ML_Final_Project

Haoyang Zhang

2024-05-10

0 Initialiaztion

```
# Data manipulation
library(dplyr)
library(tidyverse)

# preprocess
library(stats)

# Models packages
library(MASS)
library(ISLR)
library(leaps)
library(pROC)
library(glmnet)
library(boot)
library(caret)
library(randomForest)
library(gbm)
library(car)

# Plottings
library(ggplot2)
library(ggfortify)
library(GGally)
library(kableExtra)
library(cowplot)
library(corrplot)
library(reshape2)
library(pROC)
```

1 Preprocessing

i. Importing Original Dataset

```
heart_attack <- read.csv("heart_attack_prediction_dataset.csv")
```

ii. Splitting Blood Pressure Variable into Systolic & Diastolic

```r
split_bp <- do.call(rbind, strsplit(heart_attack$Blood.Pressure, split = "/"))


heart_attack$Systolic <- as.integer(split_bp[, 1])
heart_attack$Diastolic <- as.integer(split_bp[, 2])


heart_attack <- heart_attack[, -which(names(heart_attack) == "Blood.Pressure")]


bp_position <- which(names(heart_attack) == "Cholesterol") + 1
heart_attack <- heart_attack[, c(1:bp_position - 1,
                                 (ncol(heart_attack)-1):ncol(heart_attack),
                                 bp_position:(ncol(heart_attack)-2))
                             ]
# Retrieval Set
write.csv(heart_attack, "heart_attack.csv")
```

iii. Encode Dataset

```r
categoricals <- c("Sex", "Diabetes", "Family.History", "Smoking", "Obesity",
                  "Alcohol.Consumption", "Diet", "Previous.Heart.Problems",
                  "Medication.Use", "Stress.Level", "Physical.Activity.Days.Per.Week",
                  "Country", "Continent", "Hemisphere", "Heart.Attack.Risk")
for (var in categoricals) {
  heart_attack[[var]] <- factor(heart_attack[[var]])
}
```

In all categorical variables `Diet`, `Stress.Level` and `Physical.Activity.Days.Per.Week` are ordinal. However `Stress.Level` and `Physical.Activity.Days.Per.Week` are originally coded in integers and we have already factorized them, therefore we shall now encode `Diet`.

```r
levels_diet <- c("Unhealthy", "Average", "Healthy")
codes_diet <- c(-1, 0, 1)
names(codes_diet) <- levels_diet
heart_attack$Diet <- factor(codes_diet[heart_attack$Diet])
class(heart_attack$Diet)
```

```
## [1] "factor"
```

```r
print(codes_diet)
```

```
## Unhealthy   Average   Healthy
##        -1         0         1
```

The rest of categoricals are either binary (YES or NO) or stored as characters. We shall leave those binary since they have already been encoded as 1 or 0. The character variables in this dataset are all nominal except

Diet, Stress.Level and Physical.Activity.Days.Per.Week. Thus we could assign them with arbitrary integers.

```
nominals <- c("Sex", "Country", "Continent", "Hemisphere")

# Initialize an empty list to store encoding details
codebook <- list()

for (var in nominals) {

  # Get unique levels and create an encoding mapping from 1 to N
  levels_set <- levels(heart_attack[[var]])
  encoding_map <- setNames(seq_along(levels_set), levels_set)

  # Apply encoding
  heart_attack[[var]] <- as.integer(heart_attack[[var]])

  # Record the encoding in the codebook
  codebook[[var]] <- data.frame(
    Level = levels_set,
    Code = as.integer(encoding_map[levels_set])
  )

}
```

```
# add Diet into codebook
codebook[["Diet"]] <- as.data.frame(codes_diet)
print(codebook)
```

```
## $Sex
##    Level Code
## 1 Female    1
## 2   Male    2
##
## $Country
##             Level Code
## 1       Argentina    1
## 2       Australia    2
## 3          Brazil    3
## 4          Canada    4
## 5           China    5
## 6        Colombia    6
## 7          France    7
## 8         Germany    8
## 9           India    9
## 10          Italy   10
## 11          Japan   11
## 12    New Zealand   12
## 13        Nigeria   13
## 14   South Africa   14
## 15    South Korea   15
## 16          Spain   16
## 17       Thailand   17
```

```
## 18 United Kingdom   18
## 19  United States   19
## 20        Vietnam   20
##
## $Continent
##           Level Code
## 1        Africa    1
## 2          Asia    2
## 3     Australia    3
## 4        Europe    4
## 5 North America    5
## 6 South America    6
##
## $Hemisphere
##                 Level Code
## 1 Northern Hemisphere    1
## 2 Southern Hemisphere    2
##
## $Diet
##           codes_diet
## Unhealthy         -1
## Average            0
## Healthy            1
```

```
heart_attack$Patient.ID <- NULL
write.csv(heart_attack, "dataset.csv")

# Use modified dataset in following procedures
dataset <- read.csv("dataset.csv")
```

2 Exploratory Analysis

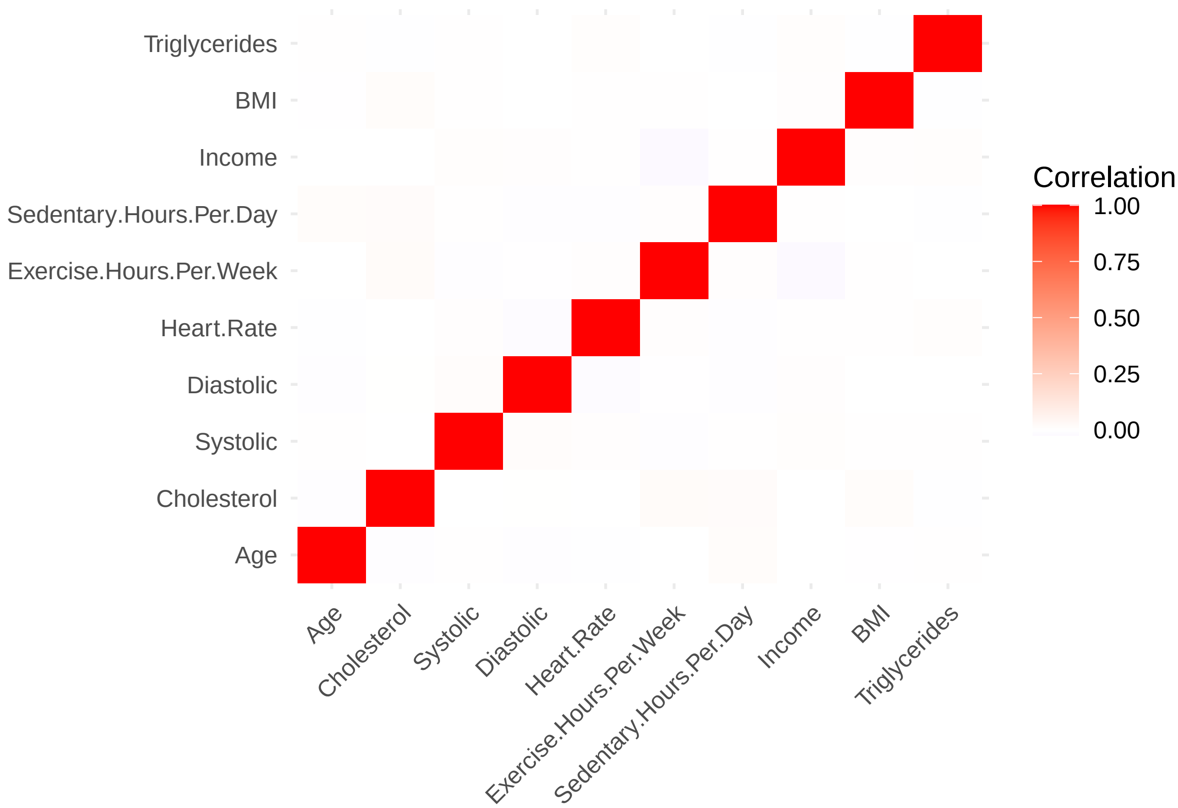
```
dataset <- read.csv("dataset.csv")
dataset$X <- NULL
```

```
# factors
factors <- c("Sex", "Diabetes", "Family.History", "Smoking", "Obesity",
                 "Alcohol.Consumption", "Diet", "Previous.Heart.Problems",
                 "Medication.Use", "Stress.Level", "Physical.Activity.Days.Per.Week",
                 "Country", "Continent", "Hemisphere", "Heart.Attack.Risk")
for (var in factors) {
  dataset[[var]] <- as.factor(dataset[[var]])
}
```

