Project

Shaopeng Cheng, Augustus Williams

2025-5-10

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.3.3
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3

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

```r
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.3.3

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

```r
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-7
```

```r
library(r02pro)
```

```
## Warning: package 'r02pro' was built under R version 4.3.3
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(ISLR)
```

```r
heart <- read.csv("heart.csv")
```

Feature Selection

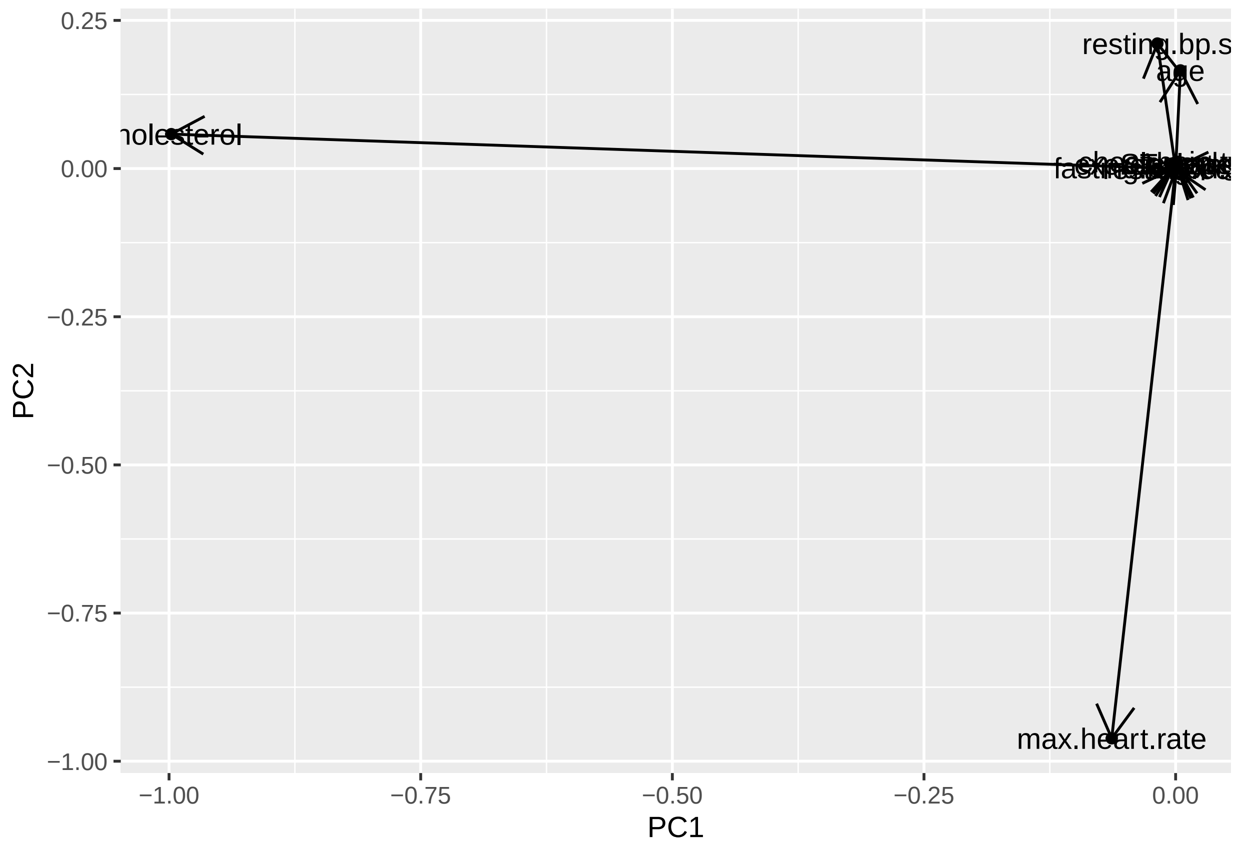
```r
# Forward Stepwise Selection with adjusted R^2
forward_fit <- regsubsets(target ~., data = heart, method = "forward", nvmax = 8)
forward_sum <- summary(forward_fit)
best_ind_for <- which.max(forward_sum$adjr2)
best_model_forward <- coef(forward_fit, best_ind_for)
best_model_forward
```

```
##         (Intercept)                 sex     chest.pain.type         cholesterol
##       -0.0745320857        0.1868783023        0.1191827974       -0.0003344296
## fasting.blood.sugar      max.heart.rate     exercise.angina             oldpeak
##        0.1295000003       -0.0021876808        0.1776171859        0.0600706989
##            ST.slope
##        0.1842095787
```

```r
# Backward Stepwise Selection with Cp
backward_fit <- regsubsets(target ~ ., data = heart, method = "backward", nvmax = 8)
backward_sum <- summary(backward_fit)
best_ind_back <- which.min(backward_sum$cp)
best_model_backward <- coef(backward_fit, best_ind_back)
best_model_backward
```

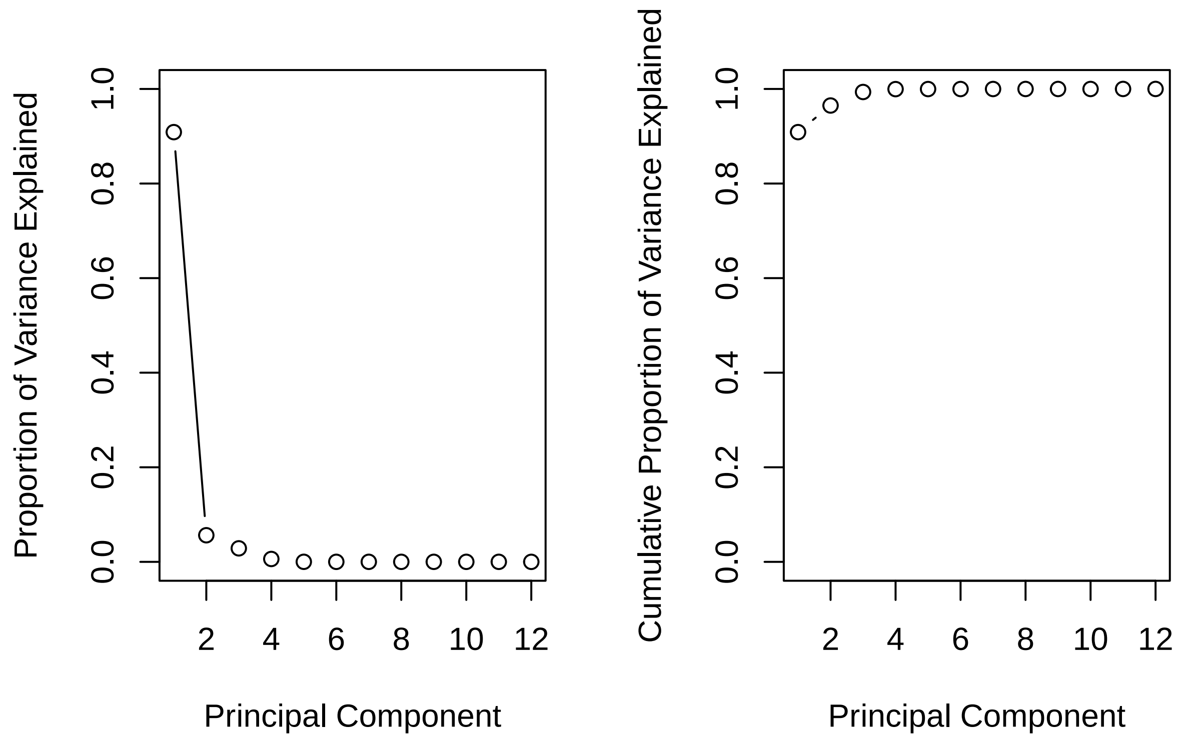
```
##         (Intercept)                 sex     chest.pain.type         cholesterol
##       -0.0745320857        0.1868783023        0.1191827974       -0.0003344296
## fasting.blood.sugar      max.heart.rate     exercise.angina             oldpeak
##        0.1295000003       -0.0021876808        0.1776171859        0.0600706989
##            ST.slope
##        0.1842095787
```

