BA810 Group Assignment

Group 5
Michael Peng, Subhiksha Sivasubramanian, Zheming Xu, Ziyue Zhou, Qianru Ai

Problem Statement

Nowadays cardiovascular disease accounts for most of the deaths. Nevertheless, heart disease is possible to be prevented with the right treatment or lifestyle, if we know what factors are related to its onset.

This dataset includes multiple features such as age and cholesterol that allow us to find out what factors significantly contribute to cardiovascular disease.

We are to build multiple prediction models with machine learning algorithm to predict the risk of having a heart disease based on dependent variables. Also, comprehensive examinations of all models will be performed to find out the best model.

Using our model, patients can assess their risk of heart disease and make life adjustments. At the same time, insurance and pharmaceutical companies can use the model to explore their customers' risk of developing the disease in order to improve their profitability.

Data

About data

Data Source: https://www.kaggle.com/fedesoriano/heart-failure-prediction

```r
library(data.table)
dd <- fread("/Users/aiqianru/Desktop/bu/BA810/group/heart_rowdata.csv")
str(dd)
```

```
## Classes 'data.table' and 'data.frame':   1068 obs. of  12 variables:
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sex           : chr  "M" "F" "M" "F" ...
##  $ ChestPainType : chr  "ATA" "NAP" "ATA" "ASY" ...
##  $ RestingBP     : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol   : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RestingECG    : chr  "Normal" "Normal" "ST" "Normal" ...
##  $ MaxHR         : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina: chr  "N" "N" "N" "Y" ...
##  $ Oldpeak       : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ ST_Slope      : chr  "Up" "Flat" "Up" "Flat" ...
##  $ HeartDisease  : int  1 0 0 1 0 0 0 0 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The original data contains 1068 rows and 12 columns. The following are details about some of the variables.

Target variable:

HeartDisease: integer. Dummy variable to indicate if the patient has heart disease or not.

Predictors:

ChestPainType: indicates the type of chest pain. TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic.

RestingBP: resting blood pressure [mm Hg].

Cholesterol: serum cholesterol [mm/dl].

FastingBS: fasting blood sugar. 1: FastingBS > 120 mg/dl, 0: otherwise.

ResstingECG: indicates resting electrocardiogram results.

Normal: Normal

ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria.

MaxHR: maximum heart rate achieved.

ExerciseAngina: exercise-induced angina. Y: Yes, N: No.

Oldpeak: the ratio of ST slope of the peak, exercise relative to rest(oldpeak).

ST_Slope: the slope shape of the peak exercise ST segment, a part of ECG

```r
# Package preperation
library(ggplot2)
library(gridExtra)
library(skimr)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-2
```

```r
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```r
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(Matrix)
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(ROCR)
library(tidyverse)
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()       masks data.table::between()
## x dplyr::combine()       masks randomForest::combine(), gridExtra::combine()
## x tidyr::expand()        masks Matrix::expand()
## x dplyr::filter()        masks stats::filter()
## x dplyr::first()         masks data.table::first()
## x dplyr::lag()           masks stats::lag()
## x dplyr::last()          masks data.table::last()
## x randomForest::margin() masks ggplot2::margin()
## x tidyr::pack()          masks Matrix::pack()
## x purrr::transpose()     masks data.table::transpose()
## x tidyr::unpack()        masks Matrix::unpack()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

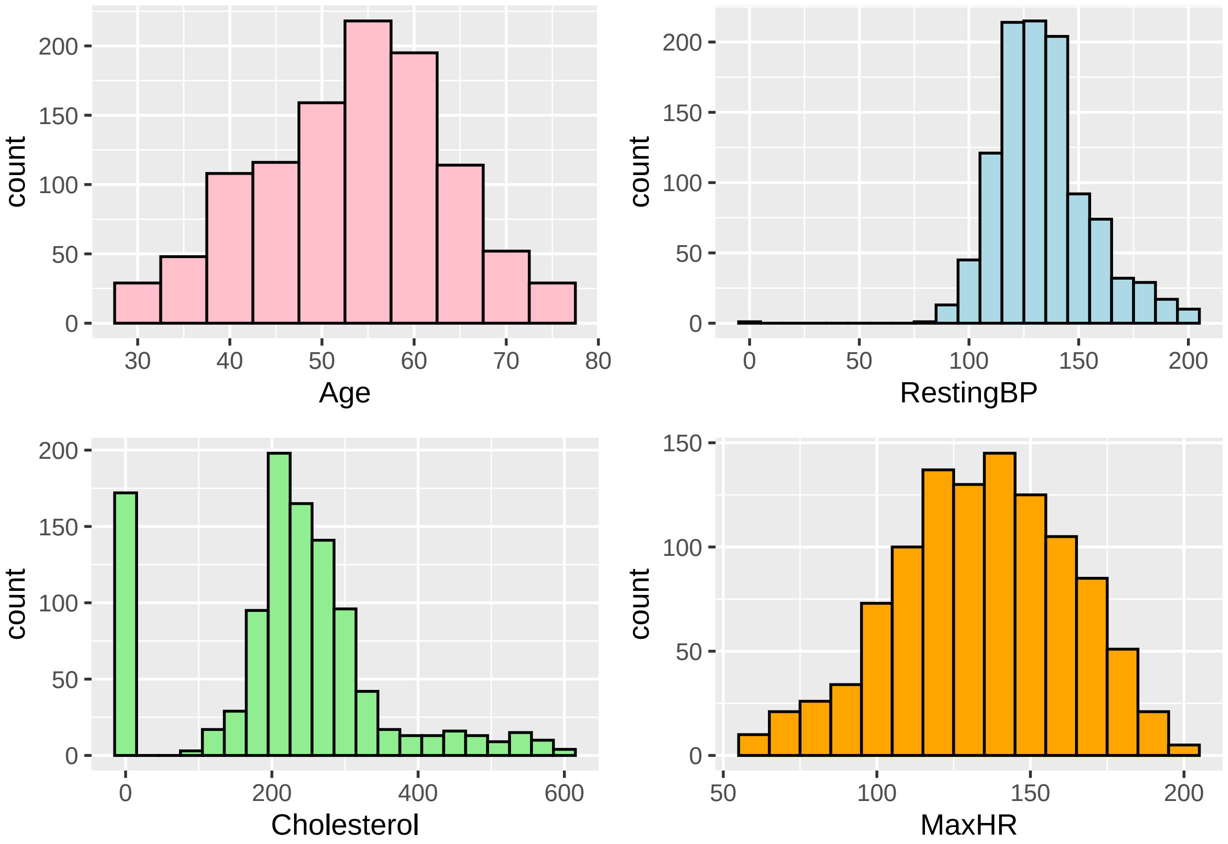
```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(modelr)
library(ISLR2)
library(ggraph)
```

Data Description

1. Distribution of continuous variables

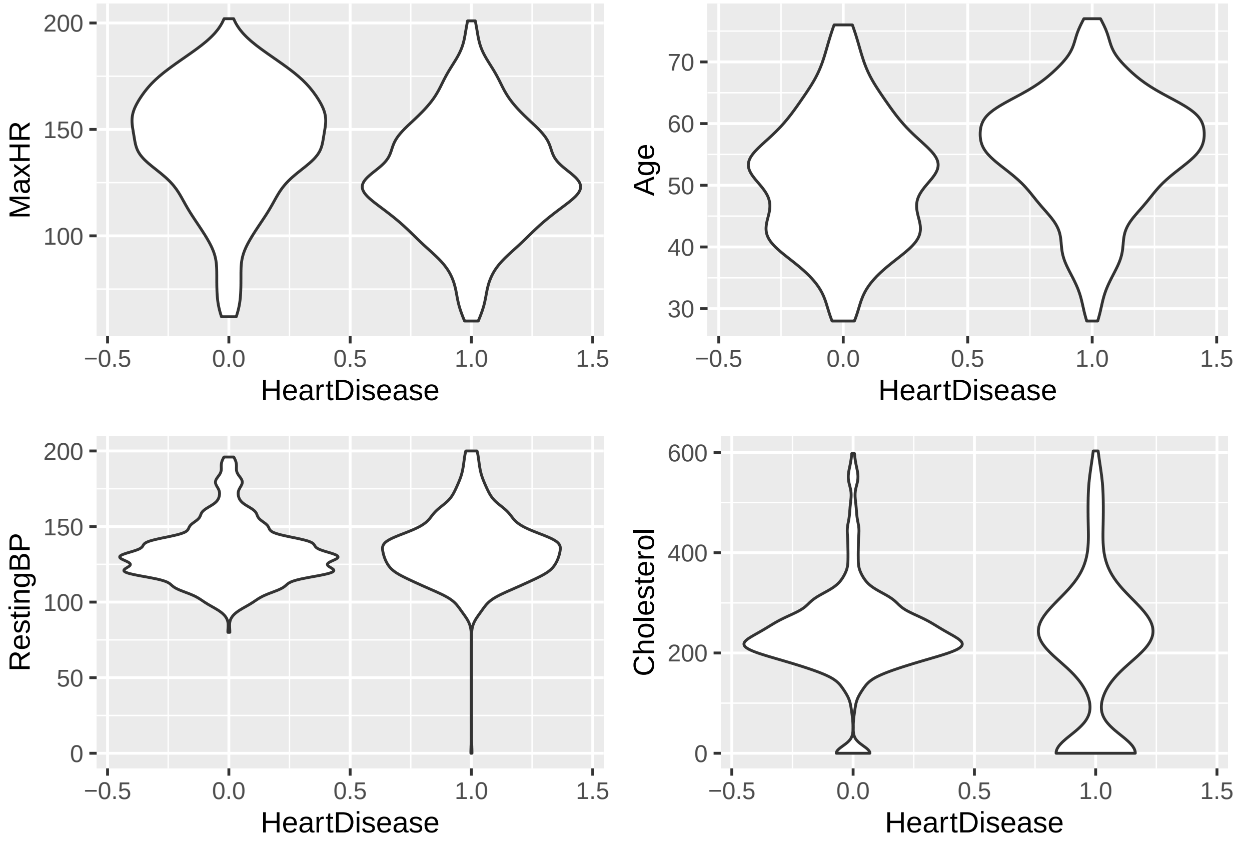
```r
con_p1 <- ggplot(dd, aes(Age)) + geom_histogram(binwidth=5,colour="black",fill='pink')
con_p2 <- ggplot(dd, aes(RestingBP)) + geom_histogram(binwidth=10,colour="black", fill='lightblue')
con_p3 <- ggplot(dd, aes(Cholesterol)) + geom_histogram(binwidth=30,colour="black", fill='lightgreen')
con_p4 <- ggplot(dd, aes(MaxHR)) + geom_histogram(binwidth=10,colour="black", fill='orange')
grid.arrange(grobs = list(con_p1,con_p2,con_p3,con_p4), ncol=2)
```



We could see that the distributions of Age and MaxHR are right skewed, while distributions of RestingBP and Cholesterol are left skewed. Meanwhile, Cholesterol has many 0 values, which need to be fixed later, as it is an impossible value.

2. Relationship between heart disease & numeric variables

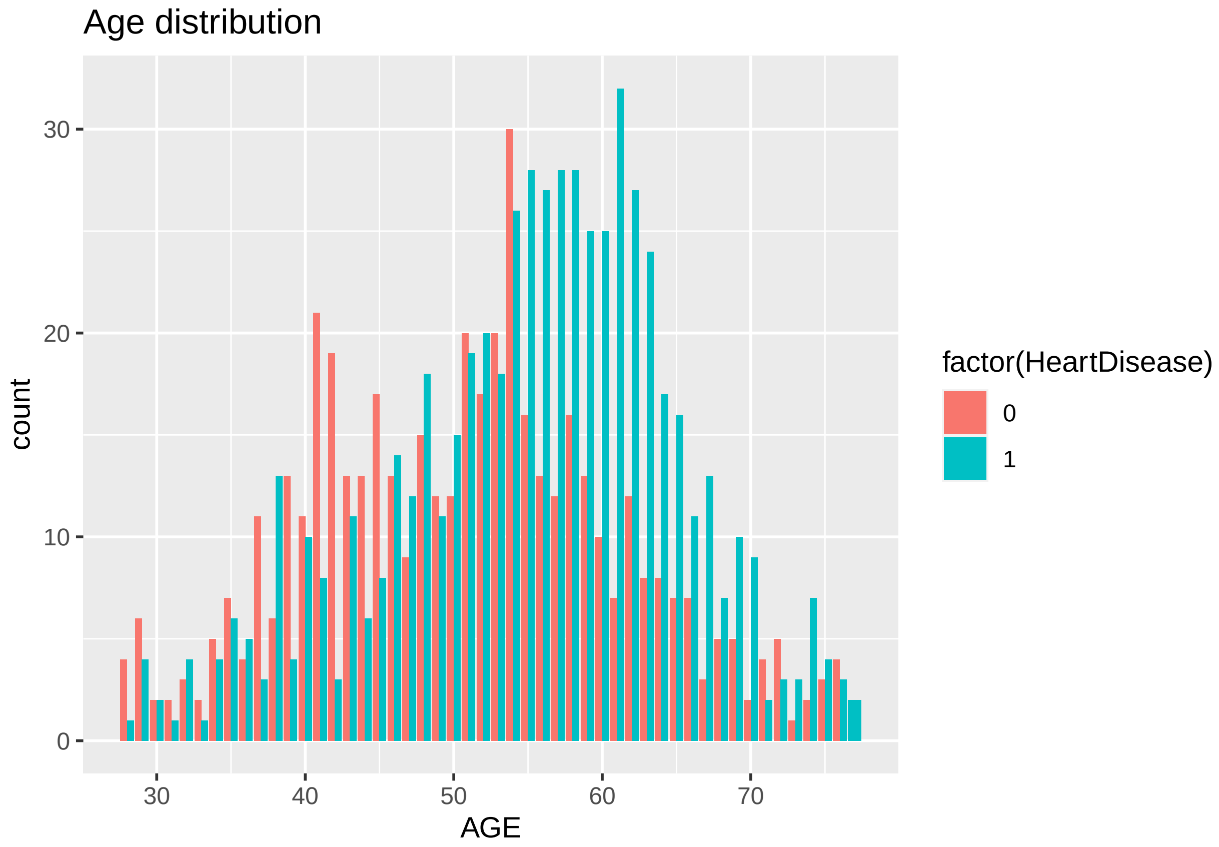
```r
rel_1 <- ggplot(dd, aes(HeartDisease, MaxHR, group=HeartDisease)) + geom_violin()
rel_2 <- ggplot(dd, aes(HeartDisease, Age, group=HeartDisease)) + geom_violin()
rel_3 <- ggplot(dd, aes(HeartDisease, RestingBP, group=HeartDisease)) + geom_violin()
rel_4 <- ggplot(dd, aes(HeartDisease, Cholesterol, group=HeartDisease)) + geom_violin()
grid.arrange(grobs = list(rel_1,rel_2,rel_3,rel_4), nrow=2)
```



The distribution of continuous variables was significantly different for all diseased and non-diseased populations. And obviously, 0 value also exists in RestingBP. We also need to fix it.

We further examined the distribution of age, as this is the variable we can most easily understand.

