

# Package ‘mlS3’

May 28, 2026

**Type** Package

**Title** Unified S3 Interface to Machine Learning Models

**Version** 0.1.1

**Date** 2026-05-24

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

**Description** Provides a unified and consistent S3 interface for training and predicting with a variety of machine learning models in R. The package wraps popular algorithms (e.g., from 'glmnet', 'lightgbm', 'ranger', 'e1071', and 'caret') under a common workflow based on simple wrap\_\*() and predict() functions, allowing users to switch between models without changing their code structure. It supports both classification and regression tasks and facilitates rapid experimentation, benchmarking, and comparison of models. By abstracting away package-specific APIs while preserving flexibility in parameter specification, the package streamlines machine learning workflows and promotes reproducibility.

**License** GPL-3

**Encoding** UTF-8

**Imports** e1071, glmnet, lightgbm, ranger

**Suggests** caret, knitr, randomForest, kernlab

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Language** en-US

**NeedsCompilation** no

**Author** T. Moudiki [aut, cre]

**Repository** CRAN

**Date/Publication** 2026-05-28 07:30:16 UTC

## Contents

predict.wrap_caret . . . . .	2
print.wrap_caret . . . . .	3
wrap_caret . . . . .	3

wrap_glmnet . . . . .	4
wrap_lightgbm . . . . .	5
wrap_ranger . . . . .	7
wrap_svm . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

predict.wrap_caret	<i>Predict method for mlS3 caret wrapper</i>
--------------------	--

---

## Description

Predict method for mlS3 caret wrapper

## Usage

```
## S3 method for class 'wrap_caret'
predict(object, newx, type = NULL, ...)
```

## Arguments

object	Object from wrap_caret
newx	New features (matrix or data frame)
type	Prediction type: "raw" (default), "class", "prob", or NULL
...	Additional arguments to caret::predict.train

## Value

Vector or matrix of predictions

## Examples

```
# Only runs if caret is installed

data(mtcars)

# Prepare data
X_reg <- mtcars[, -1] # All except mpg
y_reg <- mtcars$mpg   # Target variable

# Split into train/test
set.seed(123)
idx_reg <- sample(nrow(X_reg), 0.7 * nrow(X_reg))
X_train <- X_reg[idx_reg, ]
y_train <- y_reg[idx_reg]
X_test  <- X_reg[-idx_reg, ]
y_test  <- y_reg[-idx_reg]

mod <- wrap_caret(X_train, y_train, method = "rf", mtry = 3)
```

```
(pred <- predict(mod, X_test))
```

---

```
print.wrap_caret      Print method for wrap_caret objects
```

---

### Description

Print method for wrap\_caret objects

### Usage

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

### Arguments

x	Object from wrap_caret
...	Additional arguments

---

```
wrap_caret           Wrap caret models for mlS3
```

---

### Description

Minimal wrapper around caret::train with no tuning. Hyperparameters can be passed via ... as named arguments.

### Usage

```
wrap_caret(x, y, method = "rf", ...)
```

### Arguments

x	Feature matrix or data frame
y	Response vector
method	caret model method (default "rf")
...	Named hyperparameters (e.g., mtry = 3, ntree = 500)

### Value

Object with class "mlS3\_caret"

**Examples**

```
# Only runs if caret is installed

data(mtcars)

# Prepare data
X_reg <- mtcars[, -1] # All except mpg
y_reg <- mtcars$mpg   # Target variable

# Split into train/test
set.seed(123)
idx_reg <- sample(nrow(X_reg), 0.7 * nrow(X_reg))
X_train <- X_reg[idx_reg, ]
y_train <- y_reg[idx_reg]
X_test  <- X_reg[-idx_reg, ]
y_test  <- y_reg[-idx_reg]

mod <- wrap_caret(X_train, y_train, method = "rf", mtry = 3)
(pred <- predict(mod, X_test))
```

---

wrap\_glmnet

*S3 wrapper for glmnet*


---

**Description**

Fits a ‘glmnet’ penalized regression model with a consistent interface. Supports regression and binary classification.

**Usage**

```
wrap_glmnet(x, y, ...)

## S3 method for class 'wrap_glmnet'
predict(object, newx, type = c("class", "prob"), s = NULL, ...)

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

**Arguments**

x	A matrix or data.frame of features.
y	A factor or character vector for classification, numeric for regression.
...	Additional arguments passed to [glmnet::glmnet()]. Pass ‘family = "binomial"’ for binary classification.
object	A fitted ‘wrap_glmnet’ object.
newx	A matrix or data.frame of new observations.

type	"class" (default) for class labels, "prob" for a probability matrix. Ignored for regression.
s	Lambda value for prediction. Defaults to the midpoint of the lambda path. Pass 's = cv_fit\$lambda.min' if using [glmnet::cv.glmnet()].

### Value

An object of class 'wrap\_glmnet' with fields:

fit	The fitted glmnet model.
levels	Class levels (NULL for regression).
task	"classification" or "regression".

### Note

Multiclass ('family = "multinomial"') is not yet supported. For lambda selection, a specific 's' value can be passed to 'predict()'. By default the midpoint of the lambda path is used. For optimal lambda, use [glmnet::cv.glmnet()] externally and pass 's = fit\$lambda.min'.

### Examples

```
X <- iris[iris$Species != "virginica", 1:4]
y <- droplevels(iris[iris$Species != "virginica", "Species"])
mod <- wrap_glmnet(X, y, family = "binomial")
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")
```

```
X <- iris[iris$Species != "virginica", 1:4]
y <- droplevels(iris[iris$Species != "virginica", "Species"])
mod <- wrap_glmnet(X, y, family = "binomial")
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")
```

---

wrap\_lightgbm

*S3 wrapper for lightgbm*


---

### Description

Fits a 'lightgbm' model with a consistent interface. Supports binary classification, multiclass classification, and regression.

**Usage**

```
wrap_lightgbm(x, y, ...)

