Package ‘MTPS’

May 22, 2023

Type Package

Title Multi-Task Prediction using Stacking Algorithms

Version 1.1.1

Description Simultaneous multiple outcomes prediction based on revised stacking algorithms, which enables the integration of information from predictions of individual models. An implementation of methodologies proposed in our paper: Li Xing, Mary L Lesperance, Xuekui Zhang. (2019) Bioinformatics, ``Simultaneous prediction of multiple outcomes using revised stacking algorithms'' <doi:10.1093/bioinformatics/btz531>.

License GPL (>= 2)

URL `https://doi.org/10.1093/bioinformatics/btz531`

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports caret,
aftgee,
class,
dplyr,
e1071,
glmnet,
MASS,
rpart,
survival,
xgboost

Suggests knitr,
rmarkdown,
ggplot2,
reshape2

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.2.3

Author Li Xing [aut, cre],
Xiaowen Cao [aut],
Shuai You [aut],
Yuying Huang [aut],
Peijie Xie [ctb],
Mary Lesperance [aut],
Xuekui Zhang [aut]

Repository CRAN

Date/Publication 2023-5-20 04:30:03 UTC

R topics documented:

`AUC` *Area Under Curve*

Description

The AUC function calculates the numeric value of area under the ROC curve \(AUC\) with the trapezoidal rule and optionally plots the ROC curve

Usage

```
AUC(prob, outcome, cutoff = 1, ROC.plot = FALSE)
```

Arguments

`prob`	A numeric vector of predicted probability
`outcome`	A numeric vector of observed binary outcome
`cutoff`	Number between 0 and 1 to specify where threshold of ROC curve should be truncated. The default value is 1 \(no truncation\)
`ROC.plot`	Logical. Whether or not to plot ROC curve

Details

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at different threshold settings. By default the total area under the curve is computed, but a truncated AUC statistics can be specified with the cutoff argument. It specifies the bounds of FPR. The common choice of cutoff can be 1 (i.e. no truncate) or 0.2 (i.e. specificity > 0.8)

Value

The value of the area under the curve.

Examples

```
set.seed(1)
# simulate predictors
x1 <- rnorm(200)
x2 <- rnorm(200)
# simulate outcome
pr <- 1/(1+exp(-(3 * x1 + 2 * x2 + 1)))
y <- rbinom(200, 1, pr)
df <- data.frame(y = y,x1 = x1, x2 = x2)
# fit logistic regression model on the first 100 observation
lg.model <- glm(y ~ x1 + x2, data = df[1 : 100, ], family="binomial")
# predict outcome for the last 100 observation
prob <- predict(lg.model, df[101:200, c("x1", "x2")], type = "response")
# calculate AUC and plot thr ROC Curve
AUC(prob, y[101:200], ROC=TRUE)
# calculate AUC and plot thr ROC Curve with cutoff
AUC(prob, y[101:200], cutoff=0.2, ROC=TRUE)
```

`cindex` *Concordance Index*

Description

Compute the C-index in survival analysis.

Usage

```
cindex(pre, object)
```

Arguments

`pre`	A numeric vector of predicted survival time.
`object`	A dataframe or matrix, the first column is observed survival time the second column is survival status.

Details

The most frequently used evaluation metric of survival models is the concordance index (C-index). It is a measure of rank correlation between predicted risk scores. Only use uncensored observations

Value

The value of C-index.

Examples

```
set.seed(1)
# simulate predicted survival time
x1 <- rnorm(100)
# simulate observed survival time
x2 <- rnorm(100)
# simulate observed survival status
x3 <- rep(c(1, 0), 50)
```

```
# calculate C-index
cindex(x1, cbind(x2, x3))
```

---

cv.MTPS *Evaluation using Cross-Validation*

Description

Use cross-validation to evaluate model performance.

