

# Package ‘fastml’

March 7, 2025

**Type** Package

**Title** Fast Machine Learning Model Training and Evaluation

**Version** 0.5.0

**Date** 2025-03-07

**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.

**License** GPL (>= 2)

**Encoding** UTF-8

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

**Maintainer** Selcuk Korkmaz <[selcukorkmaz@gmail.com](mailto:selcukorkmaz@gmail.com)>

**Repository** CRAN

**Date/Publication** 2025-03-07 11:50:11 UTC

## Contents

availableMethods . . . . .	2
evaluate_models . . . . .	3
fastexplain . . . . .	4
fastexplore . . . . .	5
fastml . . . . .	8
flatten_and_rename_models . . . . .	11
get_best_model_idx . . . . .	12
get_best_model_names . . . . .	13
get_best_workflows . . . . .	13
get_default_engine . . . . .	14
get_default_params . . . . .	15
get_default_tune_params . . . . .	16
get_engine_names . . . . .	17
get_model_engine_names . . . . .	17
load_model . . . . .	18
plot.fastml_model . . . . .	18
predict.fastml_model . . . . .	19
process_model . . . . .	19
sanitize . . . . .	21
save_model . . . . .	21
summary.fastml_model . . . . .	22
train_models . . . . .	23
<b>Index</b>	<b>25</b>

---

availableMethods	<i>Get Available Methods</i>
------------------	------------------------------

---

### 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 <a href="#">match.arg</a> 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	<i>Evaluate Models Function</i>
-----------------	---------------------------------

---

## 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	<i>FastExplain the fastml_model (DALEX + SHAP + Permutation-based VI)</i>
-------------	---

---

**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_model 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. <ul style="list-style-type: none"> <li>• If NULL and task = 'classification', defaults to DALEX::loss_cross_entropy.</li> <li>• If NULL and task = 'regression', defaults to DALEX::loss_root_mean_square.</li> </ul>
...	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

<code>data</code>	A <code>data.frame</code> . The dataset to analyze.
<code>label</code>	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.
<code>visualize</code>	A character vector specifying which visualizations to produce. Possible values: <code>c("histogram", "boxplot", "barplot", "heatmap", "scatterplot")</code> .
<code>save_results</code>	Logical. If <code>TRUE</code> , saves plots and a rendered report (HTML) into a timestamped <code>EDA_Results_</code> folder inside <code>output_dir</code> .
<code>output_dir</code>	A character string specifying the output directory for saving results (if <code>save_results = TRUE</code> ). Defaults to current working directory.
<code>sample_size</code>	An integer specifying a random sample size for the data to be used in visualizations. If <code>NULL</code> , uses the entire dataset.
<code>interactive</code>	Logical. If <code>TRUE</code> , attempts to produce interactive Plotly heatmaps and other interactive elements. If required packages are not installed, falls back to static plots.
<code>corr_threshold</code>	Numeric. Threshold above which correlations (in absolute value) are flagged as high. Defaults to <code>0.9</code> .
<code>auto_convert_numeric</code>	Logical. If <code>TRUE</code> , automatically converts factor/character columns that look numeric (only digits, minus sign, or decimal point) to numeric.
<code>visualize_missing</code>	Logical. If <code>TRUE</code> , attempts to visualize missingness patterns (e.g., via an UpSet plot, if <b>UpSetR</b> is available, or <b>VIM</b> , <b>naniar</b> ).
<code>imputation_suggestions</code>	Logical. If <code>TRUE</code> , prints simple text suggestions for imputation strategies.

<code>report_duplicate_details</code>	Logical. If TRUE, shows top duplicated rows and their frequency.
<code>detect_near_duplicates</code>	Logical. Placeholder for near-duplicate (fuzzy) detection. Currently not implemented.
<code>auto_convert_dates</code>	Logical. If TRUE, attempts to detect and convert date-like strings (YYYY-MM-DD) to Date format.
<code>feature_engineering</code>	Logical. If TRUE, automatically engineers derived features (day, month, year) from any date/time columns, and identifies potential ID columns.
<code>outlier_method</code>	A character string indicating which outlier detection method(s) to apply. One of <code>c("iqr", "zscore", "dbscan", "lof")</code> . Only the first match will be used in the code (though the function is designed to handle multiple).
<code>run_distribution_checks</code>	Logical. If TRUE, runs normality tests (e.g., Shapiro-Wilk) on numeric columns.
<code>normality_tests</code>	A character vector specifying which normality tests to run. Possible values include "shapiro" or "ks" (Kolmogorov-Smirnov). Only used if <code>run_distribution_checks = TRUE</code> .
<code>pairwise_matrix</code>	Logical. If TRUE, produces a scatterplot matrix (using <b>GGally</b> ) for numeric columns.
<code>max_scatter_cols</code>	Integer. Maximum number of numeric columns to include in the pairwise matrix.
<code>grouped_plots</code>	Logical. If TRUE, produce grouped histograms, violin plots, and density plots by label (if the label is a factor).
<code>use_upset_missing</code>	Logical. If TRUE, attempts to produce an UpSet plot for missing data if <b>UpSetR</b> 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,  
  label,  
  algorithms = "all",  
  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 features and target variable.
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.
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_model` 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.

