Package 'fastml'

June 11, 2025

Type Package

Title Fast Machine Learning Model Training and Evaluation

Version 0.6.1

Description Streamlines the training, evaluation, and comparison of multiple machine learning models with minimal code by providing
comprehensive data preprocessing and support for a wide range of algorithms with hyperparameter tuning.
It offers performance metrics and visualization tools to facilitate efficient and effective machine learning workflows.

Encoding UTF-8

License MIT + file LICENSE

URL https://github.com/selcukorkmaz/fastml

BugReports https://github.com/selcukorkmaz/fastml/issues

LazyData true

Imports recipes, dplyr, ggplot2, reshape2, rsample, parsnip, tune, workflows, yardstick, tibble, rlang, dials, RColorBrewer, baguette, bonsai, discrim, doFuture, finetune, future, plsmod, probably, viridisLite, DALEX, magrittr, patchwork, pROC, janitor, stringr, DT, GGally, UpSetR, VIM, broom, dbscan, ggpubr, gridExtra, htmlwidgets, kableExtra, moments, naniar, plotly, scales, skimr, tidyr, knitr, rmarkdown, purrr, mice, missForest

Suggests testthat (>= 3.0.0), C50, glmnet, xgboost, ranger, crayon, kernlab, klaR, kknn, keras, lightgbm, rstanarm, mixOmics,

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Selcuk Korkmaz [aut, cre] (ORCID: <https://orcid.org/0000-0003-4632-6850>), Dincer Goksuluk [aut] (ORCID: <https://orcid.org/0000-0002-2752-7668>), Eda Karaismailoglu [aut] (ORCID: <https://orcid.org/0000-0003-3085-7809>)

Maintainer Selcuk Korkmaz <selcukorkmaz@gmail.com>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2025-06-10 22:50:02 UTC

Contents

`availableMethods` *Get Available Methods*

Description

Returns a character vector of algorithm names available for either classification or regression tasks.

Usage

```
availableMethods(type = c("classification", "regression"), ...)
```

Arguments

type	A character string specifying the type of task. Must be either "classification" or "regression". Defaults to c("classification", "regression") and uses match.arg to select one.
...	Additional arguments (currently not used).

Details

Depending on the specified type, the function returns a different set of algorithm names:

- For "classification", it returns algorithms such as "logistic_reg", "multinom_reg", "decision_tree", "C5_rules", "rand_forest", "xgboost", "lightgbm", "svm_linear", "svm_rbf", "nearest_neighbor", "naive_Bayes", "mlp", "discrim_linear", "discrim_quad", and "bag_tree".
- For "regression", it returns algorithms such as "linear_reg", "ridge_regression", "lasso_regression", "elastic_net", "decision_tree", "rand_forest", "xgboost", "lightgbm", "svm_linear", "svm_rbf", "nearest_neighbor", "mlp", "pls", and "bayes_glm".

Value

A character vector containing the names of the available algorithms for the specified task type.

---

evaluate_models *Evaluate Models Function*

Description

Evaluates the trained models on the test data and computes performance metrics.

Usage

```
evaluate_models(
  models,
  train_data,
  test_data,
  label,
  task,
  metric = NULL,
  event_class
)
```

Arguments

models	A list of trained model objects.
train_data	Preprocessed training data frame.
test_data	Preprocessed test data frame.
label	Name of the target variable.
task	Type of task: "classification" or "regression".
metric	The performance metric to optimize (e.g., "accuracy", "rmse").
event_class	A single string. Either "first" or "second" to specify which level of truth to consider as the "event".

Value

A list with two elements:

performance A named list of performance metric tibbles for each model.

predictions A named list of data frames with columns including truth, predictions, and probabilities per model.

---

fastexplain *FastExplain the fastml (DALEX + SHAP + Permutation-based VI)*

Description

Provides model explainability using DALEX. This function:

- Creates a DALEX explainer.
- Computes permutation-based variable importance with boxplots showing variability, displays the table and plot.
- Computes partial dependence-like model profiles if ‘features‘ are provided.
- Computes Shapley values (SHAP) for a sample of the training observations, displays the SHAP table, and plots a summary bar chart of mean(|SHAP value|) per feature. For classification, it shows separate bars for each class.

Usage

```
fastexplain(
  object,
  method = "dalex",
  features = NULL,
  grid_size = 20,
  shap_sample = 5,
  vi_iterations = 10,
  seed = 123,
  loss_function = NULL,
  ...
)
```

Arguments

object	A fastml object.
method	Currently only "dalex" is supported.
features	Character vector of feature names for partial dependence (model profiles). Default NULL.
grid_size	Number of grid points for partial dependence. Default 20.
shap_sample	Integer number of observations from processed training data to compute SHAP values for. Default 5.
vi_iterations	Integer. Number of permutations for variable importance (B). Default 10.
seed	Integer. A value specifying the random seed.
loss_function	Function. The loss function for model_parts. • If NULL and task = 'classification', defaults to DALEX::loss_cross_entropy. • If NULL and task = 'regression', defaults to DALEX::loss_root_mean_square.
...	Additional arguments (not currently used).