## S3 method for class 'wrap_lightgbm'
predict(object, newx, type = c("class", "prob"), ...)

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

**Arguments**

x	A matrix or data.frame of features.
y	A factor or character vector for classification, numeric for regression.
...	Additional arguments passed to [lightgbm::lgb.train()]. Pass 'params = list(objective = "binary")' for binary classification, 'params = list(objective = "multiclass", num_class = k)' for multiclass, or 'params = list(objective = "regression")' for regression.
object	A fitted 'wrap_lightgbm' object.
newx	A matrix or data.frame of new observations.
type	"class" (default) for class labels, "prob" for a probability matrix. Ignored for regression.

**Value**

An object of class 'wrap\_lightgbm' with fields:

fit	The fitted lgb.Booster model.
levels	Class levels (NULL for regression).
task	"classification" or "regression".
objective	The lightgbm objective string, stored at fit time.

**Examples**

```
## Not run:
library(m1S3)

X <- iris[, 1:4]
y <- iris$Species
mod <- wrap_lightgbm(X, y,
  params = list(objective = "multiclass", num_class = 3, verbose = -1),
  nrounds = 50)
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")

## End(Not run)

## Not run:
```

```

library(mIS3)

X <- iris[, 1:4]
y <- iris$Species
mod <- wrap_lightgbm(X, y,
  params = list(objective = "multiclass", num_class = 3, verbose = -1),
  nrounds = 50)
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")

## End(Not run)

```

---

wrap\_ranger

*S3 wrapper for ranger*


---

## Description

Fits a ‘ranger’ random forest with a consistent interface. Supports both classification (factor ‘y’) and regression (numeric ‘y’).

## Usage

```

wrap_ranger(x, y, ...)

## S3 method for class 'wrap_ranger'
predict(object, newx, type = c("class", "prob"), ...)

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

```

## Arguments

x	A matrix or data.frame of features.
y	A factor or character vector for classification, numeric for regression.
...	Additional arguments passed to [ranger::ranger()].
object	A fitted ‘wrap_ranger’ object.
newx	A matrix or data.frame of new observations.
type	“class” (default) for class labels, “prob” for a probability matrix. Ignored for regression.

## Value

An object of class ‘wrap\_ranger’ with fields:

fit	The fitted ranger model.
levels	Class levels (NULL for regression).
task	“classification” or “regression”.

**Examples**

```
X <- as.matrix(iris[, 1:4])
y <- iris$Species
mod <- wrap_ranger(X, y, num.trees = 100L)
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")
```

```
X <- as.matrix(iris[, 1:4])
y <- iris$Species
mod <- wrap_ranger(X, y, num.trees = 100L)
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")
```

---

wrap\_svm

*S3 wrapper for e1071 SVM*


---

**Description**

Fits an ‘e1071’ support vector machine with a consistent interface. Supports classification and regression.

**Usage**

```
wrap_svm(x, y, ...)

## S3 method for class 'wrap_svm'
predict(object, newx, type = c("class", "prob"), ...)

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

**Arguments**

x	A matrix or data.frame of features.
y	A factor or character vector for classification, numeric for regression.
...	Additional arguments passed to [e1071::svm()]. ‘probability = TRUE’ is set automatically for classification; do not override this if you need ‘type = "prob"’ predictions.
object	A fitted ‘wrap_svm’ object.
newx	A matrix or data.frame of new observations.
type	“class” (default) for class labels, “prob” for a probability matrix. Ignored for regression.

**Value**

An object of class 'wrap\_svm' with fields:

<code>fit</code>	The fitted svm model.
<code>levels</code>	Class levels (NULL for regression).
<code>task</code>	"classification" or "regression".

**Examples**

```
X <- as.matrix(iris[, 1:4])
y <- iris$Species
mod <- wrap_svm(X, y, kernel = "radial")
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")
```

```
X <- as.matrix(iris[, 1:4])
y <- iris$Species
mod <- wrap_svm(X, y, kernel = "radial")
predict(mod, newx = X, type = "class")
predict(mod, newx = X, type = "prob")
```

# Index

`predict.wrap_caret`, 2  
`predict.wrap_glmnet (wrap_glmnet)`, 4  
`predict.wrap_lightgbm (wrap_lightgbm)`, 5  
`predict.wrap_ranger (wrap_ranger)`, 7  
`predict.wrap_svm (wrap_svm)`, 8  
`print.wrap_caret`, 3  
`print.wrap_glmnet (wrap_glmnet)`, 4  
`print.wrap_lightgbm (wrap_lightgbm)`, 5  
`print.wrap_ranger (wrap_ranger)`, 7  
`print.wrap_svm (wrap_svm)`, 8

`wrap_caret`, 3  
`wrap_glmnet`, 4  
`wrap_lightgbm`, 5  
`wrap_ranger`, 7  
`wrap_svm`, 8