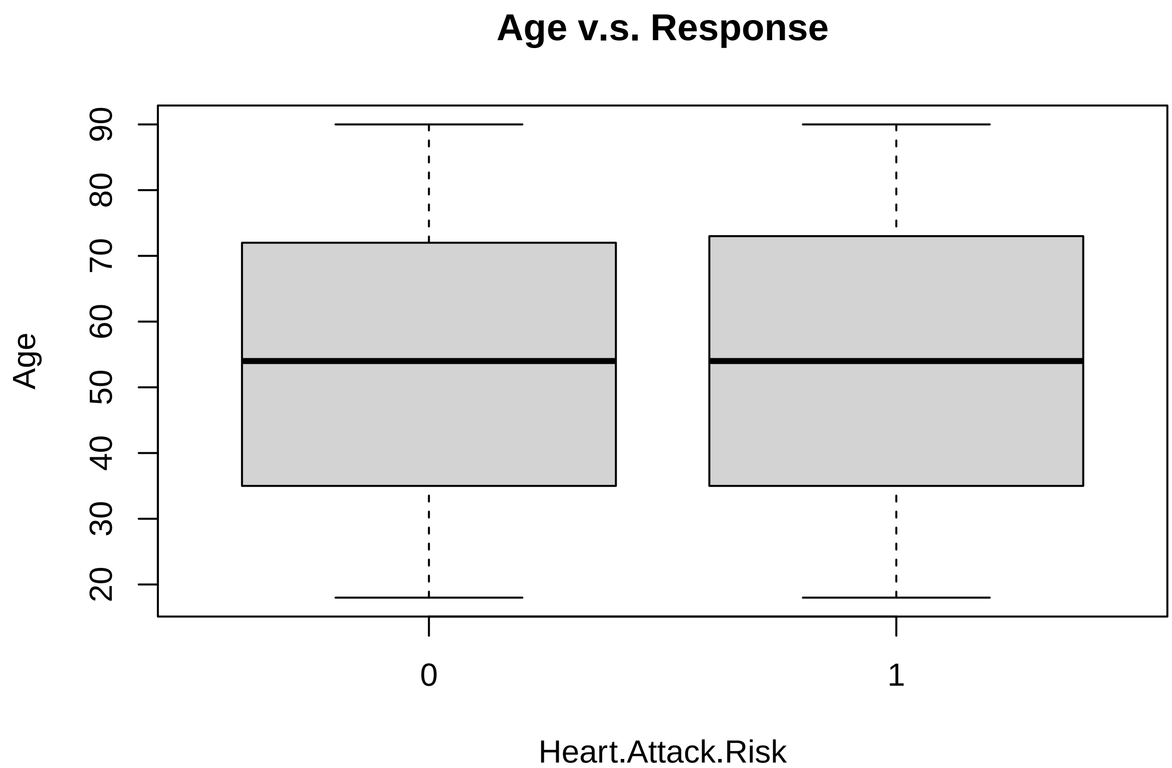
Correlation matrix for continuous variables.

```
numerics <- c("Age", "Cholesterol", "Systolic", "Diastolic", "Heart.Rate",
              "Exercise.Hours.Per.Week", "Sedentary.Hours.Per.Day",
              "Income", "BMI", "Triglycerides")
cor_matrix <- cor(dataset[numerics])
```

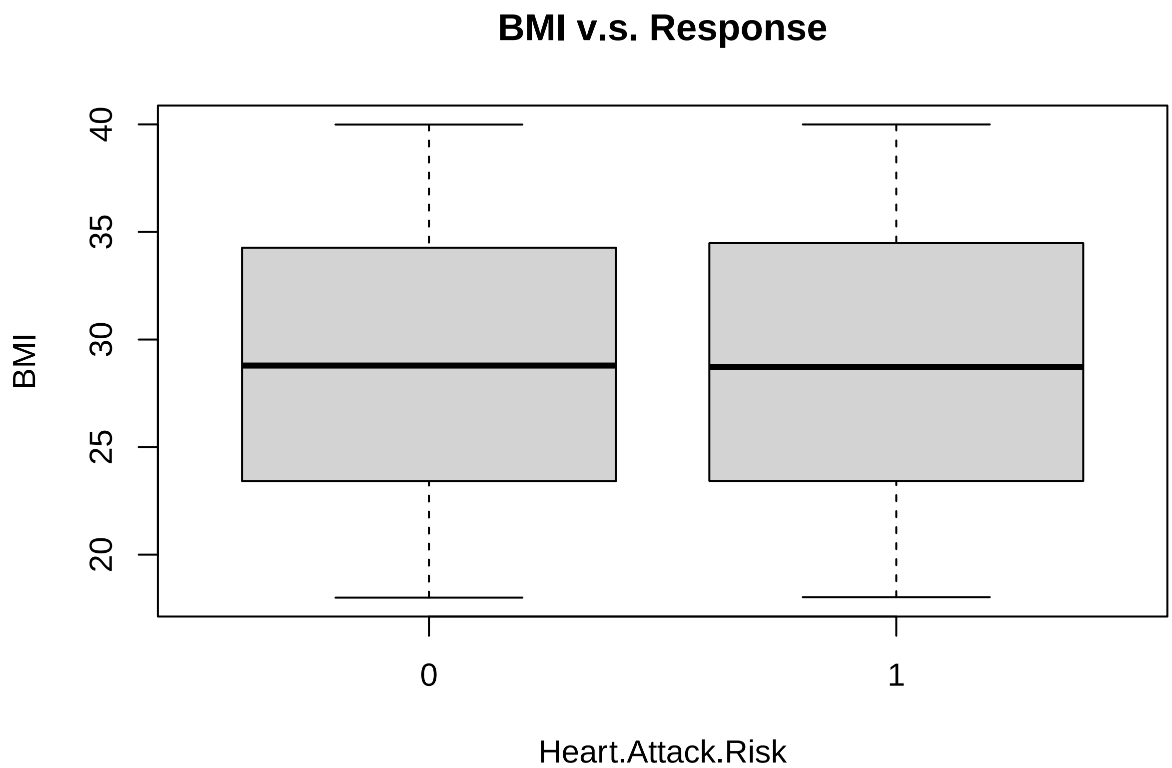
```r
# Melt the correlation matrix
melted_cor_matrix <- melt(cor_matrix)

ggplot(melted_cor_matrix, aes(Var1, Var2, fill = value)) +
    geom_tile() +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    labs(fill = "Correlation", x = "", y = "")
```



