finalbzhangfinalfinal

JIASHU MIAO

6/4/2019

```
datafull <- read.csv("full.csv")
  # full <- datafull
dataz2 <- datafull
# datafull$Gender %>% table()
 datafull$Gender <- ifelse(datafull$Gender=="Female",1,0)
# datafull$Gender

pei=data.frame(table(datafull$READ))
pei

## [1] Freq
## <0 rows> (or 0-length row.names)

7331    /68890

## [1] 0.106416

library(tidyverse)  # data manipulation and visualization

## ── Attaching packages ──── tidyverse 1.2.1 ──

## ✔ ggplot2 3.1.0     ✔ purrr   0.2.5
## ✔ tibble  2.1.1     ✔ dplyr   0.7.8
## ✔ tidyr   0.8.2     ✔ stringr 1.3.1
## ✔ readr   1.3.1     ✔ forcats 0.3.0

## ── Conflicts ─────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(modelr)     # provides easy pipeline modeling functions
library(broom,warn.conflicts = F)
library(ggplot2)
library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library("pROC")

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library("ranger")

dataz2$los_days <- as.integer(as.character(dataz2$los_days))

## Warning: NAs introduced by coercion

dataz2$LOS_Score <- dataz2$LOS_Score %>% as.character() %>% as.integer()

## Warning in function_list[[k]](value): NAs introduced by coercion

 #check there is NA
dim(dataz2)

## [1] 77000    23

dataz2 <- dataz2[!(is.na(dataz2$LOS_Score)),]
# we take out 11 rows that contains NA and they are just a few so we co
uld remove simply remove the rows

any(colSums(is.na(dataz2))>0) #no NAs now.

## [1] FALSE

SM_inf=0
for (i in 1:23)
{ SM_inf <- sum(is.infinite(dataz2[,i]))
    if (!SM_inf==0)
       print(SM_inf)
}
SM_null=0
for (i in 1:23)
{ SM_null <- sum(is.null(dataz2[,i]))
    if (!SM_null==0)
       print(SM_null)
}

SM_nan=0
for (i in 1:23)
{ SM_nan <- sum(is.nan(dataz2[,i]))
    if (!SM_nan==0)
       print(SM_nan)
}
dim(dataz2)

## [1] 76221    23

datafull=dataz2

# datafull$CMS_Readmission_unplanned_flag %>% levels()
datafull$READ <- as.integer(datafull$CMS_Readmission_unplanned_flag)-2
datafull$Clarity.LACE..Score <- as.integer(datafull$Clarity.LACE..Score)

set.seed(123)
train.index <- createDataPartition(datafull$READ,p = .75, list = FALSE)
 new_train <- datafull[train.index,]
 new_test <- datafull[-train.index,]

dim(new_train)

