

# Package ‘ensembleML’

June 5, 2026

**Type** Package

**Title** Unified Interface for Ensemble Machine Learning Methods

**Version** 0.2.5

**Date** 2026-05-20

**Description** Provides a clean, unified interface for training, predicting, and evaluating ensemble machine learning models including Random Forest, Gradient Boosting ('XGBoost'), 'AdaBoost', and 'Bagging'. All algorithms share a consistent API: `em_fit()`, `em_predict()`, `em_evaluate()`, and `em_tune()`. Includes built-in cross-validation, feature importance, calibration diagnostics, partial dependence plots, and model comparison utilities. Methods: Breiman (2001) <[doi:10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)>; Chen and Guestrin (2016) <[doi:10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785)>; Freund and Schapire (1997) <[doi:10.1006/jcss.1997.1504](https://doi.org/10.1006/jcss.1997.1504)>; Breiman (1996) <[doi:10.1007/BF00058655](https://doi.org/10.1007/BF00058655)>.

**Language** en-US

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1.0)

**Imports** randomForest (>= 4.7-1), xgboost (>= 1.7.0), adabag (>= 4.2), ggplot2 (>= 3.4.0), rlang (>= 1.1.0), stats, utils

**Suggests** pROC (>= 1.18.0), gridExtra (>= 2.3), testthat (>= 3.0.0), knitr, rmarkdown, mlbench

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Sadikul Islam [aut, cre] (ORCID: <<https://orcid.org/0000-0003-2924-7122>>)

**Maintainer** Sadikul Islam <sadikul.islamiasri@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-06-05 15:00:07 UTC

## Contents

ensembleML-package . . . . .	2
em_calibration . . . . .	3
em_compare . . . . .	4
em_confusion . . . . .	5
em_cv . . . . .	6
em_evaluate . . . . .	7
em_fit . . . . .	8
em_importance . . . . .	9
em_partial . . . . .	10
em_plot_cv . . . . .	11
em_predict . . . . .	11
em_residuals . . . . .	12
em_tune . . . . .	13
<b>Index</b>	<b>15</b>

---

ensembleML-package      *ensembleML: Unified Ensemble Machine Learning Interface*

---

## Description

A clean, consistent API for ensemble machine learning covering training, prediction, evaluation, tuning, diagnostics, and model comparison.

## Main functions

- `em_fit()`: Train any supported ensemble model
- `em_predict()`: Generate predictions or class probabilities
- `em_evaluate()`: Compute held-out performance metrics
- `em_cv()`: k-fold cross-validation for a single model
- `em_tune()`: Grid-search hyperparameter tuning via cross-validation
- `em_compare()`: Side-by-side comparison of multiple algorithms
- `em_importance()`: Feature importance extraction and visualisation
- `em_confusion()`: Styled confusion matrix (classification)
- `em_calibration()`: Calibration / reliability diagram (classification)
- `em_residuals()`: Residual diagnostics plot (regression)
- `em_partial()`: Partial dependence plot for one predictor

## References

- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. doi:10.1023/A:1010933404324
- Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. doi:10.1145/2939672.2939785
- Freund, Y. and Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), 119–139. doi:10.1006/jcss.1997.1504
- Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(2), 123–140. doi:10.1007/BF00058655

## Author(s)

**Maintainer:** Sadikul Islam <sadikul.islamiasri@gmail.com> ([ORCID](#))

---

em\_calibration

*Calibration (Reliability) Diagram*

---

## Description

Checks how well predicted class probabilities match observed frequencies. Binary classification only. A well-calibrated model lies on the diagonal.

## Usage

```
em_calibration(object, newdata, n_bins = 10L, positive = NULL)
```

## Arguments

object	An ensembleML_model with a binary classification task.
newdata	A data.frame including the true response column.
n_bins	Integer. Number of probability bins. Default 10.
positive	Character. The positive class. Defaults to the second level.

## Value

A data.frame of bin midpoints, mean predicted probability, and observed fraction (invisibly).

em\_compare

*Compare Multiple Ensemble Algorithms***Description**

Trains several ensemble algorithms on the same train/test split and returns a tidy comparison table plus an optional bar chart. Useful for algorithm selection before committing to hyperparameter tuning.

