Analysis document

Shabbeer Hassan

2021-07-20

Contents

1 Data Source

This document uses UC Irvine's Machine Learning Repository: Heart Disease Data Set. This directory contains 4 databases concerning heart disease diagnosis. The data was collected from four different locations:

Cleveland Clinic Foundation (cleveland.data) Hungarian Institute of Cardiology, Budapest (hungarian.data) V.A. Medical Center, Long Beach, CA (long-beach-va.data) University Hospital, Zurich, Switzerland (switzerland.data)

For our purposes, we would take the Cleveland Clinic Foundation data. Cleveland Clinic Heart Disease dataset which contains 13 variables related to patient diagnostics and one outcome variable indicating the presence or absence of heart disease. The data was accessed from the UCI Machine Learning Repository. The goal is to be able to accurately classify patients whether they have heart disease or not based on diagnostic test data.

2 Question Asked

Which health biomarkers are considered to be the most important when trying to predict the presence of heart disease?

3 Load Libraries

```r
# Load in Packages
library("readr")
library("readxl")
library("dplyr")
library("tidyr")
library("ggplot2")
library("VIM")
library("class")
library("tidyverse")

# For analysis
library("keras")
library("RColorBrewer")
library("broom")
library("glmnet")
library("caret")
library("pROC")
library("MLeval")
library("vip")
library("rpart")
library("rpart.plot")
library("rattle")
library("viridis")
library("mltools")
library("randomForest")
library("psych") #matrix plots
library("corrplot") # used for correlation plot
library("ggpubr") #used for qqplots
library("gridExtra") #arranging qqplots in 1 graph
library("rpart")
library("rpart.plot")
library("gbm")
library("randomForest")
library("neuralnet")
library("e1071")
library("glmnet")
library("pROC")
library("PRROC")
library("RWeka")
library("performanceEstimation")
library("NeuralNetTools")
```

4 Wrangling Data

```
analysis_data <- read.table(url("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-diseas
                           sep = ",",
                           header = FALSE)
# Retrieve column names from: "https://archive.ics.uci.edu/ml/datasets/heart+disease"
colnames(analysis_data) <- c("Age",
                              "Sex",
                              "Chest_pain_type",
                              "Resting_bp",
                              "Cholesterol",
                              "Fasting_blood_sugar",
                              "Resting_ecg",
                              "Max_heart_rate",
                              "Exercise_induced_angina",
                              "ST_depression_by_exercise",
                              "Slope_peak_exercise_ST",
                              "Num_major_vessels_flouro",
                              "Thalassemia",
                              "Diagnosis_Heart_Disease")

# Check the dataset
glimpse(analysis_data) # Get str for all columns in neat fashion
```

```
## Rows: 303
## Columns: 14
## $ Age                       <dbl> 63, 67, 67, 37, 41, 56, 62, 57, 63, 53, 57, ~
## $ Sex                       <dbl> 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,~
## $ Chest_pain_type           <dbl> 1, 4, 4, 3, 2, 2, 4, 4, 4, 4, 4, 2, 3, 2, 3,~
## $ Resting_bp                <dbl> 145, 160, 120, 130, 130, 120, 140, 120, 130,~
## $ Cholesterol               <dbl> 233, 286, 229, 250, 204, 236, 268, 354, 254,~
## $ Fasting_blood_sugar       <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,~
## $ Resting_ecg               <dbl> 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 0,~
## $ Max_heart_rate            <dbl> 150, 108, 129, 187, 172, 178, 160, 163, 147,~
## $ Exercise_induced_angina   <dbl> 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,~
## $ ST_depression_by_exercise <dbl> 2.3, 1.5, 2.6, 3.5, 1.4, 0.8, 3.6, 0.6, 1.4,~
## $ Slope_peak_exercise_ST    <dbl> 3, 2, 2, 3, 1, 1, 3, 1, 2, 3, 2, 2, 2, 1, 1,~
## $ Num_major_vessels_flouro  <chr> "0.0", "3.0", "2.0", "0.0", "0.0", "0.0", "2~
## $ Thalassemia               <chr> "6.0", "3.0", "7.0", "3.0", "3.0", "3.0", "3~
## $ Diagnosis_Heart_Disease   <int> 0, 2, 1, 0, 0, 0, 3, 0, 2, 1, 0, 0, 2, 0, 0,~
```

```
dim(analysis_data) # Dimension of the dataset
```

```
## [1] 303  14
```

```
## As we see that some qualitative variables in the data frame are included as quantitative variables.
# Correcting column types
cols <- c("Sex", "Chest_pain_type", "Fasting_blood_sugar", "Resting_ecg", "Exercise_induced_angina",
          "Slope_peak_exercise_ST")
analysis_data <- analysis_data %>% mutate_at(cols, list(as.factor))

# Cheking the dataset again
glimpse(analysis_data)
```

```
## Rows: 303
## Columns: 14
## $ Age                       <dbl> 63, 67, 67, 37, 41, 56, 62, 57, 63, 53, 57, ~
## $ Sex                       <fct> 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,~
## $ Chest_pain_type           <fct> 1, 4, 4, 3, 2, 2, 4, 4, 4, 4, 4, 2, 3, 2, 3,~
## $ Resting_bp                <dbl> 145, 160, 120, 130, 130, 120, 140, 120, 130,~
## $ Cholesterol               <dbl> 233, 286, 229, 250, 204, 236, 268, 354, 254,~
## $ Fasting_blood_sugar       <fct> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,~
## $ Resting_ecg               <fct> 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 0,~
## $ Max_heart_rate            <dbl> 150, 108, 129, 187, 172, 178, 160, 163, 147,~
## $ Exercise_induced_angina   <fct> 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,~
## $ ST_depression_by_exercise <dbl> 2.3, 1.5, 2.6, 3.5, 1.4, 0.8, 3.6, 0.6, 1.4,~
## $ Slope_peak_exercise_ST    <fct> 3, 2, 2, 3, 1, 1, 3, 1, 2, 3, 2, 2, 2, 1, 1,~
## $ Num_major_vessels_flouro  <chr> "0.0", "3.0", "2.0", "0.0", "0.0", "0.0", "2~
## $ Thalassemia               <chr> "6.0", "3.0", "7.0", "3.0", "3.0", "3.0", "3~
## $ Diagnosis_Heart_Disease   <int> 0, 2, 1, 0, 0, 0, 3, 0, 2, 1, 0, 0, 2, 0, 0,~
```