```r
# PCA selection
pr.out <- prcomp(heart, scale = FALSE)
pr.rot <- as.data.frame(pr.out$rotation)
ggplot(pr.rot, aes(x = PC1, y = PC2)) +
  geom_point() +
  geom_segment(aes(x = 0, y = 0, xend = PC1, yend = PC2),
               arrow = arrow(length = unit(0.5, "cm"))) +
  geom_text(label = rownames(pr.rot))
```



```
pr.var <- pr.out$sdev^2
pve <- pr.var / sum(pr.var)

cum_var <- cumsum(pve)
n_components <- which(cum_var >= 0.8)[1]
```

```
par(mfrow = c(1, 2))
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0, 1), type
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Explained", yl
```



```
names(heart)
```

```
##  [1] "age"                 "sex"                 "chest.pain.type"
##  [4] "resting.bp.s"        "cholesterol"         "fasting.blood.sugar"
##  [7] "resting.ecg"         "max.heart.rate"      "exercise.angina"
## [10] "oldpeak"             "ST.slope"            "target"
```

Split to train and test data

```
library(dplyr)
set.seed(1)
```

```r
heart <- dplyr::select(heart,
  `chest.pain.type`, `cholesterol`, `fasting.blood.sugar`,
  `max.heart.rate`, `exercise.angina`, `oldpeak`, `ST.slope`, `target`
)
tr_ind <- sample(1:nrow(heart), 0.8 * nrow(heart))
heart_train <- heart[tr_ind, ]
heart_test <- heart[-tr_ind, ]
```

Logistic Regression

```r
logistic_model <- glm(target ~., data = heart_train, family = "binomial")

summary(logistic_model)
```

```
##
## Call:
## glm(formula = target ~ ., family = "binomial", data = heart_train)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.2204080  0.7864720  -2.823 0.004754 **
## chest.pain.type     0.7356541  0.1022968   7.191 6.41e-13 ***
## cholesterol        -0.0033467  0.0009444  -3.544 0.000395 ***
## fasting.blood.sugar  0.8008350  0.2264151   3.537 0.000405 ***
## max.heart.rate     -0.0150355  0.0039974  -3.761 0.000169 ***
## exercise.angina     1.0956566  0.2047066   5.352 8.68e-08 ***
## oldpeak             0.4709970  0.1042426   4.518 6.23e-06 ***
## ST.slope            1.1197392  0.1764426   6.346 2.21e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1318.24  on 951  degrees of freedom
## Residual deviance:  791.39  on 944  degrees of freedom
## AIC: 807.39
##
## Number of Fisher Scoring iterations: 5
```

```r
predict_train_prob <- predict(logistic_model, type = "response")
predict_train_label <- ifelse(predict_train_prob > 0.5, "1", "0")
train_error <- mean(predict_train_label != heart_train$target)
print(train_error)
```

```
## [1] 0.1764706
```

```r
predict_test_prob <- predict(logistic_model, newdata = heart_test, type = "response")
predict_test_label <- ifelse(predict_test_prob > 0.5, "1", "0")
test_error <- mean(predict_test_label != heart_test$target)
print(test_error)
```

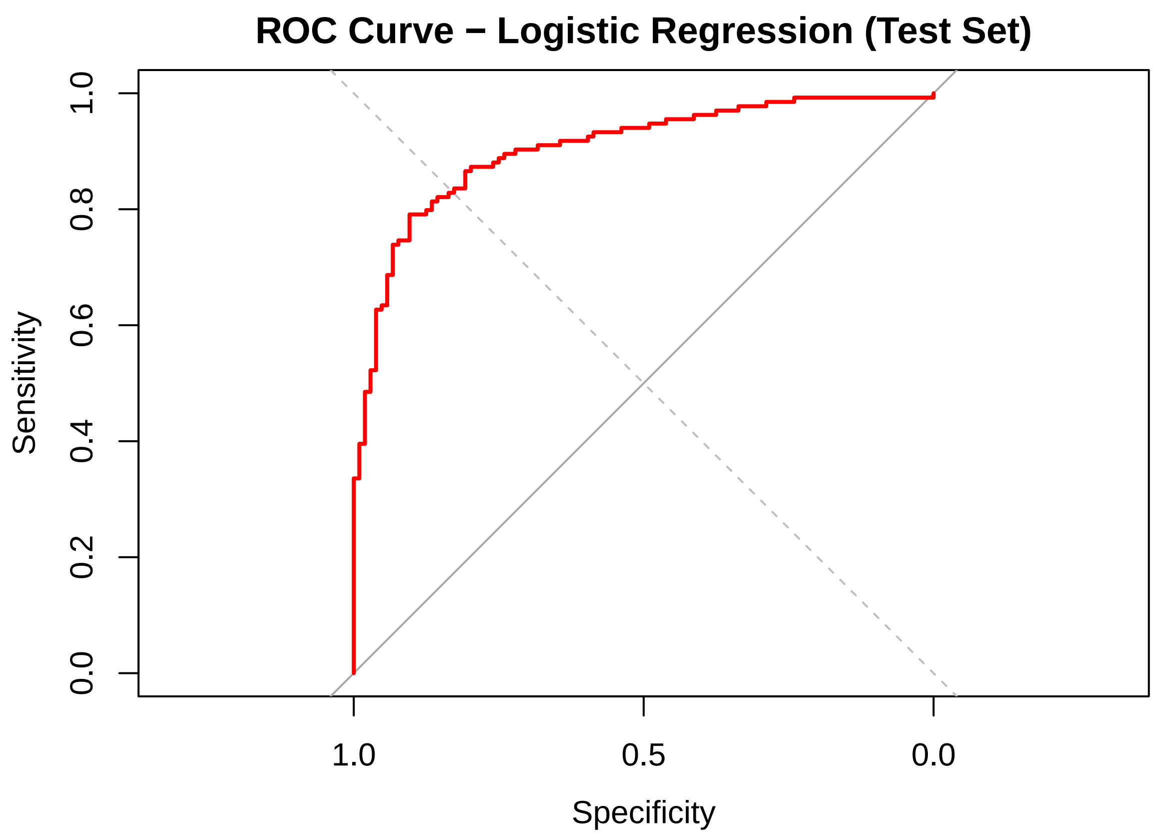
```
## [1] 0.1722689
```

```
# aucroc
roc_logistic <- roc(heart_test$target, predict_test_prob)
```

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

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

```
auc_logistic <- auc(roc_logistic)
plot(roc_logistic, col = "red", lwd = 2, main = "ROC Curve - Logistic Regression (Test Set)")
abline(a = 0, b = 1, lty = 2, col = "gray")
```



```
# cross-validation
set.seed(1)