Details

1. **Custom number of permutations for VI (vi_iterations):**
   You can now specify how many permutations (B) to use for permutation-based variable importance. More permutations yield more stable estimates but take longer.
2. **Better error messages and checks:**
   Improved checks and messages if certain packages or conditions are not met.
3. **Loss Function:**
   A loss_function argument has been added to let you pick a different performance measure (e.g., loss_cross_entropy for classification, loss_root_mean_square for regression).
4. **Parallelization Suggestion:**

Value

Prints DALEX explanations: variable importance table & plot, model profiles (if any), and SHAP table & summary plot.

---

fastexplore *Explore and Summarize a Dataset Quickly*

Description

fastexplore provides a fast and comprehensive exploratory data analysis (EDA) workflow. It automatically detects variable types, checks for missing and duplicated data, suggests potential ID columns, and provides a variety of plots (histograms, boxplots, scatterplots, correlation heatmaps, etc.). It also includes optional outlier detection, normality testing, and feature engineering.

Usage

```
fastexplore(
  data,
  label = NULL,
  visualize = c("histogram", "boxplot", "barplot", "heatmap", "scatterplot"),
  save_results = TRUE,
  output_dir = NULL,
  sample_size = NULL,
  interactive = FALSE,
  corr_threshold = 0.9,
  auto_convert_numeric = TRUE,
  visualize_missing = TRUE,
  imputation_suggestions = FALSE,
  report_duplicate_details = TRUE,
  detect_near_duplicates = TRUE,
  auto_convert_dates = FALSE,
  feature_engineering = FALSE,
  outlier_method = c("iqr", "zscore", "dbscan", "lof"),
  run_distribution_checks = TRUE,
  normality_tests = c("shapiro"),
  pairwise_matrix = TRUE,
  max_scatter_cols = 5,
  grouped_plots = TRUE,
  use_upset_missing = TRUE
)
```

Arguments

`data`	A `data.frame`. The dataset to analyze.
`label`	A character string specifying the name of the target or label column (optional). If provided, certain grouped plots and class imbalance checks will be performed.
`visualize`	A character vector specifying which visualizations to produce. Possible values: `c("histogram", "boxplot", "barplot", "heatmap", "scatterplot")`.
`save_results`	Logical. If `TRUE`, saves plots and a rendered report (HTML) into a timestamped `EDA_Results_` folder inside `output_dir`.
`output_dir`	A character string specifying the output directory for saving results (if `save_results = TRUE`). Defaults to current working directory.
`sample_size`	An integer specifying a random sample size for the data to be used in visualizations. If `NULL`, uses the entire dataset.
`interactive`	Logical. If `TRUE`, attempts to produce interactive Plotly heatmaps and other interactive elements. If required packages are not installed, falls back to static plots.
`corr_threshold`	Numeric. Threshold above which correlations (in absolute value) are flagged as high. Defaults to `0.9`.
`auto_convert_numeric`	Logical. If `TRUE`, automatically converts factor/character columns that look numeric (only digits, minus sign, or decimal point) to numeric.

`visualize_missing`
: Logical. If `TRUE`, attempts to visualize missingness patterns (e.g., via an `UpSet` plot, if **UpSetR** is available, or **VIM**, **naniar**).

`imputation_suggestions`
: Logical. If `TRUE`, prints simple text suggestions for imputation strategies.

`report_duplicate_details`
: Logical. If `TRUE`, shows top duplicated rows and their frequency.

`detect_near_duplicates`
: Logical. Placeholder for near-duplicate (fuzzy) detection. Currently not implemented.

`auto_convert_dates`
: Logical. If `TRUE`, attempts to detect and convert date-like strings (`YYYY-MM-DD`) to `Date` format.

`feature_engineering`
: Logical. If `TRUE`, automatically engineers derived features (day, month, year) from any date/time columns, and identifies potential ID columns.

`outlier_method`
: A character string indicating which outlier detection method(s) to apply. One of `c("iqr", "zscore", "dbscan", "lof")`. Only the first match will be used in the code (though the function is designed to handle multiple).

`run_distribution_checks`
: Logical. If `TRUE`, runs normality tests (e.g., Shapiro-Wilk) on numeric columns.

`normality_tests`
: A character vector specifying which normality tests to run. Possible values include `"shapiro"` or `"ks"` (Kolmogorov-Smirnov). Only used if `run_distribution_checks = TRUE`.

