Package: ReSurv (via r-universe)

November 15, 2024

Type Package

Title Machine Learning Models for Predicting Claim Counts

Version 1.0.0

Description Prediction of claim counts using the feature based development factors introduced in the manuscript Hiabu M., Hofman E. and Pittarello G. (2023) <doi:10.48550/arXiv.2312.14549>. Implementation of Neural Networks, Extreme Gradient Boosting, and Cox model with splines to optimise the partial log-likelihood of proportional hazard models.

URL https://github.com/edhofman/ReSurv

BugReports https://github.com/edhofman/ReSurv/issues

License GPL (>= 2)

Depends tidyverse

Imports stats, dplyr, dtplyr, fastDummies, forecast, data.table, purrr, tidyr, tibble, ggplot2, survival, reshape2, bshazard, SynthETIC, rpart, reticulate, xgboost, SHAPforxgboost

SystemRequirements Python (>= 3.8.0)

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.2

NeedsCompilation no

Author Emil Hofman [aut, cre, cph], Gabriele Pittarello [aut, cph] (<https://orcid.org/0000-0003-3360-5826>), Munir Hiabu [aut, cph] (<https://orcid.org/0000-0001-5846-667X>)

Maintainer Emil Hofman <emil_hofman@hotmail.dk>

Repository CRAN

Date/Publication 2024-11-14 16:00:10 UTC

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev python3 libx11-dev zlib1g-dev

Contents

`data_generator` *Individual data generator*

Description

This function generates the monthly individual claims data in the accompanying methodological paper using the `SynthETIC` package. This simple function allows to simulate from a sand-box to test out the `ReSurv` approach. Some parameters of the simulation can be changed.

Usage

```
data_generator(
  ref_claim = 2e+05,
  time_unit = 1/360,
  years = 4,
  random_seed = 1964,
```

```
  period_exposure = 200,
  period_frequency = 0.2,
  scenario = 1
)
```

Arguments

`ref_claim`	`integer`, reference claim size.
`time_unit`	`numeric`, output time unit.
`years`	`integer`, number of years to be simulated.
`random_seed`	`integer`, random seed for replicable code.
`period_exposure`	`integer`, volume (number of policies) underwritten each period.
`period_frequency`	`numeric`, expected frequency in each period.
`scenario`	`character` or `numeric`, one of the scenarios described in the accompanying manuscript. Possible choices are 'alpha' (0), 'beta' (1), 'gamma'(2), 'delta'(3),'epsilon'(4). Our simulated data are constituted of a mix of short tail claims (`claim_type` `0`) and claims with longer resolution (`claim_type` `1`). We chose the parameter of the simulator to resemble a mix of property damage (`claim_type` `0`) and bodily injuries (`claim_type` `1`). each scenario has distinctive characteristics. Scenario Alpha is a mix of `claim_type` `0` and `claim_type` `1` with same number of claims volume at each accident period. Differently from scenario Alpha, in scenario Beta the volumes of `claim_type` `1` are decreasing in the most recent accident periods. In scenario Gamma we add an interaction between `claim_type` `1` and accident period: in a real world setting this can be motivated by a change in consumer behavior or company policies resulted in different reporting patterns over time. In scenario Delta, we introduce a seasonality effect dependent on the accident period for `claim_type` `0` and `claim_type` `1`. In the real word, scenario Delta resembles seasonal changes in the workforce composition. Scenario Epsilon does not satisfy the proportionality assumption.

Value

Individual claims data. It contains the following columns:

- `claim_number`: Policy ID.
- `claim_type`: Type of claim. It can be either 0 or 1.
- `AP`: Accident period
- `RP`: Reporting period.

References

Avanzi, B., Taylor, G., Wang, M., & Wong, B. (2021). SynthETIC: an individual insurance claim simulator with feature control. Insurance: Mathematics and Economics, 100, 296-308.

Hiabu, M., Hofman, E., & Pittarello, G. (2023). A machine learning approach based on survival analysis for IBNR frequencies in non-life reserving. arXiv preprint arXiv:2312.14549.

Examples

```
input_data_0 <- data_generator(
random_seed = 1964,
scenario = "alpha",
time_unit = 1,
years = 2,
period_exposure = 100)
```

`IndividualDataPP` *Individual Data Pre-Processing*

Description

This function pre-processes the data for the application of a `ReSurv` model.

Usage

```
IndividualDataPP(
  data,
  id = NULL,
  continuous_features = NULL,
  categorical_features = NULL,
  accident_period,
  calendar_period,
  input_time_granularity = "months",
  output_time_granularity = "quarters",
  years = NULL,
  calendar_period_extrapolation = FALSE,
  continuous_features_spline = NULL,
  degrees_cf = 3,
  degrees_of_freedom_cf = 4,
  degrees_cp = 3,
  degrees_of_freedom_cp = 4
)
```

Arguments

`data`	`data.frame`, for the individual reserving. The number of development periods can be larger than the number of accident periods.
`id`	`character`, `data` column that contains the policy identifier. If `NULL` (default), we assume that each row is an observation. We assume that each observation can only have one reporting time, if not null we take the reporting time of the first row for each `id`.

