

Package ‘tgml’

January 22, 2026

Type Package

Title Tree Guided Machine Learning for Personalized Predictions and Precision Diagnostics

Version 0.4.0

Date 2026-01-22

Description Generalization of the classification and regression tree (CART) model that partitions subjects into terminal nodes and tailors machine learning model to each terminal node.

License GPL (>= 2)

Depends R (>= 4.5.0), glmnet, randomForest, e1071, pROC, stats, graphics

NeedsCompilation no

Author Yunro Chung [aut, cre] (ORCID: <<https://orcid.org/0000-0001-9125-9277>>)

Maintainer Yunro Chung <yunro.chung@asu.edu>

Repository CRAN

Date/Publication 2026-01-22 19:00:02 UTC

Contents

tgml	1
Index	6

tgml	<i>Tree Guided Machine Learning</i>
------	-------------------------------------

Description

Generalized classification and tree model that assigns the most effective predictive model to each terminal node.

Usage

```
tgml(y, x, z, ynew, xnew, znew, MLlist, cut, max_depth, min_sample)
```

Arguments

<code>y</code>	Response vector. If a factor coded as 0 or 1, classification is assumed. Otherwise, regression is assumed.
<code>x</code>	Data.frame or matrix of predictors that is used to estimate a tree structure.
<code>z</code>	Data.frame or matrix of predictors that is used in terminal node specific ML models.
<code>ynew</code>	Response vector for the test set corresponding to <code>y</code> (default <code>ynew=NULL</code>).
<code>xnew</code>	Data.frame or matrix for the test set corresponding to <code>x</code> (default <code>xnew=NULL</code>).
<code>znew</code>	Data.frame or matrix for the test set corresponding to <code>z</code> (default <code>znew=NULL</code>).
<code>MLlist</code>	Candidate predictive models that can be assigned to each terminal node (default <code>MLlist=c("lasso", "rf", "svm")</code>). Any other predictive models can be included. See the details below.
<code>cut</code>	Number of percentile-based candidate cutoff values for each <code>x[,j]</code> , $j=1,2,\dots$ (default <code>cut=10</code>). This is only used when <code>x[,j]</code> has the unique values more than <code>cut</code> .
<code>max_depth</code>	Maximum depth of trees. (default <code>max_depth=4</code>)
<code>min_sample</code>	The number of minimum sample size per each node, i.e., <code>length(y)>min_sample</code> if <code>y</code> is continuous and <code>min(length(y==1), length(y==0))>min_sample</code> (default <code>min_sample=20</code>).

Details

The `tgml` function uses recursive partitioning to simultaneously identify subgroups and subgroup-specific predictive models. Specifically, the data (y, x, z) are randomly split into training and validation sets. At each candidate split, predictive models specified in `MLlist` are fitted using the training set, and the optimal split is selected based on the validation MSE or BS.

Ideally, two distinct sets of predictors are available: `x` and `z` (e.g., clinical variables and biomarkers), where `x` is used to construct the tree splits and `z` is used for terminal-node-specific predictive models. When such a separation is not feasible, individualized prediction of `y` given `x` is also allowed by using the same variable `x` for both roles, for example:

```
tgml(y = y, x = x, z = x, ynew = ynew, xnew = xnew, znew = xnew).
```

Regarding node numbering, each internal node `s` has left and right child nodes indexed as `2s` and `2s+1`, respectively. The root node is indexed as node 1; nodes 2 and 3 are the left and right children of node 1; nodes 4 and 5 are the left and right children of node 2; and so on.

Currently, terminal-node-specific predictive models include `lasso()`, `randomForest()`, and `svm(..., kernel = "radial")` from the R packages `cv.glmnet`, `randomForest`, and `e1071`, respectively. Additional models can be flexibly incorporated; see Example 3 below for an illustration.

Value

An object of class tgml, which is a list with the following components:

terminal	Terminal node numbers.
internal	Internal node numbers.
splitVariable	splitVariable[k] (i.e., x[,k]) is used to split the internal node k.
cutoff	cutoff[k] is the cutoff value to split the internal node k.
selML	selML[k] is ML model assigned to the terminal node t.
fitML	fitML[[t]] is the fitted ML model at the terminal node t.
y_hat	Estimated y (or estimated probability) on the training set (y,x,z) if y is continuous (or binary).
node_hat	Estimated node on the training set.
mse	Training MSE.
bs	Training Brier Score.
roc	Training ROC curve.
auc	Training AUC.
y_hat_new	Estimated y (or estimated probability) on the test set (ynew,xnew,znew) if y is continuous (or binary).
node_hat_new	Estimated node on the test set.
mse_new	Test MSE.
bs_new	Test Brier Score.
roc_new	Test ROC curve.
auc_new	Test AUC.

