Team 4 - Heart Disease Detection

Claire Bentzen, John Vincent Deniega, Ravita Kartawinata

2024-06-22

## Load Libraries

suppressPackageStartupMessages(library(caret))  
library(tidyr)  
suppressPackageStartupMessages(library(tidyverse))  
library(gt)  
library(dplyr)  
library(tibble)  
suppressPackageStartupMessages(library(pROC))  
suppressPackageStartupMessages(library(ggplot2))  
suppressPackageStartupMessages(library(corrplot))  
library(ggplot2)  
suppressPackageStartupMessages(library(gridExtra))

## Data Preprocessing

seed <- 123  
#Ingest  
data <- read.csv("heart.csv") #Change to your respective local path

#Check for missing columns  
missing\_col <- colSums(is.na(data))  
cat('No missing values found: \n')

No missing values found:

missing\_col

age sex cp trestbps chol fbs restecg thalach   
 0 0 0 0 0 0 0 0   
 exang oldpeak slope ca thal target   
 0 0 0 0 0 0

#Check for duplicate rows  
duplicate\_row <- data[duplicated(data),]  
cat('Count of duplicate rows: ', nrow(duplicate\_row),'\n')

Count of duplicate rows: 723

data1 <- data[!duplicated(data),]  
cat('NewData dimension: ',nrow(data1),  
 'remaining rows. This is still sufficient since ncol^2 is less than #nrows\n')

NewData dimension: 302 remaining rows. This is still sufficient since ncol^2 is less than #nrows

# Clipping outliers to 1.5 times greater than minimum Q1 or   
# maximum Q3 quartile prior to center and scaling so as not to   
# excessively affect mean and standard deviation  
cont\_pred <- c('age', 'chol', 'oldpeak', 'thalach', 'trestbps')  
cont\_data\_toclip <- data1[, cont\_pred]  
max(cont\_data\_toclip$chol) # test chol max = 564

[1] 564

clip\_outlier <- function(x){  
 q1 <- quantile(x, .25, na.rm = TRUE)  
 q3 <- quantile(x, .75, na.rm = TRUE)  
 IQR <- q3 - q1  
 lower <- q1 - 1.5 \* IQR  
 upper <- q3 + 1.5 \* IQR  
 x <- ifelse(x < lower, lower, x)  
 x <- ifelse(x > upper, upper, x)  
 return(x)  
}  
cont\_data\_clipped <- cont\_data\_toclip |>   
 mutate(across(everything(), clip\_outlier))  
max(cont\_data\_clipped) #test chol max = 370.375

[1] 370.375

summary(data1)

age sex cp trestbps   
 Min. :29.00 Min. :0.0000 Min. :0.0000 Min. : 94.0   
 1st Qu.:48.00 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:120.0   
 Median :55.50 Median :1.0000 Median :1.0000 Median :130.0   
 Mean :54.42 Mean :0.6821 Mean :0.9636 Mean :131.6   
 3rd Qu.:61.00 3rd Qu.:1.0000 3rd Qu.:2.0000 3rd Qu.:140.0   
 Max. :77.00 Max. :1.0000 Max. :3.0000 Max. :200.0   
 chol fbs restecg thalach   
 Min. :126.0 Min. :0.000 Min. :0.0000 Min. : 71.0   
 1st Qu.:211.0 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:133.2   
 Median :240.5 Median :0.000 Median :1.0000 Median :152.5   
 Mean :246.5 Mean :0.149 Mean :0.5265 Mean :149.6   
 3rd Qu.:274.8 3rd Qu.:0.000 3rd Qu.:1.0000 3rd Qu.:166.0   
 Max. :564.0 Max. :1.000 Max. :2.0000 Max. :202.0   
 exang oldpeak slope ca   
 Min. :0.0000 Min. :0.000 Min. :0.000 Min. :0.0000   
 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:1.000 1st Qu.:0.0000   
 Median :0.0000 Median :0.800 Median :1.000 Median :0.0000   
 Mean :0.3278 Mean :1.043 Mean :1.397 Mean :0.7185   
 3rd Qu.:1.0000 3rd Qu.:1.600 3rd Qu.:2.000 3rd Qu.:1.0000   
 Max. :1.0000 Max. :6.200 Max. :2.000 Max. :4.0000   
 thal target   
 Min. :0.000 Min. :0.000   
 1st Qu.:2.000 1st Qu.:0.000   
 Median :2.000 Median :1.000   
 Mean :2.315 Mean :0.543   
 3rd Qu.:3.000 3rd Qu.:1.000   
 Max. :3.000 Max. :1.000

data1$age <- cont\_data\_clipped$age  
data1$trestbps <-cont\_data\_clipped$trestbps  
data1$chol <- cont\_data\_clipped$chol  
data1$thalach <- cont\_data\_clipped$thalach  
data1$oldpeak <- cont\_data\_clipped$oldpeak  
summary(data1)

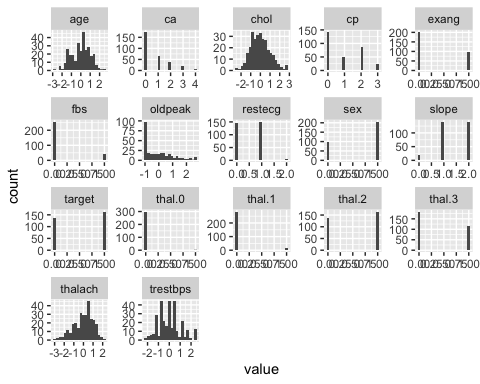
age sex cp trestbps   
 Min. :29.00 Min. :0.0000 Min. :0.0000 Min. : 94.0   
 1st Qu.:48.00 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:120.0   
 Median :55.50 Median :1.0000 Median :1.0000 Median :130.0   
 Mean :54.42 Mean :0.6821 Mean :0.9636 Mean :131.3   
 3rd Qu.:61.00 3rd Qu.:1.0000 3rd Qu.:2.0000 3rd Qu.:140.0   
 Max. :77.00 Max. :1.0000 Max. :3.0000 Max. :170.0   
 chol fbs restecg thalach   
 Min. :126.0 Min. :0.000 Min. :0.0000 Min. : 84.12   
 1st Qu.:211.0 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:133.25   
 Median :240.5 Median :0.000 Median :1.0000 Median :152.50   
 Mean :245.4 Mean :0.149 Mean :0.5265 Mean :149.61   
 3rd Qu.:274.8 3rd Qu.:0.000 3rd Qu.:1.0000 3rd Qu.:166.00   
 Max. :370.4 Max. :1.000 Max. :2.0000 Max. :202.00   
 exang oldpeak slope ca   
 Min. :0.0000 Min. :0.000 Min. :0.000 Min. :0.0000   
 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:1.000 1st Qu.:0.0000   
 Median :0.0000 Median :0.800 Median :1.000 Median :0.0000   
 Mean :0.3278 Mean :1.028 Mean :1.397 Mean :0.7185   
 3rd Qu.:1.0000 3rd Qu.:1.600 3rd Qu.:2.000 3rd Qu.:1.0000   
 Max. :1.0000 Max. :4.000 Max. :2.000 Max. :4.0000   
 thal target   
 Min. :0.000 Min. :0.000   
 1st Qu.:2.000 1st Qu.:0.000   
 Median :2.000 Median :1.000   
 Mean :2.315 Mean :0.543   
 3rd Qu.:3.000 3rd Qu.:1.000   
 Max. :3.000 Max. :1.000