`pairwise_matrix`
: Logical. If `TRUE`, produces a scatterplot matrix (using **GGally**) for numeric columns.

`max_scatter_cols`
: Integer. Maximum number of numeric columns to include in the pairwise matrix.

`grouped_plots`
: Logical. If `TRUE`, produce grouped histograms, violin plots, and density plots by label (if the label is a factor).

`use_upset_missing`
: Logical. If `TRUE`, attempts to produce an UpSet plot for missing data if **UpSetR** is available.

Details

This function automates many steps of EDA:

1. Automatically detects numeric vs. categorical variables.
2. Auto-converts columns that look numeric (and optionally date-like).
3. Summarizes data structure, missingness, duplication, and potential ID columns.
4. Computes correlation matrix and flags highly correlated pairs.
5. (Optional) Outlier detection using IQR, Z-score, DBSCAN, or LOF methods.
6. (Optional) Normality tests on numeric columns.
7. Saves all results and an R Markdown report if `save_results = TRUE`.

Value

A (silent) list containing:

- `data_overview` - A basic overview (head, unique values, skim summary).
- `summary_stats` - Summary statistics for numeric columns.
- `freq_tables` - Frequency tables for factor columns.
- `missing_data` - Missing data overview (count, percentage).
- `duplicated_rows` - Count of duplicated rows.
- `class_imbalance` - Class distribution if `label` is provided and is categorical.
- `correlation_matrix` - The correlation matrix for numeric variables.
- `zero_variance_cols` - Columns with near-zero variance.
- `potential_id_cols` - Columns with unique values in every row.
- `date_time_cols` - Columns recognized as date/time.
- `high_corr_pairs` - Pairs of variables with correlation above `corr_threshold`.
- `outlier_method` - The chosen method for outlier detection.
- `outlier_summary` - Outlier proportions or metrics (if computed).

If `save_results = TRUE`, additional side effects include saving figures, a correlation heatmap, and an R Markdown report in the specified directory.

`fastml` *Fast Machine Learning Function*

Description

Trains and evaluates multiple classification or regression models automatically detecting the task based on the target variable type.

Usage

```
fastml(
  data = NULL,
  train_data = NULL,
  test_data = NULL,
  label,
  algorithms = "all",
  task = "auto",
  test_size = 0.2,
  resampling_method = "cv",
  folds = ifelse(grepl("cv", resampling_method), 10, 25),
  repeats = ifelse(resampling_method == "repeatedcv", 1, NA),
  event_class = "first",
  exclude = NULL,
```

```
  recipe = NULL,
  tune_params = NULL,
  metric = NULL,
  algorithm_engines = NULL,
  n_cores = 1,
  stratify = TRUE,
  impute_method = "error",
  impute_custom_function = NULL,
  encode_categoricals = TRUE,
  scaling_methods = c("center", "scale"),
  summaryFunction = NULL,
  use_default_tuning = FALSE,
  tuning_strategy = "grid",
  tuning_iterations = 10,
  early_stopping = FALSE,
  adaptive = FALSE,
  learning_curve = FALSE,
  seed = 123
)
```

Arguments

`data`	A data frame containing the complete dataset. If both ‘train_data‘ and ‘test_data‘ are ‘NULL‘, ‘fastml()‘ will split this into training and testing sets according to ‘test_size‘ and ‘stratify‘. Defaults to ‘NULL‘.
`train_data`	A data frame pre-split for model training. If provided, ‘test_data‘ must also be supplied, and no internal splitting will occur. Defaults to ‘NULL‘.
`test_data`	A data frame pre-split for model evaluation. If provided, ‘train_data‘ must also be supplied, and no internal splitting will occur. Defaults to ‘NULL‘.
`label`	A string specifying the name of the target variable.
`algorithms`	A vector of algorithm names to use. Default is `"all"` to run all supported algorithms.
`task`	Character string specifying model type selection. Use "auto" to let the function detect whether the target is for classification or regression based on the data, or explicitly set to "classification" or "regression".
`test_size`	A numeric value between 0 and 1 indicating the proportion of the data to use for testing. Default is `0.2`.
`resampling_method`	A string specifying the resampling method for model evaluation. Default is `"cv"` (cross-validation). Other options include `"none"`, `"boot"`, `"repeatedcv"`, etc.
`folds`	An integer specifying the number of folds for cross-validation. Default is `10` for methods containing "cv" and `25` otherwise.
`repeats`	Number of times to repeat cross-validation (only applicable for methods like "repeatedcv").
`event_class`	A single string. Either "first" or "second" to specify which level of truth to consider as the "event". Default is "first".

`exclude` A character vector specifying the names of the columns to be excluded from the training process.

