Package 'mlr3proba'

January 31, 2020

Title Probabilistic Supervised Learning for 'mlr3'

Version 0.1.2

Description Provides extensions for probabilistic supervised learning for 'mlr3'. This includes extending the regression task to probabilistic and interval regression, adding a survival task, and other specialized models, predictions, and measures.

License MIT + file LICENSE

URL https://mlr3proba.mlr-org.com, https://github.com/mlr-org/mlr3proba

BugReports https://github.com/mlr-org/mlr3proba/issues

Depends R (>= 3.1.0)

Imports checkmate, data.table, distr6 (>= 1.3.3), mboost, mlr3 (>= 0.1.4), mlr3misc (>= 0.1.5), mlr3pipelines, paradox (>= 0.1.0), R6, survival

Suggests bibtex, flexsurv, gbm, glmnet, knitr, lgr, mlr3tuning, pec, penalized, randomForestSRC, ranger, riskRegression, rpart, simsurv, survAUC, survivalsvm, testthat

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 7.0.2

RdMacros mlr3misc

Collate 'LearnerDensity.R' 'LearnerDensityHist.R' 'LearnerDensityKDE.R' 'LearnerProbreg.R' 'LearnerProbregGaussian.R' 'LearnerSurv.R' 'LearnerSurvBlackboost.R' 'LearnerSurvCVGlmnet.R' 'LearnerSurvCoxPH.R' 'predict_flexsurvreg.R' 'LearnerSurvFlexible.R' 'LearnerSurvGBM.R'

'LearnerSurvGamboost.R' 'LearnerSurvGlmboost.R'
'LearnerSurvGlmnet.R' 'LearnerSurvKaplan.R'
'LearnerSurvMboost.R' 'LearnerSurvNelson.R' 'predict_survreg.R'
'LearnerSurvParametric.R' 'LearnerSurvPenalized.R'
'LearnerSurvRandomForestSRC.R' 'LearnerSurvRanger.R'
'LearnerSurvRpart.R' 'LearnerSurvSVM.R' 'MeasureDensity.R'
'MeasureDensityLogloss.R' 'MeasureRegrLogloss.R'
'MeasureSurv.R' 'MeasureSurvIntegrated.R' 'MeasureSurvAUC.R'
'MeasureSurvBeggC.R' 'MeasureSurvChamblessAUC.R'
'MeasureSurvGonenHellersC.R' 'MeasureSurvGraf.R'
'MeasureSurvGrafSE.R' 'MeasureSurvHarrellC.R'
'MeasureSurvHungAUC.R' 'MeasureSurvIntLogloss.R'
'MeasureSurvIntLoglossSE.R' 'MeasureSurvLogloss.R'
'MeasureSurvLoglossSE.R' 'MeasureSurvNagelkR2.R'
'MeasureSurvOQuigleyR2.R' 'MeasureSurvSongAUC.R'
'MeasureSurvSongTNR.R' 'MeasureSurvSongTPR.R'
'MeasureSurvUnoAUC.R' 'MeasureSurvUnoC.R' 'MeasureSurvUnoTNR.R'
'MeasureSurvUnoTPR.R' 'MeasureSurvXuR2.R'
'PipeOpCrankCompositor.R' 'PipeOpDistrCompositor.R'
'PredictionDensity.R' 'PredictionProbreg.R' 'PredictionSurv.R'
'TaskDensity.R' 'TaskDensity_precip.R'
'TaskGeneratorFriedman1Dens.R' 'TaskGeneratorSimsurv.R'
'TaskProbreg.R' 'TaskSurv.R' 'TaskSurv_lung.R'
'TaskSurv_rats.R' 'TaskSurv_unemployment.R' 'assertions.R'
'check_subsetpattern.R' 'crankcompositor.R' 'distrcompositor.R'
'helpers.R' 'integrated_scores.R' 'surv_logloss.R'
'weighted_survival_score.R' 'zzz.R'

Author Raphael Sonabend [aut, cre] (<https://orcid.org/0000-0001-9225-4654>),
Franz Kiraly [aut],
Michel Lang [aut] (<https://orcid.org/0000-0001-9754-0393>),
Andreas Bender [ctb] (<https://orcid.org/0000-0001-5628-8611>)

Maintainer Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk>

Repository CRAN

Date/Publication 2020-01-31 12:20:02 UTC

R topics documented:

mlr3proba-package *mlr3proba: Probabilistic Supervised Learning for 'mlr3'*

Description

Provides extensions for probabilistic supervised learning for 'mlr3'. This includes extending the regression task to probabilistic and interval regression, adding a survival task, and other specialized models, predictions, and measures.

Author(s)

Maintainer: Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk> (ORCID)

Authors:

- Franz Kiraly <f.kiraly@ucl.ac.uk>
- Michel Lang <michellang@gmail.com> (ORCID)

Other contributors:

- Andreas Bender <bender.at.R@gmail.com> (ORCID) [contributor]

See Also

Useful links:

- https://mlr3proba.mlr-org.com
- https://github.com/mlr-org/mlr3proba
- Report bugs at https://github.com/mlr-org/mlr3proba/issues

crankcompositor *Compose a Crank Predict Type for Survival Learners*

Description

This is a wrapper around the PipeOpCrankCompositor pipe operation, which simplifies graph creation.

Usage

```
crankcompositor(
  learner,
  method = c("mean", "median", "mode"),
  param_vals = list()
)
```

Arguments

learner	LearnerSurv object for which a crank is composed (or over-written)
method	One of mean, mode, or median; abbreviations allowed. Used to determine how crank is estimated from the predicted distr. Default is mean.
param_vals	Additional parameters to pass to the learner.

Details

For full details see PipeOpCrankCompositor.

Value

mlr3pipelines::GraphLearner

Examples

```
library(mlr3)
library(mlr3pipelines)

task = tgen("simsurv")$generate(20)
ranger.crank = crankcompositor(learner = lrn("surv.coxph"),
                               method = "median")
resample(task, ranger.crank, rsmp("cv", folds = 2))$predictions()
```

---

distrcompositor *Compose a Distr Predict Type for Survival Learners*

Description

This is a wrapper around the PipeOpDistrCompositor pipe operation, which simplifies graph creation.

Usage

```
distrcompositor(
  learner,
  estimator = c("kaplan", "nelson"),
  form = c("aft", "ph", "po"),
  overwrite = FALSE,
  param_vals = list()
)
```

Arguments

learner	LearnerSurv object for which a distr is composed (or over-written).
estimator	One of kaplan or nelson, corresponding to the Kaplan-Meier and Nelson-Aalen estimators respectively. Used to estimate the baseline survival distribution. Abbreviations allowed. Default is kaplan.
form	One of aft, ph, or po, corresponding to accelerated failure time, proportional hazards, and proportional odds respectively. Used to determine the form of the composed survival distribution. Default is aft.
overwrite	logical. If FALSE (default) then if the learner already has a distr, the compositor does nothing. If TRUE then the distr is overwritten by the compositor if already present, which may be required for changing the prediction distr from one model form to another.
param_vals	Additional parameters to pass to the learner.

Details

For full details see PipeOpDistrCompositor.

Value

mlr3pipelines::GraphLearner

Examples

```
## Not run:
library("mlr3")
library("mlr3pipelines")

task = tgen("simsurv")$generate(20)
cvglm.distr = distrcompositor(learner = lrn("surv.cvglmnet"),
                              estimator = "kaplan",
                              form = "aft")
resample(task, cvglm.distr,
        rsmp("cv", folds = 2))$predictions()

## End(Not run)
```

LearnerSurv *Survival Learner*

Description

This Learner specializes mlr3::Learner for survival problems. Predefined learners can be found in the mlr3misc::Dictionary mlr3::mlr_learners.

Format

R6::R6Class object inheriting from mlr3::Learner.