`continuous_features`
: `character`, continuous features columns to be scaled.

`categorical_features`
: `character`, categorical features columns to be one-hot encoded.

`accident_period`
: `character`, it contains the name of the column in data corresponding to the accident period.

`calendar_period`
: `character`, it contains the name of the column in data corresponding to the calendar period.

`input_time_granularity`
: `character`, time unit of the input data. Granularity supported:

  - `"days"`: the input data are daily.
  - `"months"`: the input data are monthly.
  - `"quarters"`: the input data are quarterly
  - `"years"`: the input data are yearly.

  Default to `months`.

`output_time_granularity`
: `character`, time unit of the output data. The granularity supported is the same as for the input data:

  - `"days"`: the output data will be on a daily scale.
  - `"months"`: the output data will be on a monthly scale.
  - `"quarters"`: the output data will be on a quarterly scale.
  - `"years"`: the output data will be on yearly scale.

  The output granularity must be bigger than the input granularity. Also, the output granularity must be consistent with the input granularity, meaning that the time conversion must be possible. E.g., it is possible to group quarters to years. It is not possible to group quarters to semesters. Default to `quarters`.

`years`
: `numeric`, number of development years in the study.

`calendar_period_extrapolation`
: `character`, whether a spline for calendar extrapolation should be considered in the cox model fit. Default is 'FALSE'.

`continuous_features_spline`
: `logical`, weather a spline for smoothing continuous features should be added.

`degrees_cf`
: `numeric`, degrees of the spline for smoothing continuous features.

`degrees_of_freedom_cf`
: `numeric`, degrees of freedom of the splines for smoothing continuous features.

`degrees_cp`
: `numeric`, degrees of the spline for smoothing the calendar period effect.

`degrees_of_freedom_cp`
: `numeric`, degrees of freedom of the splines for smoothing the calendar period effect.

Details

The input accident_period is coded as AP_i. The input development periods are derived as DP_i=calendar_period-accident_period+1.

The reverse time development factors are DP_rev_i = DP_max-DP_i, where DP_max is the maximum number of development times: DP_i $= 1, \ldots,$DP_max. Given the parameter years, DP_max is derived internally from our package.

As for the truncation time, TR_i = AP_i-1.

AP_i, DP_i, DP_rev_i and TR_i are converted to AP_o, DP_o, DP_rev_o and TR_o (from the input_time_granularity to the output_time_granularity) using a multiplicative conversion factor. E.g., AP_o = AP_i * CF.

The conversion factor is computed as

$CF = \frac{\nu^i}{(\nu^o)^{-1}}$,

where ν^i and ν^o are the fraction of a year corresponding to input_time_granularity and output_time_granularity. ν^i and ν^o take values 1/360, 1/12, 1/4, 1/2, 1 for "days", "months", "quarters", "semesters", "years" respectively. We will have RP_o = AP_o + DP_o.

Value

IndividualDataPP object. A list containing

- full.data: data.frame. The input data after pre-processing.
- starting.data: data.frame. The input data as they were provided from the user.
- training.data: data.frame. The input data pre-processed for training.
- conversion_factor: numeric. The conversion factor for going from input granularity to output granularity. E.g, the conversion factor for going from months to quarters is 1/3.
- string_formula_i: character. The survival formula to model the data in input granularity.
- string_formula_o: character. The survival formula to model the in data output granularity.
- continuous_features: character. The continuous features names as provided from the user.
- categorical_features: character. The categorical features names as provided from the user.
- calendar_period_extrapolation: logical. The value specifying if a calendar period component is extrapolated.
- years: numeric. Total number of development years in the data. Default is NULL and computed automatically from the data.
- accident_period: character. Accident period column name.
- calendar_period: character. Calendar_period column name.
- input_time_granularity: character. Input time granularity.
- output_time_granularity: character. Output time granularity.

After pre-processing, we provide a standard encoding for the time components. This regards the output in `training.data` and `full.data`. In the `ReSurv` notation:

- `AP_i`: Input granularity accident period.
- `AP_o`: Output granularity accident period.
- `DP_i`: Input granularity development period in forward time.
- `DP_rev_i`: Input granularity development period in reverse time.
- `DP_rev_o`: Output granularity development period in reverse time.
- `TR_i`: Input granularity truncation time.
- `TR_o`: Output granularity truncation time.
- `I`: event indicator, under this framework is equal to one for each entry.

References

Munir, H., Emil, H., & Gabriele, P. (2023). A machine learning approach based on survival analysis for IBNR frequencies in non-life reserving. arXiv preprint arXiv:2312.14549.

Examples

```
input_data_0 <- data_generator(
random_seed = 1964,
scenario = "alpha",
time_unit = 1,
years = 2,
period_exposure = 100)

individual_data <- IndividualDataPP(data = input_data_0,
categorical_features = "claim_type",
continuous_features = "AP",
accident_period = "AP",
calendar_period = "RP",
input_time_granularity = "years",
output_time_granularity = "years",
years = 2)
```

---

install_pyresurv *Install Python Environment for ReSurv*

Description

Install a Python environment that allows the user to apply the Neural Network (NN) models.

