at a node of a tree

Description

Julia Equivalent: `IAI.get_policy_treatment_outcome`

Usage

```julia
get_policy_treatment_outcome(lnr, node_index, ...)
```

Arguments

- `lnr` The learner to query.
- `node_index` The node in the tree to query.
- `...` Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```julia
# Not run: iai::get_policy_treatment_outcome(lnr, 1)
```

get_policy_treatment_rank

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

Description

Julia Equivalent: `IAI.get_policy_treatment_rank`

Usage

```julia
get_policy_treatment_rank(lnr, node_index, ...)
```

Arguments

- `lnr` The learner to query.
- `node_index` The node in the tree to query.
- `...` Refer to the Julia documentation for available parameters.
**IAI Compatibility**

Requires IAI version 2.0 or higher.

**Examples**

```r
## Not run: iai::get_policy_treatment_rank(lnr, 1)
```

---

**get_prediction_constant**

*Generic function for returning the prediction constant in a trained learner*

**Description**

Generic function for returning the prediction constant in a trained learner.

**Usage**

```r
get_prediction_constant(obj, ...)
```

**Arguments**

- **obj**
  The object controlling which method is used
- **...**
  Arguments depending on the specific method used

---

**get_prediction_constant.glmnetcv_learner**

*Return the constant term in the prediction in a trained GLMNet learner*

**Description**

Julia Equivalent: `IAI.get_prediction_constant`

**Usage**

```r
## S3 method for class 'glmnetcv_learner'
get_prediction_constant(obj, fit_index = NULL, ...)
```

**Arguments**

- **obj**
  The learner to query.
- **fit_index**
  The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- **...**
  Additional arguments (unused)
IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

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

---

**get_prediction_constant.optimal_feature_selection_learner**

*Return the constant term in the prediction in a trained Optimal Feature Selection learner*

Description

Julia Equivalent: `IAI.get_prediction_constant`

Usage

```r
## S3 method for class 'optimal_feature_selection_learner'
get_prediction_constant(obj, fit_index = NULL, ...)
```

Arguments

- `obj` The learner to query.
- `fit_index` The index of the cluster to use for prediction, if the `coordinated_sparsity` parameter on the learner is TRUE.
- `...` Additional arguments (unused)

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

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

Generic function for returning the prediction weights in a trained learner

Description

Generic function for returning the prediction weights in a trained learner

Usage

get_prediction_weights(obj, ...)

Arguments

obj The object controlling which method is used
...

get_prediction_weights.glmnetcv_learner

Return the weights for numeric and categoric features used for prediction in a trained GLMNet learner

Description

Julia Equivalent: IAI.get_prediction_weights

Usage

## S3 method for class glmnetcv_learner
get_prediction_weights(obj, fit_index = NULL, ...)

Arguments

obj The learner to query.
fit_index The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
...

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

## Not run: iai::get_prediction_weights(lnr)
get_prediction_weights.optimal_feature_selection_learner

Return the weights for numeric and categoric features used for prediction in a trained Optimal Feature Selection learner

Description

Julia Equivalent: `IAI.get_prediction_weights`

Usage

```r
## S3 method for class 'optimal_feature_selection_learner'
get_prediction_weights(obj, fit_index = NULL, ...)
```

Arguments

- `obj` The learner to query.
- `fit_index` The index of the cluster to use for prediction, if the coordinated_sparsity parameter on the learner is TRUE.
- `...` Additional arguments (unused)

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```r
## 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

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

Arguments

lnr    The learner to query.
node_index The node in the tree to query.
...
Refer to the Julia documentation for available parameters.

Examples

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

get_regression_constant

Generic function for returning the constant term in the regression prediction at a node of a tree

Description

Generic function for returning the constant term in the regression prediction at a node of a tree

Usage

get_regression_constant(obj, ...)

Arguments

obj    The object controlling which method is used
...
Arguments depending on the specific method used

get_regression_constant.classification_tree_learner

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

Description

Julia Equivalent: IAI.get_regression_constant

Usage

## S3 method for class 'classification_tree_learner'
get_regression_constant(obj, node_index, ...)
get_regression_constant.prescription_tree_learner

Arguments

- **obj**: The learner to query.
- **node_index**: The node in the tree to query.
- ... Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```r
## Not run: iai::get_regression_constant(lnr, 1)
```

---

## S3 method for class 'prescription_tree_learner'
get_regression_constant(obj, node_index, treatment, ...)

Description

Julia Equivalent: `IAI.get_regression_constant`

Usage

```r
## S3 method for class 'prescription_tree_learner'
get_regression_constant(obj, node_index, treatment, ...)
```

Arguments

- **obj**: The learner to query.
- **node_index**: The node in the tree to query.
- **treatment**: The treatment to query.
- ... Refer to the Julia documentation for available parameters.

Examples

```r
## Not run: iai::get_regression_constant(lnr, 1, "A")
```
get_regression_constant.regression_tree_learner

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

Description

Julia Equivalent: IAI.get_regression_constant

Usage

## S3 method for class "regression_tree_learner"
get_regression_constant(obj, node_index, ...)

Arguments

obj The learner to query.
node_index The node in the tree to query.
... Refer to the Julia documentation for available parameters.

Examples

## Not run: iai::get_regression_constant(lnr, 1)

get_regression_constant.survival_tree_learner

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

Description

Julia Equivalent: IAI.get_regression_constant

Usage

## S3 method for class "survival_tree_learner"
get_regression_constant(obj, node_index, ...)

Arguments

obj The learner to query.
node_index The node in the tree to query.
... Refer to the Julia documentation for available parameters.
get_regression_weights(classification_tree_learner)

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```r
## Not run: iai::get_regression_constant(lnr, 1)
```

---

**Description**

Generic function for returning the weights for each feature in the regression prediction at a node of a tree

**Usage**

```r
get_regression_weights(obj, ...)
```

**Arguments**

- `obj` The object controlling which method is used
- `...` Arguments depending on the specific method used

---

**Description**

Julia Equivalent: \texttt{IAI.get\_regression\_weights}

**Usage**

```r
## S3 method for class 'classification\_tree\_learner'
get_regression_weights(obj, node\_index, ...)
```

**Arguments**

- `obj` The learner to query.
- `node_index` The node in the tree to query.
- `...` Refer to the Julia documentation for available parameters.
get_regression_weights.prescription_tree_learner

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

## Not run: iai::get_regression_weights(lnr, 1)

---

get_regression_weights.prescription_tree_learner

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

Description

Julia Equivalent: \texttt{IAI.get_regression_weights}

Usage

## S3 method for class 'prescription_tree_learner'
get_regression_weights(obj, node_index, treatment, ...)

Arguments

- \texttt{obj} The learner to query.
- \texttt{node_index} The node in the tree to query.
- \texttt{treatment} The treatment to query.
- ... Refer to the Julia documentation for available parameters.

Examples

## Not run: iai::get_regression_weights(lnr, 1, "A")
get_regression_weights.regression_tree_learner

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

Description

Julia Equivalent: `IAI.get_regression_weights`

Usage

```r
## S3 method for class 'regression_tree_learner'
get_regression_weights(obj, node_index, ...)
```

Arguments

- `obj`: The learner to query.
- `node_index`: The node in the tree to query.
- `...`: Refer to the Julia documentation for available parameters.

Examples

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

get_regression_weights.survival_tree_learner

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

Description

Julia Equivalent: `IAI.get_regression_weights`

Usage

```r
## S3 method for class 'survival_tree_learner'
get_regression_weights(obj, node_index, ...)
```

Arguments

- `obj`: The learner to query.
- `node_index`: The node in the tree to query.
- `...`: Refer to the Julia documentation for available parameters.
get_rich_output_params

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```markdown
## Not run: iai::get_regression_weights(lnr, 1)
```

---

get_rich_output_params

*Return the current global rich output parameter settings*

Description

Julia Equivalent: `IAI.get_rich_output_params`

Usage

```julia
get_rich_output_params()
```

Examples

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

---

get_roc_curve_data

*Extract the underlying data from an ROC curve*

Description

ROC curves are returned by `roc_curve`, e.g. `roc_curve.classification_learner`

Usage

```julia
get_roc_curve_data(curve)
```

Arguments

- `curve` The curve to query.

Details

The data is returned as a list with two keys: `auc` giving the area-under-the-curve, and `coords` containing a vector of lists representing each point on the curve, each with keys `fpr` (the false positive rate), `tpr` (the true positive rate) and `threshold` (the threshold).

