Package 'fastml'

December 16, 2024

Type Package

Title Fast Machine Learning Model Training and Evaluation

Version 0.3.0

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

Suggests testthat (>= 3.0.0), knitr, rmarkdown, C50, glmnet, xgboost, ranger, crayon, kernlab, keras, lightgbm, rstanarm

RoxygenNote 7.3.2

NeedsCompilation no

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

Maintainer Selcuk Korkmaz <selcukorkmaz@gmail.com>

Repository CRAN

Date/Publication 2024-12-16 22:40:02 UTC

Contents

evaluate_models 2
explain 3
fastml 4
load_model 7

plot.fastml_model 7
predict.fastml_model 8
save_model 9
summary.fastml_model 9
train_models 10

Index **12**

`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)
```

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").

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.

`explain` *Explain 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 dependency-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

```
explain(
  object,
  method = "dalex",
  features = NULL,
  grid_size = 20,
  shap_sample = 5,
  vi_iterations = 10,
  colormap = "viridis",
  top_features = NULL,
  seed = 123,
  ...
)
```

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. Default 10.
`colormap`	Character. Name of a color palette to use (e.g., "viridis"). Default "viridis".
`top_features`	Integer. Limit the SHAP summary plot to top N features by mean abs SHAP. Default NULL (no limit).
`seed`	Integer. A value specifying the random seed.
`...`	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. Custom color palette (colormap): A ‘colormap‘ parameter allows you to select a color palette (e.g., "viridis") for SHAP summary plots and variable importance plots. This improves aesthetics over default palettes.

3. Top Features in SHAP Summary (top_features): You can limit the SHAP summary plot to the top N features by mean absolute SHAP value. This helps focus on the most influential features.

4. Support for calibration plot if probably is available (calibration): If ‘calibration = TRUE‘ and ‘probably‘ is installed, it attempts to produce a model-based calibration plot (e.g., ‘cal_plot_logistic‘). This provides a smoothed, nonparametric view of model calibration.

5. Better error messages and checks: Improved checks and messages if certain packages or conditions are not met.

Value

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

`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),
  tune_params = NULL,
  metric = NULL,
  n_cores = 1,
  stratify = NULL,
  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,
  recipe = NULL
)
```

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").
`tune_params`	A list specifying hyperparameter tuning ranges. Default is `NULL`.
`metric`	The performance metric to optimize during training. Default depends on the task.
`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.

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

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 with Bayesian optimization
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.

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,
  sort_metric = NULL,
  plot = TRUE,
  combined_roc = TRUE,
  notes = "",
  ...
)
```

Arguments

object	An object of class fastml_model.
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

evaluate_models, 2
explain, 3

fastml, 4

load_model, 7

plot.fastml_model, 7
predict.fastml_model, 8

save_model, 9
summary.fastml_model, 9

train_models, 10