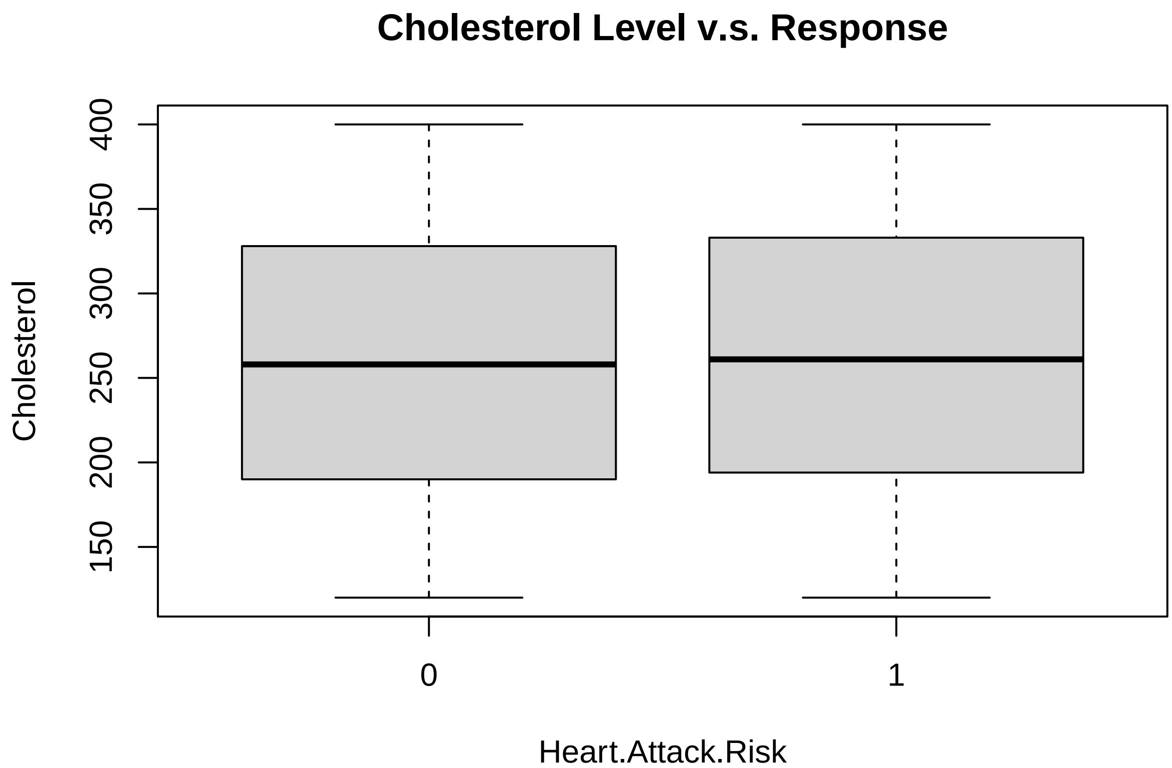

```r
boxplot(Age ~ Heart.Attack.Risk, data = dataset,
        main = "Age v.s. Response")
```



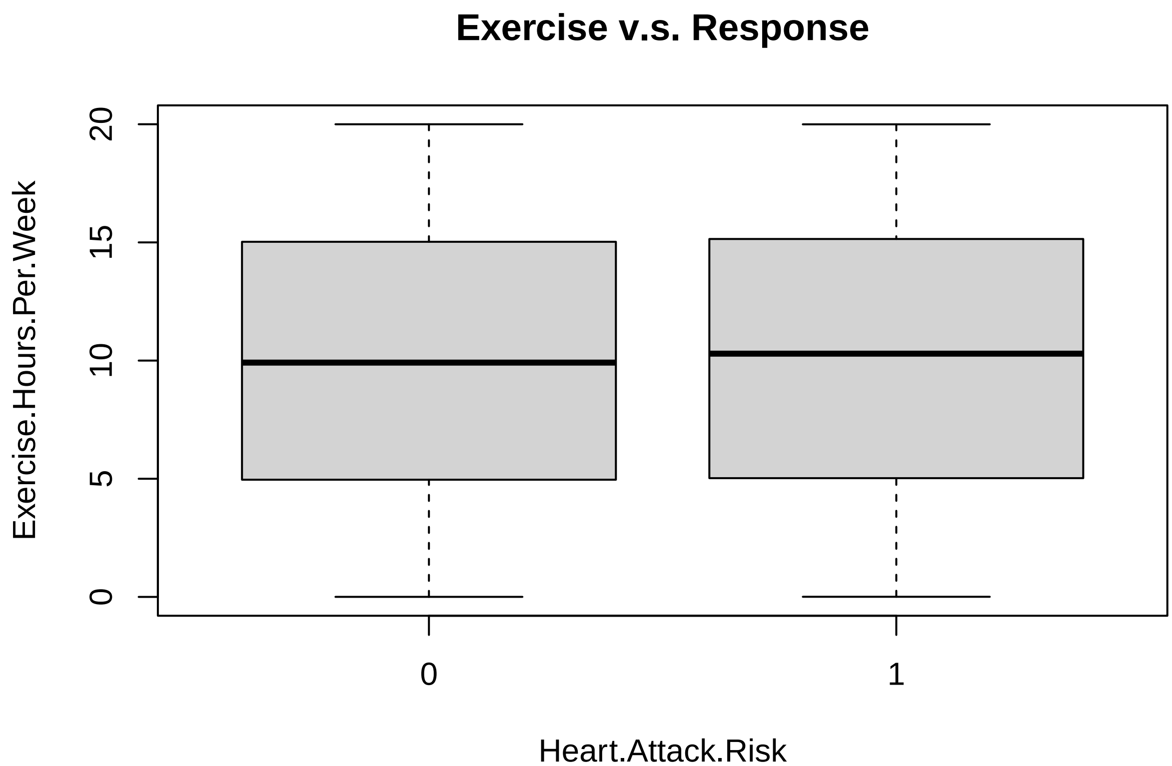

```
boxplot(BMI ~ Heart.Attack.Risk, data = dataset,
        main = "BMI v.s. Response")
```



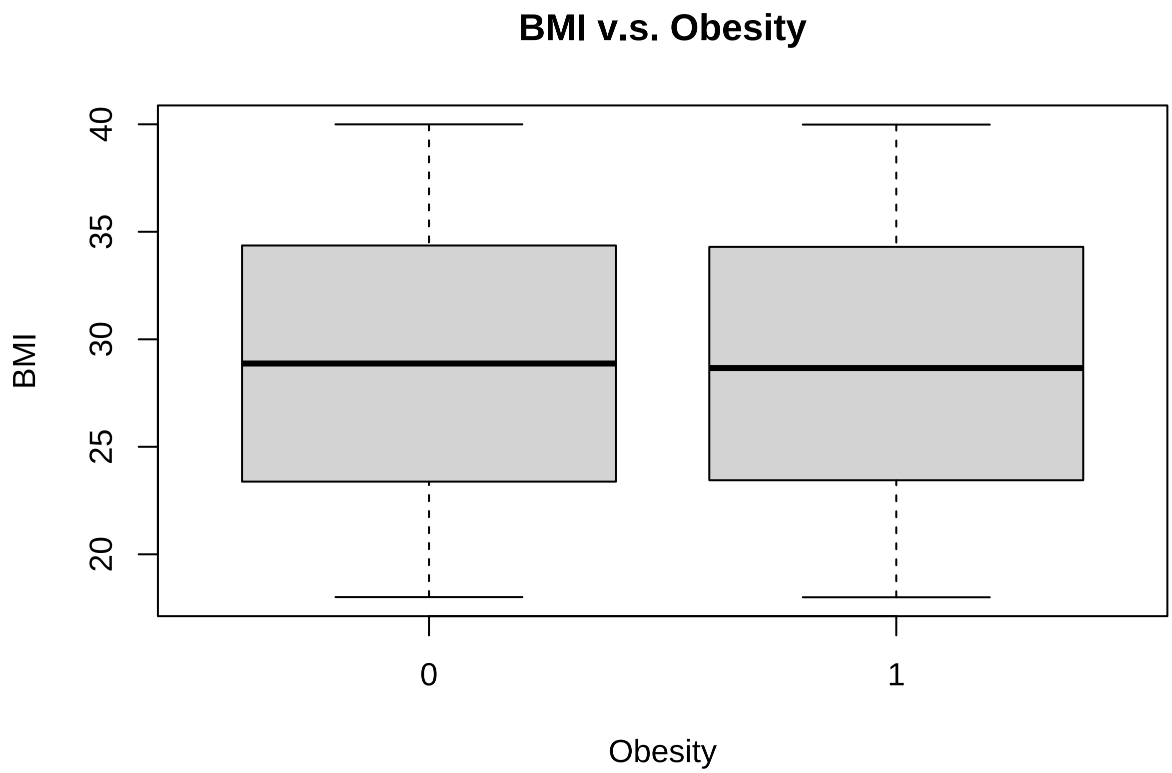

```r
boxplot(Cholesterol ~Heart.Attack.Risk, data = dataset,
        main = "Cholesterol Level v.s. Response")
```




```r
boxplot(Exercise.Hours.Per.Week ~Heart.Attack.Risk, data = dataset,
        main = "Exercise v.s. Response")
```




```r
boxplot(BMI ~ Obesity, data = dataset,
        main = "BMI v.s. Obesity")
```




```
BMI1 <- dataset$BMI[dataset$Obesity == 0]
BMI2 <- dataset$BMI[dataset$Obesity == 1]
t.test(BMI1, BMI2)
```

```
##
##  Welch Two Sample t-test
##
## data:  BMI1 and BMI2
## t = 0.56708, df = 8760.8, p-value = 0.5707
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1880972  0.3412251
## sample estimates:
## mean of x mean of y
##  28.92984  28.85327
```

