Statistical Model Manual

Part I is the usage of the model, including arguments and coding details. Part II gives an example.

Part I Model

We use lasso and group lasso to do the variable selection, and fit a logistic regression model for binary classification. Can deal with all shapes of data, including very large sparse data matrices. Return output for test data, including probabilities and classifications.

Input

reportStats Function
reportStats(data, y_name, missing_data, is_numeric, nominal_col, ordinal_col, group_lasso, a)

- **data**
  A data set with n rows (observations) and (p+1) columns (p variables and 1 binary response variable). An ordinal variable with >= 10 levels will be treated as continues one. In this case, please make sure the levels are created in increasing alphabetical order.
- **y_name**
  The name of response variables, i.e."missingAgain". Response variable must be binary.
- **missing_data**
  Does the given data set contain any missing values? (TRUE or FALSE) If TRUE, the observations with missing values will be ignored.
- **nominal_col**
  A vector of column indexes of nominal variables should be provided if is_numeric = TRUE. Otherwise give 0 to this argument then the function will detect all categorical variables automatically.
  ex.1 if column 1 is the only nominal variable, nominal_col = 1;
  ex.2 if column 2, 3, 4 are nominal variables, nominal_col = c(2, 3, 4).
  Nominal Variable*:*
  A categorical variable has several values but the order does not matter. For instance, male or female categorical variable do not have ordering. Recode males as 1 and females as 2.
- **ordinal_col**
  A vector of column indexes for ordinal variables. ex.1 if column 1 is the only categorical variable, ordinal_col = 1; ex.2 if column 2, 3, 4 are ordinal variables, ordinal_col = c(2, 3, 4).
  Ordinal Variable*:*
  Ordinal variables have a natural ordering. For example, speed of a moving object - slow, medium, fast and super fast. There is a natural ordering in it Slow < Medium < Fast < Super Fast.
- **group_lasso**
  If TRUE, use group lasso model, otherwise use plain lasso.
  If the given data have at least one categorical variable with 3 or more levels, group lasso is suggested.
- **train_pct**
  Percentage of data for training, the rest would be used for testing.

Output

The following output will be returned from ***reportStats Function***:

- ROC curve
  ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It is created by plotting the true positive rate (sensitivity) against the false positive rate ($1 -$ specificity) at various threshold settings. We can get the "best" threshold which has the highest accuracy among all threshold settings, in the meanwhile, the specificity and sensitivity will be returned at this threshold.
- Prediction of test data
  Plot each person's probabilities (risks) of missing again & classification with the "best" threshold.
- Selected variables
  All selected variables and their coefficients will be returned.
- Missing proportion. The proportion of missing values are returned, including the observations with missing values' percentage, the proportion of missing y(outcome variable) and missing x(predictor variables).
- Accuracy The accuracy and threshold of the classifier are returned

R code

Load packages

```r
# loading required packages: glmnet, pROC, readxl
suppressPackageStartupMessages({
  library(glmnet);
  library(gglasso);
  library(pROC);
  library(readxl)
})
```

