Package ‘survex’

December 1, 2022

Title Explainable Machine Learning in Survival Analysis

Version 0.2.2

Description Survival analysis models are commonly used in medicine and other areas. Many of them are too complex to be interpreted by human. Exploration and explanation is needed, but standard methods do not give a broad enough picture. 'survex' provides easy-to-apply methods for explaining survival models, both complex black-boxes and simpler statistical models. They include methods specific to survival analysis such as SurvSHAP(t) described in Krzyzinski et al., (2022) <arXiv:2208.11080>, SurvLIME introduced in Kovalov et al., (2020) <doi:10.1016/j.knosys.2020.106164> as well as extensions of existing ones described in Biecek et al., (2021) <doi:10.1201/9780429027192>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.1

Depends R (>= 3.5.0)

Imports DALEX (>= 2.2.1), ggplot2, pec, survival, patchwork

Suggests censored, covr, gbm, generics, glmnet, ingredients, knitr, mboost, parsnip, progressr, randomForestSRC, ranger, rmarkdown, testthat (>= 3.0.0), withr, xgboost

Config/testthat/edition 3

VignetteBuilder knitr

URL https://modeloriented.github.io/survex/

BugReports https://github.com/ModelOriented/survex/issues

NeedsCompilation no

Author Mikołaj Spytek [aut, cre], Mateusz Krzyziński [aut], Hubert Baniecki [aut] (<https://orcid.org/0000-0001-6661-5364>), Przemysław Biecek [aut] (<https://orcid.org/0000-0001-8423-1823>)

Maintainer Mikołaj Spytek <mikolajsyppek@gmail.com>

Repository CRAN

Date/Publication 2022-12-01 00:20:02 UTC
brier_score

Calculate Brier score

A function for calculating the Brier score for a survival model.
brier_score

Usage

brier_score(y_true = NULL, risk = NULL, surv = NULL, times = NULL)

loss_brier_score(y_true = NULL, risk = NULL, surv = NULL, times = NULL)

Arguments

y_true a survival::Surv object containing the times and statuses of observations for which the metric will be evaluated
risk ignored, left for compatibility with other metrics
surv a matrix containing the predicted survival functions for the considered observations, each row represents a single observation, whereas each column one time point
times a vector of time points at which the survival function was evaluated

Details

Brier score is used to evaluate the performance of a survival model, based on the squared distance between the predicted survival function and the actual event time, weighted to account for censored observations.

Value

numeric from 0 to 1, lower scores are better (Brier score of 0.25 represents a model which returns always returns 0.5 as the predicted survival function)

References


See Also

cd_auc()

Examples

library(survival)
library(survex)
cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)
y <- cph_exp$y
times <- cph_exp$times
surv <- cph_exp$predict_survival_function(cph, cph_exp$data, times)
cd_auc

Calculate Cumulative/Dynamic AUC

Description
This function calculates the Cumulative/Dynamic AUC metric for a survival model. It is done using the estimator proposed by Uno et al. [1], and Hung and Chang [2].

Usage
```r
cd_auc(y_true = NULL, risk = NULL, surv = NULL, times = NULL)
```

Arguments
- `y_true` a survival::Surv object containing the times and statuses of observations for which the metric will be evaluated
- `risk` ignored, left for compatibility with other metrics
- `surv` a matrix containing the predicted survival functions for the considered observations, each row represents a single observation, whereas each column one time point
- `times` a vector of time points at which the survival function was evaluated

Details
C/D AUC is an extension of the AUC metric known from classification models. Its values represent the model's performance at specific time points. It can be integrated over the considered time range.

Value
a numeric vector of length equal to the length of the times vector, each value (from the range from 0 to 1) represents the AUC metric at a specific time point, with higher values indicating better performance.

References

See Also
- `loss_one_minus_cd_auc()`
- `integrated_cd_auc()`
- `brier_score()`
cumulative_hazard_to_survival

Examples

library(survival)
library(survex)

  cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)

  y <- cph_exp$y
times <- cph_exp$times
surv <- cph_exp$predict_survival_function(cph, cph_exp$data, times)

  cd_auc(y, surv = surv, times = times)

---

cumulative_hazard_to_survival

  Transform Cumulative Hazard to Survival

Description

  Helper function to transform between CHF and survival function

Usage

  cumulative_hazard_to_survival(hazard_functions)

Arguments

  hazard_functions
  matrix or vector, with each row representing a cumulative hazard function

Value

  A matrix or vector transformed to the form of a survival function.

Examples

library(survex)

  vec <- c(1, 2, 3, 4, 5)
  matr <- matrix(c(1, 2, 3, 2, 4, 6), ncol = 3)

  cumulative_hazard_to_survival(vec)
cumulative_hazard_to_survival(matr)
c_index

Compute the Harrell’s Concordance index

Description
A function to compute the Harrell’s concordance index of a survival model.

Usage
c_index(y_true = NULL, risk = NULL, surv = NULL, times = NULL)

Arguments
- y_true: a survival::Surv object containing the times and statuses of observations for which the metric will be evaluated
- risk: a numeric vector of risk scores corresponding to each observation
- surv: ignored, left for compatibility with other metrics
- times: ignored, left for compatibility with other metrics

Value
numeric from 0 to 1, higher values indicate better performance

References

See Also
loss_one_minus_c_index()

Examples
library(survival)
library(survex)

rotterdam <- survival::rotterdam
rotterdam$year <- NULL
cox_rotterdam_rec <- coxph(Surv(rtime, recur) ~ .,
data = rotterdam,
model = TRUE, x = TRUE, y = TRUE)
coxph_explainer <- explain(cox_rotterdam_rec)
risk <- coxph_explainer$predict_function(coxph_explainer$model, coxph_explainer$data)
c_index(y_true = coxph_explainer$y, risk = risk)
explain_survival

A model-agnostic explainer for survival models

Description

Black-box models have vastly different structures. explain_survival() returns an explainer object that can be further processed for creating prediction explanations and their visualizations. This function is used to manually create explainers for models not covered by the survex package. For selected models the extraction of information can be done automatically. To do this, you can call the explain() function for survival models from mlr3proba, censored, randomForestSRC, ranger, survival packages and any other model with pec::predictSurvProb() method.