3 Modelling & Evaluation

i. Partition Dataset

Considering the number of observations is sufficient, and wishing of the testing results as accurate as possible, decided to assign 70% of total sample to testing set.

```r
set.seed(2024)
N <- nrow(dataset)
train_indices <- sample(1:N, size = N*0.7)
train_set <- dataset[train_indices, ]
test_set <- dataset[-train_indices, ]
```

```r
threshold <- mean(as.integer(dataset$Heart.Attack.Risk) - 1)
```

ii. Logistic Regression

```r
logit_full_fit <- glm(Heart.Attack.Risk ~ ., data = train_set, family = "binomial")
summary(logit_full_fit)
```

```
##
## Call:
## glm(formula = Heart.Attack.Risk ~ ., family = "binomial", data = train_set)
##
## Coefficients: (6 not defined because of singularities)
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -3.258e-01  3.626e-01  -0.899   0.3689
## Age                           8.470e-04  1.416e-03   0.598   0.5498
## Sex2                          4.868e-02  7.046e-02   0.691   0.4897
## Cholesterol                   4.255e-04  3.315e-04   1.283   0.1993
## Systolic                      1.247e-03  1.016e-03   1.227   0.2199
## Diastolic                     6.181e-04  1.821e-03   0.339   0.7343
## Heart.Rate                   -1.720e-03  1.304e-03  -1.319   0.1871
## Diabetes1                     8.548e-02  5.665e-02   1.509   0.1313
## Family.History1               6.274e-03  5.370e-02   0.117   0.9070
## Smoking1                     -1.626e-02  1.154e-01  -0.141   0.8880
## Obesity1                     -6.186e-02  5.362e-02  -1.154   0.2486
## Alcohol.Consumption1         -5.604e-02  5.459e-02  -1.027   0.3046
## Exercise.Hours.Per.Week       3.003e-03  4.629e-03   0.649   0.5165
## Diet0                        -2.490e-02  6.547e-02  -0.380   0.7037
## Diet1                        -1.469e-02  6.602e-02  -0.223   0.8239
## Previous.Heart.Problems1      4.583e-02  5.365e-02   0.854   0.3930
## Medication.Use1               5.567e-04  5.368e-02   0.010   0.9917
## Stress.Level2                 3.013e-02  1.183e-01   0.255   0.7990
## Stress.Level3                 1.476e-02  1.201e-01   0.123   0.9022
## Stress.Level4                -1.037e-01  1.209e-01  -0.858   0.3911
## Stress.Level5                 3.215e-02  1.216e-01   0.264   0.7914
## Stress.Level6                 9.431e-02  1.202e-01   0.785   0.4326
## Stress.Level7                 2.790e-02  1.195e-01   0.233   0.8154
## Stress.Level8                -7.727e-02  1.210e-01  -0.639   0.5230
## Stress.Level9                -1.313e-01  1.213e-01  -1.083   0.2789
## Stress.Level10               -3.892e-02  1.220e-01  -0.319   0.7498
## Sedentary.Hours.Per.Day      -9.728e-03  7.766e-03  -1.253   0.2103
## Income                        1.474e-08  3.323e-07   0.044   0.9646
## BMI                          -2.257e-03  4.244e-03  -0.532   0.5948
## Triglycerides                 8.647e-05  1.202e-04   0.719   0.4718
## Physical.Activity.Days.Per.Week1 -2.176e-01  1.071e-01  -2.032   0.0421 *
## Physical.Activity.Days.Per.Week2 -1.824e-01  1.084e-01  -1.683   0.0924 .
```

```
## Physical.Activity.Days.Per.Week3 -2.002e-01  1.055e-01  -1.898   0.0577 .
## Physical.Activity.Days.Per.Week4 -4.584e-02  1.072e-01  -0.427   0.6691
## Physical.Activity.Days.Per.Week5 -1.933e-01  1.075e-01  -1.798   0.0721 .
## Physical.Activity.Days.Per.Week6 -7.279e-02  1.069e-01  -0.681   0.4959
## Physical.Activity.Days.Per.Week7 -1.118e-01  1.070e-01  -1.045   0.2960
## Sleep.Hours.Per.Day              -3.116e-02  1.345e-02  -2.316   0.0206 *
## Country2                         -8.193e-02  1.636e-01  -0.501   0.6165
## Country3                         -1.780e-01  1.619e-01  -1.100   0.2715
## Country4                         -2.135e-01  1.671e-01  -1.278   0.2013
## Country5                         -1.521e-01  1.651e-01  -0.921   0.3569
## Country6                          1.281e-02  1.642e-01   0.078   0.9378
## Country7                         -6.298e-02  1.641e-01  -0.384   0.7012
## Country8                         -6.514e-02  1.583e-01  -0.412   0.6807
## Country9                         -3.385e-01  1.709e-01  -1.981   0.0476 *
## Country10                        -1.845e-01  1.677e-01  -1.100   0.2711
## Country11                        -2.437e-01  1.706e-01  -1.428   0.1532
## Country12                        -1.577e-01  1.700e-01  -0.928   0.3536
## Country13                         9.400e-02  1.631e-01   0.576   0.5643
## Country14                        -2.034e-01  1.689e-01  -1.205   0.2283
## Country15                         3.219e-03  1.650e-01   0.020   0.9844
## Country16                        -3.632e-02  1.636e-01  -0.222   0.8242
## Country17                        -2.663e-02  1.679e-01  -0.159   0.8740
## Country18                        -2.465e-01  1.667e-01  -1.479   0.1391
## Country19                         1.281e-01  1.643e-01   0.780   0.4356
## Country20                        -1.860e-01  1.690e-01  -1.100   0.2711
## Continent2                               NA         NA      NA       NA
## Continent3                               NA         NA      NA       NA
## Continent4                               NA         NA      NA       NA
## Continent5                               NA         NA      NA       NA
## Continent6                               NA         NA      NA       NA
## Hemisphere2                              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8019.3  on 6133  degrees of freedom
## Residual deviance: 7965.5  on 6077  degrees of freedom
## AIC: 8079.5
##
## Number of Fisher Scoring iterations: 4
```

```r
# training error
pred_train_prob <- predict(logit_full_fit, newdata = train_set, type = "response")
pred_train_label <- ifelse(pred_train_prob > 0.5, 1, 0)
table(pred_train_label, train_set$Heart.Attack.Risk)
```

```
##
## pred_train_label    0    1
##                0 3919 2206
##                1    4    5
```

```
train_error_logit <- mean(pred_train_label != train_set$Heart.Attack.Risk)
train_error_logit
```

```
## [1] 0.3602869
```

```
# testing error
pred_test_prob <- predict(logit_full_fit, newdata = test_set, type = "response")
pred_test_label <- ifelse(pred_test_prob > 0.5, 1, 0)
table(pred_test_label, test_set$Heart.Attack.Risk)
```

```
##
## pred_test_label    0    1
##               0 1699  928
##               1    2    0
```

```
test_error_logit <- mean(pred_test_label != test_set$Heart.Attack.Risk)
test_error_logit
```

```
## [1] 0.3537467
```