Usage

```
cv.MTPS(
  xmat,
  ymat,
  family,
  nfolds = 5,
  cv = FALSE,
  residual = TRUE,
  cv.stacking.nfold = 5,
  method.step1,
  method.step2,
  resid.type = c("deviance", "pearson", "raw"),
  resid.std = FALSE,
  dist1 = NULL,
  weights = NULL
)
```

Arguments

`xmat`	Matrix of predictors, each row is an observation vector
`ymat`	Matrix of outcomes. Quantitative for family = **"gaussian"**. A factor of two levels for family = **"binomial"**. A survival object for family = **"survival"**.
`family`	Response type for each response. If all response variable are within the same family it can be **"gaussian"**, **"binomial"** or **"survival"**, otherwise it is a vector with elements **"gaussian"**, **"binomial"** and **"survival"** to indicate each response family.
`nfolds`	Integer, number of folds for Cross-Validation to evaluate the performance of stacking algorithms.
`cv`	Logical, indicate if use Cross-Validation Stacking algorithm.
`residual`	Logical, indicate if use Residual Stacking algorithm.
`cv.stacking.nfold`	Integer, number of folds for Cross-Validation Stacking algorithm. The default value is 5.
`method.step1`	Base Learners for fitting models in Step 1 of Stacking Algorithm. It can be one base learner function for all outcomes or a list of base learner functions for each outcome. The list of all base learners can be obtained by `list.learners()`.
`method.step2`	Base Learners for fitting models in Step 2 of Stacking Algorithm. (see above)
`resid.type`	The residual type for Residual Stacking.

`resid.std`	Logical, whether or not use standardized residual.
`dist1`	Assumed distribution for survival response. If the argument is a character string, then it is assumed to name an element from survreg.distributions. These include **"weibull"**, **"exponential"**, **"gaussian"**, **"logistic"**,**"lognormal"** and **"loglogistic"**. Otherwise, it is assumed to be a user defined list conforming to the format described in `"survreg.distributions"`.
`weights`	Weight for survival response.

Details

The most frequently used evaluation metric of survival models is the concordance index (C-index). It is a measure of rank correlation between predicted risk scores. Only use uncensored observations

Value

It returns the mean squared error of continuous outcomes. AUC, accuracy, recall and precision for binary outcomes of predictions using cross-validation.

Examples

```
data("HIV")
cv.MTPS(xmat=XX, ymat=YY, family="gaussian", nfolds=2, method.step1=rpart1, method.step2=lm1)
```

`find_km_weights_mat` *Calculate Kaplan-Meier Weights*

Description

Produce the Kaplan-Meier weights

Usage

```
find_km_weights_mat(multi_dat, num_outcome)
```

Arguments

`multi_dat`	A a data frame that contains the time and status of multi-variate survival outcomes for every subject. The name of each column corresponding to the time or status must contain the characters “time” or “status”, respectively.
`num_outcome`	Number of the outcomes

Details

The Kaplan-Meier weights were defined as

$$w_{n1} = \frac{d_{(1)}}{n}, w_{ni} = \frac{d_{(i)}}{n-i+1} \prod_{j=1}^{i-1} (\frac{n-j}{n-j+1})^{d_{(j)}}, i = 2, ..., n.$$

Value

A matrix of Kaplan-Meier weights for each survival outcome.

References

Huang J, Ma S, Xie H. Regularized estimation in the accelerated failure time model with high-dimensional covariates. Biometrics. 2006;62(3):813-820.

Examples

```
#multi-outcome
data(simdat_meps)
find_km_weights_mat(ymat,2)
```

`HIV` *HIV Drug Resistance Database*

Description

The data from HIV Drug Resistance Database used for demonstration. After processing, `YY` contains 5 response variables for 1246 observations and `XX` are 228 predictors of those 1246 observations.

Format

Data objects used for demonstration

Details

In the HIV database, the resistance of five Nucleoside RT Inhibitor (NRTI) drugs were used as multivariate outcomes, including Lamivudine (3TC), Abacavir(ABC), Zidovudine (AZT), Stavudine (D4T), Didanosine (DDI). The mutation variables are used as the predictors. Some mutation variables were removed as they do not contain enough variation. The final outcome data is a matrix of size 1246×5, and the predictor data is a matrix of 1246×228 values, which is provided in the package called "HIV". In the example data in the package, "YY" refers the outcome data and "XX" refers the predictor data.

References

Rhee SY, Taylor J, Wadhera G, Ben-Hur A, Brutlag DL, Shafer RW. Genotypic predictors of human immunodeficiency virus type 1 drug resistance. Proceedings of the National Academy of Sciences. 2006 Nov 14;103(46):17355-60.