Usage

explain_survival(
  model,
  data = NULL,
  y = NULL,
  predict_function = NULL,
  predict_function_target_column = NULL,
  residual_function = NULL,
  weights = NULL,
  ..., 
  label = NULL,
  verbose = TRUE,
  colorize = !isTRUE(getOption("knitr.in.progress")),
  model_info = NULL,
  type = NULL,
  times = NULL,
  times_generation = "quantiles",
  predict_survival_function = NULL,
  predict_cumulative_hazard_function = NULL
)

explain(
  model,
  data = NULL,
  y = NULL,
  predict_function = NULL,
  predict_function_target_column = NULL,
  residual_function = NULL,
  weights = NULL,
  ..., 
  label = NULL,
  verbose = TRUE,
  colorize = !isTRUE(getOption("knitr.in.progress")),
  model_info = NULL,
## Default S3 method:
explain(
  model,
  data = NULL,
  y = NULL,
  predict_function = NULL,
  predict_function_target_column = NULL,
  residual_function = NULL,
  weights = NULL,
  ..., 
  label = NULL,
  verbose = TRUE,
  colorize = !isTRUE(getOption("knitr.in.progress")),
  model_info = NULL,
  type = NULL
)

### Arguments

- **model**: object - a survival model to be explained
- **data**: data.frame - data which will be used to calculate the explanations. If not provided, then it will be extracted from the model if possible. It should not contain the target columns. NOTE: If the target variable is present in the data some functionality breaks.
- **y**: survival::Surv object containing event/censoring times and statuses corresponding to data
- **predict_function**: function taking 2 arguments - model and newdata and returning a single number for each observation - risk score. Observations with higher score are more likely to observe the event sooner.
- **predict_function_target_column**: unused, left for compatibility with DALEX
- **residual_function**: unused, left for compatibility with DALEX
- **weights**: unused, left for compatibility with DALEX
- **...**: additional arguments, passed to DALEX::explain()
- **label**: character - the name of the model. Used to differentiate on visualizations with multiple explainers. By default it’s extracted from the 'class' attribute of the model if possible.
- **verbose**: logical, if TRUE (default) then diagnostic messages will be printed
- **colorize**: logical, if TRUE (default) then WARNINGS, ERRORS and NOTES are colorized. Will work only in the R console. By default it is FALSE while knitting and TRUE otherwise.
model_info: a named list (package, version, type) containing information about model. If NULL, survex will seek for information on its own.

type: type of a model, by default "survival"

times: numeric, a vector of times at which the survival function and cumulative hazard function should be evaluated for calculations.

times_generation: either "uniform" or "quantiles". Sets the way of generating the vector of times based on times provided in the y parameter. If "uniform" the vector contains 101 equally spaced points between the minimum and maximum observed times; if "quantiles" the vector contains 100 points between 0th and 99th percentiles of observed times. Ignored if times is not NULL.

predict_survival_function: function taking 3 arguments: model, newdata and times, and returning a matrix whose each row is a survival function evaluated at times for one observation from newdata.

predict_cumulative_hazard_function: function taking 3 arguments: model, newdata and times, and returning a matrix whose each row is a cumulative hazard function evaluated at times for one observation from newdata.

Value:

It is a list containing the following elements:

- model - the explained model.
- data - the dataset used for training.
- y - response for observations from data.
- residuals - calculated residuals.
- predict_function - function that may be used for model predictions, shall return a single numerical value for each observation.
- residual_function - function that returns residuals, shall return a single numerical value for each observation.
- class - class/classes of a model.
- label - label of explainer.
- model_info - named list containing basic information about model, like package, version of package and type.
- times - a vector of times, that are used for evaluation of survival function and cumulative hazard function by default.
- predict_survival_function - function that is used for model predictions in the form of survival function.
- predict_cumulative_hazard_function - function that is used for model predictions in the form of cumulative hazard function.
integrated_brier_score

Calculate integrated Brier score

Examples

```r
library(survival)
library(survex)

cph <- survival::coxph(survival::Surv(time, status) ~ ., data = veteran,
model = TRUE, x = TRUE)
cph_exp <- explain(cph)

rsf_ranger <- ranger::ranger(survival::Surv(time, status) ~ ., data = veteran,
respect.unordered.factors = TRUE, num.trees = 100, mtry = 3, max.depth = 5)
rsf_ranger_exp <- explain(rsf_ranger, data = veteran[, -c(3, 4)],
y = Surv(veteran$time, veteran$status))

rsf_src <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
rsf_src_exp <- explain(rsf_src)

library(censored, quietly = TRUE)

bt <- parsnip::boost_tree() %>%
parsnip::set_engine("mboost") %>%
parsnip::set_mode("censored regression") %>%
generics::fit(survival::Surv(time, status) ~ ., data = veteran)
bt_exp <- explain(bt, data = veteran[, -c(3, 4)], y = Surv(veteran$time, veteran$status))

#### explain_survival() ####

cph <- coxph(Surv(time, status) ~ ., data=veteran)

veteran_data <- veteran[, -c(3,4)]
veteran_y <- Surv(veteran$time, veteran$status)

risk_pred <- function(model, newdata) predict(model, newdata, type = "risk")
surv_pred <- function(model, newdata, times) pec::predictSurvProb(model, newdata, times)
chf_pred <- function(model, newdata, times) -log(surv_pred(model, newdata, times))

manual_cph_explainer <- explain_survival(model = cph,
data = veteran_data,
y = veteran_y,
predict_function = risk_pred,
predict_survival_function = surv_pred,
predict_cumulative_hazard_function = chf_pred,
label = "manual coxph")
```
Description

This function calculates the integrated Brier score metric for a survival model.

Usage

```r
describe_brier_score(y_true = NULL, risk = NULL, surv = NULL, times = NULL)
```

Arguments

- `y_true`: a `survival::Surv` object containing the times and statuses of observations for which the metric will be evaluated.
- `risk`: ignored, left for compatibility with other metrics.
- `surv`: a matrix containing the predicted survival functions for the considered observations, each row represents a single observation, whereas each column one time point.
- `times`: a vector of time points at which the survival function was evaluated.

Details

It is useful to see how a model performs as a whole, not at specific time points, for example for easier comparison. This function allows for calculating the integral of Brier score metric numerically using the trapezoid method.

Value

A numeric value from 0 to 1, lower values indicate better performance.

References


See Also

- `brier_score()`, `integrated_cd_auc()`, `loss_one_minus_integrated_cd_auc()`
Examples

```r
library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)

y <- cph_exp$y
times <- cph_exp$times
surv <- cph_exp$predict_survival_function(cph, cph_exp$data, times)

# calculating directly
integrated_brier_score(y, surv = surv, times = times)
```

---

**integrated_cd_auc**

*Calculate integrated C/D AUC*

**Description**

This function calculates the integrated Cumulative/Dynamic AUC metric for a survival model.

**Usage**

```r
integrated_cd_auc(y_true = NULL, risk = NULL, surv = NULL, times = NULL)
```

**Arguments**

- `y_true` a `survival::Surv` object containing the times and statuses of observations for which the metric will be evaluated
- `risk` ignored, left for compatibility with other metrics
- `surv` a matrix containing the predicted survival functions for the considered observations, each row represents a single observation, whereas each column one time point
- `times` a vector of time points at which the survival function was evaluated

**Details**

It is useful to see how a model performs as a whole, not at specific time points, for example for easier comparison. This function allows for calculating the integral of the C/D AUC metric numerically using the trapezoid method.
Value

numeric from 0 to 1, higher values indicate better performance

# References:


See Also

cd_auc() loss_one_minus_cd_auc()

Examples

library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)

y <- cph_exp$y
times <- cph_exp$times
surv <- cph_exp$predict_survival_function(cph, cph_exp$data, times)

integrated_cd_auc(y, surv = surv, times = times)
loss_integrate

Calculate integrated metrics based on time-dependent metrics.