`recipe` A user-defined `recipe` object for custom preprocessing. If provided, internal recipe steps (imputation, encoding, scaling) are skipped.

`tune_params` A list specifying hyperparameter tuning ranges. Default is `NULL`.

`metric` The performance metric to optimize during training.

`algorithm_engines`
A named list specifying the engine to use for each algorithm.

`n_cores` An integer specifying the number of CPU cores to use for parallel processing. Default is `1`.

`stratify` Logical indicating whether to use stratified sampling when splitting the data. Default is `TRUE` for classification and `FALSE` for regression.

`impute_method` Method for handling missing values. Options include:

- `"medianImpute"` Impute missing values using median imputation (recipe-based).
- `"knnImpute"` Impute missing values using k-nearest neighbors (recipe-based).
- `"bagImpute"` Impute missing values using bagging (recipe-based).
- `"remove"` Remove rows with missing values from the data (recipe-based).
- `"mice"` Impute missing values using MICE (Multiple Imputation by Chained Equations).
- `"missForest"` Impute missing values using the missForest algorithm.
- `"custom"` Use a user-provided imputation function (see 'impute_custom_function').
- `"error"` Do not perform imputation; if missing values are detected, stop execution with an error.
- `NULL` Equivalent to `"error"`. No imputation is performed, and the function will stop if missing values are present.

Default is `"error"`.

`impute_custom_function`
A function that takes a data.frame as input and returns an imputed data.frame. Used only if `impute_method = "custom"`.

`encode_categoricals`
Logical indicating whether to encode categorical variables. Default is `TRUE`.

`scaling_methods`
Vector of scaling methods to apply. Default is `c("center", "scale")`.

`summaryFunction`
A custom summary function for model evaluation. Default is `NULL`.

`use_default_tuning`
Logical indicating whether to use default tuning grids when `tune_params` is `NULL`. Default is `FALSE`.

`tuning_strategy`
A string specifying the tuning strategy. Options might include `"grid"`, `"bayes"`, or `"none"`. Default is `"grid"`.

`tuning_iterations`
Number of tuning iterations (applicable for Bayesian or other iterative search methods). Default is `10`.

early_stopping	Logical indicating whether to use early stopping in Bayesian tuning methods (if supported). Default is FALSE.
adaptive	Logical indicating whether to use adaptive/racing methods for tuning. Default is FALSE.
learning_curve	Logical. If TRUE, generate learning curves (performance vs. training size).
seed	An integer value specifying the random seed for reproducibility.

Details

Fast Machine Learning Function

Trains and evaluates multiple classification or regression models. The function automatically detects the task based on the target variable type and can perform advanced hyperparameter tuning using various tuning strategies.

Value

An object of class fastml containing the best model, performance metrics, and other information.

Examples

```
# Example 1: Using the iris dataset for binary classification (excluding 'setosa')
data(iris)
iris <- iris[iris$Species != "setosa", ]  # Binary classification
iris$Species <- factor(iris$Species)

# Train models
model <- fastml(
  data = iris,
  label = "Species",
  algorithms = c("rand_forest", "xgboost", "svm_rbf"), algorithm_engines = c(
  list(rand_forest = c("ranger","aorsf", "partykit", "randomForest")))
)

# View model summary
summary(model)
```

---

flatten_and_rename_models

Flatten and Rename Models

Description

Flattens a nested list of models and renames the elements by combining the outer and inner list names.

Usage

```
flatten_and_rename_models(models)
```

Arguments

`models`	A nested list of models. The outer list should have names. If an inner element is a named list, the names will be combined with the outer name in the format `"outer_name (inner_name)"`.

Details

The function iterates over each element of the outer list. For each element, if it is a list with names, the function concatenates the outer list name and the inner names using `paste0` and `setNames`. If an element is not a list or does not have names, it is included in the result without modification.

Value

A flattened list with each element renamed according to its original outer and inner list names.

`framingham` *Framingham Heart Study Data*

Description

This dataset is derived from the Framingham Heart Study and contains various clinical and demographic variables used to predict coronary heart disease risk over a ten-year period.

Format

male Integer indicator for male sex.

age Participant age in years.

education Education level.

currentSmoker Whether the participant currently smokes.

cigsPerDay Number of cigarettes smoked per day.

BPMeds Whether blood pressure medication is used.

prevalentStroke History of stroke at baseline.

prevalentHyp History of hypertension at baseline.

diabetes Diabetes diagnosis.

totChol Total cholesterol.

sysBP Systolic blood pressure.

diaBP Diastolic blood pressure.

BMI Body mass index.

heartRate Heart rate.

glucose Glucose level.