The main idea/goal here is to use the 13 variables to predict the Diagnosis_Heart_Disease variable, or in other words to predict the heart disease event.

4.1 Inspecting NA's if there

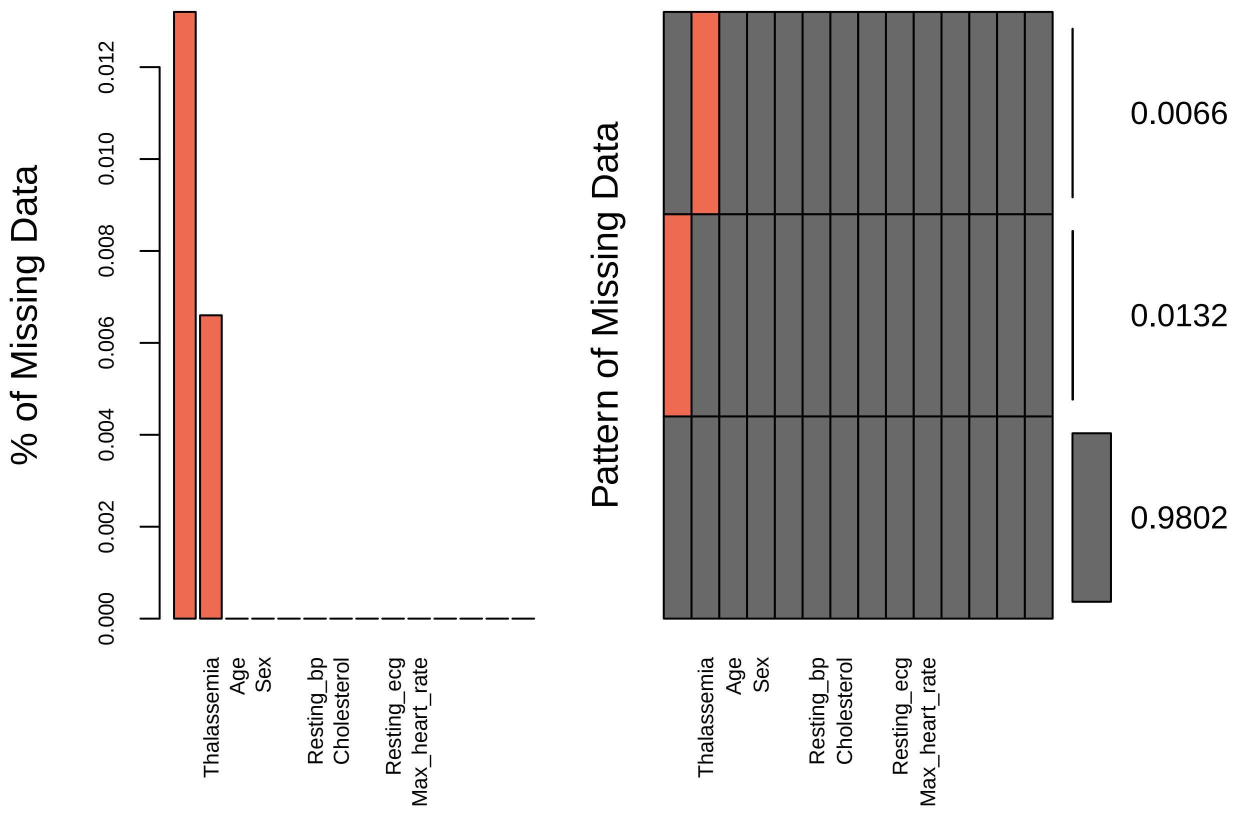
```
## Casual inspection suggests some NA's being present in the form of "?". We change them to NA
analysis_data[analysis_data == "?"] <- NA

# Now we can quantify the missing values
sum(is.na(analysis_data))
```

```
## [1] 6
```

```
analysis_data %>%
  purrr::map_df(~ sum(is.na(.)))
```

```
## # A tibble: 1 x 14
##     Age   Sex Chest_pain_type Resting_bp Cholesterol Fasting_blood_sugar
##   <int> <int>           <int>      <int>       <int>               <int>
## 1     0     0               0          0           0                   0
## # ... with 8 more variables: Resting_ecg <int>, Max_heart_rate <int>,
## #   Exercise_induced_angina <int>, ST_depression_by_exercise <int>,
## #   Slope_peak_exercise_ST <int>, Num_major_vessels_flouro <int>,
## #   Thalassemia <int>, Diagnosis_Heart_Disease <int>
```

```
# Visualizing NA Values
aggr(analysis_data, col=c("dimgrey", "coral2"), numbers=TRUE,
                   sortVars=TRUE, labels=names(analysis_data), cex.axis=0.7,
                   gap=3, ylab=c("% of Missing Data", "Pattern of Missing Data"))
```



```
##
##  Variables sorted by number of missings:
##                   Variable      Count
##   Num_major_vessels_flouro 0.01320132
##                Thalassemia 0.00660066
##                        Age 0.00000000
##                        Sex 0.00000000
##            Chest_pain_type 0.00000000
##                 Resting_bp 0.00000000
##                Cholesterol 0.00000000
##        Fasting_blood_sugar 0.00000000
##                Resting_ecg 0.00000000
##             Max_heart_rate 0.00000000
##    Exercise_induced_angina 0.00000000
##  ST_depression_by_exercise 0.00000000
##     Slope_peak_exercise_ST 0.00000000
##    Diagnosis_Heart_Disease 0.00000000
```