Julia Equivalent: `IAI.get_roc_curve_data`
IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

## Not run: iai::get_roc_curve_data(curve)

---

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

- `lnr`: The learner to query.
- `node_index`: 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.
get_split_threshold

Description
Julia Equivalent: `IAI.get_split_threshold`

Usage
```r
get_split_threshold(lnr, node_index)
```

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

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

get_split_weights

Description
Julia Equivalent: `IAI.get_split_weights`

Usage
```r
get_split_weights(lnr, node_index)
```

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

Examples
```r
## Not run: iai::get_split_threshold(lnr, 1)
```
get_stability_results  Return the trained trees in order of increasing objective value, along with their variable importance scores for each feature

Description
Julia Equivalent: `IAI.get_stability_results`

Usage
``` Julia
get_stability_results(stability)
```

Arguments
- `stability` The stability analysis to query

IAI Compatibility
Requires IAI version 2.2 or higher.

Examples
``` Julia
## Not run: iai::get_stability_results(stability)
```

get_survival_curve  Return the survival curve at a node of a tree

Description
Julia Equivalent: `IAI.get_survival_curve`

Usage
``` Julia
get_survival_curve(lnr, node_index, ...)
```

Arguments
- `lnr` The learner to query.
- `node_index` The node in the tree to query.
- `...` Refer to the Julia documentation for available parameters.

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

Extract the underlying data from a survival curve (as returned by predict.survival_learner or 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_survival_expected_time

Return the predicted expected survival time at a node of a tree

Description

Julia Equivalent: IAI.get_survival_expected_time

Usage

get_survival_expected_time(lnr, node_index, ...)

Arguments

lnr The learner to query.
node_index The node in the tree to query.
... Refer to the Julia documentation for available parameters.
get_survival_hazard

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: iai::get_survival_expected_time(lnr, 1)
```

---

get_survival_hazard  
*Return the predicted hazard ratio at a node of a tree*

Description

Julia Equivalent: `IAI.get_survival_hazard`

Usage

```r
get_survival_hazard(lnr, node_index, ...)
```

Arguments

- `lnr`  
The learner to query.
- `node_index`  
The node in the tree to query.
- `...`  
Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: iai::get_survival_hazard(lnr, 1)
```
get_train_errors

**Description**

Extract the training objective value for each candidate tree in the comparison, where a lower value indicates a better solution.

**Julia Equivalent:** `IAI.get_train_errors`

**Usage**

```julia
get_train_errors(similarity)
```

**Arguments**

- `similarity`: The similarity comparison

**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```julia
## Not run: iai::get_train_errors(similarity)
```

---

get_tree

**Description**

Return a copy of the learner that uses a specific tree rather than the tree with the best training objective.

**Julia Equivalent:** `IAI.get_tree`

**Usage**

```julia
get_tree(lnr, index)
```

**Arguments**

- `lnr`: The original learner
- `index`: The index of the tree to use

**IAI Compatibility**

Requires IAI version 2.2 or higher.
glmnetcv_classifier

Learner for training GLMNet models for classification problems with cross-validation

Description

Julia Equivalent: `IAI.GLMNetCVClassifier`

Usage

```r
glmnetcv_classifier(...)```

Arguments

```r
...
```

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

IAI Compatibility

Requires IAI version 3.0 or higher.
**Examples**

```r
## Not run: lnr <- iai::glmnetcv_regressor()
```

---

### Description

Julia Equivalent: `IAI.GLMNetCVRegressor`

### Usage

```r
glmnetcv_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 2.1 or higher.

### Examples

```r
## Not run: lnr <- iai::glmnetcv_regressor()
```

---

### Description

Julia Equivalent: `IAI.GLMNetCVSurvivalLearner`

### Usage

```r
glmnetcv_survival_learner(...)```

### Arguments

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

Requires IAI version 3.0 or higher.

Examples

```r
## Not run: lnr <- iai::glmnetcv_survival_learner()
```

---

**grid_search**

Controls grid search over parameter combinations

Description

Julia Equivalent: `IAI.GridSearch`

Usage

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

Arguments

- `lnr`: The learner to use when validating.
- `...`: The parameters to validate over.

Examples

```r
## Not run:
grid <- iai::grid_search(
  iai::optimal_tree_classifier(
    random_seed = 1,
  ),
  max_depth = 1:5,
)

## End(Not run)
```
**iai_setup**

Initialize Julia and the IAI package.

**Description**

This function is called automatically with default parameters the first time any ‘iai’ function is used in an R session. If custom parameters for Julia setup are required, this function must be called in every R session before calling other ‘iai’ functions.

**Usage**

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

**Arguments**

*...*  
All parameters are passed through to `JuliaCall::julia_setup`

**Examples**

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

---

**imputation_learner**

Generic learner for imputing missing values

**Description**

Julia Equivalent: `IAI.ImputationLearner`

**Usage**

```r
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**

```r
## 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

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

#### Arguments

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

#### Details

This function was deprecated in iai 1.7.0. This is for consistency with the IAI v3.0.0 Julia release.

#### Examples

```r
## Not run:
X <- iris
X[1, 1] <- NA
iai::impute(X)
## End(Not run)
```

### impute_cv

**Impute missing values using cross validation**

#### Description

Julia Equivalent: IAI.impute_cv

#### Usage

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

#### Arguments

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

Details

This function was deprecated in iai 1.7.0. This is for consistency with the IAI v3.0.0 Julia release.

Examples

```r
## Not run:
X <- iris
X[1, 1] <- NA
iai::impute_cv(X, list(method = c("opt_knn", "opt_tree")))

## End(Not run)
```

install_julia

Download and install Julia automatically.

Description

Download and install Julia automatically.

Usage

```r
install_julia(version = "latest", prefix = julia_default_install_dir())
```

Arguments

- **version**: The version of Julia to install (e.g. "1.6.3"). Defaults to "latest", which will install the most recent stable release.
- **prefix**: The directory where Julia will be installed. Defaults to a location determined by `rappdirs::user_data_dir`.

Examples

```r
## Not run: iai::install_julia()
```

install_system_image

Download and install the IAI system image automatically.

Description

Download and install the IAI system image automatically.
is_categoric_split

Usage

install_system_image(
    version = "latest",
    replace_default = FALSE,
    prefix = sysimage_default_install_dir(),
    accept_license = FALSE
)

Arguments

version The version of the IAI system image to install (e.g. "2.1.0"). Defaults to "latest", which will install the most recent release.
replace_default Whether to replace the default Julia system image with the downloaded IAI system image. Defaults to FALSE.
prefix The directory where the IAI system image will be installed. Defaults to a location determined by `rappdirs::user_data_dir`.
accept_license Set to TRUE to confirm that you agree to the End User License Agreement and skip the interactive confirmation dialog.

Examples

## Not run: iai::install_system_image()

---

is_categoric_split Check if a node of a tree applies a categoric split

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
```julia
# 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
```julia
# 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

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

Arguments

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

Examples

```r
## 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

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

Arguments

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

Examples

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

*Loads the Julia Graphviz library to permit certain visualizations.*

**Description**

The library will be installed if not already present.

**Usage**

```julia
load_graphviz()
```

**Examples**

```r
## Not run: iai::load_graphviz()
```

mean_imputation_learner

*Learner for conducting mean imputation*

**Description**

Julia Equivalent: `IAI.MeanImputationLearner`

**Usage**

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

**Arguments**

`...`

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

**Examples**

```r
## 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**

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

---

**multi_questionnaire**

Generic function for constructing an interactive questionnaire with multiple learners

**Description**

Generic function for constructing an interactive questionnaire with multiple learners

**Usage**

`multi_questionnaire(obj, ...)`

**Arguments**

- `obj` The object controlling which method is used
- `...` Arguments depending on the specific method used
multi_questionnaire.default

Construct an interactive questionnaire from multiple specified learners

Description

Refer to the documentation on advanced tree visualization for more information.