TenYearCHD Ten year risk of coronary heart disease.

get_best_model_idx *Get Best Model Indices by Metric and Group*

Description

Identifies and returns the indices of rows in a data frame where the specified metric reaches the overall maximum within groups defined by one or more columns.

Usage

```
get_best_model_idx(df, metric, group_cols = c("Model", "Engine"))
```

Arguments

df	A data frame containing model performance metrics and grouping columns.
metric	A character string specifying the name of the metric column in df. The metric values are converted to numeric for comparison.
group_cols	A character vector of column names used for grouping. Defaults to c("Model", "Engine").

Details

The function converts the metric values to numeric and creates a combined grouping factor using the specified group_cols. It then computes the maximum metric value within each group and determines the overall best metric value across the entire data frame. Finally, it returns the indices of rows belonging to groups that achieve this overall maximum.

Value

A numeric vector of row indices in df corresponding to groups whose maximum metric equals the overall best metric value.

get_best_model_names *Get Best Model Names*

Description

Extracts and returns the best engine names from a named list of model workflows.

Usage

```
get_best_model_names(models)
```

Arguments

models	A named list where each element corresponds to an algorithm and contains a list of model workflows. Each workflow should be compatible with tune::extract_fit_parsnip.

Details

For each algorithm, the function extracts the engine names from the model workflows using tune::extract_fit_parsnip. It then chooses "randomForest" if it is available; otherwise, it selects the first non-NA engine. If no engine names can be extracted for an algorithm, NA_character_ is returned.

Value

A named character vector. The names of the vector correspond to the algorithm names, and the values represent the chosen best engine name for that algorithm.

---

get_best_workflows *Get Best Workflows*

Description

Extracts the best workflows from a nested list of model workflows based on the provided best model names.

Usage

```
get_best_workflows(models, best_model_name)
```

Arguments

models	A nested list of model workflows. Each element should correspond to an algorithm and contain sublists keyed by engine names.
best_model_name	A named character vector where the names represent algorithm names and the values represent the chosen best engine for each algorithm.

Details

The function iterates over each element in best_model_name and attempts to extract the corresponding workflow from models using the specified engine. If the workflow for an algorithm-engine pair is not found, a warning is issued and NULL is returned for that entry.

Value

A named list of workflows corresponding to the best engine for each algorithm. Each list element is named in the format "algorithm (engine)".

get_default_engine *Get Default Engine*

Description

Returns the default engine corresponding to the specified algorithm.

Usage

```
get_default_engine(algo)
```

Arguments

algo	A character string specifying the name of the algorithm. The value should match one of the supported algorithm names.

Details

The function uses a `switch` statement to select the default engine based on the given algorithm. If the provided algorithm does not have a defined default engine, the function terminates with an error.

Value

A character string containing the default engine name associated with the provided algorithm.

get_default_params *Get Default Parameters for an Algorithm*

Description

Returns a list of default tuning parameters for the specified algorithm based on the task type, number of predictors, and engine.

Usage

```
get_default_params(algo, task, num_predictors = NULL, engine = NULL)
```

Arguments

algo	A character string specifying the algorithm name. Supported values include: `"rand_forest"`, `"C5_rules"`, `"xgboost"`, `"lightgbm"`, `"logistic_reg"`, `"multinom_reg"`, `"decision_tree"`, `"svm_linear"`, `"svm_rbf"`, `"nearest_neighbor"`, `"naive_Bayes"`, `"mlp"`, `"deep_learning"`, `"discrim_linear"`, `"discrim_quad"`, `"bag_tree"`, `"elastic_net"`, `"bayes_glm"`, `"pls"`, `"linear_reg"`, `"ridge_regression"`, and `"lasso_regression"`.

task	A character string specifying the task type, typically "classification" or "regression".
num_predictors	An optional numeric value indicating the number of predictors. This value is used to compute default values for parameters such as mtry. Defaults to NULL.
engine	An optional character string specifying the engine to use. If not provided, a default engine is chosen where applicable.

Details

The function employs a switch statement to select and return a list of default parameters tailored for the given algorithm, task, and engine. The defaults vary by algorithm and, in some cases, by engine. For example:

- For "rand_forest", if engine is not provided, it defaults to "ranger". The parameters such as mtry, trees, and min_n are computed based on the task and the number of predictors.
- For "C5_rules", the defaults include trees, min_n, and sample_size.
- For "xgboost" and "lightgbm", default values are provided for parameters like tree depth, learning rate, and sample size.
- For "logistic_reg" and "multinom_reg", the function returns defaults for regularization parameters (penalty and mixture) that vary with the specified engine.
- For "decision_tree", the parameters (such as tree_depth, min_n, and cost_complexity) are set based on the engine (e.g., "rpart", "C5.0", "partykit", "spark").
- Other algorithms, including "svm_linear", "svm_rbf", "nearest_neighbor", "naive_Bayes", "mlp", "deep_learning", "elastic_net", "bayes_glm", "pls", "linear_reg", "ridge_regression", and "lasso_regression", have their respective default parameter lists.