Description

This function allows for creating a function for calculation of integrated metrics based on a time dependent metric. A possibility to cut off the data at certain quantiles is implemented, as well as weighting the integrated metric by max time and marginal survival function [1]

Usage

loss_integrate(loss_function, ..., normalization = NULL, max_quantile = 1)

Arguments

- **loss_function** • A time dependent loss function taking arguments (y_true, risk, surv, times)
- **...** • other parameters, currently ignored
- **normalization** • either NULL, "t_max" or "survival". Decides what kind of weighting should be applied to the integrated metric. If "t_max", then the integral is calculated using dw(t) where w(t) = t/t_max. If "survival", then the integral is calculated using dw(t) where w(t) = (1 - S(t))/(1 - S(t_max)) and S(t) denotes the estimated marginal survival function. If NULL (default), the integral is calculated using dt.
- **max_quantile** • a number from the interval (0,1]. The integral will be calculated only up to the time value of quantile(max_quantile) of the observed event/censoring times in y_true.

Value

a function that can be used to calculate metrics (with parameters y_true, risk, surv, and times)

References

loss_one_minus_cd_auc  
Calculate Cumulative/Dynamic AUC loss

Description
This function subtracts the C/D AUC metric from one to obtain a loss function whose lower values indicate better model performance (useful for permutational feature importance)

Usage
loss_one_minus_cd_auc(y_true = NULL, risk = NULL, surv = NULL, times = NULL)

Arguments
- **y_true**: a survival::Surv object containing the times and statuses of observations for which the metric will be evaluated
- **risk**: ignored, left for compatibility with other metrics
- **surv**: a matrix containing the predicted survival functions for the considered observations, each row represents a single observation, whereas each column one time point
- **times**: a vector of time points at which the survival function was evaluated

Value
a numeric vector of length equal to the length of the times vector, each value (from the range from 0 to 1) represents 1 - AUC metric at a specific time point, with lower values indicating better performance.

See Also
cd_auc()

Examples
library(survival)
library(survex)
cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)
loss_one_minus_c_index

Calculate the Concordance index loss

Description

This function subtracts the C-index metric from one to obtain a loss function whose lower values indicate better model performance (useful for permutational feature importance)

Usage

loss_one_minus_c_index(y_true = NULL, risk = NULL, surv = NULL, times = NULL)

Arguments

y_true a survival::Surv object containing the times and statuses of observations for which the metric will be evaluated
risk a numeric vector of risk scores corresponding to each observation
surv ignored, left for compatibility with other metrics
times ignored, left for compatibility with other metrics

Value

numeric from 0 to 1, lower values indicate better performance

References


See Also

c_index()
Examples

library(survival)
library(survex)

rotterdam <- survival::rotterdam
rotterdam$year <- NULL
cox_rotterdam_rec <- coxph(Surv(rtime, recur) ~ ., 
data = rotterdam, 
   model = TRUE, x = TRUE, y = TRUE)
coxph_explainer <- explain(cox_rotterdam_rec)

risk <- coxph_explainer$predict_function(coxph_explainer$model, coxph_explainer$data)
loss_one_minus_c_index(y_true = coxph_explainer$y, risk = risk)

loss_one_minus_integrated_cd_auc

*Calculate integrated C/D AUC loss*

Description

This function subtracts integrated the C/D AUC metric from one to obtain a loss function whose lower values indicate better model performance (useful for permutational feature importance)

Usage

```r
loss_one_minus_integrated_cd_auc(
  y_true = NULL, 
  risk = NULL, 
  surv = NULL, 
  times = NULL 
)
```

Arguments

- `y_true`: a `survival::Surv` object containing the times and statuses of observations for which the metric will be evaluated
- `risk`: ignored, left for compatibility with other metrics
- `surv`: a matrix containing the predicted survival functions for the considered observations, each row represents a single observation, whereas each column one time point
- `times`: a vector of time points at which the survival function was evaluated
Value

numeric from 0 to 1, lower values indicate better performance

# @section References:


See Also

integrated_cd_auc() cd_auc() loss_one_minus_cd_auc()

Examples

library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)

y <- cph_exp$y
times <- cph_exp$times
surv <- cph_exp$predict_survival_function(cph, cph_exp$data, times)

# calculating directly
loss_one_minus_integrated_cd_auc(y, surv = surv, times = times)
Arguments

explainer an explainer object - model preprocessed by the explain() function

... Arguments passed on to surv_feature_importance, surv_integrated_feature_importance

B numeric, number of permutations to be calculated

variables a character vector, names of variables to be included in the calculation

variable_groups a list of character vectors of names of explanatory variables. For each vector, a single variable-importance measure is computed for the joint effect of the variables which names are provided in the vector. By default, variable_groups = NULL, in which case variable-importance measures are computed separately for all variables indicated in the variables argument

label label of the model, if provides overrides x$label

loss_function a function that will be used to assess variable importance, by default loss_brier_score for survival models. The function can be supplied manually but has to have these named parameters (y_true, risk, surv, times), where y_true represents the survival::Surv object with observed times and statuses, risk is the risk score calculated by the model, and surv is the survival function for each observation evaluated at times.

type a character vector, if "raw" the results are losses after the permutation, if "ratio" the results are in the form loss/loss_full_model and if "difference" the results are of the form loss - loss_full_model

output_type either "survival" or "risk" the type of survival model output that should be used for explanations. If "survival" the explanations are based on the survival function. Otherwise the scalar risk predictions are used by the DALEX::model_profile function.

N number of observations that should be sampled for calculation of variable importance. If NULL then variable importance will be calculated on the whole dataset.

Details

Note: This function can be run within progressr::with_progress() to display a progress bar, as the execution can take long, especially on large datasets.

Value

An object of class c("model_parts_survival", "surv_feature_importance"). It’s a list with the explanations in the result element.
Examples

library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
rsf_ranger <- ranger::ranger(Surv(time, status) ~ .,
   data = veteran,
   respect.unordered.factors = TRUE,
   num.trees = 100,
   mtry = 3,
   max.depth = 5
)

cph_exp <- explain(cph)
rsf_ranger_exp <- explain(rsf_ranger,
   data = veteran[, -c(3, 4)],
   y = Surv(veteran$time, veteran$status)
)

cph_model_parts_brier <- model_parts(cph_exp)
print(head(cph_model_parts_brier$result))
plot(cph_model_parts_brier)

rsf_ranger_model_parts <- model_parts(rsf_ranger_exp)
print(head(rsf_ranger_model_parts$result))
plot(cph_model_parts_brier, rsf_ranger_model_parts)

---

model_performance  Dataset Level Performance Measures

Description

This function calculates metrics for survival models. The metrics calculated are C/D AUC, Brier score, and their integrated versions, as well as concordance index. It also can calculate ROC curves for specific selected time points.