Usage

```r
## Default S3 method:
multi_questionnaire(obj, ...)
```

Arguments

- `obj` The questions to visualize. Refer to the Julia documentation on multi-learner visualizations for more information.
- `...` Additional arguments (unused)

Details

Julia Equivalent: `IAI.MultiQuestionnaire`

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

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

## End(Not run)
```
multi_questionnaire.grid_search

Construct an interactive tree questionnaire using multiple learners from the results of a grid search

Description

Julia Equivalent: IAI.MultiQuestionnaire

Usage

## S3 method for class 'grid_search'
multi_questionnaire(obj, ...)

Arguments

obj The grid to visualize
...

Additional arguments (unused)

IAI Compatibility

Requires IAI version 2.0 or higher.

Examples

## Not run: iai::multi_questionnaire(grid)

multi_tree_plot

Generic function for constructing an interactive tree visualization of multiple tree learners

Description

Generic function for constructing an interactive tree visualization of multiple tree learners

Usage

multi_tree_plot(obj, ...)

Arguments

obj The object controlling which method is used
...

Arguments depending on the specific method used
multi_tree_plot.default

Construct an interactive tree visualization of multiple tree learners as specified by questions.

Description

Refer to the documentation on advanced tree visualization for more information.

Usage

```r
## Default S3 method:
multi_tree_plot(obj, ...)
```

Arguments

- `obj`: The questions to visualize. Refer to the Julia documentation on multi-learner visualizations for more information.
- `...`: Additional arguments (unused)

Details

Julia Equivalent: `IAI.MultiTreePlot`

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

```r
## Not run:
iai::multi_tree_plot(list("Visualizing" = list(
  "first learner" = lrn1,
  "second learner" = lrn2
)));
## End(Not run)
```
### multi_tree_plot.grid_search

*Construct an interactive tree visualization of multiple tree learners from the results of a grid search*

---

**Description**

Julia Equivalent: `IAI.MultiTreePlot`

**Usage**

```r
## S3 method for class 'grid_search'
multi_tree_plot(obj, ...)
```

**Arguments**

- `obj` The grid to visualize
- `...` Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.0 or higher.

**Examples**

```r
## Not run: iai::multi_tree_plot(grid)
```

---

### numeric_classification_reward_estimator

*Learner for conducting reward estimation with numeric treatments and classification outcomes*

---

**Description**

Julia Equivalent: `IAI.NumericClassificationRewardEstimator`

**Usage**

```r
numeric_classification_reward_estimator(...)```

**Arguments**

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

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: lnr <- iai::numeric_classification_reward_estimator()
```

---

numeric_regression_reward_estimator

Learner for conducting reward estimation with numeric treatments and regression outcomes

Description

Julia Equivalent: `IAI.NumericRegressionRewardEstimator`

Usage

`numeric_regression_reward_estimator(...)`

Arguments

`...`

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

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: lnr <- iai::numeric_regression_reward_estimator()
```
**numeric_reward_estimator**

*Learner for conducting reward estimation with numeric treatments*

**Description**

This function was deprecated in iai 1.6.0, and [numeric_classification_reward_estimator()](#) or [numeric_classification_reward_estimator()](#) should be used instead.

**Usage**

```r
classification_reward_estimator(…)
```

**Arguments**

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

**Details**

This deprecation is no longer supported as of the IAI v3 release.

**IAI Compatibility**

Requires IAI version 2.1 or 2.2.

**Examples**

```r
## Not run: lnr <- iai::numeric_reward_estimator()
```

---

**numeric_survival_reward_estimator**

*Learner for conducting reward estimation with numeric treatments and survival outcomes*

**Description**

Julia Equivalent: `IAI.NumericSurvivalRewardEstimator`

**Usage**

```r
classification_reward_estimator(…)
```

**Details**

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

Learner for conducting Optimal Feature Selection on classification problems

**Description**

Julia Equivalent: `IAI.OptimalFeatureSelectionClassifier`

**Usage**

```r
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**

```r
## 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

```r
## 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

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

Learner for training Optimal Policy Trees where the policy should aim to maximize outcomes

Description

Julia Equivalent: IAI.OptimalTreePolicyMaximizer

Usage

optimal_tree_policy_maximizer(...)

Arguments

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

IAI Compatibility

Requires IAI version 2.0 or higher.

Examples

## Not run: lnr <- iai::optimal_tree_policy_maximizer()

optimal_tree_policy_minimizer

Learner for training Optimal Policy Trees where the policy should aim to minimize outcomes

Description

Julia Equivalent: IAI.OptimalTreePolicyMinimizer

Usage

optimal_tree_policy_minimizer(...)

Arguments

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

IAI Compatibility

Requires IAI version 2.0 or higher.

Examples

```r
## Not run: lnr <- iai::optimal_tree_policy_minimizer()
```

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

```r
## 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

...
optimal_tree_survival_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::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_survival_learner

Learner for training Optimal Survival Trees

Description

Julia Equivalent: IAI.OptimalTreeSurvivalLearner

Usage

optimal_tree_survival_learner(...)

Arguments

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

Examples

## Not run: lnr <- iai::optimal_tree_survival_learner()

---

optimal_tree_survivor  
Learner for training Optimal Survival Trees

Description

This function was deprecated and renamed to `optimal_tree_survival_learner()` in iai 1.3.0. This is for consistency with the IAI v2.0.0 Julia release.

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()
```markdown
## opt_svm_imputation_learner

**Learner for conducting optimal SVM imputation**

### Description

Julia Equivalent: `IAI.OptSVMImputationLearner`

### Usage

```r
opt_svm_imputation_learner(...)```

### Arguments

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

### Examples

```r
## 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

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

### Arguments

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

### Examples

```r
## Not run: lnr <- iai::opt_tree_imputation_learner()
```
plot.grid_search

Plot a grid search results for Optimal Feature Selection learners

Description

Plot a grid search results for Optimal Feature Selection learners

Usage

```r
## S3 method for class 'grid_search'
plot(x, ...)
```

Arguments

- `x`: The grid search to plot
- `...`: Additional arguments (passed to `autoplot.grid_search`)

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: plot(grid)
```

plot.roc_curve

Plot an ROC curve

Description

Plot an ROC curve

Usage

```r
## S3 method for class 'roc_curve'
plot(x, ...)
```

Arguments

- `x`: The ROC curve to plot
- `...`: Additional arguments (unused)

IAI Compatibility

Requires IAI version 2.1 or higher.
Examples

## Not run: plot(roc)

plot.similarity_comparison

Plot a similarity comparison

Description

Plot a similarity comparison

Usage

## S3 method for class 'similarit..." similar comparison'
plot(x, ...)

Arguments

x The similarity comparison to plot
... Additional arguments (unused)

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: plot(similarity)

plot.stability_analysis

Plot a stability analysis

Description

Plot a stability analysis

Usage

## S3 method for class 'stability_..." stability analysis'
plot(x, ...)


**Arguments**

- **x**
  - The stability analysis to plot
- **...**
  - Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: plot(stability)
```

---

**predict**

*Generic function for returning the predictions of a model*

**Description**

Generic function for returning the predictions of a model

**Usage**

```r
predict(obj, ...)
```

**Arguments**

- **obj**
  - The object controlling which method is used
- **...**
  - Arguments depending on the specific method used

---

**predict.categorical_reward_estimator**

*Return counterfactual rewards estimated by a categorical reward estimator for each observation in the supplied data*

**Description**

Julia Equivalent: `IAI.predict`

**Usage**

```r
## S3 method for class 'categorical_reward_estimator'
predict(obj, X, ...)
```
predict.glmnetcv_learner

Arguments

- **obj** The learner or grid to use for estimation
- **X** The features of the data.
- **...** Additional arguments depending on the treatment and outcome types. Refer to the Julia documentation for more information.

IAI Compatibility

Requires IAI version 2.0 or higher.

Examples

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

---

`predict.glmnetcv_learner`

*Return the predictions made by a GLMNet learner for each point in the features*

Description

Julia Equivalent: `IAI.predict`

Usage

```r
## S3 method for class 'glmnetcv_learner'
predict(obj, X, fit_index = NULL, ...)
```

Arguments

- **obj** The learner or grid to use for prediction.
- **X** The features of the data.
- **fit_index** The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- **...** Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

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

Return counterfactual rewards estimated by a numeric reward estimator for each observation in the supplied data

Description

Julia Equivalent: IAI.predict

Usage

## S3 method for class 'numeric_reward_estimator'
predict(obj, X, ...)

Arguments

obj The learner or grid to use for estimation
X The features of the data.
... Additional arguments depending on the treatment and outcome types. Refer to the Julia documentation for more information.

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

## Not run: iai::predict(lnr, X, treatments, outcomes)

predict.optimal_feature_selection_learner

Return the predictions made by an Optimal Feature Selection learner for each point in the features

Description

Julia Equivalent: IAI.predict

Usage

## S3 method for class 'optimal_feature_selection_learner'
predict(obj, X, fit_index = NULL, ...)


predict.supervised_learner

Arguments

- **obj**: The learner or grid to use for prediction.
- **X**: The features of the data.
- **fit_index**: The index of the cluster to use for prediction, if the `coordinated_sparsity` parameter on the learner is `TRUE`.
- **...**: Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

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

---

**predict.supervised_learner**

*Return the predictions made by a supervised learner for each point in the features*

Description

Julia Equivalent: `IAI.predict`

Usage