**Usage**

```
em_compare(
  formula,
  train,
  test,
  methods = NULL,
  task = NULL,
  metrics = NULL,
  sort_by = NULL,
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

**Arguments**

formula	A formula.
train	A data.frame for training.
test	A data.frame for evaluation (must share the same schema).
methods	Character vector of algorithms to compare. Defaults to all algorithms appropriate for the detected task.
task	"classification" or "regression". Auto-detected if NULL.
metrics	Metrics to compute (forwarded to <code>em_evaluate()</code> ).
sort_by	Character. Metric to sort by in the table. Defaults to the first computed metric.
plot	Logical. Print a bar chart? Default TRUE.
verbose	Logical. Print fitting progress messages? Default TRUE. Set to FALSE to suppress all console output.
...	Extra arguments forwarded to <code>em_fit()</code> for each algorithm.

**Value**

A list:

table data.frame of algorithms - metrics, sorted by sort\_by.

models Named list of fitted ensembleML\_model objects.  
 fit\_times Named numeric vector of training times (seconds).  
 plot A ggplot bar chart (if plot = TRUE).

### Examples

```
data(iris)
set.seed(42)
idx <- sample(nrow(iris), 120)
cmp <- em_compare(Species ~ ., train = iris[idx, ], test = iris[-idx, ])
cmp$table
```

---

em_confusion	<i>Confusion Matrix</i>
--------------	-------------------------

---

### Description

Computes and visualises a confusion matrix with per-class recall (sensitivity) on the diagonal. For classification tasks only.

### Usage

```
em_confusion(object, newdata, normalise = FALSE, plot = TRUE)
```

### Arguments

object	An ensembleML_model with task == "classification".
newdata	A data.frame with the true response column.
normalise	Logical. Show row-normalised proportions instead of raw counts? Default FALSE.
plot	Logical. Print a ggplot2 heatmap? Default TRUE.

### Value

The confusion matrix table (invisibly).

### Examples

```
data(iris)
set.seed(1)
idx <- sample(nrow(iris), 120)
m <- em_fit(Species ~ ., data = iris[idx, ], method = "random_forest")
em_confusion(m, iris[-idx, ])
```

em\_cv

*k-Fold Cross-Validation***Description**

Estimates generalisation performance of a model specification via repeated k-fold cross-validation. Returns fold-level metrics and aggregate statistics (mean - SD), helping you assess stability as well as average performance.

**Usage**

```
em_cv(
  formula,
  data,
  method = "random_forest",
  task = NULL,
  metrics = NULL,
  cv_folds = 5L,
  repeats = 1L,
  seed = 42L,
  verbose = TRUE,
  ...
)
```

**Arguments**

formula	A formula.
data	A data.frame.
method	Algorithm name (see <a href="#">em_fit()</a> ).
task	"classification" or "regression". Auto-detected if NULL.
metrics	Character vector of metrics. Defaults to task-appropriate set.
cv_folds	Integer. Number of folds. Default 5.
repeats	Integer. Number of complete CV repeats (increases stability of estimates). Default 1.
seed	Integer for reproducibility. Default 42.
verbose	Logical. Print fold progress? Default TRUE.
...	Extra arguments forwarded to <a href="#">em_fit()</a> .

**Value**

A list with:

summary data.frame of mean, SD, min, max per metric.  
 fold\_results data.frame of per-fold metric values.  
 cv\_folds Number of folds used.  
 repeats Number of repeats used.

**Examples**

```
data(iris)
cv_result <- em_cv(Species ~ ., data = iris, method = "random_forest",
                  cv_folds = 5, repeats = 3)
cv_result$summary
```

---

em\_evaluate

*Evaluate Model Performance*


---

**Description**

Compute held-out performance metrics for a fitted ensembleML\_model.

**Usage**

```
em_evaluate(object, newdata, metrics = NULL, positive = NULL)
```

**Arguments**

object	An ensembleML_model.
newdata	A data.frame containing the response column and all predictor columns used during training.
metrics	Character vector of metrics to compute. <b>Classification:</b> "accuracy", "kappa", "precision", "recall", "f1", "auc" (requires pROC). <b>Regression:</b> "rmse", "mae", "mape", "rsq", "adj_rsq". Defaults to all available metrics for the detected task.
positive	Character. Positive class for binary precision/recall/F1. Defaults to the second factor level (conventional for binary tasks).

**Value**

A named numeric vector of metric values.

**Examples**

