DASC_5420_Final_Project

Changda Li (T00705321)

2023-04-10

Reading data

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v ggplot2 3.4.0     v purrr   0.3.5
## v tibble  3.1.8     v stringr 1.5.0
## v tidyr   1.2.1     v forcats 0.5.2
## v readr   2.1.3
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(pheatmap)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
library(ROCR)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(plotROC)
```

```
##
## Attaching package: 'plotROC'
##
## The following object is masked from 'package:pROC':
##
##     ggroc
```

```
set.seed(123)
setwd("/Users/changdali/Desktop/data science related/DASC 5420/Final_project")
heart_data <- read.csv("heart_2020_cleaned.csv")
head(heart_data)
```

```
##   HeartDisease   BMI Smoking AlcoholDrinking Stroke PhysicalHealth MentalHealth
## 1           No 16.60     Yes              No     No              3           30
## 2           No 20.34      No              No    Yes              0            0
## 3           No 26.58     Yes              No     No             20           30
## 4           No 24.21      No              No     No              0            0
## 5           No 23.71      No              No     No             28            0
## 6          Yes 28.87     Yes              No     No              6            0
##   DiffWalking    Sex AgeCategory  Race Diabetic PhysicalActivity GenHealth
## 1          No Female       55-59 White      Yes              Yes Very good
## 2          No Female 80 or older White       No              Yes Very good
```

```
## 3          No   Male       65-69 White      Yes              Yes      Fair
## 4          No Female       75-79 White       No               No      Good
## 5         Yes Female       40-44 White       No              Yes Very good
## 6         Yes Female       75-79 Black       No               No      Fair
##   SleepTime Asthma KidneyDisease SkinCancer
## 1         5    Yes            No        Yes
## 2         7     No            No         No
## 3         8    Yes            No         No
## 4         6     No            No        Yes
## 5         8     No            No         No
## 6        12     No            No         No
```

```
colnames(heart_data)
```

```
##  [1] "HeartDisease"     "BMI"              "Smoking"          "AlcoholDrinking"
##  [5] "Stroke"           "PhysicalHealth"   "MentalHealth"     "DiffWalking"
##  [9] "Sex"              "AgeCategory"      "Race"             "Diabetic"
## [13] "PhysicalActivity" "GenHealth"        "SleepTime"        "Asthma"
## [17] "KidneyDisease"    "SkinCancer"
```

```
#heart_data <- sample_n(heart_data, 5*10^4)
```

Data Preprocessing

```
set.seed(123)
# Identify categorical and continuous variables
cat_var <- c("Smoking",
             "AlcoholDrinking",
             "Stroke",
             "DiffWalking",
             "Sex",
             "AgeCategory",
             "Race",
             "Diabetic",
             "PhysicalActivity",
             "GenHealth",
             "Asthma",
             "KidneyDisease",
             "SkinCancer")
con_var <- c("BMI",
             "PhysicalHealth",
             "MentalHealth",
             "SleepTime")

# Factor the categorical variables
for (name in cat_var){
  heart_data[[name]] <- factor(heart_data[[name]])
}

# Scale the continuous data
```

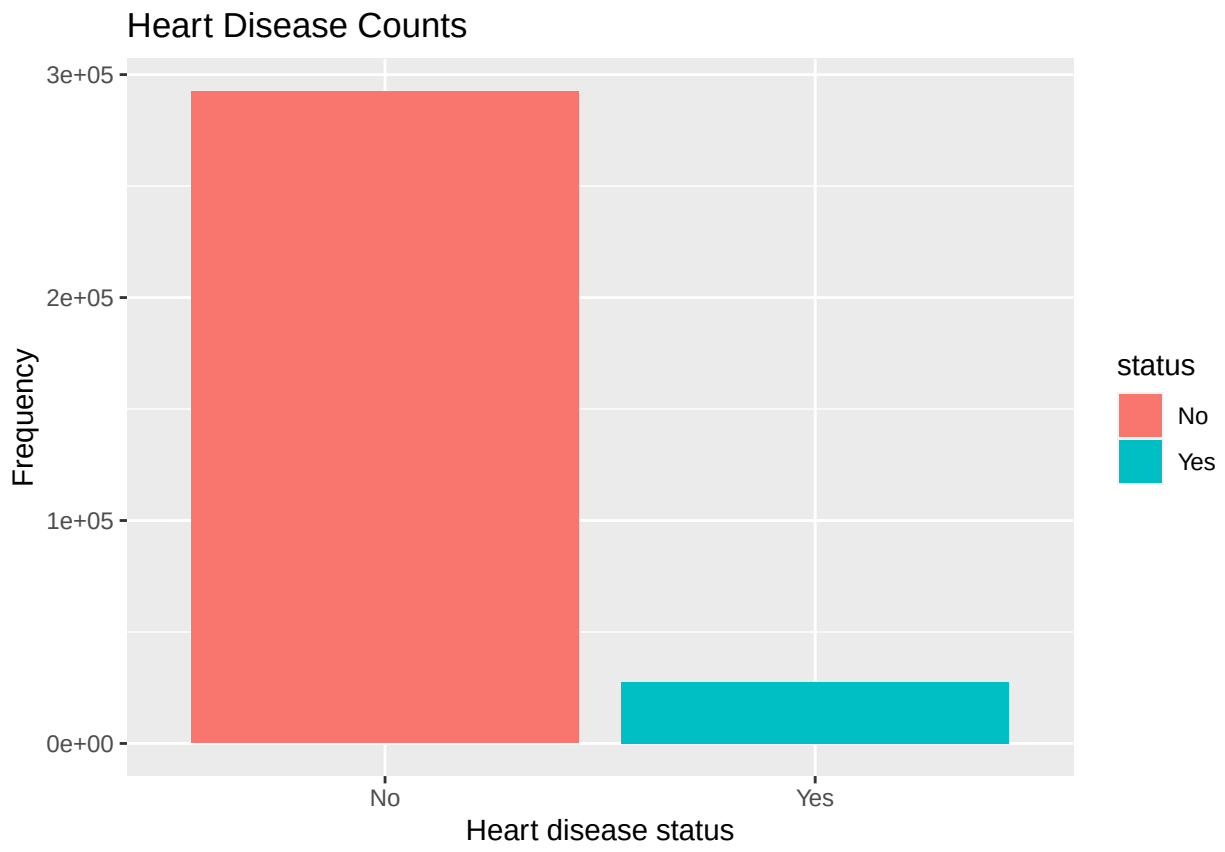
```
heart_data[con_var] <- scale(heart_data[con_var])