Conduct stepwise model selection from both sides based on AIC.

```
logit_null_fit <- glm(Heart.Attack.Risk ~ 1, data = train_set, family = "binomial")

logit_model <- stepAIC(logit_full_fit,
                       scope = list(lower = logit_null_fit, upper = logit_full_fit),
                       direction = "both", trace = FALSE, k = 2)

summary(logit_model)
```

```
##
## Call:
## glm(formula = Heart.Attack.Risk ~ Cholesterol + Diabetes + Sleep.Hours.Per.Day,
##     family = "binomial", data = train_set)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.5293869  0.1344419  -3.938 8.23e-05 ***
## Cholesterol          0.0004670  0.0003289   1.420   0.1556
## Diabetes1            0.0796777  0.0561652   1.419   0.1560
## Sleep.Hours.Per.Day -0.0311272  0.0133674  -2.329   0.0199 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8019.3  on 6133  degrees of freedom
## Residual deviance: 8009.8  on 6130  degrees of freedom
## AIC: 8017.8
##
## Number of Fisher Scoring iterations: 4
```

```r
# training error
pred_train_prob <- predict(logit_model, newdata = train_set, type = "response")
pred_train_label <- ifelse(pred_train_prob > 0.5, 1, 0)
table(pred_train_label, train_set$Heart.Attack.Risk)
```

```
##
## pred_train_label    0    1
##                0 3923 2211
```

```r
train_error_logit <- mean(pred_train_label != train_set$Heart.Attack.Risk)
train_error_logit
```

```
## [1] 0.36045
```

```r
# testing error
pred_test_prob <- predict(logit_model, newdata = test_set, type = "response")
pred_test_label <- ifelse(pred_test_prob > 0.5, 1, 0)
table(pred_test_label, test_set$Heart.Attack.Risk)
```

```
##
## pred_test_label    0    1
##               0 1701  928
```

```r
test_error_logit <- mean(pred_test_label != test_set$Heart.Attack.Risk)
test_error_logit
```

```
## [1] 0.3529859
```

The train-test evaluation results of logistic regression are not ideal, both for the full model and the model after stepwise selection. In terms of the Cross Validation evaluation, since our sample size is relatively large, decided to use k-folds CV in order to keep computational costs in check. $k = 8$ is chosen so that for each fold there is roughly 1000 observations ensuring the training sets are reasonably representative.

```r
# record the best subset according the AIC selection
best_subset <- as.formula("Heart.Attack.Risk ~ Cholesterol + Diabetes + Sleep.Hours.Per.Day")

my_cost <- function(r, pi = 0) {
  pred_lab <- pi > 0.5
  mean(pred_lab != r)
}

mod <- glm(best_subset,
           family = "binomial", data = dataset)
cv.glm(dataset, mod, cost = my_cost, K=8)$delta[1]
```

```
## [1] 0.3582107
```

iii. KNN modelling & CV

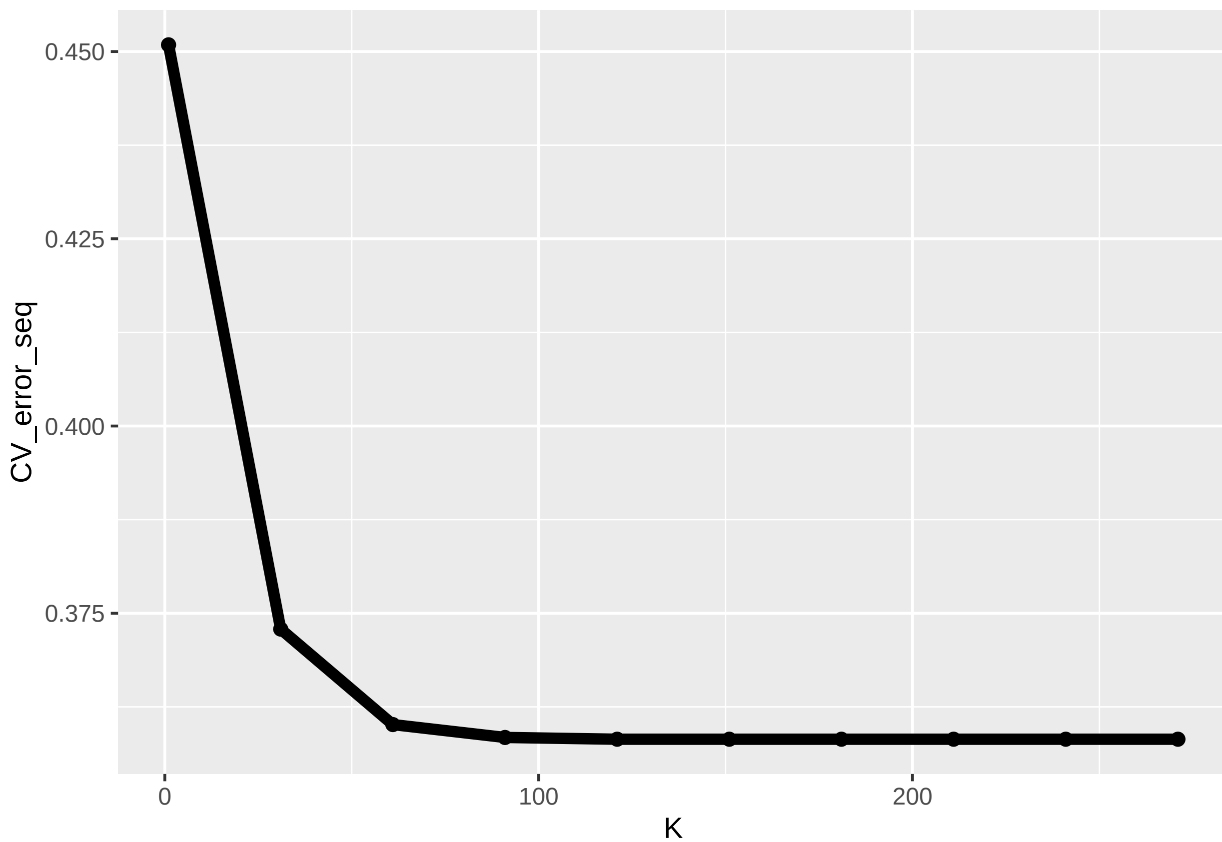
Standardize the dataset since KNN is distance sensitive.

```r
preProcValues <- preProcess(dataset, method = c("center", "scale"))
dataset_scaled <- predict(preProcValues, dataset)
```

```r
k <- 8
N <- nrow(dataset_scaled)
fold_ind <- sample(1:k, N, replace = TRUE)
K_seq <- seq(from = 1, to = 300, by = 30)
CV_error_seq <- sapply(K_seq, function(K_cur) {
  mean(sapply(1:k, function(j) {
    fit_knn <- knn3(best_subset,
                    data = dataset_scaled[fold_ind != j, ], k = K_cur)
    pred_knn <- predict(fit_knn, newdata = dataset_scaled[fold_ind == j, ], type = "class")
    mean(pred_knn != dataset_scaled$Heart.Attack.Risk[fold_ind == j])
  }))
})

KNN_errors <- data.frame(K = K_seq,
                         Errors = CV_error_seq)
print(KNN_errors)
```

```
##      K    Errors
## 1    1 0.4509025
## 2   31 0.3728806
## 3   61 0.3601332
## 4   91 0.3584114
## 5  121 0.3581776
## 6  151 0.3581776
## 7  181 0.3581776
## 8  211 0.3581776
## 9  241 0.3581776
## 10 271 0.3581776
```

```r
ggplot(KNN_errors, mapping = aes(x = K, y = CV_error_seq)) +
  geom_point(size = 2) +
  geom_line(size = 2)
```




```r
N <- nrow(dataset_scaled)
train_indices <- sample(1:N, size = N*0.7)
train_set_scaled <- dataset_scaled[train_indices, ]
test_set_scaled <- dataset_scaled[-train_indices, ]
```

```r
knn_mod <- knn3(Heart.Attack.Risk ~ ., data = train_set, k = 100)
```

```r
# training error
pred_train_class <- predict(knn_mod, train_set_scaled, type = "class")
tr_confmat <- confusionMatrix(pred_train_class, train_set_scaled$Heart.Attack.Risk)
accura_tr <- tr_confmat$overall['Accuracy']
error_tr <- 1 - accura_tr
sn_tr <- tr_confmat$byClass['Sensitivity']
sp_tr <- tr_confmat$byClass['Specificity']

# testing error
pred_test_class <- predict(knn_mod, test_set_scaled, type = "class")
te_confmat <- confusionMatrix(pred_test_class, test_set_scaled$Heart.Attack.Risk)
accura_te <- te_confmat$overall['Accuracy']
error_te <- 1 - accura_te
sn_te <- te_confmat$byClass['Sensitivity']
sp_te <- te_confmat$byClass['Specificity']

# output
```

```r
df <- data.frame(Accuracy = c(accura_tr, accura_te),
               Error = c(error_tr, error_te),
               Sensitivity = c(sn_tr, sn_te),
               Specificity = c(sp_tr, sp_te))
row.names(df) <- c("Train", "Test")
print(round(df, 3))
```

```
##       Accuracy Error Sensitivity Specificity
## Train    0.646 0.354           1           0
## Test     0.631 0.369           1           0
```