Rhee SY, Taylor J, Fessel WJ, Kaufman D, Towner W, Troia P, Ruane P, Hellinger J, Shirvani V, Zolopa A, Shafer RW. (2010). HIV-1 protease mutations and protease inhibitor cross-resistance. Antimicrobial Agents and Chemotherapy, 2010 Oct

Melikian GL, Rhee SY, Taylor J, Fessel WJ, Kaufman D, Towner W, Troia-Cancio PV, Zolopa A, Robbins GK, Kagan R, Israelski D, Shafer RW (2012). Standardized comparison of the relative impacts of HIV-1 reverse transcriptase (RT) mutations on nucleoside RT inhibitor susceptibility. Antimicrobial Agents and Chemother. 2012 May;56(5):2305-13.

Melikian GL, Rhee SY, Varghese V, Porter D, White K, Taylor J, Towner W, Troia P, Burack J, Dejesus E, Robbins GK, Razzeca K, Kagan R, Liu TF, Fessel WJ, Israelski D, Shafer RW (2013).

Non-nucleoside reverse transcriptase inhibitor (NNRTI) cross-resistance: implications for preclinical evaluation of novel NNRTIs and clinical genotypic resistance testing. J Antimicrob Chemother, 2013 Aug 9.

Examples

```
data(HIV)
```

`list.learners` *List Available Base Learners*

Description

This function lists all base learners provided in the package.

Usage

```
list.learners()
```

Details

lm1: linear regression

glm1: generalized linear models

glmnet1: Does k-fold cross-validation to chose best alpha and lambda for generalized linear models via penalized maximum likelihood.

glmnet.lasso: LASSO, lambda is chose by k-fold cross-validation for glmnet

glmnet.ridge: Ridge regression, lambda is chose by k-fold cross-validation for glmnet

rpart1: regression tree

lda1: linear discriminant analysis

qda1: quadratic discriminant analysis

KNN1: k-nearest neighbour classification, k is chose by cross-validation

svm1: support vector machine

surv: parametric survival regression

ela1: AFT model with Elasticnet, weights is the weight of survival time

lasso1: AFT model with Lasso, weights is the weight of survival time

xgb1: AFT model with Xgboost model

aftsrr1: AFT model with Smooth Rank Regression

aftgee1: AFT model with generalized estimating equations

Value

The name of all base learners provided in the package

Examples

```
list.learners()
```

modify.parameter *Modify Default Parameters For Base Learner*

Description

Modify default parameters for methods provided in the package

Usage

```
modify.parameter(FUN, ...)
```

Arguments

FUN	Method
...	Modified arguments

Value

It returns a new function with modified parameters.

Examples

```
glmnet.lasso <- modify.parameter(glmnet1, alpha=1)
glmnet.ridge <- modify.parameter(glmnet1, alpha=0)
```

mse *Mean Square Error Loss*

Description

Compute the mean square error loss in survival analysis.

Usage

```
mse(pre, object)
```

Arguments

pre	A numeric vector of predicted survival time
object	A dataframe or matrix, the first column is observed survival time the second column is survival status

Details

The mean square error is caculated on log-scale, only consider uncensored observations

Value

The value of MSE.

Examples

```
set.seed(1)
# simulate predicted survival time
x1 <- rnorm(100)+5
# simulate observed survival time
x2 <- rnorm(100)+10
# simulate observed survival status
x3 <- rep(c(1, 0), 50 )
# calculate C-index
mse(x1,  cbind(x2, x3))
```

`MTPS` *Fit Models using Revised Stacking Algorithm*

Description

Fit a model using standard stacking algorithm or revised stacking algorithms to simultaneous predict multiple outcomes

Usage

```
MTPS(
  xmat,
  ymat,
  family,
  cv = FALSE,
  residual = TRUE,
  nfold = 5,
  method.step1,
  method.step2,
  resid.type = c("deviance", "pearson", "raw"),
  resid.std = FALSE,
  dist1 = NULL,
  weights = NULL
)
```

Arguments

`xmat`	Predictor matrix, each row is an observation vector
`ymat`	Responses matrix. Quantitative for family = **"gaussian"**. A factor of two levels for family = **"binomial"**. A survival object for family = **"survival"**
`family`	Response type for each response. If all response variable are within the same family it can be "gaussian", "binomial" or "survival", otherwise it is a vector with elements **"gaussian"**, **"binomial"** and **"survival"** to indicate each response family
`cv`	Logical, indicate if use Cross-Validation Stacking algorithm
`residual`	Logical, indicate if use Residual Stacking algorithm
`nfold`	Integer, number of folds for Cross-Validation Stacking algorithm. The default value is 5

