Package 'MLSurvival'

June 26, 2019

Title Machine Learning for Survival Analysis

Version 0.1.0

Author person(``Che'', ``Ngufor'', , ``Ngufor.Che@mayo.edu'', role = c(``aut'', ``cre''))

Maintainer Che Ngufor <Ngufor.Che@mayo.edu>

Description Machine learning methods for survival analysis.

Depends R (>= 3.1.2)

Imports PresenceAbsence,
caret,
ranger,
gbm,
glmnet,
kernlab,
xgboost,
glmnetUtils

License GPL(>= 3)

LazyData true

NeedsCompilation no

RoxygenNote 6.1.1

R topics documented:

COX *Function to implement proportional hazard model*

Description

Train the Cox model. Optionally, the regularize cox can also be trained based on the implementation in glmnetUtils.

Usage

```
COX(form, dat, trControl = NULL)
```

Arguments

form	survival formula
dat	data frame
trControl	list of control parameters: 1. maxit: number of iterations in glmnet 2. regularize: train regularize cox? 3. nfolds : number of folds in glmnet 4. lambda : numeric vector of lambda values in glmnet 5. alpha : numeric vector of alpha values in cva.glmnet

Value

returns a list with items:

- model: object of class coxph
- form: survival formula

denormalize *Denormalized data*

Description

Take the ouput of normalize and convert back to original scale.

Usage

```
denormalize(normalized, min, max)
```

Arguments

min	minimum value of each variable in the original data. This value is stored as an attribute of normalize
max	maximum value of each variable in the original data. This value is stored as an attribute of normalize

Value

Original un-normalized data

getResults.ci *get Summary Results*

Description

Takes a table of performance metrics, such as cross-validation results and compute summaries (mean and confidence interval) ready for publication.

Usage

```
getResults.ci(tab, alpha = 0.05)
```

Arguments

tab	table with performance results
alpha	confidence level

Value

data frame with summaries (confidence interavals are represented in brackets)

getResults.ci2 *get Summary Results 2*

Description

Takes a table of performance metrics, such as cross-validation results and compute summaries (mean and confidence interval) ready for publication.

Usage

```
getResults.ci2(tab, alpha = 0.05, groups = c("model", "status"),
  stats = c("PCC", "AUC", "sensitivity", "specificity", "G.mean", "BER",
  "Pos Pred Value"))
```

Arguments

tab	table with performance results
alpha	confidence level
groups	variable in tab to group by

Value

data frame with summaries (confidence interavals are represented in brackets)

MLSurvival *MLSurvival*

Description

Train and evelaute machine learning survival and classification models for time to event data

Usage

MLSurvival(form, dat, method, predict.times, trControl, parallel = FALSE, dummy.vars = TRUE, mc.cores = 2, seed = 123, ...)

Arguments

form	survival formula
dat	data frame
method	character verctor of machine learning algorithms. Implemented algorithms 1. glm logistic regression 2. glmnet elastic net 3. gbm gradient boosting machine 4. ranger random forest 5. svmRadial support vector machine with radial basis kernel 6. xgbTree extreem boosting machine
predict.times	numeric vector containing the survival prediction times
trControl	list of control parameters for caret and the ranger models
parallel	run cross-validation in parallel?
dummy.vars	create dummy variables/model.matrix
mc.cores	number of cores
seed	random seed
...	further arguments passed to caret or other methods.

Value

returns a list with items:

- model: trained survival model
- perf: performance of models at each survival prediction time: PCC, AUC, sensitivity, specificity, g-mean etc.
- perf.ave: average of perf with confidence intervals

---

normalize *Normalize data*

Description

Normalize data to (0,1)

Usage

```
normalize(x)
```

Arguments

x	data frame

Value

Normalized data with attributes min and max representing the min and max of each variable in x

---

opt.thresh *Optimal Threshold*

Description

Compute the optimal classifcation threshold based on the optimal.thresholds function in the Presence.Absence package

Usage

```
opt.thresh(prob, obs, opt.methods = 9)
```

Arguments

prob	predicted probabilities
obs	binary (0-1) ground truth
opt.methods	optimal threshold method. See Presence.Absence package

Value

optimal threshold

Performance.measures *Performance metrics*

Description

Compute several performance metrics

Usage

```
Performance.measures(pred, obs, threshold = NULL)
```

Arguments

pred	predicted probabilities
obs	binary (0-1) ground truth
threshold	optimal threshold.