```
data(iris)
set.seed(42)
idx <- sample(nrow(iris), 120)
m <- em_fit(Species ~ ., data = iris[idx, ], method = "random_forest")
em_evaluate(m, iris[-idx, ])
em_evaluate(m, iris[-idx, ], metrics = c("accuracy", "f1"))
```

em\_fit

*Fit an Ensemble Model***Description**

Unified entry point for training any supported ensemble algorithm. All fitted objects share a consistent structure enabling seamless use with `em_predict()`, `em_evaluate()`, `em_tune()`, and `em_compare()`.

**Usage**

```
em_fit(
  formula,
  data,
  method = "random_forest",
  task = NULL,
  weights = NULL,
  verbose = FALSE,
  ...
)
```

**Arguments**

formula	A formula describing the model (e.g. $y \sim .$ ).
data	A data.frame for training. Predictors must be free of NA; impute first with e.g. <b>mice</b> or <b>simputation</b> .
method	Character. One of "random_forest" (default), "xgboost", "adaboost", or "bagging".
task	"classification" or "regression". Auto-detected from the response type if NULL.
weights	Optional non-negative numeric vector of observation weights (length <code>nrow(data)</code> ).
verbose	Logical. Print a model summary after fitting? Default FALSE.
...	Algorithm-specific hyperparameters forwarded to the underlying engine (e.g. <code>ntree = 500</code> for Random Forest, <code>nrounds = 200</code> for XGBoost).

**Value**

An ensembleML\_model S3 object with named fields:

- model Raw fitted object from the underlying engine.
- method Algorithm name.
- task "classification" or "regression".
- formula The formula used.
- feature\_names Character vector of predictor names.

response\_name Name of the response variable.  
 levels Factor levels of the response (classification only; NULL otherwise).  
 n\_train Number of training rows.  
 call The original function call.  
 fit\_time Training wall-clock time (seconds).  
 train\_metrics In-sample metrics – use for sanity checks only.

## References

- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. doi:[10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)
- Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. doi:[10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785)
- Freund, Y. and Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), 119–139. doi:[10.1006/jcss.1997.1504](https://doi.org/10.1006/jcss.1997.1504)
- Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(2), 123–140. doi:[10.1007/BF00058655](https://doi.org/10.1007/BF00058655)

## See Also

[em\\_predict\(\)](#), [em\\_evaluate\(\)](#), [em\\_tune\(\)](#), [em\\_compare\(\)](#), [em\\_cv\(\)](#), [em\\_importance\(\)](#)

## Examples

```
data(iris)
m <- em_fit(Species ~ ., data = iris, method = "random_forest",
            verbose = TRUE)
summary(m)
```

---

em\_importance

*Feature Importance*

---

## Description

Extracts and optionally plots variable importance scores from a fitted ensemble model. The interpretation of importance varies by algorithm (e.g. mean decrease in impurity for Random Forest, gain for XGBoost).

## Usage

```
em_importance(
  object,
  top_n = NULL,
  plot = TRUE,
  normalise = TRUE,
  type = "MeanDecreaseGini"
)
```

**Arguments**

object	An ensembleML_model.
top_n	Integer. Return/show only the top-n features. NULL = all.
plot	Logical. Print a horizontal bar chart? Default TRUE.
normalise	Logical. Scale scores to sum to 100%? Default TRUE.
type	Character. Importance measure for Random Forest: "MeanDecreaseGini" (default) or "MeanDecreaseAccuracy". Ignored for other methods.

**Value**

A data.frame with columns feature and importance (invisibly when plot = TRUE).

**Examples**

```
data(iris)
m <- em_fit(Species ~ ., data = iris, method = "random_forest")
imp <- em_importance(m, top_n = 4)
```

---

em\_partial

*Partial Dependence Plot*


---

**Description**

Shows the marginal effect of a single predictor on the model output, averaging over the joint distribution of all other predictors. Helps understand non-linear effects and is model-agnostic.

**Usage**

```
em_partial(object, data, feature, n_grid = 30L, class = NULL)
```

**Arguments**

object	An ensembleML_model.
data	The training (or any reference) data.frame.
feature	Character. Name of the predictor to vary.
n_grid	Integer. Number of equally-spaced grid points for a numeric predictor. Default 30. Ignored for factors.
class	Character. For multi-class classification, which class probability to plot? Defaults to the first class level.

**Value**

A data.frame of grid values and mean predicted response (invisibly).

**Examples**

```
data(iris)
m <- em_fit(Species ~ ., data = iris, method = "random_forest")
em_partial(m, iris, feature = "Petal.Length")
```

em\_plot\_cv

*Plot Cross-Validation Fold Results***Description**

Visualise the distribution of a metric across all CV folds, with a reference line at the mean. Useful for assessing model stability.

**Usage**