## [1] 57166    24

dim(new_test)

## [1] 19055    24

new_train %>% names()

##  [1] "Gender"                         "Age"

##  [3] "EDV_Score"                      "cadm_type_score"

##  [5] "LOS_Score"                      "Charlson_Score"

##  [7] "ALC_Score"                      "Elective_adm_Score"

##  [9] "Urgent_adm_Score"               "Teach_Score"

## [11] "Male_Score"                     "Admit_DT"

## [13] "Discharge_DT"                   "los_days"

## [15] "ADMIT_SOURCE"                   "dis_from_base_class"

## [17] "loc_name"                       "DISCH_DISPOSITION"

## [19] "marital_status"                 "PAT_HOMELESS_YN"

## [21] "zip"                            "CMS_Readmission_unplanned_fla
g"
## [23] "Clarity.LACE..Score"            "READ"
## single
set.seed(123)
single_logistic_model <-
   glm(
   data = new_train,
   formula = READ~Clarity.LACE..Score,
   binomial(link="logit"))
summary(single_logistic_model)
```

```
##
## Call:
## glm(formula = READ ~ Clarity.LACE..Score, family = binomial(link = "
logit"),
##     data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7756  -0.5293  -0.3963  -0.2947   2.8569
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -4.1316547  0.0550712  -75.02   <2e-16 ***
## Clarity.LACE..Score  0.0338960  0.0008833   38.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36265  on 57165  degrees of freedom
## Residual deviance: 34513  on 57164  degrees of freedom
## AIC: 34517
##
## Number of Fisher Scoring iterations: 5
```

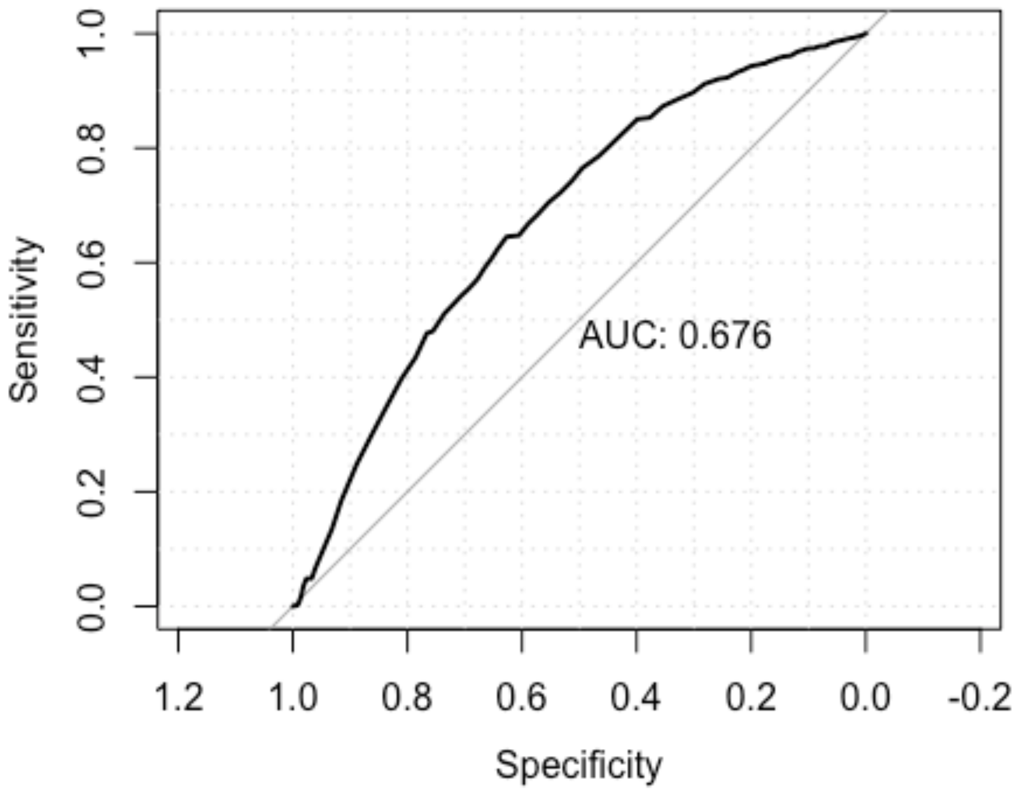
```
exp(coef(single_logistic_model))
```

```
##         (Intercept) Clarity.LACE..Score
##          0.01605629          1.03447697
```

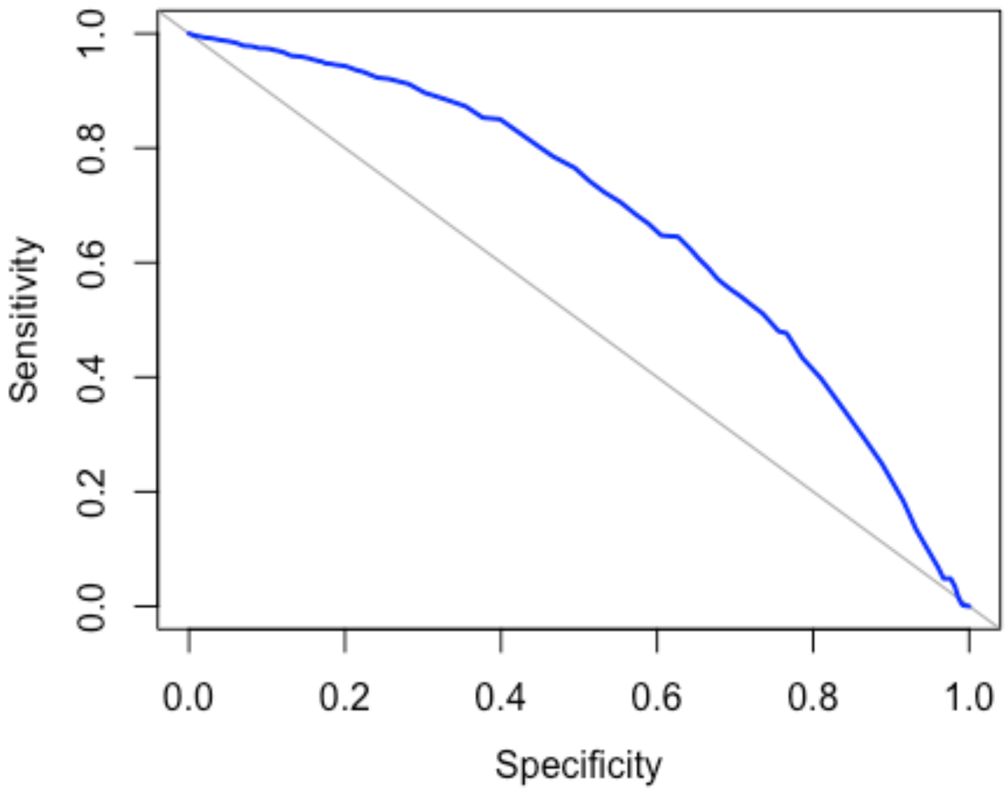
```
single_log_pre <- predict(single_logistic_model,new_test, type= "respon
se")
length(single_log_pre)
```

```
## [1] 19055
```

```
roccurve <- roc(READ~single_log_pre,data = new_test, plot=TRUE,  grid=T
RUE, print.auc=TRUE)
```



```
print(roccurve)

##
## Call:
## roc.formula(formula = READ ~ single_log_pre, data = new_test,     pl
ot = TRUE, grid = TRUE, print.auc = TRUE)
##
## Data: single_log_pre in 17237 controls (READ 0) < 1818 cases (READ
1).
