

Package: AutoMLR (via r-universe)

June 9, 2026

Type Package

Title Automated Multi-Outcome Machine Learning Combination Models

Version 1.0.0

Description Provides automated machine learning workflows for survival analysis, binary classification, continuous outcomes, and ordinal outcomes. The package trains and combines model variants across user-supplied multi-cohort data, evaluates survival models by leave-one-out cross-validation using Harrell's concordance index, binary models by leave-one-out cross-validation using receiver operating characteristic area under the curve, continuous models by out-of-fold root mean squared error and R-squared, and ordinal models by out-of-fold quadratic weighted kappa. It renders reproducible reports in Hypertext Markup Language (HTML) with figures and diagnostics. The survival workflow supports penalized and tree-based Cox proportional hazards models, stepwise Cox models, partial least squares regression for Cox models, supervised principal components, gradient boosting machine Cox models, survival support vector machines (survival-SVM), random survival forests, and optional 'CoxBoost'. The binary workflow supports penalized logistic regression, logistic baselines, gradient boosting machines, random forests, principal component analysis (PCA) logistic regression, and Gaussian naive Bayes variants. Continuous and ordinal workflows reuse an 18-variant regression registry with penalized, linear, boosted, forest, PCA, and baseline families. The optional 'CoxBoost' model is enabled when the suggested 'CoxBoost' package is installed; it is used conditionally and is not a strong dependency. Optional model backends are checked at run time so missing backend packages skip only the affected model variants rather than blocking installation of the whole package. Methods build on Friedman et al. (2010) <[doi:10.18637/jss.v033.i01](https://doi.org/10.18637/jss.v033.i01)>, Bair and Tibshirani (2004) <[doi:10.1371/journal.pbio.0020108](https://doi.org/10.1371/journal.pbio.0020108)>, Ishwaran et al. (2008) <[doi:10.1214/08-AOAS169](https://doi.org/10.1214/08-AOAS169)>, Blanche et al. (2013) <[doi:10.1002/sim.5958](https://doi.org/10.1002/sim.5958)>, and Binder and Schumacher (2008) <[doi:10.1186/1471-2105-9-14](https://doi.org/10.1186/1471-2105-9-14)>.

License MIT + file LICENSE
Encoding UTF-8
Depends R (>= 4.1)
Imports survival, graphics, grDevices, parallel, stats, utils
Suggests CoxBoost, digest, future, future.apply, glmnet, gbm, log4r,
 plsRcox, quadprog, randomForestSRC, superpc, survivalsvm,
 testthat (>= 3.0.0), timeROC
Config/testthat/edition 3
RoxygenNote 7.3.3
NeedsCompilation no
Author Peng Luo [aut, cre]
Maintainer Peng Luo <luopeng@smu.edu.cn>
Repository <https://robinllab.r-universe.dev>
Date/Publication 2026-06-07 18:50:19 UTC
RemoteUrl <https://github.com/cran/AutoMLR>
RemoteRef HEAD
RemoteSha 3d225f8841dd5dce7969ac8f7cc64c95b5f847a0

Contents

automlr_input_to_binary_xy	4
automlr_input_to_continuous_xy	5
automlr_input_to_ordinal_xy	5
automlr_input_to_surv_xy	6
automlr_parameters	6
binary_auc	8
binary_pr_auc	9
binarymlr_parameters	9
check_automlr_dependencies	11
continuous_cor	11
continuous_mae	12
continuous_r2	12
continuous_rmse	13
continuousmlr_parameters	13
count_binary_combinations	15
count_continuous_combinations	15
count_ordinal_combinations	16
count_surv_combinations	17
disable_auto_logging	18
evaluate_algorithm_loocv	18
evaluate_algorithms_loocv	19
evaluate_binary_algorithm_loocv	20
evaluate_binary_algorithms_loocv	21

evaluate_binary_combinations	22
evaluate_continuous_algorithm	23
evaluate_continuous_algorithms	24
evaluate_continuous_combinations	24
evaluate_ordinal_algorithms	25
evaluate_ordinal_combinations	26
evaluate_surv_combinations	27
export_binary_results	28
export_continuous_results	29
export_extreme_screen_results	30
export_ordinal_results	31
export_surv_results	32
extreme_surv_screen	33
fit_binary_ensemble	34
fit_continuous_ensemble	36
fit_ordinal_ensemble	37
fit_surv_ensemble	38
get_binary_registry	40
get_continuous_registry	40
get_ordinal_registry	40
get_surv_registry	41
initialize_auto_logging	41
list_binary_algorithms	42
list_binary_model_variants	42
list_continuous_algorithms	43
list_continuous_model_variants	43
list_model_variants	44
list_ordinal_algorithms	44
list_ordinal_model_variants	45
list_surv_algorithms	45
loocv_auc	46
loocv_cindex	47
ordinal_accuracy	48
ordinal_balanced_accuracy	48
ordinal_mae	49
ordinal_qwk	49
ordinalmlr_parameters	50
parallel_lapply	51
predict.automlr_binary_ensemble	52
predict.automlr_continuous_ensemble	52
predict.automlr_ordinal_ensemble	53
predict.automlr_surv_ensemble	54
prepare_binary_cohort_input	54
prepare_cohort_input	55
prepare_continuous_cohort_input	56
prepare_ordinal_cohort_input	57
print.automlr_dependency_report	57
print.automlr_extreme_screen	58

recommend_binary_auc_threshold	58
recommend_continuous_r2_threshold	59
recommend_ordinal_qwk_threshold	60
recommend_surv_cindex_threshold	60
render_binary_report	61
render_continuous_report	62
render_ordinal_report	62
render_surv_report	63
report_binary_cohort_intersection	64
report_cohort_intersection	65
report_continuous_cohort_intersection	65
report_ordinal_cohort_intersection	66
start_parallel	66
stop_parallel	67
summarize_base_models	67
summarize_binary_analysis_results	68
summarize_binary_base_models	68
summarize_binary_data_preparation	69
summarize_binary_ensemble_results	70
summarize_binary_explainability_results	70
summarize_data_preparation	71
summarize_ensemble_results	72
summarize_explainability_results	72
summarize_extreme_screen_results	73
summarize_surv_analysis_results	74

Index	75
--------------	-----------

automlr_input_to_binary_xy

Extract modeling matrices from prepared binary input.

Description

Converts an object returned by `prepare_binary_cohort_input()` into the `X`, `y`, and `cohort-label` objects used by lower-level binary functions such as `evaluate_binary_algorithm_loocv()`, `evaluate_binary_algorithm` and `evaluate_binary_combinations()`.

Usage

```
automlr_input_to_binary_xy(input)
```

Arguments

`input` An object returned by `prepare_binary_cohort_input()`.

Value

A list with components X (numeric feature matrix), y (0/1 outcome), `stability_groups` (cohort labels aligned to rows), `cohort` (alias of `stability_groups`), and `data` (combined modeling data frame).

`automlr_input_to_continuous_xy`

Extract modeling matrices from prepared continuous input.

Description

Converts an object returned by `prepare_continuous_cohort_input()` into the X , y , and cohort-label objects used by lower-level continuous-outcome functions.

Usage

```
automlr_input_to_continuous_xy(input)
```

Arguments

`input` An object returned by `prepare_continuous_cohort_input()`.

Value

A list with components X (numeric feature matrix), y (numeric outcome), `stability_groups` (cohort labels aligned to rows), `cohort` (alias of `stability_groups`), and `data` (combined modeling data frame).

`automlr_input_to_ordinal_xy`

Extract modeling matrices from prepared ordinal input.

Description

Converts an object returned by `prepare_ordinal_cohort_input()` into the X , y , and cohort-label objects used by lower-level ordinal-outcome functions.

Usage

```
automlr_input_to_ordinal_xy(input)
```

Arguments

`input` An object returned by `prepare_ordinal_cohort_input()`.

Value

A list with components *X* (numeric feature matrix), *y* (integer ordinal outcome codes), *stability_groups* (cohort labels aligned to rows), *cohort* (alias of *stability_groups*), and *data* (combined modeling data frame).

automlr_input_to_surv_xy

Extract modeling matrices from prepared survival input.

Description

Converts an object returned by `prepare_cohort_input()` into the *X*, *y*, and cohort-label objects used by lower-level survival functions such as `evaluate_algorithm_loocv()`, `evaluate_algorithms_loocv()`, and `evaluate_surv_combinations()`.

Usage

```
automlr_input_to_surv_xy(input)
```

Arguments

input An object returned by `prepare_cohort_input()`.

Value

A list with components:

X Numeric feature matrix restricted to shared features.

y A `survival::Surv` outcome object.

stability_groups Cohort labels aligned to rows of *X*.

cohort Alias of *stability_groups*.

data The combined modeling data frame.

automlr_parameters

Default parameters for AutoMLR survival pipeline.

Description

Returns a flat list of knobs consumed by data preparation, feature screening, model fitting, and evaluation.

