Package ‘caretForecast’

January 30, 2026

Title Conformal Time Series Forecasting Using State of Art Machine Learning Algorithms

Version 0.1.2

Description Conformal time series forecasting using the caret infrastructure. It provides access to state-of-the-art machine learning models for forecasting applications. The hyperparameter of each model is selected based on time series cross-validation, and forecasting is done recursively.

License GPL (>= 3)

URL https://akai01.github.io/caretForecast/, https://github.com/Akai01/caretForecast

BugReports https://github.com/Akai01/caretForecast/issues

Depends R (>= 3.6)

Imports forecast (>= 8.15), caret (>= 6.0.88), magrittr (>= 2.0.1), methods (>= 4.1.1), dplyr (>= 1.0.9), generics (>= 0.1.3)

Suggests Cubist (>= 0.3.0), knitr (>= 1.29), testthat (>= 2.3.2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Resul Akay [aut, cre]

Maintainer Resul Akay <resulakay1@gmail.com>

Repository CRAN

Date/Publication 2026-01-30 06:10:38 UTC

Contents

ARml *Autoregressive forecasting using various Machine Learning models.*

Description

Autoregressive forecasting using various Machine Learning models.

Usage

```
ARml(
  y,
  max_lag = 5,
  xreg = NULL,
  caret_method = "cubist",
  metric = "RMSE",
  pre_process = NULL,
  cv = TRUE,
  cv_horizon = 4,
  initial_window = NULL,
  fixed_window = FALSE,
  verbose = TRUE,
  seasonal = TRUE,
  K = frequency(y)/2,
  tune_grid = NULL,
  lambda = NULL,
  BoxCox_method = c("guerrero", "loglik"),
  BoxCox_lower = -1,
  BoxCox_upper = 2,
  BoxCox_biasadj = FALSE,
  BoxCox_fvar = NULL,
  allow_parallel = FALSE,
  calibrate = TRUE,
  calibration_horizon = NULL,
  n_cal_windows = NULL,
  ...
)
```

Arguments

`y` A univariate time series object.

`max_lag` Maximum value of lag.

`xreg` Optional. A numerical vector or matrix of external regressors, which must have the same number of rows as y. (It should not be a data frame.).

`caret_method` A string specifying which classification or regression model to use. Possible values are found using names(getModelInfo()). A list of functions can also be passed for a custom model function. See `https://topepo.github.io/caret/` for details.

`metric` A string that specifies what summary metric will be used to select the optimal model. See `?caret::train`.

`pre_process` A string vector that defines a pre-processing of the predictor data. Current possibilities are "BoxCox", "YeoJohnson", "expoTrans", "center", "scale", "range", "knnImpute", "bagImpute", "medianImpute", "pca", "ica" and "spatialSign". The default is no pre-processing. See preProcess and trainControl on the procedures and how to adjust them. Pre-processing code is only designed to work when x is a simple matrix or data frame.

`cv` Logical, if `cv = TRUE` model selection will be done via cross-validation. If `cv = FALSE` user need to provide a specific model via `tune_grid` argument.

`cv_horizon` The number of consecutive values in test set sample.

`initial_window` The initial number of consecutive values in each training set sample.

`fixed_window` Logical, if FALSE, all training samples start at 1.

`verbose` A logical for printing a training log.

`seasonal` Boolean. If `seasonal = TRUE` the fourier terms will be used for modeling seasonality.

`K` Maximum order(s) of Fourier terms

`tune_grid` A data frame with possible tuning values. The columns are named the same as the tuning parameters. Use getModelInfo to get a list of tuning parameters for each model or see `https://topepo.github.io/caret/available-models.html`. (NOTE: If given, this argument must be named.)

`lambda` BoxCox transformation parameter. If `lambda = NULL` If `lambda = "auto"`, then the transformation parameter lambda is chosen using `BoxCox.lambda`.

`BoxCox_method` `BoxCox.lambda` argument. Choose method to be used in calculating lambda.

`BoxCox_lower` `BoxCox.lambda` argument. Lower limit for possible lambda values.

`BoxCox_upper` `BoxCox.lambda` argument. Upper limit for possible lambda values.

