simpleEnsembleGroup3

```r
library(simpleEnsembleGroup3)
#> Loading required package: glmnet
#> Loading required package: Matrix
#> Loaded glmnet 4.1-8
#> Loading required package: randomForest
#> randomForest 4.7-1.1
#> Type rfNews() to see new features/changes/bug fixes.
```

Introduction

This is a r package use to solve regression and classification task. Current version support linear, logistic, ridge, lasso, elastic and randomfores model. Package takes as input a response variable y and matrix of candidate predictors/independent variables X, where each column is a predictor. Package work for both binary y and continuous y. binary y must be a factor variable with level 0 and 1. The predictors X can be combinations of continuous, discrete and binary predictors. If y is binary, do classification task. otherwise do regression task.

Running summaries

First the package check the number of predictors p and the number of size n. If p >> n, package have the option that allows users to pre screening for top K most "informative" predictors to be included in the model.

Second, do the task with the model

Third, package have the option to perform "bagging" for linear, logistic, ridge, lasso and elastic net and randomforest models.

Finally, package have the option that allows users to choose if they want "ensemble" learning.

In classification task, the package evaluate the predicated y with accuracy. In regression task, the package evaluate the predicated y with mean square error.

fucntion detail

main.R

The main function is mainly served as user interface, it allows users to input their data and their command of what kinds of regression functions to implement on the data, moreover, it also checks the integrity of the data, it checks if the predictor data (mainly X) are in dataframe form, if not it will prompt the user to input it into one; it checks if the data has missing value; It checks if the regression model that user wanted to implement matches the one our package support ("linear&logistic", "ridge", "lasso", "elastic", "randomforest"), if it matches, it will call the "function" package and perform corresponding operation; It checks if the predictors and outcome has the same size; It will prompt user to change the outcome data into 0 or 1 if data type is binary. It will prompt the user to choose the most informative predictors if the predictors have greater number than sample size; It will convert categorical data to dummy variables; It will also allow users to choose if they want to perform bagging learning on the data, if yes, the bagging function in the bagging package will be called. Lastly, it also prompts users to choose if they want to perform ensemble learning on the data, if yes, the ensemble.learn function in the ensemble package will be called. The input for the yes and no questions from users are managed by the interactive.check function in the “function” package.

##function.R: This package mainly serves as a bridge between the user input and the other regression model function and other data learning functions.

##ridge.R: This function will enable users to perform ridge regression. First call cv.glmnet, do cross-validation and select the best lambda. Then train the ridge model based on the best lambda.

##lasso.R: This function will enable users to perform lasso regression. First call cv.glmnet, do cross-validation and select the best lambda. Then train the lasso model based on the best lambda.

##elastic.R This function will enable users to perform elastic regression. Loop through alpha from 0 to 1 and do cross-validation tasks on each alpha value. For each alpha value, do elastic regression based on the best lambda value and compare with each other. Then choose the best elastic model based on the alpha value.

##bagging.R: This function will promote users to determine the rounds of bootstrap sampling and perform the specific regression model for those rounds. Finally, this function will aggregate the final result by averaging from all the predicted values from previous rounds.

##ensemble.R: This package enables ensemble learning for the data. The function ensemble.learn takes in X and y, and criteria and comb as input, criteria can be binary and continuous, comb are the different kinds of regression models that user wants to perform. For each regression model, it will perform bagging learning on them, the results from different will be combined into a matrix, each column is the result of a different regression model. If the criteria is binary, then the majority voting will be performed by calling the vote.max function, and result will be returned with values predicted and the accuracy. The vote.max function chooses the mode of the binary vector. If the criteria is continuous, then the the mean of matrix of each column will be found and returned as the predicted values, mean square error(MSR) will be calculated and it will return the user the MSR and the predicted value

output detail

output is a list contain original(mandatory), bagging and ensemble result(optional). the original result is the default task with the model. bagging result is bagging task and ensemble result is the ensemble task. Each result contains two value, predicated y and the evaluated result(acc or msr based on the task).

Running summaries

case 1: binary case

```r
load("~/Downloads/try/simpleEnsembleGroup3/data/banknote.rda")
banknote$class <- as.factor(banknote$class)
banknote.y <- banknote$class
banknote.b <- banknote[,-5]
# use model from ("linear&logistic", "ridge", "lasso", "elastic", "randomforest")
#result <- simpleEnsembleGroup3(banknote.b, banknote.y, "lasso")
```

case 2: continous case

```
load("~/Downloads/try/simpleEnsembleGroup3/data/mad.rda")
mad.x <- mad[,-16]
# use model from ("linear&logistic", "ridge", "lasso", "elastic", "randomforest")
#result <- simpleEnsembleGroup3(mad.x, mad$y, "ridge")
```

case 3: p >> n

```
        dat <-
read.delim("https://www.ams.sunysb.edu/~pfkuan/Teaching/AMS597/Data/leukemiaDataSet.txt",
header=T ,sep='\t')
        dat$Group <- ifelse(dat[, 1] == "AML", 1, 0)
        dat$Group <- as.factor(dat$Group)
        X <- dat[,-1]
        Y <- dat$Group
        # use model from ("linear&logistic", "ridge", "lasso", "elastic", "randomforest")
        #result <- simpleEnsembleGroup3(X, Y, "randomforest")
```