Usage

```

automlr_parameters(
  seed = 123L,
  screen_by_univariate_cox = TRUE,
  univariate_cox_p_cutoff = 0.05,
  screen_by_variance = TRUE,
  variance_quantile_cutoff = 0.1,
  loocv = TRUE,
  min_cindex_accept = 0.6,
  auto_min_cindex = FALSE,
  auto_quantile = 0.5,
  eval_times = c(365, 1095, 1825),
  time_unit = c("auto", "days", "months", "years"),
  algorithms = NULL,
  surv_svm_resampling = c("kfold", "loocv"),
  surv_svm_k_folds = 5L,
  n_cores = 1L,
  stability_resamples = 0L,
  stability_fraction = 0.8,
  stability_weight = 0.1,
  verbose = TRUE
)

```

Arguments

seed	Base random seed.
screen_by_univariate_cox	Logical, run univariate Cox p-value screen.
univariate_cox_p_cutoff	P-value cutoff for the univariate Cox screen.
screen_by_variance	Logical, drop low-variance features.
variance_quantile_cutoff	Drop features with variance below this quantile of all feature variances (e.g. 0.1 = drop bottom 10%).
loocv	Logical, use LOOCV during model selection (V1 evaluation protocol; paper 1 recipe).
min_cindex_accept	Minimum LOOCV C-index to accept a model.
auto_min_cindex	Logical. If TRUE and <code>fit_surv_ensemble()</code> uses <code>strategy = "threshold"</code> , replace <code>min_cindex_accept</code> with a recommended cutoff computed from candidate-model LOOCV C-index values.
auto_quantile	Quantile used when an automatic threshold is requested. 0.50 uses the median candidate metric; larger values are more selective.
eval_times	Vector of time points (same unit as time column) at which to compute time-dependent AUC. Default ~1-, 3-, 5-year style.

time_unit	Time unit for survival time and eval_times, one of "auto", "days", "months", or "years". "auto" infers the unit from the observed follow-up scale for plot labels only.
algorithms	Character vector of algorithm keys to run; default = all 10 in list_surv_algorithms().
surv_svm_resampling	Resampling method for survival support vector machine candidates. The default "kfold" avoids known single-row prediction failures from survivalsvm during LOOCV. Set "loocv" to use leave-one-out evaluation for this model as well.
surv_svm_k_folds	Number of folds used when surv_svm_resampling = "kfold".
n_cores	Integer, workers for parallel execution. 1 = sequential.
stability_resamples	Integer, number of repeated subsamples used for optional stability diagnostics. 0 disables resampling.
stability_fraction	Fraction of samples used in each stability subsample.
stability_weight	Non-negative penalty multiplier used only when a stability-weighted ranking or weight method is explicitly requested.
verbose	Logical, print progress messages.

Value

A named list.

binary_auc	<i>ROC AUC for binary outcomes.</i>
------------	-------------------------------------

Description

ROC AUC for binary outcomes.

Usage

```
binary_auc(y, prob)
```

Arguments

y	Binary 0/1 outcome.
prob	Predicted probability for the positive class.

Value

Scalar ROC AUC, or NA_real_ when not estimable.

binary_pr_auc	<i>Precision-recall AUC for binary outcomes.</i>
---------------	--

Description

Precision-recall AUC for binary outcomes.

Usage

```
binary_pr_auc(y, prob)
```

Arguments

y	Binary 0/1 outcome.
prob	Predicted probability for the positive class.

Value

Scalar PR-AUC, or NA_real_ when not estimable.

binarymlr_parameters	<i>Default parameters for AutoMLR binary-classification workflows.</i>
----------------------	--

Description

The binary workflow mirrors the survival workflow, but uses cross-validated ROC AUC as the primary selection metric and keeps PR-AUC, threshold metrics, calibration, and cohort stability as diagnostics.

Usage

```
binarymlr_parameters(
  seed = 123L,
  algorithms = NULL,
  loocv = TRUE,
  resampling = NULL,
  k_folds = 5L,
  repeats = 1L,
  min_auc_accept = 0.6,
  auto_min_auc = FALSE,
  auto_quantile = 0.5,
  positive_class = 1,
  threshold_methods = c("youden", "fixed_0.5"),
  missing_fraction_cutoff = 0.2,
  screen_by_variance = TRUE,
```

```

variance_quantile_cutoff = 0,
standardize_features = FALSE,
n_cores = 1L,
stability_resamples = 0L,
stability_fraction = 0.8,
stability_weight = 0.1,
verbose = TRUE
)

```

Arguments

seed	Base random seed.
algorithms	Character vector of binary algorithm keys. Defaults to all entries returned by <code>list_binary_algorithms()</code> .
loocv	Logical, use leave-one-out cross-validation.
resampling	Resampling scheme: "loocv", "kfold", or "repeated_kfold".
k_folds	Number of folds when resampling uses k-fold CV.
repeats	Number of repeats for repeated k-fold CV.
min_auc_accept	Minimum AUC accepted by threshold-style selection.
auto_min_auc	Logical. If TRUE and <code>fit_binary_ensemble()</code> uses <code>strategy = "threshold"</code> , replace <code>min_auc_accept</code> with a recommended cutoff computed from candidate-model AUC values.
auto_quantile	Quantile used when an automatic threshold is requested. 0.50 uses the median candidate metric; larger values are more selective.
positive_class	Positive-class label used during data preparation.
threshold_methods	Threshold summaries to export. Supported values are "youden" and "fixed_0.5".
missing_fraction_cutoff	Drop features with missing fraction above this cutoff before modeling.
screen_by_variance	Logical, drop zero / low-variance features.
variance_quantile_cutoff	Optional lower variance quantile to drop.
standardize_features	Logical, center and scale features before modeling.
n_cores	Integer, number of fold workers. 1 is sequential.
stability_resamples	Number of optional stability subsamples.
stability_fraction	Fraction of samples in each stability subsample.
stability_weight	Penalty multiplier for stability-weighted ranking.
verbose	Logical, print progress.

Value

A named list.

check_automlr_dependencies

Check optional AutoMLR model backends and feature dependencies.

Description

AutoMLR keeps heavyweight model engines in Suggests so the package can be installed even when some optional modelling backends are unavailable. This helper reports which model variants and optional features are currently available in the user's R library and which packages would be needed to enable the skipped pieces.

Usage

```
check_automlr_dependencies(workflow = "all")
```

Arguments

`workflow` Character vector. Use "all" to check survival, binary, continuous, and ordinal workflows, or select any subset of "survival", "binary", "continuous", and "ordinal".

Value

A list of class "automlr_dependency_report" with two data frames: `algorithms`, containing one row per algorithm registry entry, and `optional_features`, containing non-model optional capabilities such as logging, parallel execution, and time-dependent ROC diagnostics.

continuous_cor

Correlation between observed and predicted continuous outcomes.

Description

Correlation between observed and predicted continuous outcomes.

Usage

```
continuous_cor(y, pred, method = c("pearson", "spearman"))
```

Arguments

`y` Observed numeric outcome.
`pred` Predicted numeric outcome.
`method` Correlation method.

Value

A numeric scalar.

continuous_mae	<i>Mean absolute error for continuous predictions.</i>
----------------	--

Description

Mean absolute error for continuous predictions.

Usage

```
continuous_mae(y, pred)
```

Arguments

y	Observed numeric outcome.
pred	Predicted numeric outcome.

Value

A numeric scalar.

continuous_r2	<i>Coefficient of determination for continuous predictions.</i>
---------------	---

Description

Coefficient of determination for continuous predictions.

Usage

```
continuous_r2(y, pred)
```

Arguments

y	Observed numeric outcome.
pred	Predicted numeric outcome.

Value

A numeric scalar.

continuous_rmse	<i>Root mean squared error for continuous predictions.</i>
-----------------	--

Description

Root mean squared error for continuous predictions.

Usage

```
continuous_rmse(y, pred)
```

Arguments

y	Observed numeric outcome.
pred	Predicted numeric outcome.

Value

A numeric scalar.

continuousmlr_parameters	<i>Default parameters for AutoMLR continuous-outcome workflows.</i>
--------------------------	---

Description

Default parameters for AutoMLR continuous-outcome workflows.

Usage

```
continuousmlr_parameters(  
  seed = 123L,  
  algorithms = NULL,  
  resampling = "loocv",  
  k_folds = 5L,  
  repeats = 1L,  
  min_r2_accept = 0,  
  auto_min_r2 = FALSE,  
  auto_quantile = 0.5,  
  missing_fraction_cutoff = 0.2,  
  screen_by_variance = TRUE,  
  variance_quantile_cutoff = 0,  
  standardize_features = FALSE,  
  n_cores = 1L,  
  stability_resamples = 0L,
```

```

    stability_fraction = 0.8,
    stability_weight = 0.1,
    verbose = TRUE
)

```

Arguments

seed	Base random seed.
algorithms	Character vector of continuous algorithm keys.
resampling	Resampling scheme: "loocv", "kfold", or "repeated_kfold".
k_folds	Number of folds for k-fold CV.
repeats	Number of repeats for repeated k-fold CV.
min_r2_accept	Minimum R-squared accepted by threshold-style selection.
auto_min_r2	Logical. If TRUE and <code>fit_continuous_ensemble()</code> uses <code>strategy = "threshold"</code> , replace <code>min_r2_accept</code> with a recommended cutoff computed from candidate-model R-squared values.
auto_quantile	Quantile used when an automatic threshold is requested. 0.50 uses the median candidate metric; larger values are more selective.
missing_fraction_cutoff	Drop features above this missing fraction.
screen_by_variance	Logical, drop zero / low-variance features.
variance_quantile_cutoff	Optional lower variance quantile to drop.
standardize_features	Logical, center and scale features.
n_cores	Integer, number of fold workers.
stability_resamples	Number of optional stability subsamples.
stability_fraction	Fraction of samples in each stability subsample.
stability_weight	Penalty multiplier for stability-aware ranking.
verbose	Logical, print progress.

Value

A named list.

`count_binary_combinations`*Count binary model combinations without fitting.*

Description

Count binary model combinations without fitting.

Usage

```
count_binary_combinations(  
  params = binarymlr_parameters(),  
  algorithms = params$algorithms,  
  min_size = 1L,  
  max_size = 2L,  
  allow_same_algorithm = FALSE  
)
```

Arguments

<code>params</code>	Output of <code>binarymlr_parameters()</code> .
<code>algorithms</code>	Binary algorithm keys.
<code>min_size</code>	Minimum combination size.
<code>max_size</code>	Maximum combination size.
<code>allow_same_algorithm</code>	Logical, allow same base algorithm variants in one combination.

Value

A list with candidate and combination counts.

`count_continuous_combinations`*Count continuous model combinations without fitting.*

Description

Count continuous model combinations without fitting.

Usage

```
count_continuous_combinations(
  params = continuousmlr_parameters(),
  algorithms = params$algorithms,
  min_size = 1L,
  max_size = 2L,
  allow_same_algorithm = FALSE
)
```

Arguments

params	Output of continuousmlr_parameters().
algorithms	Continuous algorithm keys.
min_size	Minimum combination size.
max_size	Maximum combination size.
allow_same_algorithm	Logical.