`BoxCox_biasadj` `InvBoxCox` argument. Use adjusted back-transformed mean for Box-Cox transformations. If transformed data is used to produce forecasts and fitted values, a regular back transformation will result in median forecasts. If biasadj is TRUE, an adjustment will be made to produce mean forecasts and fitted values.

`BoxCox_fvar` `InvBoxCox` argument. Optional parameter required if biasadj=TRUE. Can either be the forecast variance, or a list containing the interval level, and the corresponding upper and lower intervals.

allow_parallel If a parallel backend is loaded and available, should the function use it?

calibrate Logical. If TRUE, performs rolling-origin calibration to compute horizon-specific conformal prediction intervals. This produces properly calibrated intervals that widen with forecast horizon (trumpet shape). Default is TRUE.

calibration_horizon
Maximum forecast horizon for calibration. If NULL (default), uses 2 * frequency(y) for seasonal data or 10 for non-seasonal data.

n_cal_windows Number of rolling windows for calibration. If NULL (default), automatically determined based on data length (max 50).

... Ignored.

Value

A list class of forecast containing the following elemets

- x : The input time series
- method : The name of the forecasting method as a character string
- mean : Point forecasts as a time series
- lower : Lower limits for prediction intervals
- upper : Upper limits for prediction intervals
- level : The confidence values associated with the prediction intervals
- model : A list containing information about the fitted model
- newx : A matrix containing regressors
- calibration : Horizon-specific conformal calibration scores (if calibrate=TRUE)

Author(s)

Resul Akay

Examples

```
library(caretForecast)

train_data <- window(AirPassengers, end = c(1959, 12))

test <- window(AirPassengers, start = c(1960, 1))

ARml(train_data, caret_method = "lm", max_lag = 12) -> fit

forecast(fit, h = length(test)) -> fc

autoplot(fc) + autolayer(test)

accuracy(fc, test)
```

conformalRegressor *Fit a conformal regressor.*

Description

Fit a conformal regressor.

Usage

```
conformalRegressor(residuals, sigmas = NULL)
```

Arguments

`residuals`	Model residuals.
`sigmas`	A vector of difficulty estimates

Value

A conformalRegressor object

Author(s)

Resul Akay

References

Boström, H., 2022. crepes: a Python Package for Generating Conformal Regressors and Predictive Systems. In Conformal and Probabilistic Prediction and Applications. PMLR, 179.

conformalRegressorByHorizon *Fit a horizon-specific conformal regressor for time series forecasting.*

Description

This function creates a conformal regressor that accounts for increasing uncertainty at longer forecast horizons. It uses separate nonconformity score distributions for each horizon h=1,2,3,..., resulting in prediction intervals that naturally widen as the forecast horizon increases (trumpet-shaped intervals).

Usage

```
conformalRegressorByHorizon(horizon_errors)
```

Arguments

`horizon_errors` A named list where each element contains sorted absolute errors for that horizon. Names should be "h1", "h2", etc. This is typically produced by `calibrate_horizon_scores()`.

Value

A conformalRegressorByHorizon object containing:

`alphas_by_horizon` List of sorted nonconformity scores for each horizon

`max_horizon` Maximum calibrated horizon

`n_samples` Number of calibration samples per horizon

Author(s)

Resul Akay

References

Boström, H., 2022. crepes: a Python Package for Generating Conformal Regressors and Predictive Systems. In Conformal and Probabilistic Prediction and Applications. PMLR, 179.

Stankeviciute, K., Alaa, A. M., & van der Schaar, M., 2021. Conformal Time-series Forecasting. NeurIPS 2021.

`forecast.ARml` *Forecasting using ARml model*

Description

Forecasting using ARml model

Usage

```
## S3 method for class 'ARml'
forecast(object, h = frequency(object$y), xreg = NULL, level = c(80, 95), ...)
```

Arguments

`object` An object of class "ARml", the result of a call to ARml.

`h` forecast horizon

`xreg` Optionally, a numerical vector or matrix of future external regressors

`level` Confidence level for prediction intervals.