## Area under the curve: 0.6764

plot(roccurve,xlim=c(0,1),col="blue",asp = NA)
```



```r
coords(roc=roccurve,"best","threshold")

##   threshold specificity sensitivity
##   0.1013310   0.6268492   0.6452145

classifer <- as.numeric(coords(roc=roccurve,"best","threshold"))
# Area under the curve: 0.6764
class <- ifelse(single_log_pre>classifer[1],1,0)
accuracy <- 1-mean(class!=new_test$READ)
cat("Accuracy is for simple logistic ", accuracy,"\n")

## Accuracy is for simple logistic  0.6286014

cat("The AUC score is ",roccurve$auc,"\n")

## The AUC score is  0.6763661

set.seed(16)
#Multivariate
# multi_logistic_model <-
#    glm(
#    data = new_train,
#    formula = READ~Clarity.LACE..Score+Age+los_days+,
#     family=binomial(link="logit")
#   )


multi_logistic_model2 <-
   glm(
   data = new_train[,c(3:11,24)],
   formula = READ~.,
    family=binomial(link="logit")
  )
summary(multi_logistic_model2)
```

```
##
## Call:
## glm(formula = READ ~ ., family = binomial(link = "logit"), data = ne
w_train[,
##     c(3:11, 24)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4958  -0.4803  -0.3282  -0.2264   3.0136
##
## Coefficients: (2 not defined because of singularities)
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.530269   0.057960 -78.162  < 2e-16 ***
## EDV_Score             0.188342   0.006842  27.529  < 2e-16 ***
## cadm_type_score       0.025770   0.003041   8.473  < 2e-16 ***
## LOS_Score             0.177146   0.006051  29.277  < 2e-16 ***
## Charlson_Score        0.070337   0.005865  11.993  < 2e-16 ***
## ALC_Score                   NA         NA      NA       NA
## Elective_adm_Score    0.035108   0.007406   4.741 2.13e-06 ***
## Urgent_adm_Score      0.019324   0.001425  13.558  < 2e-16 ***
## Teach_Score                 NA         NA      NA       NA
## Male_Score            0.006149   0.009941   0.619    0.536
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36265  on 57165  degrees of freedom
## Residual deviance: 31944  on 57158  degrees of freedom
## AIC: 31960
##
## Number of Fisher Scoring iterations: 6
```

```
exp(coef(multi_logistic_model2))
```

```
##        (Intercept)          EDV_Score    cadm_type_score
##         0.01077777         1.20724682         1.02610531
##          LOS_Score     Charlson_Score          ALC_Score
##         1.19380532         1.07286931                 NA
## Elective_adm_Score   Urgent_adm_Score        Teach_Score
##         1.03573184         1.01951167                 NA
##         Male_Score
##         1.00616761
```

```
multi_log_pre <- predict(multi_logistic_model2,new_test[,c(3:11,24)], t
ype= "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be mislead
ing
```

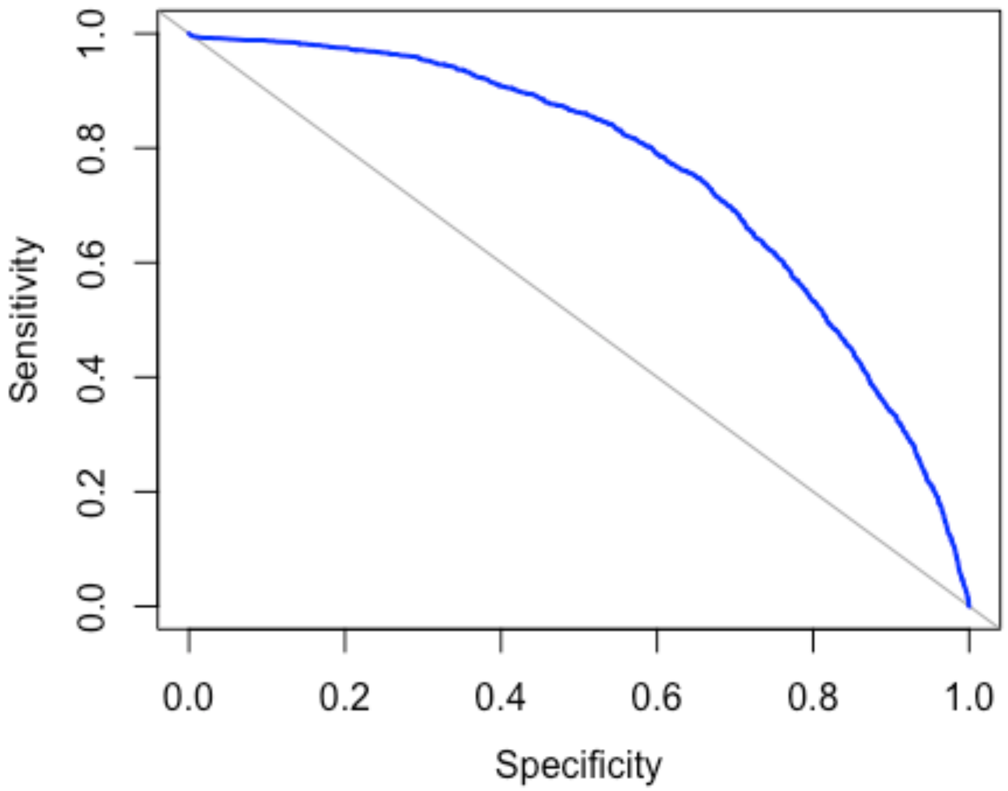
```
length(multi_log_pre)
```

```
## [1] 19055
```

```
roccurve_multi <- roc(READ~multi_log_pre,data = new_test)
print(roccurve_multi)