```r
## S3 method for class 'supervised_learner'
predict(obj, X, ...)
```

Arguments

- **obj**: The learner or grid to use for prediction.
- **X**: The features of the data.
- **...**: Refer to the Julia documentation for available parameters.

Examples

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

Return the predictions made by a survival learner for each point in the features

Description

Julia Equivalent: IAI.predict

Usage

## S3 method for class 'survival_learner'
predict(obj, X, t = NULL, ...)

Arguments

- obj: The learner or grid to use for prediction.
- X: The features of the data.
- t: The time for which to predict survival probability, defaulting to returning the entire survival curve if not supplied
- ...: Additional arguments (unused)

Examples

## Not run: iai::predict(lnr, X, t = 10)

predict_expected_survival_time

Generic function for returning the expected survival time predicted by a model

Description

Generic function for returning the expected survival time predicted by a model

Usage

predict_expected_survival_time(obj, ...)

Arguments

- obj: The object controlling which method is used
- ...: Arguments depending on the specific method used
predict_expected_survival_time.glmnetcv_survival_learner

Return the expected survival time estimate made by a glmnetcv_survival_learner for each point in the features.

Description

Julia Equivalent: IAI.predict_expected_survival_time

Usage

## S3 method for class 'glmnetcv_survival_learner'
predict_expected_survival_time(obj, X, fit_index = NULL, ...)

Arguments

- obj: The learner or grid to use for prediction.
- X: The features of the data.
- fit_index: The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- ...: Additional arguments (unused)

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

## Not run: iai::predict_expected_survival_time(lnr, X)

predict_expected_survival_time.survival_curve

Return the expected survival time estimate made by a survival curve (as returned by predict.survival_learner or get_survival_curve)

Description

Julia Equivalent: IAI.predict_expected_survival_time

Usage

## S3 method for class 'survival_curve'
predict_expected_survival_time(obj, ...)

---

predict_expected_survival_time.glmnetcv_survival_learner

Return the expected survival time estimate made by a glmnetcv_survival_learner for each point in the features.

Description

Julia Equivalent: IAI.predict_expected_survival_time

Usage

## S3 method for class 'glmnetcv_survival_learner'
predict_expected_survival_time(obj, X, fit_index = NULL, ...)

Arguments

- obj: The learner or grid to use for prediction.
- X: The features of the data.
- fit_index: The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- ...: Additional arguments (unused)

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

## Not run: iai::predict_expected_survival_time(lnr, X)

predict_expected_survival_time.survival_curve

Return the expected survival time estimate made by a survival curve (as returned by predict.survival_learner or get_survival_curve)

Description

Julia Equivalent: IAI.predict_expected_survival_time

Usage

## S3 method for class 'survival_curve'
predict_expected_survival_time(obj, ...)

---
**predict_expected_survival_time.survival_learner**

**Arguments**

- **obj** The survival curve to use for prediction.
- ... Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: iai::predict_expected_survival_time(curve)
```

---

**Description**

Julia Equivalent: `IAI.predict_expected_survival_time`

**Usage**

```r
## S3 method for class 'survival_learner'
predict_expected_survival_time(obj, X, ...)
```

**Arguments**

- **obj** The learner or grid to use for prediction.
- **X** The features of the data.
- ... Additional arguments (unused)

**IAI Compatibility**

Requires IAI version 2.0 or higher.

**Examples**

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

Generic function for returning the hazard coefficient predicted by a model.

Usage

predict_hazard(obj, ...)

Arguments

- **obj**: The object controlling which method is used.
- **...**: Arguments depending on the specific method used.

predict_hazard(glmnetcv_survival_learner)

Return the fitted hazard coefficient estimate made by a glmnetcv_survival_learner for each point in the features.

Description

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

Usage

## S3 method for class 'glmnetcv_survival_learner'
predict_hazard(obj, X, fit_index = NULL, ...)

Arguments

- **obj**: The learner or grid to use for prediction.
- **X**: The features of the data.
- **fit_index**: The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- **...**: Additional arguments (unused)

Details

Julia Equivalent: IAI.predict_hazard
IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

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

predict_hazard.survival_learner

Return the fitted hazard coefficient estimate made by a survival learner for each point in the features.

Description

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

Usage

```r
## S3 method for class 'survival_learner'
predict_hazard(obj, X, ...)
```

Arguments

- `obj`: The learner or grid to use for prediction.
- `X`: The features of the data.
- `...`: Additional arguments (unused)

Details

Julia Equivalent: IAI.predict_hazard

IAI Compatibility

Requires IAI version 1.2 or higher.

Examples

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

Generic function for returning the outcomes predicted by a model under each treatment

Description

Generic function for returning the outcomes predicted by a model under each treatment

Usage

predict_outcomes(obj, ...)

Arguments

obj The object controlling which method is used
... Arguments depending on the specific method used

predict_outcomes.policy_learner

Return the predicted outcome for each treatment made by a policy learner for each point in the features

Description

Julia Equivalent: IAI.predict_outcomes

Usage

## S3 method for class 'policy_learner'
predict_outcomes(obj, X, rewards, ...)

Arguments

obj The learner or grid to use for prediction.
X The features of the data.
rewards The estimated reward matrix for the data.
... Additional arguments (unused)

IAI Compatibility

Requires IAI version 2.0 or higher

Examples

## Not run: iai::predict_outcomes(lnr, X, rewards)
**predict_outcomes**

Return the predicted outcome for each treatment made by a prescription learner for each point in the features

---

**Description**

Julia Equivalent: `IAI.predict_outcomes`

**Usage**

```r
## S3 method for class 'prescription_learner'
predict_outcomes(obj, X, ...)
```

**Arguments**

- `obj` The learner or grid to use for prediction.
- `X` The features of the data.
- `...` Additional arguments (unused)

**Examples**

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

---

**predict_proba**

Generic function for returning the probabilities of class membership predicted by a model

---

**Description**

Generic function for returning the probabilities of class membership predicted by a model

**Usage**

```
predict_proba(obj, ...)
```

**Arguments**

- `obj` The object controlling which method is used
- `...` Arguments depending on the specific method used
predict_proba.classification_learner

Return the probabilities of class membership predicted by a classification learner for each point in the features

Description
Julia Equivalent: IAI.predict_proba

Usage
## S3 method for class 'classification_learner'
predict_proba(obj, X, ...)

Arguments

- **obj** The learner or grid to use for prediction.
- **X** The features of the data.
- **...** Additional arguments (unused)

Examples

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

predict_proba.glmnetcv_classifier

Return the probabilities of class membership predicted by a glmnetcv_classifier learner for each point in the features

Description
Julia Equivalent: IAI.predict_proba

Usage
## S3 method for class 'glmnetcv_classifier'
predict_proba(obj, X, fit_index = NULL, ...)

Arguments

- **obj** The learner or grid to use for prediction.
- **X** The features of the data.
- **fit_index** The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- **...** Additional arguments (unused)
predict_reward

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

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

```
predict_reward
Generic function for returning the counterfactual rewards estimated by a model under each treatment
```

Description

Generic function for returning the counterfactual rewards estimated by a model under each treatment

Usage

```r
predict_reward(obj, ...)
```

Arguments

- `obj` The object controlling which method is used
- `...` Arguments depending on the specific method used

```
predict_reward.categorical_reward_estimator
Return counterfactual rewards estimated by a categorical reward estimator for each observation in the supplied data and predictions
```

Description

Julia Equivalent: `IAI.predict_reward`

Usage

```r
## S3 method for class 'categorical_reward_estimator'
predict_reward(obj, X, ...)
```

Arguments

- `obj` The learner or grid to use for estimation
- `X` The features of the data.
- `...` Additional arguments depending on the treatment and outcome types. Refer to the Julia documentation for more information.
IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```r
## Not run: iai::predict_reward(lnr, X, treatments, outcomes, predictions)
```

### predict_reward.numeric_reward_estimator

*Return counterfactual rewards estimated by a numeric reward estimator for each observation in the supplied data and predictions*

#### Description

Julia Equivalent: `IAI.predict_reward`

#### Usage