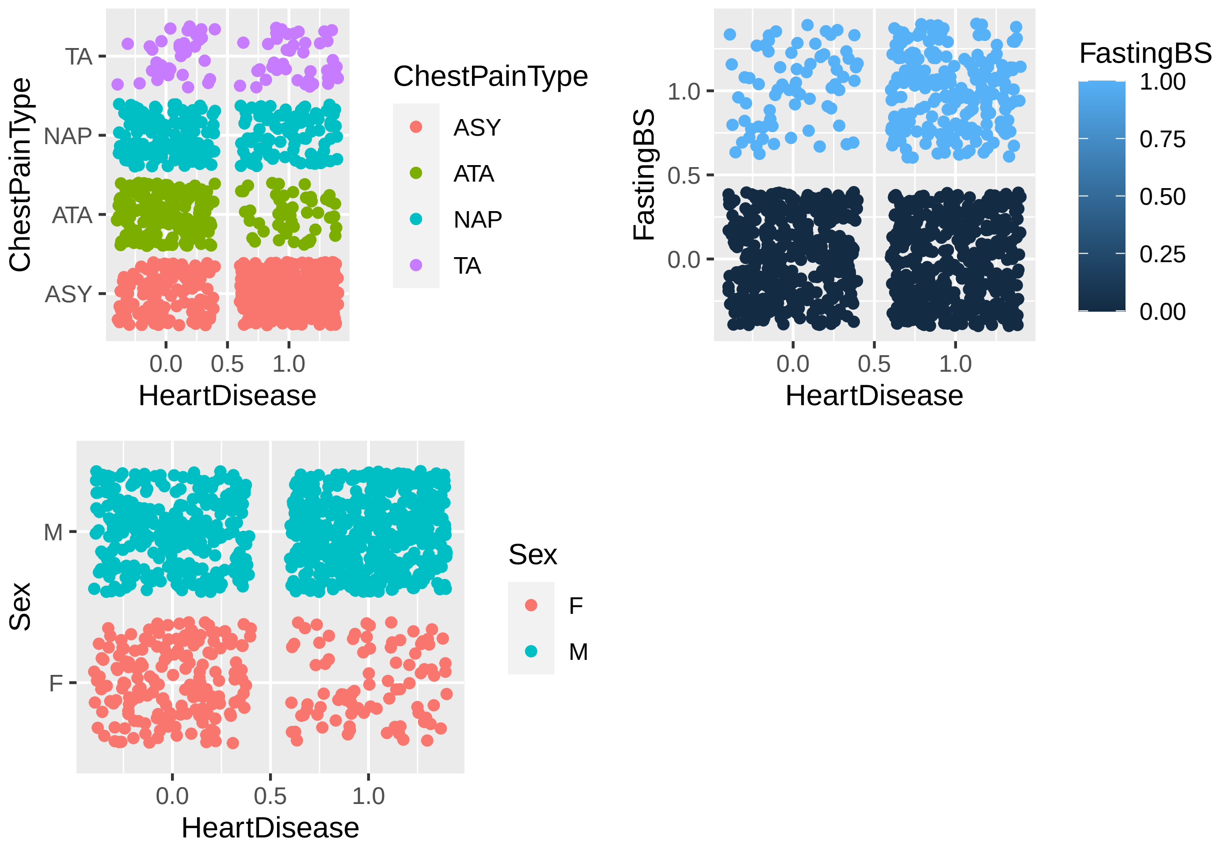
```
ggplot(dd,aes(Age,fill=factor(HeartDisease)))+
  labs(x = "AGE", y = "count",title='Age distribution ')+
  geom_bar(position = "dodge")
```



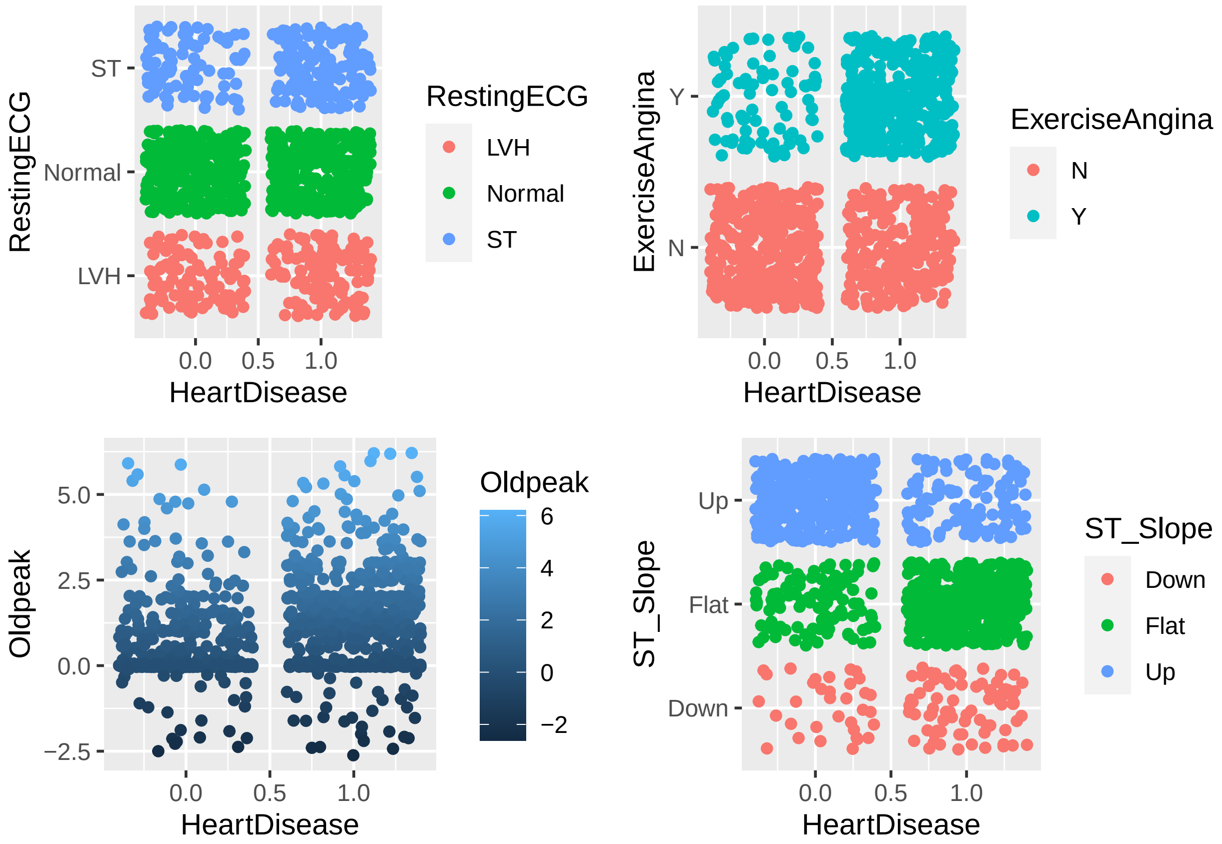
We can see that after about 55, the number of people who have heart disease is almost always greater than the number of people who do not have the disease.

3. Relationship between heart disease & categorical variables

```
rel_5 <- ggplot(dd, aes(HeartDisease, ChestPainType, colour=ChestPainType)) + geom_jitter()
rel_6 <- ggplot(dd, aes(HeartDisease, FastingBS, colour=FastingBS)) + geom_jitter()
rel_7 <- ggplot(dd, aes(HeartDisease, Sex, colour=Sex)) + geom_jitter()
rel_8 <- ggplot(dd, aes(HeartDisease, RestingECG, colour=RestingECG)) + geom_jitter()
rel_9 <- ggplot(dd, aes(HeartDisease, ExerciseAngina, colour=ExerciseAngina)) + geom_jitter()
rel_10 <- ggplot(dd, aes(HeartDisease, Oldpeak, colour=Oldpeak)) + geom_jitter()
rel_11 <- ggplot(dd, aes(HeartDisease, ST_Slope, colour=ST_Slope)) + geom_jitter()
grid.arrange(grobs = list(rel_5,rel_6,rel_7), nrow=2)
```



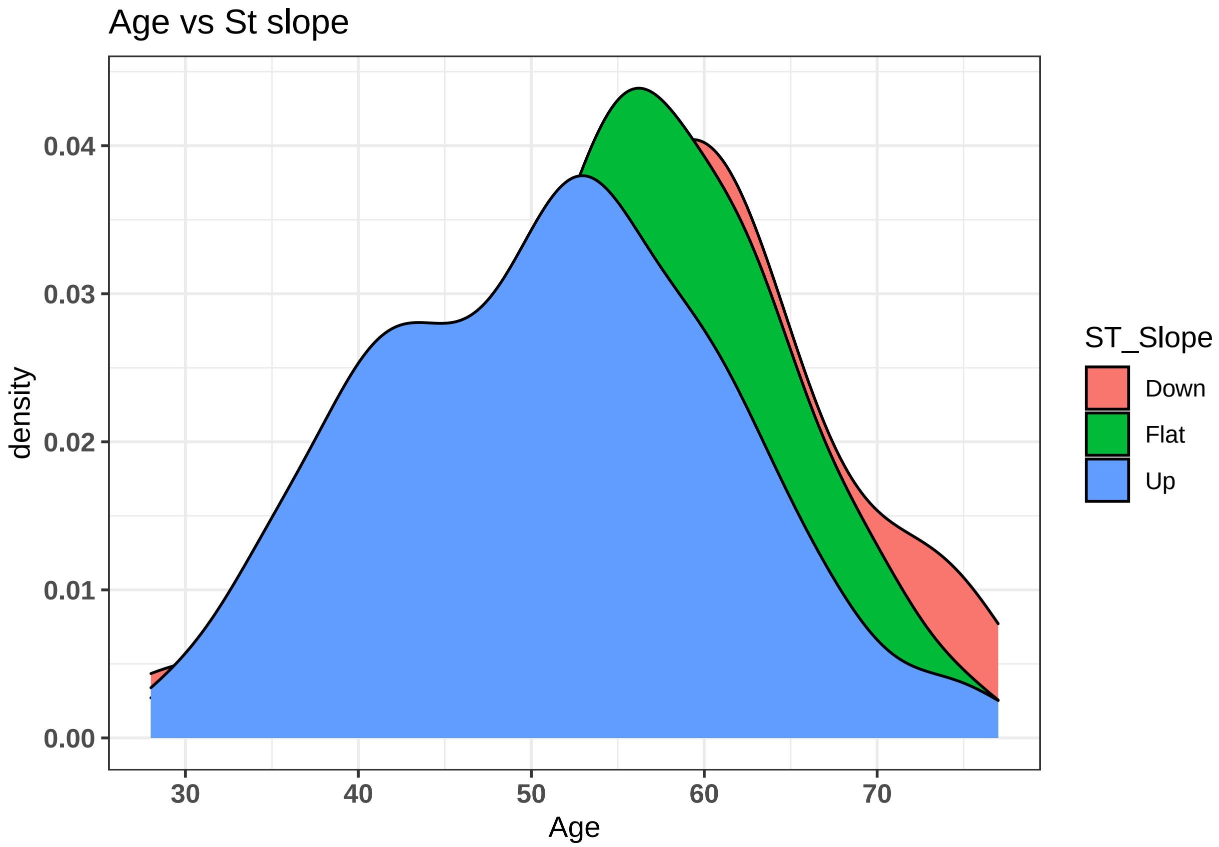
```r
grid.arrange(grobs = list(rel_8,rel_9,rel_10,rel_11), nrow=2)
```



From these plots, we find that ChestPainType, FatingBS, Sex, Oldpeak and ST_Slope might be more significant predictors.

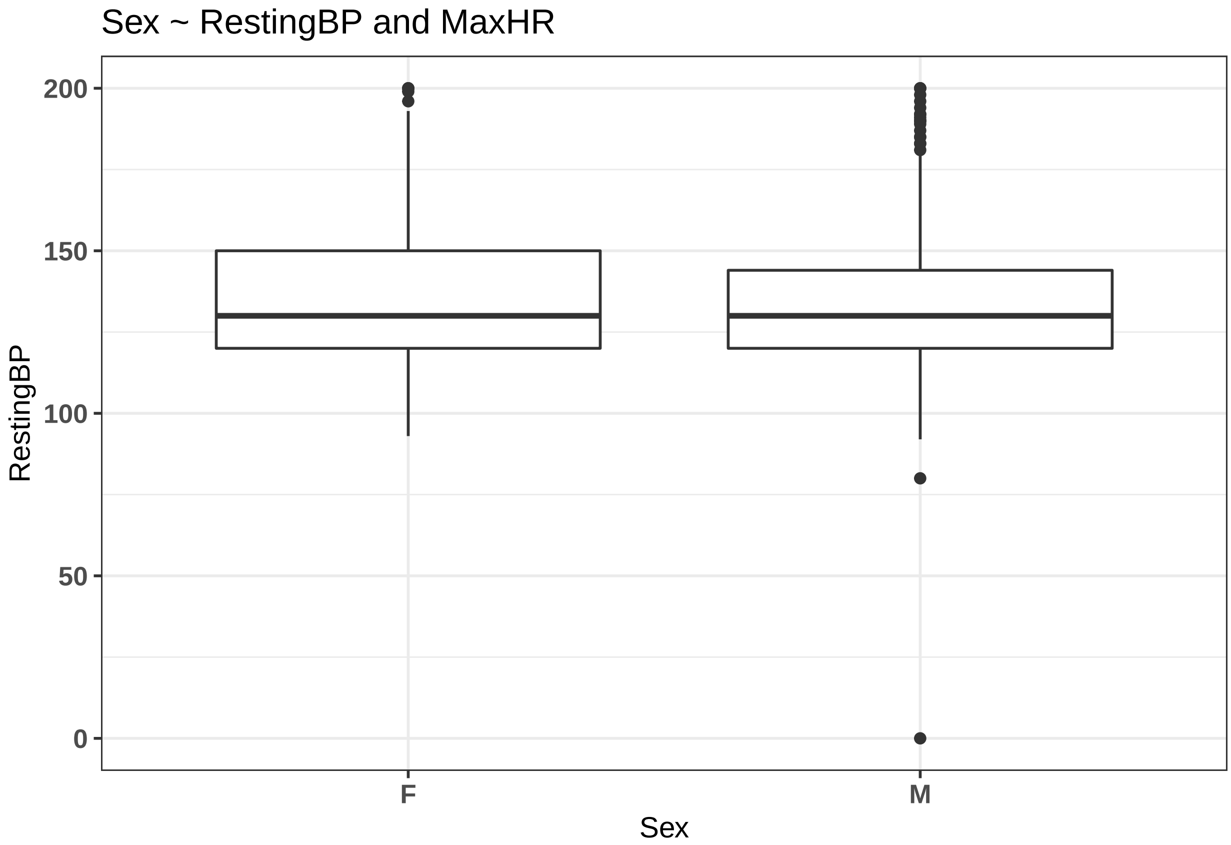
4. Relationship between predictors

```
ggplot(dd) +
  geom_density(aes(x = Age,
                   fill = ST_Slope)) +
  labs(x = 'Age') +
  ggtitle("Age vs St slope ") +
  theme_bw() +
  theme(axis.text.x = element_text(face = 'bold', size = 10),
        axis.text.y = element_text(face = 'bold', size = 10))
```



We found that the normal shape of the ST slope should be upward. From this graph, we can find that after about 54-year-old, the number of people with a flat ST slope is increasing, and then around 69-year-age, the number of people with a downward ST slope is increasing. It can be assumed that people's cardiopulmonary function becomes worse as they get older.

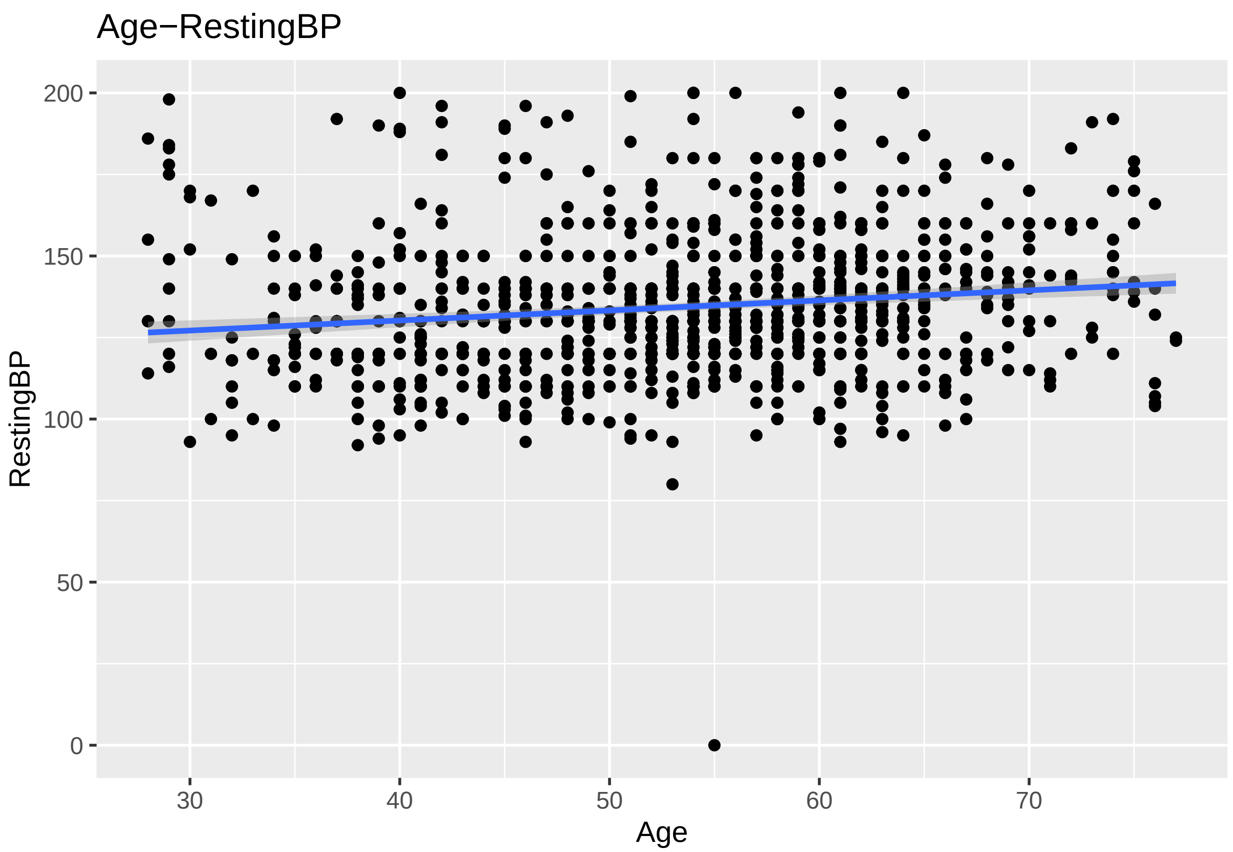
```
ggplot(dd) +
  geom_boxplot(aes(x = Sex, y = RestingBP,
                   fill = MaxHR)) +
  labs(x = 'Sex') +
  ggtitle("Sex ~ RestingBP and MaxHR ") +
  theme_bw() +
  theme(axis.text.x = element_text(face = 'bold', size = 10),
        axis.text.y = element_text(face = 'bold', size = 10))
```