Either we replace the missing values with "mode" of that variable OR we can drop the entire individual containing the NA. Since only 4 rows in variable Number of coloured vessels and Thalessemia, we delete the rows. In ideal situation, we could have compared these contrasting situations - imputation by replacing NA with mode & dropping rows, and checked whether we gain or lose power when dropping rows/imputing. Another thing to note, is that imputation by "mode" is not the only method, we can use more complex methods for imputation as implemented in the pakcage MICE.

So we just drop rows containing NA

```
analysis_data <- analysis_data %>% drop_na()

paste("Number of rows left after dropping rows with NA: ", nrow(analysis_data))
```

```
## [1] "Number of rows left after dropping rows with NA:  297"
```

Now its important to know whether the dataset is balanced across the levels of dependent variable

```
analysis_data %>%
  drop_na() %>%
  group_by(Diagnosis_Heart_Disease) %>%
  count() %>%
  ungroup()
```

```
## # A tibble: 5 x 2
##   Diagnosis_Heart_Disease     n
##                     <int> <int>
## 1                       0   160
## 2                       1    54
## 3                       2    35
## 4                       3    35
## 5                       4    13
```

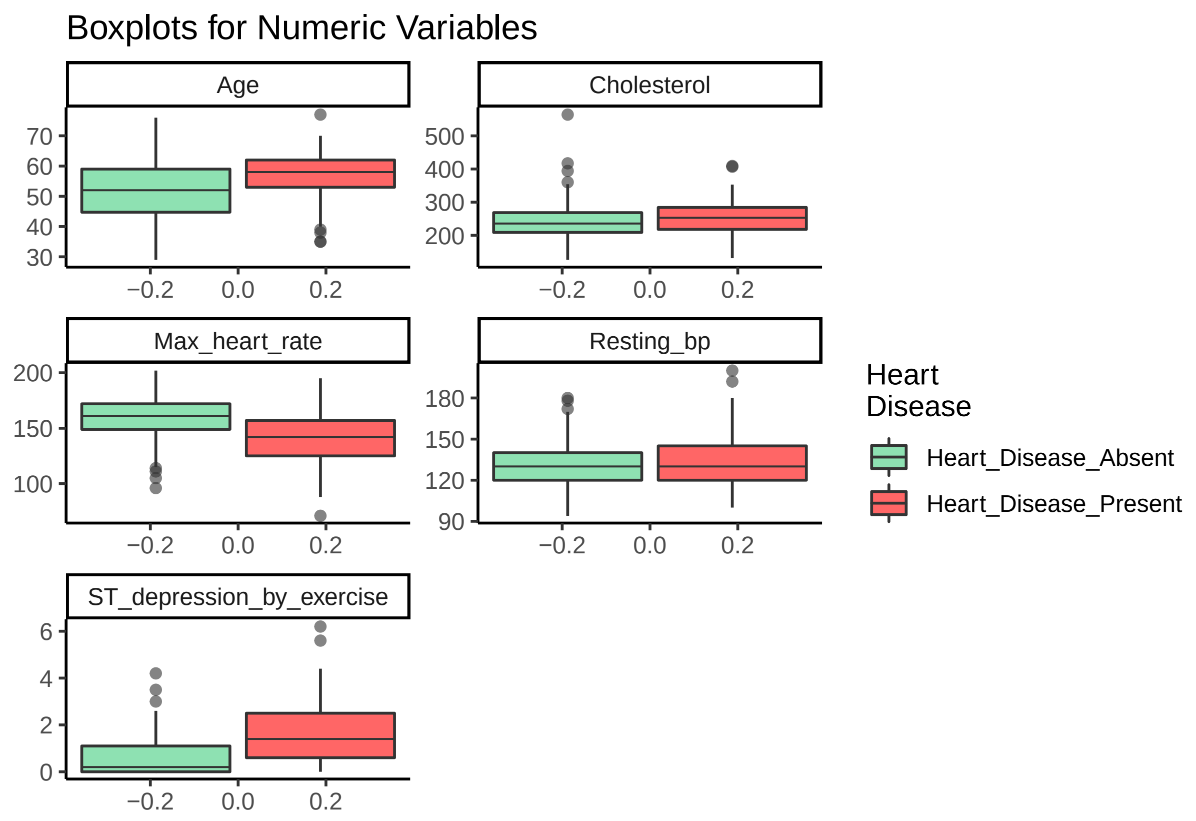
Now we see that outcome/Dependent variable has 5 levels. However, the UCI documentation suggests that any value above 0 in 'Diagnosis_Heart_Disease' indicates the presence of heart disease. As the primary motivation of our study here is to be able to predict the Heart_Disease_Status alone and not the grades within the heart disease, we can clump all levels above 0, so that we can have a binary variable woith levels 0 and 1, indicating - Heart disease absent or present.

```
analysis_data$Thalassemia <- as.factor(analysis_data$Thalassemia)
# Recoding Diagnosis_Heart_Disease column binary, 0 = no heart disease, 1 = heart disease present
# Also other qualitative variables' levels needs to be meaningful
analysis_data <- analysis_data %>%
  mutate(Diagnosis_Heart_Disease = ifelse(Diagnosis_Heart_Disease >0,1, Diagnosis_Heart_Disease)) %>%
  mutate_at("Diagnosis_Heart_Disease", as.factor) %>%
  mutate(Num_major_vessels_flouro = as.numeric(as.factor(Num_major_vessels_flouro))) %>%
  mutate(Sex = recode_factor(Sex, `0` = "female",
                                  `1` = "male" ),
         Chest_pain_type = recode_factor(Chest_pain_type, `1` = "typical",
                                                          `2` = "atypical",
                                                          `3` = "non_angina",
                                                          `4` = "asymptomatic"),
         Fasting_blood_sugar = recode_factor(Fasting_blood_sugar, `0` = "_Less_120mg_dl",
                                                                  `1` = "_More_120mg_dl"),
         Resting_ecg = recode_factor(Resting_ecg, `0` = "normal",
                                                  `1` = "ST_T_abnormality",
                                                  `2` = "LV_hypertrophy"),
         Exercise_induced_angina = recode_factor(Exercise_induced_angina, `0` = "no",
                                                                          `1` = "yes"),
         Slope_peak_exercise_ST  = recode_factor(Slope_peak_exercise_ST, `1` = "up_sloping",
                                                                           `2` = "flat",
                                                                           `3` = "down_sloping"),
         Thalassemia = recode_factor(Thalassemia, `3.0` = "normal",
                                                  `6.0` = "fixed_defect",
                                                  `7.0` = "reversible_defect"),
         Diagnosis_Heart_Disease = recode_factor(Diagnosis_Heart_Disease, `0` = "No_HD",
                                                                          `1` = "HD"))
```