Construction

```
LearnerSurv$new(id, param_set = ParamSet$new(),
     predict_types = character(),
     feature_types = character(), properties = character(),
     packages = character())
```

For a description of the arguments, see mlr3::Learner. `task_type` is set to `"surv"`. Possible values for `predict_type` are `"distr"`, `"lp"`, and `"crank"`.

Fields

See mlr3::Learner.

Methods

See mlr3::Learner.

Examples

```
library(mlr3)
ids = mlr_learners$keys("^surv")
ids

# get a specific learner from mlr_learners:
lrn = mlr_learners$get("surv.rpart")
print(lrn)
```

LearnerSurvBlackboost *Gradient Boosting with Regression Trees Survival Learner*

Description

Calls `mboost::blackboost()`.

- lp is predicted by `mboost::predict.mboost()`
- distr is predicted by `mboost::survFit()` which assumes a PH fit with a Breslow estimator
- `crank` is identical to `lp`

The `dist` parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default `"coxph"`.

If the value given to the `Family` parameter is "custom.family" then an object of class `mboost::Family()` needs to be passed to the `custom.family` parameter.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvBlackboost$new()
mlr_learners$get("surv.blackboost")
lrn("surv.blackboost")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor
- Packages: **mboost distr6 survival partykit mvtnorm**

References

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**(4), 477–505. doi: 10.1214/07sts242.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933.

Freund Y, Schapire RE, others (1996). "Experiments with a new boosting algorithm." In *In: Thirteenth International Conference on ML*, volume 96, 148–156. Citeseer.

Friedman JH (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics*, **29**(5), 1189–1232. http://www.jstor.org/stable/2699986.

Ridgeway G (1999). "The state of boosting." *Computing Science and Statistics*, 172–181.

See Also

Other survival learners: LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

Examples

```
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.blackboost")
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)
```

`LearnerSurvCoxPH` *Cox Proportional Hazards Survival Learner*

Description

Calls `survival::coxph()`.

- lp is predicted by `survival::predict.coxph()`
- distr is predicted by `survival::survfit.coxph()`
- `crank` is identical to `lp`

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvCoxPH$new()
mlr_learners$get("surv.coxph")
lrn("surv.coxph")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: logical, integer, numeric, factor
- Packages: **survival distr6**

References

Cox DR (1972). "Regression Models and Life-Tables." *Journal of the Royal Statistical Society: Series B (Methodological)*, **34**(2), 187–202. doi: 10.1111/j.25176161.1972.tb00899.x.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvCVGlmnet` *Cross-Validated GLM with Elastic Net Regularization Survival Learner*

Description

Calls `glmnet::cv.glmnet()`.

- lp is predicted by `glmnet::predict.cv.glmnet()`
- `crank` is identical to `lp`

Use LearnerSurvGlmnet and LearnerSurvCVGlmnet for glmnets without and with internal cross-validation, respectively. Tuning using the internal optimizer in LearnerSurvCVGlmnet may be more efficient when tuning lambda only. However, for tuning multiple hyperparameters, **mlr3tuning** and LearnerSurvGlmnet will likely give better results.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvCVGlmnet$new()
mlr_learners$get("surv.cvglmnet")
lrn("surv.cvglmnet")
```

Meta Information

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: integer, numeric, factor
- Packages: **glmnet survival**

References

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1). doi: 10.18637/jss.v033.i01.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvFlexible` *Flexible Parametric Spline Survival Learner*

Description

Calls `flexsurv::flexsurvspline()`.

- lp is predicted by using an internally defined `predict` method, see details
- distr is predicted by using an internally defined `predict` method, see details
- `crank` is identical to `lp`

Parameter `k` is changed to `1` and `scale` is changed to `odds`, as these are more in line with the Royston/Parmar proposed models, and the package defaults are equivalent to fitting a parametric model and therefore surv.parametric should be used instead.

If fitting a model with `k = 0` then consider using surv.parametric as this is likely to have more optimal results, and has more options for tuning.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Details

The `distr` prediction is estimated using the fitted custom distributions from `flexsurv::flexsurvspline()` and the estimated coefficients.

As flexible spline models estimate the baseline hazard as the intercept, the linear predictor, `lp`, can be calculated as in the classical setting. i.e. For fitted coefficients, $\beta = (\beta_0, ..., \beta_P)$, and covariates $X^T = (X_0, ..., X_P)^T$, where X_0 is a column of 1s: $lp = \beta X$.

Construction

```
LearnerSurvFlexible$new()
mlr_learners$get("surv.flexible")
lrn("surv.flexible")
```

Meta Information

- Type: "surv"
- Predict Types: distr, lp, crank
- Feature Types: logical, integer, factor, numeric
- Packages: **flexsurv survival distr6**

References

Royston P, Parmar MKB (2002). "Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects." *Statistics in Medicine*, **21**(15), 2175–2197. doi: 10.1002/sim.1203.

See Also

Dictionary of Learners: mlr3::mlr_learners

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

LearnerSurvGamboost *Gradient Boosting for Additive Models Survival Learner*

Description

Calls mboost::gamboost().

- lp is predicted by mboost::predict.mboost()
- distr is predicted by mboost::survFit() which assumes a PH fit with a Breslow estimator
- crank is identical to lp

The dist parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default "coxph".

If the value given to the Family parameter is "custom.family" then an object of class mboost::Family() needs to be passed to the custom.family parameter.

The only difference between LearnerSurvGamboost and LearnerSurvMboost is that the latter function allows one to specify default degrees of freedom for smooth effects specified via baselearner = "bbs". In all other cases, degrees of freedom need to be set manually via a specific definition of the corresponding base-learner.

Format

R6::R6Class() inheriting from LearnerSurv.

Construction

```
LearnerSurvGamboost$new()
mlr_learners$get("surv.gamboost")
lrn("surv.gamboost")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor, logical
- Packages: **mboost distr6 survival**

References

Bühlmann P, Yu B (2003). "Boosting With the L2 Loss." *Journal of the American Statistical Association*, **98**(462), 324–339. doi: 10.1198/016214503000125.

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**(4), 477–505. doi: 10.1214/07sts242.

Kneib T, Hothorn T, Tutz G (2008). "Variable Selection and Model Choice in Geoadditive Regression Models." *Biometrics*, **65**(2), 626–634. doi: 10.1111/j.15410420.2008.01112.x.

Schmid M, Hothorn T (2008). "Boosting additive models using component-wise P-Splines." *Computational Statistics & Data Analysis*, **53**(2), 298–311. doi: 10.1016/j.csda.2008.09.009.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2010). "Model-based boosting 2.0." *Journal of Machine Learning Research*, **11**(Aug), 2109–2113. http://www.jmlr.org/papers/v11/hothorn10a.html.

Hofner B, Mayr A, Robinzonov N, Schmid M (2012). "Model-based boosting in R: a hands-on tutorial using the R package mboost." *Computational Statistics*, **29**(1-2), 3–35. doi: 10.1007/s00180-01203825.

See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

Examples

```
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.gamboost")
learner$param_set$values = mlr3misc::insert_named(learner$param_set$values,
     list(dfbase = 3, center = TRUE, baselearner = "bols"))
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)
```

LearnerSurvGBM *Generalized Boosting Regression Modeling Survival Learner*

Description

Calls gbm::gbm().

- lp is predicted by gbm::predict.gbm()
- crank is identical to lp

Parameter distribution is set to coxph as this is the only distribution implemented in gbm::gbm() for survival analysis; parameter keep.data is set to FALSE for efficiency.

Format

R6::R6Class() inheriting from LearnerSurv.

Construction

```
LearnerSurvGBM$new()
mlr_learners$get("surv.gbm")
lrn("surv.gbm")
```

Meta Information

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: integer, numeric, factor, ordered
- Packages: **gbm**

References

Freund Y, Schapire RE (1997). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." *Journal of Computer and System Sciences*, **55**(1), 119–139. doi: 10.1006/jcss.1997.1504.