heart_data$HeartDisease <- factor(heart_data$HeartDisease)
```

EDA

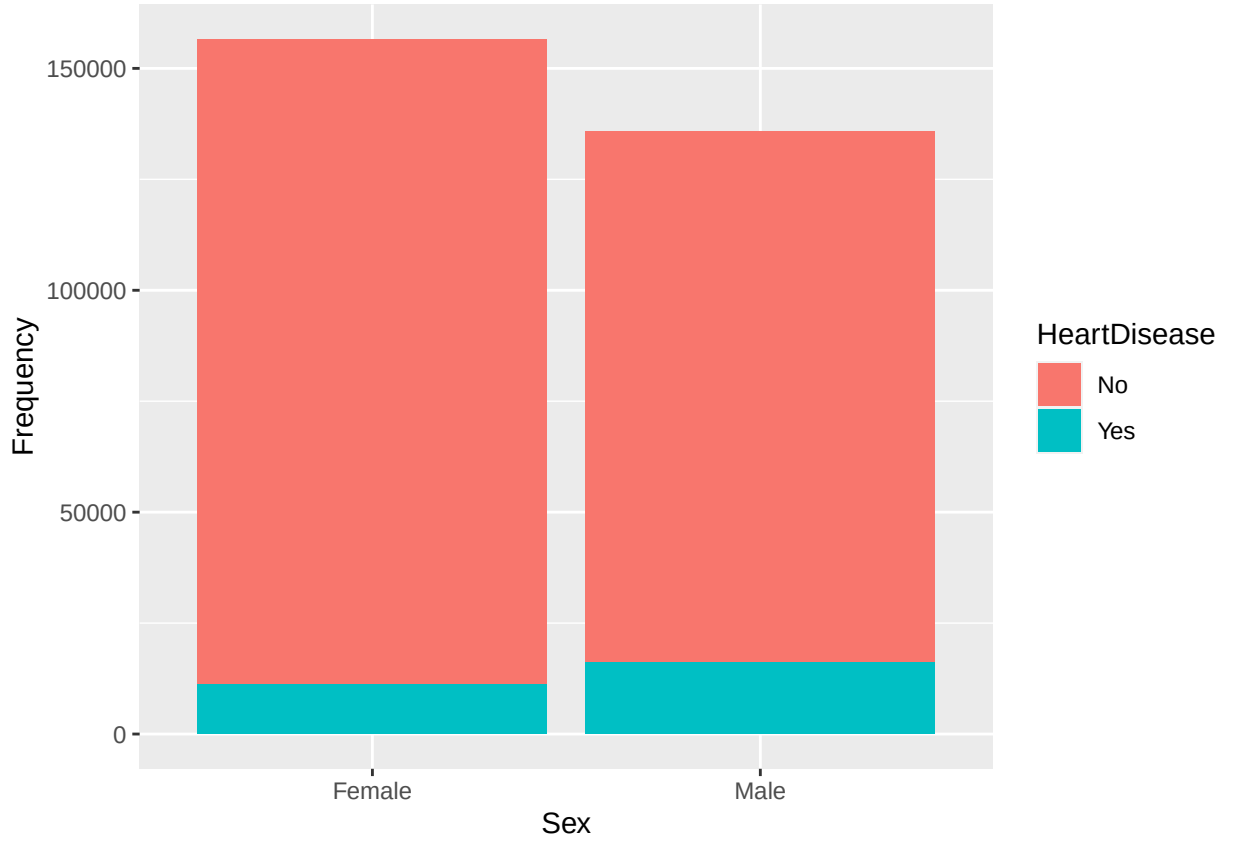
```
set.seed(123)
library(ggplot2)
library(tidyr)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# plot bar chart of the target variable
target_data <- data.frame(status = heart_data$HeartDisease)
ggplot(target_data, aes(x = status, fill = status)) +