K <- 5
n_all <- nrow(heart)
fold_ind <- sample(1:K, n_all, replace = TRUE)
error_lr <- mean(sapply(1:K, function(j){
    fit <- glm(target ~ ., data = heart[fold_ind != j, ],
               family = "binomial")
    pred_prob <- predict(fit, newdata = heart[fold_ind == j, ], type = "response")
    pred_label <- ifelse(pred_prob > 0.5, "1", "0")
```

```
  mean(heart$target[fold_ind == j] != pred_label)
  }))
error_lr
```

```
## [1] 0.1786873
```

```
set.seed(1)

X_train <- heart_train[, -which(names(heart_train) == "target")]
Y_train <- heart_train$target
X_test <- heart_test[, -which(names(heart_test) == "target")]
Y_test <- heart_test$target
Y_test <- as.factor(Y_test)

# Set up the trainControl for 5-fold cross-validation
ctrl <- trainControl(method = "cv", number = 5)

# Define the tuning grid for alpha and lambda
grid <- expand.grid(alpha = seq(0, 1, by = 0.1), lambda = seq(0.05, 0.1, by = 0.002))

#Perform 5-fold cross-validation to tune hyperparameters
logi_reg_model <- train(x = X_test,y = Y_test,method = "glmnet",trControl = ctrl,
                        tuneGrid = grid,metric = "Accuracy")
print(logi_reg_model)
```