From the plot, the mean values of RestingBP of male and female are nearly the same. But the 3rd quartile of female is larger than that of male. It can be assumed that the data distribution is wider for women, but more women have smaller values.

```
ggplot(dd,aes(Age,RestingBP))+geom_point()+geom_smooth(method=lm)+ggtitle("Age-RestingBP")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



As age going up, RestingBP slightly goes up too.

Data Processing

Create Dummy Variables

```
# create dummy variables for ChestPainType
unique(dd[,"ChestPainType"])
```

```
##    ChestPainType
## 1:           ATA
## 2:           NAP
## 3:           ASY
## 4:            TA
```

```
dd[,"ChestPain_ATA"] <- dd[,"ChestPainType"] == "ATA"
dd[,"ChestPain_NAP"] <- dd[,"ChestPainType"] == "NAP"
dd[,"ChestPain_ASY"] <- dd[,"ChestPainType"] == "ASY"
dd1 <- dd[,-3]

# create dummy variables for sex
dd1[,'Is_Female'] <- dd1[,'Sex'] == "F"
dd2 <- dd1[,-2]

# create dummy variables for restingECG
unique(dd2[,"RestingECG"])
```

```
##    RestingECG
## 1:     Normal
## 2:         ST
## 3:        LVH
```

```
dd2[,"RestingECG_Normal"] <- dd2[,"RestingECG"] == "Normal"
dd2[,"RestingECG_ST"] <- dd2[,"RestingECG"] == "ST"
dd3 <- dd2[,-5]

# create dummy variables for ST Slop
unique(dd3[,"ST_Slope"])
```

```
##    ST_Slope
## 1:       Up
## 2:     Flat
## 3:     Down
```

```
dd3[,"ST_Slope_Up"] <- dd3[,"ST_Slope"] == "Up"
dd3[,"ST_Slope_Flat"] <- dd3[,"ST_Slope"] == "Flat"
dd4 <- dd3[,-8]

# create dummy variables for Exercise Angina
dd4[,"ExerciseAngina"] <- dd4[,"ExerciseAngina"] == "Y"

str(dd4)
```

```
## Classes 'data.table' and 'data.frame':   1068 obs. of  16 variables:
##  $ Age              : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ RestingBP        : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol      : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MaxHR            : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina   : logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
##  $ Oldpeak          : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ HeartDisease     : int  1 0 0 1 0 0 0 0 1 1 ...
##  $ ChestPain_ATA    : logi  TRUE FALSE TRUE FALSE FALSE FALSE ...
##  $ ChestPain_NAP    : logi  FALSE TRUE FALSE FALSE TRUE TRUE ...
##  $ ChestPain_ASY    : logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
##  $ Is_Female        : logi  FALSE TRUE FALSE TRUE FALSE FALSE ...
##  $ RestingECG_Normal: logi  TRUE TRUE FALSE TRUE TRUE TRUE ...
##  $ RestingECG_ST    : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ ST_Slope_Up      : logi  TRUE FALSE TRUE FALSE TRUE TRUE ...
##  $ ST_Slope_Flat    : logi  FALSE TRUE FALSE TRUE FALSE FALSE ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

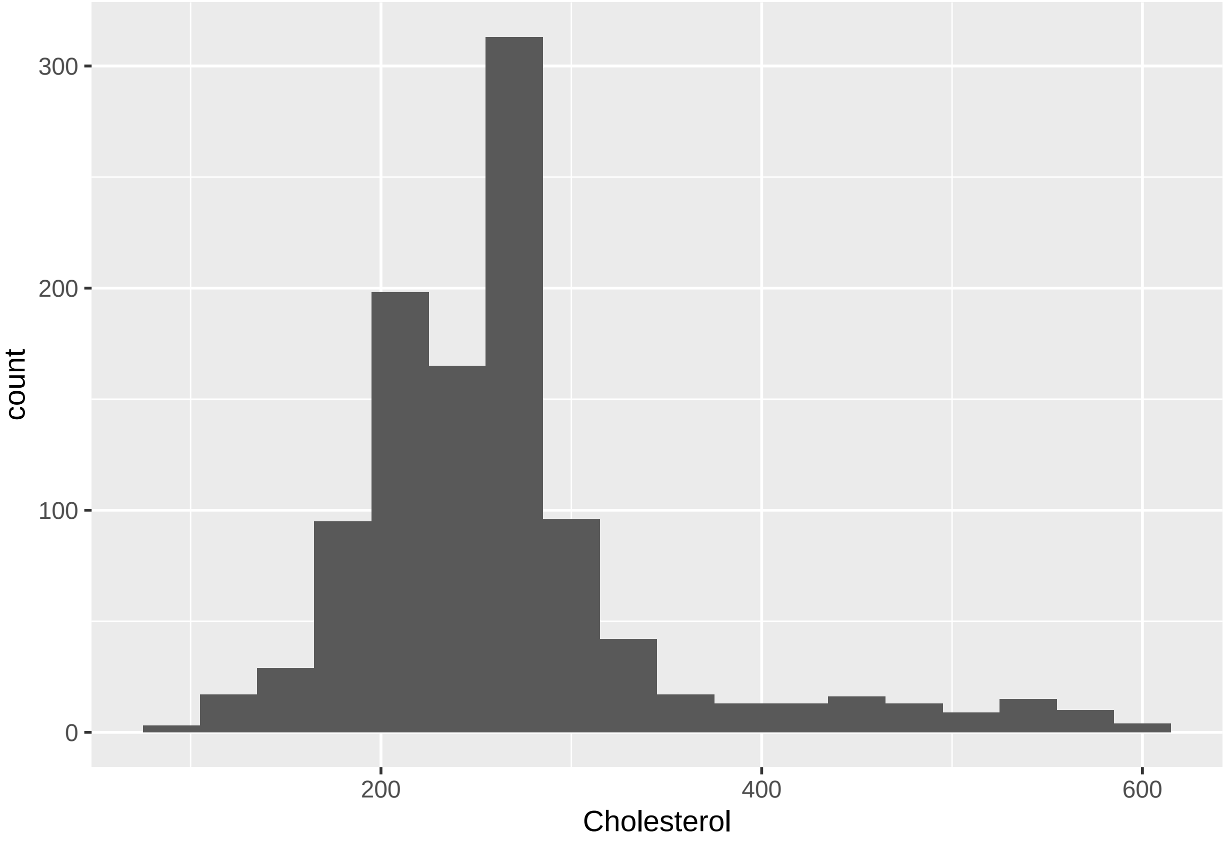
Processing Missing Value & Impossible value

```
sum(is.na(dd4))
```

```
## [1] 0
```

There is no missing value in this dataset, but we found there are many zeroes in Cholesterol, which is impossible. We decided to fill 0 with mean.

```
choleterol_value <- dd4$Cholesterol[dd4$Cholesterol  != 0 ]
choles_mean <- mean(choleterol_value)

dd4$Cholesterol[dd4$Cholesterol == 0] = choles_mean

# New distribution of Cholesterol
ggplot(dd4, aes(Cholesterol)) + geom_histogram(binwidth=30)
```

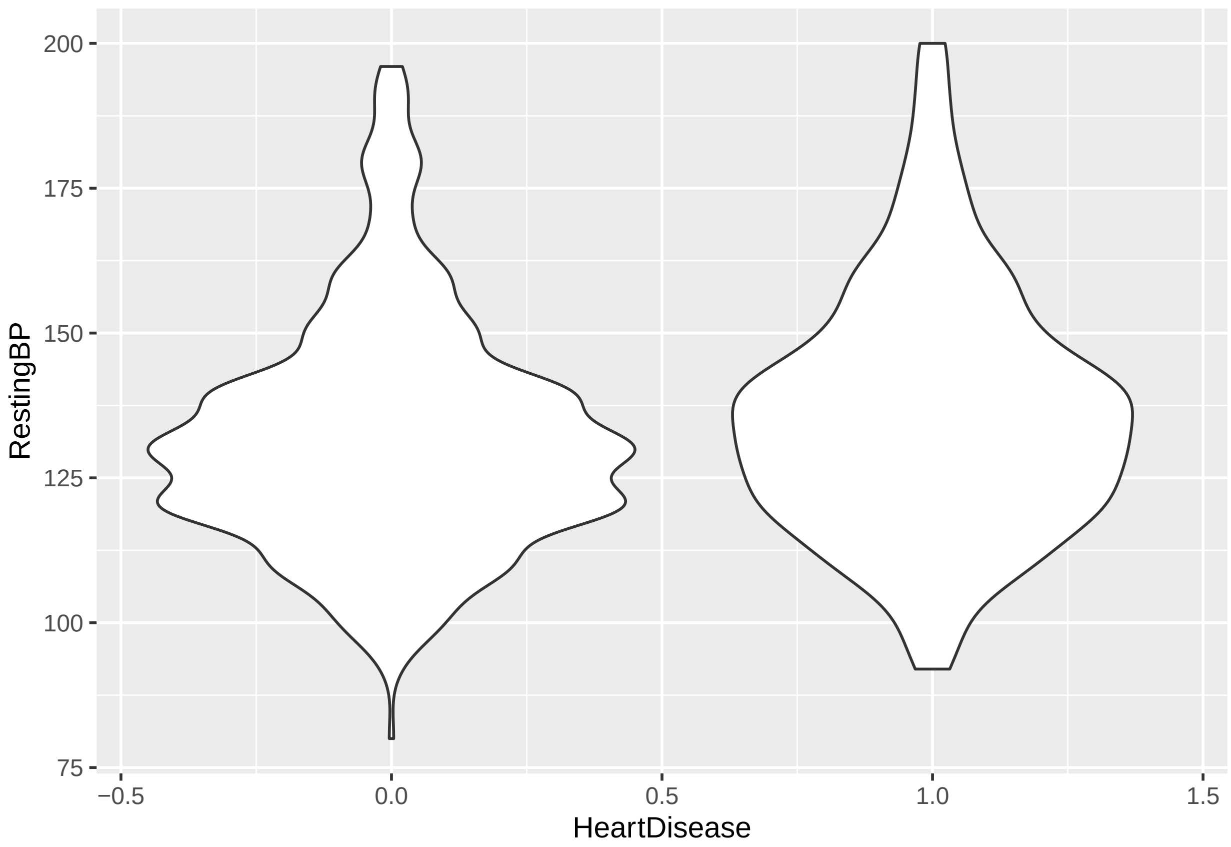


```
sum(dd4$RestingBP == 0)
```

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

There is one 0 value in RestingBP. We also fill it with the mean value.

```
RBP_mean <- mean(dd4$RestingBP[dd4$RestingBP  != 0 ])

dd4$RestingBP[dd4$RestingBP == 0] = RBP_mean

# New distribution of RestingBP
ggplot(dd4, aes(HeartDisease, RestingBP, group=HeartDisease)) + geom_violin()
```



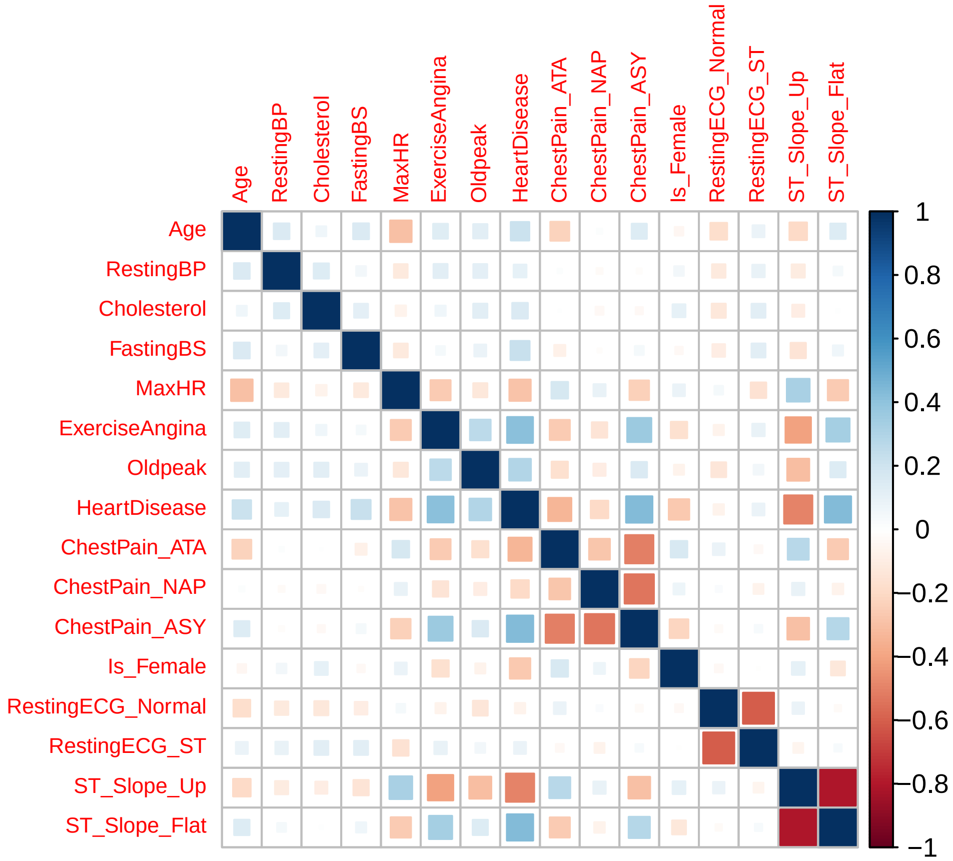
```
heart <-dd4
summary(heart)
```

```
##       Age          RestingBP      Cholesterol      FastingBS     
##  Min.   :28.00   Min.   : 80.0   Min.   : 85.0   Min.   :0.0000  
##  1st Qu.:46.00   1st Qu.:120.0   1st Qu.:216.0   1st Qu.:0.0000  
##  Median :54.00   Median :130.0   Median :261.0   Median :0.0000  
##  Mean   :53.31   Mean   :134.4   Mean   :264.7   Mean   :0.2669  
##  3rd Qu.:61.00   3rd Qu.:145.0   3rd Qu.:281.2   3rd Qu.:1.0000  
##  Max.   :77.00   Max.   :200.0   Max.   :603.0   Max.   :1.0000  
##      MaxHR       ExerciseAngina     Oldpeak      HeartDisease   
##  Min.   : 60.0   Mode :logical   Min.   :-2.6   Min.   :0.0000  
##  1st Qu.:116.0   FALSE:626       1st Qu.: 0.0   1st Qu.:0.0000  
##  Median :136.0   TRUE :442       Median : 0.7   Median :1.0000  
##  Mean   :134.9                   Mean   : 1.0   Mean   :0.5599  
##  3rd Qu.:155.2                   3rd Qu.: 1.8   3rd Qu.:1.0000  
##  Max.   :202.0                   Max.   : 6.2   Max.   :1.0000  
##  ChestPain_ATA   ChestPain_NAP   ChestPain_ASY   Is_Female      
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:853       FALSE:828       FALSE:535       FALSE:802      
##  TRUE :215       TRUE :240       TRUE :533       TRUE :266      
##                                                                 
##                                                                 
##                                                                 
##  RestingECG_Normal RestingECG_ST   ST_Slope_Up     ST_Slope_Flat  
##  Mode :logical     Mode :logical    Mode :logical    Mode :logical
##  FALSE:467         FALSE:831        FALSE:626        FALSE:555
##  TRUE :601         TRUE :237        TRUE :442        TRUE :513
##
##
##
```

Modeling

Relationship

```
cor_heart <- cor(heart)
corrplot(cor_heart,method="square",tl.cex = 0.7)
```



From the correlation matrix, we could see that all the predictors have more or less relationship with heart disease. Multicollineary may exits between some variables, but it does not affect consistency (the number of observation goes up, the accuracy will go up).

```
# Divide dataset
smp_size <- floor(0.8 * nrow(heart))

set.seed(123)
train_ind <- sample(seq_len(nrow(heart)), size = smp_size)
```

```
train <- heart[train_ind, ]
test <- heart[-train_ind, ]
x_train <- as.matrix(train[,-"HeartDisease"])
y_train <- as.matrix(train[,"HeartDisease"])
x_test <- as.matrix(test[,-"HeartDisease"])
y_test <- as.matrix(test[,"HeartDisease"])
```

Then, we will try linear regression, ridge regression, lasso regression, decision tree and random forest to build models.