  geom_bar() +
  labs(title = "Heart Disease Counts",
       x = "Heart disease status",
       y = "Frequency")
```



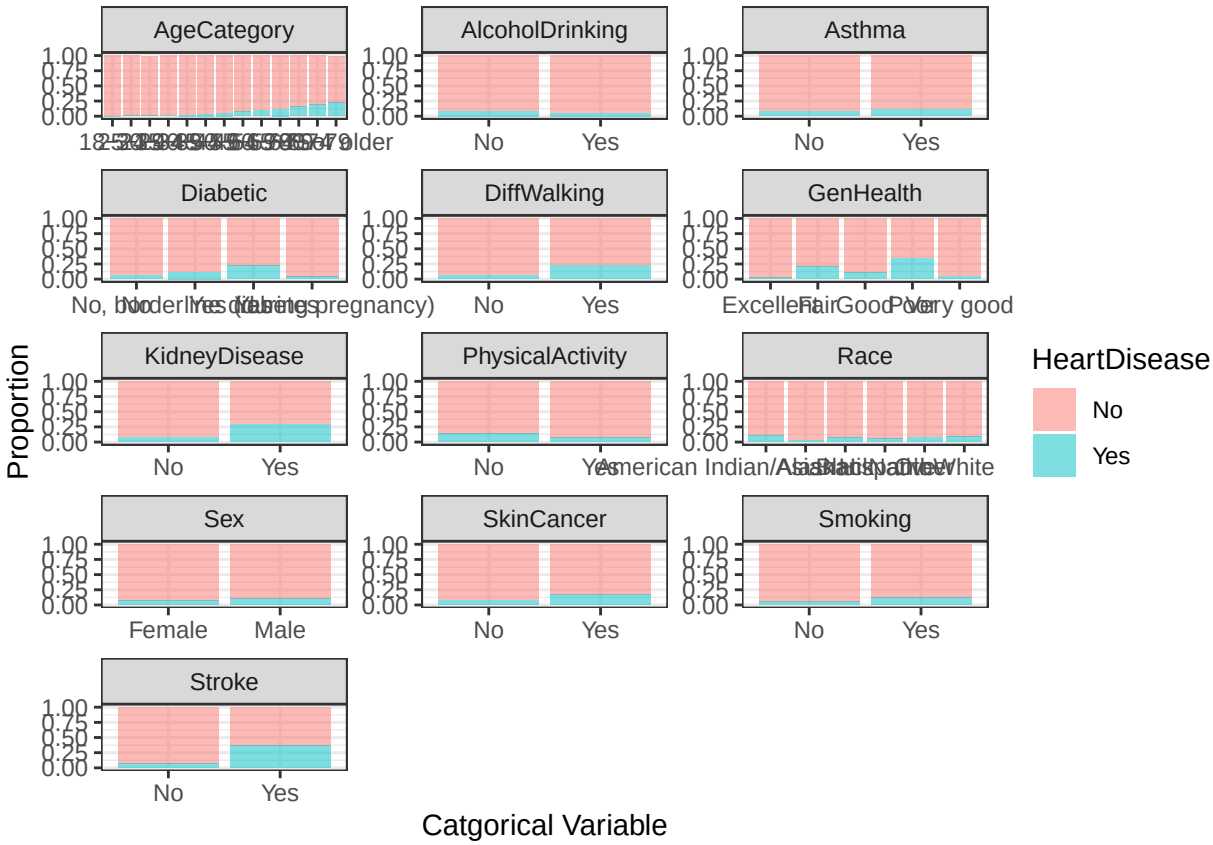
```r
# Create the plot
ggplot(heart_data, aes(x = Sex, fill = HeartDisease))+
  geom_bar(position = "identity") +
  labs(x = "Sex", y = "Frequency")
```