```r
## S3 method for class 'numeric_reward_estimator'
predict_reward(obj, X, ...)
```

#### Arguments

- **obj**: The learner or grid to use for estimation
- **X**: The features of the data.
- **...**: Additional arguments depending on the treatment and outcome types. Refer to the Julia documentation for more information.

#### IAI Compatibility

Requires IAI version 3.0 or higher.

#### Examples

```r
## Not run: iai::predict_reward(lnr, X, treatments, outcomes, predictions)
```
predict_shap

Calculate SHAP values for all points in the features using the learner

Description

Julia Equivalent: `IAI.predict_shap`

Usage

```
predict_shap(lnr, X)
```

Arguments

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

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

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

predict_treatment_outcome

Return the estimated quality of each treatment in the trained model of the learner for each point in the features

Description

Julia Equivalent: `IAI.predict_treatment_outcome`

Usage

```
predict_treatment_outcome(lnr, X)
```

Arguments

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

IAI Compatibility

Requires IAI version 2.1 or higher.
Examples

```r
## Not run: iai::predict_treatment_outcome(lnr, X)
```

---

`predict_treatment_rank`

*Return the treatments in ranked order of effectiveness for each point in the features*

Description

Julia Equivalent: `IAI.predict_treatment_rank`

Usage

```r
predict_treatment_rank(lnr, X)
```

Arguments

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

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: iai::predict_treatment_rank(lnr, X)
```

---

`print_path`

*Print the decision path through the learner for each sample in the features*

Description

Julia Equivalent: `IAI.print_path`

Usage

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

Arguments

1nr  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)

---

prune_trees  Use the trained trees in a learner along with the supplied validation
data to determine the best value for the ‘cp’ parameter and then prune the
trees according to this value

Description

Julia Equivalent: IAI.prune_trees!

Usage

prune_trees(lnr, ...)

Arguments

1nr  The learner to prune
...  Refer to the Julia documentation for available parameters

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

## Not run: iai::prune_trees(lnr, ...)
questionnaire

**Generic function for constructing an interactive questionnaire**

**Description**

Julia Equivalent: IAI.Questionnaire

**Usage**

```julia
questionnaire(obj, ...)
```

**Arguments**

- `obj`: The object controlling which method is used
- `...`: Arguments depending on the specific method used

---

**questionnaire.optimal_feature_selection_learner**

*Specify an interactive questionnaire of an Optimal Feature Selection learner*

**Description**

Julia Equivalent: IAI.Questionnaire

**Usage**

```julia
## S3 method for class 'optimal_feature_selection_learner'
questionnaire(obj, ...)
```

**Arguments**

- `obj`: The learner to visualize.
- `...`: Refer to the [Julia documentation](https://juliastats.github.io/IAI/) for available parameters.

**IAI Compatibility**

Requires IAI version 2.1 or higher.

**Examples**

```julia
## Not run: iai::questionnaire(lnr)
```
**questionnaire.tree_learner**

*Specify an interactive questionnaire of a tree learner*

---

**Description**

Julia Equivalent: `IAI.Questionnaire`

**Usage**

```julia
# S3 method for class 'tree_learner'
questionnaire(obj, ...)
```

**Arguments**

- `obj` The learner to visualize.
- `...` Refer to the [Julia documentation](https://juliastats.github.io/IAI.jl/) for available parameters.

**IAI Compatibility**

Requires IAI version 1.1 or higher.

**Examples**

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

---

**random_forest_classifier**

*Learner for training random forests for classification problems*

---

**Description**

Julia Equivalent: `IAI.RandomForestClassifier`

**Usage**

```julia
random_forest_classifier(...)
```

**Arguments**

- `...` Use keyword arguments to set parameters on the resulting learner. Refer to the [Julia documentation](https://juliastats.github.io/IAI.jl/) for available parameters.

**IAI Compatibility**

Requires IAI version 2.1 or higher.
random_forest_survival_learner

Examples

```r
## Not run: lnr <- iai::random_forest_classifier()
```

random_forest_regressor

*Learner for training random forests for regression problems*

Description

Julia Equivalent: `IAI.RandomForestRegressor`

Usage

```
random_forest_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 2.1 or higher.

Examples

```r
## Not run: lnr <- iai::random_forest_regressor()
```

random_forest_survival_learner

*Learner for training random forests for survival problems*

Description

Julia Equivalent: `IAI.RandomForestSurvivalLearner`

Usage

```
random_forest_survival_learner(...)```

Arguments

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

**IAI Compatibility**
Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: lnr <- iai::random_forest_survival_learner()
```

**Description**
Julia Equivalent: `IAI.RandImputationLearner`

**Usage**

```r
call rand_imputation_learner(...)```

**Arguments**

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

**Examples**

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

---

**read_json**

Read in a learner or grid saved in JSON format

**Description**
Julia Equivalent: `IAI.read_json`

**Usage**

```r
call read_json(filename)
```

**Arguments**

`filename` The location of the JSON file.

**Examples**

```r
## Not run: obj <- iai::read_json("out.json")
```
refit_leaves  Refit the models in the leaves of a trained learner using the supplied data

Description
Julia Equivalent: IAI.refit_leaves!

Usage
refit_leaves(lnr, ...)

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

IAI Compatibility
Requires IAI version 3.0 or higher.

Examples
## Not run: iai::refit_leaves(lnr, ...)

release_license  Release any IAI license held by the current session.

Description
Julia Equivalent: IAI.release_license

Usage
release_license()

IAI Compatibility
Requires IAI version 3.1 or higher.

Examples
## Not run: iai::release_license()
reset_display_label  Reset the predicted probability displayed to be that of the predicted label when visualizing a learner

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)

------------------------------------------------------------------------
resume_from_checkpoint  Resume training from a checkpoint file

Description
Julia Equivalent: IAI.resume_from_checkpoint

Usage
resume_from_checkpoint(checkpoint_file)

Arguments
checkpoint_file  The location of the checkpoint file.

IAI Compatibility
Requires IAI version 3.1 or higher.

Examples
## Not run: obj <- iai::resume_from_checkpoint("checkpoint.json")
**reward_estimator**  
*Learner for conducting reward estimation with categorical treatments*

**Description**

This function was deprecated and renamed to `categorical_reward_estimator()` in iai 1.4.0. This is for consistency with the IAI v2.1.0 Julia release.

**Usage**

```
reward_estimator(...)  
```

**Arguments**

```
...  
```

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

**Details**

This deprecation is no longer supported as of the IAI v3 release.

**IAI Compatibility**

Requires IAI version 2.2 or lower.

**Examples**

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

---

**roc_curve**  
*Generic function for constructing an ROC curve*

**Description**

Julia Equivalent: `IAI.ROCCurve`

**Usage**

```
roc_curve(obj, ...)  
```

**Arguments**

```
obj  
...  
```

The object controlling which method is used  
Arguments depending on the specific method used
**Description**

Julia Equivalent: `IAI.ROCCurve`

**Usage**

```r
## S3 method for class 'classification_learner'
roc_curve(obj, X, y, ...)
```

**Arguments**

- `obj` The learner or grid to use for prediction.
- `X` The features of the data.
- `y` The labels of the data.
- `...` Refer to the Julia documentation for available parameters.

**Examples**

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

---

**Description**

Julia Equivalent: `IAI.ROCCurve`

**Usage**

```r
## Default S3 method:
roc_curve(obj, y, positive_label = stop("\positive_label\ is required"), ...)
```

**Arguments**

- `obj` The predicted probabilities for each point in the data.
- `y` The true labels of the data.
- `positive_label` The label for which probability is being predicted.
- `...` Additional arguments (unused)
AI Compatibility

Requires IAI version 2.0 or higher.

Examples

```r
## Not run: iai::roc_curve(probs, y, positive_label=positive_label)
```

---

**roc_curve.glmnetcv_classifier**

*Construct an ROC curve using a trained glmnetcv_classifier on the given data*

---

**Description**

Julia Equivalent: *IAI.ROCCurve*

**Usage**

```r
## S3 method for class 'glmnetcv_classifier'
roc_curve(obj, X, y, fit_index = NULL, ...)
```

**Arguments**

- `obj`: The learner or grid to use for prediction.
- `X`: The features of the data.
- `y`: The labels of the data.
- `fit_index`: The index of the fit in the path to use for prediction, defaulting to the best fit if not supplied.
- `...`: Refer to the Julia documentation for available parameters.

**IAI Compatibility**

Requires IAI version 3.0 or higher.

**Examples**

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

Generic function for calculating scores

Usage

score(obj, ...)

Arguments

obj The object controlling which method is used
...