Linear regression

We tried backward selection and forward selection separately to find the best subset of predictors.

```
# Create a model without predictor
model_none <- glm(HeartDisease ~ 1, family = "binomial", data = train)
# Create a model with all predictor
model_all <- glm(HeartDisease ~ ., family = "binomial", data = train)

# Stepwise regression backward
model_backward <- step(object = model_all, direction = "backward", trace = F)
summary(model_backward)
```

```
## 
## Call:
## glm(formula = HeartDisease ~ Age + Cholesterol + FastingBS + 
##     MaxHR + ExerciseAngina + Oldpeak + ChestPain_ASY + Is_Female + 
##     ST_Slope_Up + ST_Slope_Flat, family = "binomial", data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.8796  -0.5376   0.2804   0.6277   2.5233  
## 
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)    
## (Intercept)        -2.360401   0.898113  -2.628 0.008584 ** 
## Age                 0.017487   0.009274   1.886 0.059332 .  
## Cholesterol         0.004134   0.001155   3.579 0.000346 ***
## FastingBS           1.067338   0.224526   4.754 2.00e-06 ***
## MaxHR              -0.005828   0.003502  -1.664 0.096107 .  
## ExerciseAnginaTRUE  0.721365   0.209449   3.444 0.000573 ***
## Oldpeak             0.330446   0.070219   4.706 2.53e-06 ***
## ChestPain_ASYTRUE   1.418148   0.197786   7.170 7.49e-13 ***
## Is_FemaleTRUE      -1.148295   0.225375  -5.095 3.49e-07 ***
## ST_Slope_UpTRUE    -0.705574   0.329649  -2.140 0.032324 *  
## ST_Slope_FlatTRUE   0.999877   0.321855   3.107 0.001892 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1170.20  on 853  degrees of freedom
## Residual deviance:  704.04  on 843  degrees of freedom
## AIC: 726.04
##
## Number of Fisher Scoring iterations: 5
```

```
# Stepwise regression forward
model_forward <- step(object = model_all, scope = list(lower = model_none, upper = model_all),
                      direction = "forward", trace = F)
summary(model_forward)
```

```
##
## Call:
## glm(formula = HeartDisease ~ Age + RestingBP + Cholesterol +
##     FastingBS + MaxHR + ExerciseAngina + Oldpeak + ChestPain_ATA +
##     ChestPain_NAP + ChestPain_ASY + Is_Female + RestingECG_Normal +
##     RestingECG_ST + ST_Slope_Up + ST_Slope_Flat, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8965  -0.5404   0.2819   0.6177   2.5056
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -2.514681   1.169635  -2.150 0.031558 *
## Age                    0.015469   0.009550   1.620 0.105286
## RestingBP              0.004726   0.004734   0.998 0.318178
## Cholesterol            0.003914   0.001185   3.304 0.000953 ***
## FastingBS              1.060458   0.226271   4.687 2.78e-06 ***
## MaxHR                 -0.005966   0.003611  -1.652 0.098470 .
## ExerciseAnginaTRUE     0.698532   0.212017   3.295 0.000985 ***
## Oldpeak                0.316048   0.071274   4.434 9.24e-06 ***
## ChestPain_ATATRUE     -0.571281   0.379372  -1.506 0.132103
## ChestPain_NAPTRUE     -0.354296   0.352409  -1.005 0.314726
## ChestPain_ASYTRUE      1.073313   0.337693   3.178 0.001481 **
## Is_FemaleTRUE         -1.148695   0.227438  -5.051 4.40e-07 ***
## RestingECG_NormalTRUE  0.117251   0.237147   0.494 0.621006
## RestingECG_STTRUE      0.024545   0.295522   0.083 0.933805
## ST_Slope_UpTRUE       -0.682442   0.331915  -2.056 0.039775 *
## ST_Slope_FlatTRUE      0.996389   0.325515   3.061 0.002206 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1170.20  on 853  degrees of freedom
## Residual deviance:  700.65  on 838  degrees of freedom
## AIC: 732.65
##
## Number of Fisher Scoring iterations: 5
```

Since we have lower Residual deviance in forward model, so we choose forward model for subsequent processing.

```
# Prediction
pred_fw0 <-  predict(model_forward, type = "response", newdata = test)
pred_fw0 <- ifelse(pred_fw0 >= 0.5, 1, 0)
```

```
# MSE
mse_fw0 <- colMeans((y_train - pred_fw0)^2)
```

```
## Warning in y_train - pred_fw0: longer object length is not a multiple of shorter
## object length
```

```
# MSE
rmse_fw0 <- sqrt(mse_fw0)
```

Validation

```
set.seed(123)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model <- train(HeartDisease ~., data = train,
                    method = "leapForward",
                    tuneGrid = data.frame(nvmax = 1:15),
                    trControl = train.control
                    )
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
step.model$results
```

```
##    nvmax      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1      1 0.4300006 0.2588135 0.3693779 0.01943769 0.06739386 0.01487313
## 2      2 0.4015586 0.3513371 0.3220232 0.02273564 0.08746182 0.01741453
## 3      3 0.3974531 0.3648660 0.3136771 0.02250186 0.08115505 0.01671913
## 4      4 0.3941031 0.3752941 0.3097340 0.02154494 0.07093367 0.01911881
## 5      5 0.3844252 0.4046397 0.3003816 0.02477060 0.07961821 0.01950010
## 6      6 0.3797803 0.4192738 0.2957935 0.02412668 0.07127111 0.02085292
## 7      7 0.3758771 0.4305240 0.2929315 0.02688646 0.08063274 0.02298978
## 8      8 0.3707984 0.4444820 0.2875876 0.02456082 0.07449770 0.02228235
## 9      9 0.3710601 0.4432988 0.2873525 0.02505747 0.07522527 0.02303559
## 10    10 0.3703057 0.4458046 0.2863147 0.02560625 0.07658221 0.02355054
## 11    11 0.3697013 0.4472721 0.2857429 0.02528037 0.07509650 0.02357650
## 12    12 0.3701535 0.4462702 0.2858463 0.02556257 0.07638272 0.02321288
## 13    13 0.3706259 0.4450154 0.2862482 0.02547085 0.07606782 0.02301460
## 14    14 0.3709263 0.4440800 0.2867476 0.02606679 0.07774185 0.02349562
## 15    15 0.3706886 0.4447107 0.2865971 0.02604576 0.07759708 0.02349540
```

```
step.model$bestTune
```

```
##    nvmax
## 11    11
```

The model with 11 predictors performs the best.

```
coef(step.model$finalModel, 11)
```

```
##        (Intercept)                Age        Cholesterol          FastingBS
##       0.1772672422       0.0021612565       0.0006802975       0.1511175773
##              MaxHR ExerciseAnginaTRUE            Oldpeak  ChestPain_ATATRUE
##      -0.0009138245       0.1145940339       0.0530067107      -0.0533265889
##   ChestPain_ASYTRUE      Is_FemaleTRUE    ST_Slope_UpTRUE  ST_Slope_FlatTRUE
##       0.2187395871      -0.1638344216      -0.1386729931       0.1536800410
```

Thus, the best predictors are: Age, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, ChestPain_ATA, ChestPain_ASY, Is_Female, ST_Slope_Up, ST_Slope_Flat.

The best model is:

```
fw_best <- glm(HeartDisease ~ Age + Cholesterol + FastingBS +  MaxHR + ExerciseAngina +
                 Oldpeak + ChestPain_ATA + ChestPain_ASY + Is_Female + ST_Slope_Up +
                 ST_Slope_Flat, family = "binomial",data=train)
```

```
# Calculate MSE and RMSE
pred_fw <- predict(fw_best, type = "response", newdata = test)
pred_fw <- ifelse(pred_fw > 0.5, 1, 0)
# MSE
mse_fw <- mean((y_test - pred_fw)^2)
# RMSE
rmse_fw <- sqrt(mse_fw)
```

```
mse_fw <= mse_fw0
```

```
## HeartDisease
##         TRUE
```

```
rmse_fw <= mse_fw0
```

```
## HeartDisease
##         TRUE
```

Thus, the forward selection model after validation is better. We use this model to predict and calculate its accuracy rate.

```
table_mat_fw <- table(test$HeartDisease, pred_fw)
table_mat_fw
```

```
##    pred_fw
##      0  1
##   0 69 28
##   1 18 99
```

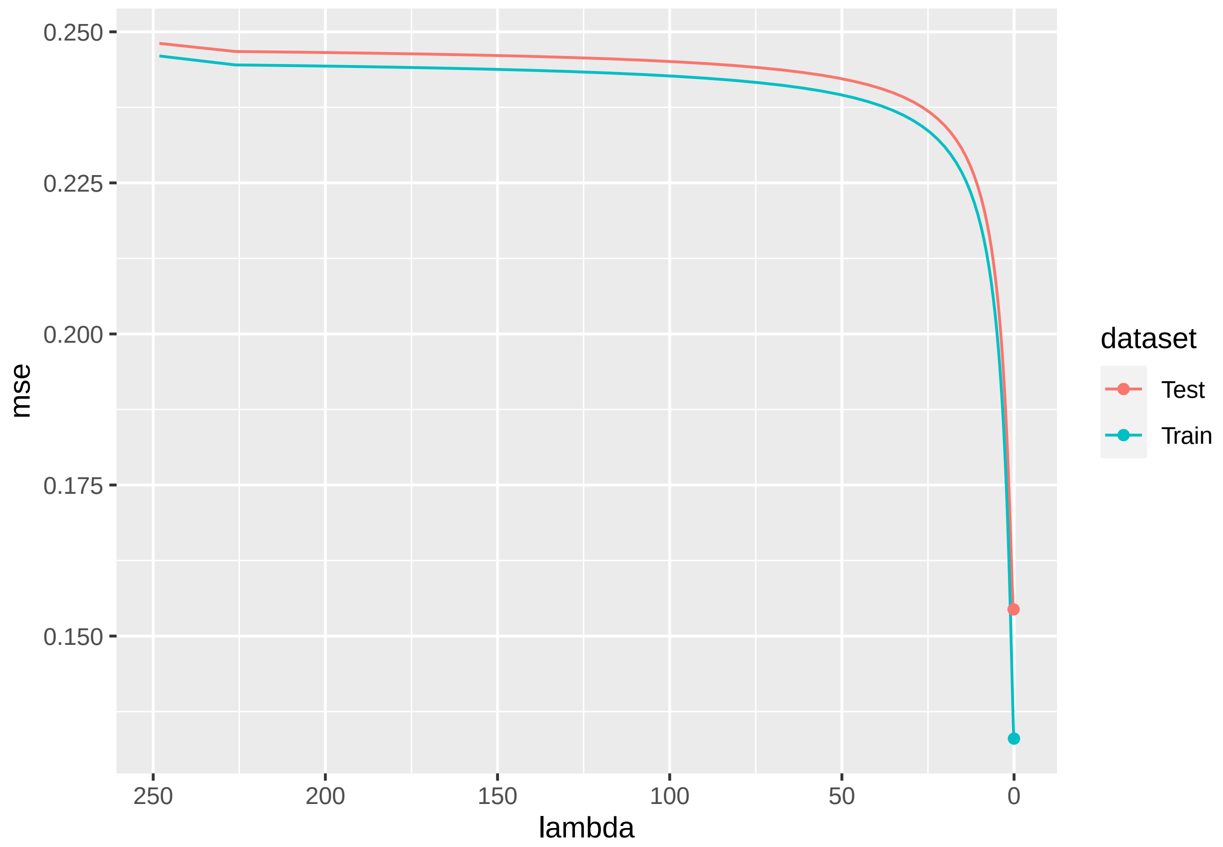
```
accuracy_fw <- sum(diag(table_mat_fw)) / sum(table_mat_fw)
print(accuracy_fw)
```

```
## [1] 0.7850467
```

Ridge regression

```
Ridge_model <- glmnet(x_train, y_train, alpha = 0, nlambda = 100)
#Prediction
y_train_hat_ridge <-data.table(predict(Ridge_model, x_train))
y_test_hat_ridge <- data.table(predict(Ridge_model, x_test))
```

```
# Compute MSEs
mse_train_ridge <- colMeans((y_train - y_train_hat_ridge)^2)
mse_test_ridge <- colMeans((y_test - y_test_hat_ridge)^2)
lambda_min_mse_train_ridge <- mse_train_ridge[which.min(mse_train_ridge)]
lambda_min_mse_test_ridge <- mse_test_ridge[which.min(mse_test_ridge)]

dd_mse_train_ridge <- data.table(
  lambda = Ridge_model$lambda,
  mse = mse_train_ridge,
  dataset = "Train"
)
dd_mse_ridge <- rbind(dd_mse_train_ridge, data.table(
  lambda = Ridge_model$lambda,
  mse = mse_test_ridge,
  dataset = "Test"
))

#plot
min <- dd_mse_ridge[mse == lambda_min_mse_test_ridge | mse == lambda_min_mse_train_ridge, ]

ggplot (dd_mse_ridge,aes(lambda,mse,col = dataset)) +
  geom_line() + geom_point(aes(lambda, mse, col = dataset), min) +
  scale_x_reverse()
```



The plot shows that when lambda approach to 0, the MSEs of both test and train data set become smaller. We conclude that the predictors are significant that no penalty should be posed.

```
# coefficients
plot(Ridge_model,xvar="lambda",lable=T)
```

```
## Warning in plot.window(...): "lable" is not a graphical parameter
```

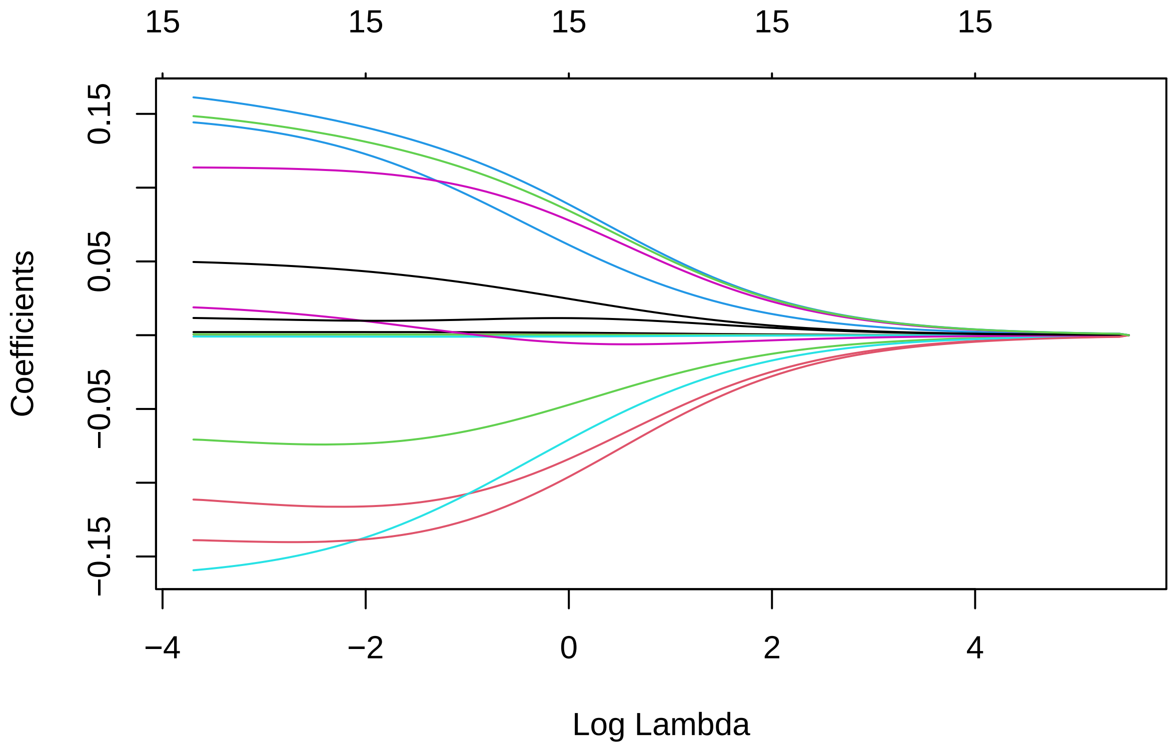
```
## Warning in plot.xy(xy, type, ...): "lable" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "lable" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "lable" is not a
## graphical parameter
```

```
## Warning in box(...): "lable" is not a graphical parameter
```

```
## Warning in title(...): "lable" is not a graphical parameter
```



```
# Validation
cv_model_ridge <- cv.glmnet(x_train,y_train, alpha = 0)
cv_best_lambda_ridge <- cv_model_ridge$lambda.min
cv_best_lambda_ridge
```