#Center and scale continuous variables  
pre\_proc <- preProcess(data1[c("age",   
 "trestbps",   
 "chol",   
 "thalach",   
 "oldpeak")], method = c("center",   
 "scale"))  
data2 <- data1  
data2[c("age", "trestbps", "chol", "thalach", "oldpeak")] <-   
 predict(pre\_proc, data1[c("age",   
 "trestbps",   
 "chol",   
 "thalach",   
 "oldpeak")])  
  
#Since "thal" is not ordinal, but categorical, make dummy variables  
data3 <- data2  
data3$thal <- factor(data3$thal)  
dummy <- dummyVars(~ thal, data = data3)  
dummy\_col <- predict(dummy, newdata = data3)  
data3 <- cbind(data3, dummy\_col)  
data3$thal <- NULL # Drop the "thal" column now that we have dummy variables  
cat('NewData dimension: ',nrow(data3),'remaining rows. This is still sufficient since ncol^2 is less than #nrows \n')

NewData dimension: 302 remaining rows. This is still sufficient since ncol^2 is less than #nrows

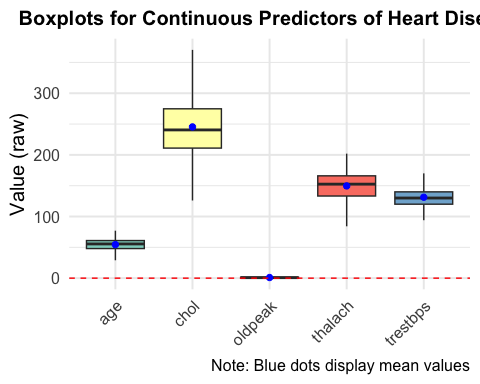
## Exploratory Data Analysis

#Check column histograms for unusual distributions  
ggplot(gather(data3), aes(value)) +  
 geom\_histogram(bins = 20) +  
 facet\_wrap(~key, scales = "free")

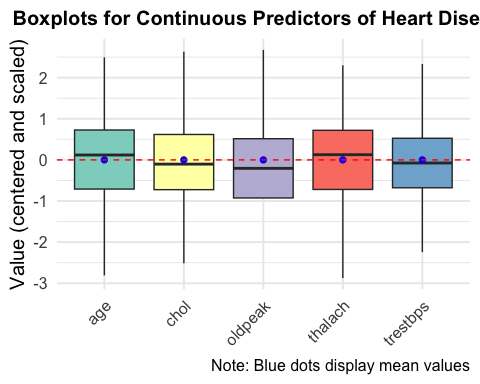


The predictors age and thalach are very slightly right skewed while the predictor oldpeak is left skewed. The predictor chol is approximately normally distributed.

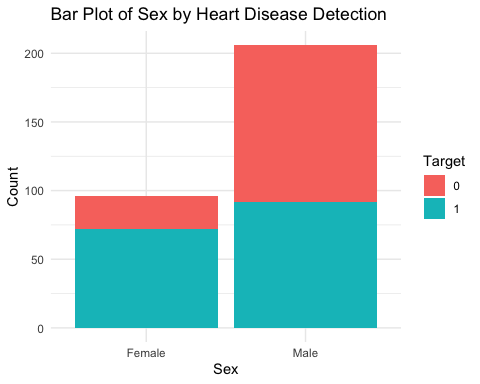
cont\_pred <- c('age', 'chol', 'oldpeak', 'thalach', 'trestbps')  
cont\_data <- data1[, cont\_pred]  
cont\_data\_long <- gather(cont\_data)  
  
# boxplots  
ggplot(cont\_data\_long, aes(x = key, y = value, fill = key)) +  
 geom\_boxplot(outlier.color = "red", outlier.shape = 1) +  
 stat\_summary(fun = mean,   
 geom = "point",   
 shape = 20,   
 size = 3,   
 color = "blue",   
 fill = "blue") +  
 labs(title = "Boxplots for Continuous Predictors of Heart Disease",  
 x = "Predictor",  
 y = "Value (raw)",  
 caption = "Note: Blue dots display mean values") +  
 theme\_minimal(base\_size = 15) +  
 theme(plot.title = element\_text(hjust = 0.5,   
 size = 15,   
 face = "bold"),  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 axis.title.x = element\_blank(),  
 legend.position = "none") +  
 scale\_fill\_brewer(palette = "Set3") +  
 suppressWarnings(  
 geom\_hline(yintercept = 0,   
 color = "red",   
 size = .5,   
 linetype = "dashed"))