```
## glmnet
##
## 238 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 190, 191, 191, 190, 190
## Resampling results across tuning parameters:
##
##   alpha  lambda  Accuracy   Kappa
##   0.0    0.050   0.8277482  0.6494130
##   0.0    0.052   0.8277482  0.6494130
##   0.0    0.054   0.8277482  0.6494130
##   0.0    0.056   0.8277482  0.6494130
##   0.0    0.058   0.8277482  0.6494130
##   0.0    0.060   0.8277482  0.6494130
##   0.0    0.062   0.8277482  0.6494130
##   0.0    0.064   0.8277482  0.6494130
##   0.0    0.066   0.8277482  0.6494130
##   0.0    0.068   0.8277482  0.6494130
##   0.0    0.070   0.8234929  0.6411552
##   0.0    0.072   0.8234929  0.6411552
##   0.0    0.074   0.8193262  0.6331673
##   0.0    0.076   0.8193262  0.6331673
##   0.0    0.078   0.8193262  0.6331673
##   0.0    0.080   0.8193262  0.6331673
##   0.0    0.082   0.8193262  0.6331673
##   0.0    0.084   0.8193262  0.6331673
##   0.0    0.086   0.8193262  0.6331673
##   0.0    0.088   0.8193262  0.6331673
##   0.0    0.090   0.8193262  0.6331673
##   0.0    0.092   0.8193262  0.6331673
##   0.0    0.094   0.8193262  0.6331673
##   0.0    0.096   0.8193262  0.6331673
##   0.0    0.098   0.8234929  0.6413138
##   0.0    0.100   0.8234929  0.6413138
##   0.1    0.050   0.8362589  0.6671656
##   0.1    0.052   0.8362589  0.6671656
##   0.1    0.054   0.8362589  0.6671656
##   0.1    0.056   0.8362589  0.6671656
##   0.1    0.058   0.8362589  0.6671656
##   0.1    0.060   0.8320035  0.6578168
##   0.1    0.062   0.8320035  0.6578168
##   0.1    0.064   0.8320035  0.6578168
##   0.1    0.066   0.8320035  0.6578168
##   0.1    0.068   0.8320035  0.6578168
##   0.1    0.070   0.8320035  0.6578168
##   0.1    0.072   0.8320035  0.6578168
##   0.1    0.074   0.8320035  0.6578168
##   0.1    0.076   0.8320035  0.6578168
##   0.1    0.078   0.8278369  0.6498289
##   0.1    0.080   0.8278369  0.6498289
##   0.1    0.082   0.8278369  0.6498289
##   0.1    0.084   0.8278369  0.6498289
##   0.1    0.086   0.8278369  0.6498289
##   0.1    0.088   0.8278369  0.6498289
##   0.1    0.090   0.8278369  0.6498289
##   0.1    0.092   0.8278369  0.6498289
##   0.1    0.094   0.8278369  0.6498289
##   0.1    0.096   0.8278369  0.6498289
##   0.1    0.098   0.8278369  0.6498289
##   0.1    0.100   0.8278369  0.6498289
##   0.2    0.050   0.8277482  0.6489341
##   0.2    0.052   0.8277482  0.6489341
##   0.2    0.054   0.8277482  0.6489341
##   0.2    0.056   0.8277482  0.6489341
##   0.2    0.058   0.8277482  0.6489341
##   0.2    0.060   0.8277482  0.6489341
##   0.2    0.062   0.8277482  0.6489341
##   0.2    0.064   0.8320035  0.6578168
##   0.2    0.066   0.8320035  0.6578168
##   0.2    0.068   0.8320035  0.6578168
##   0.2    0.070   0.8320035  0.6578168
##   0.2    0.072   0.8277482  0.6483427
##   0.2    0.074   0.8277482  0.6483427
##   0.2    0.076   0.8277482  0.6483427
##   0.2    0.078   0.8277482  0.6483427
##   0.2    0.080   0.8277482  0.6483427
##   0.2    0.082   0.8277482  0.6483427
##   0.2    0.084   0.8277482  0.6483427
##   0.2    0.086   0.8235816  0.6403548
##   0.2    0.088   0.8235816  0.6403548
##   0.2    0.090   0.8235816  0.6403548
##   0.2    0.092   0.8235816  0.6403548
##   0.2    0.094   0.8235816  0.6403548
##   0.2    0.096   0.8235816  0.6403548
##   0.2    0.098   0.8235816  0.6403548
##   0.2    0.100   0.8235816  0.6403548
##   0.3    0.050   0.8277482  0.6489341
##   0.3    0.052   0.8277482  0.6489341
##   0.3    0.054   0.8277482  0.6489341
##   0.3    0.056   0.8277482  0.6489341
##   0.3    0.058   0.8277482  0.6489341
##   0.3    0.060   0.8277482  0.6489341
##   0.3    0.062   0.8277482  0.6489341
##   0.3    0.064   0.8277482  0.6489341
##   0.3    0.066   0.8277482  0.6489341
##   0.3    0.068   0.8277482  0.6489341
##   0.3    0.070   0.8277482  0.6489341
##   0.3    0.072   0.8277482  0.6489341
##   0.3    0.074   0.8277482  0.6489341
##   0.3    0.076   0.8277482  0.6489341
##   0.3    0.078   0.8277482  0.6489341
##   0.3    0.080   0.8277482  0.6489341
##   0.3    0.082   0.8277482  0.6489341
##   0.3    0.084   0.8277482  0.6489341
##   0.3    0.086   0.8277482  0.6489341
##   0.3    0.088   0.8277482  0.6489341
##   0.3    0.090   0.8193262  0.6314721
##   0.3    0.092   0.8151596  0.6235666
##   0.3    0.094   0.8151596  0.6235666
##   0.3    0.096   0.8151596  0.6235666
##   0.3    0.098   0.8194149  0.6324493
##   0.3    0.100   0.8194149  0.6324493
##   0.4    0.050   0.8277482  0.6489341
##   0.4    0.052   0.8277482  0.6489341
##   0.4    0.054   0.8277482  0.6489341
##   0.4    0.056   0.8277482  0.6489341
##   0.4    0.058   0.8277482  0.6489341
##   0.4    0.060   0.8277482  0.6489341
##   0.4    0.062   0.8235816  0.6409462
##   0.4    0.064   0.8235816  0.6409462
##   0.4    0.066   0.8235816  0.6409462
##   0.4    0.068   0.8235816  0.6409462
##   0.4    0.070   0.8235816  0.6409462
##   0.4    0.072   0.8235816  0.6409462
##   0.4    0.074   0.8235816  0.6409462
##   0.4    0.076   0.8235816  0.6409462
##   0.4    0.078   0.8235816  0.6409462
##   0.4    0.080   0.8235816  0.6409462
##   0.4    0.082   0.8235816  0.6409462
##   0.4    0.084   0.8235816  0.6409462
##   0.4    0.086   0.8235816  0.6409462
##   0.4    0.088   0.8235816  0.6409462
##   0.4    0.090   0.8235816  0.6409462
##   0.4    0.092   0.8234929  0.6400464
##   0.4    0.094   0.8234929  0.6400464
##   0.4    0.096   0.8234929  0.6400464
##   0.4    0.098   0.8234929  0.6400464
##   0.4    0.100   0.8234929  0.6400464
##   0.5    0.050   0.8277482  0.6490108
##   0.5    0.052   0.8277482  0.6490108
##   0.5    0.054   0.8277482  0.6490108
##   0.5    0.056   0.8277482  0.6490108
##   0.5    0.058   0.8277482  0.6490108
##   0.5    0.060   0.8277482  0.6490108
##   0.5    0.062   0.8277482  0.6490108
##   0.5    0.064   0.8277482  0.6490108
##   0.5    0.066   0.8277482  0.6490108
##   0.5    0.068   0.8277482  0.6490108
##   0.5    0.070   0.8277482  0.6490108
##   0.5    0.072   0.8277482  0.6490108
##   0.5    0.074   0.8277482  0.6490108
##   0.5    0.076   0.8234929  0.6400464
##   0.5    0.078   0.8234929  0.6400464
##   0.5    0.080   0.8234929  0.6400464
##   0.5    0.082   0.8234929  0.6400464
##   0.5    0.084   0.8234929  0.6400464
##   0.5    0.086   0.8234929  0.6400464
##   0.5    0.088   0.8234929  0.6400464
##   0.5    0.090   0.8234929  0.6400464
##   0.5    0.092   0.8277482  0.6490108
##   0.5    0.094   0.8277482  0.6490108
##   0.5    0.096   0.8277482  0.6490108
##   0.5    0.098   0.8277482  0.6490108
##   0.5    0.100   0.8277482  0.6490108
##   0.6    0.050   0.8235816  0.6409462
##   0.6    0.052   0.8235816  0.6409462
##   0.6    0.054   0.8235816  0.6409462
##   0.6    0.056   0.8235816  0.6409462
##   0.6    0.058   0.8235816  0.6409462
##   0.6    0.060   0.8235816  0.6409462
##   0.6    0.062   0.8235816  0.6409462
##   0.6    0.064   0.8193262  0.6319818
##   0.6    0.066   0.8193262  0.6319818
##   0.6    0.068   0.8193262  0.6319818
##   0.6    0.070   0.8193262  0.6319818
##   0.6    0.072   0.8193262  0.6319818
##   0.6    0.074   0.8193262  0.6319818
##   0.6    0.076   0.8193262  0.6319818
##   0.6    0.078   0.8193262  0.6319818
##   0.6    0.080   0.8193262  0.6319818
##   0.6    0.082   0.8193262  0.6319818
##   0.6    0.084   0.8235816  0.6409462
##   0.6    0.086   0.8235816  0.6409462
##   0.6    0.088   0.8235816  0.6409462
##   0.6    0.090   0.8235816  0.6409462
##   0.6    0.092   0.8235816  0.6409462
##   0.6    0.094   0.8235816  0.6409462
##   0.6    0.096   0.8235816  0.6409462
##   0.6    0.098   0.8235816  0.6409462
##   0.6    0.100   0.8194149  0.6319804
##   0.7    0.050   0.8235816  0.6409462
##   0.7    0.052   0.8235816  0.6409462
##   0.7    0.054   0.8193262  0.6319818
##   0.7    0.056   0.8193262  0.6319818
##   0.7    0.058   0.8193262  0.6319818
##   0.7    0.060   0.8193262  0.6319818
##   0.7    0.062   0.8193262  0.6319818
##   0.7    0.064   0.8193262  0.6319818
##   0.7    0.066   0.8193262  0.6319818
##   0.7    0.068   0.8193262  0.6319818
##   0.7    0.070   0.8193262  0.6319818
##   0.7    0.072   0.8193262  0.6319818
##   0.7    0.074   0.8193262  0.6319818
##   0.7    0.076   0.8235816  0.6409462
##   0.7    0.078   0.8235816  0.6409462
##   0.7    0.080   0.8194149  0.6319804
##   0.7    0.082   0.8152482  0.6231138
##   0.7    0.084   0.8152482  0.6231138
##   0.7    0.086   0.8152482  0.6231138
##   0.7    0.088   0.8152482  0.6231138
##   0.7    0.090   0.8194149  0.6319804
##   0.7    0.092   0.8194149  0.6319804
##   0.7    0.094   0.8194149  0.6319804
##   0.7    0.096   0.8194149  0.6319804
##   0.7    0.098   0.8194149  0.6319804
##   0.7    0.100   0.8194149  0.6319804
##   0.8    0.050   0.8193262  0.6319818
##   0.8    0.052   0.8193262  0.6319818
##   0.8    0.054   0.8193262  0.6319818
##   0.8    0.056   0.8193262  0.6319818
##   0.8    0.058   0.8193262  0.6319818
##   0.8    0.060   0.8193262  0.6319818
##   0.8    0.062   0.8193262  0.6319818
##   0.8    0.064   0.8193262  0.6319818
##   0.8    0.066   0.8151596  0.6230160
##   0.8    0.068   0.8151596  0.6230160
##   0.8    0.070   0.8152482  0.6231138
##   0.8    0.072   0.8152482  0.6231138
##   0.8    0.074   0.8152482  0.6231138
##   0.8    0.076   0.8152482  0.6231138
##   0.8    0.078   0.8152482  0.6231138
##   0.8    0.080   0.8194149  0.6319804
##   0.8    0.082   0.8194149  0.6319804
##   0.8    0.084   0.8194149  0.6319804
##   0.8    0.086   0.8194149  0.6319804
##   0.8    0.088   0.8194149  0.6319804
##   0.8    0.090   0.8194149  0.6319804
##   0.8    0.092   0.8194149  0.6319804
##   0.8    0.094   0.8194149  0.6319804
##   0.8    0.096   0.8194149  0.6319804
##   0.8    0.098   0.8194149  0.6319804
##   0.8    0.100   0.8194149  0.6319804
##   0.9    0.050   0.8193262  0.6319818
##   0.9    0.052   0.8193262  0.6319818
##   0.9    0.054   0.8193262  0.6319818
##   0.9    0.056   0.8151596  0.6230160
##   0.9    0.058   0.8151596  0.6230160
##   0.9    0.060   0.8109929  0.6141494
##   0.9    0.062   0.8109929  0.6141494
##   0.9    0.064   0.8109929  0.6141494
##   0.9    0.066   0.8152482  0.6231138
##   0.9    0.068   0.8152482  0.6231138
##   0.9    0.070   0.8152482  0.6231138
##   0.9    0.072   0.8194149  0.6319804
##   0.9    0.074   0.8194149  0.6319804
##   0.9    0.076   0.8194149  0.6319804
##   0.9    0.078   0.8194149  0.6319804
##   0.9    0.080   0.8194149  0.6319804
##   0.9    0.082   0.8194149  0.6319804
##   0.9    0.084   0.8194149  0.6319804
##   0.9    0.086   0.8194149  0.6319804
##   0.9    0.088   0.8194149  0.6319804
##   0.9    0.090   0.8194149  0.6319804
##   0.9    0.092   0.8152482  0.6233017
##   0.9    0.094   0.8152482  0.6233017
##   0.9    0.096   0.8152482  0.6233017
##   0.9    0.098   0.8152482  0.6233017
##   0.9    0.100   0.8152482  0.6233017
##   1.0    0.050   0.8151596  0.6230160
##   1.0    0.052   0.8109929  0.6141494
##   1.0    0.054   0.8109929  0.6141494
##   1.0    0.056   0.8109929  0.6141494
##   1.0    0.058   0.8109929  0.6141494
##   1.0    0.060   0.8109929  0.6141494
##   1.0    0.062   0.8152482  0.6231138
##   1.0    0.064   0.8152482  0.6231138
##   1.0    0.066   0.8194149  0.6319804
##   1.0    0.068   0.8194149  0.6319804
##   1.0    0.070   0.8194149  0.6319804
##   1.0    0.072   0.8194149  0.6319804
##   1.0    0.074   0.8194149  0.6319804
##   1.0    0.076   0.8152482  0.6233017
##   1.0    0.078   0.8152482  0.6233017
##   1.0    0.080   0.8152482  0.6233017
##   1.0    0.082   0.8152482  0.6233017
##   1.0    0.084   0.8152482  0.6233017
##   1.0    0.086   0.8152482  0.6233017
##   1.0    0.088   0.8152482  0.6233017
##   1.0    0.090   0.8152482  0.6233017
##   1.0    0.092   0.8152482  0.6233017
##   1.0    0.094   0.8152482  0.6233017
##   1.0    0.096   0.8111702  0.6145203
##   1.0    0.098   0.8070035  0.6053661
##   1.0    0.100   0.8070035  0.6053661
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.058.
```