```

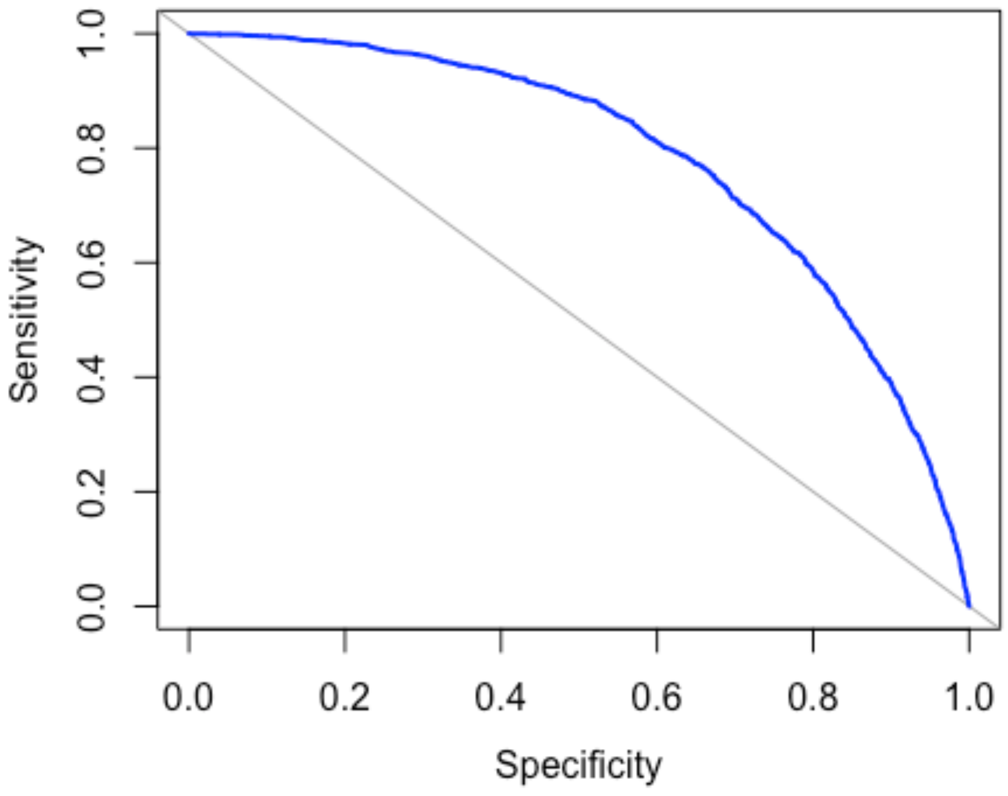
```
##
## Call:
## roc.formula(formula = READ ~ multi_log_pre, data = new_test)
##
## Data: multi_log_pre in 17237 controls (READ 0) < 1818 cases (READ 1).
## Area under the curve: 0.7588
```

```
plot(roccurve_multi,xlim=c(0,1),col="blue",asp = NA)
```



```
coords(roc=roccurve_multi,"best","threshold")

##   threshold specificity sensitivity
##  0.08967944  0.65579857  0.74642464

classifer_multi <- as.numeric(coords(roc=roccurve_multi,"best","thresho
ld"))
# Area under the curve: 0.6764
class_multi <- ifelse(multi_log_pre>classifer_multi[1],1,0)
table(class_multi)

## class_multi
##     0     1
## 11765  7290

accuracy_multi <- 1-mean(class_multi!=new_test$READ)
cat("Accuracy is for multivariate logistic ", accuracy_multi,"\n")

## Accuracy is for multivariate logistic  0.664445

cat("The AUC score is ",roccurve_multi$auc,"\n")

## The AUC score is  0.7588068
```

```
# random forest

set.seed(16)
ranger_read3 <- ranger(
    formula   = READ  ~ .,
    data      = new_train[,c(3:11,24)],
    num.trees = 1000,
    mtry      = 5,
    sample.fraction = .55,
    min.node.size=7,
    importance = "impurity"
  )

## Growing trees.. Progress: 85%. Estimated remaining time: 5 seconds.

# ranger_read2 <- ranger(
#     formula   = READ  ~ .,
#     data      = new_train[,c(1,2,3,4,5,6,8,9,11,14,24)],
#     num.trees = 1000,
#     mtry      = 5,
#     sample.fraction = .55,
#     min.node.size=7,
#     importance = "impurity"
#   )

rangerpre2 <- predict(ranger_read3,new_test,type = "response")
rangerpre2$predictions %>% range

## [1] 0.0001826687 0.7181991760

logpre_rf <- rangerpre2$predictions
roccurve_rf <- roc(READ~logpre_rf,data = new_test)
plot(roccurve_rf,xlim=c(0,1),col="blue",asp = NA)
```

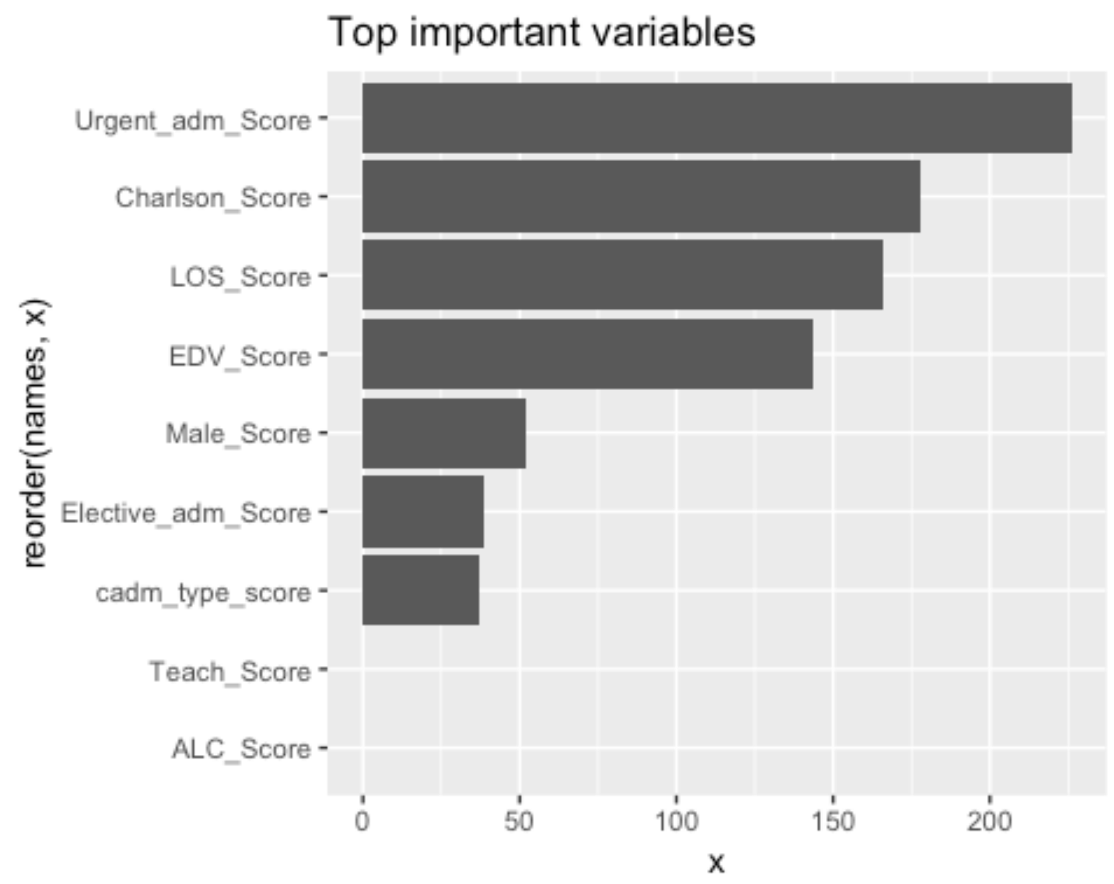


```
coords(roc=roccurve_rf,"best","threshold")

##   threshold specificity sensitivity
##  0.08616275  0.65980159  0.76677668

classifer_rf <- as.numeric(coords(roccurve_rf,"best","threshold"))
class_rf <- ifelse(logpre_rf>classifer_rf[1],1,0)
table(class_rf)

## class_rf
##     0     1
## 11797  7258

accuracy_rf <- 1-mean(class_rf!=new_test$READ)
cat("Accuracy is for random forest ", accuracy_rf,"\n")

## Accuracy is for random forest  0.6700079

cat("The AUC score is ",roccurve_rf$auc,"\n")

## The AUC score is  0.7808986

# names(new_train[,c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,24)])

new_train %>% names()
```

```
