Package 'T1FF'

September 15, 2026

Title Type-1 Fuzzy Functions for Classification, Regression, and Forecasting

Version 0.1.0

Description Fits Type-1 Fuzzy Function models for binary classification, numeric regression, and time-series forecasting with user-supplied temporal predictors. The package combines fuzzy C-means memberships, nonlinear membership transformations, cluster-specific linear or support vector machine models, and membership-weighted predictions. It also provides model evaluation, validation, K-fold and stratified K-fold tuning, and repeated nested cross-validation with task-appropriate metrics. The regression workflow can be used for forecasting when temporal dependence is represented by lagged or seasonal predictors and assessment partitions preserve chronological order.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports e1071, kernlab, stats, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Nihat Tak [aut, cre]

Maintainer Nihat Tak <nihattak@gmail.com>

Repository CRAN

Date/Publication 2026-09-15 10:50:33 UTC

Contents

`benchmark.T1FF` *Benchmark T1FF with nested cross-validation*

Description

Uses repeated outer cross-validation for unbiased performance estimation. Within every outer training partition, an inner cross-validation tunes the T1FF cluster count and fuzziness. A standard logistic or linear regression baseline is evaluated on the same outer partitions.

Usage

```
benchmark.T1FF(
  da = NULL,
  target_col = NULL,
  task = c("classification", "regression"),
  c_values = 2:5,
  m_values = c(1.5, 2, 2.5),
  tune_metric = NULL,
  metrics = NULL,
  outer_folds = 5,
  inner_folds = 5,
  repeats = 1,
  inner_repeats = 1,
  threshold = 0.5,
  positive_class = NULL,
  confidence = 0.95,
  seed = 123,
  verbose = TRUE,
  formula = NULL,
  data = NULL,
  na_action = c("fail", "omit"),
  logistic_method = c("auto", "glm", "ridge"),
  ridge_lambda = 0.01,
  probability_clip = 1e-06,
  ...
)
```

Arguments

da	A data frame containing predictors and outcome, or a formula.
target_col	Outcome column name, or the data frame with the positional formula interface.
task	Either "classification" or "regression".
c_values	Candidate cluster counts used by inner tuning.
m_values	Candidate fuzziness values used by inner tuning.
tune_metric	Metric optimized in the inner loop. Defaults to logloss for classification and rmse for regression.
metrics	Metrics reported on outer test partitions. Classification supports logloss, brier, roc_auc, pr_auc, accuracy, balanced_accuracy, f1, sensitivity, and specificity. Regression supports rmse, mse, mae, mape, smape, and r2. MAPE excludes zero actual values and both percentage metrics are reported on a 0–100 percentage scale (SMAPE can range up to 200).
outer_folds, inner_folds	Numbers of outer and inner folds.
repeats	Number of repeated outer fold assignments.
inner_repeats	Number of repeated inner fold assignments.
threshold	Classification threshold.
positive_class	Value identifying the positive class.
confidence	Confidence level for mean performance intervals.
seed	Random seed.
verbose	Whether to report outer-fold progress.
formula	Optional model formula.
data	Optional data frame used with formula.
na_action	Either "fail" or "omit".
logistic_method, ridge_lambda, probability_clip	Classification stability controls passed through inner tuning to T1FF().
...	Additional arguments passed to T1FF fitting during tuning.

Details

For each repeated outer split, the outer training observations alone are passed to tune.T1FF(). The selected T1FF configuration is refitted on that outer training partition and evaluated on the untouched outer test partition. The baseline is fitted and tested on exactly the same partition, providing paired fold-level comparisons and avoiding use of outer test data during hyperparameter selection.

If q_{rj} is a test score from repeat r and outer fold j, the reported performance mean is

$$\bar{q} = \frac{1}{N} \sum_{r,j} q_{rj}.$$

The summary table also reports the sample standard deviation, standard error $SE = s/\sqrt{N}$, and a two-sided Student-t confidence interval

$$\bar{q} \pm t_{1-\alpha/2,N-1} SE.$$

Classification intervals for metrics bounded by zero and one are truncated to that range; nonnegative loss intervals are truncated at zero. Fold-level scores, selected inner-loop parameters, and elapsed fitting times remain available in the returned object for further analysis.