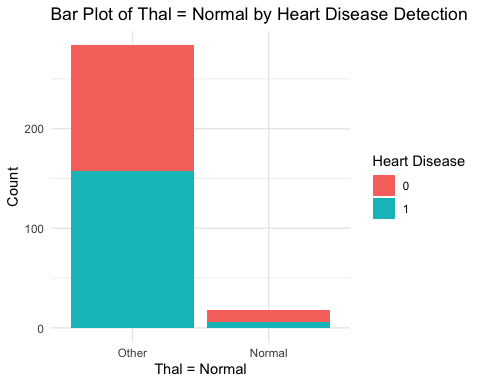
# continuous predictors  
cont\_pred <- c('age', 'chol', 'oldpeak', 'thalach', 'trestbps')  
cont\_data <- data3[, cont\_pred]  
cont\_data\_long <- gather(cont\_data) #   
ex\_cont\_data <- data3[,!names(data3) %in% cont\_pred]  
# boxplots  
ggplot(cont\_data\_long, aes(x = key, y = value, fill = key)) +  
 geom\_boxplot(outlier.color = "red", outlier.shape = 1) +  
 stat\_summary(fun = mean,   
 geom = "point",   
 shape = 20,   
 size = 3,   
 color = "blue",   
 fill = "blue") +  
 labs(title = "Boxplots for Continuous Predictors of Heart Disease",  
 x = "Predictor",  
 y = "Value (centered and scaled)",  
 caption = "Note: Blue dots display mean values") +  
 theme\_minimal(base\_size = 15) +  
 theme(plot.title = element\_text(hjust = 0.5,   
 size = 15,   
 face = "bold"),  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 axis.title.x = element\_blank(),  
 legend.position = "none") +  
 scale\_fill\_brewer(palette = "Set3") +  
 geom\_hline(yintercept = 0,   
 color = "red",   
 size = .5,   
 linetype = "dashed")



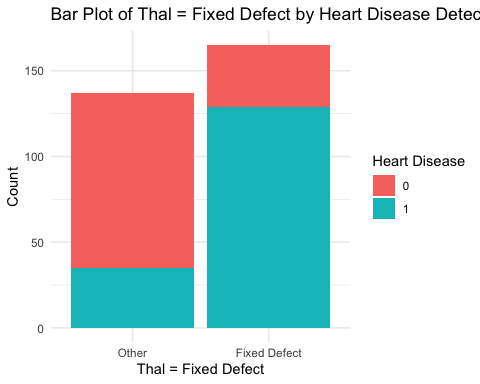
data3$target <- as.factor(data3$target)  
data3$sex <- as.factor(data3$sex)  
# stacked bar plot for heart disease detection in males vs females  
  
ggplot(data3, aes(x = sex, fill = target)) +  
 geom\_bar() +  
 labs(title = "Bar Plot of Sex by Heart Disease Detection",  
 x = "Sex",  
 y = "Count",  
 fill = "Target") +  
 scale\_x\_discrete(labels = c("0" = "Female", "1" = "Male")) +  
 theme\_minimal()



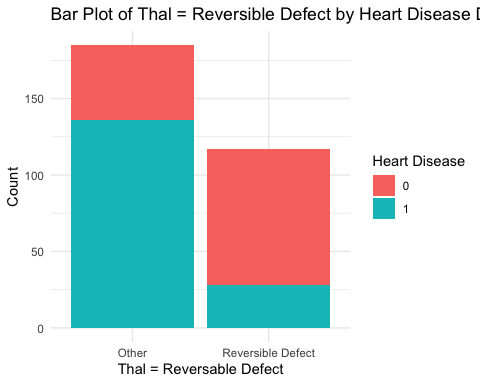
data3$target <- as.factor(data3$target)  
data3$thal.1 <- as.factor(data3$thal.1)  
data3$thal.2 <- as.factor(data3$thal.2)  
data3$thal.3 <- as.factor(data3$thal.3)  
  
# stacked bar plot for heart disease detection in thal.1  
ggplot(data3, aes(x = thal.1, fill = target)) +  
 geom\_bar() +  
 labs(title = "Bar Plot of Thal = Normal by Heart Disease Detection",  
 x = "Thal = Normal",  
 y = "Count",  
 fill = "Heart Disease") +  
 scale\_x\_discrete(labels = c("0" = "Other", "1" = "Normal")) +  
 theme\_minimal()



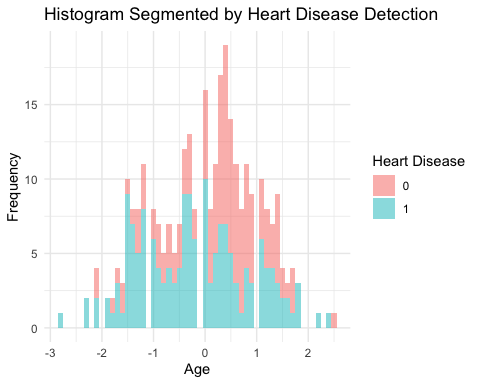
# stacked bar plot for heart disease detection in thal.2  
ggplot(data3, aes(x = thal.2, fill = target)) +  
 geom\_bar() +  
 labs(title = "Bar Plot of Thal = Fixed Defect by Heart Disease Detection",  
 x = "Thal = Fixed Defect",  
 y = "Count",  
 fill = "Heart Disease") +  
 scale\_x\_discrete(labels = c("0" = "Other",   
 "1" = "Fixed Defect")) +  
 theme\_minimal()



# stacked bar plot for heart disease detection in thal.3  
ggplot(data3, aes(x = thal.3, fill = target)) +  
 geom\_bar() +  
 labs(title = "Bar Plot of Thal = Reversible Defect by Heart Disease Detection",  
 x = "Thal = Reversable Defect",  
 y = "Count",  
 fill = "Heart Disease") +  
 scale\_x\_discrete(labels = c("0" = "Other", "1" = "Reversible Defect")) +  
 theme\_minimal()



# histogram of age segmented by heart disease detection  
ggplot(data3, aes(x = age, fill = target)) +  
 geom\_histogram(binwidth = .1, position = "stack", alpha = 0.5) +  
 labs(title = "Histogram Segmented by Heart Disease Detection",  
 x = "Age",  
 y = "Frequency",  
 fill = "Heart Disease") +  
 theme\_minimal()



#Check for near-zero variance columns  
nzv <- nearZeroVar(data3)  
cat('Removed near zero predictor: ', colnames(data3)[nzv],'\n')

Removed near zero predictor: thal.0

data4 <- data3[, -nearZeroVar(data3)]  
cat('NewData dimension: ',nrow(data4),'rows', ncol(data4), 'columns\n')