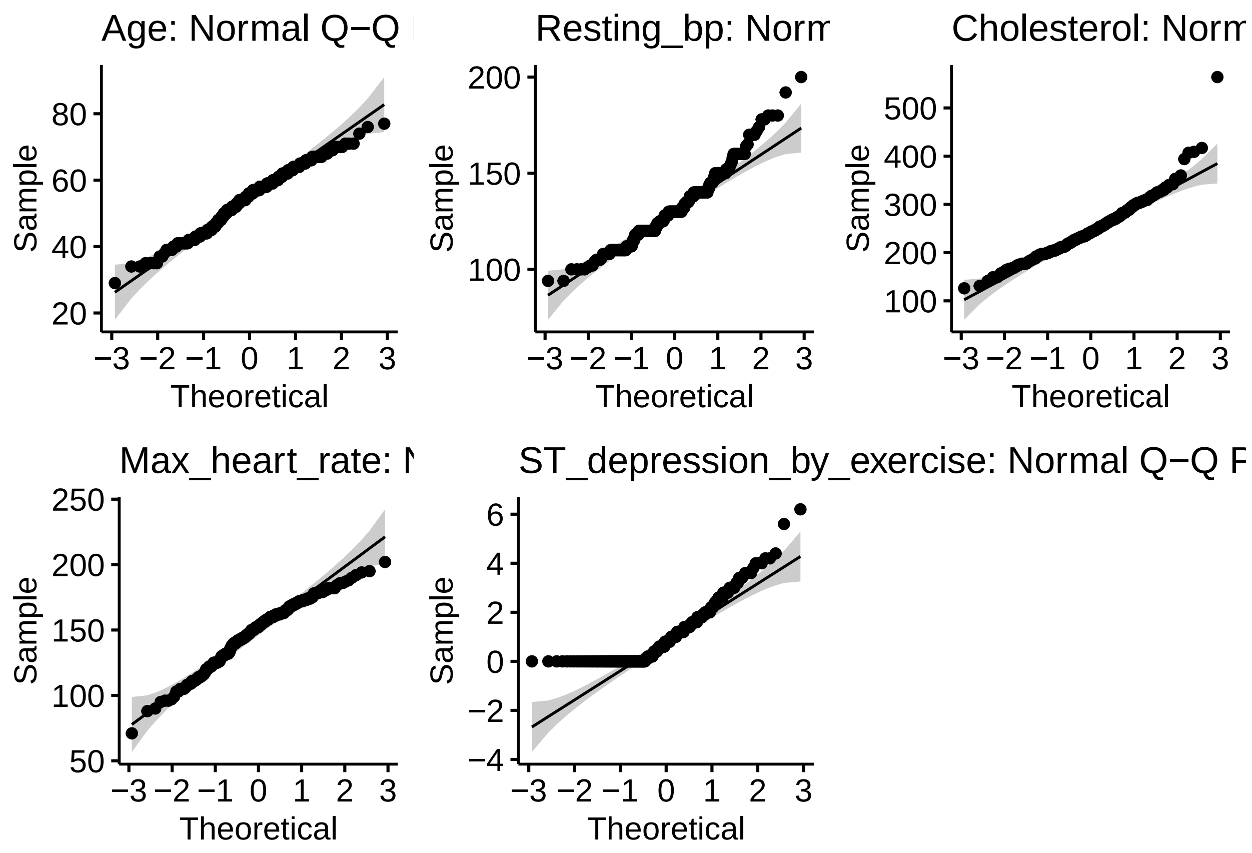
5 Exploratory Analysis

5.1 Distribution of quantitative variables

```
# Boxplots for quantitative variables, split by Disease status

# Gather the variables to plot by key
df1 <- analysis_data  %>% dplyr::select(Age,
                                 Resting_bp,
                                 Cholesterol,
                                 Max_heart_rate,
                                 ST_depression_by_exercise,
                                 Diagnosis_Heart_Disease) %>%
                          gather(key = "key",
                                 value = "value",
                                 -Diagnosis_Heart_Disease)

# Visualize numeric variables as boxplots
df1 %>% ggplot(aes(y = value)) +
               geom_boxplot(aes(fill = Diagnosis_Heart_Disease),
                            alpha  = .6,
                            fatten = .7) +
                ggtitle("Boxplots for Numeric Variables") +
                xlab("") +
                ylab("") +
                scale_fill_manual(values = c("seagreen3", "red"),
                                  name   = "Heart\nDisease",
                                  labels = c("Heart_Disease_Absent", "Heart_Disease_Present")) +
                theme(axis.text.x  = element_blank(), axis.ticks.x = element_blank()) +
                theme_classic() +
                facet_wrap(~ key, scales = "free", ncol   = 2)
```



```
# Checking for normality of continuous variables
q1 <- ggqqplot(analysis_data$Age, main="Age: Normal Q-Q Plot")
q2 <- ggqqplot(analysis_data$Resting_bp, main="Resting_bp: Normal Q-Q Plot")
q3 <- ggqqplot(analysis_data$Cholesterol, main="Cholesterol: Normal Q-Q Plot")
q4 <- ggqqplot(analysis_data$Max_heart_rate, main="Max_heart_rate: Normal Q-Q Plot")
q5 <- ggqqplot(analysis_data$ST_depression_by_exercise, main="ST_depression_by_exercise: Normal Q-Q Plo

grid.arrange(q1, q2, q3, q4, q5, ncol=3)
```