Outer and inner folds are randomly assigned. Consequently, this function is not a rolling-origin benchmark and should not be used as evidence of time-aware forecasting performance. Forecasting studies should construct chronological or rolling-origin resamples externally.

Value

An object of class `t1ff_benchmark`. `fold_results` stores each outer test score, `summary` stores means and confidence intervals, `selected_parameters` records inner-loop choices, and `timings` and `timing_summary` report elapsed computation time.

Examples

```
data(iris)
d <- droplevels(subset(iris, Species != "setosa"))
bench <- benchmark.T1FF(
  d, "Species", c_values = 2, m_values = 2,
  outer_folds = 2, inner_folds = 2,
  positive_class = "virginica", seed = 1, verbose = FALSE
)
summary(bench)
```

---

evaluate.T1FF *Evaluate a fitted T1FF model*

Description

Computes task-appropriate out-of-sample performance measures for a fitted model or the final model stored by `tune.T1FF()`. Missing truth or prediction pairs are omitted and reported in the returned object.

Usage

```
evaluate.T1FF(object, data, truth, threshold = NULL)
```

Arguments

`object`	A fitted `t1ff` or `t1ff_tuning` object.
`data`	An independent test data frame containing the predictor columns. When `truth` is supplied as a column name, `data` must also contain that observed outcome column.
`truth`	Name of the observed outcome column in `data`, or an outcome vector with one value per row of `data`.
`threshold`	Classification threshold strictly between zero and one. If `NULL`, a tuned object's selected threshold is used; otherwise the default is 0.5.

Details

`evaluate.T1FF()` is intended for an independent test set and does not use a separate prediction algorithm. It first calls the model's `predict.t1ff()` method internally (`type = "prob"` for classification and `type = "response"` for regression), then compares those predictions with `truth` and calculates the relevant metrics. Call `predict.t1ff()` directly when only predictions are needed; use `evaluate.T1FF()` when the outcome values are available and performance must be assessed. In forecasting applications, `data` should contain observations later in time than the fitting data, together with the observed outcomes used for evaluation.

Value

An object of class `t1ff_evaluation`. Classification evaluations contain a metric table and confusion matrix. Regression evaluations also contain predictions and residuals. Common components include `metrics`, `actual`, `predicted`, `n_used`, and `n_omitted`.

Examples

```
data(iris)
d <- droplevels(subset(iris, Species != "setosa"))
fit <- T1FF(d, "Species", c = 2,
  positive_class = "virginica", seed = 1)
evaluate.T1FF(fit, d, truth = "Species")
```

---

predict.t1ff *Predict from a T1FF model*

Description

Predict from a T1FF model

Usage

```
## S3 method for class 't1ff'
predict(object, newdata, type = NULL, threshold = 0.5, ...)
```

Arguments

`object`	A fitted `t1ff` model.
`newdata`	A data frame containing the predictor columns.
`type`	Prediction type. Classification supports `"class"`, `"prob"`, `"cluster_prob"`, and `"membership"`. Regression supports `"response"`, `"cluster_response"`, and `"membership"`.
`threshold`	Classification threshold for the positive class.
`...`	Reserved for future use.

Details

Prediction reuses the preprocessing recipe, feature scaling, fuzzy cluster centers, fuzziness value, and local learners estimated during training; none of these quantities is re-estimated from `newdata`. For each new observation, memberships are calculated relative to the stored centers, the three membership features are reconstructed for every cluster, and each local learner produces a cluster-specific prediction. Except for the diagnostic matrix types, the returned prediction is

$$\widehat{f}(x) = \sum_{k=1}^{c} \mu_k(x) \widehat{f}_k(x).$$

For classification, $\widehat{f}_k(x)$ is a positive-class probability; the combined probability is clipped to the fitted model's probability bounds before `threshold` is applied. `"membership"`, `"cluster_prob"`, and `"cluster_response"` expose the intermediate quantities without changing the fitted object. When the model was fitted with `na_action = "omit"`, rows with incomplete predictors are restored as missing predictions in their original positions.