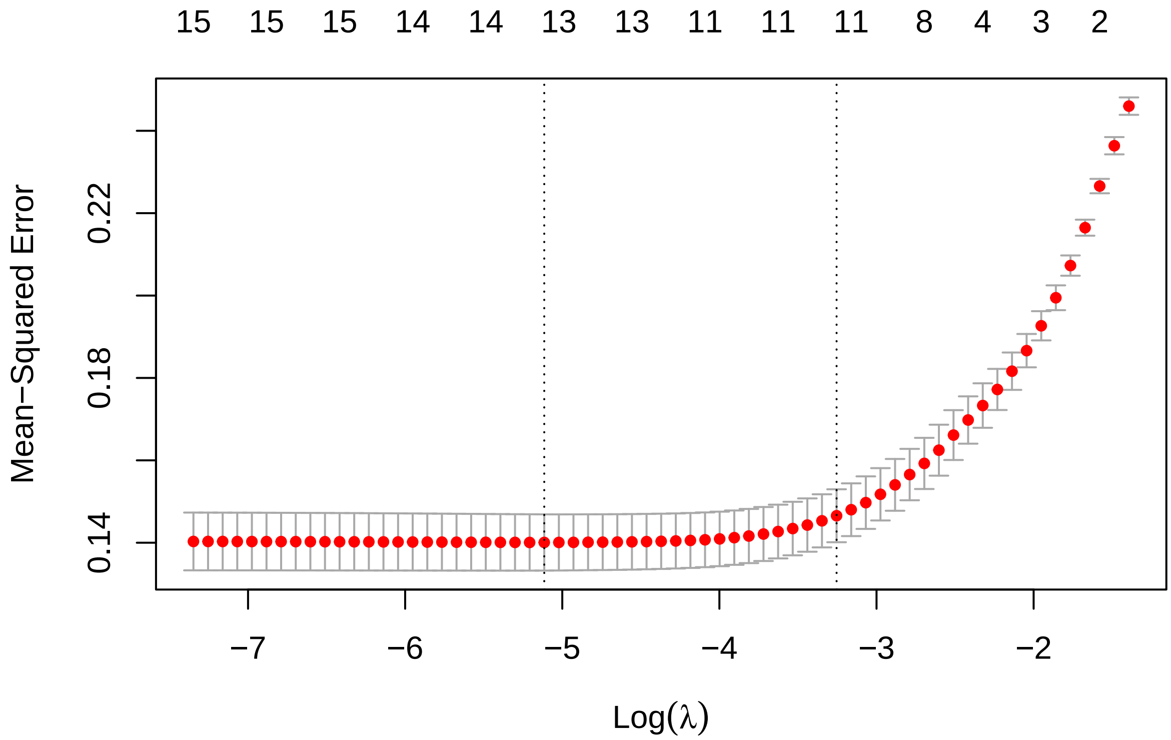
```
## [1] 0.1002031
```

```
best_cv_model_ridge <- glmnet(x_train, y_train, alpha = 0, lambda = cv_best_lambda_ridge)
cv_pred_ridge <- predict(best_cv_model_ridge, s = cv_best_lambda_ridge, newx = x_test)
cv_pred_ridge <- ifelse(cv_pred_ridge > 0.5, 1, 0)
cv_mse_test_ridge <- colMeans((y_test - cv_pred_ridge)^2)
```

```
log(cv_best_lambda_ridge)
```

```
## [1] -2.300556
```

```
cvfit_ridge <- cv.glmnet(x_train,y_train)
plot(cvfit_ridge)
```



The plot indicates that, as the log cv lambda gets smaller (cv lambda gets smaller), MSE gets smaller too.

```
print(paste('MSE test for ridge', lambda_min_mse_test_ridge))
```

```
## [1] "MSE test for ridge 0.154410856980364"
```

```
print(paste('MSE test for cross validation', cv_mse_test_ridge))
```

```
## [1] "MSE test for cross validation 0.205607476635514"
```

```
print(paste('RMSE test for ridge', sqrt(lambda_min_mse_test_ridge)))
```

```
## [1] "RMSE test for ridge 0.392951468988683"
```

```
print(paste('RMSE test for cross validation',sqrt(cv_mse_test_ridge)))
```

```
## [1] "RMSE test for cross validation 0.453439606381615"
```

The MSE and RMSE after cross validation are larger. The original model performs better. So we use the original ridge model to find the optimal lambda.

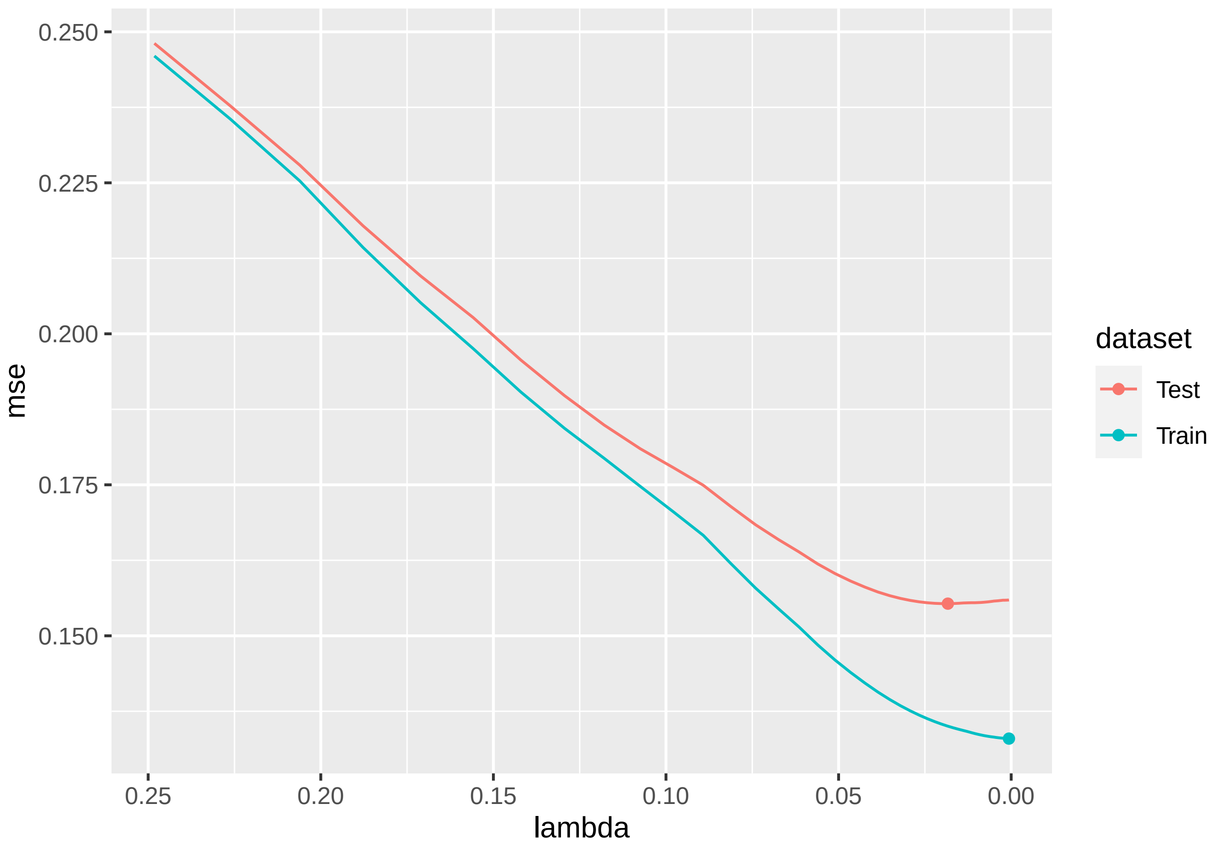
```
optimal_lambda_ridge <- min[dataset=="Test", lambda]
ridge_best <-  glmnet(x_train, y_train, alpha = 0, lambda=optimal_lambda_ridge)
pred_ridge <- predict(ridge_best, newx=x_test)
pred_ridge <- ifelse(pred_ridge > 0.5, 1, 0)
table_mat_ridge <- table(test$HeartDisease, pred_ridge)
accuracy_ridge <- sum(diag(table_mat_ridge)) / sum(table_mat_ridge)
print(accuracy_ridge)
```

```
## [1] 0.7943925
```

Lasso regression

```
lasso_model <- glmnet(x_train, y_train, alpha = 1, nlambda = 100)
# Predict responses
y_train_hat_lasso <-data.table(predict(lasso_model, x_train))
y_test_hat_lasso <- data.table(predict(lasso_model, x_test))
```

```
# Compute MSEs
mse_train_lasso <- colMeans((y_train - y_train_hat_lasso)^2)
mse_test_lasso <- colMeans((y_test - y_test_hat_lasso)^2)

lambda_min_mse_train_lasso <- mse_train_lasso[which.min(mse_train_lasso)]
lambda_min_mse_test_lasso <- mse_test_lasso[which.min(mse_test_lasso)]

dd_mse_train_lasso <- data.table(
  lambda = lasso_model$lambda,
  mse = mse_train_lasso,
  dataset = "Train"
)
dd_mse_lasso <- rbind(dd_mse_train_lasso, data.table(
  lambda = lasso_model$lambda,
  mse = mse_test_lasso,
  dataset = "Test"
))

#plot
min_lasso <- dd_mse_lasso[mse == lambda_min_mse_test_lasso | mse == lambda_min_mse_train_lasso, ]

ggplot (dd_mse_lasso,aes(lambda,mse,col = dataset)) +
  geom_line() + geom_point(aes(lambda, mse, col = dataset), min_lasso) +
  scale_x_reverse()
```



```
# coefficients
plot(lasso_model,xvar="lambda",lable=T)
```

```
## Warning in plot.window(...): "lable" is not a graphical parameter
```

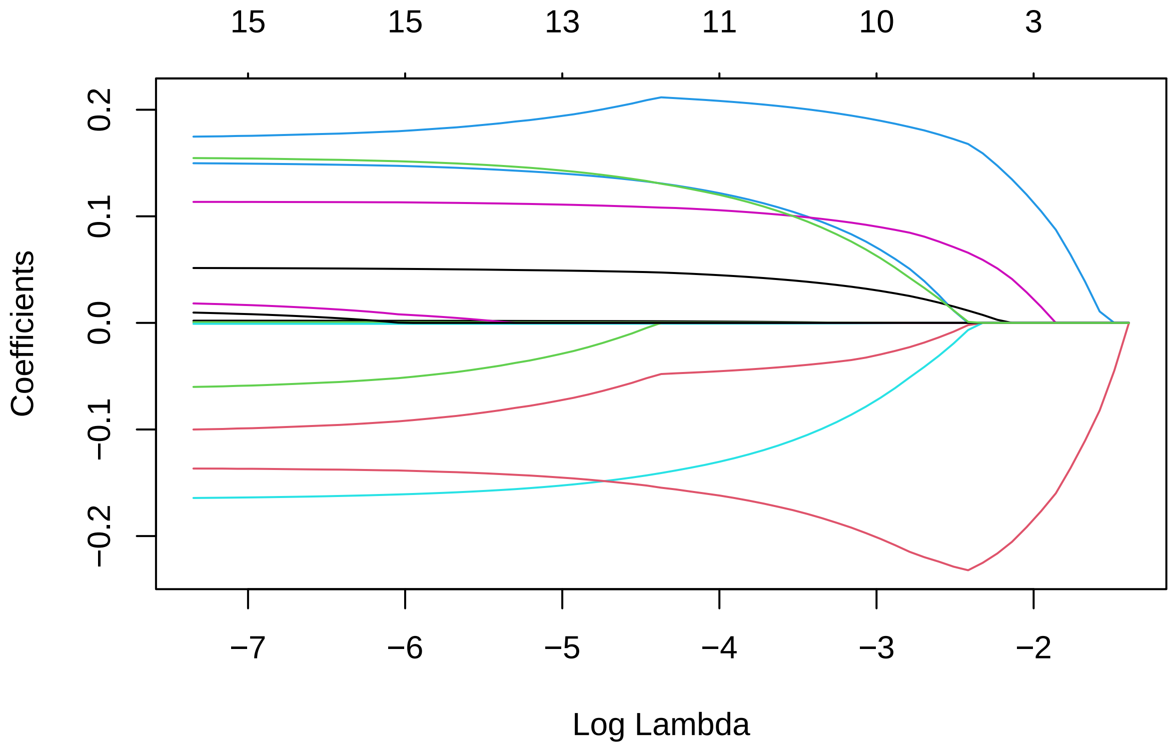
```
## Warning in plot.xy(xy, type, ...): "lable" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "lable" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "lable" is not a
## graphical parameter
```

```
## Warning in box(...): "lable" is not a graphical parameter
```

```
## Warning in title(...): "lable" is not a graphical parameter
```



```
# Validation
cv_model_lasso <- cv.glmnet(x_train,y_train, alpha = 1)
cv_best_lambda_lasso <- cv_model_lasso$lambda.min
cv_best_lambda_lasso
```

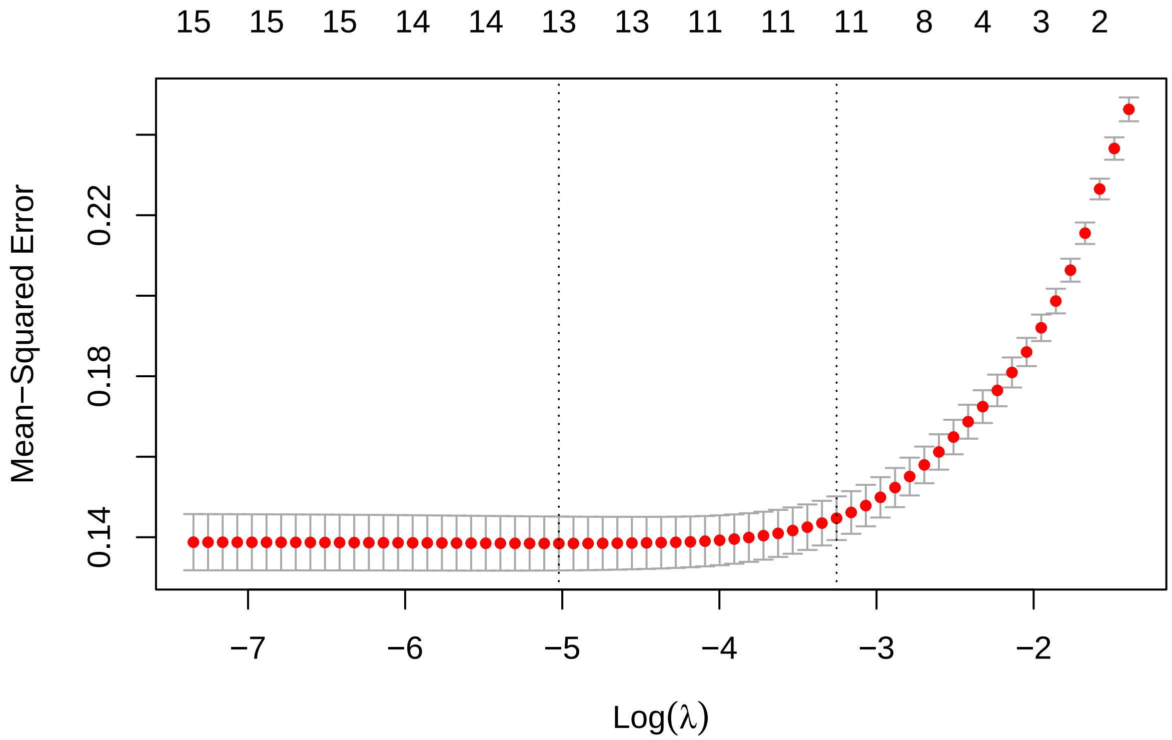
```
## [1] 0.007940922
```

```
best_cv_model_lasso <- glmnet(x_train, y_train, alpha = 1, lambda = cv_best_lambda_lasso)
cv_pred_lasso <- predict(best_cv_model_lasso, s = cv_best_lambda_lasso, newx = x_test)
cv_pred_lasso <- ifelse(cv_pred_lasso > 0.5, 1, 0)
cv_mse_test_lasso <- colMeans((y_test - cv_pred_lasso)^2)
```

```
log(cv_best_lambda_lasso)
```

```
## [1] -4.835726
```

```
cvfit_lasso <- cv.glmnet(x_train,y_train)
plot(cvfit_lasso)
```