Usage

```
install_pyresurv(
  ...,
  envname = "pyresurv",
  new_env = identical(envname, "pyresurv")
)
```

Arguments

...	Additional arguments for ‘virtualenv_create‘.
envname	‘character‘. Name of the environment created. Default ‘pyresurv‘.
new_env	‘logical‘. If ‘TRUE‘, any existing Python virtual environment and/or ‘conda‘ environment specified by ‘envname‘ is deleted first.

Value

No return value.

---

ooslkh *Compute the out-of-sample likelihood*

Description

When the lower triangle data are available, this method computes the likelihood on the lower triangle.

Usage

```
ooslkh(object, ...)
```

Arguments

object	`ReSurvFit` object.
...	Other arguments to pass to ooslkh.

Value

`numeric`, out-of-sample likelihood.

ooslkh.default *Compute the out-of-the sample likelihood*

Description

When the lower triangle data are available, this method computes the likelihood on the lower triangle.

Usage

```
## Default S3 method:
ooslkh(object, ...)
```

Arguments

object	ReSurvFit object.
...	Other arguments to pass to ooslkh.

Value

numeric, out-of-sample likelihood.

ooslkh.ReSurvFit *Compute the out-of-the sample likelihood*

Description

When the lower triangle data are available, this method computes the likelihood on the lower triangle.

Usage

```
## S3 method for class 'ReSurvFit'
ooslkh(object, ...)
```

Arguments

object	ReSurvFit object.
...	Other arguments to pass to ooslkh.

Value

numeric, out-of-sample likelihood.

pkg.env *Helper functions*

Description

This script contains the utils functions that are used in ReSurv.

Usage

```
pkg.env
```

Format

An object of class `environment` of length 60.

plot.ReSurvFit *Plot for machine learning models*

Description

This function plots the mean absolute SHAP values for the ReSurv fits of machine learning models.

Usage

```
## S3 method for class 'ReSurvFit'
plot(x, nsamples = NULL, ...)
```

Arguments

`x`	`ReSurvFit` x.
`nsamples`	`integer`, number of observations to sample for neural networks features importance plot.
`...`	Other arguments to be passed to plot.

Value

`ggplot2` of the SHAP values for an `"XGB"` model or a `"NN"` model.

plot.ReSurvPredict *Plot of the development factors*

Description

Plots the development factors by group code.

Usage

```
## S3 method for class 'ReSurvPredict'
plot(
  x,
  granularity = "input",
  group_code = 1,
  color_par = "royalblue",
  linewidth_par = 2.5,
  ylim_par = NULL,
  ticks_by_par = NULL,
  base_size_par = NULL,
  title_par = NULL,
  x_text_par = NULL,
  plot.title.size_par = NULL,
  ...
)
```

Arguments

x	"ReSurvPredict" object specifying hazard and development factors.
granularity	character, either "input" for input_time_granularity or "output" for output_time_granularity.
group_code	numeric: Identifier for the group that will be plotted. Default is 1. The code identifiers can be find in the ReSurvPredict$long_triangle_format_out list. Depending on the granularity of interest, it will be either in ReSurvPredict$long_triangle_format_ou for input_time_granularity or ReSurvPredict$long_triangle_format_out$output_granularity for output_time_granularity.
color_par	character: ggplot2 Colour of the line plot. Default is 'royalblue'. Optional.
linewidth_par	numeric: Line plot width. Optional.
ylim_par	numeric: Highest intercept on the y-axis (development factors). The default is the highest predicted development factor. Optional.
ticks_by_par	numeric: gap between each x-axis label (development period). Default is 2. Optional.
base_size_par	numeric: base size of the plot. Default is 5. See base_size in the ?theme_bw documentation. Optional.
title_par	character: Title of the plot. Optional.

`x_text_par` `character`: Text on the x-axis. Optional.

`plot.title.size_par`
`numeric`: size of the plot title. Default is 20. See `size` in the `?element_text` documentation. Optional.

`...` Other arguments to be passed to Plot. Optional.

Value

`ggplot2` of the development factors

`predict.ReSurvFit` *Predict IBNR frequency*

Description

This function predicts the results from the ReSurv fits.

Usage

```
## S3 method for class 'ReSurvFit'
predict(
  object,
  newdata = NULL,
  grouping_method = "probability",
  check_value = 1.85,
  ...
)
```

Arguments

`object` `ResurvFit` object specifying start time, end time and status.

`newdata` `IndividualDataPP` object that contains new data to predict.

`grouping_method`
`character`, use probability or exposure approach to group from input to output development factors. Choice between:

- `"exposure"`
- `"probability"`

Default is `"exposure"`.

`check_value` `numeric`, check hazard value on initial granularity, if above threshold we increase granularity to try and adjust the development factor.

`...` Additional arguments to pass to the predict function.