Value

A vector or matrix, depending on `type`. For classification, `"prob"` is the positive-class probability, `"class"` is the thresholded class, `"cluster_prob"` contains one column per local classifier, and `"membership"` contains fuzzy memberships. Regression uses `"response"`, `"cluster_response"`, and `"membership"`.

`predict.t1ff_tuning` *Predict using the best tuned T1FF model*

Description

Predict using the best tuned T1FF model

Usage

```
## S3 method for class 't1ff_tuning'
predict(object, newdata, ...)
```

Arguments

`object`	A `t1ff_tuning` object containing a final model.
`newdata`	A data frame containing predictors.
`...`	Additional arguments passed to the fitted model's prediction method.

Details

This method is a convenience wrapper around `predict.t1ff()`. It requires `fit_final = TRUE` during `tune.T1FF()` so that `best_model` exists, then forwards `newdata` and all prediction arguments to that model. For class predictions, the cross-validated `best_threshold` is used automatically when the caller does not supply `threshold`; an explicitly supplied threshold takes precedence. Probability, membership, and cluster-level prediction types retain the behavior of `predict.t1ff()`.

Value

The prediction produced by the stored best model.

`print.t1ff` *Print a T1FF model*

Description

Print a T1FF model

Usage

```
## S3 method for class 't1ff'
print(x, ...)
```

Arguments

`x`	A fitted `t1ff` model.
`...`	Additional arguments passed to the model summary method.

Details

`print.t1ff()` is the automatic console display method for a fitted `t1ff` object. It delegates to `summary.t1ff()` and therefore presents the same structural and numerical diagnostics without modifying the model.

Value

`x`, invisibly.

`print.t1ff_benchmark` *Print or summarize a T1FF benchmark*

Description

Print or summarize a T1FF benchmark

Usage

```
## S3 method for class 't1ff_benchmark'
print(x, ...)

## S3 method for class 't1ff_benchmark'
summary(object, ...)
```

Arguments

x, object	A t1ff_benchmark object.
...	Reserved for future use.

Details

These methods display the benchmark design and the aggregated table already stored in object$summary. They do not rerun the outer folds, inner tuning, baselines, or confidence-interval calculations. Detailed fold scores, chosen hyperparameters, and timing results can be inspected separately in fold_results, selected_parameters, timings, and timing_summary. summary() delegates to the print method, and both return the benchmark object invisibly.

Value

The benchmark object, invisibly.

---

print.t1ff_evaluation *Print or summarize a T1FF evaluation*

Description

Print or summarize a T1FF evaluation

Usage

```
## S3 method for class 't1ff_evaluation'
print(x, ...)

## S3 method for class 't1ff_evaluation'
summary(object, ...)
```

Arguments

x, object	A t1ff_evaluation object.
...	Reserved for future use.

Details

Both methods display results already computed by evaluate.T1FF(); they do not call predict() again. Classification output includes the positive class, applied threshold, confusion matrix, and metric table. Regression output includes the metric table. summary() delegates to the print method, and both methods return the evaluation object invisibly.

Value

The evaluation object, invisibly.

print.t1ff_tuning *Print T1FF tuning results*

Description

Print T1FF tuning results

Usage

```
## S3 method for class 't1ff_tuning'
print(x, ...)
```

Arguments

x	A t1ff_tuning object.
...	Additional arguments passed to the tuning summary method.

Details

print.t1ff_tuning() is the automatic console method for tuning objects. It delegates to summary.t1ff_tuning() using that method's default display size and returns the original object invisibly. No cross-validation or final-model fitting is repeated.

Value

x, invisibly.

summary.t1ff *Summarize a T1FF model*

Description

Summarize a T1FF model

Usage

```
## S3 method for class 't1ff'
summary(object, ...)
```

Arguments

object	A fitted t1ff model.
...	Reserved for future use.