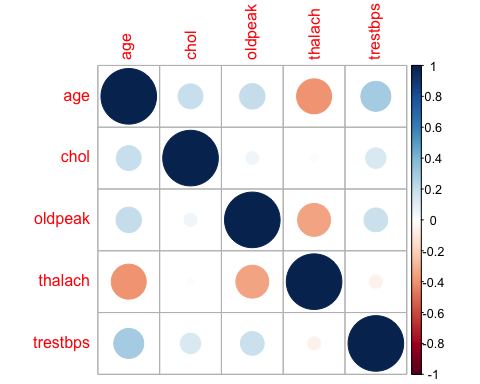
NewData dimension: 302 rows 16 columns

# Note: There appears to be an error in documentation  
# where thal is actually thal+1 category  
# Normal = 1  
# Fixed Defect = 2  
# Reversable Defect = 3

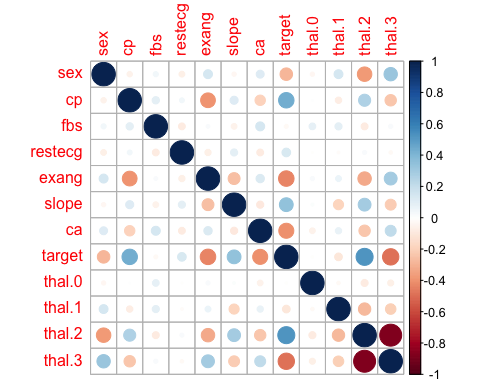
# remove highly correlated predictors  
p\_correlations <- cor(cont\_data, method = "pearson")  
p\_highCorr <- findCorrelation(p\_correlations, cutoff = .75)  
p\_highCorr # No continuous variables correlated > .75

integer(0)

ex\_cont\_data$sex <- as.integer(ex\_cont\_data$sex)  
ex\_cont\_data$thal.1 <- as.integer(ex\_cont\_data$thal.1)  
ex\_cont\_data$thal.2 <- as.integer(ex\_cont\_data$thal.2)  
ex\_cont\_data$thal.3 <- as.integer(ex\_cont\_data$thal.3)  
  
s\_correlations <- round(cor(ex\_cont\_data[, !names(ex\_cont\_data) %in% "target"],   
 method = "spearman"), 3)  
s\_highCorr <- findCorrelation(s\_correlations, cutoff = .75)  
data5 <- data4[, -s\_highCorr]  
  
# correlation plot among predictor variables  
p\_correlations <- cor(cont\_data)  
c\_correlations <- cor(ex\_cont\_data)  
p\_corr\_plot <- corrplot(p\_correlations)



c\_corr\_plot <- corrplot(c\_correlations)



There are no high correlations among the continuous predictors. The correlation plot suggests that there is a strong negative correlation between thal.2 and thal.3. These predictors represent fixed and reversible defects, respectively.

# logistic regression model with possible confounding predictors  
confounding\_model <- glm(target ~ age + sex + thal.2 + thal.3,   
 data = data5,   
 family = "binomial")  
summary(confounding\_model)

Call:  
glm(formula = target ~ age + sex + thal.2 + thal.3, family = "binomial",   
 data = data5)  
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 0.1933 0.5670 0.341 0.733192   
age -0.5142 0.1476 -3.484 0.000494 \*\*\*  
sex1 -0.8214 0.3330 -2.466 0.013651 \*   
thal.21 1.5569 0.5280 2.949 0.003189 \*\*   
thal.31 -0.6236 0.5311 -1.174 0.240334   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for binomial family taken to be 1)  
  
 Null deviance: 416.42 on 301 degrees of freedom  
Residual deviance: 311.46 on 297 degrees of freedom  
AIC: 321.46  
  
Number of Fisher Scoring iterations: 4

The low p-values for age, sex, and thal.2 indicate that these are possible confounding variables, however it does not confirm it. We have to now look at the correlation between the predictors and the target variable.

# correlation between possible confounders and target  
data\_conf\_check <- data5  
  
data\_conf\_check$target <- as.integer(data\_conf\_check$target)  
data\_conf\_check$age <- as.integer(data\_conf\_check$age)  
data\_conf\_check$sex <- as.integer(data\_conf\_check$sex)  
data\_conf\_check$thal.2 <- as.integer(data\_conf\_check$thal.2)  
cor(data\_conf\_check[c("target", "age", "sex", "thal.2")])

target age sex thal.2  
target 1.0000000 -0.13817219 -0.28360936 0.52602967  
age -0.1381722 1.00000000 -0.07318834 -0.07977954  
sex -0.2836094 -0.07318834 1.00000000 -0.37922291  
thal.2 0.5260297 -0.07977954 -0.37922291 1.00000000

Age and sex have a very low correlation with the target variable, so we can keep them. The correlation thal.2 has with the target variable is moderate, but not high enough to indicate that there might be significant changes to the outcome of the model if we keep it.

## Data Splitting

set.seed(seed)  
y <- data5$target  
x <- data5[, !names(data5) %in% "target"] # Predictors only  
  
trainingRows <- createDataPartition(y, p = .80, list = FALSE)  
train\_y <- y[trainingRows]  
test\_y <- y[-trainingRows]  
train\_x <- x[trainingRows, ]   
test\_x <- x[-trainingRows, ]  
train\_x <- data.frame(lapply(train\_x, function(x)   
 if(is.factor(x)) as.numeric(as.character(x)) else x))  
cat('Number of training sample:', nrow(train\_x), 'and test samples: ',nrow(test\_x), 'number of predictors:', ncol(train\_x))