Since some variables like Max_heart_rate and cholesterol are numeric types, and the values range for quite a bit(cholesterol has a min of 126 and a max of 564), its best we normalise them before proceeding with any analysis.

```
# Normalizing data
scaling <- scale((analysis_data) %>%
                                dplyr::select(c("Resting_bp", "Cholesterol", "Max_heart_rate", "ST_depre
scaling <- data.frame(scaling)
analysis_data$Resting_bp <- scaling$Resting_bp
analysis_data$Cholesterol <- scaling$Cholesterol
analysis_data$Max_heart_rate <- scaling$Max_heart_rate
analysis_data$ST_depression_by_exercise <- scaling$ST_depression_by_exercise
analysis_data$Age <- scaling$Age
analysis_data$Num_major_vessels_flouro <- scaling$Num_major_vessels_flouro

glimpse(analysis_data)
```

```
## Rows: 297
## Columns: 14
## $ Age                       <dbl> 0.93460326, 1.37660512, 1.37660512, -1.93840~
## $ Sex                       <fct> male, male, male, male, female, male, female~
## $ Chest_pain_type           <fct> typical, asymptomatic, asymptomatic, non_ang~
## $ Resting_bp                <dbl> 0.74911571, 1.59357687, -0.65831955, -0.0953~
## $ Cholesterol               <dbl> -0.27597761, 0.74330055, -0.35290426, 0.0509~
## $ Fasting_blood_sugar       <fct> _More_120mg_dl, _Less_120mg_dl, _Less_120mg_~
## $ Resting_ecg               <fct> LV_hypertrophy, LV_hypertrophy, LV_hypertrop~
## $ Max_heart_rate            <dbl> 0.01746496, -1.81327350, -0.89790427, 1.6302~
## $ Exercise_induced_angina   <fct> no, yes, yes, no, no, no, no, yes, no, yes, ~
## $ ST_depression_by_exercise <dbl> 1.06716416, 0.38113006, 1.32442695, 2.096215~
## $ Slope_peak_exercise_ST    <fct> down_sloping, flat, flat, down_sloping, up_s~
## $ Num_major_vessels_flouro  <dbl> -0.7207596, 2.4742493, 1.4092463, -0.7207596~
## $ Thalassemia               <fct> fixed_defect, normal, reversible_defect, nor~
## $ Diagnosis_Heart_Disease   <fct> No_HD, HD, HD, No_HD, No_HD, No_HD, HD, No_H~
```