Details

This method reports the fitted model structure rather than recalculating predictions or goodness-of-fit statistics. For classification, it includes the class orientation, local fitting engines, convergence, detected separation, captured GLM warnings, and probability clipping rule. It also reports the cluster count, fuzziness, scaling choice, predictors, omitted training rows, and number of local models. The fitted object is returned invisibly so the call can be assigned or used in a pipeline.

Value

`object`, invisibly.

`summary.t1ff_tuning` *Summarize T1FF tuning results*

Description

Summarize T1FF tuning results

Usage

```
## S3 method for class 't1ff_tuning'
summary(object, top = 10L, ...)
```

Arguments

`object`	A `t1ff_tuning` object.
`top`	Number of leading configurations to print.
`...`	Reserved for future use.

Details

The method identifies whether the tuning metric was minimized or maximized, reports the resampling design and selected c, m, and classification threshold, and prints up to `top` candidate rows. Candidate rows are ordered from best to worst according to the optimization direction; the complete, untruncated grid remains available in `object$results`. Calling this method does not refit any candidate model.

Value

`object`, invisibly.

T1FF *Fit a Type-1 Fuzzy Function model*

Description

Fits a binary classification or numeric regression model by fuzzy C-means clustering, cluster-specific statistical models, and membership-weighted aggregation.

Usage

```
T1FF(
  da = NULL,
  target_col = NULL,
  c,
  m = 2,
  task = c("classification", "regression"),
  positive_class = NULL,
  scale_features = TRUE,
  cluster_method = "fcm",
  eps = 1e-10,
  seed = NULL,
  formula = NULL,
  data = NULL,
  na_action = c("fail", "omit"),
  local_model = c("linear", "svm"),
  logistic_method = c("auto", "glm", "ridge"),
  ridge_lambda = 0.01,
  probability_clip = 1e-06,
  svm_kernel = "rbfdot",
  svm_C = 1,
  svm_sigma = NULL,
  svm_epsilon = 0.1
)
```

Arguments

da	A data frame containing predictors and the target, or a formula when using the positional formula interface.
target_col	Name of the target column, or the data frame when da is supplied as a formula.
c	Integer number of fuzzy clusters.
m	Fuzziness parameter greater than 1.
task	Either "classification" or "regression".
positive_class	Value identifying the positive class for binary classification. If NULL, the second observed class level is used.
scale_features	Whether to standardize predictors using training-data means and standard deviations.

`cluster_method` Clustering method. The current release implements only `"fcm"` (fuzzy C-means).

`eps` Positive numerical safeguard used in the logarithmic membership transformation.

`seed` Optional random seed used by fuzzy C-means.

`formula` Optional model formula with a response.

`data` Optional data frame used with `formula`.

`na_action` Either `"fail"` or `"omit"`.

`local_model` Cluster-specific learner. `"linear"` uses logistic regression for classification and ordinary least squares for regression; `"svm"` uses a probabilistic SVM or epsilon-SVR.

`logistic_method`
Classification fitting strategy: `"auto"` detects unstable local GLMs and refits them by ridge logistic regression while retaining preliminary warnings in the fitted object; `"glm"` always uses ordinary logistic regression; `"ridge"` always uses ridge.

`ridge_lambda` Positive ridge penalty used by local ridge-logistic models.

`probability_clip`
Lower probability bound; predictions are constrained to this value and one minus this value.

`svm_kernel`, `svm_C`, `svm_sigma`, `svm_epsilon`
Local SVM controls. A `NULL` `svm_sigma` is estimated separately within each local model.

Details

Predictor preprocessing is learned from the training data. Fuzzy C-means memberships and their logarithmic and exponential transformations enter each cluster-specific response model. Final predictions are membership- weighted averages of local predictions. Classification currently requires exactly two outcome classes. With `logistic_method = "auto"`, unstable local logistic models are automatically replaced by ridge-logistic models. Warnings from preliminary GLM fits are retained in the model object and are displayed only if a numerical problem remains unresolved.