Usage

model_performance(explainer, ...)

## S3 method for class 'surv_explainer'
model_performance(
   explainer,
   ...,
   type = "metrics",
)
model_performance

metrics = c('C-index' = c_index, 'Integrated C/D AUC' = integrated_cd_auc,
'Brier score' = brier_score, 'Integrated Brier score' = integrated_brier_score,
'C/D AUC' = cd_auc),
times = NULL

Arguments

explainer an explainer object - model preprocessed by the explain() function
... other parameters, currently ignored
type character, either "metrics" or "roc". If "metrics" then performance metrics
are calculated, if "roc" ROC curves for selected time points are calculated.
metrics a named vector containing the metrics to be calculated. The values should be
standardized loss functions. The functions can be supplied manually but has
to have these named parameters (y_true, risk, surv, times), where y_true
represents the survival::Surv object with observed times and statuses, risk
is the risk score calculated by the model, and surv is the survival function for
each observation evaluated at times.
times a numeric vector of times. If type == "metrics" then the survival function is
evaluated at these times, if type == "roc" then the ROC curves are calculated
at these times.

Value

An object of class "model_performance_survival". It's a list of metric values calculated for the
model. It contains:

- Harrell’s concordance index [1]
- Brier score [2, 3]
- C/D AUC using the estimator proposed by Uno et. al [4]
- integral of the Brier score
- integral of the C/D AUC

References

- [1] Harrell, F.E., Jr., et al. "Regression modelling strategies for improved prognostic predic-
- [2] Brier, Glenn W. "Verification of forecasts expressed in terms of probability." Monthly
Examples

```r
library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
rsf_ranger <- ranger::ranger(Surv(time, status) ~ .,
                              data = veteran,
                              respect.unordered.factors = TRUE,
                              num.trees = 100,
                              mtry = 3,
                              max.depth = 5)

rsf_src <- randomForestSRC::rfsrc(Surv(time, status) ~ .,
                                  data = veteran)

cph_exp <- explain(cph)
rsf_ranger_exp <- explain(rsf_ranger, data = veteran[, -c(3, 4)],
                          y = Surv(veteran$time, veteran$status))
rsf_src_exp <- explain(rsf_src)

cph_model_performance <- model_performance(cph_exp)
rsf_ranger_model_performance <- model_performance(rsf_ranger_exp)
rsf_src_model_performance <- model_performance(rsf_src_exp)

print(cph_model_performance)
plot(rsf_ranger_model_performance, cph_model_performance,
     rsf_src_model_performance, metrics_type = "scalar")

plot(rsf_ranger_model_performance, cph_model_performance, rsf_src_model_performance)

cph_model_performance_roc <- model_performance(cph_exp, type = "roc", times = c(100, 250, 500))
plot(cph_model_performance_roc)
```

---

**model_profile**

*Dataset Level Variable Profile as Partial Dependence Explanations for Survival Models*

**Description**

This function calculates explanations on a dataset level that help explore model response as a function of selected variables. The explanations are calculated as an extension of Partial Dependence Profiles with the inclusion of the time dimension.
Usage

model_profile(
  explainer,
  variables = NULL,
  N = 100,
  ..., 
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "partial",
  output_type = "survival"
)

## S3 method for class 'surv_explainer'
model_profile(
  explainer,
  variables = NULL,
  N = 100,
  ..., 
  categorical_variables = NULL,
  grid_points = 51,
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "partial",
  output_type = "survival"
)

Arguments

explainer an explainer object - model preprocessed by the explain() function
variables character, a vector of names of variables to be explained
N number of observations used for the calculation of aggregated profiles. By de-
   fault 100. If NULL all observations are used.
... other parameters passed to DALEX::model_profile if output_type == "risk", other-
   wise ignored
groups if output_type == "risk" a variable name that will be used for grouping. By
default NULL, so no groups are calculated. If output_type == "survival" then
   ignored
k passed to DALEX::model_profile if output_type == "risk", otherwise igno-
   red
center logical, should profiles be centered before clustering
type the type of variable profile. If output_type == "survival" then only "partial"
   is implemented, otherwise passed to DALEX::model_profile.
output_type either "survival" or "risk" the type of survival model output that should
   be considered for explanations. If "survival" the explanations are based on
the survival function. Otherwise the scalar risk predictions are used by the DALEX::model_profile function.

categorical_variables
character, a vector of names of additional variables which should be treated as categorical (factors are automatically treated as categorical variables)

grid_points
maximum number of points for profile calculations. Note that the final number of points may be lower than grid_points. Will be passed to internal function. By default 51.

Value
An object of class model_profile_survival. It is a list with the element result containing the results of the calculation.

Examples

library(survival)
library(survex)
cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
rsf_src <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
cph_exp <- explain(cph)
rsf_src_exp <- explain(rsf_src)
cph_model_profile <- model_profile(cph_exp, output_type = "survival",
variables = c("age"))
head(cph_model_profile$result)
plot(cph_model_profile)
rsf_model_profile <- model_profile(rsf_src_exp, output_type = "survival",
variables = c("age", "celltype"))
head(rsf_model_profile$result)
plot(rsf_model_profile, variables = c("age", "celltype"), numerical_plot_type = "contours")
**Description**

This function plots variable importance calculated as changes in the loss function after variable drops. It uses output from `feature_importance` function that corresponds to permutation based measure of variable importance. Variables are sorted in the same order in all panels. The order depends on the average drop out loss. In different panels variable contributions may not look like sorted if variable importance is different in different in different models.

**Usage**

```r
## S3 method for class 'feature_importance_explainer'
plot(
  x,
  ...,
  max_vars = NULL,
  show_boxplots = TRUE,
  bar_width = 10,
  desc_sorting = TRUE,
  title = "Feature Importance",
  subtitle = NULL
)
```

**Arguments**

- `x` a feature importance explainer produced with the `feature_importance()` function
- `...` other explainers that shall be plotted together
- `max_vars` maximum number of variables that shall be presented for each model. By default NULL what means all variables
- `show_boxplots` logical if TRUE (default) boxplot will be plotted to show permutation data.
- `bar_width` width of bars. By default 10
- `desc_sorting` logical. Should the bars be sorted descending? By default TRUE
- `title` the plot’s title, by default 'Feature Importance'
- `subtitle` the plot’s subtitle. By default - NULL, which means the subtitle will be 'created for the XXX model’, where XXX is the label of explainer(s)

**Details**

Find more details in the Feature Importance Chapter.

**Value**

a ggplot2 object

**References**

Examples

```r
library(survec)
library(randomForestSRC)
library(survival)

model <- rfsrc(Surv(time, status) ~ ., data = veteran)
explainer <- explain(model)

mp <- model_parts(explainer, loss = loss_one_minus_c_index, output_type = "risk")
plot(mp)
```

---

**plot.model_parts_survival**

*Plot Model Parts for Survival Models*

Description

This function is a wrapper for plotting `model_parts` objects created for survival models using the `model_parts()` function.