```r
cat_w_target <- data.frame(heart_data[,cat_var],
                           HeartDisease = heart_data[,1])
cat_w_target_long <- gather(cat_w_target, key = "variable", value = "value", -HeartDisease)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```r
ggplot(cat_w_target_long, aes(x = value, fill = HeartDisease)) +
  geom_bar(position = "fill", alpha = 0.5) +
  facet_wrap(~variable, scales = "free", ncol = 3) +
  labs(x = "Catgorical Variable", y = "Proportion") +
  theme_bw()
```



Train test split

```
set.seed(123)
library(caret)
train_ind <- createDataPartition(heart_data$HeartDisease,
                                 p = 0.7,
                                 list = FALSE)
train <- heart_data[train_ind,]
test <- heart_data[-train_ind,]
```

Logistic regression

Train logistic model and evaluate

```
set.seed(123)
ctrl <- trainControl(method = "cv", number = 10)
log_model <- train(HeartDisease~.,
                   data = train,
                   method = "glm",
                   trControl = ctrl,
                   family = binomial)

vali_accuracy <- log_model$results$Accuracy
cat("The validation accuracy is", vali_accuracy, "\n")
```

```
## The validation accuracy is 0.9159155
```

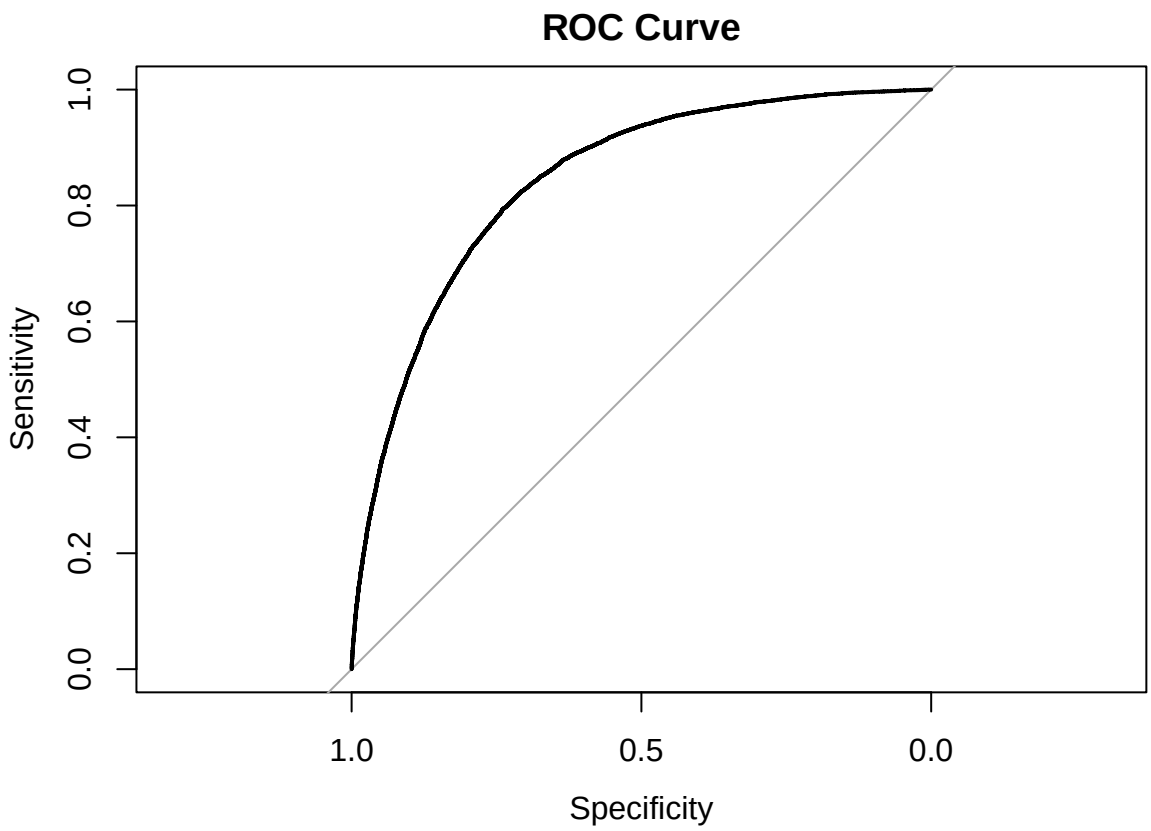
```
log_pred <- predict(log_model, newdata = test)
log_pred_auc <- predict(log_model, newdata = test, type = "prob")

roc_obj <- roc(response = test$HeartDisease,
               predictor = log_pred_auc$Yes,
               levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```
cat("The AUC is", auc(roc_obj), "\n")
```

```
## The AUC is 0.8408975
```

```
plot(roc_obj, main = "ROC Curve")
```




```
log_cm_no <- confusionMatrix(log_pred,
                             test$HeartDisease,
                             mode = "everything",
                             positive="No")
log_cm_yes <- confusionMatrix(log_pred,
                              test$HeartDisease,
```

```
                               mode = "everything",
                               positive="Yes")
cat("The test accuracy is", log_cm_no$overall["Accuracy"], "\n")
```

```
## The test accuracy is 0.9159865
```

```
cat("The precision, recall and F1 of 'No' class are \n",
    log_cm_no$byClass["Precision"],"\n",
    log_cm_no$byClass["Recall"],"\n",
    log_cm_no$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'No' class are
##  0.9221297
##  0.9918838
##  0.9557357
```

```
cat("The precision, recall and F1 of 'Yes' class are \n",
    log_cm_yes$byClass["Precision"],"\n",
    log_cm_yes$byClass["Recall"],"\n",
    log_cm_yes$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'Yes' class are
##  0.5479365
##  0.1051029
##  0.1763744
```

Even though the overall accuracy of the model is high, the values of precision, recall and F1 score are relatively low. This is due to the class imbalance of the data set.

Balance the data

under sample the data:

```
library(ROSE)
set.seed(123)
num_of_yes <- sum(heart_data$HeartDisease == "Yes")
new_frac <- 0.5
new_n <- num_of_yes / new_frac
balance_heart <- ovun.sample(formula = HeartDisease ~.,
                             data = heart_data,
                             method = "under",
                             N = new_n,
                             seed = 5420)
balance_heart <- balance_heart$data
```

Train test split for balance data

```r
set.seed(123)
train_ind_b <- createDataPartition(balance_heart$HeartDisease,
                                   p = 0.7,
                                   list = FALSE)
train_b <- balance_heart[train_ind_b,]
test_b <- balance_heart[-train_ind_b,]
```

Train logistic model on balance data

```r
set.seed(123)
# Using 10 folds cross validation
ctrl <- trainControl(method = "cv", number = 10)
log_model_b <- train(HeartDisease~.,
                     data = train_b,
                     method = "glm",
                     trControl = ctrl,
                     family = binomial)
vali_accuracy_b <- log_model_b$results$Accuracy
cat("The validation accuracy is", vali_accuracy_b, "\n")
```