Value

A list of default parameter settings for the specified algorithm. If the algorithm is not recognized, the function returns NULL.

---

get_default_tune_params

Get Default Tuning Parameters

Description

Returns a list of default tuning parameter ranges for a specified algorithm based on the provided training data, outcome label, and engine.

Usage

```
get_default_tune_params(algo, train_data, label, engine)
```

Arguments

algo	A character string specifying the algorithm name. Supported values include: "rand_forest", "C5_rules", "xgboost", "lightgbm", "logistic_reg", "multinom_reg", "decision_tree", "svm_linear", "svm_rbf", "nearest_neighbor", "naive_Bayes", "mlp", "deep_learning", "discrim_linear", "discrim_quad", "bag_tree", "elastic_net", "bayes_glm", "pls", "linear_reg", "ridge_regression", and "lasso_regression".
train_data	A data frame containing the training data.
label	A character string specifying the name of the outcome variable in train_data. This column is excluded when calculating the number of predictors.
engine	A character string specifying the engine to be used for the algorithm. Different engines may have different tuning parameter ranges.

Details

The function first determines the number of predictors by removing the outcome variable (specified by label) from train_data. It then uses a switch statement to select a list of default tuning parameter ranges tailored for the specified algorithm and engine. The tuning ranges have been adjusted for efficiency and may include parameters such as mtry, trees, min_n, and others depending on the algorithm.

Value

A list of tuning parameter ranges for the specified algorithm. If no tuning parameters are defined for the given algorithm, the function returns NULL.

---

get_engine_names *Get Engine Names from Model Workflows*

Description

Extracts and returns a list of unique engine names from a list of model workflows.

Usage

```
get_engine_names(models)
```

Arguments

models	A list where each element is a list of model workflows. Each workflow is expected to contain a fitted model that can be processed with tune::extract_fit_parsnip.

Details

The function applies tune::extract_fit_parsnip to each model workflow to extract the fitted model object. It then retrieves the engine name from the model specification (spec$engine). If the extraction fails, NA_character_ is returned for that workflow. Finally, the function removes any duplicate engine names using unique.

Value

A list of character vectors. Each vector contains the unique engine names extracted from the corresponding element of `models`.

`get_model_engine_names` *Get Model Engine Names*

Description

Extracts and returns a named vector mapping algorithm names to engine names from a nested list of model workflows.

Usage

```
get_model_engine_names(models)
```

Arguments

`models`	A nested list of model workflows. Each inner list should contain model objects from which a fitted model can be extracted using `tune::extract_fit_parsnip`.

Details

The function iterates over a nested list of model workflows and, for each workflow, attempts to extract the fitted model object using `tune::extract_fit_parsnip`. If successful, it retrieves the algorithm name from the first element of the class attribute of the model specification and the engine name from the specification. The results are combined into a named vector.

Value

A named character vector where the names correspond to algorithm names (e.g., `"rand_forest"`, `"logistic_reg"`) and the values correspond to the associated engine names (e.g., `"ranger"`, `"glm"`).

`load_model` *Load Model Function*

Description

Loads a trained model object from a file.

Usage

```
load_model(filepath)
```

Arguments

filepath	A string specifying the file path to load the model from.

Value

An object of class fastml.

---

plot.fastml *Plot Methods for* fastml *Objects*

Description

plot.fastml produces visual diagnostics for a trained fastml object.

Usage

```
## S3 method for class 'fastml'
plot(
  x,
  algorithm = "best",
  type = c("all", "bar", "roc", "confusion", "calibration", "residual"),
  ...
)
```

Arguments

x	A fastml object (output of fastml()).
algorithm	Character vector specifying which algorithm(s) to include when generating certain plots (e.g., ROC curves). Defaults to "best".
type	Character vector indicating which plot(s) to produce. Options are: "bar" Bar plot of performance metrics across all models/engines. "roc" ROC curve(s) for binary classification models. "confusion" Confusion matrix for the best model(s). "calibration" Calibration plot for the best model(s). "residual" Residual diagnostics for the best model. "all" Produce all available plots.
...	Additional arguments (currently unused).

Details

When type = "all", plot.fastml will produce a bar plot of metrics, ROC curves (classification), confusion matrix, calibration plot, and residual diagnostics (regression). If you specify a subset of types, only those will be drawn.