iv. Discriminant Analysis

```r
# record the best subset according the AIC selection
best_subset <- as.formula("Heart.Attack.Risk ~ Cholesterol + Diabetes + Sleep.Hours.Per.Day")
```

```r
lda_fit <- lda(best_subset, data = train_set)
lda_fit
```

```
## Call:
## lda(best_subset, data = train_set)
##
## Prior probabilities of groups:
##       0       1
## 0.63955 0.36045
##
## Group means:
##   Cholesterol Diabetes1 Sleep.Hours.Per.Day
## 0    258.2264 0.6474637            7.053785
## 1    261.1805 0.6657621            6.929896
##
## Coefficients of linear discriminants:
##                              LD1
## Cholesterol          0.005709874
## Diabetes1            0.971164108
## Sleep.Hours.Per.Day -0.380626268
```

```r
# training
lda_pred <- predict(lda_fit, train_set)
lda_class <- lda_pred$class
mean(lda_class != train_set$Heart.Attack.Risk)
```

```
## [1] 0.36045
```

```r
table(lda_class, train_set$Heart.Attack.Risk)
```

```
##
## lda_class    0    1
##         0 3923 2211
##         1    0    0
```

```r
# testing
lda_pred <- predict(lda_fit, test_set)
lda_class <- lda_pred$class
mean(lda_class != test_set$Heart.Attack.Risk)
```

```
## [1] 0.3529859
```

```r
table(lda_class, test_set$Heart.Attack.Risk)
```

```
##
## lda_class    0    1
##         0 1701  928
##         1    0    0
```

v. Random Forest

```r
rf_fit <- randomForest(Heart.Attack.Risk ~ ., data = train_set, importance = TRUE)
rf_pred <- predict(rf_fit, newdata = test_set)
mean((rf_pred != test_set$Heart.Attack.Risk)**2)
```