Ridgeway G (1999). "The state of boosting." *Computing Science and Statistics*, 172–181.

Friedman J, Hastie T, Tibshirani R (2000). "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)." *The Annals of Statistics*, **28**(2), 337–407. doi: 10.1214/aos/1016218223.

Friedman JH (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics*, **29**(5), 1189–1232. http://www.jstor.org/stable/2699986.

Friedman JH (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics*, **29**(5), 1189–1232. http://www.jstor.org/stable/2699986.

Friedman JH (2002). "Stochastic gradient boosting." *Computational Statistics & Data Analysis*, **38**(4), 367–378. doi: 10.1016/s01679473(01)000652.

Kriegler B (2007). *Cost-Sensitive Stochastic Gradient Boosting Within a Quantitative Regression Framework*. Ph.D. thesis, University of California at Los Angeles. https://dl.acm.org/citation.cfm?id=1354603.

Burges CJ (2010). "From RankNet to LambdaRank to LambdaMART: An Overview." Technical Report MSR-TR-2010-82, Microsoft Research. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

`LearnerSurvGlmboost` *Gradient Boosting with Component-wise Linear Models Survival Learner*

Description

Calls `mboost::glmboost()`.

- lp is predicted by `mboost::predict.mboost()`
- distr is predicted by `mboost::survFit()` which assumes a PH fit with a Breslow estimator
- `crank` is identical to `lp`

The `dist` parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default `"coxph"`.

If the value given to the `Family` parameter is "custom.family" then an object of class `mboost::Family()` needs to be passed to the `custom.family` parameter.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvGlmboost$new()
mlr_learners$get("surv.glmboost")
lrn("surv.glmboost")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor, logical
- Packages: **mboost distr6 survival**

References

Bühlmann P, Yu B (2003). "Boosting With the L2 Loss." *Journal of the American Statistical Association*, **98**(462), 324–339. doi: 10.1198/016214503000125.

Bühlmann P (2006). "Boosting for High-Dimensional Linear Models." *The Annals of Statistics*, **34**(2), 559–583. http://www.jstor.org/stable/25463430.

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**(4), 477–505. doi: 10.1214/07sts242.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2010). "Model-based boosting 2.0." *Journal of Machine Learning Research*, **11**(Aug), 2109–2113. http://www.jmlr.org/papers/v11/hothorn10a.html.

Hofner B, Mayr A, Robinzonov N, Schmid M (2012). "Model-based boosting in R: a hands-on tutorial using the R package mboost." *Computational Statistics*, **29**(1-2), 3–35. doi: 10.1007/s00180-01203825.

See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

Examples

```
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.glmboost")
resampling = rsmp("cv", folds = 3)
resample(task, learner, resampling)
```

LearnerSurvGlmnet *GLM with Elastic Net Regularization Survival Learner*

Description

Calls glmnet::glmnet().

- lp is predicted by glmnet::predict.glmnet()
- crank is identical to lp

Use LearnerSurvGlmnet and LearnerSurvCVGlmnet for glmnets without and with internal cross-validation, respectively. Tuning using the internal optimizer in LearnerSurvCVGlmnet may be more efficient when tuning lambda only. However, for tuning multiple hyperparameters, **mlr3tuning** and LearnerSurvGlmnet will likely give better results.

Parameter s (value of the regularization parameter used for predictions) is set to the median of the lambda sequence by default, but needs to be tuned by the user.

Format

R6::R6Class() inheriting from LearnerSurv.

Construction

```
LearnerSurvGlmnet$new()
mlr_learners$get("surv.glmnet")
lrn("surv.glmnet")
```

Meta Information

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: integer, numeric, factor
- Packages: **glmnet survival**

References

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1). doi: 10.18637/jss.v033.i01.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvKaplan` *Kaplan-Meier Estimator Survival Learner*

Description

Calls `survival::survfit()`.

- distr is predicted by estimating the survival function with `survival::survfit()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvKaplan$new()
mlr_learners$get("surv.kaplan")
lrn("surv.kaplan")
```

Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, character, factor, ordered
- Packages: **survival distr6**

References

Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*, **53**(282), 457–481. doi: 10.1080/01621459.1958.10501452.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvMboost` *Gradient Boosting for Generalized Additive Models Survival Learner*

Description

Calls `mboost::mboost()`.

- lp is predicted by `mboost::predict.mboost()`
- distr is predicted by `mboost::survFit()` which assumes a PH fit with a Breslow estimator
- `crank` is identical to `lp`

The `dist` parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default `"coxph"`.

If the value given to the `Family` parameter is "custom.family" then an object of class `mboost::Family()` needs to be passed to the `custom.family` parameter.

The only difference between LearnerSurvGamboost and LearnerSurvMboost is that the latter function allows one to specify default degrees of freedom for smooth effects specified via `baselearner = "bbs"`. In all other cases, degrees of freedom need to be set manually via a specific definition of the corresponding base-learner.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvMboost$new()
mlr_learners$get("surv.mboost")
lrn("surv.mboost")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor, logical
- Packages: **mboost distr6 survival**

References

Bühlmann P, Yu B (2003). "Boosting With the L2 Loss." *Journal of the American Statistical Association*, **98**(462), 324–339. doi: 10.1198/016214503000125.

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**(4), 477–505. doi: 10.1214/07sts242.

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**(4), 477–505. doi: 10.1214/07sts242.

Kneib T, Hothorn T, Tutz G (2008). "Variable Selection and Model Choice in Geoadditive Regression Models." *Biometrics*, **65**(2), 626–634. doi: 10.1111/j.15410420.2008.01112.x.

Schmid M, Hothorn T (2008). "Boosting additive models using component-wise P-Splines." *Computational Statistics & Data Analysis*, **53**(2), 298–311. doi: 10.1016/j.csda.2008.09.009.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2010). "Model-based boosting 2.0." *Journal of Machine Learning Research*, **11**(Aug), 2109–2113. http://www.jmlr.org/papers/v11/hothorn10a.html.

Hofner B, Mayr A, Robinzonov N, Schmid M (2012). "Model-based boosting in R: a hands-on tutorial using the R package mboost." *Computational Statistics*, **29**(1-2), 3–35. doi: 10.1007/s00180-01203825.

See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

Examples

```
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.mboost")
learner$param_set$values = mlr3misc::insert_named(learner$param_set$values,
    list(center = TRUE, baselearner = "bols"))
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)
```

`LearnerSurvNelson` *Nelson-Aalen Estimator Survival Learner*

Description

Calls `survival::survfit()`.

- distr is predicted by estimating the cumulative hazard function with `survival::survfit()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvNelson$new()
mlr_learners$get("surv.nelson")
lrn("surv.nelson")
```

Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, character, factor, ordered
- Packages: **survival distr6**

References

Nelson W (1969). "Hazard Plotting for Incomplete Failure Data." *Journal of Quality Technology*, **1**(1), 27–52. doi: 10.1080/00224065.1969.11980344.

Nelson W (1972). "Theory and Applications of Hazard Plotting for Censored Failure Data." *Technometrics*, **14**(4), 945–966. doi: 10.1080/00401706.1972.10488991.

Aalen O (1978). "Nonparametric Inference for a Family of Counting Processes." *The Annals of Statistics*, **6**(4), 701–726. http://www.jstor.org/stable/2958850.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvParametric` *Fully Parametric Survival Learner*

Description

Calls `survival::survreg()`.

- lp is predicted by using an internally defined `predict` method, see details
- distr is predicted by using an internally defined `predict` method, see details
- `crank` is identical to `lp`

This learner allows you to choose a distribution and a model form to compose a predicted survival probability distribution. Note: Just because any combination of distribution and model form is possible, this does not mean it will necessarily be sensible or interpretable.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Details

The internal predict method is implemented in `mlr3proba`, which is more efficient for composition to distributions than `survival::predict.survreg()`.