Examples

```
## Create a binary classification dataset from iris
data(iris)
iris <- iris[iris$Species != "setosa",]
iris$Species <- factor(iris$Species)

## Fit fastml model on binary classification task
model <- fastml(data = iris, label = "Species", algorithms = c("rand_forest", "svm_rbf"))

## 1. Plot all available diagnostics
plot(model, type = "all")

## 2. Bar plot of performance metrics
plot(model, type = "bar")

## 3. ROC curves (only for classification models)
plot(model, type = "roc")

## 4. Calibration plot (requires 'probably' package)
plot(model, type = "calibration")

## 5. ROC curves for specific algorithm(s) only
plot(model, type = "roc", algorithm = "rand_forest")

## 6. Residual diagnostics (only available for regression tasks)
model <- fastml(data = mtcars, label = "mpg", algorithms = c("linear_reg", "xgboost"))
plot(model, type = "residual")
```

`predict.fastml` *Predict method for fastml objects*

Description

Generates predictions from a trained ‘fastml‘ object on new data. Supports both single-model and multi-model workflows, and handles classification and regression tasks with optional post-processing and verbosity.

Usage

```
## S3 method for class 'fastml'
predict(
  object,
  newdata,
  type = "auto",
  model_name = NULL,
  verbose = FALSE,
  postprocess_fn = NULL,
```

```
  ...
)
```

Arguments

object	A fitted ‘fastml‘ object created by the ‘fastml()‘ function.
newdata	A data frame or tibble containing new predictor data for which to generate predictions.
type	Type of prediction to return. One of ‘"auto"‘ (default), ‘"class"‘, ‘"prob"‘, or ‘"numeric"‘. - ‘"auto"‘: chooses ‘"class"‘ for classification and ‘"numeric"‘ for regression. - ‘"prob"‘: returns class probabilities (only for classification). - ‘"class"‘: returns predicted class labels. - ‘"numeric"‘: returns predicted numeric values (for regression).
model_name	(Optional) Name of a specific model to use when ‘object$best_model‘ contains multiple models.
verbose	Logical; if ‘TRUE‘, prints progress messages showing which models are used during prediction.
postprocess_fn	(Optional) A function to apply to the final predictions (e.g., inverse transforms, thresholding).
...	Additional arguments (currently unused).

Value

A vector of predictions, or a named list of predictions (if multiple models are used). If ‘postprocess_fn‘ is supplied, its output will be returned instead.

Examples

```
## Not run:
  set.seed(123)
  model <- fastml(iris, label = "Species")
  test_data <- iris[sample(1:150, 20),-5]

  ## Best model(s) predictions
  preds <- predict(model, newdata = test_data)

  ## Predicted class probabilities using best model(s)
  probs <- predict(model, newdata = test_data, type = "prob")

  ## Prediction from a specific model by name
 single_model_preds <- predict(model, newdata = test_data, model_name = "rand_forest (ranger)")


## End(Not run)
```

process_model *Process Model and Compute Performance Metrics*

Description

Finalizes a tuning result or utilizes an already fitted workflow to generate predictions on test data and compute performance metrics.

Usage

```
process_model(
  model_obj,
  model_id,
  task,
  test_data,
  label,
  event_class,
  engine,
  train_data,
  metric
)
```

Arguments

model_obj	A model object, which can be either a tuning result (an object inheriting from "tune_results") or an already fitted workflow.
model_id	A unique identifier for the model, used in warning messages if issues arise during processing.
task	A character string indicating the type of task, either "classification" or "regression".
test_data	A data frame containing the test data on which predictions will be generated.
label	A character string specifying the name of the outcome variable in test_data.
event_class	For classification tasks, a character string specifying which event class to consider as positive (accepted values: "first" or "second").
engine	A character string specifying the modeling engine used. This parameter affects prediction types and metric computations.
train_data	A data frame containing the training data used to fit tuned models.
metric	A character string specifying the metric name used to select the best tuning parameters.

Details

The function first checks if model_obj is a tuning result. If so, it attempts to:

- Select the best tuning parameters using tune::select_best (note that the metric used for selection should be defined in the calling environment).

- Extract the model specification and preprocessor from `model_obj` using `workflows::pull_workflow_spec` and `workflows::pull_workflow_preprocessor`, respectively.
- Finalize the model specification with the selected parameters via `tune::finalize_model`.
- Rebuild the workflow using `workflows::workflow`, `workflows::add_recipe`, and `workflows::add_model`, and fit the finalized workflow with `parsnip::fit` on the supplied `train_data`.

If `model_obj` is already a fitted workflow, it is used directly.

For classification tasks, the function makes class predictions (and probability predictions if `engine` is not `"LiblineaR"`) and computes performance metrics using functions from the `yardstick` package. In binary classification, the positive class is determined based on the `event_class` argument and ROC AUC is computed accordingly. For multiclass classification, macro-averaged metrics and ROC AUC (using weighted estimates) are calculated.