Value

A list with candidate and combination counts.

count_ordinal_combinations

Count ordinal model combinations without fitting.

Description

Count ordinal model combinations without fitting.

Usage

```
count_ordinal_combinations(
  params = ordinalmlr_parameters(),
  algorithms = params$algorithms,
  min_size = 1L,
  max_size = 2L,
  allow_same_algorithm = FALSE
)
```

Arguments

params	Output of ordinalmlr_parameters().
algorithms	Ordinal algorithm keys.
min_size	Minimum combination size.
max_size	Maximum combination size.
allow_same_algorithm	Logical.

Value

A list with candidate and combination counts.

count_surv_combinations

Count model combinations without fitting models.

Description

Count model combinations without fitting models.

Usage

```
count_surv_combinations(  
  params = automlr_parameters(),  
  algorithms = params$algorithms,  
  min_size = 2L,  
  max_size = 2L,  
  allow_same_algorithm = FALSE  
)
```

Arguments

params	Output of automlr_parameters().
algorithms	Character vector of registry keys.
min_size	Minimum combination size.
max_size	Maximum combination size.
allow_same_algorithm	Logical, allow two variants from the same base algorithm to appear in one combination.

Value

A list with candidate and combination counts.

disable_auto_logging *Disable AutoMLR auto logging.*

Description

Disable AutoMLR auto logging.

Usage

```
disable_auto_logging()
```

Value

Invisibly returns TRUE when logging was disabled and FALSE when logging was already inactive.

evaluate_algorithm_loocv

Run LOOCV for a named algorithm in the registry.

Description

Convenience wrapper: looks up algo_key in get_surv_registry(), takes the first row of grid(params) as the hyperparameters, and calls loocv_cindex().

Usage

```
evaluate_algorithm_loocv(
  algo_key,
  X,
  y,
  params = automlr_parameters(),
  hparam = NULL,
  candidate_key = NULL,
  verbose = FALSE
)
```

Arguments

algo_key	Character, one of list_surv_algorithms().
X	Feature matrix.
y	survival::Surv object.
params	Output of automlr_parameters(); used to pick the grid row.
hparam	Optional named list of hyperparameters. If NULL, the first row of the registry grid is used.
candidate_key	Optional identifier for this algorithm + hyperparameter variant.
verbose	Logical.

Value

A list — the output of `loocv_cindex()` plus `algo_key`, `algo_label`, `candidate_key`, and `hparam` used.

evaluate_algorithms_loocv

Evaluate multiple survival model variants by LOOCV C-index.

Description

Evaluate multiple survival model variants by LOOCV C-index.

Usage

```
evaluate_algorithms_loocv(
  X,
  y,
  params = automlr_parameters(),
  algorithms = params$algorithms,
  stability_groups = NULL,
  stability_resamples = params$stability_resamples %||% 0L,
  stability_fraction = params$stability_fraction %||% 0.8,
  verbose = params$verbose
)
```

Arguments

<code>X</code>	Feature matrix, or an "automlr_input" object returned by <code>prepare_cohort_input()</code> .
<code>y</code>	<code>survival::Surv</code> object. Leave <code>NULL</code> when <code>X</code> is an "automlr_input" object.
<code>params</code>	Output of <code>automlr_parameters()</code> .
<code>algorithms</code>	Character vector of registry keys to expand and evaluate.
<code>stability_groups</code>	Optional group/queue labels used to compute per-candidate stability diagnostics from out-of-fold risks.
<code>stability_resamples</code>	Number of repeated subsamples for stability diagnostics.
<code>stability_fraction</code>	Fraction of samples in each stability subsample.
<code>verbose</code>	Logical.

Value

A list of class "automlr_loocv_set" with a summary table and raw per-candidate results.

`evaluate_binary_algorithm_loocv`*Evaluate one binary algorithm by LOOCV AUC.*

Description

Evaluate one binary algorithm by LOOCV AUC.

Usage

```
evaluate_binary_algorithm_loocv(  
  algo_key,  
  X,  
  y,  
  params = binarymlr_parameters(),  
  hparam = NULL,  
  candidate_key = NULL,  
  verbose = FALSE,  
  resampling_plan = NULL  
)
```

Arguments

<code>algo_key</code>	Algorithm key.
<code>X</code>	Feature matrix.
<code>y</code>	Binary 0/1 outcome.
<code>params</code>	Output of <code>binarymlr_parameters()</code> .
<code>hparam</code>	Optional hyperparameters.
<code>candidate_key</code>	Optional variant identifier.
<code>verbose</code>	Logical.
<code>resampling_plan</code>	Optional internal resampling plan.

Value

A list with LOOCV results and metadata.

`evaluate_binary_algorithms_loocv`*Evaluate binary model variants by LOOCV AUC.*

Description

Evaluate binary model variants by LOOCV AUC.

Usage

```
evaluate_binary_algorithms_loocv(  
  X,  
  y,  
  params = binarymlr_parameters(),  
  algorithms = params$algorithms,  
  stability_groups = NULL,  
  stability_resamples = params$stability_resamples %||% 0L,  
  stability_fraction = params$stability_fraction %||% 0.8,  
  verbose = params$verbose  
)
```

Arguments

<code>X</code>	Feature matrix.
<code>y</code>	Binary 0/1 outcome.
<code>params</code>	Output of <code>binarymlr_parameters()</code> .
<code>algorithms</code>	Algorithm keys.
<code>stability_groups</code>	Optional cohort labels.
<code>stability_resamples</code>	Optional stability resamples.
<code>stability_fraction</code>	Stability subsample fraction.
<code>verbose</code>	Logical.

Value

A list of class "automlr_binary_loocv_set".

```
evaluate_binary_combinations
```

Evaluate all-subset binary probability combinations.

Description

Evaluate all-subset binary probability combinations.

Usage

```
evaluate_binary_combinations(
  loocv_set,
  y,
  min_size = 1L,
  max_size = 2L,
  weight_method = c("auc", "equal", "auc_stability"),
  allow_same_algorithm = FALSE,
  max_failed_fraction = 0.2,
  min_prob_sd = 1e-08,
  stability_groups = NULL,
  stability_resamples = loocv_set$params$stability_resamples %||% 0L,
  stability_fraction = loocv_set$params$stability_fraction %||% 0.8,
  rank_by = c("auc", "stability_weighted"),
  stability_weight = loocv_set$params$stability_weight %||% 0.1,
  top_n = 50L
)
```

Arguments

loocv_set	Output of evaluate_binary_algorithms_loocv().
y	Binary 0/1 outcome.
min_size	Minimum member count.
max_size	Maximum member count.
weight_method	One of "auc", "equal", or "auc_stability".
allow_same_algorithm	Logical.
max_failed_fraction	Maximum LOOCV failure fraction.
min_prob_sd	Minimum probability standard deviation.
stability_groups	Optional cohort labels.
stability_resamples	Optional stability resamples.
stability_fraction	Stability subsample fraction.

rank_by	Ranking method.
stability_weight	Stability penalty multiplier.
top_n	Number of rows to keep.

Value

A list of class "automlr_binary_combination_set".

evaluate_continuous_algorithm

Evaluate one continuous algorithm by out-of-fold performance.

Description

Evaluate one continuous algorithm by out-of-fold performance.

Usage

```
evaluate_continuous_algorithm(
  algo_key,
  X,
  y,
  params = continuousmlr_parameters(),
  hparam = NULL,
  candidate_key = NULL,
  verbose = FALSE,
  resampling_plan = NULL
)
```

Arguments

algo_key	Algorithm key.
X	Feature matrix.
y	Numeric outcome.
params	Output of continuousmlr_parameters().
hparam	Optional hyperparameters.
candidate_key	Optional variant identifier.
verbose	Logical.
resampling_plan	Optional internal resampling plan.

Value

A list with OOF predictions and metrics.

`evaluate_continuous_algorithms`*Evaluate continuous model variants by out-of-fold performance.*

Description

Evaluate continuous model variants by out-of-fold performance.

Usage

```
evaluate_continuous_algorithms(  
  X,  
  y,  
  params = continuousmlr_parameters(),  
  algorithms = params$algorithms,  
  verbose = params$verbose  
)
```

Arguments

<code>X</code>	Feature matrix.
<code>y</code>	Numeric outcome.
<code>params</code>	Output of <code>continuousmlr_parameters()</code> .
<code>algorithms</code>	Algorithm keys.
<code>verbose</code>	Logical.

Value

A list of class "automlr_continuous_resample_set".

`evaluate_continuous_combinations`*Evaluate all-subset continuous prediction combinations.*

Description

Evaluate all-subset continuous prediction combinations.

Usage

```

evaluate_continuous_combinations(
  resample_set,
  y,
  min_size = 1L,
  max_size = 2L,
  weight_method = c("inverse_rmse", "equal", "r2"),
  allow_same_algorithm = FALSE,
  max_failed_fraction = 0.2,
  min_pred_sd = 1e-08,
  rank_by = c("rmse", "r2"),
  top_n = 50L
)

```

Arguments

resample_set	Output of evaluate_continuous_algorithms().
y	Numeric outcome.
min_size	Minimum member count.
max_size	Maximum member count.
weight_method	One of "inverse_rmse", "equal", or "r2".
allow_same_algorithm	Logical.
max_failed_fraction	Maximum sample prediction failure fraction.
min_pred_sd	Minimum prediction standard deviation.
rank_by	Ranking method.
top_n	Number of rows to keep.

Value

A list of class "automlr_continuous_combination_set".

evaluate_ordinal_algorithms

Evaluate ordinal model variants by out-of-fold performance.

Description

Ordinal outcomes are encoded as ordered integer scores and fitted with the continuous-model registry; predictions are rounded back to ordered classes for QWK, accuracy, balanced accuracy, and class MAE.