`method.step1`	Base Learners for fitting models in Step 1 of Stacking Algorithm. It can be one base learner function for all outcomes or a list of base learner functions for each outcome. The list of all base learners can be obtained by `list.learners()`
`method.step2`	Base Learners for fitting models in Step 2 of Stacking Algorithm. (see above)
`resid.type`	The residual type for Residual Stacking
`resid.std`	Logical, whether or not use standardized residual
`dist1`	Assumed distribution for response variable. If the argument is a character string, then it is assumed to name an element from `"survreg.distributions"`. These include **"weibull"**, **"exponential"**, **"gaussian"**, **"logistic"**, **"lognormal"** and **"loglogistic"**. Otherwise, it is assumed to be a user defined list conforming to the format described in `"survreg.distributions"`
`weights`	Weight for response

Value

It returns a MTPS object. It is a list of 4 parameters containing
information about step 1 and step 2 models and the revised stacking algorithm
method.

Examples

```
data("HIV")
set.seed(1)
xmat <- as.matrix(XX)
ymat <- as.matrix(YY)
id <- createFolds(rowMeans(XX), k=5, list=FALSE)
training.id <- id != 1
y.train <- ymat[training.id, ]
y.test  <- ymat[!training.id, ]
x.train <- xmat[training.id, ]
x.test  <- xmat[!training.id, ]
# Residual Stacking
fit.rs <- MTPS(xmat = x.train, ymat = y.train,
               family = "gaussian",cv = FALSE, residual = TRUE,
               method.step1 = rpart1, method.step2 = lm1)
predict(fit.rs, x.test)
# using different base learners for different outcomes
fit.mixOut <- MTPS(xmat=x.train, ymat=y.train,
                   family="gaussian",cv = FALSE, residual = TRUE,
                    method.step1 = c(rpart1,glmnet.ridge,rpart1,lm1,lm1),
                    method.step2 = c(rpart1,lm1,lm1,lm1, glmnet.ridge))
 predict(fit.mixOut, x.test)
 # Residual Stacking for Survival Analysis
 set.seed(1)
 data("simdat_mtps")
 # prepare training and test set
 id.train <- sample(1:100, 80)
 xmat.train <- xmat[id.train,]
 xmat.test <- xmat[-id.train,]
 ymat.train <- cbind(list(survival::Surv(ymat[id.train,"time01"],
 ymat[id.train,"status01"])),
 list(survival::Surv(ymat[id.train,"time02"],ymat[id.train,"status02"])))
 # Produce the Kaplan-Meier estimator
 weights <- find_km_weights_mat(ymat[id.train,],num_outcome = 2)
 # fit Residual Stacking Model for Survival Data
 fit <- MTPS(xmat.train, ymat.train, family = 'survival', cv=FALSE, residual
 = TRUE, nfold=5, method.step1 = surv,
 method.step2 = lm1, dist1 = "lognormal", weights = weights)
 # predict the survival time on test set
 pre <- predict.MTPS(fit, xmat.test)
```

`multiFit` *Fit models on multiple outcomes*

Description

This function fit individual models to predict each outcome separately.

Usage

```
multiFit(xmat, ymat, method, family = family, dist1 = NULL, weights = NULL)
```

Arguments

`xmat`	Matrix of predictors, each row is an observation vector
`ymat`	Matrix of outcomes. Quantitative for family = **"gaussian"**. A factor of two levels for family = **"binomial"**. A survival object for family = **"survival"**.
`method`	Method for fitting models. It can be one base learner function for all outcomes or a list of base learner functions for each outcome. The list of all base learners can be obtained by `list.learners()`.
`family`	Response type for each response. If all response variable are within the same family it can be **"gaussian"**, **"binomial"** or **"survival"**, otherwise it is a vector with elements **"gaussian"**, **"binomial"** and **"survival"** to indicate each response family
`dist1`	Assumed distribution for response variable. If the argument is a character string, then it is assumed to name an element from survreg.distributions. These include **"weibull"**, **"exponential"**, **"gaussian"**, **"logistic"**,**"lognormal"** and **"loglogistic"**. Otherwise, it is assumed to be a user defined list conforming to the format described in `"survreg.distributions"`.
`weights`	Weight for response

Value

It returns a multiFit object. It is a list of 6 parameters containing information about the fitted models and fitted values for each outcome.