Function

```r
## mp_data is a data frame for the target data set
## a is the proportion of training data
reportStats <- function(data, y_name, missing_data, nominal_col,
                        ordinal_col, group_lasso, train_pct) {
  ####################################Data Pre Processing##################################
  ############# delete obs including missing values
  set.seed(33)
  if(missing_data==TRUE){
    data <- data[complete.cases(data), ]
  }else{
    data <- data
  }
  ############# factor categorical variables
  data[, as.vector(nominal_col)] <-
    data.frame(sapply(mp_data[, as.vector(nominal_col)], factor))
  data[, as.vector(ordinal_col)] <-
    data.frame(sapply(mp_data[, as.vector(ordinal_col)], factor))
  ############# ordinal>10 levels, transfer to continuous
  for(i in ordinal_col){
    if(length(levels(data[,i]))>=10){
      data[, ordinal_col] <- sapply(mp_data[, ordinal_col], as.numeric)
    }
  }
  ############# recode y
  names(data)[names(data)==y_name] <- "y"
  data$y <- as.factor(data$y)
  data$y <- ifelse(data$y != levels(data$y)[1], 1, -1)
  ############# training & test data
  trainDataIndex <- sample(1:nrow(data), train_pct*nrow(data))
  trainData <- data[trainDataIndex, ]
  testData <- data[-trainDataIndex, ]
  Y_train <- data$y[trainDataIndex]
  Y_test <- data$y[-trainDataIndex]
  #######################################################################################



  ######################Logistic Model & Lasso Variables Selection#####################
  ############# data & group
  X1 <- model.matrix(y ~., data = data.frame(trainData))
  X <- X1[,-1]
  group <- attributes(X1)$assign[-1]
  X1_test <- model.matrix(y ~., data = data.frame(testData))
  X_test <- X1_test[,-1]
  if(group_lasso==TRUE){
    #################################### group lasso
    ############# fit group lasso penalized logistic regression
    # finds the best lambda parameter by cross validation
    # and returns the corresponding model
    cv <- cv.gglasso(x=X ,y=Y_train,group=group,loss="logit",pred.loss = "loss")
    model <- gglasso(x=X ,y=Y_train,group=group,loss="logit", lambda = cv$lambda.min)
    ############# test
    l <- predict(model, newx=X_test, type="link")
    prob <- exp(l) / (1 + exp(l))
  } else{
    #################################### lasso
    # finds the best lambda parameter by cross validation
    # and returns the corresponding model
    cv <- cv.glmnet(X, y=Y_train, alpha=1, family="binomial",type.measure = "mse")
    model <- glmnet(X, y=Y_train, alpha=1, family="binomial", lambda=cv$lambda.min)
    # test data
    prob <- predict(model,newx = X_test, s=cv$lambda.1se, type="response")
  }
  #######################################################################################
```

```r
######################################Prediction########################################
# selected variables
coefs <- as.matrix(coef(cv))
ix <- which(abs(coefs[,1]) > 0)
selected_coefs_n <- length(ix) - 1
selected_coefs <- coefs[ix,1, drop=FALSE]
# coef transformation
Raw_Coef <- round(selected_coefs, digits=6);
Exponentiated_Coef <- round(exp(selected_coefs),digits=6);
Change_in_Odds <- paste0(format((Exponentiated_Coef-1)*100,digits=6), "%")
table <- data.frame(cbind(Raw_Coef, Exponentiated_Coef, Change_in_Odds))[-1,]
colnames(table)<- c("Raw Coef", "Exponentiated Coef", "Change in Odds")
# choose the best cut off
modelroc1 <- roc(Y_test, as.vector(prob))
threshold <- coords(modelroc1, "best", "threshold")
# probabilities(risks) to predictions
predict1 <- ifelse(as.vector(prob)>threshold[1],1,-1)
# accuracy, confusion matrix
accuracy <- mean(predict1==Y_test)
confusion_matrix <- table(pred=predict1,true=Y_test)
# missing data proportion
p_miss_obs <- 1-sum(complete.cases(data))/nrow(data)
p_miss_y <- sum(is.na(data$y))/nrow(data)
XX <- data[ ,!(names(data) %in% "y")]
p_miss_x <- sum(is.na(XX))/prod(dim(XX))
########################################################################################



########################################Output##########################################
print("*********************************************************************************")
print("****************************Statistical Report**********************************")
# ROC curve
print("ROC curve")
plot(modelroc1, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1,0.2),
     grid.col=c("green","red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue",print.thres=TRUE)
cat("Prediction of test data", "\n")
cat("Plot each person's probabilities of missing again", "\n")
# predict test data
plot(prob, pch = 16, col="grey", ylab = "Probability")
points(which(predict1!=Y_test),
       as.vector(prob)[which(predict1!=Y_test)],
       pch = 13,col="red", cex = 1.2)
abline(h=threshold[1], col="purple")
print(threshold)
print(paste('Selected variables:', selected_coefs_n))
print(table)
cat("Confusion Matrix", "\n")
print(confusion_matrix)
print(paste('Accuracy:', format(accuracy,digits=2)))
# missing proportion
print(paste("The proportion of observations with missing values:", p_miss_obs))
```

```
  print(paste("The proportion of missing y(outcome variable):", p_miss_y))
  print(paste("The proportion of missing x(predictor variables):", p_miss_x))
  print("******************************************************************************")
  #######################################################################################
}
```