`...` Ignored

Value

A list class of forecast containing the following elemets

- x : The input time series
- method : The name of the forecasting method as a character string
- mean : Point forecasts as a time series
- lower : Lower limits for prediction intervals
- upper : Upper limits for prediction intervals
- level : The confidence values associated with the prediction intervals
- model : A list containing information about the fitted model
- newxreg : A matrix containing regressors

Author(s)

Resul Akay

Examples

```
library(caretForecast)

train_data <- window(AirPassengers, end = c(1959, 12))

test <- window(AirPassengers, start = c(1960, 1))

ARml(train_data, caret_method = "lm", max_lag = 12) -> fit

forecast(fit, h = length(test), level = c(80,95)) -> fc

autoplot(fc)+ autolayer(test)

accuracy(fc, test)
```

`get_var_imp` *Variable importance for forecasting model.*

Description

Variable importance for forecasting model.

Usage

```
get_var_imp(object, plot = TRUE)
```

Arguments

`object`	A list class of ARml or forecast object derived from ARml
`plot`	Boolean, if TRUE, variable importance will be ploted.

Value

A list class of "varImp.train". See `varImp` or a "trellis" plot.

Author(s)

Resul Akay

Examples

```
train <- window(AirPassengers, end = c(1959, 12))

test <- window(AirPassengers, start = c(1960, 1))

ARml(train, caret_method = "lm", max_lag = 12, trend_method = "none",
 pre_process = "center") -> fit

forecast(fit, h = length(test), level = c(80,95)) -> fc

autoplot(fc)+ autolayer(test)

accuracy(fc, test)

get_var_imp(fc, plot = TRUE)
```

---

predict.conformalRegressor *Predict a conformalRegressor*

Description

Predict a conformalRegressor

Usage

```
## S3 method for class 'conformalRegressor'
predict(
  object,
  y_hat = NULL,
  sigmas = NULL,
  confidence = 0.95,
  y_min = -Inf,
  y_max = Inf,
  ...
)
```

Arguments

`object`	A conformalRegressor object
`y_hat`	Predicted values
`sigmas`	Difficulty estimates
`confidence`	Confidence level
`y_min`	The minimum value to include in prediction intervals
`y_max`	The maximum value to include in prediction intervals
`...`	Ignored

Value

Prediction intervals

Author(s)

Resul Akay

`predict.conformalRegressorByHorizon` *Predict intervals from a horizon-specific conformal regressor*

Description

This function generates prediction intervals that account for increasing uncertainty at longer forecast horizons. Each horizon h uses its own calibrated nonconformity score distribution, resulting in trumpet-shaped prediction intervals.

Usage

```
## S3 method for class 'conformalRegressorByHorizon'
predict(
  object,
  y_hat = NULL,
  confidence = 0.95,
  y_min = -Inf,
  y_max = Inf,
  ...
)
```

Arguments

`object`	A conformalRegressorByHorizon object
`y_hat`	Predicted values (one per horizon)
`confidence`	Confidence level(s) between 0 and 1 (e.g., 0.95 for 95 percent)
`y_min`	The minimum value to include in prediction intervals
`y_max`	The maximum value to include in prediction intervals
`...`	Ignored

Value

A data frame with lower and upper bounds for each confidence level

Author(s)

Resul Akay

`retail` *Grouped sales data from an Australian Retailer*

Description

A dataset containing 42 products' sales

Usage

```
retail
```

Format

A data class of "tbl_df", "tbl", "data.frame" with 13986 rows and 3 columns:

date date

item products

value sales

Source

https://robjhyndman.com/data/ausretail.csv

`retail_wide` *Sales data from an Australian Retailer in time series format*

Description

A dataset containing 42 products' sales

Usage

```
retail_wide
```

Format

An object of class mts (inherits from ts, matrix) with 333 rows and 43 columns.

This data set is the wide format of `retail` data.

Source

https://robjhyndman.com/data/ausretail.csv

`split_ts` *Split a time series into training and testing sets*

Description

Split a time series into training and testing sets

Usage

```
split_ts(y, test_size = 10)
```

Arguments

`y`	A univariate time series
`test_size`	The number of observations to keep in the test set

Value

A list with train and test elements

Author(s)

Resul Akay

Examples

```
dlist <- split_ts(retail_wide[,1], test_size = 12)
```

`suggested_methods` *Suggested methods for ARml*

Description

Suggested methods for ARml

Usage

```
suggested_methods()
```

Value

A character vector of Suggested methods

Author(s)

Resul Akay

Examples

```
suggested_methods()
```

Index