Number of training sample: 243 and test samples: 59 number of predictors: 14

## Modeling

# Evaluation Metric  
sens\_spec\_harm <- function(data, lev = NULL, model = NULL) {  
 sens <- sensitivity(data$pred,   
 data$obs,   
 positive = levels(data$obs)[1])  
 spec <- specificity(data$pred,   
 data$obs,   
 positive = levels(data$obs)[1])  
 harmonic <- (2 \* sens \* spec) / (sens + spec)  
 suppressMessages({  
 roc <- roc(response = data$obs,   
 predictor = as.numeric(data$pred),   
 levels = rev(levels(data$obs)))  
 })  
 auc <- auc(roc)  
   
 c(harmonic = harmonic,   
 sensitivity = sens,   
 specificity = spec,  
 auc = as.numeric(auc))  
}  
  
ctrl <- trainControl(method = "repeatedcv", repeats = 5,  
 summaryFunction = sens\_spec\_harm,  
 classProbs = TRUE,  
 savePredictions = TRUE)  
  
tunegrid <- expand.grid(alpha = c(0, .1, .2, .4, .6, .8, 1),  
 lambda = seq(.01, .2, length = 10))

levels(train\_y) <- make.names(levels(train\_y))  
# Logistic Regression  
LR\_model <-suppressWarnings(  
 train(x = train\_x, y = train\_y,  
 method = "glm",  
 #tuneGrid = tunegrid,  
 preProc = c("center", "scale"),  
 metric = "sens\_spec\_harm",  
 trControl = ctrl))  
  
#linear discriminant  
set.seed(476)  
LDA\_model <-suppressWarnings(  
 train(x = train\_x, y = train\_y,  
 method = "lda",  
 preProc = c("center", "scale"),  
 metric = "sens\_spec\_harm",  
 trControl = ctrl))  
#penalized logistic regression  
set.seed(476)  
PLR\_model <- suppressWarnings(  
 train(x = train\_x, y = train\_y,  
 method = "glmnet",  
 tuneGrid = tunegrid,  
 preProc = c("center", "scale"),  
 metric = "sens\_spec\_harm",  
 trControl = ctrl))  
#nearest shrunken centroids  
set.seed(476)  
tunegrid <- expand.grid(threshold = seq(0, 25, length = 30))  
NSC\_model <-suppressWarnings(  
 train(x = train\_x, y = train\_y,  
 method = "pam",  
 preProc = c("center", "scale"),  
 tuneGrid = tunegrid,  
 metric = "sens\_spec\_harm",  
 trControl = ctrl))

111111111111111111111111111111111111111111111111111

nnetGrid <- expand.grid(decay = c(0, 0.01, .1),   
 size = c(3, 7, 11, 13))  
  
# Neural Network  
set.seed(476)  
nn\_model <- suppressWarnings(train(x = train\_x, y = train\_y,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = ctrl,  
 preProc = c("center", "scale"),  
 metric = "sens\_spec\_harm",  
 linout = FALSE,  
 trace = FALSE))  
  
# Support Vector Machine   
set.seed(476)  
suppressWarnings({  
svm\_model <- train(x = train\_x, y = train\_y,  
 method = "svmRadial",  
 preProc = c("center", "scale"),  
 metric = "sens\_spec\_harm",  
 tuneLength = 14,  
 trControl = ctrl)  
})  
# k-Nearest Neighbors  
set.seed(476)  
suppressWarnings({  
knn\_model <- train(x = train\_x, y = train\_y,  
 method = "knn",  
 preProc = c("center", "scale"),  
 metric = "sens\_spec\_harm",  
 tuneGrid = data.frame(k = 1:20),  
 trControl = ctrl)  
})

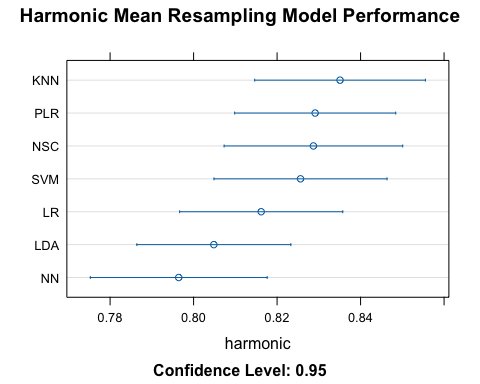
## Model Validation and Performance

#Harmonic  
LR\_sens\_spec <- LR\_model$results$harmonic  
LDA\_sens\_spec <- LDA\_model$results$harmonic  
PLR\_sens\_spec <- mean(PLR\_model$results$harmonic)  
NSC\_sens\_spec <- mean(NSC\_model$results$harmonic)  
  
# Using values of models optimized for "harmonic"  
NN\_sens\_spec <- max(nn\_model$results$harmonic) #size = 13; decay = .1  
SVM\_sens\_spec <- max(svm\_model$results$harmonic) #S=.05041146; C=.25  
KNN\_sens\_spec <- max(knn\_model$results$harmonic) #k=20  
  
sens\_spec\_values <- data.frame(  
 Model = c("LogisticRegression",   
 "LinearDiscriminant",  
 "PenalizedLogisticRegression",  
 "NearestShrunkenCentroid",  
 "Neural Net",  
 "Support Vector Machine",  
 "KNN"),  
 F\_Score\_Sens\_Spec = c(LR\_sens\_spec,   
 LDA\_sens\_spec,   
 PLR\_sens\_spec,  
 NSC\_sens\_spec,  
 NN\_sens\_spec,  
 SVM\_sens\_spec,  
 KNN\_sens\_spec)  
)  
  
#Confusion Matrices  
LR\_CM <- confusionMatrix(LR\_model, norm="none")  
LDA\_CM <- confusionMatrix(LDA\_model, norm="none")  
PLR\_CM <- confusionMatrix(PLR\_model, norm="none")  
NSC\_CM <- confusionMatrix(NSC\_model, norm="none")  
NN\_CM <- confusionMatrix(nn\_model, norm="none")  
SVM\_CM <- confusionMatrix(svm\_model, norm="none")  
KNN\_CM <- confusionMatrix(knn\_model, norm="none")  
  
#ROC-AUC --> added auc field  
LR\_auc <- LR\_model$results$auc  
LDA\_auc <- LDA\_model$results$auc  
PLR\_auc <- mean(PLR\_model$results$auc)  
NSC\_auc <- mean(NSC\_model$results$auc)  
NN\_auc <- mean(nn\_model$results$auc)  
SVM\_auc <- mean(svm\_model$results$auc)  
KNN\_auc <- mean(knn\_model$results$auc)  
  