Part II Example

Read in data

```
# read data
mp_data1 <- readxl::read_xlsx("data/test.xlsx")
```

```
## New names:
## * `` -> ...1
```

```
mp_data2 <- readxl::read_xlsx("data/train.xlsx")
```

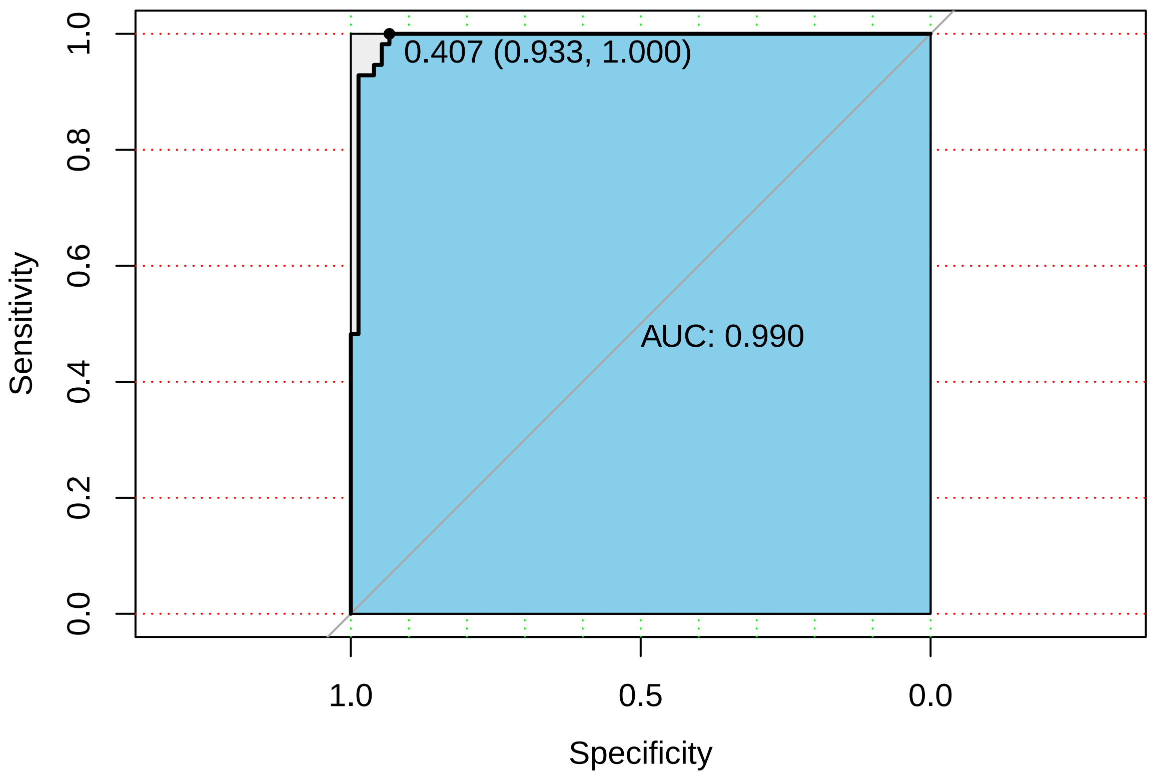
```
## New names:
## * `` -> ...1
```

```
mp_data <- rbind(mp_data1, mp_data2)
mp_data <- data.frame(mp_data)
mp_data <- mp_data[,-1]
```