`lp` is predicted using the formula $lp = X\beta$ where X are the variables in the test data set and β are the fitted coefficients.

The distribution `distr` is composed using the `lp` and specifying a model form in the `type` hyperparameter. These are as follows, with respective survival functions,

- Accelerated Failure Time (`aft`)

$$S(t) = S_0(\frac{t}{exp(lp)})$$

- Proportional Hazards (`ph`)

$$S(t) = S_0(t)^{exp(lp)}$$

- Proportional Odds (`po`)

$$S(t) = \frac{S_0(t)}{exp(-lp) + (1 - exp(-lp))S_0(t)}$$

where S_0 is the estimated baseline survival distribution (in this case with a given parametric form), and lp is the predicted linear predictor.

Construction

```
LearnerSurvParametric$new()
mlr_learners$get("surv.parametric")
lrn("surv.parametric")
```

Meta Information

- Type: "surv"
- Predict Types: distr, lp, crank
- Feature Types: logical, integer, numeric, factor
- Packages: **survival distr6**

References

Kalbfleisch JD, Prentice RL (2002). *The Statistical Analysis of Failure Time Data*. John Wiley & Sons, Inc. doi: 10.1002/9781118032985.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSR` `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvPenalized` *L1 and L2 Penalized Estimation in GLMs Survival Learner*

Description

Calls `penalized::penalized()`.

- distr is predicted using `penalized::predict()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

The `penalized` and `unpenalized` arguments in the learner are implemented slightly differently than in `penalized::penalized()`. Here, there is no parameter for `penalized` but instead it is assumed that every variable is penalized unless stated in the `unpenalized` parameter, see examples.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvPenalized$new()
mlr_learners$get("surv.penalized")
lrn("surv.penalized")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank
- Feature Types: integer, numeric, factor, ordered
- Packages: **penalized distr6**

References

Goeman JJ (2009). "L1Penalized Estimation in the Cox Proportional Hazards Model." *Biometrical Journal*, NA–NA. doi: 10.1002/bimj.200900028.

See Also

Dictionary of Learners: mlr3::mlr_learners

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvRandomForestS LearnerSurvRanger, LearnerSurvRpart, LearnerSurvSVM

Examples

```
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.penalized")
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)

# specifying penalized and unpenalized variables
task = tgen("simsurv")$generate(20)
learner = lrn("surv.penalized", unpenalized = c("height"))
learner$train(task)
learner$model@penalized
learner$model@unpenalized
```

LearnerSurvRandomForestSRC

RandomForestSRC Survival Forest Survival Learner

Description

Calls randomForestSRC::rfsrc().

- distr is predicted using randomForestSRC::predict.rfsrc()
- crank is predicted as the expectation of the survival distribution, distr

randomForestSRC::predict.rfsrc() returns both cumulative hazard function (chf) and survival function (surv) but uses different estimators to derive these. chf uses a bootstrapped Nelson-Aalen estimator, (Ishwaran, 2008) whereas surv uses a bootstrapped Kaplan-Meier estimator https://kogalur.github.io/randomForestSRC/theory.html. The choice of which estimator to use is given by the extra estimator hyper-parameter, default is nelson.

Format

R6::R6Class() inheriting from LearnerSurv.

Construction

```
LearnerSurvRandomForestSRC$new()
mlr_learners$get("surv.randomForestSRC")
lrn("surv.randomForestSRC")
```

Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, factor, ordered
- Packages: **randomForestSRC distr6**

References

Ishwaran H. and Kogalur U.B. (2019). Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC), R package version 2.9.1.

Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS, others (2008). "Random survival forests." *The annals of applied statistics*, **2**(3), 841–860.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvRanger` *Ranger Survival Forest Survival Learner*

Description

Calls `ranger::ranger()`.

- distr is predicted using `ranger::predict.ranger()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvRanger$new()
mlr_learners$get("surv.ranger")
lrn("surv.ranger")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank
- Feature Types: logical, integer, numeric, character, factor, ordered
- Packages: **ranger distr6**

References

Wright MN, Ziegler A (2017). "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software*, **77**(1), 1–17. doi: 10.18637/jss.v077.i01.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRpart`, `LearnerSurvSVM`

`LearnerSurvRpart` *Rpart Survival Forest Survival Learner*

Description

Calls `rpart::rpart()`.

- distr is predicted using `pec::pecRpart()` and `pec::predictSurvProb()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

Parameter `xval` is set to 0 in order to save some computation time.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvRpart$new()
mlr_learners$get("surv.rpart")
lrn("surv.rpart")
```

Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, character, factor, ordered
- Packages: **rpart distr6 survival**

References

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification And Regression Trees*. Routledge. doi: 10.1201/9781315139470.

See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvNelson, LearnerSurvParametric, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvSVM

`LearnerSurvSVM` *Regression, Ranking and Hybrid Support Vector Machines Survival Learner*

Description

Calls `survivalsvm::survivalsvm()`.

- crank is predicted by `survivalsvm::predict.survivalsvm()`

Four possible SVMs can be implemented, dependent on the `type` parameter. These correspond to predicting the survival time via regression (`regression`), predicting a continuous rank (`vanbelle1`, `vanbelle2`), or a hybrid of the two (`hybrid`). Whichever `type` is chosen determines how the `crank` predict type is calculated, but in any case all can be considered a valid continuous ranking.

To be in line with the Van Belle papers and to prevent the learner crashing without user-set parameters, the default `diff.meth` is set to `diffmeth3` with `gamma.mu` equal to 0.1.

Format

`R6::R6Class()` inheriting from LearnerSurv.

Construction

```
LearnerSurvSVM$new()
mlr_learners$get("surv.svm")
lrn("surv.svm")
```

Meta Information

- Type: "surv"
- Predict Types: `crank`
- Feature Types: integer, numeric
- Packages: **survivalsvm**

References

Belle VV, Pelckmans K, Huffel SV, Suykens JAK (2010). "Improved performance on high-dimensional survival data by application of Survival-SVM." *Bioinformatics*, **27**(1), 87–94. doi: 10.1093/bioinformatics/btq617.

Belle VV, Pelckmans K, Huffel SV, Suykens JA (2011). "Support vector methods for survival analysis: a comparison between ranking and regression approaches." *Artificial Intelligence in Medicine*, **53**(2), 107–118. doi: 10.1016/j.artmed.2011.06.006.

See Also

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCVGlmnet`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvNelson`, `LearnerSurvParametric`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`

`MeasureSurv` *Survival Measure*

Description

This measure specializes mlr3::Measure for survival problems. Predefined measures can be found in the mlr3misc::Dictionary mlr3::mlr_measures.

Format

R6::R6Class object inheriting from mlr3::Measure.

Construction

```
MeasureSurv$new(id, range, minimize, aggregator = NULL,
                properties = character(), predict_type = "distr",
                task_properties = character(), packages = character())
```

For a description of the arguments, see mlr3::Measure. The `task_type` is set to `"surv"`. Possible values for `predict_type` are `"distr"`, `"lp"`, and `"crank"`.

Fields

See Measure.

Methods

See Measure.

See Also

Example survival measures: `surv.graf`, `surv.harrellC`.

MeasureSurvBeggC *Begg's C-Index Survival Measure*

Description

Calls `survAUC::BeggC()`.

Assumes Cox PH model specification.

Format

`R6::R6Class()` inheriting from MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvBeggC$new()
mlr_measures$get("surv.beggC")
msr("surv.beggC")
```

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Begg CB, Cramer LD, Venkatraman ES, Rosai J (2000). "Comparing tumour staging and grading systems: a case study and a review of the issues, using thymoma as a model." *Statistics in Medicine*, **19**(15), 1997–2014. doi: 10.1002/10970258(20000815)19:15<1997::aidsim511>3.0.co;2c.