Value

Predictions for the ReSurvFit model. It includes

- ReSurvFit: Fitted ReSurv model.
- long_triangle_format_out: data.frame. Predicted development factors and IBNR claim counts for each feature combination in long format.
  - input_granularity: data.frame. Predictions for each feature combination in long format for input_time_granularity.
    - AP_i: Accident period, input_time_granularity.
    - DP_i: Development period, input_time_granularity.
    - f_i: Predicted development factors, input_time_granularity.
    - group_i: Group code, input_time_granularity. This associates to each feature combination an identifier.
    - expected_counts: Expected counts, input_time_granularity.
    - IBNR: Predicted IBNR claim counts, input_time_granularity.
  - output_granularity: data.frame. Predictions for each feature combination in long format for output_time_granularity.
    - AP_o: Accident period, output_time_granularity.
    - DP_o: Development period, output_time_granularity.
    - f_o: Predicted development factors, output_time_granularity.
    - group_o: Group code, output_time_granularity. This associates to each feature combination an identifier.
    - expected_counts: Expected counts, output_time_granularity.
    - IBNR: Predicted IBNR claim counts, output_time_granularity.
- lower_triangle: Predicted lower triangle.
  - input_granularity: data.frame. Predicted lower triangle for input_time_granularity.
  - output_granularity: data.frame. Predicted lower triangle for output_time_granularity.
- predicted_counts: numeric. Predicted total frequencies.
- grouping_method: character. Chosen grouping method.

---

print.summaryReSurvPredict

Print summary of IBNR predictions

Description

Gives overview of IBNr predictions

Usage

```
## S3 method for class 'summaryReSurvPredict'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	"ReSurvPredict" object specifying hazard and development factors.
digits	numeric, number of digits to print for IBNR and Likelihood.
...	Other arguments to be passed to print.

Value

print of summary of predictions

ReSurv *Fit* ReSurv *models on the individual data.*

Description

This function fits and computes the reserves for the ReSurv models

Usage

```
ReSurv(
  IndividualDataPP,
  hazard_model = "COX",
  tie = "efron",
  baseline = "spline",
  continuous_features_scaling_method = "minmax",
  random_seed = 1,
  hparameters = list(),
  percentage_data_training = 0.8,
  grouping_method = "exposure",
  check_value = 1.85
)
```

Arguments

IndividualDataPP	IndividualDataPP object to use for the ReSurv fit.
hazard_model	character, hazard model supported from our package, must be provided as a string. The model can be chosen from: • "COX": Standard Cox model for the hazard. • "NN": Deep Survival Neural Network. • "XGB": eXtreme Gradient Boosting.
tie	ties handling, default is the Efron approach.
baseline	handling the baseline hazard. Default is a spline.
continuous_features_scaling_method	method to preprocess the features

`random_seed` `integer`, random seed set for reproducibility

`hparameters` `list`, hyperparameters for the machine learning models. It will be disregarded for the cox approach.

`percentage_data_training`
`numeric`, percentage of data used for training on the upper triangle.

`grouping_method`
`character`, use probability or exposure approach to group from input to output development factors. Choice between:

- `"exposure"`
- `"probability"`

Default is `"exposure"`.

`check_value` `numeric`, check hazard value on initial granularity, if above threshold we increase granularity to try and adjust the development factor.

Details

The model fit uses the theoretical framework of Hiabu et al. (2023), that relies on the correspondence between hazard models and development factors:

To be completed with final notation of the paper.

The `ReSurv` package assumes proportional hazard models. Given an i.i.d. sample $\{y_i, x_i\}_{i=1,\dots,n}$ the individual hazard at time t is:

$\lambda_i(t) = \lambda_0(t)e^{y_i(x_i)}$

Composed of a baseline $\lambda_0(t)$ and a proportional effect $e^{y_i(x_i)}$.

Currently, the implementation allows to optimize the partial likelihood (concerning the proportional effects) using one of the following statistical learning approaches:

- COX
- Neural Networks
- eXtreme Gradient Boosting

Value

`ReSurv` fit. A list containing

- `model.out`: `list` containing the pre-processed covariates data for the fit (`data`) and the basic model output (`model.out`;COX, XGB or NN).
- `is_lkh`: `numeric` Training negative log likelihood.
- `os_lkh`: `numeric` Validation negative log likelihood. Not available for COX.
- `hazard_frame`: `data.frame` containing the fitted hazard model with the corresponding covariates. It contains:
  - `expg`: fitted risk score.
  - `baseline`: fitted baseline.
  - `hazard`: fitted hazard rate (`expg`*`baseline`).
  - `f_i`: fitted development factors.

  - `cum_f_i`: fitted cumulative development factors.
  - `S_i`:fitted survival function.
  - `S_i_lag`:fitted survival function (lag version, for further information see `?dplyr::lag`).
  - `S_i_lead`:fitted survival function (lead version, for further information see `?dplyr::lead`).
- `hazard_model`: `string` chosen hazard model (COX, NN or XGB)
- `IndividualDataPP`: starting `IndividualDataPP` object.

References

Munir, H., Emil, H., & Gabriele, P. (2023). A machine learning approach based on survival analysis for IBNR frequencies in non-life reserving. arXiv preprint arXiv:2312.14549.

Therneau, T. M., & Lumley, T. (2015). Package 'survival'. R Top Doc, 128(10), 28-33.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC medical research methodology, 18(1), 1-12.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. R version, 90, 1-66.

Examples