  
Model\_performance <- data.frame(  
 Model = c("LogisticRegression",   
 "LinearDiscriminant",  
 "PenalizedLogisticRegression",  
 "NearestShrunkenCentroid",  
 "Neural Net",  
 "Support Vector Machine",  
 "KNN"),  
 Accuracy = c(  
 sum(diag(LR\_CM$table))/ sum(LR\_CM$table),  
 sum(diag(LDA\_CM$table))/ sum(LDA\_CM$table),  
 sum(diag(PLR\_CM$table))/ sum(PLR\_CM$table),  
 sum(diag(NSC\_CM$table))/ sum(NSC\_CM$table),  
 sum(diag(NN\_CM$table))/ sum(NN\_CM$table),  
 sum(diag(SVM\_CM$table))/ sum(SVM\_CM$table),  
 sum(diag(KNN\_CM$table))/ sum(KNN\_CM$table)  
 ),  
 AUC = c(LR\_auc, LDA\_auc, PLR\_auc, NSC\_auc, NN\_auc, SVM\_auc, KNN\_auc),  
 Harmonic = c(LR\_sens\_spec,   
 LDA\_sens\_spec,   
 PLR\_sens\_spec,   
 NSC\_sens\_spec,  
 NN\_sens\_spec,  
 SVM\_sens\_spec,  
 KNN\_sens\_spec)  
)  
#Sort by optimal detection of all disease cases  
# while balancing/minimizing false negatives  
Model\_performance |> arrange(desc(Harmonic))

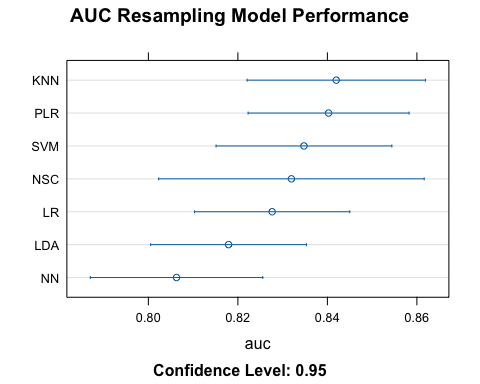
Model Accuracy AUC Harmonic  
1 KNN 0.8427984 0.8121395 0.8350820  
2 Support Vector Machine 0.8378601 0.7828974 0.8256088  
3 LogisticRegression 0.8312757 0.8276548 0.8161924  
4 PenalizedLogisticRegression 0.8444444 0.8197260 0.8094117  
5 LinearDiscriminant 0.8238683 0.8179179 0.8048535  
6 Neural Net 0.8074074 0.7806507 0.7964431  
7 NearestShrunkenCentroid 0.8460905 0.5391219 0.1172121

The KNN model performs the best on the training data.

# resample validation results  
model\_metrics <- resamples(list(  
 LDA = LDA\_model,  
 PLR = PLR\_model,  
 SVM = svm\_model,  
 KNN = knn\_model,  
 LR = LR\_model,  
 NSC = NSC\_model,  
 NN = nn\_model  
))  
  
# plot harmonic mean confidence intervals  
dotplot(model\_metrics,   
 metric = "harmonic",   
 main = "Harmonic Mean Resampling Model Performance")



# plot AUC confidence intervals  
dotplot(model\_metrics,   
 metric = "auc",   
 main = "AUC Resampling Model Performance")



# convert all predictors to numeric  
test\_x <- data.frame(lapply(test\_x, function(x)   
 if (is.factor(x)) as.numeric(as.character(x)) else x))  
  
levels(test\_y) <- make.names(levels(test\_y))  
   
# predict test data with each model   
test\_results <- data.frame(obs = test\_y,   
 LR = predict(LR\_model, test\_x))  
  
test\_results$LDA <- predict(LDA\_model, test\_x)  
  
test\_results$PLR <- predict(PLR\_model, test\_x)  
  
test\_results$NSC <- predict(NSC\_model, test\_x)  
  
test\_results$nnet <- predict(nn\_model, test\_x)  
test\_results$svm <- predict(svm\_model, test\_x)  
test\_results$knn <- predict(knn\_model, test\_x)

# model comparison using confusion matrix  
LR\_CM\_pred <- confusionMatrix(test\_results$LR,   
 test\_results$obs,   
 positive = "X1")  
LDA\_CM\_pred <- confusionMatrix(test\_results$LDA,   
 test\_results$obs,   
 positive = "X1")  
PLR\_CM\_pred <- confusionMatrix(test\_results$PLR,   
 test\_results$obs,   
 positive = "X1")  
NSC\_CM\_pred <- confusionMatrix(test\_results$NSC,   
 test\_results$obs,   
 positive = "X1")  
nnet\_CM\_pred <- confusionMatrix(test\_results$nnet,   
 test\_results$obs,   
 positive = "X1")  
svm\_CM\_pred <- confusionMatrix(test\_results$svm,   
 test\_results$obs,   
 positive = "X1")  
knn\_CM\_pred <- confusionMatrix(test\_results$knn,   
 test\_results$obs,   
 positive = "X1")  
  
#sens and spec are temporary variables to be overwritten with   
#each model for the purpose of saving result in their own   
# permanent variable  
sens <- LR\_CM\_pred$byClass['Sensitivity']  
spec <- LR\_CM\_pred$byClass['Specificity']  
LR\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
sens <- LDA\_CM\_pred$byClass['Sensitivity']  
spec <- LDA\_CM\_pred$byClass['Specificity']  
LDA\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
sens <- PLR\_CM\_pred$byClass['Sensitivity']  
spec <- PLR\_CM\_pred$byClass['Specificity']  
PLR\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
sens <- NSC\_CM\_pred$byClass['Sensitivity']  
spec <- NSC\_CM\_pred$byClass['Specificity']  
NSC\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
sens <- nnet\_CM\_pred$byClass['Sensitivity']  
spec <- nnet\_CM\_pred$byClass['Specificity']  
nnet\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
sens <- svm\_CM\_pred$byClass['Sensitivity']  
spec <- svm\_CM\_pred$byClass['Specificity']  
svm\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
sens <- knn\_CM\_pred$byClass['Sensitivity']  
spec <- knn\_CM\_pred$byClass['Specificity']  
knn\_harm\_pred <- (2 \* sens \* spec) / (sens + spec)  
  
#ROC  
LR\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$LR)))

Setting levels: control = X0, case = X1

Setting direction: controls < cases

LDA\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$LDA)))