The plot indicates that, as the log cv lambda gets smaller (cv lambda gets smaller), MSE gets smaller too.

```
print(paste('MSE test for lasso', lambda_min_mse_test_lasso))
```

```
## [1] "MSE test for lasso 0.155325663655102"
```

```
print(paste('MSE test for cross validation', cv_mse_test_lasso))
```

```
## [1] "MSE test for cross validation 0.214953271028037"
```

```
print(paste('RMSE test for lasso', sqrt(lambda_min_mse_test_lasso)))
```

```
## [1] "RMSE test for lasso 0.394113769938456"
```

```
print(paste('RMSE test for cross validation',sqrt(cv_mse_test_lasso)))
```

```
## [1] "RMSE test for cross validation 0.463630532890186"
```

The MSE and RMSE after cross validation are larger. The original model performs better. So we use the original lasso model to find the optimal lambda.

```r
optimal_lambda_lasso <- min_lasso[dataset=="Test", lambda]
optimal_lambda_lasso
```

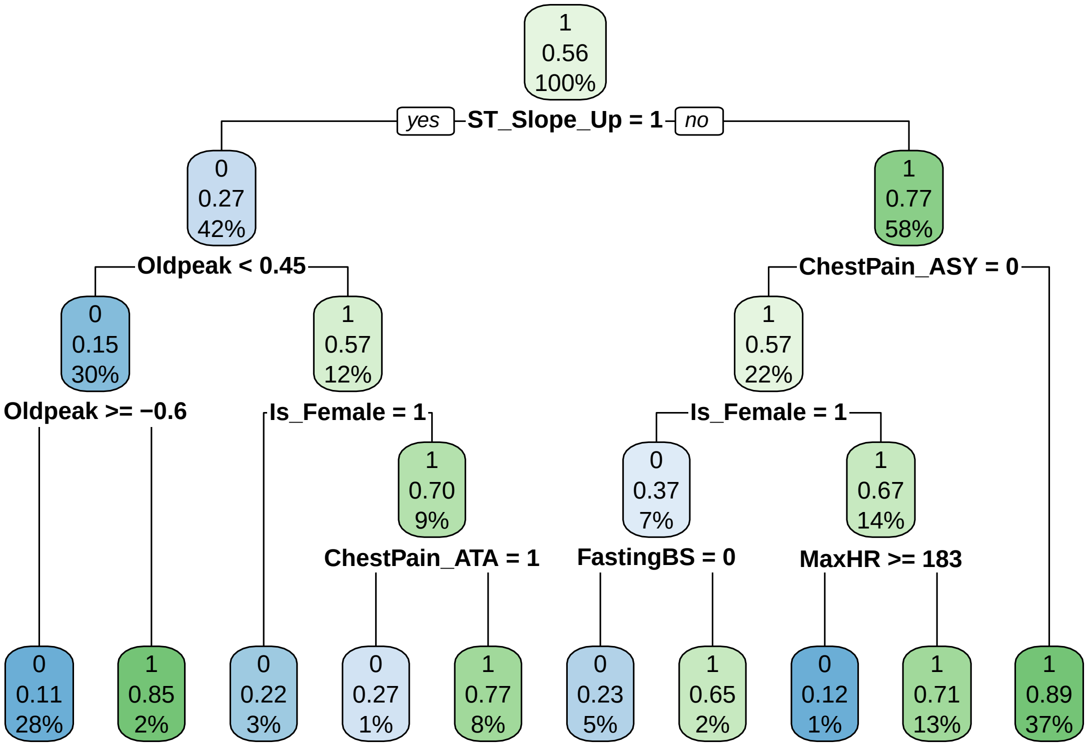
```
## [1] 0.01834456
```

```r
lasso_best <-  glmnet(x_train, y_train, alpha = 1, lambda=min_lasso[dataset=="Test", lambda])
pred_lasso <- predict(lasso_best, newx=x_test)
pred_lasso <- ifelse(pred_lasso > 0.5, 1, 0)
table_mat_lasso <- table(test$HeartDisease, pred_lasso)
accuracy_lasso <- sum(diag(table_mat_lasso)) / sum(table_mat_lasso)
print(accuracy_lasso)
```

```
## [1] 0.7990654
```

Decision tree

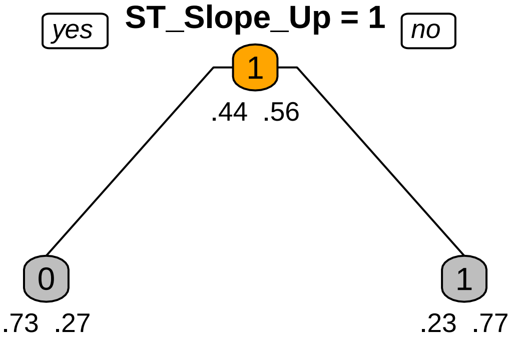
```r
tree_model <- rpart(HeartDisease~., data = train, method = 'class')
rpart.plot(tree_model, extra = 106)
```



Complexity parameter used in pruning is called as cp which is used to determine the optimal size of the tree and prevent over fitting. For pruning, we tried two ways: post pruning and pre pruning.

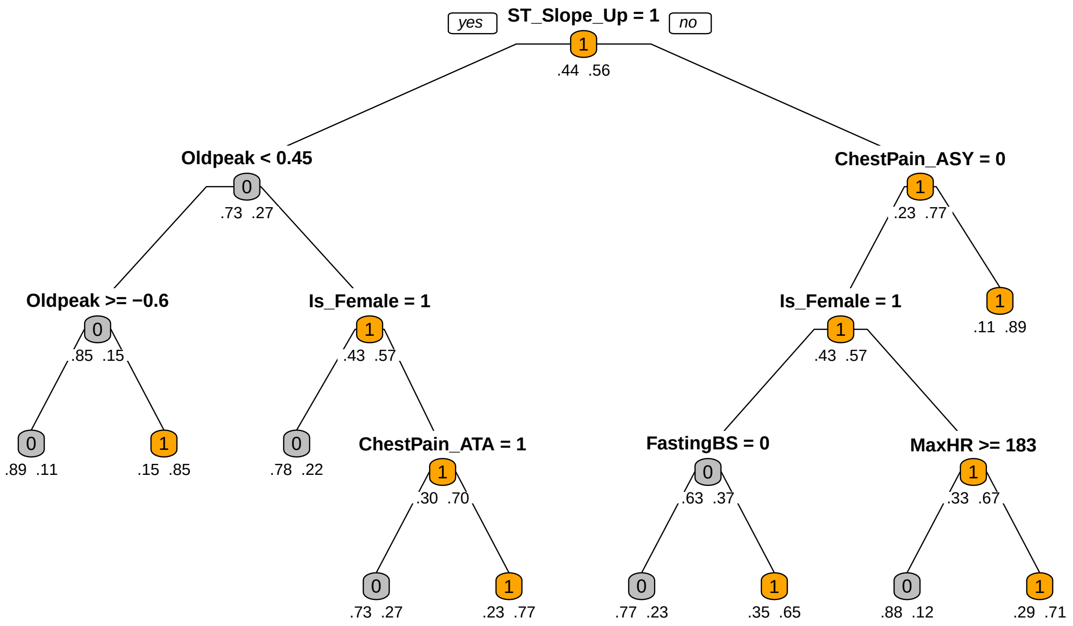
Post pruning

```r
#Pruning
##Post pruning the tree
Pruned_tree<-prune(tree_model,cp=0.16)
prp(Pruned_tree,box.col=c("Grey", "Orange")[tree_model$frame$yval],varlen=0,faclen=0, type=1,extra=4,und
```



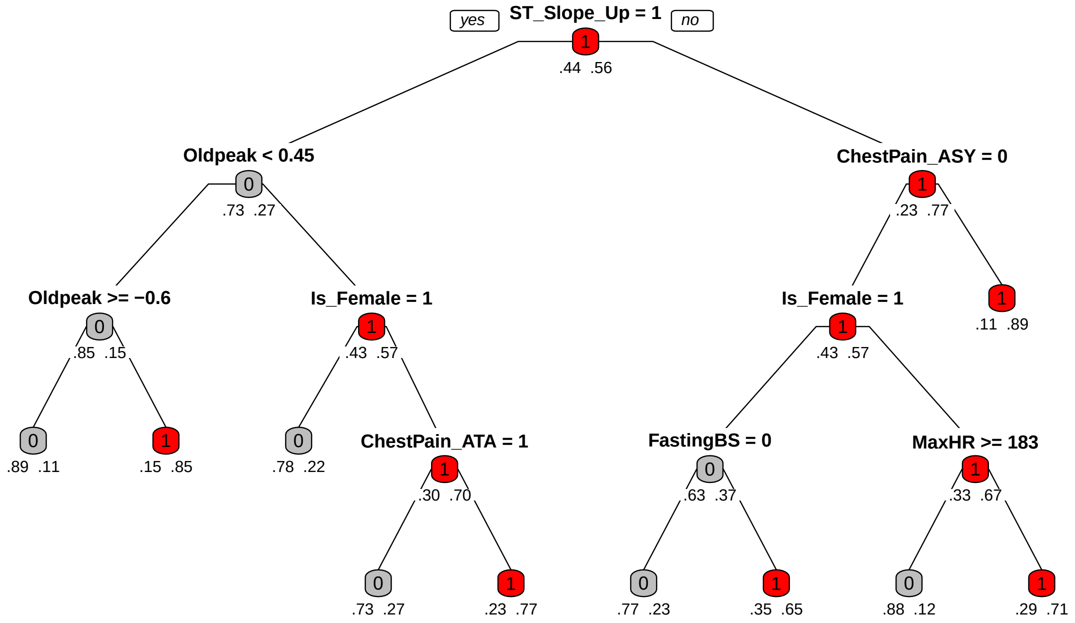
```r
# Conducting pruning
Ecom_Tree_prune<-prune(tree_model,cp=0.0029646)
#Before pruning
prp(tree_model,box.col=c("Grey", "Orange")[tree_model$frame$yval],varlen=0,faclen=0, type=1,extra=4,unde
```

TREE BEFORE PRUNING



```
#After Pruning
prp(Ecom_Tree_prune,box.col=c("Grey", "red")[tree_model$frame$yval],varlen=0,faclen=0, type=1,extra=4,u
```

TREE AFTER PRUNING



Two sample trees would be built to analyze the difference with due to different cp values.

```
Sample_tree<-rpart(HeartDisease~., method="class", data=train,
                   control=rpart.control(minsplit=2, cp=0.001))
object.size(Sample_tree)
```

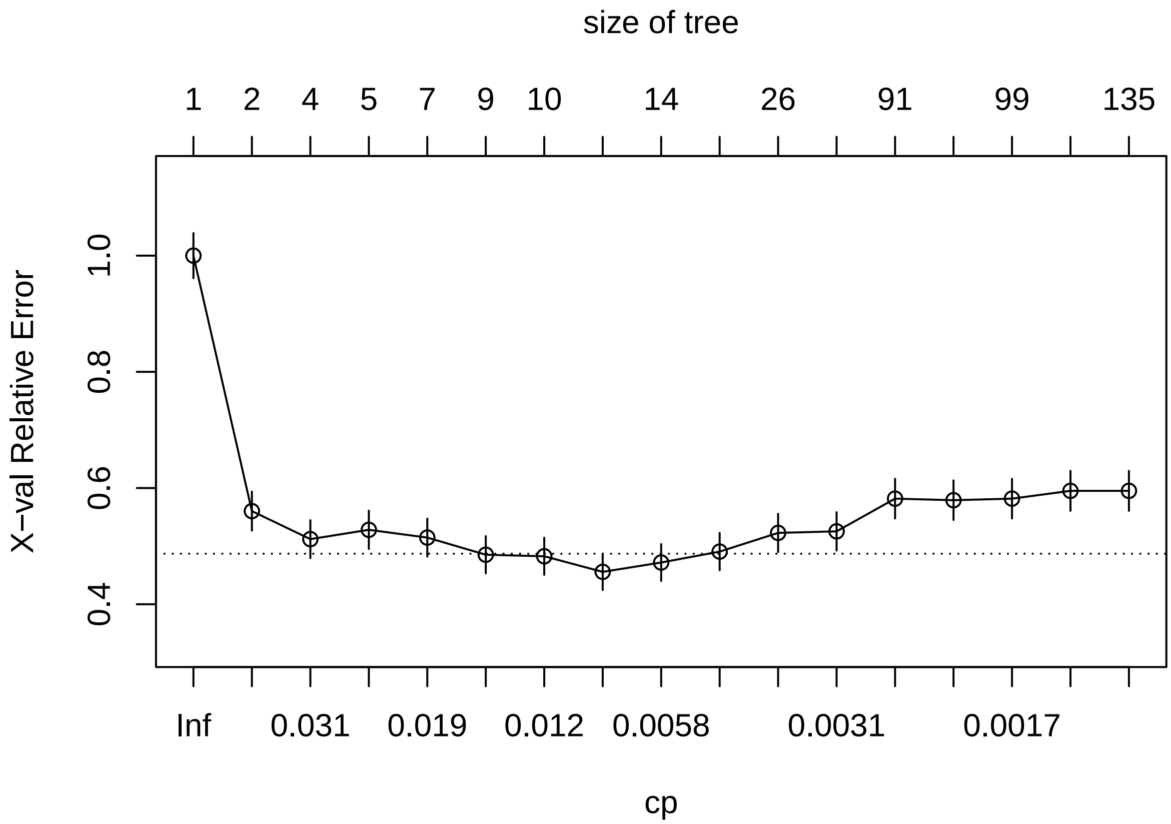
```
## 206968 bytes
```

```
Sample_tree_1<-rpart(HeartDisease~., method="class", data=train,
                     control=rpart.control(minsplit=2, cp=0.1))
object.size(Sample_tree_1)
```

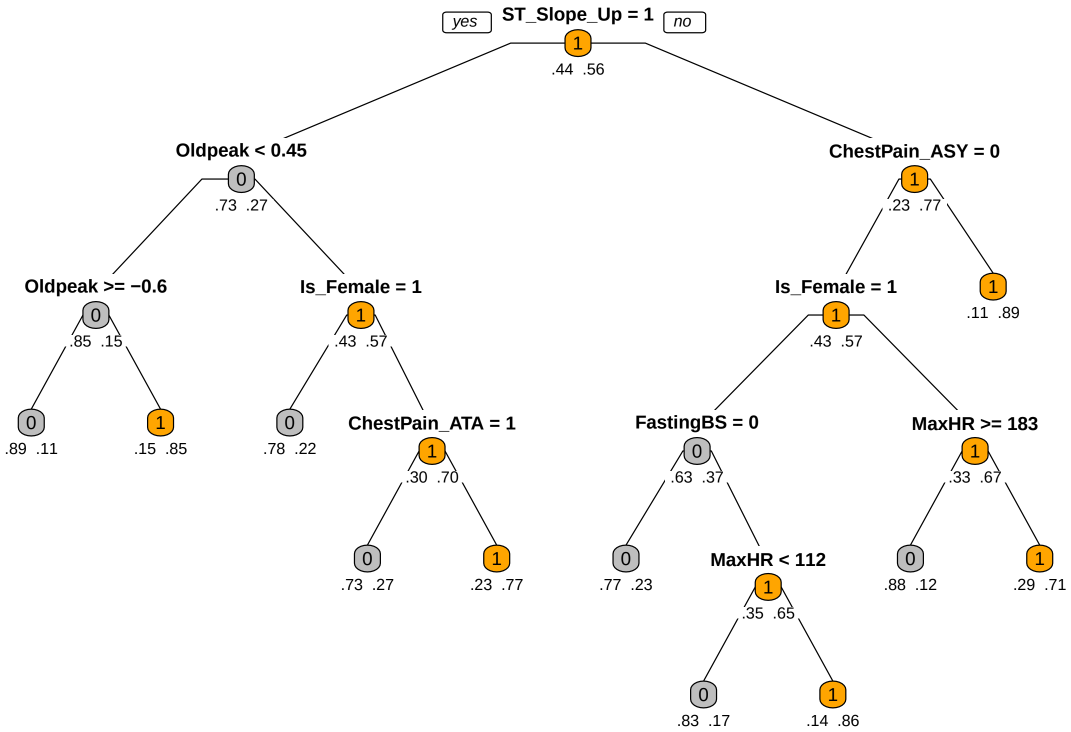
```
## 135368 bytes
```

We could see that the tree with smaller cp (Sample tree) is much larger than the one with larger cp (Sample tree 1). This is because model with larger cp stops running earlier.