```
input_data_0 <- data_generator(
random_seed = 1964,
scenario = "alpha",
time_unit = 1,
years = 4,
period_exposure = 100)

individual_data <- IndividualDataPP(data = input_data_0,
categorical_features = "claim_type",
continuous_features = "AP",
accident_period = "AP",
calendar_period = "RP",
input_time_granularity = "years",
output_time_granularity = "years",
years=4)


resurv_fit_cox <- ReSurv(individual_data,
hazard_model = "COX")
```

ReSurv.default *Fit* ReSurv *models on the individual data.*

Description

This function fits and computes the reserves for the ReSurv models

Usage

```
## Default S3 method:
ReSurv(
  IndividualDataPP,
  hazard_model = "COX",
  tie = "efron",
  baseline = "spline",
  continuous_features_scaling_method = "minmax",
  random_seed = 1,
  hparameters = list(),
  percentage_data_training = 0.8,
  grouping_method = "exposure",
  check_value = 1.85
)
```

Arguments

IndividualDataPP	IndividualDataPP object to use for the ReSurv fit.
hazard_model	character, hazard model supported from our package, must be provided as a string. The model can be chosen from: • "COX": Standard Cox model for the hazard. • "NN": Deep Survival Neural Network. • "XGB": eXtreme Gradient Boosting.
tie	ties handling, default is the Efron approach.
baseline	handling the baseline hazard. Default is a spline.
continuous_features_scaling_method	method to preprocess the features
random_seed	integer, random seed set for reproducibility
hparameters	list, hyperparameters for the machine learning models. It will be disregarded for the cox approach.
percentage_data_training	numeric, percentage of data used for training on the upper triangle.
grouping_method	character, use probability or exposure approach to group from input to output development factors. Choice between:

- "exposure"
- "probability"

Default is "exposure".

check_value numeric, check hazard value on initial granularity, if above threshold we increase granularity to try and adjust the development factor.

Details

The model fit uses the theoretical framework of Hiabu et al. (2023), that relies on the correspondence between hazard models and development factors:

To be completed with final notation of the paper.

The ReSurv package assumes proportional hazard models. Given an i.i.d. sample $\{y_i, x_i\}_{i=1,\dots,n}$ the individual hazard at time t is:

$\lambda_i(t) = \lambda_0(t)e^{y_i(x_i)}$

Composed of a baseline $\lambda_0(t)$ and a proportional effect $e^{y_i(x_i)}$.

Currently, the implementation allows to optimize the partial likelihood (concerning the proportional effects) using one of the following statistical learning approaches:

- COX
- Neural Networks
- eXtreme Gradient Boosting

Value

ReSurv fit. A list containing

- model.out: list containing the pre-processed covariates data for the fit (data) and the basic model output (model.out;COX, XGB or NN).
- is_lkh: numeric Training negative log likelihood.
- os_lkh: numeric Validation negative log likelihood. Not available for COX.
- hazard_frame: data.frame containing the fitted hazard model with the corresponding covariates. It contains:
  - expg: fitted risk score.
  - baseline: fitted baseline.
  - hazard: fitted hazard rate (expg*baseline).
  - f_i: fitted development factors.
  - cum_f_i: fitted cumulative development factors.
  - S_i:fitted survival function.
  - S_i_lag:fitted survival function (lag version, for further information see ?dplyr::lag).
  - S_i_lead:fitted survival function (lead version, for further information see ?dplyr::lead).
- hazard_model: string chosen hazard model (COX, NN or XGB)
- IndividualDataPP: starting IndividualDataPP object.

References

Pittarello, G., Hiabu, M., & Villegas, A. M. (2023). Chain Ladder Plus: a versatile approach for claims reserving. arXiv preprint arXiv:2301.03858.

Therneau, T. M., & Lumley, T. (2015). Package 'survival'. R Top Doc, 128(10), 28-33.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC medical research methodology, 18(1), 1-12.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. R version, 90, 1-66.

Examples

```
input_data_0 <- data_generator(
random_seed = 1964,
scenario = "alpha",
time_unit = 1,
years = 4,
period_exposure = 100)

individual_data <- IndividualDataPP(data = input_data_0,
categorical_features = "claim_type",
continuous_features = "AP",
accident_period = "AP",
calendar_period = "RP",
input_time_granularity = "years",
output_time_granularity = "years",
years=4)


resurv_fit_cox <- ReSurv(individual_data,
hazard_model = "COX")
```

ReSurv.IndividualDataPP

Fit ReSurv *models on the individual data.*

Description

This function fits and computes the reserves for the ReSurv models

Usage