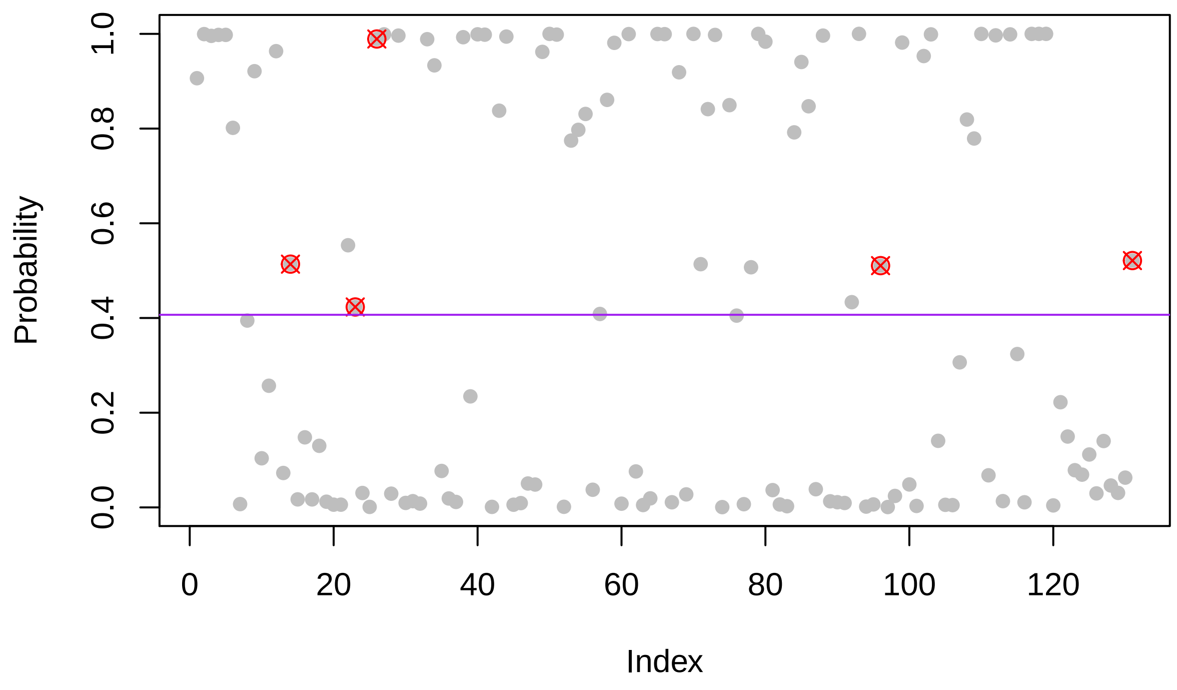
Output

```
reportStats(data=mp_data, y_name="missingAgain", missing_data=FALSE,
  nominal_col=c(1, 71, 75, 78, 74), ordinal_col=2, group_lasso=TRUE, train_pct=0.7)
```

```
## [1] "******************************************************************************"
## [1] "****************************Statistical Report********************************"
## [1] "ROC curve"
```



```
## Prediction of test data
## Plot each person's probabilities of missing again
```



```
##   threshold specificity sensitivity
##   0.4067741   0.9333333   1.0000000
## [1] "Selected variables: 5"
##                Raw Coef Exponentiated Coef Change in Odds
## age_last       0.006029           1.006048        0.6048%
## missing_period 0.034148           1.034737        3.4737%
## date_last_seen 0.001545           1.001546        0.1546%
## occ_num        0.000495           1.000495        0.0495%
## y_coordinate   0.000867           1.000867        0.0867%
## Confusion Matrix
##     true
## pred -1  1
##   -1 70  0
##   1   5 56
## [1] "Accuracy: 0.96"
## [1] "The proportion of observations with missing values: 0"
## [1] "The proportion of missing y(outcome variable): 0"
## [1] "The proportion of missing x(predictor variables): 0"
## [1] "*******************************************************************************"
```