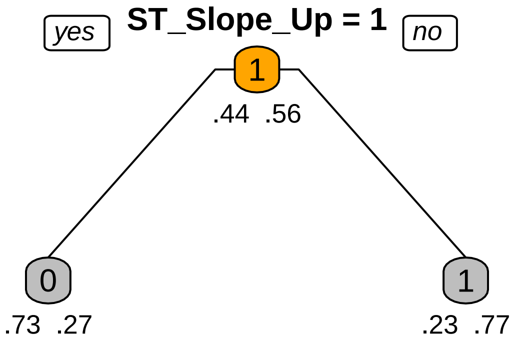
```
#CROSS VALIDATION RESULTS
plotcp(Sample_tree)
```



```
#New model with selected cp
cp <- data.table(Sample_tree[["cptable"]])
optimal_cp <- mean(cp[xerror == min(xerror),CP])
Sample_tree_2<-rpart(HeartDisease~., method="class", data=train,
                     control=rpart.control(minsplit=2, cp=optimal_cp))
#Plotting the Tree
prp(Sample_tree_2,box.col=c("Grey", "Orange")[Sample_tree_2$frame$yval],
    varlen=0,faclen=0, type=1,extra=4,under=TRUE)
```



```
#Post pruning the old tree
Pruned_tree<-prune(Sample_tree,cp=0.23)
prp(Pruned_tree,box.col=c("Grey", "Orange")[Sample_tree$frame$yval],
    varlen=0,faclen=0, type=1,extra=4,under=TRUE)
```

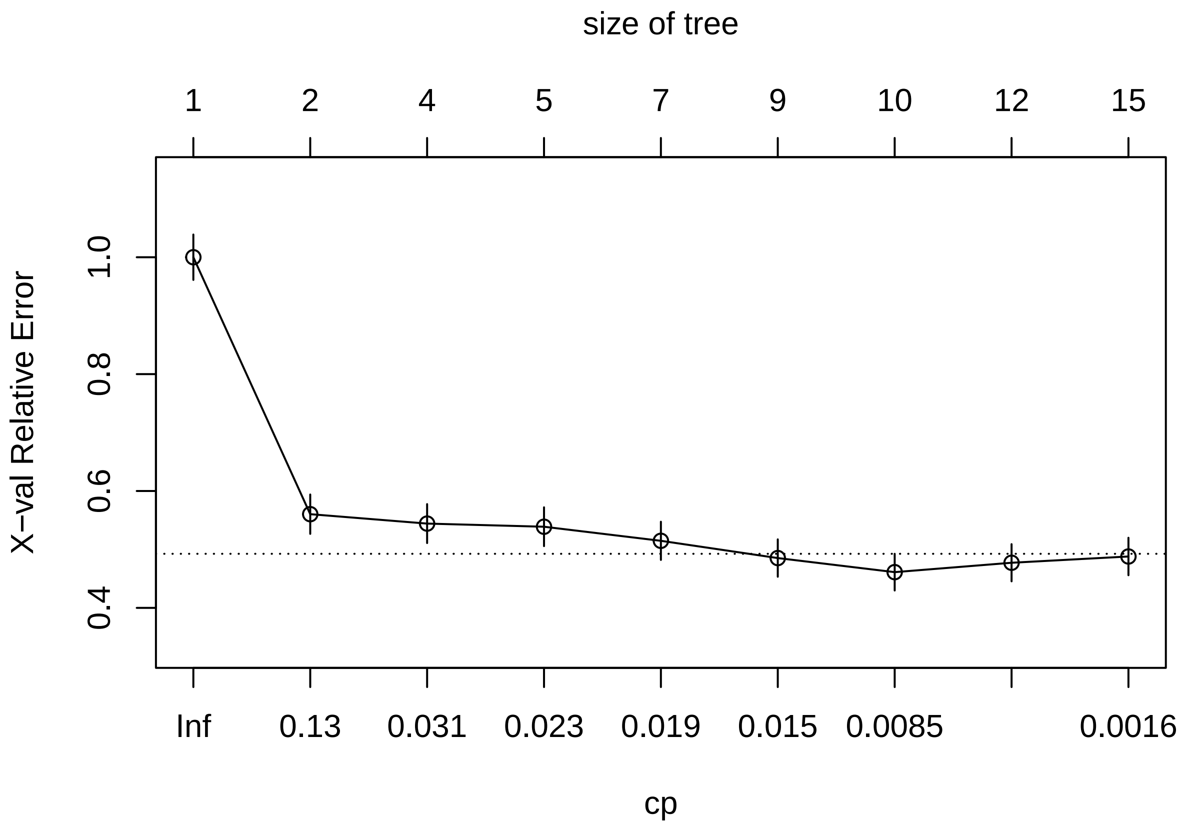


Pre pruning

```
#choosing cp
Ecom_Tree<-rpart(HeartDisease~., method="class",
                 control=rpart.control(minsplit=30,cp=0.001),data=train)
printcp(Ecom_Tree)
```

```
##
## Classification tree:
## rpart(formula = HeartDisease ~ ., data = train, method = "class",
##     control = rpart.control(minsplit = 30, cp = 0.001))
##
## Variables actually used in tree construction:
##  [1] Age               ChestPain_ASY     ChestPain_ATA     Cholesterol
##  [5] FastingBS         Is_Female         MaxHR             Oldpeak
##  [9] RestingBP         RestingECG_Normal ST_Slope_Up
##
## Root node error: 373/854 = 0.43677
##
## n= 854
##
##          CP nsplit rel error  xerror     xstd
## 1 0.4396783      0   1.00000 1.00000 0.038859
## 2 0.0402145      1   0.56032 0.56032 0.033683
## 3 0.0241287      3   0.47989 0.54424 0.033350
## 4 0.0227882      4   0.45576 0.53887 0.033237
## 5 0.0160858      6   0.41019 0.51475 0.032707
## 6 0.0134048      8   0.37802 0.48525 0.032019
## 7 0.0053619      9   0.36461 0.46113 0.031421
## 8 0.0026810     11   0.35389 0.47721 0.031823
## 9 0.0010000     14   0.34584 0.48794 0.032084
```

```
plotcp(Ecom_Tree)
```

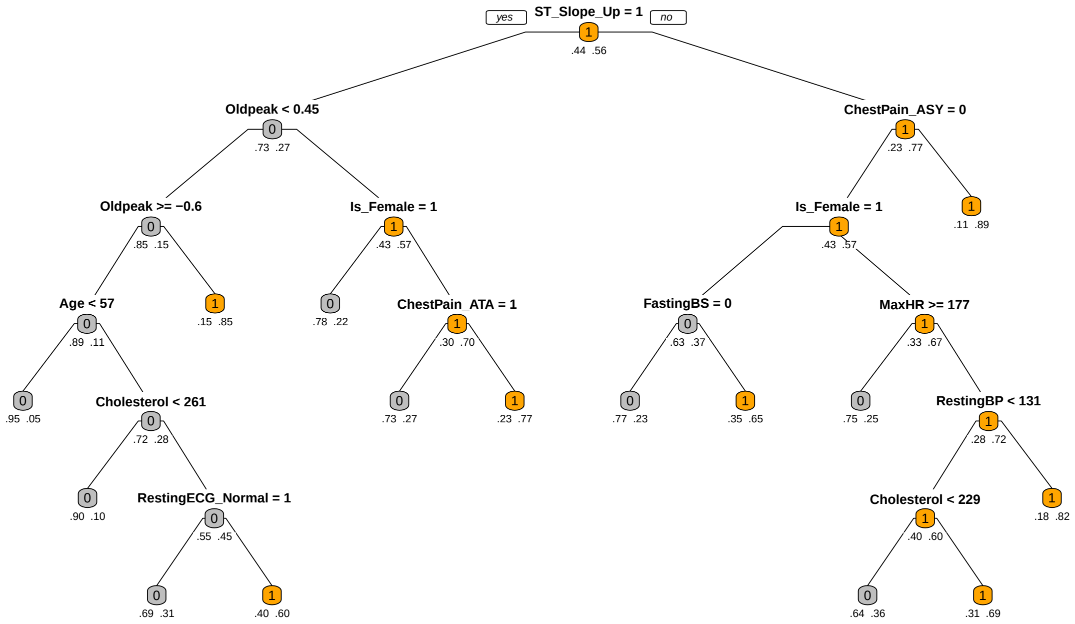


```
cp_econ <- data.table(Sample_tree[["cptable"]])
optimal_cp_econ <- mean(cp[xerror == min(xerror),CP])
optimal_cp_econ
```

```
## [1] 0.006255585
```

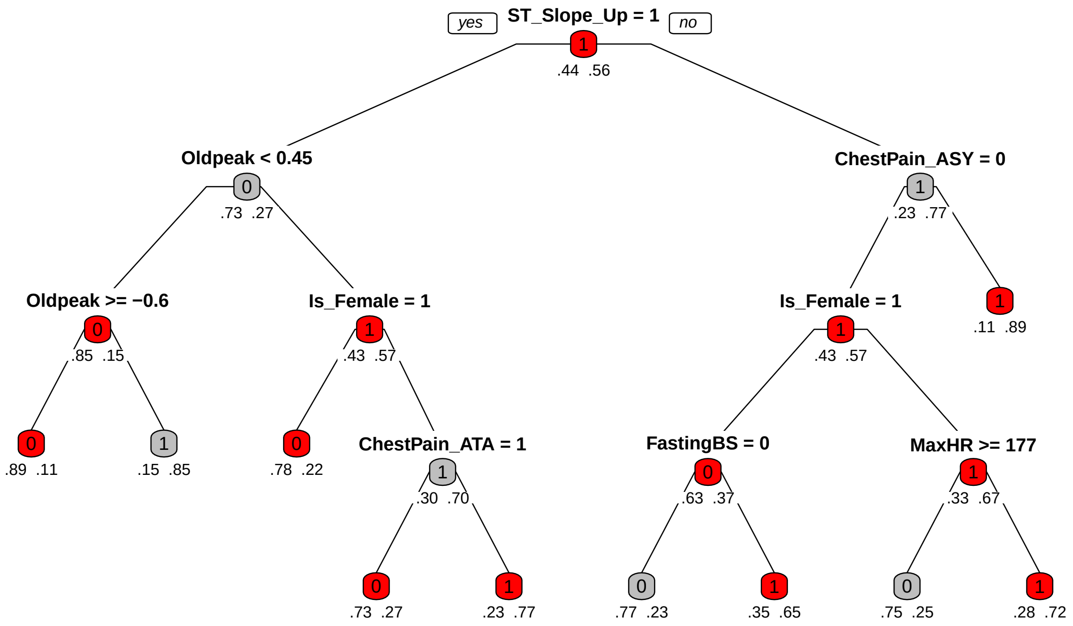
```
#Code to prune
Ecom_Tree_prune<-prune(Ecom_Tree,optimal_cp_econ)
#Plot before pruning
prp(Ecom_Tree,box.col=c("Grey", "Orange")[Ecom_Tree$frame$yval],
    varlen=0,faclen=0, type=1,extra=4,under=TRUE,main="TREE BEFORE PRUNING")
```

TREE BEFORE PRUNING



```
#Plot after pruning
prp(Ecom_Tree_prune,box.col=c("Grey", "red")[Sample_tree_1$frame$yval],
    varlen=0,faclen=0, type=1,extra=4,under=TRUE, main="TREE AFTER PRUNING")
```

TREE AFTER PRUNING



```r
# Choose the best tree model
pred_tree <- predict(tree_model,test, type="class")
MSE_tree <- mean((y_test - (as.numeric(as.character(pred_tree))))^2)

pred_Ecom <- predict(Ecom_Tree,test, type="class")
MSE_Ecom <- mean((y_test - (as.numeric(as.character(pred_Ecom))))^2)

predict_Ecom_prune <- predict(Ecom_Tree_prune, test, type="class")
MSE_Ecom_prune <- mean((y_test - (as.numeric(as.character(predict_Ecom_prune))))^2)
```

```r
MSE_Ecom < MSE_tree
```

```
## [1] TRUE
```

```r
MSE_Ecom_prune < MSE_Ecom
```

```
## [1] FALSE
```

```r
c(MSE_Ecom_prune, MSE_Ecom)
```

```
## [1] 0.2336449 0.2242991
```

As the pruned model has a higher MSE, we decide to use the Ecom_model.

```
# predict
predict_tree <- predict(Ecom_Tree, test, type="class")
table_mat <- table(test$HeartDisease, predict_tree)
table_mat
```

```
##    predict_tree
##       0   1
##   0  66  31
##   1  17 100
```

```
# Accuracy
accuracy_tree <- sum(diag(table_mat)) / sum(table_mat)
print(accuracy_tree)
```

```
## [1] 0.7757009
```

```
# MSE
with(test, table(predict_tree, HeartDisease))
```

```
##             HeartDisease
## predict_tree   0   1
##            0  66  17
##            1  31 100
```

```
MSE_tree<-(mean((as.numeric(as.character(predict_tree))-y_test)^2))
print(MSE_tree)
```

```
## [1] 0.2242991
```

```
# RMSE
RMSE_tree<-sqrt(mean((as.numeric(as.character(predict_tree))-y_test)^2))
print(RMSE_tree)
```

```
## [1] 0.4736022
```

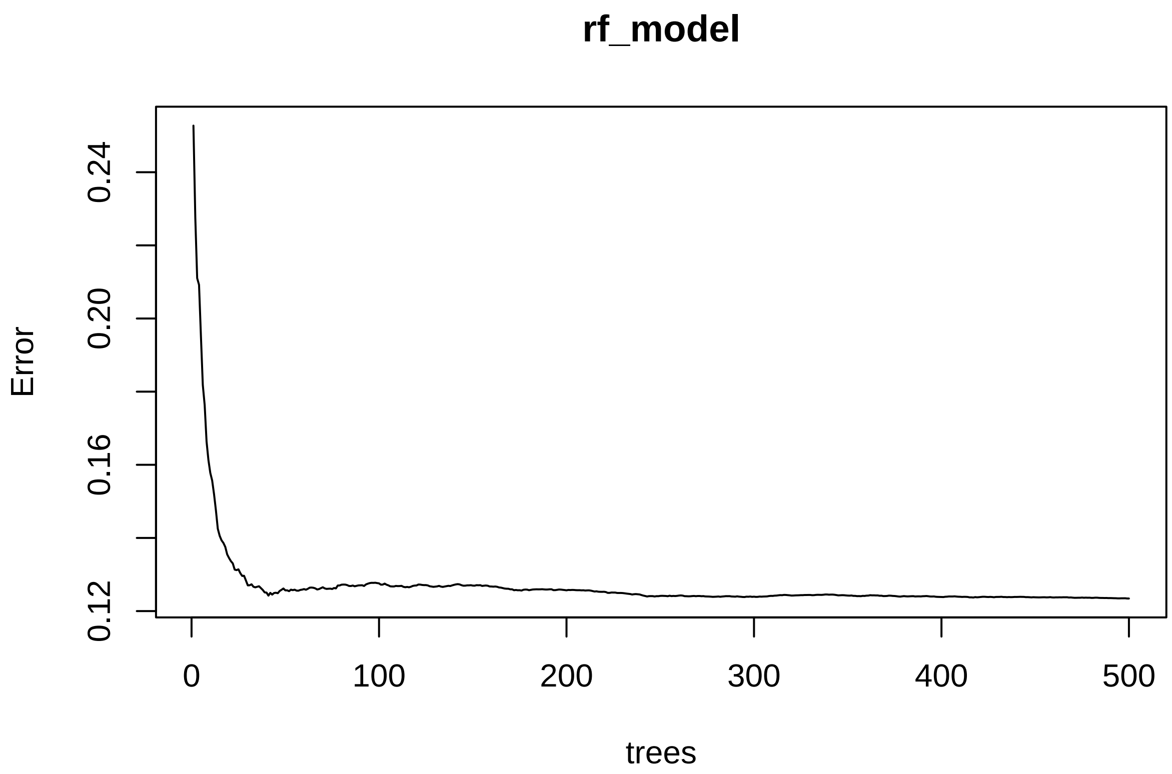
Random forest

We use a bagging random forest.

```
set.seed(123)
rf_model <- randomForest(HeartDisease~.,data=train, replace=T,
                         keep.forest = TRUE, keep.inbag=TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```r
plot(rf_model)
```



```r
# Calculate MSE, RMSE and accuracy for the original model
pred_rf0 <- predict(rf_model, newdata = x_test)
pred_rf0 <- ifelse(pred_rf0 > 0.5, 1, 0)
table_mat_rf0 <- table(test$HeartDisease, pred_rf0)
table_mat_rf0
```

```
##    pred_rf0
##       0   1
##   0  69  28
##   1  12 105
```

```r
## MSE
with(test, table(pred_rf0, HeartDisease))
```