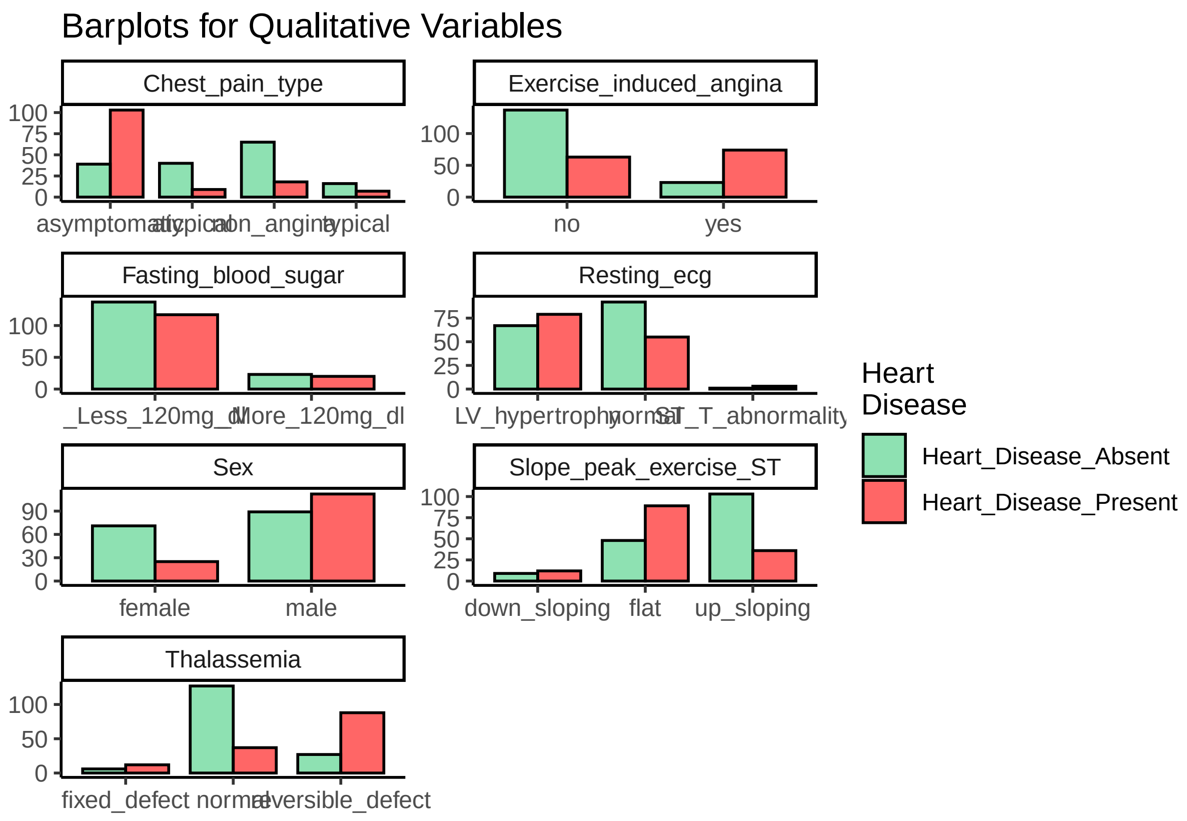
The boxplots suggests that these conditions are associated in various degrees (some very slightly, some in a major way as seen by the median lines not aligning between the Disease status) with an increased prevalence of heart disease: Higher age Lower max heart rate achieved Higher cholesterol Higher ST depression induced by exercise relative to rest

5.2 Lets see how the categorical variables are distributed wrt disease status

```r
# Barplots for qualitative variables, split by Disease status

# Gather the variables to plot by key
df2 <- analysis_data  %>% dplyr::select(Sex,
                                 Chest_pain_type,
                                 Fasting_blood_sugar,
                                 Resting_ecg,
                                 Exercise_induced_angina,
                                 Slope_peak_exercise_ST,
                                 Thalassemia,
                                 Diagnosis_Heart_Disease) %>%
                          gather(key = "key",
                                 value = "value",
                                 -Diagnosis_Heart_Disease)
```

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

```r
#Visualize numeric variables as boxplots
df2 %>% ggplot(aes(value)) +
               geom_bar(aes(x = value,
                            fill     = Diagnosis_Heart_Disease),
                            alpha    = .6,
                            position = "dodge",
                            color    = "black",
                            width    = .8) +
                ggtitle("Barplots for Qualitative Variables") +
                xlab("") +
                ylab("") +
                scale_fill_manual(values = c("seagreen3", "red"),
                                  name   = "Heart\nDisease",
                                  labels = c("Heart_Disease_Absent", "Heart_Disease_Present")) +
                theme(axis.text.y  = element_blank(), axis.ticks.y = element_blank()) +
                theme_classic() +
                facet_wrap(~ key, scales = "free", nrow   = 4)
```