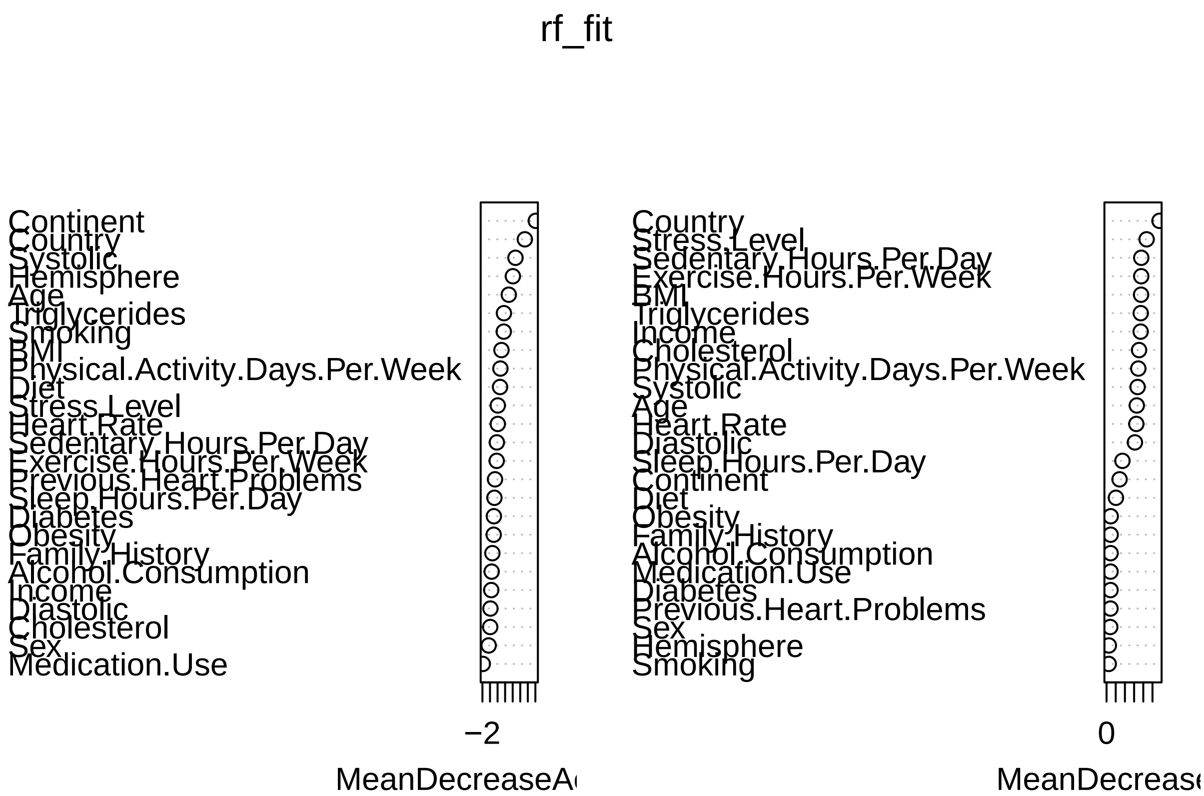
```
## [1] 0.3583111
```

```r
importance(rf_fit)
```

```
##                                       0           1 MeanDecreaseAccuracy
## Age                          3.26860786  -1.8506680           1.48466039
## Sex                         -0.44381437  -1.2825579          -1.17450119
## Cholesterol                 -0.66248337  -0.6946685          -0.97451254
## Systolic                     2.47434256   0.4288417           2.36494003
## Diastolic                   -0.62793968  -0.7456362          -0.94421090
## Heart.Rate                  -0.58436203   0.7593503           0.02943688
## Diabetes                    -0.75852785   0.1454544          -0.48876472
## Family.History               0.01730466  -1.1640367          -0.68712360
## Smoking                      1.83227025  -1.2316925           0.82090393
## Obesity                     -0.07256300  -0.9095120          -0.51381469
## Alcohol.Consumption         -0.52703975  -0.7259894          -0.78123216
## Exercise.Hours.Per.Week     -0.77332023   0.8699377          -0.11511161
## Diet                         0.32803433   0.0556475           0.32804895
## Previous.Heart.Problems      0.60199959  -1.2893698          -0.33797226
## Medication.Use              -0.55122299  -2.4184845          -1.95164064
## Stress.Level                -0.08676978   0.1843831           0.03027648
## Sedentary.Hours.Per.Day     -1.61689683   1.8205206          -0.09520895
## Income                       1.03265767  -2.8304743          -0.83482920
## BMI                          0.44866105   0.2032029           0.51477468
## Triglycerides                0.49753010   0.7369578           0.84166718
## Physical.Activity.Days.Per.Week  0.66147616  -0.2745046       0.37412204
## Sleep.Hours.Per.Day         -1.09399817   0.8451356          -0.41757647
## Country                     10.75455836 -10.1499864           3.60542683
## Continent                   12.20619368 -11.4047230           5.05051723
```

```
## Hemisphere                          4.75242065  -5.0358854            2.02233946
##                                   MeanDecreaseGini
## Age                                      164.59631
## Sex                                       20.13407
## Cholesterol                              177.02194
## Systolic                                 169.01798
## Diastolic                                154.34702
## Heart.Rate                               162.06958
## Diabetes                                  21.71750
## Family.History                            22.57176
## Smoking                                   11.95256
## Obesity                                   22.70301
## Alcohol.Consumption                       22.08381
## Exercise.Hours.Per.Week                  188.83848
## Diet                                      51.11372
## Previous.Heart.Problems                   21.55484
## Medication.Use                            21.80187
## Stress.Level                             218.04148
## Sedentary.Hours.Per.Day                  188.92755
## Income                                   185.79115
## BMI                                      187.86912
## Triglycerides                            186.20824
## Physical.Activity.Days.Per.Week          172.95695
## Sleep.Hours.Per.Day                       86.42027
## Country                                  288.23884
## Continent                                 70.85541
## Hemisphere                                12.50816
```

```r
varImpPlot(rf_fit)
```




```r
rf_mod <- randomForest(Heart.Attack.Risk ~ Country + Income + Triglycerides +
                         Physical.Activity.Days.Per.Week + Systolic,
                       data = train_set, importance = TRUE)
```

```r
# training
rf_pred_tr <- predict(rf_mod, train_set, type = "response")
rf_confmat_tr <- confusionMatrix(rf_pred_tr, train_set$Heart.Attack.Risk)
accura_tr <- rf_confmat_tr$overall['Accuracy']
error_tr <- 1 - accura_tr
sn_tr <- rf_confmat_tr$byClass['Sensitivity']
sp_tr <- rf_confmat_tr$byClass['Specificity']

# testing
rf_pred_te <- predict(rf_mod, test_set, type = "response")
rf_confmat_te <- confusionMatrix(rf_pred_te, test_set$Heart.Attack.Risk)
accura_te <- rf_confmat_te$overall['Accuracy']
error_te <- 1 - accura_te
sn_te <- rf_confmat_te$byClass['Sensitivity']
sp_te <- rf_confmat_te$byClass['Specificity']

# output
df <- data.frame(Accuracy = c(accura_tr, accura_te),
               Error = c(error_tr, error_te),
               Sensitivity = c(sn_tr, sn_te),
               Specificity = c(sp_tr, sp_te))
```

```r
row.names(df) <- c("Train", "Test")
print(round(df, 3))
```

```
##       Accuracy Error Sensitivity Specificity
## Train    1.000 0.000       1.000       1.000
## Test     0.599 0.401       0.851       0.139
```

vi. LASSO

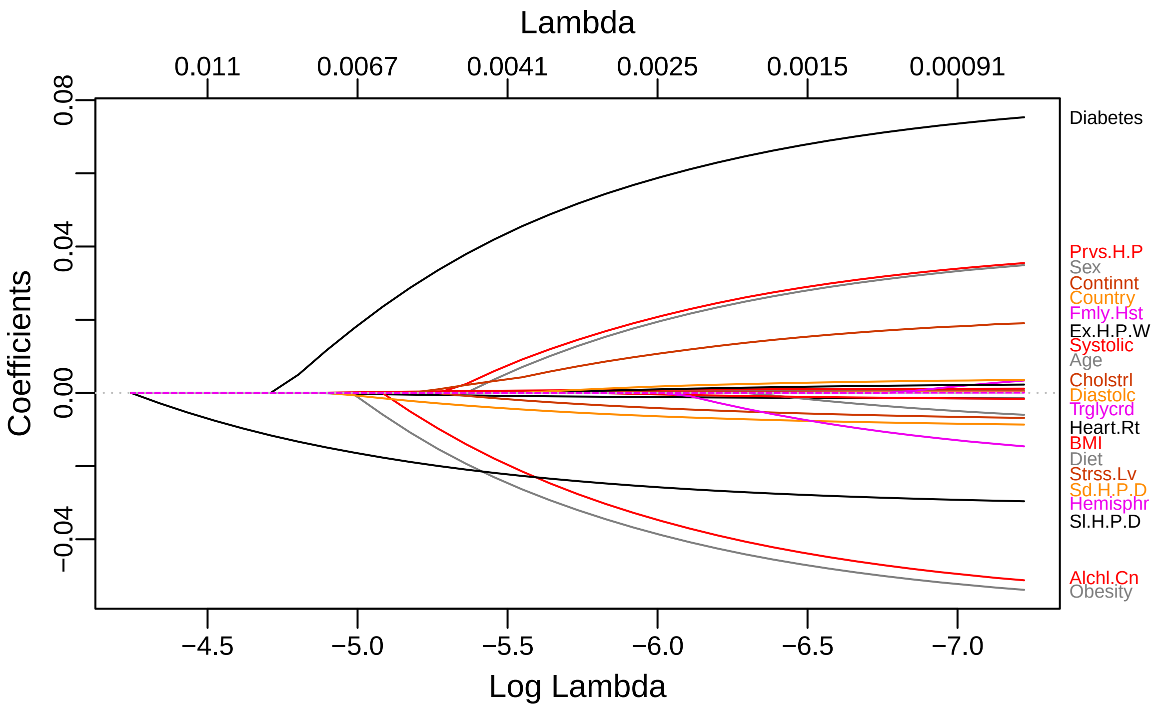
```r
library(glmnet)
library(caret)
x_tr <- as.matrix(train_set[, -26])
y_tr <- train_set[, 26]
x_te <- as.matrix(test_set[, -26])
y_te <- test_set[, 26]
std_fit <- preProcess(x_tr, method = c("center", "scale"))
```

```
## Warning in pre_process_options(method, column_types): The following
## pre-processing methods were eliminated: 'center', 'scale'
```

```r
x_tr_std <- predict(std_fit, x_tr)
x_te_std <- predict(std_fit, x_te)
```