Value

A data frame with performance metrics.

---

predictSurvProb_Cox *Predict Cox*

Description

Get predicted probabilities from the cox model for new data at different time points

Usage

```
predictSurvProb_Cox(object, newdata, times)
```

Arguments

object	trained cox model. Output of train_Cox
newdata	out of sample data
times	new time points

Value

predicted probabilities at eact time point in times

predictSurvProb_ranger *Predict Ranger*

Description

Get predicted probabilities from the ranger model for new data at different time points

Usage

```
predictSurvProb_ranger(object, newdata, times, ...)
```

Arguments

object	trained ranger model. Output of train_ranger
newdata	out of sample data
times	new time points
...	further arguments passed to caret or other methods.

Value

predicted probabilities at eact time point in times

RSF *RSF: Random Survival Forest*

Description

Train the random survival forest through the ranger package. The optimal RSF tuning parameters: min.node.size,mtry, and splitrule can be selected through grid search.

Usage

```
train_RSF(form, dat, predict.times, trControl = NULL, seed = 123,
  parallel = FALSE, mc.cores = 2, ...)
```

Arguments

form	survival formula
dat	data frame
predict.times	survival prediction times
trControl	list of control parameters: 1. ntrees: mumber of trees 2. number: number of cross-validations 3. tuneLength: tuning paramer grid size 4. importance: ranger variable importance
parallel	run cross-validation in parallel? Uses mclapply which works only on linux
...	further arguments passed to caret or other methods.
tuneLength	same as tuneLength in the caret package

Value

returns a list with items:

- finalModel: final model trained on the complete data (dat) using optimal tuning paramters
- fitted: predictions on complete data (dat)
- threshold: optimal classification threshold
- resamples: cross-validation results: predictions on resampled data
- predict.times: survival prediction times
- bestTune: optimal tuning parameters

---

train.classifier *Predict Survival with classifcation methods*

Description

Train machine learning classification models on time to event data using the caret package

Usage

```
train_classifier(form, dat, method = "gbm", predict.times,
  trControl = NULL, parallel = FALSE, mc.cores = 2, seed = 123,
  ...)
```

Arguments

form	survival formula
dat	data frame
method	classifcation algorithm. The following algorithms have been implemented. 1. glm logistic regression 2. glmnet elastic net 3. gbm gradient boosting machine 4. ranger random forest 5. svmRadial support vector machine with radial basis kernel 6. xgbTree extreme gradient boosting machine
predict.times	survival prediction times
trControl	control parameters for the caret train function. Set to NULL to use a default 5-fold cross-validation
parallel	run cross-validation in parallel? Uses mclapply which works only on linux
...	further arguments passed to caret or other methods.
tuneLength	same as tuneLength in the caret package

Value

returns a list with items:

- finalModel: final model trained on the complete data (dat) using optimal tuning paramters
- fitted: predictions on complete data (dat)
- threshold: optimal classification threshold
- resamples: cross-validation results: predictions on resampled data
- predict.times: survival prediction times
- bestTune: optimal tuning parameters
- method: classification algorithm

---

train.cox *Cox proportional hazard model*

Description

Train the Cox model through cross-validation and select the optimal survival classification threshold. A regularized Cox approach which performs feature selection is also implemeted. For regularize cox, the optimal set of variables is selected through cross-validation and used to train the final model on the complete data

Usage

```
train_cox(form, dat, predict.times, trControl = NULL, parallel = FALSE,
  mc.cores = 2, seed = 123, ...)
```

Arguments

form	survival formula
dat	data frame
predict.times	survival prediction times
trControl	list of control parameters: 1. number: number of cross-validations 2. regularize: train regularize cox?
parallel	run cross-validation in parallel? Uses mclapply which works only on linux
...	further arguments passed to caret or other methods.
tuneLength	same as tuneLength in the caret package

Value

returns a list with items:

- finalModel: final model trained on the complete data (dat) using optimal tuning paramters
- fitted: predictions on complete data (dat)
- threshold: optimal classification threshold
- resamples: cross-validation results: predictions on resampled data
- predict.times: survival prediction times
- bestTune: optimal tuning parameters

VimPlot *Plot variable importance*

Description

Plot variable importance

Usage

```
VimPlot(x, top = min(20, length(x$importance)), ...)
```

Arguments

x	data frame with variable importance
top	number of variables to plot

Index