Usage

```
evaluate_ordinal_algorithms(  
  X,  
  y,  
  params = ordinalmlr_parameters(),  
  algorithms = params$algorithms,  
  verbose = params$verbose  
)
```

Arguments

X	Feature matrix.
y	Ordinal positive integer outcome codes.
params	Output of ordinalmlr_parameters().
algorithms	Algorithm keys.
verbose	Logical.

Value

A list of class "automlr_ordinal_resample_set".

evaluate_ordinal_combinations

Evaluate all-subset ordinal score combinations.

Description

Evaluate all-subset ordinal score combinations.

Usage

```
evaluate_ordinal_combinations(  
  resample_set,  
  y,  
  min_size = 1L,  
  max_size = 2L,  
  weight_method = c("qwk", "equal", "inverse_mae"),  
  allow_same_algorithm = FALSE,  
  rank_by = c("qwk", "class_mae"),  
  top_n = 50L  
)
```

Arguments

resample_set	Output of evaluate_ordinal_algorithms().
y	Ordinal integer outcome codes.
min_size	Minimum member count.
max_size	Maximum member count.
weight_method	One of "qwk", "equal", or "inverse_mae".
allow_same_algorithm	Logical.
rank_by	Ranking method.
top_n	Number of rows to keep.

Value

A list of class "automlr_ordinal_combination_set".

evaluate_surv_combinations

Evaluate all-subset survival model combinations.

Description

Builds combinations from the out-of-fold risk scores in evaluate_algorithms_loocv(). For each subset, member risks are standardized and averaged using equal weights or C-index-derived weights; the resulting combination is scored by Harrell's C-index.

Usage

```
evaluate_surv_combinations(
  loocv_set,
  y,
  min_size = 1L,
  max_size = 2L,
  weight_method = c("cindex", "equal", "cindex_stability"),
  allow_same_algorithm = FALSE,
  max_failed_fraction = 0.2,
  min_risk_sd = 1e-08,
  stability_groups = NULL,
  stability_resamples = loocv_set$params$stability_resamples %||% 0L,
  stability_fraction = loocv_set$params$stability_fraction %||% 0.8,
  rank_by = c("cindex", "stability_weighted"),
  stability_weight = loocv_set$params$stability_weight %||% 0.1,
  diagnostic_times = NULL,
  top_n = 50L
)
```

Arguments

loocv_set	Output of evaluate_algorithms_loocv().
y	A survival::Surv object used to score combined risks.
min_size	Minimum number of member model variants in a combination.
max_size	Maximum number of member model variants in a combination.
weight_method	One of "cindex", "equal", or "cindex_stability".
allow_same_algorithm	Logical, allow two variants from the same base algorithm to appear in one combination.
max_failed_fraction	Maximum allowed LOOCV fold failure fraction for a candidate to enter combinations.
min_risk_sd	Minimum standard deviation of LOOCV risk scores for a candidate to enter combinations.
stability_groups	Optional group/queue labels used for combination stability diagnostics.
stability_resamples	Number of repeated subsamples for stability diagnostics.
stability_fraction	Fraction of samples in each stability subsample.
rank_by	Ranking method. "cindex" keeps C-index as the primary selection criterion; "stability_weighted" subtracts a stability penalty.
stability_weight	Non-negative multiplier for the stability penalty when rank_by = "stability_weighted".
diagnostic_times	Optional time points for time-dependent AUC diagnostics. Requires the suggested timeROC package.
top_n	Number of top rows to keep in the returned summary.

Value

A list of class "automlr_combination_set".

export_binary_results *Export binary AutoMLR results.*

Description

Export binary AutoMLR results.

Usage

```
export_binary_results(
  object,
  output_dir = "automlr_binary_results",
  publication = TRUE,
  formats = c("pdf", "png"),
  dpi = 300L,
  top_n = 20L,
  overwrite = TRUE,
  summary_language = c("bilingual", "en", "zh")
)
```

Arguments

object	Object returned by fit_binary_ensemble().
output_dir	Output directory.
publication	Logical, write publication-style figures.
formats	Figure formats: "png" and/or "pdf".
dpi	PNG resolution.
top_n	Number of top rows.
overwrite	Logical.
summary_language	"bilingual", "en", or "zh".

Value

A list of exported paths.

export_continuous_results

Export continuous AutoMLR results.

Description

Export continuous AutoMLR results.

Usage

```
export_continuous_results(
  object,
  output_dir = "automlr_continuous_results",
  publication = TRUE,
  formats = c("pdf", "png"),
  dpi = 300L,
  top_n = 20L,
  overwrite = TRUE
)
```

Arguments

object	Object returned by <code>fit_continuous_ensemble()</code> .
output_dir	Output directory.
publication	Logical, write publication-style figures.
formats	Figure formats.
dpi	PNG resolution.
top_n	Number of top rows.
overwrite	Logical.

Value

A list of exported paths.

export_extreme_screen_results

Export extreme-screening tables and publication-style audit figures

Description

Writes the complete apparent C-index table, apparent top-N table, seed-search table, best rows, failure diagnostics, summary CSVs, a Markdown summary report, and a Morandi-toned audit figure set for an object returned by `extreme_surv_screen()`.

Usage

```
export_extreme_screen_results(
  x,
  output_dir,
  formats = c("png", "pdf"),
  top_n = 10L,
  top_seed_rows = 30L,
  dpi = 300L,
  summary_language = c("bilingual", "en", "zh")
)
```

Arguments

x	An object returned by <code>extreme_surv_screen()</code> .
output_dir	Directory to write tables, figures, and the RDS object.
formats	Figure formats. Any subset of "png" and "pdf".
top_n	Number of apparent top combinations to emphasize in figures.
top_seed_rows	Number of seed-search rows to show in ranked-row figures.
dpi	PNG resolution.
summary_language	Language used in <code>summary_report.md</code> , either "bilingual", "zh", or "en".

Value

A list with written table and figure paths.

`export_ordinal_results`

Export ordinal AutoMLR results.

Description

Export ordinal AutoMLR results.

Usage

```
export_ordinal_results(  
  object,  
  output_dir = "automlr_ordinal_results",  
  publication = TRUE,  
  formats = c("pdf", "png"),  
  dpi = 300L,  
  top_n = 20L,  
  overwrite = TRUE  
)
```

Arguments

<code>object</code>	Object returned by <code>fit_ordinal_ensemble()</code> .
<code>output_dir</code>	Output directory.
<code>publication</code>	Logical, write publication-style figures.
<code>formats</code>	Figure formats.
<code>dpi</code>	PNG resolution.
<code>top_n</code>	Number of top rows.
<code>overwrite</code>	Logical.

Value

A list of exported paths.

export_surv_results *Export AutoMLR survival results as a reproducible result bundle.*

Description

Creates a directory containing an HTML report, publication figures, diagnostic figures, CSV tables, fitted R objects, risk scores, optional timeROC diagnostics, cohort diagnostics, a deduplicated final publication figure set, and session metadata.

Usage

```
export_surv_results(  
  object,  
  output_dir = "automlr_results",  
  publication = TRUE,  
  formats = c("pdf", "png"),  
  dpi = 300L,  
  top_n = 20L,  
  overwrite = TRUE,  
  summary_language = c("bilingual", "en", "zh")  
)
```

Arguments

object	An object returned by <code>fit_surv_ensemble()</code> .
output_dir	Directory for the exported result bundle.
publication	Logical, create publication-style figures.
formats	Figure formats. Supported values are "pdf" and "png".
dpi	Resolution for PNG figures.
top_n	Number of top models / combinations to show in figures.
overwrite	Logical, overwrite report files where applicable.
summary_language	Language used in Markdown summaries, either "bilingual", "en", or "zh".

Value

A list with paths to exported files.

extreme_surv_screen *Extreme two-stage screening for survival model combinations*

Description

Runs an optimistic "full data as train and validation" screen first, then searches random 70/30 train-validation splits only for the top combinations. The first stage estimates an apparent upper bound and should not be reported as external validation performance. Returned apparent-stage tables include performance_scope and performance_note columns carrying this warning.

Usage

```
extreme_surv_screen(
  X,
  y = NULL,
  params = automlr_parameters(),
  algorithms = params$algorithms,
  top_n = 5L,
  seeds = 1:500,
  train_fraction = 0.7,
  min_models = 1L,
  max_models = 2L,
  weight_method = c("cindex", "equal"),
  allow_same_algorithm = FALSE,
  min_risk_sd = 1e-08,
  rank_by = c("apparent_cindex", "mean_cohort_cindex"),
  stability_groups = NULL,
  n_cores = params$n_cores %||% 1L,
  verbose = params$verbose
)
```

Arguments

X	Feature matrix, or an "automlr_input" object returned by prepare_cohort_input().
y	survival::Surv object. Leave NULL when X is an "automlr_input" object.
params	Output of automlr_parameters().
algorithms	Character vector of registry keys.
top_n	Number of apparent-screen combinations carried into the seed search.
seeds	Integer vector of random seeds used for 70/30 split search.
train_fraction	Fraction of samples assigned to training in the seed search.
min_models	Minimum combination size in the apparent screen.
max_models	Maximum combination size in the apparent screen.
weight_method	One of "cindex" or "equal".

allow_same_algorithm	Logical, allow multiple variants from the same base algorithm in one combination.
min_risk_sd	Minimum apparent risk-score standard deviation required for a candidate to enter the apparent combination screen.
rank_by	"apparent_cindex" ranks by the pooled apparent C-index; "mean_cohort_cindex" ranks by cohort-average apparent C-index when cohort labels are available.
stability_groups	Optional group/cohort labels. Automatically taken from prepare_cohort_input() when X is an "automlr_input".
n_cores	Reserved for future seed-level parallel execution. The seed search currently runs sequentially for reproducible access to package internals.
verbose	Logical.

Value

A list of class "automlr_extreme_screen" with apparent-screen summaries, top combinations, seed-search results, and the best seed/model row by validation C-index. The notes component stores interpretation text for apparent performance and seed-search performance.

fit_binary_ensemble *Fit a binary probability ensemble.*

Description

Fit a binary probability ensemble.