Usage

```r
## S3 method for class 'model_parts_survival'
plot(x, ...)
```

Arguments

- `x` an object of class "model_parts_survival" to be plotted
- `...` additional parameters passed to the `plot.surv_feature_importance` function

Value

An object of the class ggplot.

Plot options

- `title` character, title of the plot
- `subtitle` character, subtitle of the plot, if NULL automatically generated as "created for XXX, YYY models", where XXX and YYY are explainer labels
- `max_vars` maximum number of variables to be plotted (least important variables are ignored)
- `colors` character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

See Also

Other functions for plotting 'model_parts_survival' objects: `plot.surv_feature_importance()`
Examples

```r
library(survival)
library(survex)

model <- coxph(Surv(time, status) ~ ., data = veteran, x = TRUE, model = TRUE, y = TRUE)
explainer <- explain(model)

mp <- model_parts(explainer)

plot(mp)
```

---

plot.model_performance_survival

**Plot Model Performance for Survival Models**

Description

This function is a wrapper for plotting `model_performance` objects created for survival models using the `model_performance()` function.

Usage

```r
## S3 method for class 'model_performance_survival'
plot(x, ...)
```

Arguments

- `x` an object of class "model_performance_survival" to be plotted
- `...` additional parameters passed to the `plot.surv_model_performance` or `plot.surv_model_performance_rocs` function

Value

An object of the class ggplot.

Plot options

- `plot.surv_model_performance`:
  - `x` - an object of class "surv_model_performance" to be plotted
  - `...` - additional objects of class "surv_model_performance" to be plotted together
  - `metrics` - character, names of metrics to be plotted (subset of C/D AUC", "Brier score" for `metrics_type %in% c("time_dependent", "functional")` or subset of "C-index","Integrated Brier score", "Integrated C/D AUC" for `metrics_type == "scalar"`), by default (NULL) all metrics of a given type are plotted
  - `metrics_type` - character, either one of c("time_dependent","functional") for functional metrics or "scalar" for scalar metrics
plot.model_profile_survival

• title - character, title of the plot
• subtitle - character, subtitle of the plot. 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
• facet_ncol - number of columns for arranging subplots
• colors - character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

plot.surv.model_performance_rocs:
• x - an object of class "surv_model_performance_rocs" to be plotted
• ... - additional objects of class "surv_model_performance_rocs" to be plotted together
• title - character, title of the plot
• subtitle - character, subtitle of the plot. 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
• colors - character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")
• facet_ncol - number of columns for arranging subplots

See Also

Other functions for plotting 'model_performance_survival' objects: plot.surv.model_performance_rocs(), plot.surv.model.performance()

Examples

library(survival)
library(survex)

model <- randomForestSRC::rfsr(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

m_perf <- model_performance(exp)
plot(m_perf, metrics_type = "functional")

m_perf_roc <- model_performance(exp, type = "roc", times = c(100, 300))
plot(m_perf_roc)

plot.model_profile_survival

Plot Model Profile for Survival Models

Description

This function plots objects of class "model_profile_survival" created using the model_profile() function.
## Usage

```r
## S3 method for class 'model_profile_survival'
plot(
  x,
  ..., 
  variables = NULL,
  variable_type = NULL,
  facet_ncol = NULL,
  numerical_plot_type = "lines",
  title = "Partial dependence survival profile",
  subtitle = "default",
  colors = NULL
)
```

## Arguments

- `x` an object of class `model_profile_survival` to be plotted
- `...` additional parameters, unused, currently ignored
- `variables` character, names of the variables to be plotted
- `variable_type` character, either "numerical", "categorical" or `NULL` (default), select only one type of variable for plotting, or leave `NULL` for all
- `facet_ncol` number of columns for arranging subplots
- `numerical_plot_type` character, either "lines", or "contours" selects the type of numerical variable plots
- `title` character, title of the plot
- `subtitle` character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
- `colors` character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

## Value

A grid of ggplot objects arranged with the `gridExtra::grid.arrange` function.

## Examples

```r
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

m_prof <- model_profile(exp, categorical_variables = "trt")

plot(m_prof)
```
plot(m_prof, numerical_plot_type = "contours")

plot(m_prof, variables = c("trt", "age"), facet_ncol = 1)

---

plot.predict_parts_survival

Plot Predict Parts for Survival Models

Description

This function plots objects of class "predict_parts_survival" - local explanations for survival models created using the `predict_parts()` function.

Usage

```r
## S3 method for class 'predict_parts_survival'
plot(x, ...)
```

Arguments

- `x` - an object of class "predict_parts_survival" to be plotted
- `...` - additional parameters passed to the `plot.surv_shap` or `plot.surv_lime` functions

Value

An object of the class `ggplot`.

Plot options

- `plot.surv_shap`:
  - `x` - an object of class "surv_shap" to be plotted
  - `...` - additional objects of class `surv_shap` to be plotted together
  - `title` - character, title of the plot
  - `subtitle` - character, subtitle of the plot, "default" automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
  - `colors` - character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

- `plot.surv_lime`:
  - `x` - an object of class "surv_lime" to be plotted
  - `type` - character, either "coefficients" or "local_importance", selects the type of plot
  - `show_survival_function` - logical, if the survival function of the explanations should be plotted next to the barplot
• ... - other parameters currently ignored
• title - character, title of the plot
• subtitle - character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
• colors - character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

**See Also**

Other functions for plotting 'predict_parts_survival' objects: `plot.surv_lime()`, `plot.surv_shap()`

**Examples**

```r
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

p_parts_shap <- predict_parts(exp, veteran[1, -c(3, 4)], type = "survshap")
plot(p_parts_shap)

p_parts_lime <- predict_parts(exp, veteran[1, -c(3, 4)], type = "survlime")
plot(p_parts_lime)
```

---

**plot.predict_profile_survival**  
*Plot Predict Profile for Survival Models*

**Description**

This function plots objects of class "predict_profile_survival" - local explanations for survival models created using the `predict_profile()` function.

**Usage**

```r
## S3 method for class 'predict_profile_survival'
plot(x, ...)
```

**Arguments**

- `x` - an object of class "predict_profile_survival" to be plotted
- `...` - additional parameters passed to the `plot.surv_ceteris_paribus` function

**Value**

A grid of `ggplot` objects arranged with the `gridExtra::grid.arrange` function.
Plot options

plot.surv.ceteris_paribus:

- `x` - an object of class `predict_profile_survival` to be plotted
- ... - additional parameters, unused, currently ignored
- `colors` - character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")
- `variable_type` - character, either "numerical", "categorical" or NULL (default), select only one type of variable for plotting, or leave NULL for all
- `facet_ncol` - number of columns for arranging subplots
- `variables` - character, names of the variables to be plotted
- `numerical_plot_type` - character, either "lines", or "contours" selects the type of numerical variable plots
- `title` - character, title of the plot
- `subtitle` - character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels

See Also

Other functions for plotting 'predict_profile_survival' objects: `plot.surv.ceteris_paribus()`

Examples

```r
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

p_profile <- predict_profile(exp, veteran[1, -c(3, 4)])
plot(p_profile)

p_profile_with_cat <- predict_profile(exp,
  veteran[1, -c(3, 4)],
  categorical_variables = c("trt", "prior")
)
plot(p_profile_with_cat)
```
plot.surv_ceteris_paribus

Description

This function plots objects of class "predict_profile_survival" created using the predict_profile() function.