```r
logi_reg_model$bestTune
```

```
##    alpha lambda
## 31   0.1  0.058
```

```r
# training error
train_predict_logiregu <- predict(logi_reg_model, X_train)
train_error_logiregu <- mean(train_predict_logiregu != Y_train)
train_error_logiregu
```

```
## [1] 0.1838235
```

```r
# test error
test_predict_logiregu <- predict(logi_reg_model, X_test)
test_error_logiregu <- mean(test_predict_logiregu != Y_test)
test_error_logiregu
```

```
## [1] 0.1554622
```

```r
# rocauc
roc_logiregu <- roc(Y_test, as.numeric(test_predict_logiregu), levels = rev(levels(Y_test)))
```

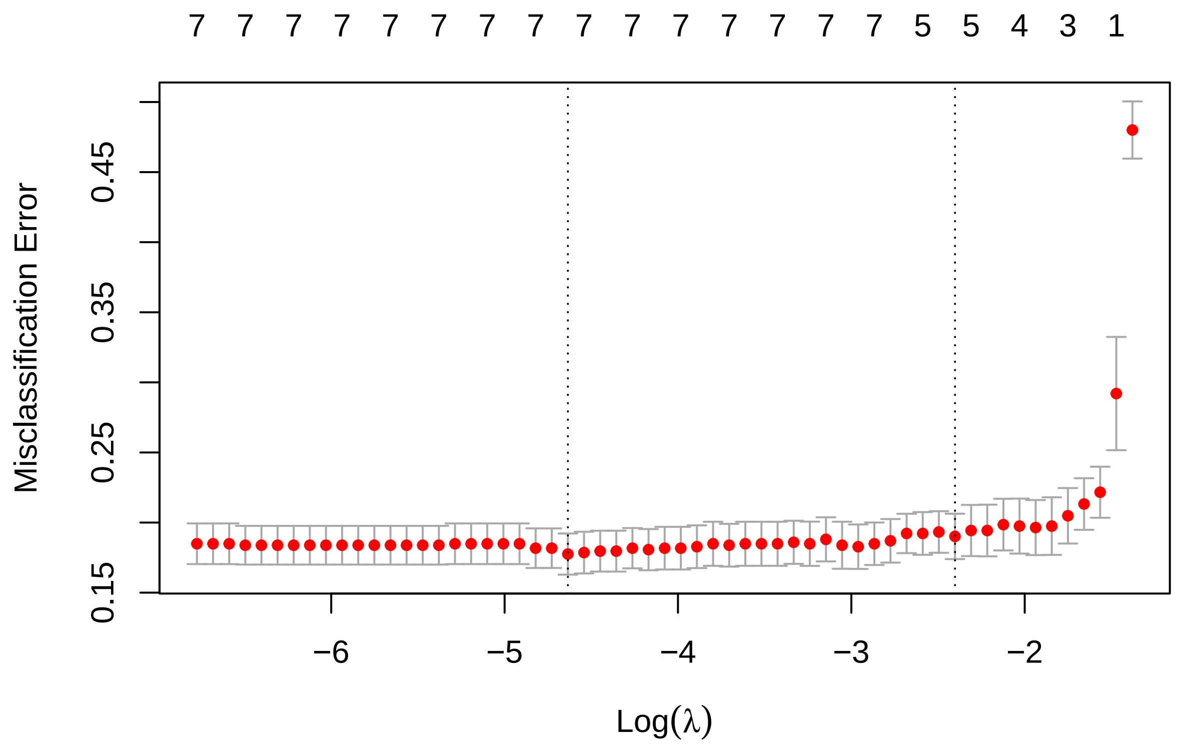
```
## Setting direction: controls > cases
```

```r
auc_logiregu <- auc(roc_logiregu)
auc_logiregu
```

```
## Area under the curve: 0.8415
```

Lasso Regression

```r
set.seed(1)

x_train <- model.matrix(target ~ ., data = heart_train)[, -1]
y_train <- heart_train$target
x_test  <- model.matrix(target ~ ., data = heart_test)[, -1]
y_test  <- heart_test$target

cv_fit <- cv.glmnet(x_train, y_train, alpha = 1, family = "binomial", type.measure = "class", nfolds = 
plot(cv_fit)
```



```
best_lambda <- cv_fit$lambda.min
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 0.009707011
```

```
lasso_model <- glmnet(x_train, y_train, family="binomial", alpha=1, lambda=best_lambda)
```

```
pred_prob_train <- predict(lasso_model, newx = x_train, type = "response")
pred_prob_test  <- predict(lasso_model, newx = x_test, type = "response")

pred_class_train <- ifelse(pred_prob_train > 0.5, "1", "0")
pred_class_test  <- ifelse(pred_prob_test > 0.5, "1", "0")

train_error_lasso <- mean((pred_class_train != y_train)^2)
test_error_lasso  <- mean((pred_class_test  != y_test)^2)

print(train_error_lasso)
```

```
## [1] 0.1722689
```

```
print(test_error_lasso)
```

```
## [1] 0.1722689
```

```r
# cross-validation
set.seed(1)