##  [1] "Gender"                          "Age"                            
##  [3] "EDV_Score"                       "cadm_type_score"                
##  [5] "LOS_Score"                       "Charlson_Score"                 
##  [7] "ALC_Score"                       "Elective_adm_Score"             
##  [9] "Urgent_adm_Score"                "Teach_Score"                    
## [11] "Male_Score"                      "Admit_DT"                       
## [13] "Discharge_DT"                    "los_days"                       
## [15] "ADMIT_SOURCE"                    "dis_from_base_class"            
## [17] "loc_name"                        "DISCH_DISPOSITION"              
## [19] "marital_status"                  "PAT_HOMELESS_YN"                
## [21] "zip"                             "CMS_Readmission_unplanned_fla
g"
## [23] "Clarity.LACE..Score"             "READ"
```

```
ranger_read3$variable.importance %>%
  tidy() %>%
  dplyr::arrange(desc(x)) %>%
  dplyr::top_n(25) %>%
  ggplot(aes(reorder(names, x), x)) +
  geom_col() +
  coord_flip() +
  ggtitle("Top important variables")
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")

## Selecting by x
```



```r
library(kernlab,warn.conflicts = F)      # SVM methodology
library(e1071)         # SVM methodology
library(ISLR)          # contains example data set "Khan"
# install.packages("ISLR")
# install.packages("RColorBrewer")
library(RColorBrewer) # customized coloring of plots

# # sample(1:nrow(new_train),2000)
# # new_train[sample(new_train,,replace = F),c(1,2,3,4,5,6,8,9,11,14,2
4)]
# #svm
# set.seed(16)
# svmfit <- svm(READ~., data = new_train[sample(1:nrow(new_train),3000
0),c(1,2,3,4,5,6,8,9,11,14,24)], kernel = "linear", scale = FALSE)
# svmpre <- predict(svmfit,new_test)
# svmpre %>% range()
# roccurve_svm <- roc(READ~svmpre,data = new_test)
# plot(roccurve_svm )
# coords(roc=roccurve_svm ,"best","threshold")
# classifer_svm <- as.numeric(coords(roccurve_svm,"best","threshold"))
# class_svm <- ifelse(svmpre>classifer_svm[1],1,0)
# table(class_svm)
# accuracy_svm <- 1-mean(class_svm!=new_test$READ)
# cat("Accuracy is for SVM ", accuracy_svm,"\n")
# cat("The AUC score is ",roccurve_svm$auc,"\n")
# ```
#
# ```{r}
#
# set.seed(16)
# svmfit4 <- svm(READ~., data = new_train[sample(1:nrow(new_train)),c(1,
2,3,4,5,6,8,9,11,14,24)], kernel = "radial", gamma = 1, cost = 1)
# svmpre <- predict(svmfit4,new_test)
# svmpre %>% range()
# roccurve_svm <- roc(READ~svmpre,data = new_test)
# plot(roccurve_svm,xlim=c(0,1),col="blue",asp = NA)
# coords(roc=roccurve_svm ,"best","threshold")
# classifer_svm <- as.numeric(coords(roccurve_svm,"best","threshold"))
# class_svm <- ifelse(svmpre>classifer_svm[1],1,0)
# table(class_svm)
# # table(new_test$READ)
# accuracy_svm <- 1-mean(class_svm!=new_test$READ)
# cat("The Accuracy is for SVM ", accuracy_svm,"\n")
# cat("The AUC score is ",roccurve_svm$auc,"\n")
#
# misclass <- table(predict = class_svm, truth = new_test$READ)

library(rsample)     # data splitting
library(dplyr)       # data wrangling
library(rpart)       # performing regression trees
library(rpart.plot)  # plotting regression trees
# install.packages("rpart.plot")  # plotting regression trees
library(ipred)       # bagging
library(caret)       # bagging