Usage

## S3 method for class 'surv_ceteris_paribus'
plot(
  x,
  ..., colors = NULL,
  variable_type = NULL,
  facet_ncol = NULL,
  variables = NULL,
  numerical_plot_type = "lines",
  title = "Ceteris paribus survival profile",
  subtitle = "default"
)

Arguments

x an object of class predict_profile_survival to be plotted
... additional parameters, unused, currently ignored
colors character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")
variable_type character, either "numerical", "categorical" or NULL (default), select only one type of variable for plotting, or leave NULL for all
facet_ncol number of columns for arranging subplots
variables character, names of the variables to be plotted
numerical_plot_type character, either "lines", or "contours" selects the type of numerical variable plots
title character, title of the plot
subtitle character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels

Value

A grid of ggplot objects arranged with the gridExtra::grid.arrange function.

See Also

Other functions for plotting 'predict_profile_survival' objects: plot.predict_profile_survival()
**Examples**

```r
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

p_profile <- predict_profile(exp, veteran[1, -c(3, 4)])
plot(p_profile)

p_profile_with_cat <- predict_profile(
  exp,
  veteran[1, -c(3, 4)],
  categorical_variables = c("trt", "prior")
)

plot(p_profile_with_cat)
```

---

**plot.surv_feature_importance**

*Plot Permuatational Feature Importance for Survival Models*

**Description**

This function plots feature importance objects created for survival models using the `model_parts()` function with a time-dependent metric, that is `loss_one_minus_cd_auc()` or `loss_brier_score()`.

**Usage**

```r
## S3 method for class 'surv_feature_importance'
plot(
  x,
  ..., 
  title = "Time-dependent feature importance",
  subtitle = "default",
  max_vars = 6,
  colors = NULL
)
```

**Arguments**

- `x` an object of class "surv_feature_importance" to be plotted
- `...` additional objects of class "surv_feature_importance" to be plotted together
- `title` character, title of the plot
subtitle character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
max_vars maximum number of variables to be plotted (least important variables are ignored)
colors character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

Value
An object of the class ggplot.

See Also
Other functions for plotting 'model_parts_survival' objects: plot.model_parts_survival()

Examples

library(survival)
library(survelx)

model <- coxph(Surv(time, status) ~ ., data = veteran, x = TRUE, model = TRUE, y = TRUE)
model_rf <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
explainer <- explain(model)
explainer_rf <- explain(model_rf)

mp <- model_parts(explainer)
mp_rf <- model_parts(explainer_rf)

plot(mp, mp_rf)
Arguments

x

an object of class "surv_lime" to be plotted

type

character, either "coefficients" or "local_importance" (default), selects the type of plot

show_survival_function

logical, if the survival function of the explanations should be plotted next to the barplot

... other parameters currently ignored

title

character, title of the plot

subtitle

character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels

colors

character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

Value

An object of the class ggplot.

See Also

Other functions for plotting 'predict_parts_survival' objects: plot.predict_parts_survival(), plot.surv_shap()

Examples

library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

p_parts_lime <- predict_parts(exp, veteran[1, -c(3, 4)], type = "survlime")
plot(p_parts_lime)
plot.surv_model_performance

Plot Model Performance Metrics for Survival Models

Description

This function plots objects of class "surv_model_performance" - visualization of metrics of different models created using the model_performance(..., type="metrics") function.

Usage

## S3 method for class 'surv_model_performance'
plot(
  x,
  ..., 
  metrics = NULL,
  metrics_type = "time_dependent",
  title = "Model performance",
  subtitle = "default",
  facet_ncol = NULL,
  colors = NULL
)

Arguments

x
an object of class "surv_model_performance" to be plotted

... additional objects of class "surv_model_performance" to be plotted together

metrics character, names of metrics to be plotted (subset of C/D AUC", "Brier score" for metrics_type %in% c("time_dependent", "functional") or subset of "C-index", "Integrated Brier score", "Integrated C/D AUC" for metrics_type == "scalar"), by default (NULL) all metrics of a given type are plotted

metrics_type character, either one of c("time_dependent","functional") for functional metrics or "scalar" for scalar metrics

title character, title of the plot

subtitle character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels

facet_ncol number of columns for arranging subplots

colors character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

Value

An object of the class ggplot.
See Also

Other functions for plotting 'model_performance_survival' objects: plot.model_performance_survival(), plot.surv_model_performance_rocs()

Examples

```r
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)
m_perf <- model_performance(exp)
plot(m_perf)
```

### plot.surv_model_performance_rocs

**Plot ROC Curves for Survival Models**

**Description**

This function plots objects of class "surv_model_performance_rocs" - ROC curves for specific time points for survival models created using the \texttt{model_performance(..., type="roc")}.

**Usage**

```r
## S3 method for class 'surv_model_performance_rocs'
plot(
  x,
  ...,
  title = "ROC curves for selected timepoints",
  subtitle = "default",
  colors = NULL,
  facet_ncol = NULL
)
```

**Arguments**

- \texttt{x} an object of class "surv_model_performance_rocs" to be plotted
- \texttt{...} additional objects of class "surv_model_performance_rocs" to be plotted together
- \texttt{title} character, title of the plot
- \texttt{subtitle} character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
- \texttt{colors} character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")
- \texttt{facet_ncol} number of columns for arranging subplots
plot.surv_shap

Value

An object of the class ggplot.

See Also

Other functions for plotting 'model_performance_survival' objects: plot.model_performance_survival(), plot.surv_model_performance()

Examples

```r
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

m_perf_roc <- model_performance(exp, type = "roc", times = c(100, 300))
plot(m_perf_roc)
```

---

**plot.surv_shap**

*Plot SurvSHAP(t) Explanations for Survival Models*

Description

This function plots objects of class surv_shap - time-dependent explanations of survival models created using the predict_parts(..., type="survshap") function.

Usage

```r
## S3 method for class 'surv_shap'
plot(x, ..., title = "SurvSHAP(t)", subtitle = "default", colors = NULL)
```

Arguments

- `x`: an object of class "surv_shap" to be plotted
- `...`: additional objects of class surv_shap to be plotted together
- `title`: character, title of the plot
- `subtitle`: character, subtitle of the plot, 'default' automatically generates "created for XXX, YYY models", where XXX and YYY are the explainer labels
- `colors`: character vector containing the colors to be used for plotting variables (containing either hex codes "#FF69B4", or names "blue")

Value

An object of the class ggplot.
predict.surv_explainer

Model Predictions for Survival Models

Description

This function allows for calculating model prediction in a unified way.