Usage

```
fit_binary_ensemble(
  X,
  y = NULL,
  params = binarymlr_parameters(),
  algorithms = params$algorithms,
  min_auc = params$min_auc_accept,
  auto_min_auc = params$auto_min_auc %||% FALSE,
  auto_quantile = params$auto_quantile %||% 0.5,
  strategy = c("best_subset", "threshold"),
  min_models = 1L,
  max_models = 2L,
  weight_method = c("auc", "equal", "auc_stability"),
  allow_same_algorithm = FALSE,
  max_failed_fraction = 0.2,
  stability_groups = NULL,
  stability_resamples = params$stability_resamples %||% 0L,
  stability_fraction = params$stability_fraction %||% 0.8,
```

```

rank_by = c("auc", "stability_weighted"),
stability_weight = params$stability_weight %||% 0.1,
threshold_method = "youden",
verbose = params$verbose
)

```

Arguments

X	Feature matrix, or an "automlr_binary_input" object.
y	Binary 0/1 outcome. Leave NULL when X is an input object.
params	Output of binarymlr_parameters().
algorithms	Binary algorithm keys.
min_auc	Minimum AUC for threshold strategy.
auto_min_auc	Logical. If TRUE, compute min_auc from the candidate AUC distribution using auto_quantile.
auto_quantile	Quantile used for automatic threshold selection.
strategy	"best_subset" or "threshold".
min_models	Minimum combination size.
max_models	Maximum combination size.
weight_method	One of "auc", "equal", or "auc_stability".
allow_same_algorithm	Logical.
max_failed_fraction	Maximum fold-failure fraction.
stability_groups	Optional cohort labels.
stability_resamples	Stability resamples.
stability_fraction	Stability subsample fraction.
rank_by	Ranking method.
stability_weight	Stability penalty.
threshold_method	Threshold used for final class labels.
verbose	Logical.

Value

An object of class "automlr_binary_ensemble".

```
fit_continuous_ensemble
```

Fit a continuous-outcome prediction ensemble.

Description

Fit a continuous-outcome prediction ensemble.

Usage

```
fit_continuous_ensemble(
  X,
  y = NULL,
  params = continuousmlr_parameters(),
  algorithms = params$algorithms,
  min_r2 = params$min_r2_accept,
  auto_min_r2 = params$auto_min_r2 %||% FALSE,
  auto_quantile = params$auto_quantile %||% 0.5,
  strategy = c("best_subset", "threshold"),
  min_models = 1L,
  max_models = 2L,
  weight_method = c("inverse_rmse", "equal", "r2"),
  allow_same_algorithm = FALSE,
  max_failed_fraction = 0.2,
  rank_by = c("rmse", "r2"),
  verbose = params$verbose
)
```

Arguments

X	Feature matrix, or an "automlr_continuous_input" object.
y	Numeric outcome. Leave NULL when X is an input object.
params	Output of continuousmlr_parameters().
algorithms	Continuous algorithm keys.
min_r2	Minimum R-squared for threshold strategy.
auto_min_r2	Logical. If TRUE, compute min_r2 from the candidate R-squared distribution using auto_quantile.
auto_quantile	Quantile used for automatic threshold selection.
strategy	"best_subset" or "threshold".
min_models	Minimum combination size.
max_models	Maximum combination size.
weight_method	One of "inverse_rmse", "equal", or "r2".
allow_same_algorithm	Logical.

max_failed_fraction	Maximum failed sample fraction.
rank_by	Ranking method.
verbose	Logical.

Value

An object of class "automlr_continuous_ensemble".

fit_ordinal_ensemble *Fit an ordinal-outcome ensemble.*

Description

Fit an ordinal-outcome ensemble.

Usage

```
fit_ordinal_ensemble(
  X,
  y = NULL,
  params = ordinalmlr_parameters(),
  algorithms = params$algorithms,
  min_qwk = params$min_qwk_accept,
  auto_min_qwk = params$auto_min_qwk %||% FALSE,
  auto_quantile = params$auto_quantile %||% 0.5,
  strategy = c("best_subset", "threshold"),
  min_models = 1L,
  max_models = 2L,
  weight_method = c("qwk", "equal", "inverse_mae"),
  allow_same_algorithm = FALSE,
  rank_by = c("qwk", "class_mae"),
  verbose = params$verbose
)
```

Arguments

X	Feature matrix, or an "automlr_ordinal_input" object.
y	Ordinal integer outcome codes. Leave NULL when X is an input object.
params	Output of ordinalmlr_parameters().
algorithms	Ordinal algorithm keys.
min_qwk	Minimum quadratic weighted kappa for threshold strategy.
auto_min_qwk	Logical. If TRUE, compute min_qwk from the candidate kappa distribution using auto_quantile.
auto_quantile	Quantile used for automatic threshold selection.

strategy	"best_subset" or "threshold".
min_models	Minimum combination size.
max_models	Maximum combination size.
weight_method	One of "qwk", "equal", or "inverse_mae".
allow_same_algorithm	Logical.
rank_by	Ranking method.
verbose	Logical.

Value

An object of class "automlr_ordinal_ensemble".

fit_surv_ensemble	<i>Fit a weighted ensemble of survival-risk models.</i>
-------------------	---

Description

The ensemble first estimates each candidate model's LOOCV C-index. With the default "best_subset" strategy, it enumerates model subsets up to max_models variants and chooses the subset with the highest combined LOOCV C-index. With "threshold", it keeps all single models meeting min_cindex (or the best finite model if none meet it).

Usage

```
fit_surv_ensemble(
  X,
  y = NULL,
  params = automlr_parameters(),
  algorithms = params$algorithms,
  min_cindex = params$min_cindex_accept,
  auto_min_cindex = params$auto_min_cindex %||% FALSE,
  auto_quantile = params$auto_quantile %||% 0.5,
  strategy = c("best_subset", "threshold"),
  min_models = 1L,
  max_models = 2L,
  weight_method = c("cindex", "equal", "cindex_stability"),
  allow_same_algorithm = FALSE,
  max_failed_fraction = 0.2,
  stability_groups = NULL,
  stability_resamples = params$stability_resamples %||% 0L,
  stability_fraction = params$stability_fraction %||% 0.8,
  rank_by = c("cindex", "stability_weighted"),
  stability_weight = params$stability_weight %||% 0.1,
  diagnostic_times = NULL,
  verbose = params$verbose
)
```

Arguments

<code>X</code>	Feature matrix, or an "automlr_input" object returned by <code>prepare_cohort_input()</code> .
<code>y</code>	<code>survival::Surv</code> object. Leave NULL when <code>X</code> is an "automlr_input" object.
<code>params</code>	Output of <code>automlr_parameters()</code> .
<code>algorithms</code>	Character vector of registry keys.
<code>min_cindex</code>	Minimum C-index for automatic inclusion.
<code>auto_min_cindex</code>	Logical. If TRUE, compute <code>min_cindex</code> from the candidate LOOCV C-index distribution using <code>auto_quantile</code> .
<code>auto_quantile</code>	Quantile used for automatic threshold selection.
<code>strategy</code>	One of "best_subset" or "threshold".
<code>min_models</code>	Minimum subset size for "best_subset".
<code>max_models</code>	Maximum subset size for "best_subset".
<code>weight_method</code>	One of "cindex", "equal", or "cindex_stability".
<code>allow_same_algorithm</code>	Logical, allow multiple variants from the same base algorithm in one selected combination.
<code>max_failed_fraction</code>	Maximum allowed LOOCV fold failure fraction for candidates entering combination search.
<code>stability_groups</code>	Optional group/queue labels used for stability diagnostics.
<code>stability_resamples</code>	Number of repeated subsamples for stability diagnostics.
<code>stability_fraction</code>	Fraction of samples in each stability subsample.
<code>rank_by</code>	Ranking method passed to <code>evaluate_surv_combinations()</code> .
<code>stability_weight</code>	Non-negative stability penalty multiplier used when <code>rank_by = "stability_weighted"</code> .
<code>diagnostic_times</code>	Optional time points for time-dependent AUC diagnostics in the combination table.
<code>verbose</code>	Logical.

Value

A list of class "automlr_surv_ensemble".

get_binary_registry *Return the binary-classification algorithm registry.*

Description

Return the binary-classification algorithm registry.

Usage

```
get_binary_registry()
```

Value

A named list of algorithm specs.

get_continuous_registry
 Return the continuous-outcome algorithm registry.

Description

Return the continuous-outcome algorithm registry.

Usage

```
get_continuous_registry()
```

Value

A named list of algorithm specs.

get_ordinal_registry *Return the ordinal-outcome algorithm registry.*

Description

Return the ordinal-outcome algorithm registry.

Usage

```
get_ordinal_registry()
```

Value

A named list of algorithm specs.

get_surv_registry	<i>Return the full survival-algorithm registry.</i>
-------------------	---

Description

Return the full survival-algorithm registry.

Usage

```
get_surv_registry()
```

Value

A named list of algorithm specs.

`initialize_auto_logging`

Enable file + console logging for the current R session.

Description

Writes a timestamped log file to `log_dir` (default: `file.path(getwd(), "automlr_logs")`) and keeps at most `max_log_files`.

Usage

```
initialize_auto_logging(log_dir = NULL, max_log_files = 10)
```

Arguments

<code>log_dir</code>	Directory to write logs into.
<code>max_log_files</code>	Retain only the N most recent logs.

Value

Invisibly returns the log4r logger when available, otherwise NULL.

```
list_binary_algorithms
```

List supported binary-classification algorithms.

Description

List supported binary-classification algorithms.

Usage

```
list_binary_algorithms()
```

Value

Character vector of algorithm keys.

```
list_binary_model_variants
```

List binary-classification model variants.

Description

List binary-classification model variants.

Usage

```
list_binary_model_variants(  
  params = binarymlr_parameters(),  
  algorithms = params$algorithms  
)
```

Arguments

params	Output of <code>binarymlr_parameters()</code> .
algorithms	Binary algorithm keys.