K <- 5
n_all <- nrow(heart)
fold_ind <- sample(1:K, n_all, replace = TRUE)
x <- model.matrix(target ~ ., data = heart)[, -1]
y <- heart$target
error_lasso <- mean(sapply(1:K, function(j) {
  # Training and validation split
  x_train <- x[fold_ind != j, ]
  y_train <- y[fold_ind != j]
  x_valid <- x[fold_ind == j, ]
  y_valid <- y[fold_ind == j]

  # Fit LASSO model with CV on training set to get best lambda
  cv_fit <- cv.glmnet(x_train, y_train, family = "binomial", alpha = 1, type.measure = "class")
  best_lambda <- cv_fit$lambda.min

  # Predict on validation fold
  pred_prob <- predict(cv_fit, newx = x_valid, s = best_lambda, type = "response")
  pred_label <- ifelse(pred_prob > 0.5, 1, 0)

  # Misclassification error
  mean(y_valid != pred_label)
}))
error_lasso
```

```
## [1] 0.1745477
```

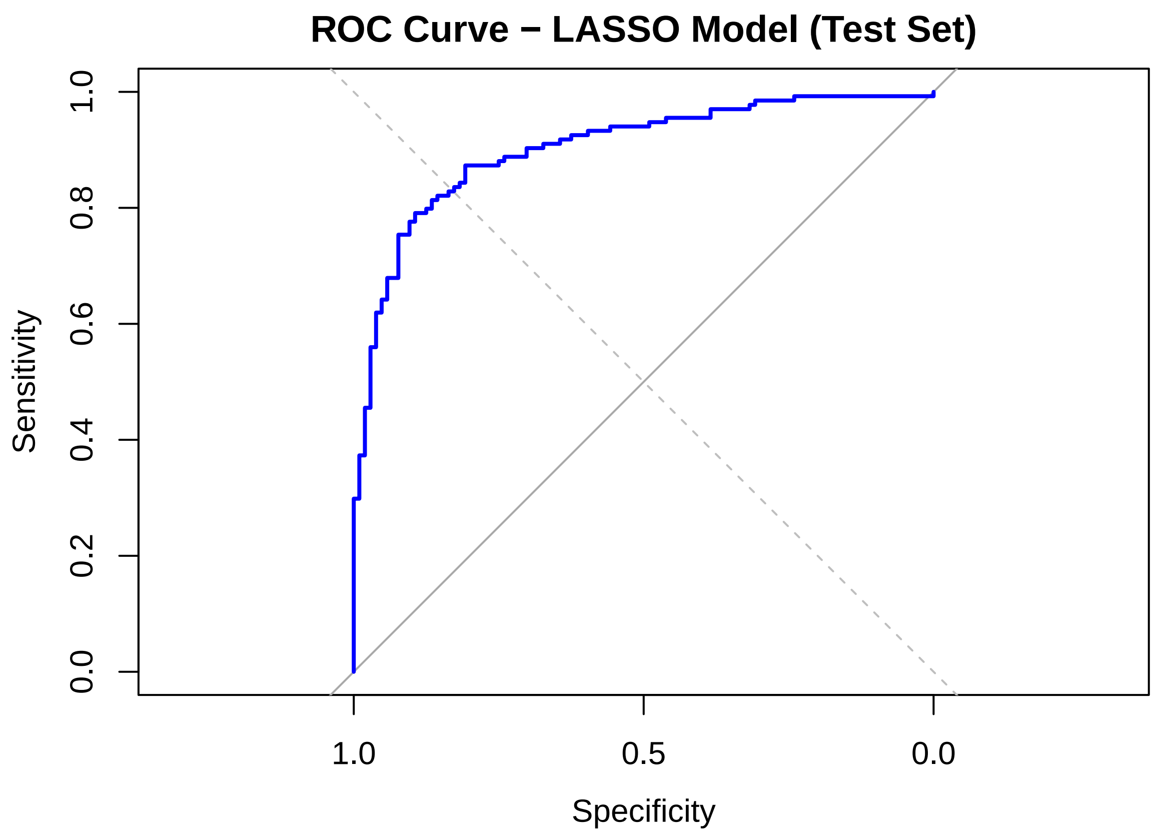
```r
library(pROC)

# Ensure both are numeric vectors of the same length
roc_lasso <- roc(as.numeric(y_test), as.numeric(pred_prob_test[, 1]))
```

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

## Setting direction: controls < cases
```

```r
# AUC value
auc_lasso <- auc(roc_lasso)
cat("LASSO Test AUC:", round(auc_lasso, 4), "\n")
```

```
## LASSO Test AUC: 0.9032
```

```r
# Plot ROC
plot(roc_lasso, col = "blue", lwd = 2, main = "ROC Curve - LASSO Model (Test Set)")
abline(a = 0, b = 1, lty = 2, col = "gray")
```



Compare AIC

```
set.seed(1)

aic_lr <- AIC(logistic_model)
deviance_lasso <- deviance(lasso_model)
df_lasso <- lasso_model$df
aic_lasso <- deviance_lasso + 2 * df_lasso

cat("AIC - Logistic Regression (glm):", round(aic_lr, 2), "\n")
```

```
## AIC - Logistic Regression (glm): 807.39
```

```
cat("AIC - LASSO (glmnet):", round(aic_lasso, 2), "\n")
```

```
## AIC - LASSO (glmnet): 809.04
```

```
library(MASS)
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.3
```

```r
library(readr)
library(caret)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.2

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

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

```r
library(forcats)

set.seed(1)
#Load data
heart_data <- read_csv("heart.csv")
```

```
## Rows: 1190 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): age, sex, chest pain type, resting bp s, cholesterol, fasting bloo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#Recode sex
heart_data$sex <- as.factor(heart_data$sex)
heart_data$sex <- fct_recode(heart_data$sex, "male" = "1", "female" = "0")

#Recode target to valid R variable names
heart_data$target <- as.factor(heart_data$target)
heart_data$target <- fct_recode(heart_data$target, "No" = "0", "Yes" = "1")

#Creation of train/test data
h_ind <- createDataPartition(heart_data$target, p = 0.8, list = FALSE)
heart_train <- heart_data[h_ind, ]
heart_test <- heart_data[-h_ind, ]

#PCA for variable selection (exclude target and separate numeric/categorical)
```

```
numeric_cols <- sapply(heart_train, is.numeric) & colnames(heart_train) != "target"
heart_train_numeric <- heart_train[, numeric_cols]
categorical_cols <- !numeric_cols & colnames(heart_train) != "target"
heart_train_categorical <- heart_train[, categorical_cols, drop = FALSE]

# Scale numeric predictors
heart_scale <- scale(heart_train_numeric)

#Perform PCA on numeric predictors
pca_result <- prcomp(heart_scale, center = TRUE, scale. = TRUE)
loadings <- pca_result$rotation
important_vars <- names(sort(abs(loadings[, 1]), decreasing = TRUE))[1:5]  # Top 5 from PC1
important_vars <- unique(c(important_vars, names(sort(abs(loadings[, 2]), decreasing = TRUE))[1:4]))  #
important_vars2 <- important_vars[!important_vars %in% c("target", "resting ecg")] # Exclude target and

#Combine selected numeric variables with all categorical variables
selected_cols <- c(important_vars2, colnames(heart_train_categorical))
train_selected <- heart_train[, selected_cols]
test_selected <- heart_test[, selected_cols]

#Prepare training data for Random Forest
rf_train_data <- data.frame(train_selected, target = as.factor(heart_train$target))