Usage

## S3 method for class 'surv_explainer'
predict(object, newdata = NULL, output_type = "survival", times = NULL, ...)

Arguments

- **object**: an explainer object - model preprocessed by the `explain()` function
- **newdata**: data used for the prediction
- **output_type**: character, either "risk", "survival" or "chf" depending on the desired output
- **times**: a numeric vector of times for the survival and cumulative hazard function predictions to be evaluated at. If "output_type == "risk" this argument is ignored, if left NULL then it is extracted from `object$times`.
- ...: other arguments, currently ignored

Value

A vector or matrix containing the prediction.

See Also

Other functions for plotting 'predict_parts_survival' objects: `plot.predict_parts_survival()`, `plot.surv_lime()`

Examples

```
library(survival)
library(survex)

model <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
exp <- explain(model)

p_parts_shap <- predict_parts(exp, veteran[1, -c(3, 4)], type = "survshap")
plot(p_parts_shap)
```
Examples

```r
library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
rsf_ranger <- ranger::ranger(Surv(time, status) ~ .,
data = veteran,
  respect.unordered.factors = TRUE,
  num.trees = 100,
  mtry = 3,
  max.depth = 5)

cph_exp <- explain(cph)
rsf_ranger_exp <- explain(rsf_ranger, data = veteran[, -c(3, 4)],
y = Surv(veteran$time, veteran$status))

predict(cph_exp, veteran[1, ], output_type = "survival")[, 1:10]
predict(cph_exp, veteran[1, ], output_type = "risk")
predict(rsf_ranger_exp, veteran[1, ], output_type = "chf")[, 1:10]
```

### predict_parts

**Instance Level Parts of Survival Model Predictions**

**Description**

This function decomposes the model prediction into individual parts, which are attributions of particular variables. The explanations can be made via the SurvLIME and SurvSHAP(t) methods.

**Usage**

```r
predict_parts(explainer, ...)
```

`## S3 method for class 'surv_explainer`
```
```r
predict_parts(
  explainer,
  new_observation,
  ...,
  N = NULL,
  type = "survshap",
  output_type = "survival"
)
```
Arguments

explainer  
an explainer object - model preprocessed by the explain() function

...  
other parameters which are passed to iBreakDown::break_down if output_type="risk", 
or if output_type="survival" to surv_shap() or surv_lime() functions 
depending on the selected type

new_observation  
a new observation for which prediction need to be explained

N  
the maximum number of observations used for calculation of attributions. If 
NULL (default) all observations will be used.

type  
if output_type == "survival" must be either "survshap" or "survlime", 
otherwise refer to the DALEX::predict_parts

output_type  
either "survival" or "risk" the type of survival model output that should 
be considered for explanations. If "survival" the explanations are based on 
the survival function. Otherwise the scalar risk predictions are used by the 
DALEX::predict_parts function.

Value

An object of class "predict_parts_survival" and additional classes depending on the type of 
explanations. It is a list with the element result containing the results of the calculation.

Additional parameters

There are additional parameters that are passed to internal functions

• for surv_lime
  - N - a positive integer, number of observations generated in the neighbourhood
  - distance_metric - character, name of the distance metric to be used, only "euclidean" 
is implemented
  - kernel_width - a numeric, parameter used for calculating weights, by default it's sqrt(ncol(data)*0.75)
  - sampling_method - character, name of the method of generating neighbourhood, only 
"gaussian" is implemented
  - sample_around_instance - logical, if the neighbourhood should be generated with the 
new observation as the center (default), or should the mean of the whole dataset be used 
as the center
  - max_iter - a numeric, maximal number of iteration for the optimization problem
  - categorical_variables - character vector, names of variables that should be treated as 
categories (factors are included by default)
  - k - a small positive number > 1, added to chf before taking log, so that weights aren't 
negative

• for surv_shap
  - timestamps - a numeric vector, time points at which the survival function will be evalu-
ated
  - y_true - a two element numeric vector or matrix of one row and two columns, the first 
element being the true observed time and the second the status of the observation, used 
for plotting
- calculation_method - a character, only "kernel" is implemented for now.
- aggregation_method - a character, either "mean_absolute" or "integral", "max_absolute", "sum_of_squares"

References


Examples

```r
library(survival)
library(survex)

cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
cph_exp <- explain(cph)

cph_predict_parts_survshap <- predict_parts(cph_exp, new_observation = veteran[1, -c(3, 4)])
head(cph_predict_parts_survshap$result)
plot(cph_predict_parts_survshap)

cph_predict_parts_survlime <- predict_parts(cph_exp, new_observation = veteran[1, -c(3, 4)],
                                            type = "survlime")
head(cph_predict_parts_survlime$result)
plot(cph_predict_parts_survlime, type = "local_importance")
```

**predict_profile**  
*Instance Level Profile as Ceteris Paribus for Survival Models*

**Description**

This function calculates Ceteris Paribus Profiles for a specific observation with the possibility to take the time dimension into account.

**Usage**

```r
predict_profile(
    explainer,
    new_observation,
    variables = NULL,
    categorical_variables = NULL,
    ...
    type = "ceteris_paribus",
)```
predict_profile

```r
variable_splits_type = "uniform"

## S3 method for class 'surv_explainer'
predict_profile(
  explainer,
  new_observation,
  variables = NULL,
  categorical_variables = NULL,
  ...,
  type = "ceteris_paribus",
  output_type = "survival",
  variable_splits_type = "uniform"
)
```

**Arguments**

- `explainer`: An explainer object - model preprocessed by the `explain()` function.
- `new_observation`: A new observation for which the prediction needs to be explained.
- `variables`: A character vector containing names of variables to be explained.
- `categorical_variables`: A character vector of names of additional variables which should be treated as categorical (factors are automatically treated as categorical variables).
- `...`: Additional parameters passed to `DALEX::predict_profile` if `output_type` == "risk".
- `type`: Character, only "ceteris_paribus" is implemented.
- `variable_splits_type`: Character, decides how variable grids should be calculated. Use "quantiles" for percentiles or "uniform" (default) to get uniform grid of points.
- `output_type`: Either "survival" or "risk" the type of survival model output that should be considered for explanations. If "survival" the explanations are based on the survival function. Otherwise the scalar risk predictions are used by the `DALEX::predict_profile` function.

**Value**

An object of class `c("predict_profile_survival", "surv_ceteris_paribus")`. It is a list with the final result in the `result` element.

**Examples**

```r
library(survival)
library(survex)
cph <- coxph(Surv(time, status) ~ ., data = veteran, model = TRUE, x = TRUE, y = TRUE)
rsf_src <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
```
risk_from_chf

Generate Risk Prediction based on the Survival Function

Description
Some models do not come with a ready to use risk prediction. This function allows for its generation based on the cumulative hazard function.

Usage
risk_from_chf(predict_cumulative_hazard_function, times)

Arguments
predict_cumulative_hazard_function
  a function of three arguments (model, newdata, times) that allows for making cumulative hazard predictions.
times
  a numeric vector of times at which the function should be evaluated.