Value

A data.frame of concrete model variants.

`list_continuous_algorithms`*List supported continuous-outcome algorithms.*

Description

List supported continuous-outcome algorithms.

Usage

```
list_continuous_algorithms()
```

Value

Character vector of algorithm keys.

`list_continuous_model_variants`*List continuous-outcome model variants.*

Description

List continuous-outcome model variants.

Usage

```
list_continuous_model_variants(  
  params = continuousmlr_parameters(),  
  algorithms = params$algorithms  
)
```

Arguments

`params` Output of `continuousmlr_parameters()`.

`algorithms` Continuous algorithm keys.

Value

A data.frame of concrete model variants.

list_model_variants *List concrete model variants generated from algorithm grids.*

Description

Each row is one candidate model that can enter a model combination. Multiple rows can come from the same base algorithm when its registry grid contains multiple hyperparameter settings.

Usage

```
list_model_variants(
  params = automlr_parameters(),
  algorithms = params$algorithms
)
```

Arguments

params Output of automlr_parameters().
 algorithms Character vector of registry keys.

Value

A data.frame of candidate model variants.

list_ordinal_algorithms
 List supported ordinal-outcome algorithms.

Description

List supported ordinal-outcome algorithms.

Usage

```
list_ordinal_algorithms()
```

Value

Character vector of algorithm keys.

```
list_ordinal_model_variants
```

List ordinal-outcome model variants.

Description

List ordinal-outcome model variants.

Usage

```
list_ordinal_model_variants(  
  params = ordinalmlr_parameters(),  
  algorithms = params$algorithms  
)
```

Arguments

params Output of ordinalmlr_parameters().
algorithms Ordinal algorithm keys.

Value

A data.frame of concrete model variants.

```
list_surv_algorithms
```

List the supported survival algorithms (keys).

Description

List the supported survival algorithms (keys).

Usage

```
list_surv_algorithms()
```

Value

Character vector of algorithm keys.

loocv_auc	<i>Leave-one-out cross-validation AUC for one binary algorithm.</i>
-----------	---

Description

Leave-one-out cross-validation AUC for one binary algorithm.

Usage

```
loocv_auc(  
  X,  
  y,  
  fit_fn,  
  predict_fn,  
  hparam = list(),  
  seed = NULL,  
  verbose = FALSE,  
  n_cores = 1L  
)
```

Arguments

X	Numeric feature matrix.
y	Binary 0/1 outcome.
fit_fn	Function (X_train, y_train, hparam) -> model.
predict_fn	Function (model, X_test) -> positive-class probability.
hparam	Hyperparameter list.
seed	Optional seed.
verbose	Logical.
n_cores	Integer number of fold workers.

Value

A list with AUC, PR-AUC, Brier score, probabilities, and failures.

loocv_index	<i>Leave-one-out cross-validation C-index for one survival algorithm.</i>
-------------	---

Description

Loops over every sample, fits the model on the remaining $n-1$, predicts the held-out one, then computes Harrell's C-index on the full vector of held-out risk scores. Any fold that errors records NA for that sample and the error message; the C-index is computed on the remaining folds.

Usage

```
loocv_index(
  X,
  y,
  fit_fn,
  predict_fn,
  hparam = list(),
  seed = NULL,
  verbose = FALSE,
  n_cores = 1L
)
```

Arguments

X	Numeric matrix or data.frame of features ($n \times p$).
y	A <code>survival::Surv</code> object of length n .
fit_fn	Function ($X_{train}, y_{train}, hparam$) \rightarrow model.
predict_fn	Function (model, X_{test}) \rightarrow numeric risk score (one per row of X_{test}). Higher score = higher hazard.
hparam	Named list of hyperparameters passed to <code>fit_fn</code> . Use <code>list()</code> for algorithms that tune internally.
seed	Integer seed set once before the loop for reproducibility; NULL leaves the RNG untouched.
verbose	Logical, print a progress message every $\sim 10\%$.
n_cores	Integer, number of workers for fold-level parallelism.

Value

A list with components:

- cindex** Scalar Harrell's C-index on the aggregated predictions.
- risk** Numeric vector of length n with per-sample LOOCV risk scores (NA where the fold errored).
- n_folds** n .
- n_failed** Number of folds that errored.
- errors** Character vector of error messages (may be length 0).
- elapsed_sec** Wall-clock seconds.

ordinal_accuracy	<i>Accuracy for ordinal class predictions.</i>
------------------	--

Description

Accuracy for ordinal class predictions.

Usage

```
ordinal_accuracy(y, pred_class)
```

Arguments

y	Observed positive integer level codes.
pred_class	Predicted positive integer level codes.

Value

A numeric scalar.

ordinal_balanced_accuracy	<i>Balanced accuracy for ordinal class predictions.</i>
---------------------------	---

Description

Balanced accuracy for ordinal class predictions.

Usage

```
ordinal_balanced_accuracy(y, pred_class)
```

Arguments

y	Observed positive integer level codes.
pred_class	Predicted positive integer level codes.

Value

A numeric scalar.

ordinal_mae	<i>Mean absolute class error for ordinal predictions.</i>
-------------	---

Description

Mean absolute class error for ordinal predictions.

Usage

```
ordinal_mae(y, pred_class)
```

Arguments

y	Observed positive integer level codes.
pred_class	Predicted positive integer level codes.

Value

A numeric scalar.

ordinal_qwk	<i>Quadratic weighted kappa for ordinal predictions.</i>
-------------	--

Description

Quadratic weighted kappa for ordinal predictions.

Usage

```
ordinal_qwk(y, pred_class, n_levels = max(y, pred_class, na.rm = TRUE))
```

Arguments

y	Observed positive integer level codes.
pred_class	Predicted positive integer level codes.
n_levels	Number of ordered outcome levels.

Value

A numeric scalar.

ordinalmlr_parameters *Default parameters for AutoMLR ordinal-outcome workflows.*

Description

Default parameters for AutoMLR ordinal-outcome workflows.

Usage

```
ordinalmlr_parameters(
  seed = 123L,
  algorithms = NULL,
  resampling = "loocv",
  k_folds = 5L,
  repeats = 1L,
  min_qwk_accept = 0,
  auto_min_qwk = FALSE,
  auto_quantile = 0.5,
  missing_fraction_cutoff = 0.2,
  screen_by_variance = TRUE,
  variance_quantile_cutoff = 0,
  standardize_features = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

seed	Base random seed.
algorithms	Character vector of ordinal algorithm keys.
resampling	Resampling scheme.
k_folds	Number of folds for k-fold CV.
repeats	Number of repeats for repeated k-fold CV.
min_qwk_accept	Minimum quadratic weighted kappa for threshold strategy.
auto_min_qwk	Logical. If TRUE and <code>fit_ordinal_ensemble()</code> uses <code>strategy = "threshold"</code> , replace <code>min_qwk_accept</code> with a recommended cutoff computed from candidate-model kappa values.
auto_quantile	Quantile used when an automatic threshold is requested. <code>0.50</code> uses the median candidate metric; larger values are more selective.
missing_fraction_cutoff	Drop features above this missing fraction.
screen_by_variance	Logical, drop zero / low-variance features.

variance_quantile_cutoff Optional lower variance quantile to drop.
standardize_features Logical, center and scale features.
n_cores Integer, number of fold workers.
verbose Logical, print progress.

Value

A named list.

parallel_lapply *Parallel lapply that transparently falls back to sequential.*

Description

Parallel lapply that transparently falls back to sequential.

Usage

`parallel_lapply(X, FUN, ..., cores = NULL)`

Arguments

X Input list / vector.
FUN Function.
... Extra args to FUN.
cores Optionally override the global core count.

Value

List of results.

predict.automlr_binary_ensemble

Predict binary ensemble probabilities or classes.

Description

Predict binary ensemble probabilities or classes.

Usage

```
## S3 method for class 'automlr_binary_ensemble'
predict(
  object,
  newX,
  type = c("prob", "class"),
  threshold = object$threshold,
  ...
)
```

Arguments

object	Object returned by fit_binary_ensemble().
newX	Feature matrix.
type	"prob" for positive-class probability or "class" for 0/1.
threshold	Optional class threshold.
...	Ignored.

Value

Numeric vector.

predict.automlr_continuous_ensemble

Predict continuous ensemble values.

Description

Predict continuous ensemble values.

Usage

```
## S3 method for class 'automlr_continuous_ensemble'
predict(object, newX, ...)
```

Arguments

object	Object returned by fit_continuous_ensemble().
newX	Feature matrix.
...	Ignored.

Value

Numeric vector.

predict.automlr_ordinal_ensemble
Predict ordinal ensemble scores or classes.

Description

Predict ordinal ensemble scores or classes.

Usage

```
## S3 method for class 'automlr_ordinal_ensemble'  
predict(object, newX, type = c("score", "code", "class"), ...)
```

Arguments

object	Object returned by fit_ordinal_ensemble().
newX	Feature matrix.
type	"score", "code", or "class".
...	Ignored.

Value

Numeric scores/codes or class labels.

```
predict.automlr_surv_ensemble
```

Predict weighted ensemble risk.

Description

Predict weighted ensemble risk.

Usage

```
## S3 method for class 'automlr_surv_ensemble'  
predict(object, newX, ...)
```

Arguments

object	An object returned by <code>fit_surv_ensemble()</code> .
newX	Feature matrix.
...	Ignored.

Value

Numeric risk score; higher means higher predicted hazard.