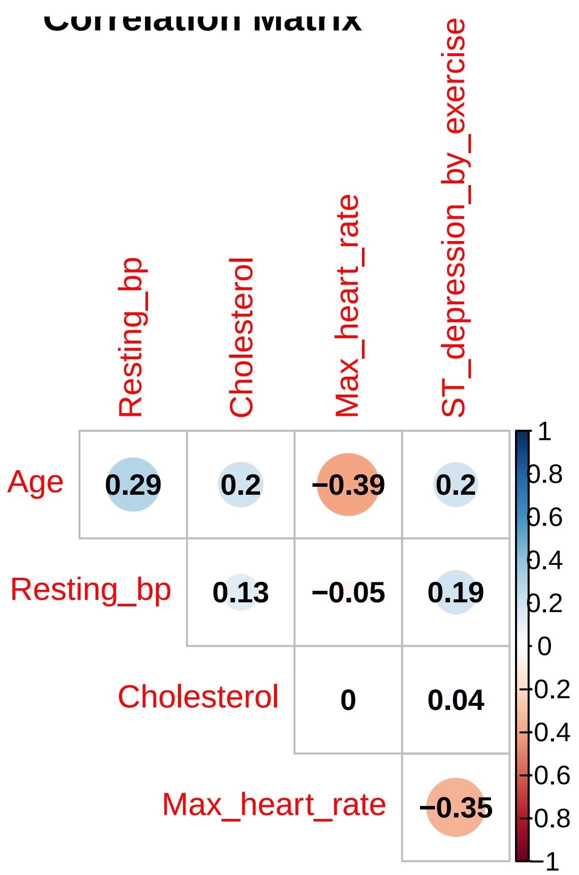


The barplots suggests that these conditions are associated in various degrees (judging by the spread of the distributuion of categorical variables across disease status) with an increased prevalence of heart disease: Higher grade angina chest pain (grade 4) Presence of exercise induced angina Male Higher thalassemia score Higher Resting ECG score (grade 2) We see that >50% males have heart disease whereas, ~25% females, have the heart disease. Could this mean that the data is sex-biased? Also, in general, sex, age, cholesteril, angina status are known variables to be associated with Heart Disease and wrt data we have here we see similar trend.

Correlated variables in any model causes incorrect/inflated inferences and hence such multicollinearity should be checked before implmenting any analysis methods.

5.3 We check the correlations between all the numerical attributes

```
cols <- c('Sex', 'Chest_pain_type', 'Fasting_blood_sugar', 'Resting_ecg', 'Exercise_induced_angina', 'Sl
corrplot::corrplot(cor(analysis_data[, -match(cols, colnames(analysis_data))]), # subset the data for o
         method="circle",
         type = "upper",
         number.cex = .9,
         insig = 'blank',
         diag=FALSE,
         addCoef.col ='black',
         title = "Correlation Matrix")
```



There are no highly correlated variables so we can include all the 14 variables and now can proceed to modelling the dataset. Now some modelling. As usage of linear regression is inappropriate in this case since the response variable is binary in nature (Disease type), we go for a suite of regression anmd classification methods (both supervised and unsupervised to select the best features which can help predict the disease status. We would use: LOgistic regression, Regression Trees,

5.4 Prepare for Model Building

```
# Splitting the data into training and test set using caret package
set.seed(123)
training_samples <- analysis_data$Diagnosis_Heart_Disease %>%
                        createDataPartition(p = 0.7, list = FALSE)
train_data  <- analysis_data[training_samples, ]
test_data <- analysis_data[-training_samples, ]

# Now using the model.matrix() to create matrix of predictors which then automatically converts categor
# Dummy code categorical predictor variables
x <- model.matrix(Diagnosis_Heart_Disease ~ ., data = train_data[, -c(2:3)])[,-1]

paste("No. of rows in the Training data: ", nrow(train_data))
```

```
## [1] "No. of rows in the Training data:  208"
```

```
paste("No. of rows in the Test data: ", nrow(test_data))
```

```
## [1] "No. of rows in the Test data:  89"
```

```
paste("Test to Train data ratio is: ", nrow(test_data)/(nrow(train_data) + nrow(test_data)))
```

```
## [1] "Test to Train data ratio is:  0.2996632996633"
```

6 Models

6.1 Logistic Regression (GLM)

```
set.seed(223)
# Fit the logistic regression model with glm function: generalized linear model using binomial for logi
glm_fit <- train(Diagnosis_Heart_Disease ~ .,
                 data = train_data,
                 method = "glm",
                 family = "binomial")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
# Test the model using the predict function
glm_pred <- predict(glm_fit, newdata = test_data)

# Confusion matrix
glm_conf_matrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease,
                                         glm_pred))

# Extract parameters from the confusion matrix
accuracy_glm <- glm_conf_matrix$overall["Accuracy"]
sensitivity_glm <- glm_conf_matrix$byClass["Sensitivity"]
specificity_glm <- glm_conf_matrix$byClass["Specificity"]
f1_glm <- 2*sensitivity_glm*specificity_glm/(sensitivity_glm+specificity_glm)

# AUC
glm_AUC <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(glm_pred))["auc"]
```

```
## Setting levels: control = No_HD, case = HD
```

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

```
glm_AUC <- as.numeric(substr(glm_AUC, -7, 5))

# GLM output for viewing later
model_accs <- tibble(Method = "Generalized Linear Model",
                     Accuracy = accuracy_glm,
                     "Sensitivity/Recall" = sensitivity_glm,
                     "F1 score" = f1_glm,
                     AUC = glm_AUC)
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858

Interestingly, we get a warning message when running glm() above : "prediction from a rank-deficient fit may be misleading". Now this could be due to several reasons, either there are multicollinear variables present OR several variables exist with NA's in them OR adding more variables than the number of samples etc. Could we do something about it by specifying a Bayesian version of GLM, invoking a flat but not completely noninformative prior?

6.2 Bayesian GLM

```
set.seed(123)
# Fit the logistic regression model with bayesglm function: generalized linear model using binomial for
bayesglm_fit <- train(Diagnosis_Heart_Disease ~ .,
                 data = train_data,
                 method = "bayesglm",
                 family = "binomial")

# Test the model using the predict function
bayesglm_pred <- predict(bayesglm_fit, newdata = test_data)

# Confusion matrix
bayesglm_conf_matrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease,
                                         bayesglm_pred))