```
## S3 method for class 'IndividualDataPP'
ReSurv(
  IndividualDataPP,
  hazard_model = "COX",
  tie = "efron",
  baseline = "spline",
  continuous_features_scaling_method = "minmax",
  random_seed = 1,
  hparameters = list(),
  percentage_data_training = 0.8,
  grouping_method = "exposure",
  check_value = 1.85
)
```

Arguments

`IndividualDataPP`
: IndividualDataPP object to use for the `ReSurv` fit.

`hazard_model`
: `character`, hazard model supported from our package, must be provided as a string. The model can be chosen from:

  - `"COX"`: Standard Cox model for the hazard.
  - `"NN"`: Deep Survival Neural Network.
  - `"XGB"`: eXtreme Gradient Boosting.

`tie`
: ties handling, default is the Efron approach.

`baseline`
: handling the baseline hazard. Default is a spline.

`continuous_features_scaling_method`
: method to preprocess the features

`random_seed`
: `integer`, random seed set for reproducibility

`hparameters`
: `list`, hyperparameters for the machine learning models. It will be disregarded for the cox approach.

`percentage_data_training`
: `numeric`, percentage of data used for training on the upper triangle.

`grouping_method`
: `character`, use probability or exposure approach to group from input to output development factors. Choice between:

  - `"exposure"`
  - `"probability"`

  Default is `"exposure"`.

`check_value`
: `numeric`, check hazard value on initial granularity, if above threshold we increase granularity to try and adjust the development factor.

Details

The model fit uses the theoretical framework of Hiabu et al. (2023), that relies on the correspondence between hazard models and development factors:

To be completed with final notation of the paper.

The `ReSurv` package assumes proportional hazard models. Given an i.i.d. sample $\{y_i, x_i\}_{i=1,\dots,n}$ the individual hazard at time t is:

$\lambda_i(t) = \lambda_0(t)e^{y_i(x_i)}$

Composed of a baseline $\lambda_0(t)$ and a proportional effect $e^{y_i(x_i)}$.

Currently, the implementation allows to optimize the partial likelihood (concerning the proportional effects) using one of the following statistical learning approaches:

- COX
- Neural Networks
- eXtreme Gradient Boosting

Value

`ReSurv` fit. A list containing

- `model.out`: `list` containing the pre-processed covariates data for the fit (`data`) and the basic model output (`model.out`;COX, XGB or NN).
- `is_lkh`: `numeric` Training negative log likelihood.
- `os_lkh`: `numeric` Validation negative log likelihood. Not available for COX.
- `hazard_frame`: `data.frame` containing the fitted hazard model with the corresponding covariates. It contains:
  - `expg`: fitted risk score.
  - `baseline`: fitted baseline.
  - `hazard`: fitted hazard rate (`expg`*`baseline`).
  - `f_i`: fitted development factors.
  - `cum_f_i`: fitted cumulative development factors.
  - `S_i`:fitted survival function.
  - `S_i_lag`:fitted survival function (lag version, for further information see `?dplyr::lag`).
  - `S_i_lead`:fitted survival function (lead version, for further information see `?dplyr::lead`).
- `hazard_model`: `string` chosen hazard model (COX, NN or XGB)
- `IndividualDataPP`: starting `IndividualDataPP` object.

References

Pittarello, G., Hiabu, M., & Villegas, A. M. (2023). Chain Ladder Plus: a versatile approach for claims reserving. arXiv preprint arXiv:2301.03858.

Therneau, T. M., & Lumley, T. (2015). Package 'survival'. R Top Doc, 128(10), 28-33.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC medical research methodology, 18(1), 1-12.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. R version, 90, 1-66.

Examples

```
input_data_0 <- data_generator(
random_seed = 1964,
scenario = "alpha",
time_unit = 1,
years = 4,
period_exposure = 100)

individual_data <- IndividualDataPP(data = input_data_0,
categorical_features = "claim_type",
continuous_features = "AP",
accident_period = "AP",
calendar_period = "RP",
input_time_granularity = "years",
output_time_granularity = "years",
years=4)


resurv_fit_cox <- ReSurv(individual_data,
hazard_model = "COX")
```

ReSurvCV *K fold cross-validation of a* ReSurv *model.*

Description

This function computes a K fold cross-validation of a pre-specified machine learning model supported from the ReSurv package for a given grid of hyperparameters. The hyperparameters to be tested are provided in a list, namely hparameters_grid. Conversely, the parameters for the models run are provided separately as arguments and they are specific for each machine learning model support from.

Usage

```
ReSurvCV(
  IndividualDataPP,
  model,
  hparameters_grid,
  folds,
  random_seed,
  continuous_features_scaling_method = "minmax",
  print_every_n = 1L,
  nrounds = NULL,
```