Arguments depending on the specific method used

score.categorical_reward_estimator

Calculate the scores for a categorical reward estimator on the given data

Description

Julia Equivalent: IAI.score

Usage

## S3 method for class 'categorical_reward_estimator'
score(obj, X, ...)

Arguments

obj The learner or grid to evaluate.
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 other available parameters.

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

## Not run: iai::score(lnr, X, treatments, outcomes)
score.default

*Calculate the score for a set of predictions on the given data*

**Description**

Julia Equivalent: `IAI.score`

**Usage**

```r
## Default S3 method:
score(obj, predictions, truths, ...)
```

**Arguments**

- `obj` The type of problem.
- `predictions` The predictions to evaluate.
- `truths` The true target values for these observations.
- `...` Other parameters, including the criterion. Refer to the Julia documentation for available parameters.

**IAI Compatibility**

Requires IAI version 2.1 or higher.

**Examples**

```r
## Not run: iai::score("regression", y_pred, y_true, criterion="mse")
```

---

score.glmnetcv_learner

*Calculate the score for a GLMNet learner on the given data*

**Description**

Julia Equivalent: `IAI.score`

**Usage**

```r
## S3 method for class 'glmnetcv_learner'
score(obj, X, ...)
```

**Examples**

```r
## Not run: iai::score("regression", X, criterion="mse")
```
score.numeric_reward_estimator

**Arguments**

- **obj**: The learner or grid to evaluate.
- **X**: The features of the data.
- **...**: Other parameters, including zero or more target vectors as required by the problem type. `fit_index` can be used to specify the index of the fit in the path to use for prediction, defaulting to the best fit if not supplied. Refer to the Julia documentation for other available parameters.

**IAI Compatibility**

Requires IAI version 2.1 or higher.

**Examples**

```r
## Not run: iai::score(lnr, X, y, fit_index=1)
```

---

**Description**

Julia Equivalent: `IAI.score`

**Usage**

```r
## S3 method for class 'numeric_reward_estimator'
score(obj, X, ...)  
```

**Arguments**

- **obj**: The learner or grid to evaluate.
- **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 other available parameters.

**IAI Compatibility**

Requires IAI version 2.1 or higher.

**Examples**

```r
## Not run: iai::score(lnr, X, treatments, outcomes)
```
**score.optimal_feature_selection_learner**

_Calculate the score for an Optimal Feature Selection learner on the given data_

**Description**

Julia Equivalent: `IAI.score`

**Usage**

```r
## S3 method for class 'optimal_feature_selection_learner'
score(obj, X, ...)
```

**Arguments**

- `obj`: The learner or grid to evaluate.
- `X`: The features of the data.
- `...`: Other parameters, including zero or more target vectors as required by the problem type. If the `coordinated_sparsity` parameter on the learner is TRUE, then `fit_index` must be used to specify which cluster should be used. Refer to the Julia documentation for available parameters.

**IAI Compatibility**

Requires IAI version 1.1 or higher.

**Examples**

```r
## Not run: iai::score(lnr, X, y, fit_index=1)
```

**score.supervised_learner**

_Calculate the score for a model on the given data_

**Description**

Julia Equivalent: `IAI.score`

**Usage**

```r
## S3 method for class 'supervised_learner'
score(obj, X, ...)
```

**Examples**

```r
## Not run: iai::score(lnr, X, y, fit_index=1)
```
**Arguments**

- **obj**  
  The learner or grid to evaluate.
- **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**

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

---

**Description**

Julia Equivalent: `IAI.set_display_label!`

**Usage**

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

**Arguments**

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

**Examples**

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

---

**Description**

Julia Equivalent: `Random.seed!`

**Usage**

```julia
set_julia_seed(seed)
```

**Arguments**

- **seed**  
  The seed to set
set_reward_kernel_bandwidth

Save a new reward kernel bandwidth inside a learner, and return new reward predictions generated using this bandwidth for the original data used to train the learner.

Description
Julia Equivalent: `IAI.set_reward_kernel_bandwidth!`

Usage
```r
set_reward_kernel_bandwidth(lnr, ...)
```

Arguments
- `lnr` The learner to modify
- `...` Refer to the Julia documentation for available parameters.

Examples
```r
## Not run: iai::set_reward_kernel_bandwidth(lnr, random_seed = 1)
```

---

set_params

Set all supplied parameters on a learner

Description
Julia Equivalent: `IAI.set_params!`

Usage
```r
set_params(lnr, ...)
```

Arguments
- `lnr` The learner to modify
- `...` The parameters to set on the learner.

Examples
```r
## Not run: iai::set_params(lnr, random_seed = 1)
```
**set_rich_output_param**

Sets a global rich output parameter

**IAI Compatibility**

Requires IAI version 2.2 or higher.

**Examples**

```r
## Not run: iai::set_reward_kernel_bandwidth(lnr, ...)
```

---

**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**

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

---

**set_threshold**

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.

**Description**

Julia Equivalent: `IAI.set_threshold!`

**Usage**

`set_threshold(lnr, label, threshold, ...)"
show_in_browser.abstract_visualization

Arguments

lnr        The learner to modify.
label      The referenced label.
threshold  The probability threshold above which label will be predicted.
...

Refer to the Julia documentation for available parameters.

Examples

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

<table>
<thead>
<tr>
<th>show_in_browser</th>
<th>Generic function for showing interactive visualization in browser</th>
</tr>
</thead>
</table>

Description

Generic function for showing interactive visualization in browser

Usage

show_in_browser(obj, ...)

Arguments

obj        The object controlling which method is used
...

Arguments depending on the specific method used

<table>
<thead>
<tr>
<th>show_in_browser.abstract_visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show interactive visualization of an object in the default browser</td>
</tr>
</tbody>
</table>

Description

Julia Equivalent: IAI.show_in_browser

Usage

## S3 method for class 'abstract_visualization'
show_in_browser(obj, ...)

Arguments

obj        The object to visualize.
...

Refer to the Julia documentation for available parameters.
show_in_browser.roc_curve

Examples

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

show_in_browser.roc_curve

Show interactive visualization of a roc_curve in the default browser

Description

Julia Equivalent: IAI.show_in_browser

Usage

## S3 method for class 'roc_curve'
show_in_browser(obj, ...)

Arguments

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

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

## Not run: iai::show_in_browser(curve)

show_in_browser.tree_learner

Show interactive tree visualization of a tree learner in the default browser

Description

Julia Equivalent: IAI.show_in_browser

Usage

## S3 method for class 'tree_learner'
show_in_browser(obj, ...)

Arguments

obj  The learner or grid to visualize.
...

Refer to the Julia documentation for available parameters.

IAI Compatibility

Showing a grid search requires IAI version 2.0 or higher.

Examples

```r
eai::show_in_browser(lnr)
```

Generic function for showing interactive questionnaire in browser

Description

Generic function for showing interactive questionnaire in browser

Usage

```r
show_questionnaire(obj, ...)
```

Arguments

obj  The object controlling which method is used
...

Arguments depending on the specific method used

---

Show an interactive questionnaire based on an Optimal Feature Selection learner in default browser

Description

Julia Equivalent: `IAI.show_questionnaire`

Usage

```r
## S3 method for class 'optimal_feature_selection_learner'
show_questionnaire(obj, ...)
```
**Show an interactive questionnaire based on a tree learner in default browser**

**Description**

Julia Equivalent: `IAI.show_questionnaire`

**Usage**

```r
## S3 method for class 'tree_learner'
show_questionnaire(obj, ...)
```

**Arguments**

- `obj`  
  The learner or grid to visualize.

- `...`  
  Refer to the Julia documentation for available parameters.

**IAI Compatibility**

Showing a grid search requires IAI version 2.0 or higher.

**Examples**

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

Conduct a similarity comparison between the final tree in a learner and all trees in a new learner to consider the tradeoff between training performance and similarity to the original tree

Description

Refer to the documentation on tree stability for more information.

Usage

similarity_comparison(lnr, new_lnr, deviations)

Arguments

- lnr: The original learner
- new_lnr: The new learner
- deviations: The deviation between the original tree and each tree in the new learner

Details

Julia Equivalent: IAI.SimilarityComparison

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: iai::similarity_comparison(lnr, new_lnr, deviations)

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

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

split_data

Split the data into training and test datasets

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

```r
## Not run:
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

## End(Not run)
```
stability_analysis  
Conduct a stability analysis of the trees in a tree learner

Description

Refer to the documentation on tree stability for more information.

Usage

stability_analysis(lnr, ...)