Setting levels: control = X0, case = X1  
Setting direction: controls < cases

PLR\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$PLR)))

Setting levels: control = X0, case = X1  
Setting direction: controls < cases

NSC\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$NSC)))

Setting levels: control = X0, case = X1  
Setting direction: controls < cases

nnet\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$nnet)))

Setting levels: control = X0, case = X1  
Setting direction: controls < cases

svm\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$svm)))

Setting levels: control = X0, case = X1  
Setting direction: controls < cases

knn\_roc\_pred <- suppressWarnings(roc(test\_results$obs,   
 as.numeric(test\_results$knn)))

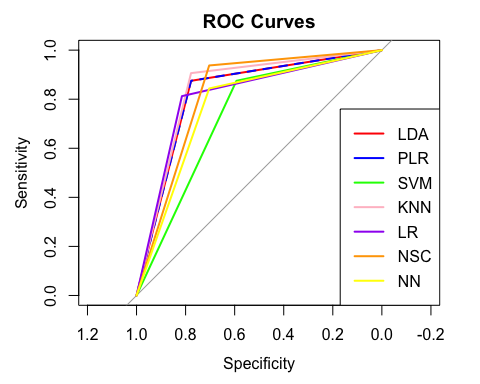
Setting levels: control = X0, case = X1  
Setting direction: controls < cases

Test\_performance <- data.frame(  
 Model = c("LogisticRegression", "LinearDiscriminant",  
 "PenalizedLogisticRegression", "NearestShrunken",  
 "NeutralNetwork","SVM","kNN"),  
 Accuracy = c(  
 LR\_CM\_pred$overall['Accuracy'],  
 LDA\_CM\_pred$overall['Accuracy'],  
 PLR\_CM\_pred$overall['Accuracy'],   
 NSC\_CM\_pred$overall['Accuracy'],  
 nnet\_CM\_pred$overall['Accuracy'],   
 svm\_CM\_pred$overall['Accuracy'],   
 knn\_CM\_pred$overall['Accuracy']  
 ),  
 AUC = c(  
 LR\_roc\_pred$auc,  
 LDA\_roc\_pred$auc,  
 PLR\_roc\_pred$auc,  
 NSC\_roc\_pred$auc,  
 nnet\_roc\_pred$auc,  
 svm\_roc\_pred$auc,  
 knn\_roc\_pred$auc  
 ),  
 Harmonic = c(LR\_harm\_pred,   
 LDA\_harm\_pred,   
 PLR\_harm\_pred,   
 NSC\_harm\_pred,  
 nnet\_harm\_pred,   
 svm\_harm\_pred,   
 knn\_harm\_pred)  
)  
Test\_performance |> arrange(desc(Harmonic))

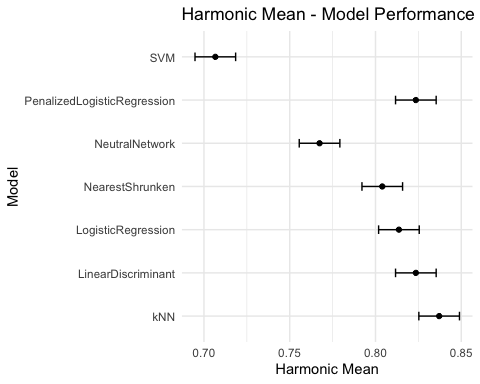
Model Accuracy AUC Harmonic  
1 kNN 0.8474576 0.8420139 0.8371134  
2 LinearDiscriminant 0.8305085 0.8263889 0.8235294  
3 PenalizedLogisticRegression 0.8305085 0.8263889 0.8235294  
4 LogisticRegression 0.8135593 0.8136574 0.8136558  
5 NearestShrunken 0.8305085 0.8206019 0.8039492  
6 NeutralNetwork 0.7796610 0.7737269 0.7673897  
7 SVM 0.7457627 0.7337963 0.7066246

The KNN model performs the best on the test data. This model is selected as the optimal model.

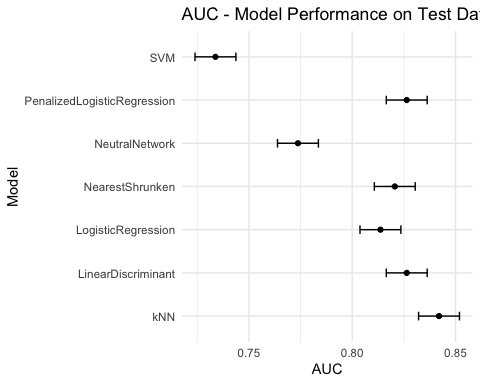
# compare ROC curves  
plot(LDA\_roc\_pred, col = "red", main = "ROC Curves", lty = 1)  
lines(PLR\_roc\_pred, col = "blue", lty = 2)  
lines(svm\_roc\_pred, col = "green")  
lines(knn\_roc\_pred, col = "pink")  
lines(LR\_roc\_pred, col = "purple")  
lines(NSC\_roc\_pred, col = "orange")  
lines(nnet\_roc\_pred, col = "yellow")  
legend("bottomright",   
 legend = c("LDA", "PLR", "SVM", "KNN", "LR", "NSC", "NN"),   
 col = c("red", "blue", "green", "pink", "purple", "orange", "yellow"),   
 lwd = 2)



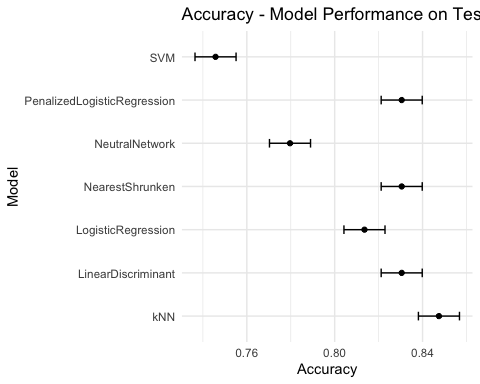
# calculate 95% confidence intervals for harmonic test results  
test\_harmonic <- Test\_performance[, c('Model', 'Harmonic')] %>%  
 mutate(  
 LowerCI = Harmonic - qt(0.975, df = nrow(test\_x) - 1) \*   
 (sd(Harmonic) / sqrt(nrow(test\_x))),  
 UpperCI = Harmonic + qt(0.975, df = nrow(test\_x) - 1) \*   
 (sd(Harmonic) / sqrt(nrow(test\_x)))  
 )  
  