```
  early_stopping_rounds = NULL,
  epochs = 1,
  parallel = FALSE,
  ncores = 1,
  num_workers = 0,
  verbose = FALSE,
  verbose.cv = FALSE
)
```

Arguments

IndividualDataPP	IndividualDataPP object to use for the ReSurv fit cross-validation.
model	character, machine learning for cross validation.
hparameters_grid	list, grid of the hyperparameters to cross-validate.
folds	integer, number of folds (i.e. K).
random_seed	integer, random seed for making the code reproducible.
continuous_features_scaling_method	character, method for scaling continuous features.
print_every_n	integer, specific to the XGB approach, see xgboost::xgb.train documentation.
nrounds	integer, specific to XGB, max number of boosting iterations.
early_stopping_rounds	integer, specific to the XGB approach, see xgboost::xgb.train documentation.
epochs	integer, specific to the NN approach, epochs to be checked.
parallel	logical, specific to the NN approach, whether to use parallel computing.
ncores	integer, specific to NN, max number of cores used.
num_workers	numeric, number of workers for the NN approach, multi-process data loading with the specified number of loader worker processes.
verbose	logical, whether messages from the machine learning models must be printed.
verbose.cv	logical, whether messages from cross-validation must be printed.

Value

Best ReSurv model fit. The output is different depending on the machine learning approach that is required for cross-validation. A list containing:

- out.cv: data.frame, total output of the cross-validation (all the input parameters combinations).
- out.cv.best.oos: data.frame, combination with the best out of sample likelihood.

For XGB the columns in out.cv and out.cv.best.oos are the hyperparameters booster, eta, max_depth, subsample, alpha, lambda, min_child_weight. They also contain the metrics train.lkh, test.lkh, and the computational time time. For NN the columns in out.cv and out.cv.best.oos are the hyperparameters num_layers, optim, activation, lr, xi, eps, tie, batch_size, early_stopping, patience, node train.lkh test.lkh. They also contain the metrics train.lkh, test.lkh, and the computational time time.

References

Munir, H., Emil, H., & Gabriele, P. (2023). A machine learning approach based on survival analysis for IBNR frequencies in non-life reserving. arXiv preprint arXiv:2312.14549.

ReSurvCV.default *K fold cross-validation of ReSurv model.*

Description

This function computes a K fold cross-validation of a pre-specified ReSurv model for a given grid of parameters.

Usage

```
## Default S3 method:
ReSurvCV(
  IndividualDataPP,
  model,
  hparameters_grid,
  folds,
  random_seed,
  continuous_features_scaling_method = "minmax",
  print_every_n = 1L,
  nrounds = NULL,
  early_stopping_rounds = NULL,
  epochs = 1,
  parallel = FALSE,
  ncores = 1,
  num_workers = 0,
  verbose = FALSE,
  verbose.cv = FALSE
)
```

Arguments

IndividualDataPP	IndividualDataPP object to use for the ReSurv fit cross-validation.
model	character, machine learning for cross validation.

hparameters_grid
: list, grid of the hyperparameters to cross-validate.

folds
: integer, number of folds (i.e. K).

random_seed
: integer, random seed for making the code reproducible.

continuous_features_scaling_method
: character, method for scaling continuous features.

print_every_n
: integer, specific to the XGB approach, see xgboost::xgb.train documentation.

nrounds
: integer, specific to XGB, max number of boosting iterations.

early_stopping_rounds
: integer, specific to the XGB approach, see xgboost::xgb.train documentation.

epochs
: integer, specific to the NN approach, epochs to be checked.

parallel
: logical, specific to the NN approach, whether to use parallel computing.

ncores
: integer, specific to NN, max number of cores used.

num_workers
: numeric, number of workers for the NN approach, multi-process data loading with the specified number of loader worker processes.

verbose
: logical, whether messages from the machine learning models must be printed.

verbose.cv
: logical, whether messages from cross-validation must be printed.

Value

Best ReSurv model fit. The output is different depending on the machine learning approach that is required for cross-validation. A list containing:

- out.cv: data.frame, total output of the cross-validation (all the input parameters combinations).
- out.cv.best.oos: data.frame, combination with the best out of sample likelihood.

For XGB the columns in out.cv and out.cv.best.oos are the hyperparameters booster, eta, max_depth, subsample, alpha, lambda, min_child_weight. They also contain the metrics train.lkh, test.lkh, and the computational time time. For NN the columns in out.cv and out.cv.best.oos are the hyperparameters num_layers, optim, activation, lr, xi, eps, tie, batch_size, early_stopping, patience, node train.lkh test.lkh. They also contain the metrics train.lkh, test.lkh, and the computational time time.

References

Munir, H., Emil, H., & Gabriele, P. (2023). A machine learning approach based on survival analysis for IBNR frequencies in non-life reserving. arXiv preprint arXiv:2312.14549.

ReSurvCV.IndividualDataPP

K fold cross-validation of ReSurv model.

Description

This function computes a K fold cross-validation of a pre-specified ReSurv model for a given grid of parameters.

Usage