Author(s)

Yunro Chung [aut, cre]

References

Nishtha Shah, Hassan Ghasemzadeh and Yunro Chung, Treed-guided machine learning for precision diagnostics (in preperation)

Examples

```
set.seed(99)
###
#1. continuous y
###
n=200*2 #n=200 & 200 for training & test sets

x=matrix(rnorm(n*10),n,10) #10 predictors
z=matrix(rnorm(n*10),n,10) #10 biomarkers
```

```

xcut=median(x[,1])
subgr=1*(x[,1]<xcut)+2*(x[,1]>=xcut) #2 subgroups

lp=rep(NA,n)
for(i in 1:n)
  lp[i]=1+3*z[i,subgr[i]]
y=lp+rnorm(n,0,1)

idx.nex=sample(1:n,n*1/2,replace=FALSE)
ynew=y[idx.nex]
xnew=x[idx.nex,]
znew=z[idx.nex,]

y=y[-idx.nex]
x=x[-idx.nex,]
z=z[-idx.nex,]

fit1=tgml(y,x,z,ynew=ynew,xnew=xnew,znew=znew)
fit1$mse_new
plot(fit1$y_hat_new~ynew,ylab="Predicted y",xlab="ynew")
abline(a=0,b=1)

###
#2. binary y
###
x=matrix(rnorm(n*10),n,10) #10 predictors
z=matrix(rnorm(n*10),n,10) #10 biomarkers

xcut=median(x[,1])
subgr=1*(x[,1]<xcut)+2*(x[,1]>=xcut) #2 subgroups

lp=rep(NA,n)
for(i in 1:n)
  lp[i]=1+3*z[i,subgr[i]]
prob=1/(1+exp(-lp))
y=rbinom(n,1,prob)
y=as.factor(y)

idx.nex=sample(1:n,n*1/2,replace=FALSE)
ynew=y[idx.nex]
xnew=x[idx.nex,]
znew=z[idx.nex,]

y=y[-idx.nex]
x=x[-idx.nex,]
z=z[-idx.nex,]

fit2=tgml(y,x,z,ynew=ynew,xnew=xnew,znew=znew)
fit2$auc_new
plot(fit2$roc_new)

###
#3. add new ML models

```

```

# 1) write two functions:
#   c_xx & c_xx_predict if y is continuous or
#   b_xx & b_xx.predict if y is binary
# 2) update MLlist that includes xx, not c_xx nor b_xx.
# 3) run tgml using updated MLlist.
# The below is an example of adding ridge regression.
###
#3.1. ridge regression for continuous y.
c_ridge=function(y,x){
  x=data.matrix(x)
  fit=NULL
  suppressWarnings(try(fit<-glmnet::cv.glmnet(x,y,alpha=0),silent=TRUE))
  return(fit)
}
c_ridge_predict=function(fit,xnew){
  y.hat=rep(NA,nrow(xnew))
  if(!is.null(fit)){
    xnew=data.matrix(xnew)
    y.hat=as.numeric(predict(fit,newx=xnew,s="lambda.min",type="response"))
  }
  return(y.hat)
}

#3.2. ridge regression for binary y.
b_ridge=function(y,x){
  x=data.matrix(x)
  fit=NULL
  suppressWarnings(try(fit<-glmnet::cv.glmnet(x,y,alpha=1,family="binomial"),silent=TRUE))
  return(fit)
}
b_ridge_predict=function(fit,xnew){
  y.hat=rep(NA,nrow(xnew))
  if(!is.null(fit)){
    xnew=data.matrix(xnew)
    y.hat=as.numeric(predict(fit,newx=xnew,s="lambda.min",type="response"))
  }
  return(y.hat)
}

#3.3. update MLlist
MLlist=c("lasso","ridge")
fit3=tgml(y,x,z,ynew=ynew,xnew=xnew,znew=znew,MLlist=MLlist)
fit3$auc_new
plot(fit3$roc_new)

```

Index

tgml, [1](#)