Value
A function of two arguments (model, newdata) returning a vector of risks.

Examples
library(survelx)
library(survival)
rsf_src <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)
chf_function <- transform_to_stepfunction(predict,
  type = "chf",
  prediction_element = "chf",
  times_element = "time.interest")
risk_function <- risk_from_chf(chf_function, unique(veteran$time))
survival_to_cumulative_hazard

Transform Survival to Cumulative Hazard

Description

Helper function to transform between survival function and CHF

Usage

survival_to_cumulative_hazard(survival_functions, epsilon = 0)

Arguments

- **survival_functions**: matrix or vector, with each row representing a survival function
- **epsilon**: a positive numeric number to add, so that the logarithm can be taken

Value

A matrix or vector transformed to the form of a cumulative hazard function.

Examples

library(survex)

vec <- c(1, 0.9, 0.8, 0.7, 0.6)
matr <- matrix(c(1, 0.9, 0.8, 1, 0.8, 0.6), ncol = 3)

survival_to_cumulative_hazard(vec)
survival_to_cumulative_hazard(matr)
surv_model_info

Extract additional information from the model

Description
This generic function lets users extract base information about a model. The function returns a named list of class `model_info` that contains information about the package of the model, version, and task type. For wrappers like `mlr` or `parsnip`, both, package and wrapper information are stored.

Usage

```r
surv_model_info(model, ...)  
## S3 method for class 'coxph'
surv_model_info(model, ...)  
## S3 method for class 'rfsRC'
surv_model_info(model, ...)  
## S3 method for class 'ranger'
surv_model_info(model, ...)  
## S3 method for class 'model_fit'
surv_model_info(model, ...)  
## S3 method for class 'cph'
surv_model_info(model, ...)  
## S3 method for class 'LearnerSurv'
surv_model_info(model, ...)  
## Default S3 method:
surv_model_info(model, ...)
```

Arguments

- `model` • model object
- `...` • other arguments

Details
Currently supported packages are:

- class `coxph` - Cox proportional hazards regression model created with `survival` package
- class `model_fit` - models created with `parsnip` package
- class `ranger` - random survival forest models created with `ranger` package
- class `rfsRC` - random forest models created with `randomForestSRC` package
Value
A named list of class `model_info`

Examples
```r
library(survival)
library(survex)
cph <- survival::coxph(survival::Surv(time, status) ~ ., data = veteran,
                     model = TRUE, x = TRUE, y = TRUE)
surv_model_info(cph)

library(ranger)
rsf_ranger <- ranger::ranger(survival::Surv(time, status) ~ ., data = veteran,
                            num.trees = 50, mtry = 3, max.depth = 5)
surv_model_info(rsf_ranger)
```

transform_to_stepfunction

*Transform Fixed Point Prediction into a Stepfunction*

Description
Some models return the survival function or cumulative hazard function prediction at the times of events present in the training data set. This is a convenient utility to allow the prediction to be evaluated at any time.

Usage
```r
transform_to_stepfunction(
    predict_function,
    eval_times = NULL,
    ..., type = NULL,
    prediction_element = NULL,
    times_element = NULL
)
```

Arguments
- `predict_function`:
a function making the prediction based on `model` and `newdata` arguments, the `...` parameter is also passed to this function. It has to return either a numeric vector of the same length as `eval_times`, a matrix with this number of columns and the same number of rows as `nrow(newdata)`. It can also return a list, with one of the elements containing such an object.
transform_to_stepfunction

**eval_times**
a numeric vector of times, at which the fixed predictions are made. This can be NULL, if predict_function returns a list which contains such a vector.

**...**
other parameters passed to predict_function

**type**
the type of function to be returned, either "survival", "chf" or NULL this chooses the value of the step function before the first prediction time. If "survival" then it is 1, if "chf" then 0, otherwise, it is the value of the prediction for the first time in numerical order.

**prediction_element**
if predict_function returns a list with the matrix as one of its elements, this parameter should contain the name of this element

**times_element**
if predict_function returns a list with the matrix as one of its elements, this parameter should contain the name of this element

**Value**
The function returns a function with three arguments, (model, newdata, times), ready to supply it to an explainer.

**Examples**

```r
library(survex)
library(survival)

rsf_src <- randomForestSRC::rfsrc(Surv(time, status) ~ ., data = veteran)

chf_function <- transform_to_stepfunction(predict,
                                          type = "chf",
                                          prediction_element = "chf",
                                          times_element = "time.interest")

explainer <- explain(rsf_src, predict_cumulative_hazard_function = chf_function)
```

Index

* functions for plotting
  'model_parts_survival' objects
  plot.model_parts_survival, 26
  plot.surv_feature_importance, 34
* functions for plotting
  'model_performance_survival' objects
  plot.model_performance_survival, 27
  plot.surv_model_performance, 37
  plot.surv_model_performance_rocs, 38
* functions for plotting
  'predict_parts_survival' objects
  plot.predict_parts_survival, 30
  plot.surv_lime, 35
  plot.surv_shap, 39
* functions for plotting
  'predict_profile_survival' objects
  plot.predict_profile_survival, 31
  plot.surv_ceteris_paribus, 32
brier_score, 2
brier_score(), 4, 11
c_index, 6
c_index(), 16
cd_auc, 4
cd_auc(), 3, 13, 15, 18
cumulative_hazard_to_survival, 5
explain(explain_survival), 7
explain_survival, 7
integrated_brier_score, 10
integrated_cd_auc, 12
integrated_cd_auc(), 4, 11, 18
loss_adapt_mlr3proba, 13
loss_brier_score(brier_score), 2
loss_integrate, 14
loss_integrated_brier_score
(integrated_brier_score), 10
loss_one_minus_c_index, 16
loss_one_minus_c_index(), 6
loss_one_minus_cd_auc, 15
loss_one_minus_cd_auc(), 4, 13, 18
loss_one_minus_integrated_cd_auc, 17
loss_one_minus_integrated_cd_auc(), 11
model_parts, 18
model_performance, 20
model_profile, 22
plot.feature_importance_explainer, 24
plot.model_parts_survival, 26, 35
plot.model_performance_survival, 27, 38, 39
plot.model_profile_survival, 28
plot.predict_parts_survival, 30, 36, 40
plot.predict_profile_survival, 31, 33
plot.surv_ceteris_paribus, 32, 32
plot.surv_feature_importance, 26, 34
plot.surv_lime, 31, 35, 40
plot.surv_model_performance, 28, 37, 39
plot.surv_model_performance_rocs, 28, 38, 38
plot.surv_shap, 31, 36, 39
predict.surv_explainer, 40
predict_parts, 41
predict_profile, 43
risk_from_chf, 45
surv_feature_importance, 19
survIntegratedFeatureImportance, 19
surv_model_info, 47
survival_to_cumulative_hazard, 46
transform_to_stepfunction, 48