```
## The validation accuracy is 0.7671436
```

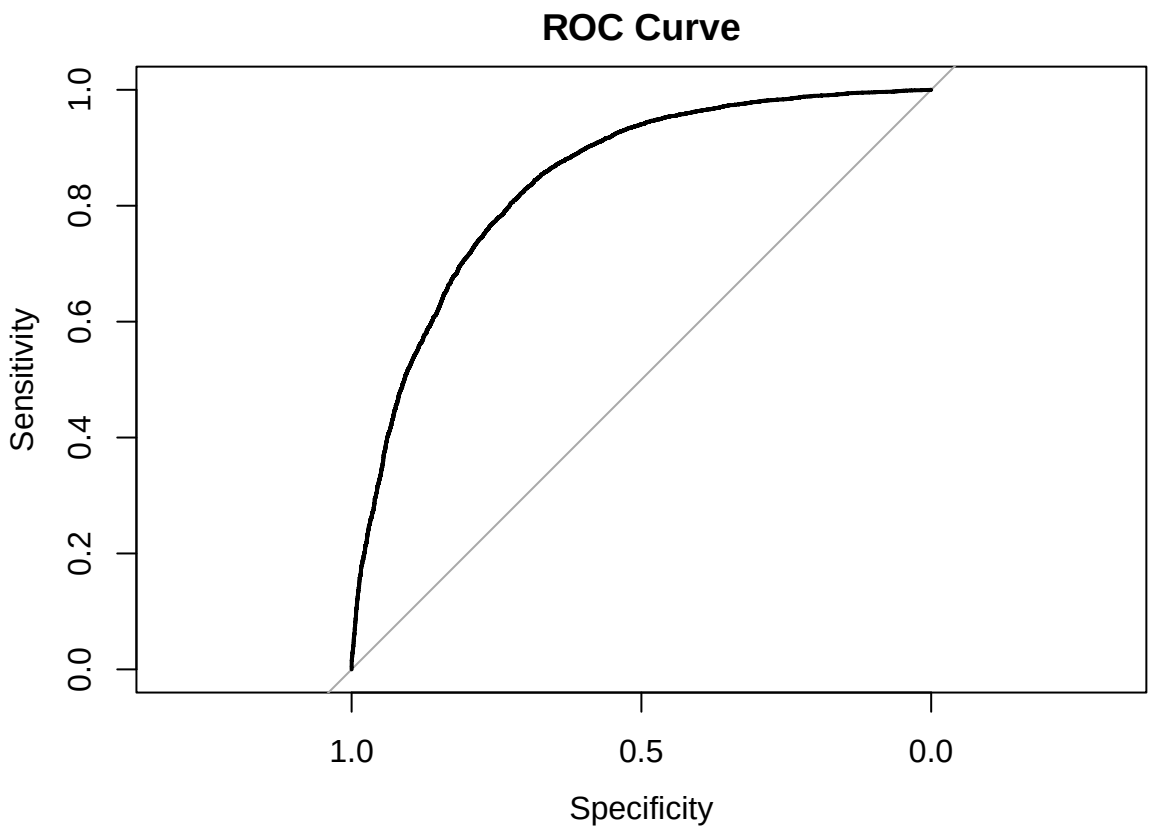
```r
log_pred_b <- predict(log_model_b, newdata = test_b)
log_pred_b_auc <- predict(log_model_b,
                          newdata = test_b,
                          type = "prob")

roc_obj_b <- roc(response = test_b$HeartDisease,
                 predictor = log_pred_b_auc$Yes,
                 levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```r
cat("The AUC is", auc(roc_obj_b), "\n")
```

```
## The AUC is 0.8409228
```

```r
plot(roc_obj_b, main = "ROC Curve")
```




```
log_cm_no_b <- confusionMatrix(log_pred_b,
                               test_b$HeartDisease,
                               mode = "everything",
                               positive="No")
log_cm_yes_b <- confusionMatrix(log_pred_b,
                                test_b$HeartDisease,
                                mode = "everything",
                                positive="Yes")
cat("The test accuracy is", log_cm_no_b$overall["Accuracy"], "\n")
```

```
## The test accuracy is 0.7631226
```

```
cat("The precision, recall and F1 of 'No' class for balanced data set are \n",
    log_cm_no_b$byClass["Precision"],"\n",
    log_cm_no_b$byClass["Recall"],"\n",
    log_cm_no_b$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'No' class for balanced data set are
##  0.770977
##  0.7486299
##  0.7596391
```

```
cat("The precision, recall and F1 of 'Yes' class for balanced data set are \n",
    log_cm_yes_b$byClass["Precision"],"\n",
```

```
    log_cm_yes_b$byClass["Recall"],"\n",
    log_cm_yes_b$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'Yes' class for balanced data set are
##  0.7557107
##  0.7776154
##  0.7665066
```

Lasso regression imbalance

Train test split

```
set.seed(123)
# Transfer the factor variables of character to numbers
formula <- formula(paste("~", paste(cat_var, collapse = " + ")))
fac_col <- model.matrix(formula, heart_data)
heart_lasso <- data.frame(HeartDisease =
                            factor(heart_data$HeartDisease,
                                   levels = c("Yes","No"),
                                   labels = c(1,0)),
                          heart_data[, con_var],
                          fac_col[, 2:ncol(fac_col)])
# Train test split
train_ind_l <- createDataPartition(heart_lasso$HeartDisease,
                                   p = 0.7,
                                   list = FALSE)
train_l <- heart_lasso[train_ind_l,]
test_l <- heart_lasso[-train_ind_l,]
train_x_l <- train_l[,-1]
train_y_l <- train_l[,1]

test_x_l <- test_l[,-1]
test_y_l <- test_l[,1]
```

Find the optimal lambda using cv

```
set.seed(123)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-6
```

```
# train the model with training set using cross validation
cv_lasso <- cv.glmnet(as.matrix(train_x_l),
                      train_y_l,
                      family = "binomial",
                      alpha=1,
                      type.measure="class",
                      nfolds=10)

optimal_lambda <- cv_lasso$lambda.min
```

Evaluation lasso imbalance

```
set.seed(123)
# The validation error for the optimal lambda
cvm_min <- cv_lasso$cvm[which.min(cv_lasso$cvm)]
vali_acc_la <- 1 - cvm_min
cat("The validation accuracy is", vali_acc_la , "\n")
```