```
## S3 method for class 'IndividualDataPP'
ReSurvCV(
  IndividualDataPP,
  model,
  hparameters_grid,
  folds,
  random_seed,
  continuous_features_scaling_method = "minmax",
  print_every_n = 1L,
  nrounds = NULL,
  early_stopping_rounds = NULL,
  epochs = NULL,
  parallel = FALSE,
  ncores = 1,
  num_workers = 0,
  verbose = FALSE,
  verbose.cv = FALSE
)
```

Arguments

IndividualDataPP	IndividualDataPP object to use for the ReSurv fit cross-validation.
model	character, machine learning for cross validation.
hparameters_grid	list, grid of the hyperparameters to cross-validate.
folds	integer, number of folds (i.e. K).
random_seed	integer, random seed for making the code reproducible.
continuous_features_scaling_method	character, method for scaling continuous features.
print_every_n	integer, specific to the XGB approach, see xgboost::xgb.train documentation.
nrounds	integer, specific to XGB, max number of boosting iterations.

early_stopping_rounds
: integer, specific to the XGB approach, see xgboost::xgb.train documentation.

epochs
: integer, specific to the NN approach, epochs to be checked.

parallel
: logical, specific to the NN approach, whether to use parallel computing.

ncores
: integer, specific to NN, max number of cores used.

num_workers
: numeric, number of workers for the NN approach, multi-process data loading with the specified number of loader worker processes.

verbose
: logical, whether messages from the machine learning models must be printed.

verbose.cv
: logical, whether messages from cross-validation must be printed.

Value

Best ReSurv model fit. The output is different depending on the machine learning approach that is required for cross-validation. A list containing:

- out.cv: data.frame, total output of the cross-validation (all the input parameters combinations).
- out.cv.best.oos: data.frame, combination with the best out of sample likelihood.

For XGB the columns in out.cv and out.cv.best.oos are the hyperparameters booster, eta, max_depth, subsample, alpha, lambda, min_child_weight. They also contain the metrics train.lkh, test.lkh, and the computational time time. For NN the columns in out.cv and out.cv.best.oos are the hyperparameters num_layers, optim, activation, lr, xi, eps, tie, batch_size, early_stopping, patience, node train.lkh test.lkh. They also contain the metrics train.lkh, test.lkh, and the computational time time.

---

summary.ReSurvPredict *Summary of IBNR predictions*

Description

Gives overview of IBNR predictions

Usage

```
## S3 method for class 'ReSurvPredict'
summary(object, granularity = "input", ...)
```

Arguments

object
: "ReSurvPredict" object specifying hazard and development factors.

granularity
: character, specify if which granularity the summary should be on.
  - "input"
  - "output"

  Default is "input".

...
: Other arguments to be passed to summary.

Value

Summary of predictions

`survival_crps` *Survival continuously ranked probability score.*

Description

Return the Survival Continuously Ranked Probability Score (SCRPS) of a `ReSurv` model.

Usage

```
survival_crps(ReSurvFit, user_data_set = NULL)
```

Arguments

`ReSurvFit`	ReSurvFit object to use for the score computation.
`user_data_set`	data.frame provided from the user to compute the survival CRPS, optional.

Details

The model fit uses the theoretical framework of Hiabu et al. (2023), that relies on the

Value

Survival CRPS, `data.table` that contains the CRPS (`crps`) for each observation (`id`).

References

Pittarello, G., Hiabu, M., & Villegas, A. M. (2023). Chain Ladder Plus: a versatile approach for claims reserving. arXiv preprint arXiv:2301.03858.

Therneau, T. M., & Lumley, T. (2015). Package 'survival'. R Top Doc, 128(10), 28-33.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC medical research methodology, 18(1), 1-12.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. R version, 90, 1-66.

`survival_crps.default` *Survival continuously ranked probability score.*

Description

Return the Survival Continuously Ranked Probability Score (SCRPS) of a `ReSurv` model.

Usage

```
## Default S3 method:
survival_crps(ReSurvFit, user_data_set = NULL)
```

Arguments

`ReSurvFit`	ReSurvFit object to use for the score computation.
`user_data_set`	data.frame provided from the user to compute the survival CRPS, optional.

Details

The model fit uses the theoretical framework of Hiabu et al. (2023), that relies on the

Value

Survival CRPS, `data.table` that contains the CRPS (`crps`) for each observation (`id`).

References

Pittarello, G., Hiabu, M., & Villegas, A. M. (2023). Chain Ladder Plus: a versatile approach for claims reserving. arXiv preprint arXiv:2301.03858.

Therneau, T. M., & Lumley, T. (2015). Package 'survival'. R Top Doc, 128(10), 28-33.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC medical research methodology, 18(1), 1-12.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. R version, 90, 1-66.

`survival_crps.ReSurvFit` *Survival continuously ranked probability score.*

Description

Return the Survival Continuously Ranked Probability Score (SCRPS) of a `ReSurv` model.

Usage

```
## S3 method for class 'ReSurvFit'
survival_crps(ReSurvFit, user_data_set = NULL)
```

Arguments

`ReSurvFit`	ReSurvFit object to use for the score computation.
`user_data_set`	data.frame provided from the user to compute the survival CRPS, optional.

Details

The model fit uses the theoretical framework of Hiabu et al. (2023), that relies on the

Value

Survival CRPS, `data.table` that contains the CRPS (`crps`) for each observation (`id`).

References

Pittarello, G., Hiabu, M., & Villegas, A. M. (2023). Chain Ladder Plus: a versatile approach for claims reserving. arXiv preprint arXiv:2301.03858.

Therneau, T. M., & Lumley, T. (2015). Package 'survival'. R Top Doc, 128(10), 28-33.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC medical research methodology, 18(1), 1-12.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. R version, 90, 1-66.

Index