Package 'unifiedml'

April 3, 2026

Type Package

Title Unified Interface for Machine Learning Models

Version 0.2.1

Date 2026-04-03

Maintainer T. Moudiki <thierry.moudiki@gmail.com>

Description Provides a unified R6-based interface for various machine learning models with automatic interface detection, consistent cross-validation, model interpretations via numerical derivatives, and visualization. Supports both regression and classification tasks with any model function that follows R's standard modeling conventions (formula or matrix interface).

License MIT + file LICENSE

URL https://github.com/Techtonique/unifiedml

BugReports https://github.com/Techtonique/unifiedml/issues

Depends R (>= 3.5.0), doParallel, R6, foreach

Imports Rcpp (>= 1.1.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, glmnet, randomForest, e1071, covr, spelling, MASS

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation yes

Author T. Moudiki [aut, cre]

Repository CRAN

Date/Publication 2026-04-03 15:30:02 UTC

Contents

`unifiedml-package` *Unified Interface for Machine Learning Models*

Description

Provides a unified R6-based interface for various machine learning models with automatic interface detection, consistent cross-validation, model interpretations via numerical derivatives, and visualization. Supports both regression and classification tasks with any model function that follows R's standard modeling conventions (formula or matrix interface).

Package Content

Index of help topics:

```
Model                   Unified Machine Learning Interface using R6
cross_val_score         Cross-Validation for Model Objects
formula_to_matrix       Convert a formula-based model to a matrix
                        interface
matrix_to_formula       Convert a matrix-based model to a formula
                        interface
unifiedml-package       Unified Interface for Machine Learning Models
```

Maintainer

T. Moudiki <thierry.moudiki@gmail.com>

Author(s)

T. Moudiki [aut, cre]

`cross_val_score` *Cross-Validation for Model Objects*

Description

Perform k-fold cross-validation with consistent scoring metrics across different model types. The scoring metric is automatically selected based on the detected task type.

Usage

```
cross_val_score(
  model,
  X,
  y,
  cv = 5,
  scoring = NULL,
  show_progress = TRUE,
  cl = NULL,
  ...
)
```

Arguments

`model`	A Model object
`X`	Feature matrix or data.frame
`y`	Target vector (type determines regression vs classification)
`cv`	Number of cross-validation folds (default: 5)
`scoring`	Scoring metric: "rmse", "mae", "accuracy", or "f1" (default: auto-detected based on task)
`show_progress`	Whether to show progress bar (default: TRUE)
`cl`	Optional cluster for parallel processing (not yet implemented)
`...`	Additional arguments passed to model$fit()

Value

Vector of cross-validation scores for each fold

Examples

```
library(glmnet)
X <- matrix(rnorm(100), ncol = 4)
y <- 2*X[,1] - 1.5*X[,2] + rnorm(25)  # numeric -> regression

mod <- Model$new(glmnet::glmnet)
mod$fit(X, y, alpha = 0, lambda = 0.1)
cv_scores <- cross_val_score(mod, X, y, cv = 5)  # auto-uses RMSE
```

```
mean(cv_scores)  # Average RMSE

# Classification with accuracy scoring
data(iris)
X_class <- as.matrix(iris[, 1:4])
y_class <- iris$Species  # factor -> classification

mod2 <- Model$new(e1071::svm)
cv_scores2 <- cross_val_score(mod2, X_class, y_class, cv = 5)  # auto-uses accuracy
mean(cv_scores2)  # Average accuracy
```

`formula_to_matrix` *Convert a formula-based model to a matrix interface*

Description

Wraps a model function that expects `formula + data` so it can be called with a plain `X` (data.frame or matrix) and `y` (vector). Factors in `X` are preserved; special column names are safely backtick-quoted in the generated formula so they survive the formula parser.

Usage

```
formula_to_matrix(fit_func, predict_func = stats::predict)
```

Arguments

`fit_func`	A model-fitting function whose first two arguments are `formula` and `data` (e.g. `lm`, `glm`).
`predict_func`	A prediction function with signature `function(model, newdata, ...)`. Defaults to `stats::predict`.

Value

A named list with two elements:

`fit(X, y, weights, ...)` Fits the model. `X` is a data.frame (or coercible matrix), `y` is the response vector. Extra arguments are forwarded to `fit_func`.

`predict(model, newdata, ...)` Generates predictions. `newdata` must have the same columns as the `X` used in `fit`. Extra arguments are forwarded to `predict_func`.

Examples

```
lm_matrix <- formula_to_matrix(lm)
X <- data.frame(wt = mtcars$wt, hp = mtcars$hp, cyl = factor(mtcars$cyl))
y <- mtcars$mpg
model <- lm_matrix$fit(X, y)
lm_matrix$predict(model, X[1:5, ])
```

`matrix_to_formula` *Convert a matrix-based model to a formula interface*

Description

Wraps a model function that expects a numeric matrix `X` and a response vector `y` (like `glmnet::glmnet`) so it can be called with the familiar `formula + data` interface. The formula is expanded via `model.matrix`, which handles factor dummy-coding, interactions, and inline transformations automatically.