Examples

```
data("HIV")
set.seed(1)
xmat <- as.matrix(XX)
ymat <- as.matrix(YY)
id <- createFolds(rowMeans(XX), k=5, list=FALSE)
training.id <- id != 1
y.train <- ymat[training.id, ]
y.test  <- ymat[!training.id, ]
x.train <- xmat[training.id, ]
x.test  <- xmat[!training.id, ]
fit <- multiFit(xmat = x.train, ymat = y.train,
                method = rpart1, family = "gaussian")
predict(fit, x.test)

# using different base learners for different outcomes
fit.mixOut <- multiFit(xmat = x.train, ymat = y.train,
                       method = c(rpart1, rpart1, glmnet.ridge,lm1,lm1),
                       family = "gaussian")
predict(fit.mixOut, x.test)
```

`predict.MTPS` *Make predictions from a "MTPS" model*

Description

This function makes predictions from a revised stacking model.

Usage

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

Arguments

`object`	A fitted object from `"MTPS"`
`newdata`	Matrix of new predictors at which predictions are to be made
`...`	additional arguments affecting the predictions produced

Value

The predicted value from new predictors.

Examples

```
data("HIV")
set.seed(1)
xmat <- as.matrix(XX)
ymat <- as.matrix(YY)
id <- createFolds(rowMeans(XX), k=5, list=FALSE)
training.id <- id != 1
y.train <- ymat[training.id, ]
y.test  <- ymat[!training.id, ]
```

```
x.train <- xmat[training.id, ]
x.test  <- xmat[!training.id, ]
# Cross-Validation Residual Stacking
fit.rs <- MTPS(xmat = x.train, ymat = y.train,
  family = "gaussian",cv = FALSE, residual = TRUE,
  method.step1 = rpart1, method.step2 = lm1)
pred.rs <- predict(fit.rs, x.test)

# Residual Stacking for Survival Analysis
  set.seed(1)
  data("simdat_mtps")

 # prepare training and test set
 id.train <- sample(1:100, 80)
 xmat.train <- xmat[id.train,]
 xmat.test <- xmat[-id.train,]
 ymat.train <- cbind(list(survival::Surv(ymat[id.train,"time01"],ymat[id.train,"status01"])),
 list(survival::Surv(ymat[id.train,"time02"],ymat[id.train,"status02"])))
 # Produce the Kaplan-Meier estimator
 weights <- find_km_weights_mat(ymat[id.train,],num_outcome = 2)
 # fit Residual Stacking Model for Survival Data
 fit <- MTPS(xmat.train, ymat.train, family = 'survival', cv=FALSE,
 residual = TRUE, nfold=5, method.step1 = surv,
 method.step2 = lm1, dist1 = "lognormal", weights = weights)
 # predict the survival time on test set
 pre <- predict.MTPS(fit, xmat.test)
```

predict.multiFit *Make predictions for multiple outcomes*

Description

This function makes predictions from a multiFit object.

Usage

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

Arguments

`object`	A fitted object from "multiFit"
`newdata`	Matrix of new predictors at which predictions are to be made
`...`	additional arguments affecting the predictions produced

Value

The predicted value from new predictors.

Examples

```
data("HIV")
set.seed(1)
xmat <- as.matrix(XX)
ymat <- as.matrix(YY)
id <- createFolds(rowMeans(XX), k=5, list=FALSE)
training.id <- id != 1
y.train <- ymat[training.id, ]
y.test  <- ymat[!training.id, ]
x.train <- xmat[training.id, ]
x.test  <- xmat[!training.id, ]
fit <- multiFit(xmat = x.train, ymat = y.train,
                method = rpart1, family = "gaussian")
predict(fit, x.test)
```

`simdat_mtps` *A Simulated Database for Survival analysis*

Description

The data is simulated for predicting multiple survival outcomes. `ymat` contains 2 survival response variables with 10% censoring for 100 observations and `xmat` contains 2 predictors of those observations.

Details

In the simdat_mtps database, the simulated survival time and status are used as multivariate outcomes, including **time01**,**status01**,**time02**,**status02**. The predictors both follow a standard normal distribution. In this simulated data, "ymat" refers multiple survival outcomes and "xmat" refers the predictor data.

Examples

```
data(simdat_mtps)
```