# bagging decision trees.

# set.seed(16)
# bagged_m1 <- bagging(
#   formula = READ~.,
#   data    = new_train[sample(1:nrow(new_train)),c(1,2,3,4,5,6,8,9,11,
14,24)],
#   coob    = TRUE,
#   nbagg   = 31
# )
# bagpre <- predict(bagged_m1,new_test)
# bagpre %>% range()
# roccurve_bag <- roc(READ~bagpre,data = new_test)
# plot(roccurve_bag,xlim=c(0,1),col="blue",asp = NA)
# coords(roc=roccurve_bag ,"best","threshold")
# classifer_bag <- as.numeric(coords(roccurve_bag,"best","threshold"))
# class_bag <- ifelse(bagpre>classifer_bag[1],1,0)
# table(class_bag)
```

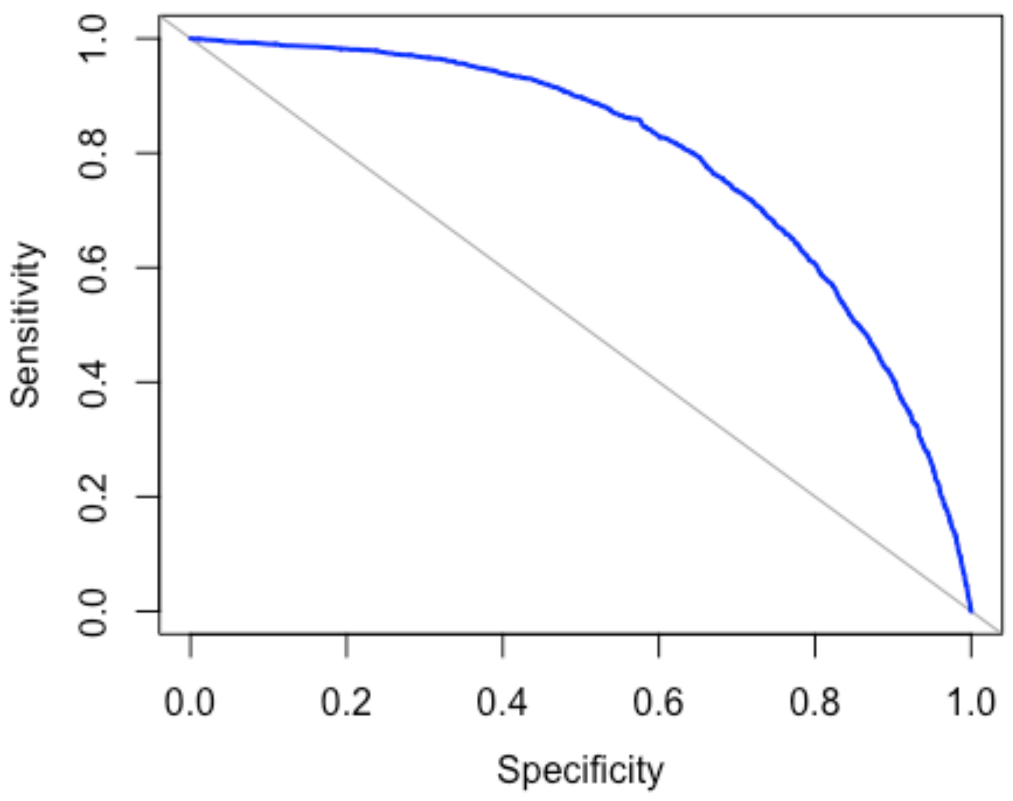
```
# table(new_test$READ)
# accuracy_bag <- 1-mean(class_bag!=new_test$READ)
# cat("Accuracy is for Bootstrap aggregating (bagging) ", accuracy_bag,
"\n")
# cat("The AUC score is ",roccurve_bag$auc,"\n")
# plot(varImp(bagged_m1), 20)


# set.seed(16)
# ctrl <- trainControl(method = "cv",  number = 3)
# # bagged_cv <- train(
# #   factor(READ) ~ .,
# #   data = new_train[sample(1:nrow(new_train)),c(1,2,3,4,5,6,8,9,11,1
4,24)],
# #   method = "treebag",
# #   trControl = ctrl,
# #   importance = TRUE
# #   )
# set.seed(16)
# bagged_cv2 <- train(
#   factor(READ) ~ .,
#   data = new_train[sample(1:nrow(new_train),2000),c(3:11,24)],
#   method = "treebag",
#   trControl = ctrl,
#   importance = TRUE
#   )
# bagpre <- predict(bagged_cv,new_test,type="prob")
# bagpre
# table(as.integer(bagpre)-1)
# l=(as.integer(bagpre)-1)
# roccurve_bag <- roc(READ~ bagpre[,2],data = new_test)
# plot(roccurve_bag,xlim=c(0,1),col="blue",asp = NA)
# auc(roccurve_bag)
# coords(roc=roccurve_bag ,"best","threshold")
# classifer_bag <- as.numeric(coords(roccurve_bag,"best","threshold"))
# class_bag <- ifelse(bagpre>classifer_bag[1],1,0)
# table(class_bag)
# table(new_test$READ)
# accuracy_bag <- 1-mean(bagpre!=new_test$READ)
# cat("Accuracy is for Bootstrap aggregating (bagging) ", accuracy_bag,
"\n")
# cat("The AUC score is ",roccurve_bag$auc,"\n")
# plot(varImp(bagged_m1), 20)
#
#
#
# plot(varImp(bagged_cv2),10)
#
# test.bagprob=bagpre
#  library(ipred)
# #prepare bagged model for curve
# test.bagprob = predict(train_bag, type = “prob”, newdata = test)
# bagpred = prediction( bagpre[,2], test$class)
#  bagperf = performance(bagpre,"tpr","fpr")
#  bagprek <- ifelse(bagpre[,2]>bagpre[,1],bagpre[,2],bagpre[,1])
# > plot(perf, main=”ROC”, colorize=T)
# > plot(bagperf, col=2, add=TRUE)
# > plot(perf, col=1, add=TRUE)
# > legend(0.6, 0.6, c(‘ctree’, ‘bagging’), 1:2)
# ```
# ```{r}
# # # assess 10-50 bagged trees
# # ntree <- 25:40
# #
# # # create empty vector to store OOB RMSE values
# # rmse <- vector(mode = "numeric", length = length(ntree))
# #
# # for (i in seq_along(ntree)) {
# #   # reproducibility
# #   set.seed(123)
# #
# #   # perform bagged model
# #   model <- bagging(
# #   formula = READ~.,
# #   data    = new_train[sample(1:nrow(new_train)),c(1,2,3,4,5,6,8,9,1
1,14,24)],