Usage

```
matrix_to_formula(
  fit_func,
 predict_func = function(model, newX, ...) stats::predict(model, newdata = newX, ...)
)
```

Arguments

`fit_func`	A model-fitting function whose first two positional arguments are `x` (numeric matrix) and `y` (response vector), e.g. `glmnet::glmnet`.
`predict_func`	A prediction function with signature `function(model, newX, ...)` where `newX` is a numeric matrix. Defaults to a thin wrapper around `stats::predict` that passes `newdata` as `newx`.

Value

A named list with two elements:

`fit(formula, data, ...)` Fits the model. The formula is expanded with `model.matrix`; the intercept column is dropped before passing to `fit_func` (add it back via `...` if your model needs it). Extra arguments are forwarded to `fit_func`.

`predict(model, newdata, ...)` Generates predictions. `newdata` is expanded with the same `model.matrix` terms captured at fit time. Extra arguments are forwarded to `predict_func`.

Examples

```
## Not run:
glmnet_formula <- matrix_to_formula(
  fit_func = glmnet::glmnet,
  predict_func = function(model, newX, ...) {
    glmnet::predict.glmnet(model, newx = newX, s = 0.01, ...)
  }
)
model <- glmnet_formula$fit(mpg ~ wt + hp + factor(cyl), data = mtcars)
glmnet_formula$predict(model, newdata = mtcars[1:5, ])

## End(Not run)
```

Model *Unified Machine Learning Interface using R6*

Description

Provides a consistent interface for various machine learning models in R, with automatic detection of formula vs matrix interfaces, built-in cross-validation, model interpretability, and visualization.

An R6 class that provides a unified interface for regression and classification models with automatic interface detection, cross-validation, and interpretability features. The task type (regression vs classification) is automatically detected from the response variable type.

Public fields

`model_fn` The modeling function (e.g., glmnet::glmnet, randomForest::randomForest)

`fitted` The fitted model object

`task` Type of task: "regression" or "classification" (automatically detected)

`X_train` Training features matrix

`y_train` Training target vector

Methods

Public methods:

- `Model$new()`
- `Model$fit()`
- `Model$predict()`
- `Model$print()`
- `Model$summary()`
- `Model$plot()`
- `Model$clone_model()`
- `Model$clone()`

Method `new()`**:** Initialize a new Model

Usage:

```
Model$new(model_fn)
```

Arguments:

`model_fn` A modeling function (e.g., glmnet, randomForest, svm)

Returns: A new Model object

Method `fit()`**:** Fit the model to training data

Automatically detects task type (regression vs classification) based on the type of the response variable y. Numeric y -> regression, factor y -> classification.

Usage:

```
Model$fit(X, y, ...)
```

Arguments:

`X` Feature matrix or data.frame

`y` Target vector (numeric for regression, factor for classification)

`...` Additional arguments passed to the model function

Returns: self (invisible) for method chaining

Method `predict()`**:** Generate predictions from fitted model

Usage:

```
Model$predict(X, ...)
```

Arguments:

`X` Feature matrix for prediction

`...` Additional arguments passed to predict function

Returns: Vector of predictions

Method `print()`**:** Print model information

Usage:

```
Model$print()
```

Returns: self (invisible) for method chaining

Method `summary()`**:** Compute numerical derivatives and statistical significance

Uses finite differences to compute approximate partial derivatives for each feature, providing model-agnostic interpretability.

Usage:

```
Model$summary(h = 0.01, alpha = 0.05)
```

Arguments:

`h` Step size for finite differences (default: 0.01)

`alpha` Significance level for p-values (default: 0.05)

Details: The method computes numerical derivatives using central differences.
Statistical significance is assessed using t-tests on the derivative estimates across samples.

Returns: A data.frame with derivative statistics (invisible)

Method `plot()`**:** Create partial dependence plot for a feature

Visualizes the relationship between a feature and the predicted outcome while holding other features at their mean values.

Usage:

```
Model$plot(feature = 1, n_points = 100)
```

Arguments:

`feature` Index or name of feature to plot

`n_points` Number of points for the grid (default: 100)

Returns: self (invisible) for method chaining

Method `clone_model()`**:** Create a deep copy of the model

Useful for cross-validation and parallel processing where multiple independent model instances are needed.

Usage:

```
Model$clone_model()
```

Returns: A new Model object with same configuration

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Model$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Your Name

Examples

```
# Regression example with glmnet
library(glmnet)
X <- matrix(rnorm(100), ncol = 4)
y <- 2*X[,1] - 1.5*X[,2] + rnorm(25)  # numeric -> regression

mod <- Model$new(glmnet::glmnet)
mod$fit(X, y, alpha = 0, lambda = 0.1)
mod$summary()
predictions <- mod$predict(X)

# Classification example
data(iris)
iris_binary <- iris[iris$Species %in% c("setosa", "versicolor"), ]
X_class <- as.matrix(iris_binary[, 1:4])
y_class <- iris_binary$Species  # factor -> classification

mod2 <- Model$new(e1071::svm)
mod2$fit(X_class, y_class, kernel = "radial")
mod2$summary()

# Cross-validation
cv_scores <- cross_val_score(mod, X, y, cv = 5)
```

Index