See Also

Other survival measures: MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvGonenC, MeasureSurvHarrellC, MeasureSurvUnoC

`MeasureSurvChamblessAUC`

Chambless and Diao's AUC Survival Measure

Description

Calls `survAUC::AUC.cd()`.

Assumes Cox PH model specification.

Format

`R6::R6Class()` inheriting from `MeasureSurvAUC`/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvChamblessAUC$new(integrated = TRUE, times)
mlr_measures$get("surv.chamblessAUC")
msr("surv.chamblessAUC")
```

- `integrated` :: `logical(1)`
  If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- `times` :: `vector()`
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Chambless LE, Diao G (2006). "Estimation of time-dependent area under the ROC curve for long-term risk prediction." *Statistics in Medicine*, **25**(20), 3474–3486. doi: 10.1002/sim.2299.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvHungAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

MeasureSurvGonenC *Gonen and Heller's C-Index Survival Measure*

Description

Calls survAUC::GHCI().

Assumes Cox PH model specification.

Format

R6::R6Class() inheriting from MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvGonenC$new()
mlr_measures$get("surv.gonenC")
msr("surv.gonenC")
```

Meta Information

- Type: "surv"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: lp

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Gönen M, Heller G (2005). "Concordance probability and discriminatory power in proportional hazards regression." *Biometrika*, **92**(4), 965–970. doi: 10.1093/biomet/92.4.965.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvBeggC, MeasureSurvHarrellC, MeasureSurvUnoC

MeasureSurvGraf *Integrated Graf Score Survival Measure*

Description

Calculates the Integrated Graf Score, aka integrated Brier score or squared loss.

For an individual who dies at time t, with predicted Survival function, S, the Graf Score at time t^* is given by

$$L(S, t|t^*) = [(S(t^*)^2)I(t \leq t^*, \delta = 1)(1/G(t))] + [((1 - S(t^*))^2)I(t > t^*)(1/G(t^*))]$$

where G is the Kaplan-Meier estimate of the censoring distribution.

Note: If comparing the integrated graf score to other packages, e.g. `pec::pec()`, then `method = 2` should be used, however the results may still be very slightly different as this package uses `survfit` to estimate the censoring distribution, in line with the Graf 1999 paper. Whereas some other packages use `prodlim` with `reverse = TRUE` (meaning Kaplan-Meier is not used).

If `integrated == FALSE` then the sample mean is taken for the single specified `times`, t^*, and the returned score is given by

$$L(S, t|t^*) = \frac{1}{N}\sum_{i=1}^{N} L(S_i, t_i|t^*)$$

where N is the number of observations, S_i is the predicted survival function for individual i and t_i is their true survival time.

If `integrated == TRUE` then an approximation to integration is made by taking the mean over all T unique time-points, and then the sample mean over all N observations.

$$L(S) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{j=1}^{T} L(S_i, t_i|t_j^*)$$

Format

R6::R6Class() inheriting from MeasureSurvIntegrated/MeasureSurv.

Construction

```
MeasureSurvGraf$new(integrated = TRUE, times, method = 2)
mlr_measures$get("surv.graf")
msr("surv.graf")
```

- integrated :: logical(1)
  If TRUE (default), returns the integrated score; otherwise, not integrated.

- times :: vector()
  If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

Meta Information

- Type: "surv"
- Range: $[0, \infty)$
- Minimize: TRUE
- Required prediction: distr

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Graf E, Schmoor C, Sauerbrei W, Schumacher M (1999). "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in Medicine*, **18**(17-18), 2529–2545. doi: 10.1002/(sici)10970258(19990915/30)18:17/18<2529::aidsim274>3.0.co;25.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGrafSE, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss

`MeasureSurvGrafSE` *Standard Error of Integrated Graf Score Survival Measure*

Description

Calculates the standard error of MeasureSurvGraf.

If `integrated == FALSE` then the standard error of the loss, L, is approximated via,

$$se(L) = sd(L)/\sqrt{N}$$

where N are the number of observations in the test set, and sd is the standard deviation.

If `integrated == TRUE` then correlations between time-points need to be taken into account, therefore

$$se(L) = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\Sigma_{i,j}}{NT^2}}$$

where $\Sigma_{i,j}$ is the sample covariance matrix over M distinct time-points.

Format

`R6::R6Class()` inheriting from `MeasureSurvIntegrated`/MeasureSurv.

Construction

```
MeasureSurvGrafSE$new(integrated = TRUE, times)
mlr_measures$get("surv.grafSE")
msr("surv.grafSE")
```

- `integrated` :: `logical(1)`
  If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- `times` :: `vector()`
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

Meta Information

- Type: `"surv"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `distr`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Graf E, Schmoor C, Sauerbrei W, Schumacher M (1999). "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in Medicine*, **18**(17-18), 2529–2545. doi: 10.1002/(sici)10970258(19990915/30)18:17/18<2529::aidsim274>3.0.co;25.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGraf, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss

---

MeasureSurvHarrellC *Harrell's C-Index Survival Measure*

Description

Calculates Harrell's C, equivalent to the Fortran implementation in **Hmisc**.

Format

R6::R6Class() inheriting from MeasureSurv.

Construction

```
MeasureSurvHarrellC$new()
mlr_measures$get("surv.harrellC")
msr("surv.harrellC")
```

Meta Information

- Type: "surv"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: crank

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Harrell FE, Califf RM, Pryor DB, Lee KL, Rosati RA (1982). "Evaluating the yield of medical tests." *Jama*, **247**(18), 2543–2546.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvUnoC

MeasureSurvHungAUC *Hung and Chiang's AUC Survival Measure*

Description

Calls survAUC::AUC.hc().

Assumes random censoring.

Format

R6::R6Class() inheriting from MeasureSurvAUC/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvHungAUC$new(integrated = TRUE, times)
mlr_measures$get("surv.hungAUC")
msr("surv.hungAUC")
```

- integrated :: logical(1)
  If TRUE (default), returns the integrated score; otherwise, not integrated.

- times :: vector()
  If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

Meta Information

- Type: "surv"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: lp

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Hung H, Chiang C (2010). "Estimation methods for time-dependent AUC models with survival data." *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, **38**(1), 8–26. http://www.jstor.org/stable/27805213.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

MeasureSurvIntLogloss *Integrated Log loss Survival Measure*

Description

Calculates the integrated logarithmic (log), loss, aka integrated cross entropy.

For an individual who dies at time t, with predicted Survival function, S, the probabilistic log loss at time t^* is given by

$$L(S, t|t^*) = -[log(1 - S(t^*))I(t \leq t^*, \delta = 1)(1/G(t))] - [log(S(t^*))I(t > t^*)(1/G(t^*))]$$

where G is the Kaplan-Meier estimate of the censoring distribution.

If `integrated == FALSE` then the sample mean is taken for the single specified `times`, t^*, and the returned score is given by

$$L(S, t|t^*) = \frac{1}{N}\sum_{i=1}^{N} L(S_i, t_i|t^*)$$

where N is the number of observations, S_i is the predicted survival function for individual i and t_i is their true survival time.

If `integrated == TRUE` then an approximation to integration is made by taking the mean over all T unique time-points, and then the sample mean over all N observations.

$$L(S) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{j=1}^{T} L(S_i, t_i|t_j^*)$$

Format

R6::R6Class() inheriting from MeasureSurvIntegrated/MeasureSurv.

Construction