#Convert categorical predictors to factors
if ("st slope" %in% colnames(rf_train_data)) rf_train_data$`st slope` <- as.factor(rf_train_data$`st sl
if ("resting ecg" %in% colnames(rf_train_data)) rf_train_data$`resting ecg` <- as.factor(rf_train_data$
if ("exercise angina" %in% colnames(rf_train_data)) rf_train_data$`exercise angina` <- as.factor(rf_tra
if ("sex" %in% colnames(rf_train_data)) rf_train_data$sex <- as.factor(rf_train_data$sex)
if ("chest pain type" %in% colnames(rf_train_data)) rf_train_data$`chest pain type` <- as.factor(rf_tra

#Define custom summary function to include Accuracy, Kappa, ROC, Sens, Spec
customSummary <- function(data, lev = NULL, model = NULL) {
  out <- c(defaultSummary(data, lev, model), twoClassSummary(data, lev, model))
  out
}

#Define 5-fold cross-validation
train_control <- trainControl(
  method = "cv",
  number = 5,
  savePredictions = "final",
  classProbs = TRUE,
  summaryFunction = customSummary,  # Includes Accuracy, Kappa, ROC, Sens, Spec
  returnResamp = "all"
)
#Train Random Forest with 5-fold CV
rf_cv_model <- train(
  target ~ .,
  data = rf_train_data,
  method = "rf",
  trControl = train_control,
  ntree = 500,
  tuneGrid = data.frame(mtry = sqrt(ncol(rf_train_data) - 1)),
```

```r
  metric = "Accuracy"
)

#Cross-validation training error
cv_accuracy <- mean(rf_cv_model$results$Accuracy)
cv_error <- 1 - cv_accuracy
cat("5-Fold CV Training Error:", cv_error, "\n")
```

```
## 5-Fold CV Training Error: 0.09024073
```

```r
#Explicit training error (predict on full training data)
train_predictions <- predict(rf_cv_model, newdata = rf_train_data, type = "raw")
train_confusion <- table(rf_train_data$target, train_predictions)
train_accuracy <- sum(diag(train_confusion)) / sum(train_confusion)
train_error <- 1 - train_accuracy
cat("Training Error (Explicit):", train_error, "\n")
```

```
## Training Error (Explicit): 0
```

```r
#Print CV results
print(rf_cv_model)
```

```
## Random Forest
##
## 953 samples
##   9 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 762, 762, 764, 762, 762
## Resampling results:
##
##   Accuracy   Kappa      ROC        Sens       Spec
##   0.9097593  0.8184171  0.9542393  0.8797753  0.9364752
##
## Tuning parameter 'mtry' was held constant at a value of 3
```

```r
#Prepare test data for prediction
rf_test_data <- data.frame(test_selected, target = as.factor(heart_test$target))

#Convert categorical predictors to factors in test data
if ("st slope" %in% colnames(rf_test_data)) rf_test_data$`st slope` <- as.factor(rf_test_data$`st slope
if ("resting ecg" %in% colnames(rf_test_data)) rf_test_data$`resting ecg` <- as.factor(rf_test_data$`res
if ("exercise angina" %in% colnames(rf_test_data)) rf_test_data$`exercise angina` <- as.factor(rf_test_d
if ("sex" %in% colnames(rf_test_data)) rf_test_data$sex <- as.factor(rf_test_data$sex)
if ("chest pain type" %in% colnames(rf_test_data)) rf_test_data$`chest pain type` <- as.factor(rf_test_d

#Predict on test data
test_predictions <- predict(rf_cv_model, newdata = rf_test_data, type = "raw")
```

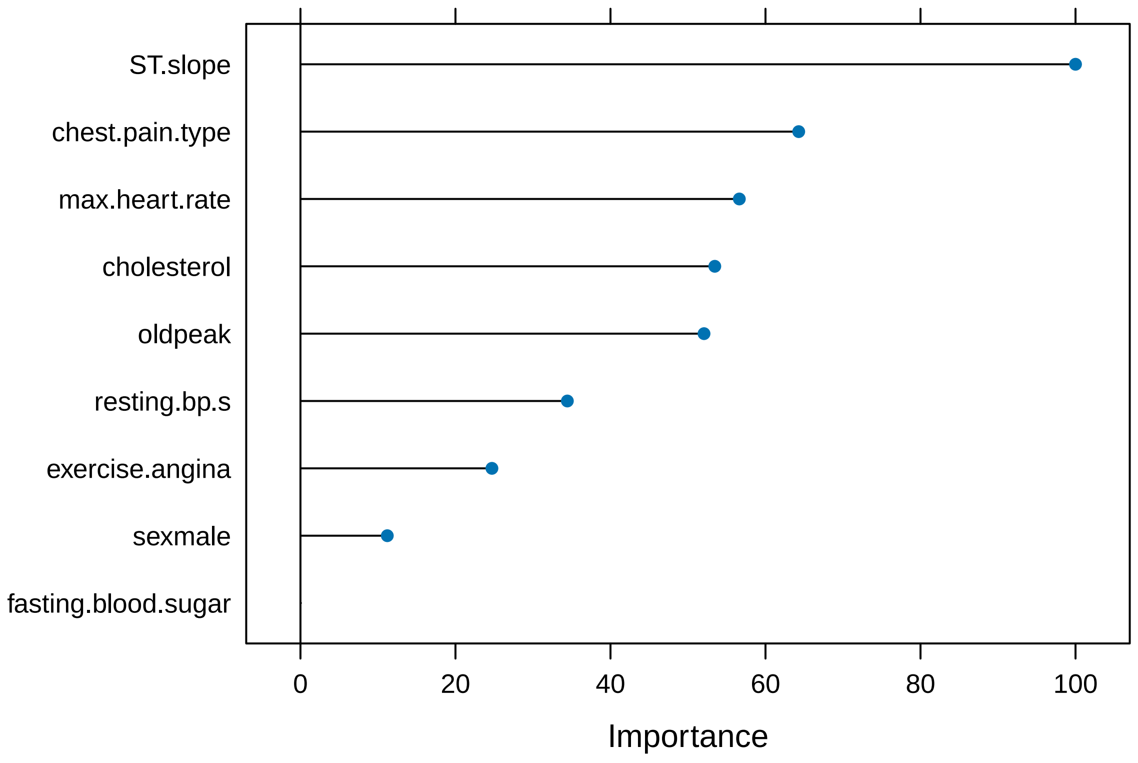
```
#Test error
test_confusion <- table(rf_test_data$target, test_predictions)
test_accuracy <- sum(diag(test_confusion)) / sum(test_confusion)
test_error <- 1 - test_accuracy
cat("Test Error:", test_error, "\n")
```