# #   coob    = TRUE,
# #   nbagg   = ntree[i]
# # )
# #   # get OOB error
# #   rmse[i] <- model$err
# # }
# # plot(ntree, rmse, type = 'l', lwd = 2)
# # abline(v = 25, col = "red", lty = "dashed")
# # #

# 1,2,3,4,5,6,7,8,9,10,11,24

###numeric only
features_train <- as.matrix(new_train[,c(3:11)])
response_train <- as.matrix(new_train[,24])
# names(new)

features_test <- as.matrix(new_test[,c(3:11)])
response_test <- as.matrix(new_test[,24])

library("xgboost")

##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
# GBM xgboost: Training and tuning with the xgboost package
params <- list(
  eta = 0.1,
  max_depth =  9,
  min_child_weight = 3,
  subsample = 1,
  colsample_bytree =0.8

)
set.seed(16)
xgb.fit.final <- xgboost(
  params = params,
  data = features_train,
  label = response_train,
  nrounds = 46,
  objective = "reg:linear",
  verbose = 0
)

gbmpre <- predict(xgb.fit.final,features_test)
gbmpre %>% range()
```

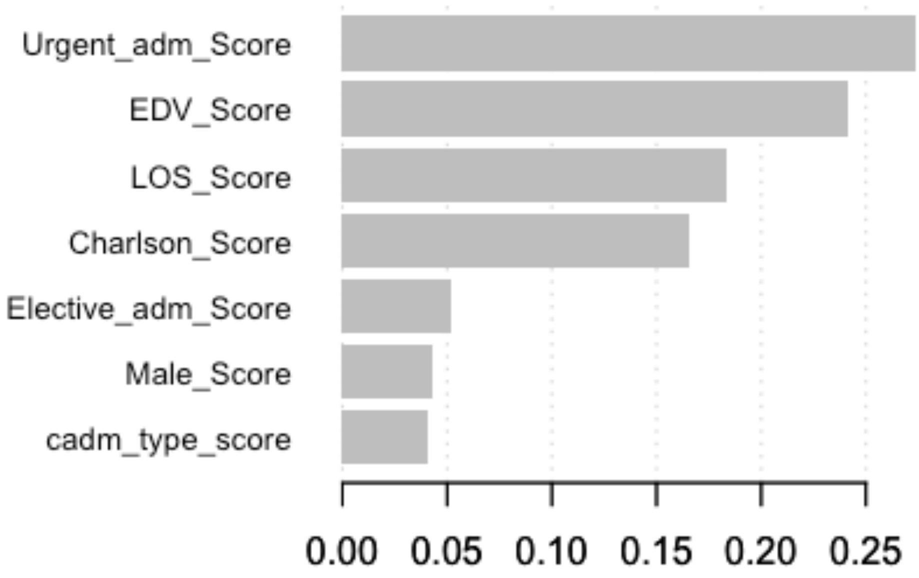
```
## [1] -0.03919309  0.90631318
```

```r
roccurve_gbm <- roc(new_test$READ~gbmpre)
plot(roccurve_gbm,xlim=c(0,1),col="blue",asp = NA)
```



```
coords(roc=roccurve_gbm ,"best","threshold")

##   threshold specificity sensitivity
##  0.09373079  0.65278181  0.79262926

classifer_gbm <- as.numeric(coords(roccurve_gbm,"best","threshold"))
class_gbm <- ifelse(gbmpre>classifer_gbm[1],1,0)
table(class_gbm)

## class_gbm
##     0     1
## 11629  7426

# table(response_test)s
accuracy_bgm <- 1-mean(class_gbm!=new_test$READ)
cat("Accuracy is for Gradient Boosting ", accuracy_bgm,"\n")

## Accuracy is for Gradient Boosting  0.6661244

cat("The AUC score is ",roccurve_gbm$auc,"\n")

## The AUC score is  0.7907732

# create importance matrix
importance_matrix <- xgb.importance(model = xgb.fit.final)
```

```r
# variable importance plot
xgb.plot.importance(importance_matrix, top_n = 13, measure = "Gain")
```



```r
# # cat("The Accuracy is for simple logistic ", accuracy,"\n")
# # cat("The AUC score is ",roccurve$auc,"\n")
# cat("The Accuracy is for multivariate logistic ", accuracy_multi,"\n")
# cat("The AUC score is ",roccurve_multi$auc,"\n")
# cat("The Accuracy is for random forest ", accuracy_rf,"\n")
# cat("The AUC score is ",roccurve_rf$auc,"\n")
# # cat("The Accuracy is for SVM ", accuracy_svm,"\n")
# # cat("The AUC score is ",roccurve_svm$auc,"\n")
# cat("The Accuracy is for Bagging ", accuracy_bag,"\n")
# cat("The AUC score is ",roccurve_bag$auc,"\n")
# cat("The Accuracy is for Gradient Boosting ", accuracy_bgm,"\n")
# cat("The AUC score is ",roccurve_gbm$auc,"\n")
# # plot(ranger_read$variable.importance)
```

Balance data

```r
library(DMwR,warn.conflicts = F)

## Loading required package: grid
```

```
new_train$READ <- as.factor(new_train$READ)
new_train <- SMOTE(READ~.,new_train[,c(1,2,3,4,5,6,8,9,11,14,24)],perc.
over = 100,perc.under = 200)
new_train$READ <- as.numeric(new_train$READ)