```
## The validation accuracy is 0.9159512
```

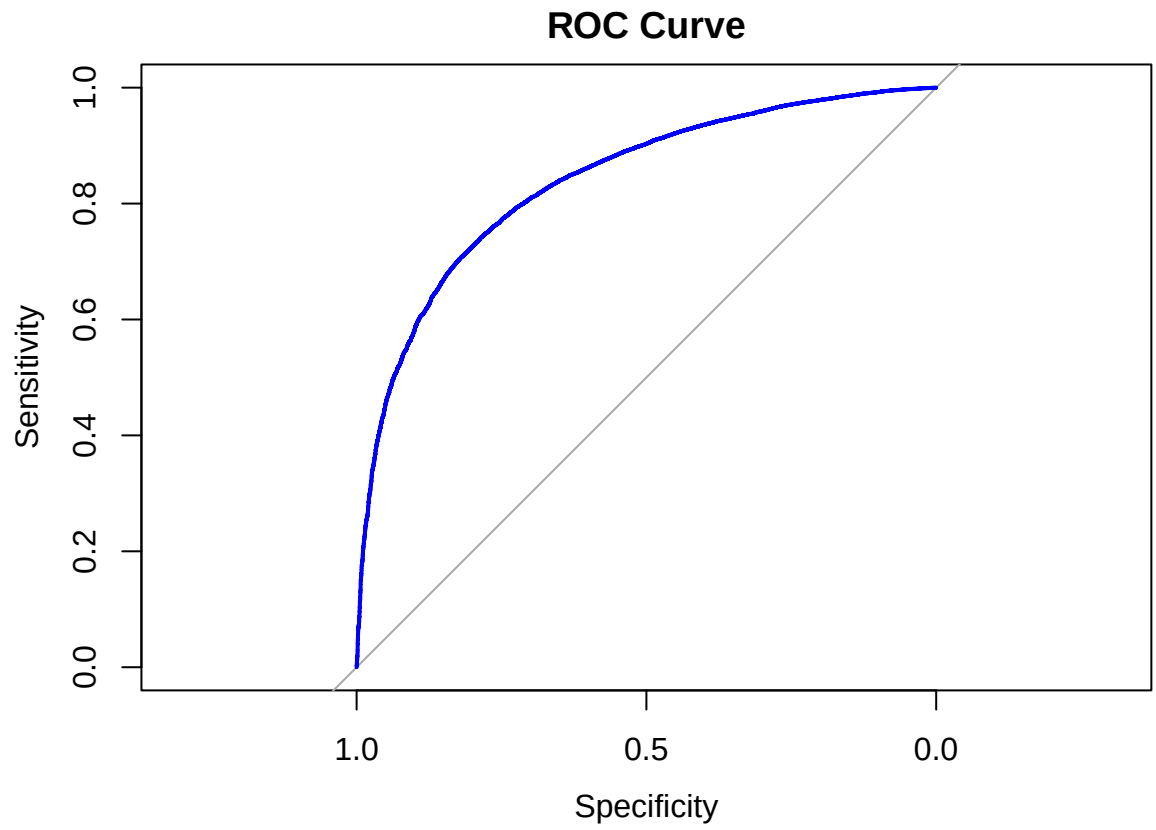
```
lasso_model <- glmnet(as.matrix(train_x_l),
                      train_y_l,
                      family = "binomial",
                      alpha = 1,
                      lambda = optimal_lambda)

lasso_pred <- predict(lasso_model,
                      newx = as.matrix(test_x_l),
                      type = "class")
lasso_pred_auc <- predict(lasso_model,
                          newx = as.matrix(test_x_l),
                          type = "response")

roc_l <- roc(test_y_l, lasso_pred_auc)
```

```
## Setting levels: control = 1, case = 0

## Warning in roc.default(test_y_l, lasso_pred_auc): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.

## Setting direction: controls < cases
```

```
cat("The AUC is", auc(roc_l), "\n")
```

```
## The AUC is 0.8385847
```

```
plot(roc_l, main = "ROC Curve",
     legacy.axes = FALSE,
     col = "blue")
```




```
lasso_cm_no <- confusionMatrix(factor(lasso_pred,
                                      levels = c(1,0)),
                               test_y_l,
                               mode = "everything",
                               positive = "0")
lasso_cm_yes <- confusionMatrix(factor(lasso_pred,
                                      levels = c(1,0)),
                                test_y_l,
                                mode = "everything",
                                positive = "1")

cat("The test accuracy is", lasso_cm_yes$overall["Accuracy"] , "\n")
```

```
## The test accuracy is 0.9163514
```

```
cat("The precision, recall and F1 of 'No' class are \n",
    lasso_cm_no$byClass["Precision"],"\n",
    lasso_cm_no$byClass["Recall"],"\n",
   lasso_cm_no$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'No' class are
##  0.9213105
##  0.9933657
##  0.9559823
```

```
cat("The precision, recall and F1 of 'Yes' class are \n",
    lasso_cm_yes$byClass["Precision"],"\n",
    lasso_cm_yes$byClass["Recall"],"\n",
    lasso_cm_yes$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'Yes' class are
##  0.5688889
##  0.09353307
##  0.1606527
```

Balance the data undersample

```
set.seed(123)
num_of_yes <- sum(heart_data$HeartDisease == "Yes")
new_frac <- 0.5
new_n <- num_of_yes / new_frac
balance_heart_lasso <- ovun.sample(formula = HeartDisease ~.,
                                   data = heart_lasso,
                                   method = "under",
                                   N = new_n,
                                   seed = 5420)
balance_heart_lasso <- balance_heart_lasso$data
```

Redo lasso for the balanced data

