

Package ‘unifiedml’

January 10, 2026

Type Package

Title Unified Interface for Machine Learning Models

Version 0.2.0

Date 2026-01-10

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

LinkingTo Rcpp

Config/testthat/edition 3

NeedsCompilation yes

Author T. Moudiki [aut, cre]

Repository CRAN

Date/Publication 2026-01-10 11:10:02 UTC

Contents

unifiedml-package	2
cross_val_score	3
formula_to_matrix	4
matrix_to_formula	4
Model	5
print.model_adapter	8
rcpp_hello_world	9
Index	10

unifiedml-package	<i>Unified Interface for Machine Learning Models</i>
-------------------	--

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 formula-based model function to matrix interface
matrix_to_formula	Convert matrix-based model function to formula interface
print.model_adapter	Print Method for Model Adapters
rcpp_hello_world	Simple function using Rcpp
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	<i>Convert formula-based model function to matrix interface</i>
-------------------	---

Description

Convert formula-based model function to matrix interface

Usage

```
formula_to_matrix(fit_func, predict_func = NULL, intercept = TRUE)
```

Arguments

fit_func	Function accepting formula and data
predict_func	Optional prediction function; if NULL, uses generic predict()
intercept	Include intercept in formula (default TRUE)

Value

List with fit and predict methods (class: model_adapter)

matrix_to_formula	<i>Convert matrix-based model function to formula interface</i>
-------------------	---

Description

Convert matrix-based model function to formula interface

Usage

```
matrix_to_formula(fit_func, predict_func = NULL, drop_intercept = TRUE)
```

Arguments

<code>fit_func</code>	Function accepting x matrix and y vector
<code>predict_func</code>	Optional prediction function; if NULL, uses generic <code>predict()</code>
<code>drop_intercept</code>	Remove intercept column from model.matrix (default TRUE)

Value

List with fit and predict methods (class: `model_adapter`)

Model	<i>Unified Machine Learning Interface using R6</i>
-------	--

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

<code>model_fn</code>	The modeling function (e.g., <code>glmnet::glmnet</code> , <code>randomForest::randomForest</code>)
<code>fitted</code>	The fitted model object
<code>task</code>	Type of task: "regression" or "classification" (automatically detected)
<code>X_train</code>	Training features matrix
<code>y_train</code>	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, type = NULL, ...)`

Arguments:

`X` Feature matrix for prediction

`type` Type of prediction ("response", "class", "probabilities")

`...` 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)
```

`print.model_adapter` *Print Method for Model Adapters*

Description

Prints a short summary of a `model_adapter` object, including the signatures of its `fit` and `predict` functions.

Usage

```
## S3 method for class 'model_adapter'
print(x, ...)
```

Arguments

<code>x</code>	An object of class <code>model_adapter</code> .
<code>...</code>	Further arguments passed to or from other methods (ignored).

Value

Invisibly returns `x`.

Examples

```
adapter <- list(
  fit = function(data, y) NULL,
  predict = function(newdata) numeric(0)
)
class(adapter) <- "model_adapter"
print(adapter)
```

rcpp_hello_world	<i>Simple function using Rcpp</i>
------------------	-----------------------------------

Description

Simple function using Rcpp

Usage

```
rcpp_hello_world()
```

Examples

```
## Not run:  
rcpp_hello_world()  
  
## End(Not run)
```

Index

- * **package**
 - unifiedml-package, [2](#)
- cross_val_score, [3](#)
- formula_to_matrix, [4](#)
- matrix_to_formula, [4](#)
- Model, [5](#)
- print.model_adapter, [8](#)
- rcpp_hello_world, [9](#)
- unifiedml (unifiedml-package), [2](#)
- unifiedml-package, [2](#)