For regression tasks, the function predicts outcomes and computes regression metrics (RMSE, R-squared, and MAE).

If the number of predictions does not match the number of rows in `test_data`, the function stops with an informative error message regarding missing values and imputation options.

Value

A list with two components:

performance A data frame of performance metrics. For classification tasks, metrics include accuracy, kappa, sensitivity, specificity, precision, F-measure, and ROC AUC (when applicable). For regression tasks, metrics include RMSE, R-squared, and MAE.

predictions A data frame containing the test data augmented with predicted classes and, when applicable, predicted probabilities.

`sanitize` *Clean Column Names or Character Vectors by Removing Special Characters*

Description

This function can operate on either a data frame or a character vector:

- **Data frame**: Detects columns whose names contain any character that is not a letter, number, or underscore, removes colons, replaces slashes with underscores, and spaces with underscores.
- **Character vector**: Applies the same cleaning rules to every element of the vector.

Usage

```
sanitize(x)
```

Arguments

`x`	A data frame or character vector to be cleaned.

Value

- If x is a data frame: returns a data frame with cleaned column names.
- If x is a character vector: returns a character vector with cleaned elements.

---

save.fastml *Save Model Function*

Description

Saves the trained model object to a file.

Usage

```
save.fastml(model, filepath)
```

Arguments

model	An object of class fastml.
filepath	A string specifying the file path to save the model.

Value

No return value, called for its side effect of saving the model object to a file.

---

summary.fastml *Summary Function for fastml (Using yardstick for ROC Curves)*

Description

Summarizes the results of machine learning models trained using the ‘fastml‘ package. Depending on the task type (classification or regression), it provides customized output such as performance metrics, best hyperparameter settings, and confusion matrices. It is designed to be informative and readable, helping users quickly interpret model results.

Usage

```
## S3 method for class 'fastml'
summary(
  object,
  algorithm = "best",
  type = c("all", "metrics", "params", "conf_mat"),
  sort_metric = NULL,
  ...
)
```

Arguments

object	An object of class fastml.
algorithm	A vector of algorithm names to display summary. Default is "best".
type	Character vector indicating which outputs to produce. Options are "all" (all available outputs), "metrics" (performance metrics), "params" (best hyperparameters), and "conf_mat" (confusion matrix). Default is "all".
sort_metric	The metric to sort by. Default uses optimized metric.
...	Additional arguments.

Details

For classification tasks, the summary includes metrics such as Accuracy, F1 Score, Kappa, Precision, ROC AUC, Sensitivity, and Specificity. A confusion matrix is also provided for the best model(s). For regression tasks, the summary reports RMSE, R-squared, and MAE.

Users can control the type of output with the 'type' argument: 'metrics' displays model performance metrics. 'params' shows the best hyperparameter settings. 'conf_mat' prints confusion matrices (only for classification). 'all' includes all of the above.

If multiple algorithms are trained, the summary highlights the best model based on the optimized metric.

Value

Prints summary of fastml models.

---

train_models *Train Specified Machine Learning Algorithms on the Training Data*

Description

Trains specified machine learning algorithms on the preprocessed training data.

Usage

```
train_models(
  train_data,
  label,
  task,
  algorithms,
  resampling_method,
  folds,
  repeats,
  tune_params,
  metric,
  summaryFunction = NULL,
  seed = 123,
```

```
  recipe,
  use_default_tuning = FALSE,
  tuning_strategy = "grid",
  tuning_iterations = 10,
  early_stopping = FALSE,
  adaptive = FALSE,
  algorithm_engines = NULL
)
```

Arguments

train_data	Preprocessed training data frame.
label	Name of the target variable.
task	Type of task: "classification" or "regression".
algorithms	Vector of algorithm names to train.
resampling_method	Resampling method for cross-validation (e.g., "cv", "repeatedcv", "boot", "none").
folds	Number of folds for cross-validation.
repeats	Number of times to repeat cross-validation (only applicable for methods like "repeatedcv").
tune_params	List of hyperparameter tuning ranges.
metric	The performance metric to optimize.
summaryFunction	A custom summary function for model evaluation. Default is NULL.
seed	An integer value specifying the random seed for reproducibility.
recipe	A recipe object for preprocessing.
use_default_tuning	Logical indicating whether to use default tuning grids when tune_params is NULL.
tuning_strategy	A string specifying the tuning strategy ("grid", "bayes", or "none"), possibly with adaptive methods.
tuning_iterations	Number of iterations for iterative tuning methods.
early_stopping	Logical for early stopping in Bayesian tuning.
adaptive	Logical indicating whether to use adaptive/racing methods.
algorithm_engines	A named list specifying the engine to use for each algorithm.

Value

A list of trained model objects.

Index