Arguments

lnr  The original learner
...  Additional arguments (refer to Julia documentation)

Details

Julia Equivalent: IAI.StabilityAnalysis

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: iai::stability_analysis(lnr, ...)

---

transform  
Impute missing values in a dataframe using a fitted imputation model

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

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

### Description

Julia Equivalent: `IAI.transform_and_expand`

### Usage

```r
transform_and_expand(lnr, X, ...)
```

### Arguments

- `lnr` The learner to use for imputation.
- `X` The dataframe in which to impute missing values.
- `...` Refer to the Julia documentation for available parameters.

### IAI Compatibility

Requires IAI version 3.0 or higher.

#### Examples

```r
## Not run: lnr <- iai::transform_and_expand(lnr, X, type = "finite")
```

---

### tree_plot

Specify an interactive tree visualization of a tree learner

#### Description

Julia Equivalent: `IAI.TreePlot`

#### Usage

```r
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)
```

---

tune_reward_kernel_bandwidth

*Conduct the reward kernel bandwidth tuning procedure for a range of starting bandwidths and return the final tuned values.*

Description

Julia Equivalent: `IAI.tune_reward_kernel_bandwidth`

Usage

```
tune_reward_kernel_bandwidth(lnr, ...)
```

Arguments

*lnr*  
The learner to use for tuning the bandwidth

...  
Refer to the Julia documentation for other parameters

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```
## Not run: iai::tune_reward_kernel_bandwidth(lnr, ...)
```
variable_importance

Generic function for calculating variable importance

**Description**

Generic function for calculating variable importance

**Usage**

```r
variable_importance(obj, ...)
```

**Arguments**

- `obj`: The object controlling which method is used
- `...`: Arguments depending on the specific method used

---

variable_importance.learner

*Generate a ranking of the variables in a learner according to their importance during training. The results are normalized so that they sum to one.*

**Description**

Julia Equivalent: `IAI.variable_importance`

**Usage**

```r
## S3 method for class 'learner'
variable_importance(obj, ...)
```

**Arguments**

- `obj`: The learner to query.
- `...`: Refer to the Julia documentation for available parameters.

**Examples**

```r
## Not run: iai::variable_importance(lnr, ...)
```
variable_importance.optimal_feature_selection_learner

Generate a ranking of the variables in an Optimal Feature Selection learner according to their importance during training. The results are normalized so that they sum to one.

Description

Julia Equivalent: IAI.variable_importance

Usage

## S3 method for class 'optimal_feature_selection_learner'
variable_importance(obj, fit_index = NULL, ...)

Arguments

- **obj**: The learner to query.
- **fit_index**: The index of the cluster to use for prediction, if the coordinated_sparsity parameter on the learner is TRUE.
- **...**: Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

## Not run: iai::variable_importance(lnr, ...)

variable_importance.tree_learner

Generate a ranking of the variables in a tree learner according to their importance during training. The results are normalized so that they sum to one.

Description

Julia Equivalent: IAI.variable_importance

Usage

## S3 method for class 'tree_learner'
variable_importance(obj, ...)

Arguments

- **obj**: The learner to query.
variable_importance_similarity

Arguments

obj The learner to query.

... Refer to the Julia documentation for available parameters.

Examples

## Not run: iai::variable_importance(lnr, ...)

variable_importance_similarity

Calculate similarity between the final tree in a tree learner with all
trees in new tree learner using variable importance scores.

Description

Julia Equivalent: IAI.variable_importance_similarity

Usage

variable_importance_similarity(lnr, new_lnr, ...)

Arguments

lnr The original learner
new_lnr The new learner
... Additional arguments (refer to Julia documentation)

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

## Not run: iai::variable_importance_similarity(lnr, new_lnr)
write_booster

*Write the internal booster saved in the learner to file*

**Description**

Julia Equivalent: `IAI.write_booster`

**Usage**

```r
write_booster(filename, lnr)
```

**Arguments**

- `filename`: Where to save the output.
- `lnr`: The XGBoost learner with the booster to output.

**IAI Compatibility**

Requires IAI version 2.1 or higher.

**Examples**

```r
## Not run: iai::write_booster(file.path(tempdir(), "out.json"), lnr)
```

write_dot

*Output a learner in [Graphviz](http://www.graphviz.org/content/dot-language/.dot format)*

**Description**

Julia Equivalent: `IAI.write_dot`

**Usage**

```r
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**

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

Generic function for writing interactive visualization to file

Description

Generic function for writing interactive visualization to file

Usage

write_html(filename, obj, ...)

Arguments

filename Where to save the output.
obj The object controlling which method is used
... Arguments depending on the specific method used

write_html.abstract_visualization

Output an object as an interactive browser visualization in HTML format

Description

Julia Equivalent: IAI.write_html

Usage

## S3 method for class 'abstract_visualization'
write_html(filename, obj, ...)

Arguments

filename Where to save the output.
obj The object to output.
... Refer to the Julia documentation for available parameters.

Examples

## Not run: iai::write_html(file.path(tempdir(), "out.html"), lnr)
write_html.roc_curve   Output an ROC curve as an interactive browser visualization in HTML format

Description

Julia Equivalent: IAI.write_html

Usage

## S3 method for class 'roc_curve'
write_html(filename, obj, ...)

Arguments

filename Where to save the output.
obj The curve to output.
... Refer to the Julia documentation for available parameters.

IAI Compatibility

Requires IAI version 1.1 or higher.

Examples

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

write_html.tree_learner

Output a tree learner as an interactive browser visualization in HTML format

Description

Julia Equivalent: IAI.write_html

Usage

## S3 method for class 'tree_learner'
write_html(filename, obj, ...)

write_json

Arguments

filename Where to save the output.
obj The learner or grid to output.
...

Refer to the Julia documentation for available parameters.

IAI Compatibility

Outputting a grid search requires IAI version 2.0 or higher.

Examples

```julia
## 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

```julia
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

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

Output a learner as a PDF image

Description

Before using this function, either run `load_graphviz` or ensure that `Graphviz` is installed and on the system `PATH`.

Usage

```
write_pdf(filename, lnr, ...)
```

Arguments

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

Details

Julia Equivalent: `IAI.write_pdf`

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

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

---

write_png

Output a learner as a PNG image

Description

Before using this function, either run `load_graphviz` or ensure that `Graphviz` is installed and on the system `PATH`.

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.

**Details**

Julia Equivalent: `IAI.write_png`

**Examples**

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

---

**write_questionnaire**

*Generic function for writing interactive questionnaire to file*

**Description**

Generic function for writing interactive questionnaire to file

**Usage**

```r
write_questionnaire(filename, obj, ...)
```

**Arguments**

- **filename**: Where to save the output.
- **obj**: The object controlling which method is used
- **...**: Arguments depending on the specific method used

---

**write_questionnaire.optimal_feature_selection_learner**

*Output an Optimal Feature Selection learner as an interactive questionnaire in HTML format*

**Description**

Julia Equivalent: `IAI.write_questionnaire`

**Usage**

```r
## S3 method for class 'optimal_feature_selection_learner'
write_questionnaire(filename, obj, ...)
```
Arguments

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

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

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

---

**Description**

Julia Equivalent: `IAI.write_questionnaire`

**Usage**

```r
## S3 method for class 'tree_learner'
write_questionnaire(filename, obj, ...)
```

**Arguments**

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

**IAI Compatibility**

Outputting a grid search requires IAI version 2.0 or higher.

**Examples**

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

Output a learner as a SVG image

Description
Before using this function, either run load_graphviz or ensure that Graphviz is installed and on the system PATH.

Usage
write_svg(filename, lnr, ...)

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

Details
Julia Equivalent: IAI.write_svg

IAI Compatibility
Requires IAI version 2.1 or higher.

Examples
## Not run: iai::write_svg(file.path(tempdir(), "tree.svg"), lnr)

---
xgboost_classifier

Learner for training XGBoost models for classification problems

Description
Julia Equivalent: IAI.XGBoostClassifier

Usage
xgboost_classifier(...)

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

IAI Compatibility

Requires IAI version 2.1 or higher.

Examples

```r
## Not run: lnr <- iai::xgboost_classifier()
```

---

xgboost_regressor  

Learner for training XGBoost models for regression problems

Description

Julia Equivalent: `IAI.XGBoostRegressor`

Usage

```r
xgboost_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 2.1 or higher.

Examples

```r
## Not run: lnr <- iai::xgboost_regressor()
```

---

xgboost_survival_learner  

Learner for training XGBoost models for survival problems

Description

Julia Equivalent: `IAI.XGBoostSurvivalLearner`

Usage

```r
xgboost_survival_learner(...)```

---
zero_imputation_learner

Arguments

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

IAI Compatibility

Requires IAI version 2.2 or higher.