```
set.seed(123)
# Train test split
train_ind_l_b <- createDataPartition(balance_heart_lasso$HeartDisease,
                                     p = 0.7,
                                     list = FALSE)

train_l_b <- balance_heart_lasso[train_ind_l_b,]
test_l_b <- balance_heart_lasso[-train_ind_l_b,]

train_x_l_b <- train_l_b[,-1]
train_y_l_b <- train_l_b[,1]

test_x_l_b <- test_l_b[,-1]
test_y_l_b <- test_l_b[,1]
```

Fit lasso on balanced data

```
set.seed(123)
# Find the optimal lambda
cv_lasso_b <- cv.glmnet(as.matrix(train_x_l_b),
                        train_y_l_b,
                        family = "binomial",
                        alpha=1,
                        type.measure="class",
                        nfolds=10)

optimal_lambda_b <- cv_lasso_b$lambda.min
```

Evaluation lasso balance

```
set.seed(123)
# The validation acc for the optimal lambda
cvm_min_b <- cv_lasso_b$cvm[which.min(cv_lasso_b$cvm)]
vali_acc_la_b <- 1 - cvm_min_b
cat("The validation accuracy is", vali_acc_la_b, "\n")
```

```
## The validation accuracy is 0.7668563
```

```
lasso_model_b <- glmnet(as.matrix(train_x_l_b),
                        train_y_l_b,
                        family = "binomial",
                        alpha = 1,
                        lambda = optimal_lambda_b)
lasso_pred_b <- predict(lasso_model_b,
                        newx = as.matrix(test_x_l_b),
                        type = "class")

lasso_pred_b_auc <- predict(lasso_model_b,
                            newx = as.matrix(test_x_l_b),
                            type = "response")

roc_l_b <- roc(test_y_l_b, lasso_pred_b_auc)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(test_y_l_b, lasso_pred_b_auc): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```

```
cat("The AUC is", auc(roc_l_b), "\n")
```

```
## The AUC is 0.8409352
```

```
lasso_cm_no_b <- confusionMatrix(factor(lasso_pred_b),
                                 mode = "everything",
                                 positive = "0")
lasso_cm_yes_b <- confusionMatrix(factor(lasso_pred_b),
                                  mode = "everything",
                                  positive = "1")

cat("The test accuracy is", lasso_cm_yes_b$overall["Accuracy"] , "\n")
```

```
## The test accuracy is 0.7630009
```

```
cat("The precision, recall and F1 of 'No' class are \n",
    lasso_cm_no_b$byClass["Precision"],"\n",
    lasso_cm_no_b$byClass["Recall"],"\n",
    lasso_cm_no_b$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'No' class are
##  0.7701739
##  0.749726
##  0.7598124
```

```
cat("The precision, recall and F1 of 'Yes' class are \n",
    lasso_cm_yes_b$byClass["Precision"],"\n",
    lasso_cm_yes_b$byClass["Recall"],"\n",
    lasso_cm_yes_b$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'Yes' class are
##  0.7561988
##  0.7762757
##  0.7661058
```

Decision tree

```
library(rpart)
folds <- createFolds(train$HeartDisease, k = 10)

# Train and evaluate the decision tree model with cross-validation
tree_model <- rpart(HeartDisease ~ ., data = train)
tree_cv <- train(HeartDisease ~ .,
                 data = train,
                 method = "rpart",
                 trControl =
                   trainControl(method = "cv", index =folds))
# Print
cat("The validation accuracy is",
    tree_cv$results$Accuracy[which.max(tree_cv$results$Accuracy)], "\n")
```

```
## The validation accuracy is 0.9145078
```

Evaluation decsion tree imbalance

```
set.seed(123)
# Train the model with the best cp
tree_model <- rpart(HeartDisease ~ .,
                    data=train,
                    cp=tree_cv$bestTune$cp)

d_pred <- predict(tree_model, newdata=test, type="class")
d_pred_auc <- predict(tree_model, newdata=test, type="prob")

roc_dt <- roc(response = test$HeartDisease,
              predictor = d_pred_auc[,2],
              levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```
cat("The AUC is", auc(roc_dt), "\n")
```

```
## The AUC is 0.6448598
```

```
d_cm_yes <- confusionMatrix(d_pred,
                            test$HeartDisease,
                            mode = "everything",
                            positive = "Yes")

d_cm_no <- confusionMatrix(d_pred,
                           test$HeartDisease,
                           mode = "everything",
                           positive = "No")

cat("The test accuracy is", d_cm_no$overall["Accuracy"] , "\n")
```

```
## The test accuracy is 0.9150484
```

```
cat("The precision, recall and F1 of 'No' class are \n",
    d_cm_no$byClass["Precision"],"\n",
    d_cm_no$byClass["Recall"],"\n",
   d_cm_no$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'No' class are
##  0.9172399
##  0.997059
##  0.9554854
```

```
cat("The precision, recall and F1 of 'Yes' class are \n",
    d_cm_yes$byClass["Precision"],"\n",
    d_cm_yes$byClass["Recall"],"\n",
    d_cm_yes$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'Yes' class are
##  0.5528596
##  0.03885032
##  0.072599
```

Balance the data for DT

```
set.seed(123)
num_of_yes <- sum(heart_data$HeartDisease == "Yes")
new_frac <- 0.5
new_n <- num_of_yes / new_frac
balance_heart_d <- ovun.sample(formula = HeartDisease ~.,
                               data = heart_data,
                               method = "under",
                               N = new_n,
                               seed = 5420)
balance_heart_d <- balance_heart_d$data
#Train test split
train_ind_d <- createDataPartition(balance_heart_d$HeartDisease,
                                   p = 0.7,
                                   list = FALSE)
train_d <- balance_heart_d[train_ind_d,]
test_d <- balance_heart_d[-train_ind_d,]
```

