Package 'fastml'

January 8, 2025

Type Package

Title Fast Machine Learning Model Training and Evaluation

Version 0.4.0

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,

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>

Repository CRAN

Date/Publication 2025-01-08 08:10:05 UTC

Contents

`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_model (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_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. • 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,
  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,
```

```
  n_cores = 1,
  stratify = TRUE,
  impute_method = "error",
  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,
  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.
`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. `"knnImpute"` Impute missing values using k-nearest neighbors.

"bagImpute" Impute missing values using bagging.

"remove" Remove rows with missing values from the data.

"error" Do not perform imputation; if missing values are detected after preprocessing, 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".

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.

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("random_forest", "xgboost", "svm_radial")
)

# View model summary
summary(model)

  # Example 2: Using the mtcars dataset for regression
  data(mtcars)

  # Train models
  model <- fastml(
    data = mtcars,
    label = "mpg",
    algorithms = c("random_forest", "xgboost", "svm_radial")
  )

  # View model summary
  summary(model)
```

`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_model`.

plot.fastml_model *Plot Function for fastml_model*

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.

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_model *Save Model Function*

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,
  combined_roc = TRUE,
  notes = "",
  ...
)
```

Arguments

object	An object of class fastml_model.
algorithm	A vector of algorithm names to display summary. Default is "best".
sort_metric	The metric to sort by. Default uses optimized metric.
plot	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).
combined_roc	Logical. If TRUE, combined ROC plot; else separate ROC plots.
notes	User-defined commentary.
...	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` *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
)
```

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.

Value

A list of trained model objects.

Index