```
MeasureSurvIntLogloss$new(integrated = TRUE, times, eps = 1e-15, method = 2)
mlr_measures$get("surv.intlogloss")
msr("surv.intlogloss")
```

- `integrated` :: `logical(1)`
  If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- `times` :: `vector()`
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

- `eps` :: `numeric(1)`
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

Meta Information

- Type: `"surv"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `distr`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

As well as

- eps :: numeric(1)
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

References

Graf E, Schmoor C, Sauerbrei W, Schumacher M (1999). "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in Medicine*, **18**(17-18), 2529–2545. doi: 10.1002/(sici)10970258(19990915/30)18:17/18<2529::aidsim274>3.0.co;25.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvIntLoglossSE, MeasureSurvLoglossSE, MeasureSurvLogloss

MeasureSurvIntLoglossSE

Standard Error of Integrated Log loss Survival Measure

Description

Calculates the standard error of MeasureSurvIntLogloss.

If `integrated == FALSE` then the standard error of the loss, L, is approximated via,

$$se(L) = sd(L)/\sqrt{N}$$

where N are the number of observations in the test set, and sd is the standard deviation.

If `integrated == TRUE` then correlations between time-points need to be taken into account, therefore

$$se(L) = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\Sigma_{i,j}}{NT^2}}$$

where $\Sigma_{i,j}$ is the sample covariance matrix over M distinct time-points.

Format

`R6::R6Class()` inheriting from `MeasureSurvIntegrated`/MeasureSurv.

Construction

```
MeasureSurvIntLoglossSE$new(integrated = TRUE, times, eps = 1e-15)
mlr_measures$get("surv.intloglossSE")
msr("surv.intloglossSE")
```

- `integrated` :: `logical(1)`
  If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- `times` :: `vector()`
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

- `eps` :: `numeric(1)`
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

Meta Information

- Type: `"surv"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `distr`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

As well as

- eps :: numeric(1)
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

References

Graf E, Schmoor C, Sauerbrei W, Schumacher M (1999). "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in Medicine*, **18**(17-18), 2529–2545. doi: 10.1002/(sici)10970258(19990915/30)18:17/18<2529::aidsim274>3.0.co;25.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss

---

MeasureSurvLogloss *Log loss Survival Measure*

Description

Calculates the cross-entropy, or logarithmic (log), loss.

The logloss, in the context of probabilistic predictions, is defined as the negative log probability density function, f, evaluated at the observation time, t,

$$L(f, t) = -log(f(t))$$

Censored observations in the test set are ignored.

Format

R6::R6Class() inheriting from MeasureSurv.

Construction

```
MeasureSurvLogloss$new(eps = 1e-15)
mlr_measures$get("surv.logloss")
msr("surv.logloss")
```

- eps :: `numeric(1)`
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

Meta Information

- Type: `"surv"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `distr`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

As well as

- eps :: numeric(1)
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE

MeasureSurvLoglossSE *Standard Error of Log loss Survival Measure*

Description

Calculates the standard error of MeasureSurvLogloss.

The standard error of the Logloss, L, is approximated via,

$$se(L) = sd(L)/\sqrt{N}$$

where N are the number of observations in the test set, and sd is the standard deviation.

Censored observations in the test set are ignored.

Format

`R6::R6Class()` inheriting from MeasureSurvLogloss/MeasureSurv.

Construction

```
MeasureSurvLoglossSE$new(eps = 1e-15)
mlr_measures$get("surv.loglossSE")
msr("surv.loglossSE")
```

- `eps` :: `numeric(1)`
  Very small number to set zero-valued predicted probabilities to, in order to prevent errors in log(0) calculation.

Meta Information

- Type: `"surv"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `distr`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLogloss`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other Probabilistic survival measures: `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLogloss`

`MeasureSurvNagelkR2` *Nagelkerke's R2 Survival Measure*

Description

Calls `survAUC::Nagelk()`.

Assumes Cox PH model specification.

Format

`R6::R6Class()` inheriting from MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvNagelkR2$new()
mlr_measures$get("surv.nagelkR2")
msr("surv.nagelkR2")
```

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Nagelkerke NJ, others (1991). "A note on a general definition of the coefficient of determination." *Biometrika*, **78**(3), 691–692.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other R2 survival measures: `MeasureSurvOQuigleyR2`, `MeasureSurvXuR2`

`MeasureSurvOQuigleyR2` *O'Quigley, Xu, and Stare's R2 Survival Measure*

Description

Calls `survAUC::OXS()`.

Assumes Cox PH model specification.

Format

`R6::R6Class()` inheriting from MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvOQuigleyR2$new()
mlr_measures$get("surv.oquigleyR2")
msr("surv.oquigleyR2")
```

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

O'Quigley J, Xu R, Stare J (2005). "Explained randomness in proportional hazards models." *Statistics in Medicine*, **24**(3), 479–489. doi: 10.1002/sim.1946.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvNagelkR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other R2 survival measures: `MeasureSurvNagelkR2`, `MeasureSurvXuR2`

`MeasureSurvSongAUC` *Song and Zhou's AUC Survival Measure*

Description

Calls `survAUC::AUC.sh()`.

Assumes Cox PH model specification.

Format

`R6::R6Class()` inheriting from `MeasureSurvAUC`/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvSongAUC$new(integrated = TRUE, times, type = c("incident","cumulative"))
mlr_measures$get("surv.songAUC")
msr("surv.songAUC")
```

- `integrated` :: `logical(1)`
  If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- `times` :: `vector()`
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

- `type` :: `character(1)`
  Determines the type of score, one of: `'cumulative'`, `'incident'`. Default `'incident'`.

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Song X, Zhou X (2008). "A semiparametric approach for the covariate specific ROC curve with survival outcome." *Statistica Sinica*, **18**(3), 947–65. http://www.jstor.org/stable/24308524.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

MeasureSurvSongTNR *Song and Zhou's TNR Survival Measure*

Description

Calls survAUC::spec.sh().

Assumes Cox PH model specification.

times and lp_thresh are arbitrarily set to 0 to prevent crashing, these should be further specified.

Format

R6::R6Class() inheriting from MeasureSurvAUC/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvSongTNR$new(times = 0, lp_thresh = 0)
mlr_measures$get("surv.songTNR")
msr("surv.songTNR")
```

- times :: vector()
  If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

- lp_thresh :: numeric(1)
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

Meta Information

- Type: "surv"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: lp

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Song X, Zhou X (2008). "A semiparametric approach for the covariate specific ROC curve with survival outcome." *Statistica Sinica*, **18**(3), 947–65. http://www.jstor.org/stable/24308524.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongAUC, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

MeasureSurvSongTPR *Song and Zhou's TPR Survival Measure*

Description

Calls survAUC::sens.sh().

Assumes Cox PH model specification.

times and lp_thresh are arbitrarily set to 0 to prevent crashing, these should be further specified.

Format

R6::R6Class() inheriting from MeasureSurvAUC/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvSongTPR$new(times = 0, lp_thresh = 0, type = c("incident","cumulative"))
mlr_measures$get("surv.songTPR")
msr("surv.songTPR")
```

- times :: vector()
  If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

- lp_thresh :: numeric(1)
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

- type :: character(1)
  Determines the type of score, one of: 'cumulative', 'incident'. Default 'incident'.

Meta Information

- Type: "surv"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: lp

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Song X, Zhou X (2008). "A semiparametric approach for the covariate specific ROC curve with survival outcome." *Statistica Sinica*, **18**(3), 947–65. http://www.jstor.org/stable/24308524.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

MeasureSurvUnoAUC *Uno's AUC Survival Measure*

Description

Calls survAUC::AUC.uno().

Assumes random censoring.