For an observation x_i, let v_k denote fuzzy-cluster center k, $d_{ik} = ||x_i - v_k||_2$ its Euclidean distance to that center, and $m > 1$ the fuzziness parameter. Unless an observation coincides with one or more centers, its membership in cluster k is

$$\mu_{ik} = \frac{d_{ik}^{-2/(m-1)}}{\sum_{\ell=1}^{c} d_{i\ell}^{-2/(m-1)}}.$$

If x_i coincides with one or more centers, its membership is split equally across those centers. Thus, for every observation, $\sum_{k=1}^{c} \mu_{ik} = 1$.

A separate local learner is fitted for each cluster. Its design row is

$$z_{ik} = \left(\mu_{ik},\ \log\{\max(\mu_{ik}, \epsilon)\},\ \exp(\mu_{ik}),\ x_i^{\top}\right)^{\top},$$

where ϵ prevents evaluating $\log(0)$. With `local_model = "linear"`, the local learner is ordinary least squares for regression and logistic regression for binary classification. The latter yields a

local positive-class probability $p_k(x_i)$. With `local_model = "svm"`, the corresponding probabilistic SVM or epsilon-SVR supplies the local prediction instead.

The final regression prediction and binary positive-class probability are, respectively,

$$\widehat{y}(x_i) = \sum_{k=1}^{c} \mu_{ik} \widehat{y}_k(x_i)$$

and

$$\widehat{p}(x_i) = \operatorname{clip}\left\{ \sum_{k=1}^{c} \mu_{ik} p_k(x_i),\ \delta,\ 1-\delta \right\},$$

where δ is `probability_clip`. A class label is then obtained by comparing $\widehat{p}(x_i)$ with the selected classification threshold.

For forecasting, temporal dependence must be encoded explicitly in the predictors, such as through response lags, seasonal indicators, and trend terms. Rows should be divided chronologically so that every validation or test observation occurs after the observations used for fitting. `T1FF()` does not infer time order, generate lagged values, or recursively update multi-step forecasts; those operations remain part of data preparation.

Value

An object of class `t1ff`. For linear classification models, `local_glm_warnings` stores warnings from each preliminary local GLM and `unresolved_warnings` records issues that are also reported to the user.

Examples

```
data(iris)
d <- droplevels(subset(iris, Species != "setosa"))
fit <- T1FF(d, "Species", c = 2,
  task = "classification", positive_class = "virginica", seed = 1)
predict(fit, d[1:3, ], type = "prob")
fit_formula <- T1FF(Species ~ ., d, c = 2,
  task = "classification", positive_class = "virginica", seed = 1)
```

`tune.T1FF` *Tune a Type-1 Fuzzy Function model*

Description

Evaluates combinations of cluster counts and fuzziness parameters using a validation split, K-fold cross-validation, or stratified K-fold cross-validation. Stratification is available for classification only.

Usage

```
tune.T1FF(
  da = NULL,
  target_col = NULL,
  task = c("classification", "regression"),
  c_values = 2:10,
  m_values = seq(1.2, 3, by = 0.2),
  metric = NULL,
  resampling = NULL,
  folds = 5,
  repeats = 1,
  nvalid = NULL,
  threshold = 0.5,
  threshold_values = NULL,
  positive_class = NULL,
  scale_features = TRUE,
  seed = 123,
  verbose = TRUE,
  fit_final = TRUE,
  formula = NULL,
  data = NULL,
  na_action = c("fail", "omit"),
  local_model = c("linear", "svm"),
  logistic_method = c("auto", "glm", "ridge"),
  ridge_lambda = 0.01,
  probability_clip = 1e-06,
  svm_kernel = "rbfdot",
  svm_C = 1,
  svm_sigma = NULL,
  svm_epsilon = 0.1,
  ...
)
```

Arguments

da	A data frame containing predictors and the target, or a formula when using the positional formula interface.
target_col	Name of the target column, or the data frame when da is supplied as a formula.
task	Either "classification" or "regression".
c_values	Candidate cluster counts.
m_values	Candidate fuzziness values greater than 1.
metric	Performance metric. Classification supports logloss, brier, roc_auc, pr_auc, accuracy, balanced_accuracy, f1, sensitivity, and specificity. Regression supports rmse, mse, mae, mape, smape, and r2. MAPE excludes observations whose actual value is zero; it is NA if all actual values are zero. SMAPE assigns zero error when both the actual and predicted values are zero. Both are reported as percentages.