```
em_plot_cv(cv_result, metric = NULL)
```

**Arguments**

cv\_result      Output from [em\\_cv\(\)](#).  
metric         Character. Metric to plot. Defaults to first available metric.

**Value**

A ggplot object (invisibly).

em\_predict

*Predict from an Ensemble Model***Description**

Generate predictions from a fitted ensembleML\_model. Output type is consistent regardless of the underlying algorithm.

**Usage**

```
em_predict(object, newdata, type = NULL, ...)
```

**Arguments**

object         An ensembleML\_model returned by [em\\_fit\(\)](#).  
newdata        A data.frame of new observations containing all training features.  
type            Character. "class" for predicted labels (classification), "prob" for class-probability matrix, or "response" for numeric fitted values (regression). Defaults to "class" / "response" based on the task.  
...             Currently unused.

**Value**

For type = "class": a factor. For type = "prob": a numeric matrix with one column per class.  
For regression: a numeric vector.

**Examples**

```
data(iris)
m <- em_fit(Species ~ ., data = iris, method = "random_forest")
preds <- em_predict(m, iris[1:10, ])
probs <- em_predict(m, iris[1:10, ], type = "prob")
```

---

em\_residuals

*Residual Diagnostics for Regression Models*

---

**Description**

Produces a 2-panel diagnostic plot: (1) residuals vs fitted values, and (2) a QQ plot of residuals.  
Useful for detecting heteroscedasticity and departures from normality.

**Usage**

```
em_residuals(object, newdata)
```

**Arguments**

object            An ensembleML\_model with task == "regression".  
newdata           A data.frame including the true response column.

**Value**

A data.frame of fitted values and residuals (invisibly).

**Examples**

```
set.seed(1)
d <- data.frame(x = rnorm(200), y = 3 + 2*rnorm(200) + rnorm(200))
m <- em_fit(y ~ x, data = d[1:160,], method = "random_forest")
em_residuals(m, d[161:200,])
```

**Description**

Performs an exhaustive grid search over a named list of hyperparameter values, using k-fold cross-validation to select the best configuration. After selection the best model is refit on the full dataset.

**Usage**

```
em_tune(
  formula,
  data,
  method = "random_forest",
  param_grid = list(),
  task = NULL,
  metric = NULL,
  cv_folds = 5L,
  seed = 42L,
  verbose = TRUE
)
```

**Arguments**

formula	A formula.
data	A data.frame.
method	Algorithm name (see <a href="#">em_fit()</a> ).
param_grid	A named list of hyperparameter vectors. All combinations are evaluated (Cartesian product). Pass an empty list for default params.
task	"classification" or "regression". Auto-detected if NULL.
metric	Optimisation criterion. Defaults to "accuracy" / "rmse".
cv_folds	Integer. Number of CV folds. Default 5.
seed	Integer. Random seed. Default 42.
verbose	Logical. Print progress? Default TRUE.

**Value**

A list with:

- best\_params Named list of the best hyperparameters found.
- best\_score Cross-validated score for the best configuration.
- metric The metric that was optimised.
- results data.frame of all configurations sorted by score.
- best\_model ensembleML\_model refit on the full dataset.

**Examples**

```
data(iris)
tuned <- em_tune(
  Species ~ ., data = iris, method = "random_forest",
  param_grid = list(ntree = c(100, 300), mtry = c(1, 2, 3))
)
tuned$best_params
tuned$best_score
tuned$results
```

# Index

em\_calibration, 3  
em\_calibration(), 2  
em\_compare, 4  
em\_compare(), 2, 8, 9  
em\_confusion, 5  
em\_confusion(), 2  
em\_cv, 6  
em\_cv(), 2, 9, 11  
em\_evaluate, 7  
em\_evaluate(), 2, 4, 8, 9  
em\_fit, 8  
em\_fit(), 2, 4, 6, 11, 13  
em\_importance, 9  
em\_importance(), 2, 9  
em\_partial, 10  
em\_partial(), 2  
em\_plot\_cv, 11  
em\_predict, 11  
em\_predict(), 2, 8, 9  
em\_residuals, 12  
em\_residuals(), 2  
em\_tune, 13  
em\_tune(), 2, 8, 9  
ensembleML (ensembleML-package), 2  
ensembleML-package, 2