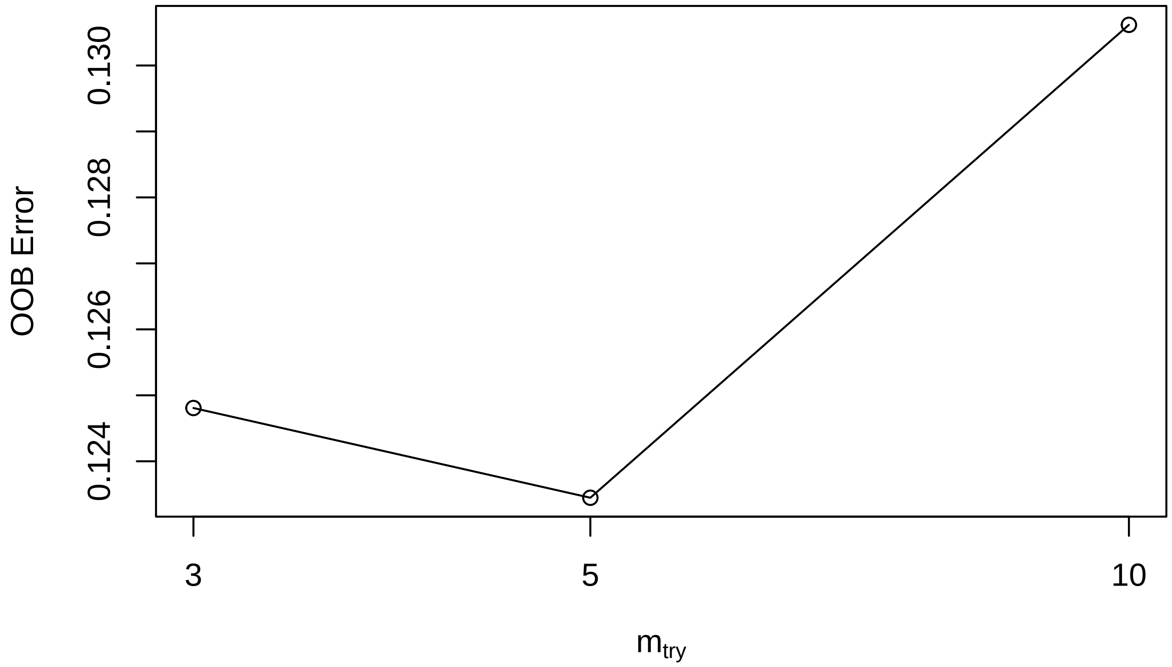
```
##          HeartDisease
## pred_rf0   0   1
##        0  69  12
##        1  28 105
```

```r
MSE_rf0<-(mean((as.numeric(as.character(pred_rf0))-y_test)^2))
## RMSE
```

```
RMSE_rf0<-sqrt(MSE_rf0)
## accuracy rate
accuracy_rf0 <- sum(diag(table_mat_rf0)) / sum(table_mat_rf0)
```

```
# Validation
# select mtry
set.seed(123)
mtry <- tuneRF(x_train,y_train, ntreeTry=500,
               stepFactor=2,improve=0.02,trace=FALSE,plot=TRUE)
```

```
## Warning in randomForest.default(x, y, mtry = mtryStart, ntree = ntreeTry, :
## The response has five or fewer unique values. Are you sure you want to do
## regression?

## Warning in rfout$mse/(var(y) * (n - 1)/n): Recycling array of length 1 in vector-array arithmetic is
##   Use c() or as.vector() instead.

## Warning in randomForest.default(x, y, mtry = mtryCur, ntree = ntreeTry, :
## The response has five or fewer unique values. Are you sure you want to do
## regression?

## Warning in rfout$mse/(var(y) * (n - 1)/n): Recycling array of length 1 in vector-array arithmetic is
##   Use c() or as.vector() instead.

## -0.01101859 0.02

## Warning in randomForest.default(x, y, mtry = mtryCur, ntree = ntreeTry, : The response has five or f

## Warning in randomForest.default(x, y, mtry = mtryCur, ntree = ntreeTry, : Recycling array of length
##   Use c() or as.vector() instead.

## -0.05807001 0.02
```



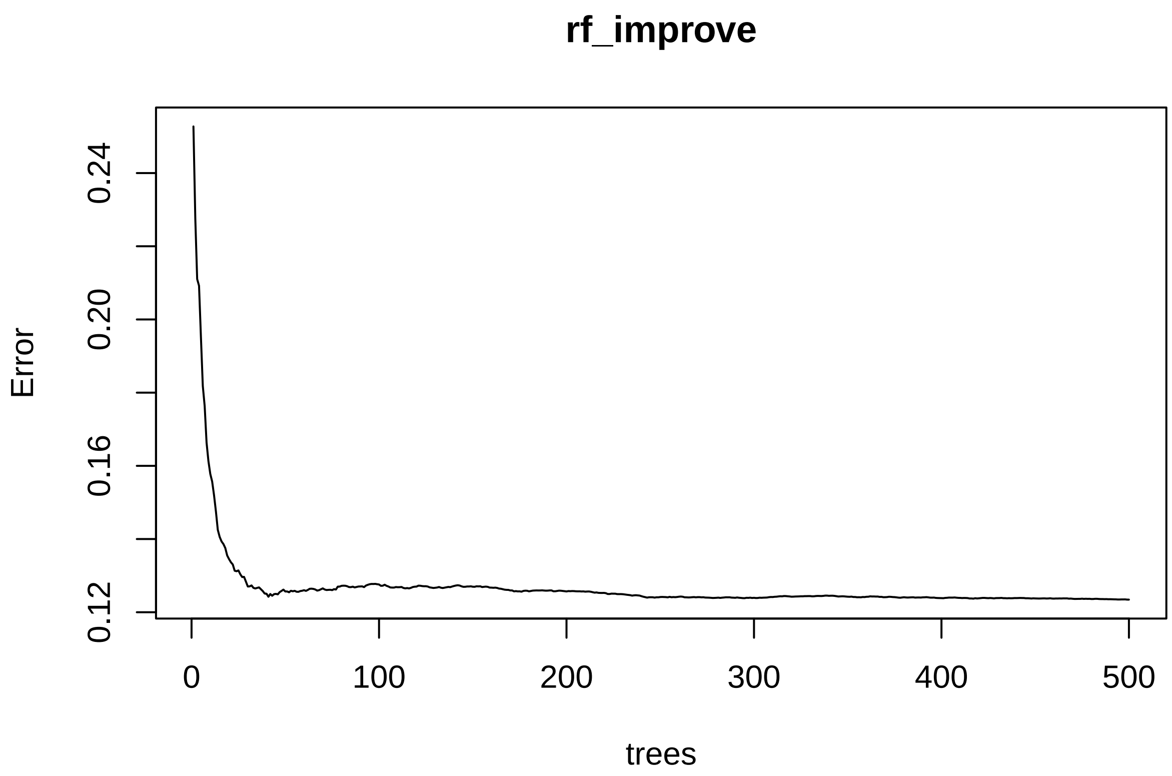
```r
set.seed(123)
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
# use the mtry that minimizes the oob error
rf_improve <- randomForest(HeartDisease~.,data=train, mtry=best.m,
                           replace=T,keep.forest = TRUE, keep.inbag=TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```r
print(rf_improve)
```

```
##
## Call:
##  randomForest(formula = HeartDisease ~ ., data = train, mtry = best.m,      replace = T, keep.forest
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.1234472
##                     % Var explained: 49.82
```

```r
plot(rf_improve)
```



```
# Evaluation
pred_rf <- predict(rf_improve, newdata=x_test)
pred_rf <- ifelse(pred_rf > 0.5, 1, 0)
table_mat_rf <- table(test$HeartDisease, pred_rf)
table_mat_rf
```

```
##    pred_rf
##       0   1
##   0  69  28
##   1  12 105
```

```
# MSE
with(test, table(pred_rf, HeartDisease))
```

```
##        HeartDisease
## pred_rf   0   1
##       0  69  12
##       1  28 105
```

```
MSE_rf<-(mean((as.numeric(as.character(pred_rf))-y_test)^2))
# RMSE
RMSE_rf<-sqrt(MSE_rf)
# Accuracy
accuracy_rf <- sum(diag(table_mat_rf)) / sum(table_mat_rf)
```

```r
# Compare
MSE_rf <= MSE_rf0
```

```
## [1] TRUE
```

```r
MSE_rf
```

```
## [1] 0.1869159
```

```r
MSE_rf0
```

```
## [1] 0.1869159
```

It turns out that the results of the original random forest and the improved random forest are identical.

```r
RMSE_rf == RMSE_rf0
```

```
## [1] TRUE
```

```r
accuracy_rf == accuracy_rf0
```

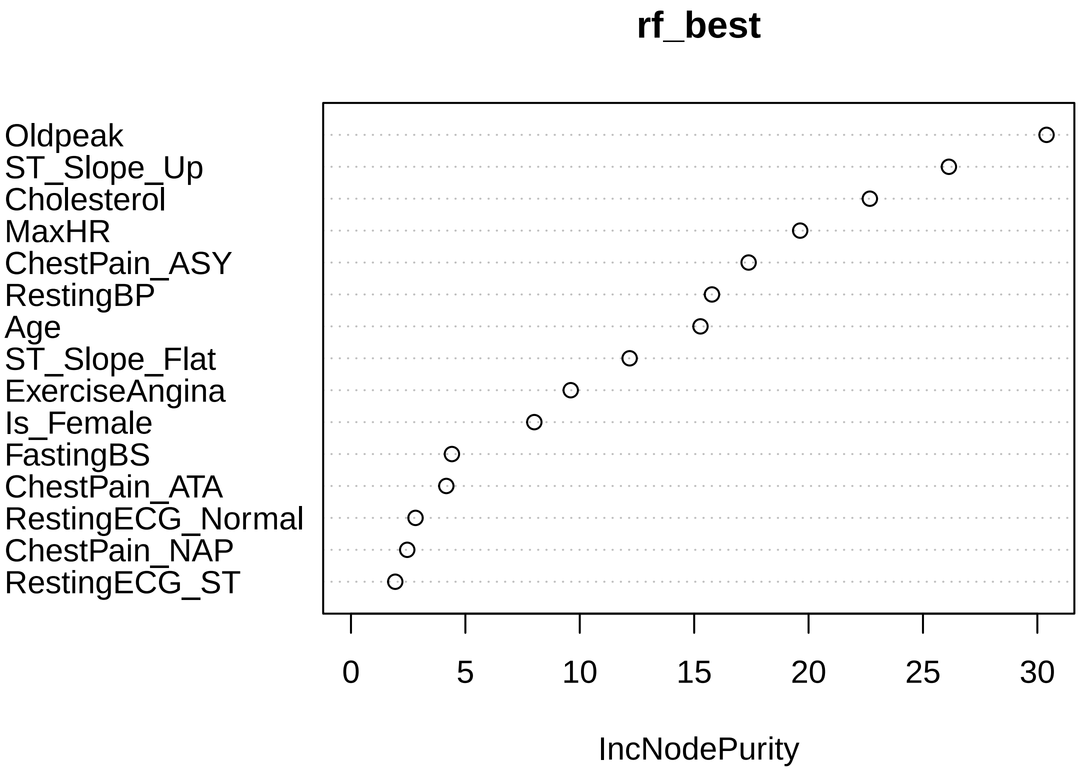
```
## [1] TRUE
```

We set the improved model as the best fit random forest.

```r
rf_best <- rf_improve
importance(rf_best)
```

```
##                    IncNodePurity
## Age                    15.274660
## RestingBP              15.777299
## Cholesterol            22.678859
## FastingBS               4.411565
## MaxHR                  19.633427
## ExerciseAngina          9.604314
## Oldpeak                30.401157
## ChestPain_ATA           4.166881
## ChestPain_NAP           2.458839
## ChestPain_ASY          17.380469
## Is_Female               8.012499
## RestingECG_Normal       2.820962
## RestingECG_ST           1.934376
## ST_Slope_Up            26.133635
## ST_Slope_Flat          12.179815
```

```r
varImpPlot(rf_best)
```



From the result, we can conclude that the most important factors to predict heart disease are: Oldpeak, ST_Slope_Up, Cholesterol, MaxHR, ChestPain_ASY, RestingBP, Age and ST_Slope_Flat.

```
MSE_rf_best <- MSE_rf
MSE_rf_best
```

```
## [1] 0.1869159
```

```
RMSE_rf_best <- RMSE_rf
RMSE_rf_best
```

```
## [1] 0.4323377
```

```
accuracy_rf_best <- accuracy_rf
accuracy_rf_best
```

```
## [1] 0.8130841
```

Model Comparison

We use MSE, RMSE and accuracy rate to compare the models.

```
# MSE for different models
c(mse_fw, lambda_min_mse_test_ridge,lambda_min_mse_test_lasso, MSE_tree, MSE_rf_best)
```

```
##                 s81       s28
## 0.2149533 0.1544109 0.1553257 0.2242991 0.1869159
```

```
# RMSE for different models
c(rmse_fw, sqrt(lambda_min_mse_test_ridge),sqrt(lambda_min_mse_test_lasso), RMSE_tree, RMSE_rf_best)
```

```
##                 s81       s28
## 0.4636305 0.3929515 0.3941138 0.4736022 0.4323377
```

```
# Accuracy rate
c(accuracy_fw, accuracy_ridge, accuracy_lasso, accuracy_tree, accuracy_rf_best)
```

```
## [1] 0.7850467 0.7943925 0.7990654 0.7757009 0.8130841
```

Conclusion

Best model

We could see that ridge regression has the lowest MSE and RMSE, and the decision tree has the highest accuracy rate. Thus, we refer the ridge regression as the best model in fitting, and decision tree as the best model in predicting. But we believe if we have a larger data set, the result will be more uniform. Also, the accuracy rates for all the models are around 80%, so we have a great outcome.

Limitation

1. Can not separate causal relationship. Like, ST_slope & ChestPainType might be the result of having heart disease; thus, they might not be used as predictors.

2. Because the best fit model is ridge regression, which could not perform subset selection, we cannot find out which variables bring greater impact through the best model, but can only get this kind of information through other models.

Challenge

1. Validation for each model.Because the validation methods for each model are not quite the same, we spent a lot of time studying how to validate the different models by reading textbook and searching for papers.

2. Pruning decision tree. We learned decision tree only a short time ago and we are not very familiar with pruning. We tried four methods of pruning the decision tree to find the best one and spent a lot of time on pruning logic and programming.

3. Visualization of random forest. We would love to be able to visualize the best fit random forest, just like a tree model. But we have not found a way to draw its roots. Hopefully, we can discover a way to visualize the random forest in the future.

4. Work Integration. Because we assign each person to complete a different model, the variable names have to be checked several times to make sure there is no duplication. Also, there are times when unexpected

bugs occur during integrating due to different versions of R or packages and different dataset's or value's names.

We found that learning concepts and code from the textbook and actually completing a project are very different things. Hope we will have more opportunities to practice in the future.

Practical Application

Find out significant factors

From the forward selection, we found eleven important predictors of heart disease: Age, Cholesterol, FastingBS, MaxHR, ExerciseAngina, Oldpeak, ChestPain_ATA, ChestPain_ASY, Is_Female, ST_Slope_Up, ST_Slope_Flat.

From random forest, we found the following indicators have higher importance: Oldpeak, ST_Slope_Up, Cholesterol, MaxHR, ChestPain_ASY, RestingBP, Age and ST_Slope_Flat.

We therefore believe that the following factors have a greater impact on heart disease: age, cholesterol, fasting blood sugar (FastingBS), exercise-induced angina (ExerciseAngina), the ratio of ST slope of the peak, exercise relative to rest (Oldpeak), chest pain type (especially ChestPain_ATA and ChestPain_ASY), gender (Is_Female), ST slope (ST_Slope_Up, ST_Slope_Flat), maximum heart rate achieved (MaxHR).

Remind the coefficients:

```r
coef(step.model$finalModel, 11)
```

```
##        (Intercept)                Age        Cholesterol          FastingBS
##       0.1772672422       0.0021612565       0.0006802975       0.1511175773
##              MaxHR ExerciseAnginaTRUE            Oldpeak  ChestPain_ATATRUE
##      -0.0009138245       0.1145940339       0.0530067107      -0.0533265889
##  ChestPain_ASYTRUE      Is_FemaleTRUE    ST_Slope_UpTRUE  ST_Slope_FlatTRUE
##       0.2187395871      -0.1638344216      -0.1386729931       0.1536800410
```

Thus, we have following suggestions.

1. Men and seniors should pay more attention to heart disease prevention because they are more likely to develop the disease.
2. The non-diseased population should undergo regular medical examinations and pay attention to the following indicators:

1) Cholesterol, when it is becoming high,
2) Fasting blood sugar, when it is becoming high,
3) Old peak, when it is becoming high,
4) ST slope, when it is flat or down.

3. For people who have angina when exercising, and who have chest pains from time to time, they need to consider the possibility of heart disease and go to the hospital as soon as possible for investigation.
4. For people who already have heart disease, pay attention to there cholesterol, fasting blood sugar and old peak. When these measures have abnormal elevation, go to the hospital as soon as possible.

Predict using the best model

For insurance company and other organizations who are concerned about their customer's physical condition, our model is helpful for predicting the possibility for customers of getting heart disease, or check for false information about heart disease in the information submitted by the client. For example, it is possible to predict the probability of a client getting heart disease based on the data in the medical examination report he/she submits. If the probability of getting heart disease is high but the client does not claim so, he/she can be asked to provide more specific information.