Format

R6::R6Class() inheriting from MeasureSurvAUC/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvUnoAUC$new(integrated = TRUE, times)
mlr_measures$get("surv.unoAUC")
msr("surv.unoAUC")
```

- `integrated` :: `logical(1)`
  If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- `times` :: `vector()`
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Uno H, Cai T, Tian L, Wei LJ (2007). "Evaluating Prediction Rules fort-Year Survivors With Censored Regression Models." *Journal of the American Statistical Association*, **102**(478), 527–537. doi: 10.1198/016214507000000149.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoTNR, MeasureSurvUnoTPR

MeasureSurvUnoC *Uno's C-Index Survival Measure*

Description

Calls `survAUC::UnoC()`.

Assumes random censoring.

Format

`R6::R6Class()` inheriting from MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvUnoC$new()
mlr_measures$get("surv.unoC")
msr("surv.unoC")
```

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `crank`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Uno H, Cai T, Pencina MJ, D'Agostino RB, Wei LJ (2011). "On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data." *Statistics in Medicine*, n/a–n/a. doi: 10.1002/sim.4154.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvHarrellC

MeasureSurvUnoTNR *Uno's TNR Survival Measure*

Description

Calls survAUC::spec.uno().

Assumes random censoring.

times and lp_thresh are arbitrarily set to 0 to prevent crashing, these should be further specified.

Format

R6::R6Class() inheriting from MeasureSurvAUC/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvUnoTNR$new(times = 0, lp_thresh = 0)
mlr_measures$get("surv.unoTNR")
msr("surv.unoTNR")
```

- times :: vector()
  If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

- lp_thresh :: numeric(1)
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

Meta Information

- Type: "surv"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: lp

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Uno H, Cai T, Tian L, Wei LJ (2007). "Evaluating Prediction Rules fort-Year Survivors With Censored Regression Models." *Journal of the American Statistical Association*, **102**(478), 527–537. doi: 10.1198/016214507000000149.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTPR

MeasureSurvUnoTPR *Uno's TPR Survival Measure*

Description

Calls survAUC::sens.uno().

Assumes random censoring.

times and lp_thresh are arbitrarily set to 0 to prevent crashing, these should be further specified.

Format

R6::R6Class() inheriting from MeasureSurvAUC/MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Construction

```
MeasureSurvUnoTPR$new(times = 0, lp_thresh = 0)
mlr_measures$get("surv.unoTPR")
msr("surv.unoTPR")
```

- times :: vector()
  If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

- `lp_thresh` :: `numeric(1)`
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Uno H, Cai T, Tian L, Wei LJ (2007). "Evaluating Prediction Rules fort-Year Survivors With Censored Regression Models." *Journal of the American Statistical Association*, **102**(478), 527–537. doi: 10.1198/016214507000000149.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvXuR2`

Other AUC survival measures: `MeasureSurvChamblessAUC`, `MeasureSurvHungAUC`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoTNR`

`MeasureSurvXuR2` *Xu and O'Quigley's R2 Survival Measure*

Description

Calls `survAUC::XO()`.

Assumes Cox PH model specification.

Format

`R6::R6Class()` inheriting from MeasureSurv.

Details

All measures implemented from **survAUC** should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Construction

```
MeasureSurvXuR2$new()
mlr_measures$get("surv.xuR2")
msr("surv.xuR2")
```

Meta Information

- Type: `"surv"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `lp`

Fields

See MeasureSurv, as well as all variables passed to the constructor.

References

Xu R, O'Quigley J (1999). "A R2 type measure of dependence for proportional hazards models." *Journal of Nonparametric Statistics*, **12**(1), 83–107. doi: 10.1080/10485259908832799.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

Other R2 survival measures: MeasureSurvNagelkR2, MeasureSurvOQuigleyR2

---

mlr_tasks_lung *Lung Cancer Survival Task*

Description

A survival task for the survival::lung data set. Column "sex" has been converted to a factor, all others have been converted to integer.

Format

R6::R6Class inheriting from TaskSurv.

Construction

```
mlr3::mlr_tasks$get("lung")
mlr3::tsk("lung")
```

See Also

Dictionary of Tasks: mlr3::mlr_tasks

`mlr_tasks_rats` *Rats Survival Task*

Description

A survival task for the survival::rats data set. Column "sex" has been converted to a factor, all others have been converted to integer.

Format

R6::R6Class inheriting from TaskSurv.

Construction

```
mlr3::mlr_tasks$get("rats")
mlr3::tsk("rats")
```

See Also

Dictionary of Tasks: mlr3::mlr_tasks

`mlr_tasks_unemployment`
Unemployment Duration Task

Description

A survival task for the "UnempDur" data set in package **Ecdat**. Contains the following columns of the original data set: "spell" (time), "censor1" (status), "age", "ui", "logwage", and "tenure".

Format

R6::R6Class inheriting from TaskSurv.

Construction

```
mlr3::mlr_tasks$get("unemployment")
mlr3::tsk("unemployment")
```

See Also

Dictionary of Tasks: mlr3::mlr_tasks

PipeOpCrankCompositor *PipeOpCrankCompositor*

Description

Uses a predicted `distr` in a PredictionSurv to estimate (or 'compose') a `crank` prediction.

Format

`R6Class` inheriting from `PipeOp`.

Construction

```
PipeOpCrankCompositor$new(id = "crankcompose", param_vals = list(method = "mean"))
```

- `id` :: `character(1)`
  Identifier of the resulting object, default `"crankcompose"`.
- `param_vals` :: named `list`
  List of hyperparameter settings, overwriting the hyperparameter settings that would otherwise be set during construction. Default `list(method = "mean")`.

Input and Output Channels

PipeOpCrankCompositor has one input channel named "input", which takes `NULL` during training and PredictionSurv during prediction.

PipeOpCrankCompositor has one output channel named "output", producing `NULL` during training and a PredictionSurv during prediction.

The output during prediction is the PredictionSurv from the "pred" input but with the `crank` predict type overwritten by the given estimation method.

State

The $state is left empty (`list()`).

Parameters

- `method` :: `character(1)`
  Determines what method should be used to produce a continuous ranking from the distribution. One of `median`, `mode`, or `mean` corresponding to the respective functions in the predicted survival distribution. Note that for models with a proportional hazards form, the ranking implied by `mean` and `median` will be identical (but not the value of `crank` itself). Default is `mean`.

Internals

The `median`, `mode`, or `mean` will use analytical expressions if possible but if not they are calculated using distr6::median.Distribution, distr6::mode, or distr6::mean.Distribution respectively.

Fields

Only fields inherited from PipeOp.

Methods

Only methods inherited from PipeOp.

See Also

mlr3pipelines::PipeOp and crankcompositor

Other survival compositors: PipeOpDistrCompositor

Examples

```
library(mlr3)
library(mlr3pipelines)
set.seed(1)

# Three methods to predict a `crank` from `surv.rpart`
task = tgen("simsurv")$generate(30)

# Method 1 - Train and predict separately then compose
learn = lrn("surv.coxph")$train(task)$predict(task)
poc = po("crankcompose", param_vals = list(method = "mean"))
poc$predict(list(learn))

# Examples not run to save run-time.
## Not run:
# Method 2 - Create a graph manually
gr = Graph$new()$
  add_pipeop(po("learner", lrn("surv.ranger")))$
  add_pipeop(po("crankcompose"))$
  add_edge("surv.ranger", "crankcompose")
gr$train(task)
gr$predict(task)

# Method 3 - Syntactic sugar: Wrap the learner in a graph
ranger.crank = crankcompositor(learner = lrn("surv.ranger"),
                               method = "median")
resample(task, ranger.crank, rsmp("cv", folds = 2))$predictions()

## End(Not run)
```

PipeOpDistrCompositor *PipeOpDistrCompositor*

Description

Estimates (or 'composes') a survival distribution from a predicted baseline `distr` and a `crank` or `lp` from two PredictionSurvs.

Compositor Assumptions:

- The baseline `distr` is a discrete estimator, i.e. LearnerSurvKaplan or LearnerSurvNelson
- The composed `distr` is of a linear form
- If `lp` is missing then `crank` is equivalent

These assumptions are strong and may not be reasonable. Future updates will upgrade this compositor to be more flexible.

Format

`R6Class` inheriting from `PipeOp`.

Construction