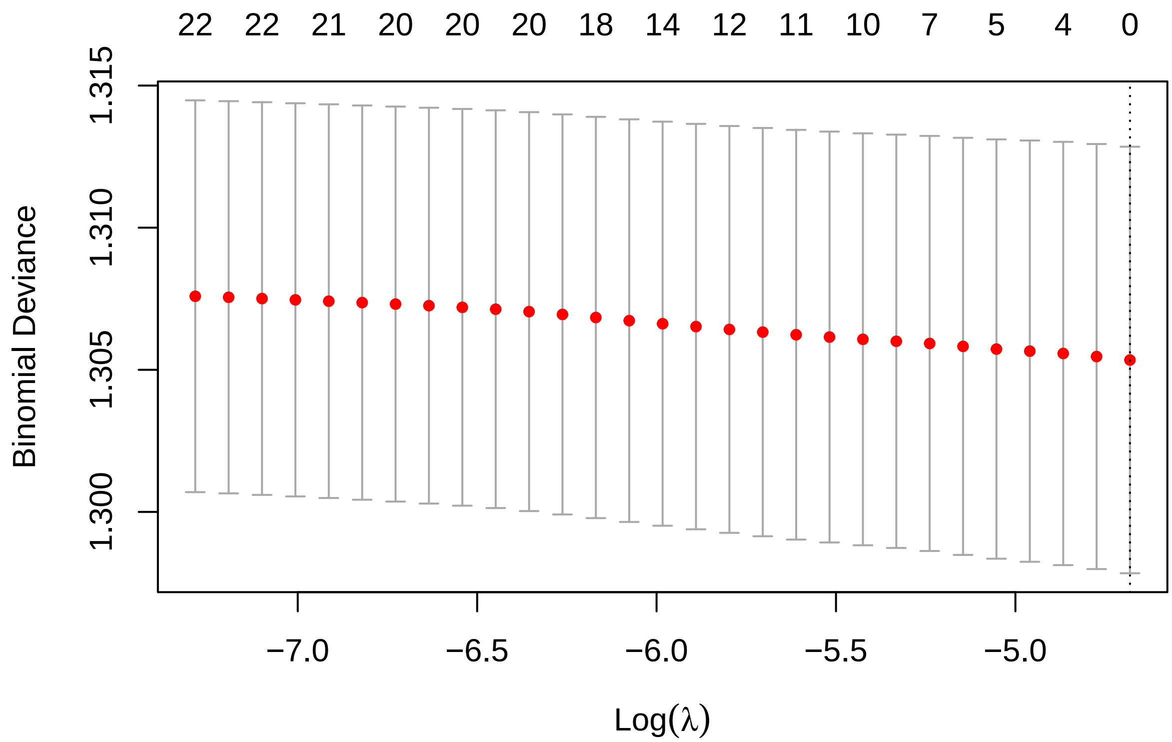
```r
x <- as.matrix(dataset[, -26])
y <- dataset[, 26]
std_fit <- preProcess(x, method = c("center", "scale"))
x_std <- predict(std_fit, x)
```

```r
fit_lasso <- glmnet(x_tr_std, as.numeric(y_tr), family = "binomial", alpha = 1)
library(plotmo)
plot_glmnet(fit_lasso, label = TRUE)
```




```r
x <- as.matrix(dataset[, -26])
y <- dataset[, 26]
std_fit <- preProcess(x, method = c("center", "scale"))
x_std <- predict(std_fit, x)
```

```r
cv_fit_lasso <- cv.glmnet(x_std, as.numeric(y), family = "binomial", alpha = 1)
plot(cv_fit_lasso)
```




```
best_lambda_lasso <- cv_fit_lasso$lambda.min
```

```
lasso_best <- glmnet(x_tr_std, as.numeric(y_tr), alpha = 1,
                     family = "binomial", lambda = best_lambda_lasso)
```

```
# training
lasso_pred_tr <- predict(lasso_best, x_tr_std, type = "class")
lasso_table_tr <- table(lasso_pred_tr, as.matrix(y_tr))
TP <- lasso_table_tr[1, 2]
FP <- lasso_table_tr[1, 1]
TN <- 0
FN <- 0
error_tr <- mean(lasso_pred_tr != y_tr)
accura_tr <- 1 - error_tr
sn_tr <- TP / TP + FN
sp_tr <- TN / TN + FP
```

```
# testing
lasso_pred_te <- predict(lasso_best, x_te_std, type = "class")
lasso_table_te <- table(lasso_pred_te, as.matrix(y_te))
TP <- lasso_table_te[1, 2]
FP <- lasso_table_te[1, 1]
TN <- 0
FN <- 0
error_te <- mean(lasso_pred_te != y_te)
```

```r
accura_te <- 1 - error_te
sn_te <- TP / TP + FN
sp_te <- TN / TN + FP
```

```r
# output
df <- data.frame(Accuracy = c(accura_tr, accura_te),
                 Error = c(error_tr, error_te),
                 Sensitivity = c(sn_tr, sn_te),
                 Specificity = c(sp_tr, sp_te))
row.names(df) <- c("Train", "Test")
print(round(df, 3))
```

```
##       Accuracy Error Sensitivity Specificity
## Train    0.360 0.640           1         NaN
## Test     0.353 0.647           1         NaN
```

vii. ROC

```r
logit <- glm(Heart.Attack.Risk ~ Cholesterol +
               Diabetes + Sleep.Hours.Per.Day,
             family = "binomial", data = train_set)
logit_pred <- predict(logit, test_set, type = "response")
logit_roc <- roc(test_set$Heart.Attack.Risk, logit_pred)
logit_auc <- auc(logit_roc)
```

```r
lda <- lda(Heart.Attack.Risk ~ Cholesterol +
               Diabetes + Sleep.Hours.Per.Day, data = train_set)
lda_pred <- predict(lda, test_set)$posterior[, 2]
lda_roc <- roc(test_set$Heart.Attack.Risk, lda_pred)
lda_auc <- auc(lda_roc)
```

```r
knn <- knn3(Heart.Attack.Risk ~ ., data = train_set_scaled, k = 100)
knn_pred <- predict(knn, newdata = test_set_scaled, type = "prob")
knn_roc <- roc(test_set_scaled$Heart.Attack.Risk, knn_pred[, 2])
knn_auc <- auc(knn_roc)
```

```r
rf <- randomForest(Heart.Attack.Risk ~ Country + Income + Triglycerides +
                     Physical.Activity.Days.Per.Week + Systolic,
                   data = train_set, importance = TRUE)
rf_pred <- predict(rf, test_set, type = "prob")
rf_roc <- roc(test_set$Heart.Attack.Risk, rf_pred[, 2])
rf_auc <- auc(rf_roc)
```

```r
lasso <- glmnet(x_tr_std, as.numeric(y_tr), alpha = 1,
                family = "binomial", lambda = best_lambda_lasso)
lasso_pred <- predict(lasso, x_te_std, type = "response")
lasso_roc <- roc(y_te, lasso_pred)
lasso_auc <- auc(lasso_roc)
```

```r
rocobjs <- list(Logistic = logit_roc,
                LDA = lda_roc,
                KNN = knn_roc,
                RandomForest = rf_roc,
                LASSO = lasso_roc)

methods_auc <- paste(c("Logistic", "LDA", "KNN", "Random Forest", "LASSO"),
                     "AUC = ",
                     round(c(logit_auc, lda_auc, knn_auc, rf_auc, lasso_auc), 3))

ggroc(rocobjs, size = 1, alpha = 0.5) +
  scale_color_discrete(labels = methods_auc)
```