# plot harmonic confidence intervals  
ggplot(test\_harmonic, aes(x = Harmonic, y = Model)) +  
 geom\_point() +  
 geom\_errorbar(aes(xmin = LowerCI, xmax = UpperCI), width = 0.2) +  
 labs(title = "Harmonic Mean - Model Performance on Test Data",  
 x = "Harmonic Mean",  
 y = "Model") +  
 theme\_minimal()



# calculate 95% confidence intervals for auc test results  
test\_auc <- Test\_performance[, c('Model', 'AUC')] %>%  
 mutate(  
 LowerCI = AUC - qt(0.975, df = nrow(test\_x) - 1) \*   
 (sd(AUC) / sqrt(nrow(test\_x))),  
 UpperCI = AUC + qt(0.975, df = nrow(test\_x) - 1) \*   
 (sd(AUC) / sqrt(nrow(test\_x)))  
 )  
  
# plot auc confidence intervals  
ggplot(test\_auc, aes(x = AUC, y = Model)) +  
 geom\_point() +  
 geom\_errorbar(aes(xmin = LowerCI, xmax = UpperCI), width = 0.2) +  
 labs(title = "AUC - Model Performance on Test Data",  
 x = "AUC",  
 y = "Model") +  
 theme\_minimal()



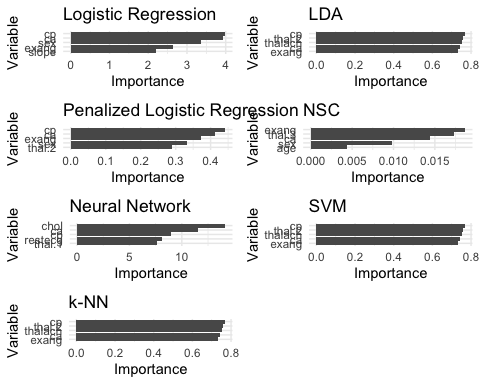
# calculate 95% confidence intervals for accuracy test results  
test\_accuracy <- Test\_performance[, c('Model', 'Accuracy')] %>%  
 mutate(  
 LowerCI = Accuracy - qt(0.975, df = nrow(test\_x) - 1) \*   
 (sd(Accuracy) / sqrt(nrow(test\_x))),  
 UpperCI = Accuracy + qt(0.975, df = nrow(test\_x) - 1) \*   
 (sd(Accuracy) / sqrt(nrow(test\_x)))  
 )  
  
# plot auc confidence intervals  
ggplot(test\_accuracy, aes(x = Accuracy, y = Model)) +  
 geom\_point() +  
 geom\_errorbar(aes(xmin = LowerCI, xmax = UpperCI), width = 0.2) +  
 labs(title = "Accuracy - Model Performance on Test Data",  
 x = "Accuracy",  
 y = "Model") +  
 theme\_minimal()



plots <- list()  
top\_vars <- list()  
  
model\_names <- c("LR\_model",   
 "LDA\_model",   
 "PLR\_model",   
 "NSC\_model",   
 "nn\_model",   
 "svm\_model",   
 "knn\_model")  
titles <- c("Logistic Regression",   
 "LDA",   
 "Penalized Logistic Regression",   
 "NSC",   
 "Neural Network",   
 "SVM",   
 "k-NN")  
  
for (i in seq\_along(model\_names)) {  
 model <- get(model\_names[i])  
 title <- titles[i]  
   
 imp\_var <- varImp(model, scale = FALSE)  
 imp\_var\_df <- as.data.frame(imp\_var$importance)  
 imp\_var\_df$Variable <- rownames(imp\_var$importance)  
   
 # Check if the 'Overall' column exists  
 if (!("Overall" %in% colnames(imp\_var\_df))) {  
 imp\_var\_df <- imp\_var\_df %>%  
 rowwise() %>%  
 mutate(Overall = mean(c\_across(starts\_with("X")),   
 na.rm = TRUE)) %>%  
 ungroup()  
 }  
   
 top5\_imp\_var <- imp\_var\_df %>%   
 arrange(desc(Overall)) %>% slice(1:5)  
   
 p <- ggplot(top5\_imp\_var,   
 aes(x = reorder(Variable, Overall),   
 y = Overall)) +  
 geom\_bar(stat = "identity") +  
 coord\_flip() +  
 ggtitle(title) +  
 theme\_minimal() +  
 labs(x = "Variable", y = "Importance")  
   
 plots[[i]] <- p  
 top\_vars[[i]] <- top5\_imp\_var$Variable  
  
}  
all\_top\_vars <- unlist(top\_vars)  
most\_common\_vars <- names(head(sort(table(all\_top\_vars),   
 decreasing = TRUE), 5))  
most\_common\_vars

[1] "ca" "cp" "exang" "thal.2" "sex"

do.call(grid.arrange, c(plots, ncol = 2))

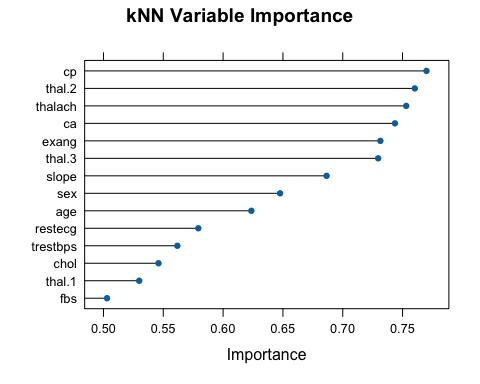


The most importance variables among all the models appear to be cp, ca, exang, thal.2, and sex.

## Optimal model

The KNN model was chosen as the optimal model due to the best performance in terms of AUC, harmonic mean between sensitivity and specificity, and accuracy.

# plot kNN variable importance  
plot(varImp(knn\_model, scale = FALSE), main = "kNN Variable Importance")



# kNN Test Performance  
Test\_performance[7, ]

Model Accuracy AUC Harmonic  
7 kNN 0.8474576 0.8420139 0.8371134