```
PipeOpDistrCompositor$new(id = "distrcompose", param_vals = list())
```

- `id` :: `character(1)`
  Identifier of the resulting object, default `"distrcompose"`.
- `param_vals` :: named `list`
  List of hyperparameter settings, overwriting the hyperparameter settings that would otherwise be set during construction. Default `list()`.

Input and Output Channels

PipeOpDistrCompositor has two input channels, "base" and "pred". Both input channels take `NULL` during training and PredictionSurv during prediction.

PipeOpDistrCompositor has one output channel named "output", producing `NULL` during training and a PredictionSurv during prediction.

The output during prediction is the PredictionSurv from the "pred" input but with an extra (or overwritten) column for `distr` predict type; which is composed from the `distr` of "base" and `lp` or `crank` of "pred".

State

The $state is left empty (`list()`).

Parameters

The parameters are:

- `form` :: `character(1)`
  Determines the form that the predicted linear survival model should take. This is either, accelerated-failure time, `aft`, proportional hazards, `ph`, or proportional odds, `po`. Default `aft`.

- `overwrite` :: `logical(1)`
  If `FALSE` (default) then if the "pred" input already has a `distr`, the compositor does nothing and returns the given PredictionSurv. If `TRUE` then the `distr` is overwritten with the `distr` composed from `lp`/`crank` - this is useful for changing the prediction `distr` from one model form to another.

Internals

The respective `forms` above have respective survival distributions:

$$aft : S(t) = S_0(\frac{t}{exp(lp)})$$

$$ph : S(t) = S_0(t)^{exp(lp)}$$

$$po : S(t) = \frac{S_0(t)}{exp(-lp) + (1 - exp(-lp))S_0(t)}$$

where S_0 is the estimated baseline survival distribution, and lp is the predicted linear predictor. If the input model does not predict a linear predictor then `crank` is assumed to be the `lp` - **this may be a strong and unreasonable assumption.**

Fields

Only fields inherited from PipeOp.

Methods

Only methods inherited from PipeOp.

See Also

mlr3pipelines::PipeOp and distrcompositor

Other survival compositors: `PipeOpCrankCompositor`

Examples

```
library(mlr3)
library(mlr3pipelines)
set.seed(42)

# Three methods to transform the cox ph predicted `distr` to an
#  accelerated failure time model
task = tgen("simsurv")$generate(30)

# Method 1 - Train and predict separately then compose
base = lrn("surv.kaplan")$train(task)$predict(task)
pred = lrn("surv.coxph")$train(task)$predict(task)
pod = po("distrcompose", param_vals = list(form = "aft", overwrite = TRUE))
pod$predict(list(base = base, pred = pred))

# Examples not run to save run-time.
```

```
## Not run:
# Method 2 - Create a graph manually
gr = Graph$new()$
  add_pipeop(po("learner", lrn("surv.kaplan")))$
  add_pipeop(po("learner", lrn("surv.glmnet")))$
  add_pipeop(po("distrcompose"))$
  add_edge("surv.kaplan", "distrcompose", dst_channel = "base")$
  add_edge("surv.glmnet", "distrcompose", dst_channel = "pred")
gr$train(task)$gr$predict(task)

# Method 3 - Syntactic sugar: Wrap the learner in a graph.
cvglm.distr = distrcompositor(learner = lrn("surv.cvglmnet"),
                              estimator = "kaplan",
                              form = "aft")
cvglm.distr$fit(task)$predict(task)

## End(Not run)
```

`PredictionSurv` *Prediction Object for Survival*

Description

This object stores the predictions returned by a learner of class LearnerSurv.

The `task_type` is set to `"surv"`.

Format

R6::R6Class object inheriting from mlr3::Prediction.

Construction

```
PredictionSurv$new(task = NULL, row_ids = task$row_ids, truth = task$truth(),
                   crank = NULL, distr = NULL, lp = NULL)
```

- `task` :: TaskSurv
  Task, used to extract defaults for `row_ids` and `truth`.
- `row_ids` :: (`integer()` | `character()`)
  Row ids of the task. Per default, these are extracted from the `task`.
- `truth` :: `survival::Surv()`
  Observed survival times. Per default, these are extracted from the `task`.
- `crank` :: `numeric()`
  Vector of continuous ranks. One element for each observation in the test set. For a pair of continuous ranks, a higher rank indicates that the observation is more likely to experience the event. Used in discrimination measures like surv.harrellC.
- `distr` :: `distr6::Distribution()`
  VectorDistribution from **distr6**. Each individual distribution in the vector represents the random variable 'survival time' for an individual observation. Used in measures like surv.graf.

- `lp` :: `numeric()`
  Vector of linear predictor scores. One element for each observation in the test set. $lp = X\beta$ where X is a matrix of covariates and β is a vector of estimated coefficients. Used in discrimination measures like surv.harrellC.

Fields

See mlr3::Prediction.

The field `task_type` is set to `"surv"`.

Examples

```
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = mlr_learners$get("surv.rpart")
p = learner$train(task)$predict(task)
head(as.data.table(p))
```

`TaskGeneratorSimsurv` *Survival Task Generator for Package 'simsurv'*

Description

A mlr3::TaskGenerator calling `simsurv::simsurv()` from package **simsurv**.

This generator currently only exposes a small subset of the flexibility of **simsurv**, and just creates a small data set with the following numerical covariates:

- `treatment`: Bernoulli distributed with log hazard ratio `-0.5`.
- `height`: Normally distributed with log hazard ratio `1`.
- `weight`: normally distributed with log hazard ratio `0`.

See `simsurv::simsurv()` for an explanation of the hyperparameters.

Format

R6::R6Class inheriting from mlr3::TaskGenerator.

Construction

```
TaskGeneratorSimsurv$new()
mlr_task_generators$get("simsurv")
tgen("simsurv")
```

See Also

Dictionary of TaskGenerators: mlr3::mlr_task_generators

Examples

```
generator = mlr3::mlr_task_generators$get("simsurv")
task = generator$generate(20)
task$head()
```

`TaskSurv` *Survival Task*

Description

This task specializes mlr3::Task and mlr3::TaskSupervised for possibly-censored survival problems. The target is comprised of survival times and an event indicator. Predefined tasks are stored in mlr3::mlr_tasks.

The `task_type` is set to `"surv"`.

Format

R6::R6Class object inheriting from Task/TaskSupervised.

Construction

```
TaskSurv$new(id, backend, time, event, time2,
   type = c("right","left","counting","interval","interval2","mstate"))
```

- `id` :: `character(1)`
  Name of the task.
- `backend` :: DataBackend
- `time` :: `numeric()`
  Event time if data is right censored. Starting time if interval censored.
- `event` :: `integer()` | `logical()`
  Event indicator. If data is right censored then "0"/`FALSE` means alive (no event), "1"/`TRUE` means dead (event). If `type` is `"interval"` then "0" means right censored, "1" means dead (event), "2" means left censored, and "3" means interval censored. If `type` is `"interval2"` then `event` is ignored.
- `time2` :: `numeric()`
  Ending time for interval censored data. Ignored otherwise.
- `type` :: character()
  Type of censoring. Default is 'right' censoring.

Fields

All fields from mlr3::TaskSupervised, and additionally:

- `censtype` :: `character()`
  Returns the type of censoring, one of "right", "left", "counting", "interval", "interval2" or "mstate".

Methods

See mlr3::TaskSupervised.

Examples

```
library(mlr3)
lung = mlr3misc::load_dataset("lung", package = "survival")
lung$status = (lung$status == 2L)
b = as_data_backend(lung)
task = TaskSurv$new("lung", backend = b, time = "time",
   event = "status")

task$target_names
task$feature_names
task$formula()
task$truth()
```

Index