Examples

```r
## Not run: lnr <- iai::xgboost_survival_learner()
```

zero_imputation_learner

Learner for conducting zero-imputation

Description

Julia Equivalent: `IAI.ZeroImputationLearner`

Usage

generation_learner(...)

Arguments

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

IAI Compatibility

Requires IAI version 3.0 or higher.

Examples

```r
## Not run: lnr <- iai::zero_imputation_learner()
```
Index

acquire_license, 6
add_julia_processes, 7
all_treatment_combinations, 8
apply, 8
apply_nodes, 9
as.mixeddata, 9
autoplot.grid_search, 10, 81
autoplot.roc_curve, 11
autoplot.similarity_comparison, 11
autoplot.stability_analysis, 12
categorical_classification_reward_estimator, 13
categorical_regression_reward_estimator, 13
categorical_reward_estimator, 14
categorical_reward_estimator(). 106
categorical_survival_reward_estimator, 15
cleanup_installation, 15
clean, 16
convert_treatments_to_numeric, 16
copy_splits_and_refit_leaves, 17
decision_path, 17
delete_rich_output_param, 18
equal_propensity_estimator, 18
fit, 19
fit.grid_search, 19
fit.imputation_learner, 20, 25
fit.learner, 21
fit.optimal_feature_selection_learner, 21
fit.and_expand, 22
fit_cv, 23
fit_predict, 23
fit_predict.categorical_reward_estimator, 24
fit_predict.numeric_reward_estimator, 24
fit_transform, 25
fit_transform_cv, 26
get_best_params, 27
get_classification_label, 27
get_classification_proba, 28
get_cluster_assignments, 28
get_cluster_details, 29
get_cluster_distances, 29
get_depth, 30
get_estimation_densities, 30
get_features_used, 31
get_grid_result_details, 32
get_grid_result_summary, 32
get_grid_results, 31
get_learner, 33
get_lower_child, 33
get_machine_id, 34
get_num_fits, 34
get_num_fits.glmnetcv_learner, 35
get_num_fits.optimal_feature_selection_learner, 35
get_num_nodes, 36
get_num_samples, 36
get_params, 37
get_parent, 37
get_policy_treatment_outcome, 38
get_policy_treatment_rank, 38
get_prediction_constant, 39
get_prediction_constant.glmnetcv_learner, 39
get_prediction_constant.optimal_feature_selection_learner, 40
get_prediction_weights, 41
get_prediction_weights.glmnetcv_learner, 41
get_prediction_weights.optimal_feature_selection_learner, 42
get_prescription_treatment_rank, 42  
get_regression_constant, 43  
get_regression_constant.classification_tree_learner, 43  
get_regression_constant.prescription_tree_learner, 44  
get_regression_constant.regression_tree_learner, 44  
get_regression_constant.survival_tree_learner, 44  
get_regression_weights, 46  
get_regression_weights.classification_tree_learner, 46  
get_regression_weights.prescription_tree_learner, 47  
get_regression_weights.regression_tree_learner, 47  
get_regression_weights.survival_tree_learner, 47  
get_rich_output_params, 49  
get_roc_curve_data, 49  
get_split_categories, 50  
get_split_feature, 50  
get_split_threshold, 50  
get_split_weights, 51  
get_stability_results, 52  
get_survival_curve, 52, 53, 88  
get_survival_curve_data, 53  
get_survival_expected_time, 53  
get_survival_hazard, 54  
get_train_errors, 55  
get_tree, 55  
get_upper_child, 56  
glmnetcv_classifier, 56, 94, 108  
glmnetcv_regressor, 57  
glmnetcv_survival_learner, 57, 88, 90  
grid_search, 19, 58  

iai_setup, 59  
imputation_learner, 59  
impute, 60  
impute_cv, 60  
install_julia, 15, 61  
install_system_image, 15, 61  
is_categorical_split, 62  
is_hyperplane_split, 63  
is_leaf, 63  
is_mixed_ordinal_split, 64  
is_mixed_parallel_split, 64  
is_ordinal_split, 65  
is_parallel_split, 65  
load_graphviz, 66, 132, 135  
mean_imputation_learner, 66  
multi_questionnaire, 67  
multi_questionnaire.default, 68  
multi_questionnaire.grid_search, 69  
multi_tree_plot, 69  
multi_tree_plot.default, 70  
multi_tree_plot.grid_search, 71  
numeric_classification_reward_estimator, 71  
numeric_regression_reward_estimator, 72  
numeric_reward_estimator, 73  
numeric_survival_reward_estimator, 73  
opt_knn_imputation_learner, 79  
opt_svm_imputation_learner, 80  
opt_tree_imputation_learner, 80  
optimal_feature_selection_classifier, 74  
optimal_feature_selection_regressor, 75  
optimal_tree_classifier, 75  
optimal_tree_policy_maximizer, 76  
optimal_tree_policy_minimizer, 76  
optimal_tree_prescription_maximizer, 77  
optimal_tree_prescription_minimizer, 77  
optimal_tree_regressor, 78  
optimal_tree_survival_learner, 78  
optimal_tree_survivor, 79  
plot.grid_search, 81  
plot.roc_curve, 81  
plot.similarity_comparison, 82  
plot.stability_analysis, 82  
predict, 83  
predict.categorical_reward_estimator, 83  
predict.glmnetcv_learner, 84  
predict.numeric_reward_estimator, 85
predict.optimal_feature_selection_learner, 85
predict.supervised_learner, 86
predict.survival_learner, 53, 87, 88
predict.expected_survival_time, 87
predict.expected_survival_time.glmnetcv_survival_learner, 88
predict.expected_survival_time.survival_curve, 88
predict.expected_survival_time.survival_learner, 89
predict.hazard, 90
predict.hazard.glmnetcv_survival_learner, 90
predict.hazard.survival_learner, 91
predict.outcomes, 92
predict.outcomes.policy_learner, 92
predict.outcomes.prescription_learner, 93
predict.proba, 93
predict.proba.classification_learner, 94
predict.proba.glmnetcv_classifier, 94
predict.reward, 95
predict.reward.categorical_reward_estimator, 95
predict.reward.numeric_reward_estimator, 96
predict.shap, 97
predict.treatment_outcome, 97
predict.treatment_rank, 98
print.path, 98
prune.trees, 99
questionnaire, 100
questionnaire.optimal_feature_selection_learner, 100
questionnaire.tree_learner, 101
rand.imputation_learner, 103
random_forest_classifier, 101
random_forest_regressor, 102
random_forest_survival_learner, 102
read.json, 103
refit.leaves, 104
release_license, 104
reset.display_label, 105
resume_from_checkpoint, 105
reward.estimator, 106
roc_curve, 106, 117
roc_curve.classification_learner, 107
roc_curve.default, 107
roc_curve.glmnetcv_classifier, 108
score, 109
score.default, 110
score.glmnetcv_learner, 110
score.numeric_reward_estimator, 111
score.optimal_feature_selection_learner, 112
score.supervised_learner, 112
set.display_label, 113
set.julia_seed, 113
set.params, 114
set.reward_kernel_bandwidth, 114
set.rich_output_param, 115
set.threshold, 115
show_in_browser, 116
show_in_browser.abstract_visualization, 116
show_in_browser.roc_curve, 117
show_in_browser.tree_learner, 117
show_questionnaire, 118
show_questionnaire.optimal_feature_selection_learner, 118
show_questionnaire.tree_learner, 119
similarity.comparison, 120
single.knn.imputation_learner, 120
split.data, 121
stability_analysis, 122
transform, 25, 122
transform_and_expand, 123
tree.plot, 123
tune.reward_kernel_bandwidth, 124
variable_importance, 125
variable_importance.learner, 125
variable_importance.optimal_feature_selection_learner, 126
variable_importance.tree_learner, 126
variable_importance.similarity, 127
write_booster, 128
write.dot, 128
write_html, 129
write_html.abstract_visualization, 129
write_html.roc_curve, 130
write_html.tree_learner, 130
write_json, 131
write_pdf, 132
write_png, 132
write_questionnaire, 133
write_questionnaire.optimal_feature_selection_learner, 133
write_questionnaire.tree_learner, 134
write_svg, 135
xgboost_classifier, 135
xgboost_regressor, 136
xgboost_survival_learner, 136
zero_imputation_learner, 137