```
## Test Error: 0.05485232
```

```
#Variable importance
varImp(rf_cv_model)
```

```
## rf variable importance
##
##                     Overall
## ST.slope             100.00
## chest.pain.type       64.29
## max.heart.rate        56.62
## cholesterol           53.46
## oldpeak               52.07
## resting.bp.s          34.44
## exercise.angina       24.70
## sexmale               11.21
## fasting.blood.sugar    0.00
```

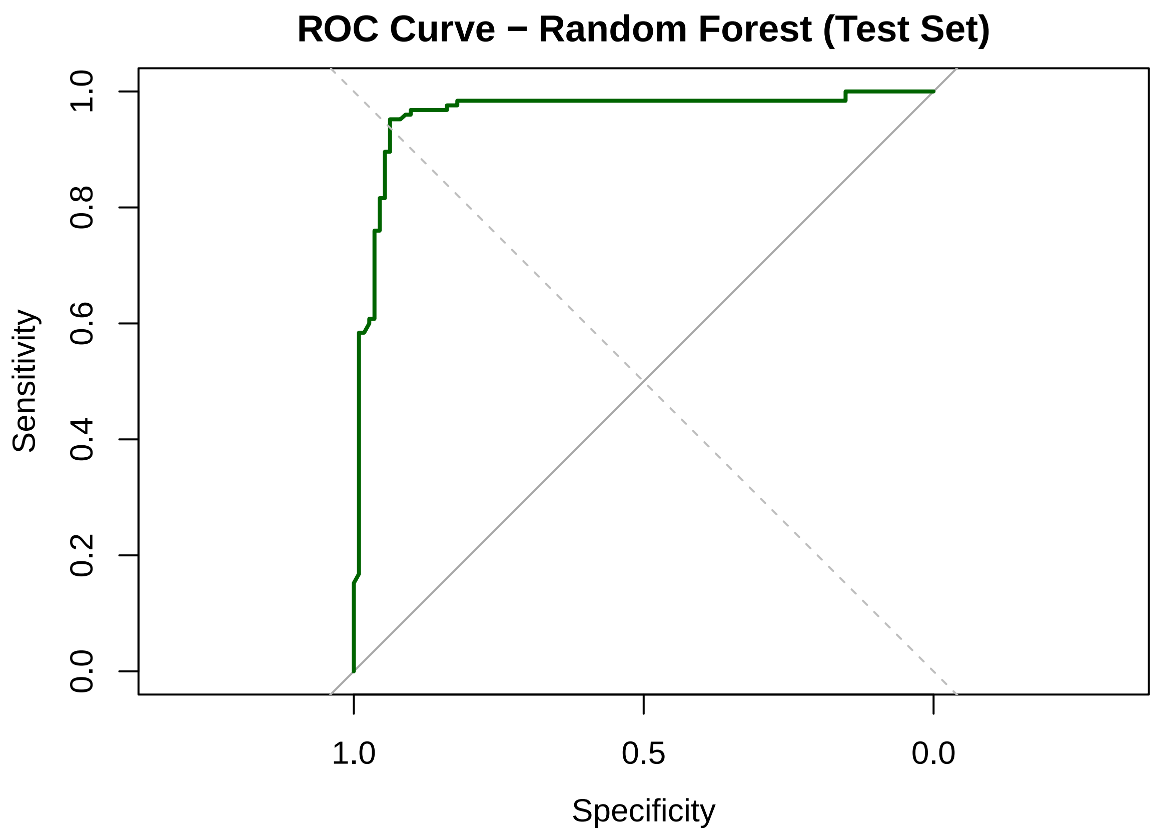
```
plot(varImp(rf_cv_model))
```



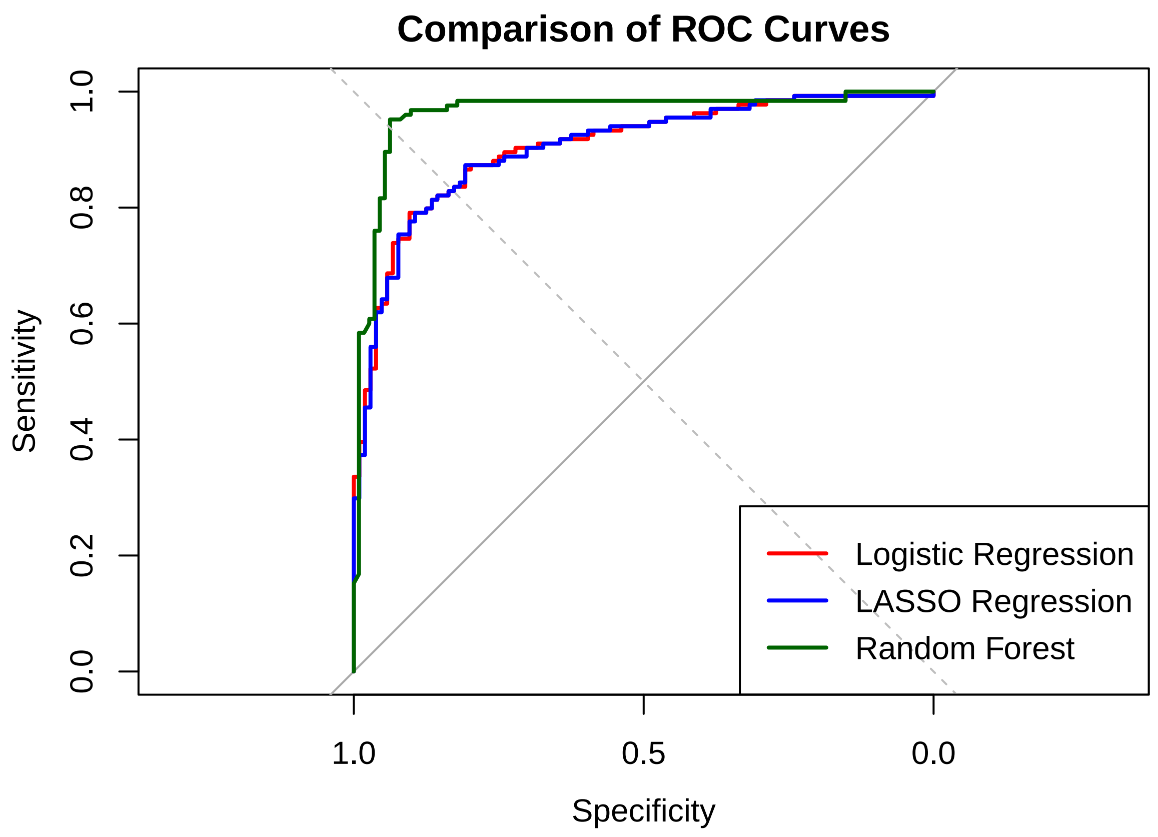
```r
library(pROC)

# Get predicted probabilities for test data
test_prob <- predict(rf_cv_model, newdata = rf_test_data, type = "prob")

# Ensure target is binary factor with levels: "No", "Yes"
rf_test_data$target <- factor(rf_test_data$target, levels = c("No", "Yes"))

# Compute ROC and AUC for "Yes" class
roc_rf <- roc(response = rf_test_data$target,
              predictor = test_prob$Yes,
              levels = c("No", "Yes"),
              direction = "<")

# Print AUC
auc_rf <- auc(roc_rf)
cat("Random Forest Test AUC:", round(auc_rf, 4), "\n")
```

```
## Random Forest Test AUC: 0.9622
```

```r
# Plot ROC curve
plot(roc_rf, col = "darkgreen", lwd = 2, main = "ROC Curve - Random Forest (Test Set)")
abline(a = 0, b = 1, lty = 2, col = "gray")
```



```r
library(pROC)

# Plot all ROC curves
plot(roc_logistic, col = "red", lwd = 2, main = "Comparison of ROC Curves")
plot(roc_lasso, col = "blue", lwd = 2, add = TRUE)
plot(roc_rf, col = "darkgreen", lwd = 2, add = TRUE)
abline(a = 0, b = 1, lty = 2, col = "gray")

legend("bottomright",
       legend = c("Logistic Regression", "LASSO Regression", "Random Forest"),
       col = c("red", "blue", "darkgreen"),
       lwd = 2)
```


Comparison of ROC Curves
Sensitivity
Specificity
Logistic Regression
LASSO Regression
Random Forest