Redo DT for the balanced data

```
set.seed(123)
folds_d <- createFolds(train_d$HeartDisease, k = 10)
tree_cv_d <- train(HeartDisease ~ .,
                   data = train_d,
                   method = "rpart",
                   trControl =
                   trainControl(method = "cv", index =folds_d))
optimal_cp_d <- tree_cv_d$bestTune$cp

cat("The validation accuracy is",
    tree_cv_d$results$Accuracy[which.max(tree_cv_d$results$Accuracy)], "\n")
```

```
## The validation accuracy is 0.6807396
```

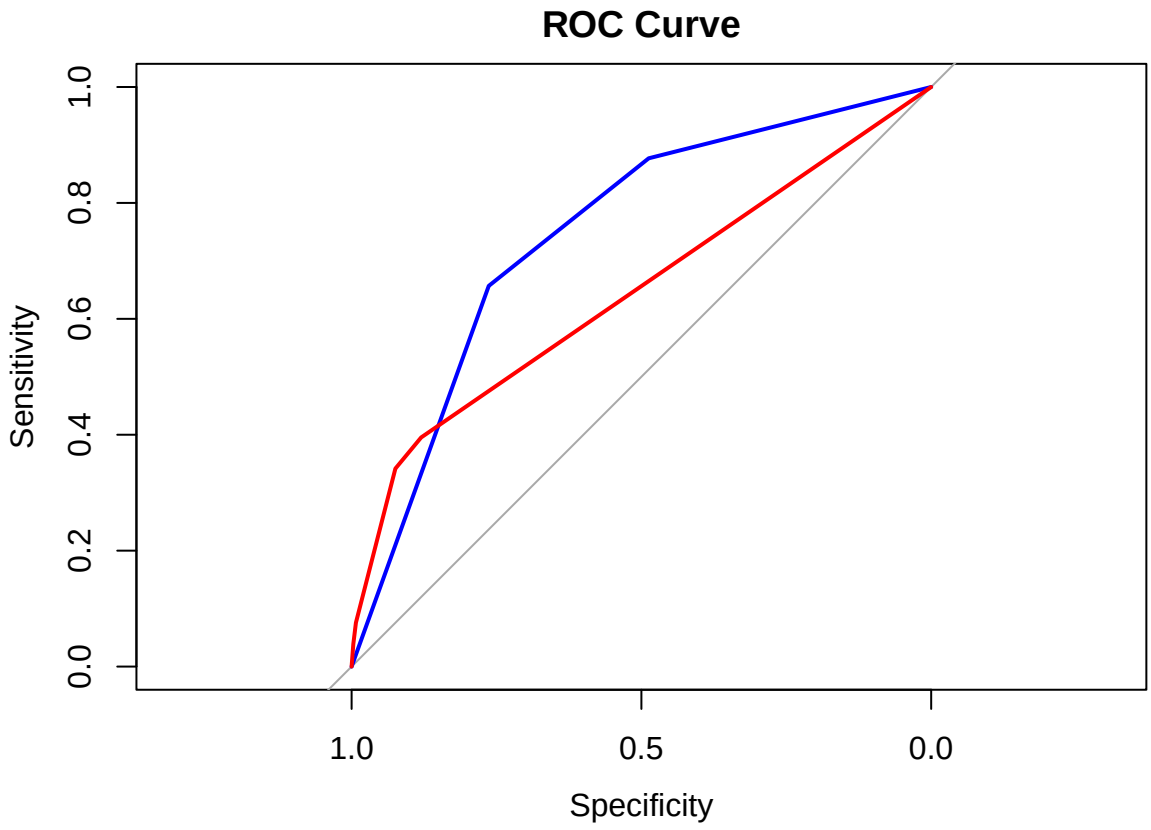
```
tree_model_d <- rpart(HeartDisease ~ .,
                      data=train_d,
                      cp=optimal_cp_d)
d_pred_b <- predict(tree_model_d, newdata = test_d, type = "class")
d_pred_b_auc <- predict(tree_model_d, newdata = test_d, type = "prob")

roc_dt_b <- roc(response = test_d$HeartDisease,
                predictor = d_pred_b_auc[,2],
                levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```
cat("The AUC is", auc(roc_dt_b), "\n")
```

```
## The AUC is 0.7467714
```

```
plot(roc_dt_b,
     main = "ROC Curve",
     col = "blue")
lines(roc_dt, col = "red")
```




```
cm_d_b_no <- confusionMatrix(d_pred_b,
                             test_d$HeartDisease,
                             mode = "everything",
                             positive = "No")
cm_d_b_yes <- confusionMatrix(d_pred_b,
                              test_d$HeartDisease,
                              mode = "everything",
                              positive = "Yes")

cat("The test accuracy is",
    cm_d_b_yes$overall["Accuracy"], "\n")
```

```
## The test accuracy is 0.710084
```

```
cat("The precision, recall and F1 of 'No' class are \n",
    cm_d_b_no $byClass["Precision"],"\n",
    cm_d_b_no $byClass["Recall"],"\n",
   cm_d_b_no $byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'No' class are
##  0.6898525
##  0.7633662
##  0.72475
```

```
cat("The precision, recall and F1 of 'Yes' class are \n",
    cm_d_b_yes$byClass["Precision"],"\n",
    cm_d_b_yes$byClass["Recall"],"\n",
    cm_d_b_yes$byClass["F1"],"\n")
```

```
## The precision, recall and F1 of 'Yes' class are
##  0.7351418
##  0.6568019
##  0.6937673
```