---

`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)".

---

<code>get_default_engine</code>	<i>Get Default Engine</i>
---------------------------------	---------------------------

---

**Description**

Returns the default engine corresponding to the specified algorithm.

**Usage**

```
get_default_engine(algo)
```

**Arguments**

<code>algo</code>	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	<i>Load Model Function</i>
------------	----------------------------

---

**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_model`.

---

plot.fastml_model	<i>Plot Function for fastml_model</i>
-------------------	---------------------------------------

---

**Description**

Generates plots to compare the performance of different models.

**Usage**

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

**Arguments**

x              An object of class `fastml_model`.  
 ...            Additional arguments (not used).

**Value**

Displays comparison plots of model performances.

---

predict.fastml\_model *Predict Function for fastml\_model*

---

**Description**

Makes predictions on new data using the trained model.

**Usage**

```
## S3 method for class 'fastml_model'
predict(object, newdata, type = "auto", ...)
```

**Arguments**

object	An object of class fastml_model.
newdata	A data frame containing new data for prediction.
type	Type of prediction. Default is "auto", which returns class labels for classification and numeric predictions for regression. Other options include "prob" for class probabilities (classification only).
...	Additional arguments (not used).

**Value**

A vector or data frame of predictions.

---

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)
```

### 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.

### 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 training data (the variable `train_data` is expected to be available in the environment).

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	<i>Clean Column Names or Character Vectors by Removing Special Characters</i>
----------	---

---

### 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_model	<i>Save Model Function</i>
------------	----------------------------

---

### Description

Saves the trained model object to a file.

### Usage

```
save_model(model, filepath)
```

### Arguments

model                An object of class fastml\_model.  
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\_model *Summary Function for fastml\_model (Using yardstick for ROC Curves)*

---

### Description

Provides a concise, user-friendly summary of model performances. For classification: - Shows Accuracy, F1 Score, Kappa, Precision, ROC AUC, Sensitivity, Specificity. - Produces a bar plot of these metrics. - Shows ROC curves for binary classification using `yardstick::roc_curve()`. - Displays a confusion matrix and a calibration plot if probabilities are available.

### Usage

```
## S3 method for class 'fastml_model'
summary(
  object,
  algorithm = "best",
  sort_metric = NULL,
  plot = TRUE,
  notes = "",
  ...
)
```

### Arguments

<code>object</code>	An object of class <code>fastml_model</code> .
<code>algorithm</code>	A vector of algorithm names to display summary. Default is "best".
<code>sort_metric</code>	The metric to sort by. Default uses optimized metric.
<code>plot</code>	Logical. If TRUE, produce bar plot, yardstick-based ROC curves (for binary classification), confusion matrix (classification), smooth calibration plot (if probabilities), and residual plots (regression).
<code>notes</code>	User-defined commentary.
<code>...</code>	Additional arguments.

### Details

For regression: - Shows RMSE, R-squared, and MAE. - Produces a bar plot of these metrics. - Displays residual diagnostics (truth vs predicted, residual distribution).

### Value

Prints summary and plots if requested.

---

train_models	<i>Train Specified Machine Learning Algorithms on the Training Data</i>
--------------	---

---

### 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.

<code>seed</code>	An integer value specifying the random seed for reproducibility.
<code>recipe</code>	A recipe object for preprocessing.
<code>use_default_tuning</code>	Logical indicating whether to use default tuning grids when <code>tune_params</code> is NULL.
<code>tuning_strategy</code>	A string specifying the tuning strategy ("grid", "bayes", or "none"), possibly with adaptive methods.
<code>tuning_iterations</code>	Number of iterations for iterative tuning methods.
<code>early_stopping</code>	Logical for early stopping in Bayesian tuning.
<code>adaptive</code>	Logical indicating whether to use adaptive/racing methods.
<code>algorithm_engines</code>	A named list specifying the engine to use for each algorithm.

**Value**

A list of trained model objects.



# Index

`availableMethods`, 2

`evaluate_models`, 3

`fastexplain`, 4

`fastexplore`, 5

`fastml`, 8

`flatten_and_rename_models`, 11

`get_best_model_idx`, 12

`get_best_model_names`, 13

`get_best_workflows`, 13

`get_default_engine`, 14

`get_default_params`, 15

`get_default_tune_params`, 16

`get_engine_names`, 17

`get_model_engine_names`, 17

`load_model`, 18

`match.arg`, 2

`plot.fastml_model`, 18

`predict.fastml_model`, 19

`process_model`, 19

`sanitize`, 21

`save_model`, 21

`summary.fastml_model`, 22

`train_models`, 23