head(new_train,2)
## 76219                   6               30          0       70    1
## 31487.1                 0               18          0        2    1
##  [ reached 'max' / getOption("max.print") -- omitted 12962 rows ]

# library(xgboost)
#
# # grid search
# for(i in 1:nrow(hyper_grid)) {
#
#   # create parameter list
#   params <- list(
#     eta = hyper_grid$eta[i],
#     max_depth = hyper_grid$max_depth[i],
#     min_child_weight = hyper_grid$min_child_weight[i],
#     subsample = hyper_grid$subsample[i],
#     colsample_bytree = hyper_grid$colsample_bytree[i]
#   )
#
#   # reproducibility
#   set.seed(123)
#
#   # train model
#   xgb.tune <- xgb.cv(
#     params = params,
#     data = features_train,
#     label = response_train,
#     nrounds = 5000,
#     nfold = 5,
#     objective = "reg:linear",  # for regression models
#     verbose = 0,               # silent,
#     early_stopping_rounds = 10 # stop if no improvement for 10 consec
utive trees
#   )
#
#   # add min training error and trees to grid
#   hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$te
st_rmse_mean)
#   hyper_grid$min_RMSE[i] <- min(xgb.tune$evaluation_log$test_rmse_mea
n)
# }
#
# hyper_grid %>%
#   dplyr::arrange(min_RMSE) %>%
#   head(10)
```

```r
# create hyperparameter grid
hyper_grid <- expand.grid(
  eta = 0.1,
  max_depth =  7,
  min_child_weight = 3,
  subsample = 1,
  colsample_bytree =0.8,
  optimal_trees = 0,               # a place to dump results
  min_RMSE = 0                     # a place to dump results
)

# hyper_grid <- expand.grid(
#   eta = c(.01, .05, .1, .3),
#   max_depth = c(1, 3, 5, 7),
#   min_child_weight = c(1, 3, 5, 7),
#   subsample = c(.65, .8, 1),
#   colsample_bytree = c(.8, .9, 1),
#   optimal_trees = 0,               # a place to dump results
#   min_RMSE = 0                     # a place to dump results
# )

# grid search
# for(i in 1:nrow(hyper_grid)) {
#
#   # create parameter list
#   params <- list(
#     eta = hyper_grid$eta[i],
#     max_depth = hyper_grid$max_depth[i],
#     min_child_weight = hyper_grid$min_child_weight[i],
#     subsample = hyper_grid$subsample[i],
#     colsample_bytree = hyper_grid$colsample_bytree[i]
#   )
#
#   # reproducibility
#   set.seed(123)
#
#   # train model
#   xgb.tune <- xgb.cv(
#     params = params,
#     data = features_train,
#     label = response_train,
#     nrounds = 5000,
#     nfold = 5,
#     objective = "reg:linear",  # for regression models
#     verbose = 0,               # silent,
#     early_stopping_rounds = 10 # stop if no improvement for 10 consec
utive trees
#   )
#
#   # add min training error and trees to grid
#   hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$te
st_rmse_mean)
#   hyper_grid$min_RMSE[i] <- min(xgb.tune$evaluation_log$test_rmse_mea
n)
# }
#
# hyper_grid %>%
#   dplyr::arrange(min_RMSE) %>%
#   head(10)
#46 54 55
```