```
prepare_binary_cohort_input
```

Prepare multi-cohort binary-classification data.

Description

Splits a long-format data frame by cohort, maps the outcome to 0/1, keeps numeric shared features across cohorts, and returns an object for binary AutoMLR workflows.

Usage

```
prepare_binary_cohort_input(  
  data,  
  cohort,  
  outcome,  
  id = NULL,  
  positive_class = 1,  
  negative_class = NULL,  
  collapse_other = FALSE,  
  drop_cohorts = NULL  
)
```

Arguments

data	A data.frame with one row per sample.
cohort	Name of the cohort / dataset column.
outcome	Name of the binary outcome column.
id	Optional sample-id column.
positive_class	Value in outcome that should be treated as the positive class. Defaults to 1.
negative_class	Optional value in outcome that should be treated as the negative class. When NULL, the remaining level is inferred only if the outcome has exactly two non-missing levels.
collapse_other	Logical. If TRUE, all classes other than positive_class are treated as negative. Defaults to FALSE to avoid accidental multi-class collapse.
drop_cohorts	Optional cohorts to exclude.

Value

An object of class "automlr_binary_input".

prepare_cohort_input *Prepare multi-cohort survival data from a single long-format table.*

Description

Splits data by the cohort column and computes the **feature intersection** across cohorts. Returns a tidy object for downstream fitting plus diagnostic info for the user.

Usage

```
prepare_cohort_input(
  data,
  cohort,
  time,
  status,
  id = NULL,
  drop_cohorts = NULL
)
```

Arguments

data	A data.frame with one row per sample.
cohort	Name of the column identifying cohort membership.
time	Name of the survival time column (numeric, > 0).
status	Name of the event indicator column (0/1; 1 = event).
id	Optional name of a sample-id column; just passed through.
drop_cohorts	Optional character vector of cohorts to exclude.

Value

An S3 list of class "automlr_input" with components:

cohorts Named list of per-cohort data frames restricted to shared_features + time + status (+ id).

shared_features Character vector of feature columns present in every cohort (the intersection).

per_cohort_features Named list of each cohort's raw feature set.

dropped_features Features present in at least one cohort but not all, therefore excluded from the intersection.

summary data.frame: cohort, n_samples, n_events, median_time, n_raw_features, n_shared_features.

meta Echo of column names used.

```
prepare_continuous_cohort_input
```

Prepare multi-cohort continuous-outcome data.

Description

Splits a long-format data frame by cohort, keeps numeric shared features, and returns an object for continuous AutoMLR workflows.

Usage

```
prepare_continuous_cohort_input(
  data,
  cohort,
  outcome,
  id = NULL,
  drop_cohorts = NULL
)
```

Arguments

data	A data.frame with one row per sample.
cohort	Name of the cohort / dataset column.
outcome	Name of the numeric outcome column.
id	Optional sample-id column.
drop_cohorts	Optional cohorts to exclude.

Value

An object of class "automlr_continuous_input".

```
prepare_ordinal_cohort_input
```

Prepare multi-cohort ordinal-outcome data.

Description

Maps an ordered outcome to integer scores, keeps numeric shared features, and returns an object for ordinal AutoMLR workflows.

Usage

```
prepare_ordinal_cohort_input(  
  data,  
  cohort,  
  outcome,  
  ordered_levels = NULL,  
  id = NULL,  
  drop_cohorts = NULL  
)
```

Arguments

data	A data.frame with one row per sample.
cohort	Name of the cohort / dataset column.
outcome	Name of the ordinal outcome column.
ordered_levels	Optional ordered outcome levels from low to high.
id	Optional sample-id column.
drop_cohorts	Optional cohorts to exclude.

Value

An object of class "automlr_ordinal_input".

```
print.automlr_dependency_report
```

Print an AutoMLR dependency report.

Description

Print an AutoMLR dependency report.

Usage

```
## S3 method for class 'automlr_dependency_report'  
print(x, ...)
```

Arguments

x An object returned by check_automlr_dependencies().
 ... Unused.

Value

Invisibly returns x.

```
print.automlr_extreme_screen
```

Print method for extreme survival screening

Description

Print method for extreme survival screening

Usage

```
## S3 method for class 'automlr_extreme_screen'
print(x, ...)
```

Arguments

x An object returned by extreme_surv_screen().
 ... Ignored.

Value

Invisibly returns x.

```
recommend_binary_auc_threshold
```

Recommend a binary AUC cutoff from candidate model results.

Description

Uses the finite receiver operating characteristic area under the curve (AUC) values in a binary candidate summary and returns the requested quantile as a threshold for "threshold" strategy model selection.

Usage

```
recommend_binary_auc_threshold(loocv_set, auto_quantile = 0.5, minimum = 0.5)
```

Arguments

loocv_set	An object returned by <code>evaluate_binary_algorithms_loocv()</code> , or a data frame containing a numeric auc column.
auto_quantile	Numeric quantile in $[0, 1]$. Larger values are more selective; <code>0.50</code> uses the median candidate AUC.
minimum	Lower bound for the returned threshold. Defaults to <code>0.5</code> .

Value

A numeric scalar giving the recommended minimum AUC.

```
recommend_continuous_r2_threshold
```

Recommend a continuous R-squared cutoff from candidate model results.

Description

Uses finite out-of-fold R-squared values and returns the requested quantile as a threshold for "threshold" strategy model selection.

Usage

```
recommend_continuous_r2_threshold(
  resample_set,
  auto_quantile = 0.5,
  minimum = 0
)
```

Arguments

resample_set	An object returned by <code>evaluate_continuous_algorithms()</code> , or a data frame containing a numeric r2 column.
auto_quantile	Numeric quantile in $[0, 1]$. Larger values are more selective; <code>0.50</code> uses the median candidate R-squared.
minimum	Lower bound for the returned threshold. Defaults to <code>0</code> .

Value

A numeric scalar giving the recommended minimum R-squared.

recommend_ordinal_qwk_threshold

Recommend an ordinal kappa cutoff from candidate model results.

Description

Uses finite out-of-fold quadratic weighted kappa values and returns the requested quantile as a threshold for "threshold" strategy model selection.

Usage

```
recommend_ordinal_qwk_threshold(resample_set, auto_quantile = 0.5, minimum = 0)
```

Arguments

resample_set	An object returned by evaluate_ordinal_algorithms(), or a data frame containing a numeric qwk column.
auto_quantile	Numeric quantile in $[0, 1]$. Larger values are more selective; 0.50 uses the median candidate kappa.
minimum	Lower bound for the returned threshold. Defaults to 0.

Value

A numeric scalar giving the recommended minimum quadratic weighted kappa.

recommend_surv_cindex_threshold

Recommend a survival C-index cutoff from candidate model results.

Description

Uses the finite leave-one-out cross-validation concordance index (C-index) values in a survival candidate summary and returns the requested quantile as a threshold for "threshold" strategy model selection.

Usage

```
recommend_surv_cindex_threshold(loocv_set, auto_quantile = 0.5, minimum = 0.5)
```

Arguments

loocv_set	An object returned by evaluate_algorithms_loocv(), or a data frame containing a numeric cindex column.
auto_quantile	Numeric quantile in $[0, 1]$. Larger values are more selective; 0.50 uses the median candidate C-index.
minimum	Lower bound for the returned threshold. Defaults to 0.5 so models below random concordance are not automatically accepted.

Value

A numeric scalar giving the recommended minimum C-index.

render_binary_report *Render an HTML report for a fitted binary ensemble.*

Description

Render an HTML report for a fitted binary ensemble.

Usage

```
render_binary_report(  
  object,  
  output_dir = "automlr_binary_report",  
  report_file = "index.html",  
  title = "AutoMLR Binary Report",  
  top_n = 20L,  
  overwrite = TRUE,  
  summary_language = c("bilingual", "en", "zh")  
)
```

Arguments

object	Object returned by fit_binary_ensemble().
output_dir	Report directory.
report_file	HTML file name.
title	Report title.
top_n	Number of top rows.
overwrite	Logical.
summary_language	"bilingual", "en", or "zh".

Value

Invisibly returns report path.

`render_continuous_report`*Render an HTML report for a fitted continuous ensemble.*

Description

Render an HTML report for a fitted continuous ensemble.

Usage

```
render_continuous_report(  
  object,  
  output_dir = "automlr_continuous_report",  
  report_file = "index.html",  
  title = "AutoMLR Continuous Report",  
  top_n = 20L,  
  overwrite = TRUE  
)
```

Arguments

<code>object</code>	Object returned by <code>fit_continuous_ensemble()</code> .
<code>output_dir</code>	Report directory.
<code>report_file</code>	HTML file name.
<code>title</code>	Report title.
<code>top_n</code>	Number of top rows.
<code>overwrite</code>	Logical.

Value

Invisibly returns report path.

`render_ordinal_report` *Render an HTML report for a fitted ordinal ensemble.*

Description

Render an HTML report for a fitted ordinal ensemble.

Usage

```
render_ordinal_report(  
  object,  
  output_dir = "automlr_ordinal_report",  
  report_file = "index.html",  
  title = "AutoMLR Ordinal Report",  
  top_n = 20L,  
  overwrite = TRUE  
)
```

Arguments

object	Object returned by <code>fit_ordinal_ensemble()</code> .
output_dir	Report directory.
report_file	HTML file name.
title	Report title.
top_n	Number of top rows.
overwrite	Logical.

Value

Invisibly returns report path.

render_surv_report	<i>Render an HTML report for a fitted survival ensemble.</i>
--------------------	--

Description

Writes a self-contained HTML summary plus separate figures/ and tables/ folders. Model selection remains whatever was used by `fit_surv_ensemble()`; this function only reports diagnostics.

Usage

```
render_surv_report(  
  object,  
  output_dir = "automlr_report",  
  report_file = "index.html",  
  title = "AutoMLR Survival Report",  
  top_n = 20L,  
  overwrite = TRUE,  
  summary_language = c("bilingual", "en", "zh")  
)
```

Arguments

object	An object returned by <code>fit_surv_ensemble()</code> .
output_dir	Directory where the report folder should be written.
report_file	HTML file name.
title	Report title.
top_n	Number of top single models / combinations to show.
overwrite	Logical, overwrite existing report files.
summary_language	Language used in <code>summary_report.md</code> , either "bilingual", "en", or "zh".

Value

Invisibly returns the HTML report path.

report_binary_cohort_intersection

Print a binary cohort-intersection report.

Description

Print a binary cohort-intersection report.

Usage

```
report_binary_cohort_intersection(input)
```

Arguments

input	An object returned by <code>prepare_binary_cohort_input()</code> .
-------	--

Value

Invisibly returns input.

`report_cohort_intersection`*Print a human-readable report of the cohort intersection.*

Description

Tells the user:

- per-cohort sample/event counts and median follow-up,
- how many features each cohort has vs. the intersection,
- how many features were dropped because they were missing in some cohorts.

Usage`report_cohort_intersection(input)`**Arguments**

`input` An object returned by `prepare_cohort_input()`.

Value

Invisibly returns `input`.

`report_continuous_cohort_intersection`*Print a continuous cohort-intersection report.*

Description

Print a continuous cohort-intersection report.

Usage`report_continuous_cohort_intersection(input)`**Arguments**

`input` An object returned by `prepare_continuous_cohort_input()`.

Value

Invisibly returns `input`.