`resampling`	One of `"validation"`, `"kfold"`, or `"stratified_kfold"`.
`folds`	Number of folds.
`repeats`	Number of repeated fold assignments.
`nvalid`	Validation size when `resampling = "validation"`.
`threshold`	Classification threshold for threshold-dependent metrics.
`threshold_values`	Optional candidate classification thresholds. When more than one value is supplied, `metric` must be threshold-dependent. Fold probabilities are computed once per `c` and `m` combination and then reused across thresholds.
`positive_class`	Value identifying the positive class.
`scale_features`	Whether to standardize predictors within training data.
`seed`	Random seed.
`verbose`	Whether to print progress.
`fit_final`	Whether to refit the best model on all observations.
`formula`	Optional model formula with a response.
`data`	Optional data frame used with `formula`.
`na_action`	Either `"fail"` or `"omit"`.
`local_model`, `logistic_method`, `ridge_lambda`, `probability_clip`	Cluster learner and classification stability controls passed to `T1FF()`.
`svm_kernel`, `svm_C`, `svm_sigma`, `svm_epsilon`	Local SVM controls passed to `T1FF()`.
`...`	Additional arguments passed to the internal fitting routine.

Details

The preprocessing recipe is re-estimated within every training fold. After selection, `fit_final = TRUE` refits the best configuration on all supplied observations and stores it in `best_model`.

Let $\mathcal{G} = \mathcal{C} \times \mathcal{M}$ be the candidate grid formed by `c_values` and `m_values`, and let $\mathcal{S}$ denote the validation splits. For every candidate $g = (c, m) \in \mathcal{G}$ and split s, T1FF is fitted only on the training partition D_s^{train}. Scaling, factor encoding, fuzzy centers, and local learners are all estimated anew within that partition. Predictions on D_s^{valid} produce a split score

$$q_s(g) = \text{Metric}\left(y_s^{valid}, \widehat{y}_s^{valid}(g)\right).$$

The reported `mean_score` is

$$\overline{q}(g) = \frac{1}{|\mathcal{S}_g|} \sum_{s \in \mathcal{S}_g} q_s(g),$$

where $\mathcal{S}_g$ contains the splits that fitted successfully. The selected configuration is

$$g^* = \underset{g \in \mathcal{G}}{\text{argmin}}\ \overline{q}(g)$$

for loss metrics (for example, RMSE or log-loss), and

$$g^* = \underset{g \in \mathcal{G}}{\text{argmax}}\ \overline{q}(g)$$

for utility metrics (for example, AUC, accuracy, or R^2).

For threshold-dependent classification metrics, `threshold_values` extends the candidate set to $\mathcal{G} \times \mathcal{T}$. The same fold probabilities are reused for each threshold $t \in \mathcal{T}$, and the selected triple is (c^*, m^*, t^*). If `fit_final = TRUE`, the resulting (c^*, m^*) model is finally refitted once using all supplied training observations.

The built-in `"validation"`, `"kfold"`, and `"stratified_kfold"` splits are randomly assigned and do not preserve time order. For forecasting, select hyperparameters with external chronological validation or rolling-origin resampling, then fit the selected configuration with `T1FF()`. Random-fold tuning should not be reported as time-aware forecasting validation.

Value

An object of class `t1ff_tuning` containing the complete grid in `results`, selected values in `best_c` and `best_m`, the selected score in `best_score`, fold definitions in `splits`, and optionally `best_model`.

Examples

```
data(iris)
d <- droplevels(subset(iris, Species != "setosa"))
tuned <- tune.T1FF(d, "Species", task = "classification",
  c_values = 2, m_values = 2, metric = "logloss",
  resampling = "stratified_kfold", folds = 2,
  positive_class = "virginica", seed = 1, verbose = FALSE)
predict(tuned, d[1:3, ], type = "prob")
```

Index