```
report_ordinal_cohort_intersection
    Print an ordinal cohort-intersection report.
```

Description

Print an ordinal cohort-intersection report.

Usage

```
report_ordinal_cohort_intersection(input)
```

Arguments

input An object returned by `prepare_ordinal_cohort_input()`.

Value

Invisibly returns input.

```
start_parallel            Start the parallel backend.
```

Description

Thin wrapper around `future::plan()` — see `?future::plan`. If the optional future package is unavailable, AutoMLR keeps using sequential execution.

Usage

```
start_parallel(
  cores = get_parallel_cores(),
  strategy = c("multisession", "multicore")
)
```

Arguments

cores Integer, number of workers.

strategy One of "multisession" (default, cross-platform) or "multicore" (Linux/macOS only).

Value

Invisibly TRUE when the backend was started, or FALSE when the optional future package is unavailable.

stop_parallel	<i>Stop the parallel backend.</i>
---------------	-----------------------------------

Description

Stop the parallel backend.

Usage

```
stop_parallel()
```

Value

Invisibly TRUE when a future backend was reset, or FALSE when the optional future package is unavailable.

summarize_base_models	<i>Summarize base-model screening results in Markdown</i>
-----------------------	---

Description

Summarize base-model screening results in Markdown

Usage

```
summarize_base_models(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

object	An object returned by <code>fit_surv_ensemble()</code> .
top_n	Number of top model rows to show.
language	Summary language: "bilingual", "en", or "zh".

Value

A Markdown string.

`summarize_binary_analysis_results`*Summarize a complete binary AutoMLR analysis in Markdown.*

Description

Summarize a complete binary AutoMLR analysis in Markdown.

Usage

```
summarize_binary_analysis_results(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

<code>object</code>	A fitted binary ensemble.
<code>top_n</code>	Number of rows per section.
<code>language</code>	"bilingual", "en", or "zh".

Value

Markdown string.

`summarize_binary_base_models`*Summarize binary base-model screening in Markdown.*

Description

Summarize binary base-model screening in Markdown.

Usage

```
summarize_binary_base_models(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

object	A fitted binary ensemble.
top_n	Number of rows.
language	"bilingual", "en", or "zh".

Value

Markdown string.

summarize_binary_data_preparation
Summarize binary data preparation in Markdown.

Description

Summarize binary data preparation in Markdown.

Usage

```
summarize_binary_data_preparation(  
  x,  
  top_n = 10L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

x	An automlr_binary_input or fitted binary ensemble.
top_n	Number of rows.
language	"bilingual", "en", or "zh".

Value

Markdown string.

`summarize_binary_ensemble_results`*Summarize binary ensemble selection in Markdown.*

Description

Summarize binary ensemble selection in Markdown.

Usage

```
summarize_binary_ensemble_results(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

<code>object</code>	A fitted binary ensemble.
<code>top_n</code>	Number of rows.
<code>language</code>	"bilingual", "en", or "zh".

Value

Markdown string.

`summarize_binary_explainability_results`*Summarize binary explainability outputs in Markdown.*

Description

Summarize binary explainability outputs in Markdown.

Usage

```
summarize_binary_explainability_results(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

object	A fitted binary ensemble.
top_n	Number of rows.
language	"bilingual", "en", or "zh".

Value

Markdown string.

summarize_data_preparation

Summarize data-preparation results in Markdown

Description

Creates a bilingual or single-language summary of cohort/sample/event counts, shared features, dropped non-shared features, and simple data-risk notes.

Usage

```
summarize_data_preparation(  
  x,  
  top_n = 10L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

x	An automlr_input object or an object returned by fit_surv_ensemble().
top_n	Number of cohort rows to show.
language	Summary language: "bilingual", "en", or "zh".

Value

A Markdown string.

`summarize_ensemble_results`*Summarize ensemble-selection results in Markdown*

Description

Summarize ensemble-selection results in Markdown

Usage

```
summarize_ensemble_results(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

<code>object</code>	An object returned by <code>fit_surv_ensemble()</code> .
<code>top_n</code>	Number of top combination rows to show.
<code>language</code>	Summary language: "bilingual", "en", or "zh".

Value

A Markdown string.

`summarize_explainability_results`*Summarize explainability and clinical-utility outputs in Markdown*

Description

Summarize explainability and clinical-utility outputs in Markdown

Usage

```
summarize_explainability_results(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

object	An object returned by <code>fit_surv_ensemble()</code> .
top_n	Number of top feature/model rows to show.
language	Summary language: "bilingual", "en", or "zh".

Value

A Markdown string.

summarize_extreme_screen_results
Summarize extreme-screening results in readable Markdown

Description

Creates a compact interpretation of an `extreme_surv_screen()` result: apparent-screen leaders, best seed-search model, top seed rows, best rows after seed de-duplication, combination-level stability, and failure notes.

Usage

```
summarize_extreme_screen_results(  
  x,  
  top_n = 3L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

x	An object returned by <code>extreme_surv_screen()</code> .
top_n	Number of rows to include in each top-results section.
language	Summary language, either "bilingual", "zh", or "en".

Value

A single Markdown string.

`summarize_surv_analysis_results`*Summarize a complete regular survival AutoML analysis in Markdown*

Description

Combines data-preparation, base-model, and ensemble-selection summaries. This is the regular-analysis counterpart to `summarize_extreme_screen_results()`.

Usage

```
summarize_surv_analysis_results(  
  object,  
  top_n = 5L,  
  language = c("bilingual", "en", "zh")  
)
```

Arguments

<code>object</code>	An object returned by <code>fit_surv_ensemble()</code> .
<code>top_n</code>	Number of top rows to show per section.
<code>language</code>	Summary language: "bilingual", "en", or "zh".

Value

A Markdown string.

Index

automlr_input_to_binary_xy, 4
automlr_input_to_continuous_xy, 5
automlr_input_to_ordinal_xy, 5
automlr_input_to_surv_xy, 6
automlr_parameters, 6

binary_auc, 8
binary_pr_auc, 9
binarymlr_parameters, 9

check_automlr_dependencies, 11
continuous_cor, 11
continuous_mae, 12
continuous_r2, 12
continuous_rmse, 13
continuousmlr_parameters, 13
count_binary_combinations, 15
count_continuous_combinations, 15
count_ordinal_combinations, 16
count_surv_combinations, 17

disable_auto_logging, 18

evaluate_algorithm_loocv, 18
evaluate_algorithms_loocv, 19
evaluate_binary_algorithm_loocv, 20
evaluate_binary_algorithms_loocv, 21
evaluate_binary_combinations, 22
evaluate_continuous_algorithm, 23
evaluate_continuous_algorithms, 24
evaluate_continuous_combinations, 24
evaluate_ordinal_algorithms, 25
evaluate_ordinal_combinations, 26
evaluate_surv_combinations, 27
export_binary_results, 28
export_continuous_results, 29
export_extreme_screen_results, 30
export_ordinal_results, 31
export_surv_results, 32
extreme_surv_screen, 33

fit_binary_ensemble, 34
fit_continuous_ensemble, 36
fit_ordinal_ensemble, 37
fit_surv_ensemble, 38

get_binary_registry, 40
get_continuous_registry, 40
get_ordinal_registry, 40
get_surv_registry, 41

initialize_auto_logging, 41

list_binary_algorithms, 42
list_binary_model_variants, 42
list_continuous_algorithms, 43
list_continuous_model_variants, 43
list_model_variants, 44
list_ordinal_algorithms, 44
list_ordinal_model_variants, 45
list_surv_algorithms, 45
loocv_auc, 46
loocv_cindex, 47

ordinal_accuracy, 48
ordinal_balanced_accuracy, 48
ordinal_mae, 49
ordinal_qwk, 49
ordinalmlr_parameters, 50

parallel_lapply, 51
predict.automlr_binary_ensemble, 52
predict.automlr_continuous_ensemble, 52
predict.automlr_ordinal_ensemble, 53
predict.automlr_surv_ensemble, 54
prepare_binary_cohort_input, 54
prepare_cohort_input, 55
prepare_continuous_cohort_input, 56
prepare_ordinal_cohort_input, 57
print.automlr_dependency_report, 57
print.automlr_extreme_screen, 58

recommend_binary_auc_threshold, 58
recommend_continuous_r2_threshold, 59
recommend_ordinal_qwk_threshold, 60
recommend_surv_cindex_threshold, 60
render_binary_report, 61
render_continuous_report, 62
render_ordinal_report, 62
render_surv_report, 63
report_binary_cohort_intersection, 64
report_cohort_intersection, 65
report_continuous_cohort_intersection,
65
report_ordinal_cohort_intersection, 66

start_parallel, 66
stop_parallel, 67
summarize_base_models, 67
summarize_binary_analysis_results, 68
summarize_binary_base_models, 68
summarize_binary_data_preparation, 69
summarize_binary_ensemble_results, 70
summarize_binary_explainability_results,
70
summarize_data_preparation, 71
summarize_ensemble_results, 72
summarize_explainability_results, 72
summarize_extreme_screen_results, 73
summarize_surv_analysis_results, 74