# Extract parameters from the confusion matrix
accuracy_bayesglm <- bayesglm_conf_matrix$overall["Accuracy"]
sensitivity_bayesglm <- bayesglm_conf_matrix$byClass["Sensitivity"]
specificity_bayesglm <- bayesglm_conf_matrix$byClass["Specificity"]
f1_bayesglm <- 2*sensitivity_bayesglm*specificity_bayesglm/(sensitivity_bayesglm+specificity_bayesglm)

# AUC
bayesglm_AUC <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(bayesglm_pred))["auc"]
```

```
## Setting levels: control = No_HD, case = HD
```

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

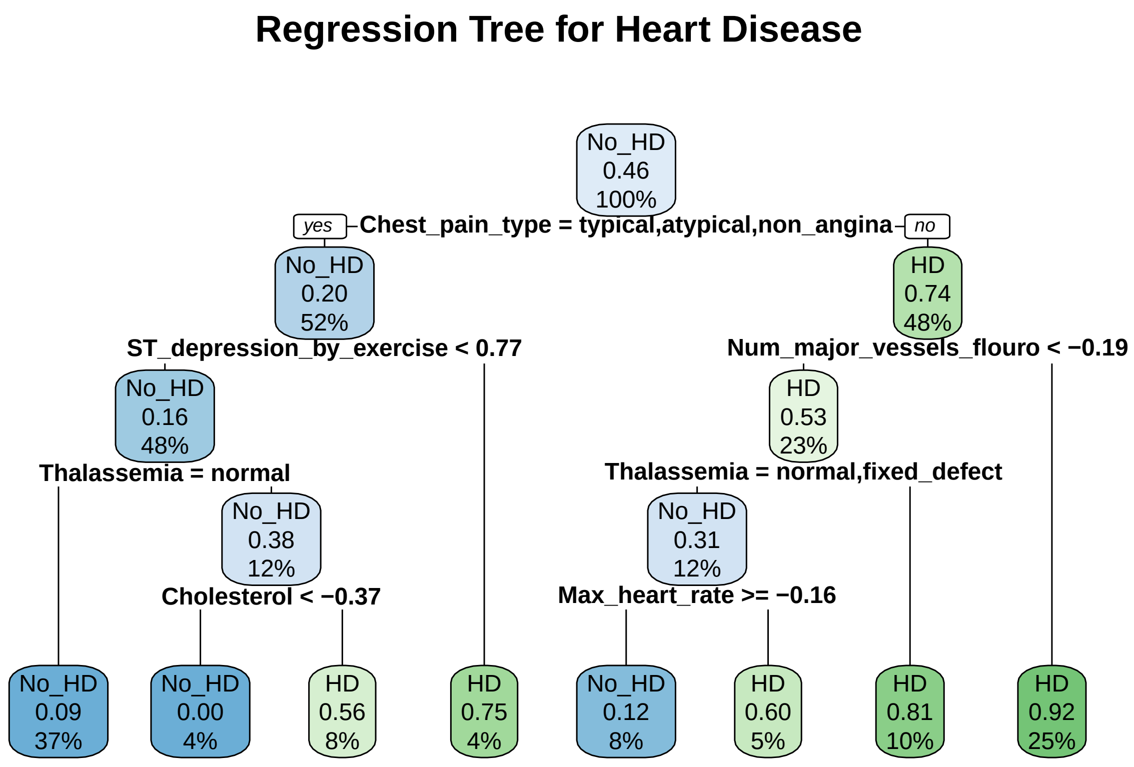
```
bayesglm_AUC <- as.numeric(substr(bayesglm_AUC, -7, 5))

# bayesglm output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Bayesian GLM",
                               Accuracy = accuracy_bayesglm,
                               "Sensitivity/Recall" = sensitivity_bayesglm,
                               "F1 score" = f1_bayesglm,
                               AUC = bayesglm_AUC))
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881

We see that with this the warning of rank deficient matrix is avoided.

6.3 Regression Tree

```
set.seed(123)
# Fit the regression tree model with rpart function
rt_fit <- rpart(Diagnosis_Heart_Disease ~ .,
                 data = train_data,
                 method = "class")

# Plot the regression tree
rpart.plot(rt_fit,
           cex=0.75,
           main = "Regression Tree for Heart Disease")
```



```r
# Test the model using the predict function
rt_pred <- predict(rt_fit, test_data, type = "class")

# Confusion matrix
rt_cMatrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease, rt_pred))

# Extract parameters from the confusion matrix
accuracy_rt <- rt_cMatrix$overall["Accuracy"]
sensitivity_rt <- rt_cMatrix$byClass["Sensitivity"]
specificity_rt <- rt_cMatrix$byClass["Specificity"]
f1_rt <- 2*sensitivity_rt*specificity_rt/(sensitivity_rt+specificity_rt)

# AUC
rt_AUC <- roc(test_data$Diagnosis_Heart_Disease,as.numeric(rt_pred))["auc"]
```

```
## Setting levels: control = No_HD, case = HD

## Setting direction: controls < cases
```

```r
rt_AUC <- as.numeric(substr(rt_AUC, -7, 5))

# RT output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Regression Tree Model",
                               Accuracy = accuracy_rt,
                               "Sensitivity/Recall" = sensitivity_rt,
                               "F1 score" = f1_rt,
                               AUC = rt_AUC)
)
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793

The regression tree is interesting since Chest_pain_type is shown to be the most important classifier for the patient samples.

6.4 Random Forest (RF)

```
set.seed(123)
# Fit the Random Forest model with rpart function
rf_fit <- randomForest(Diagnosis_Heart_Disease ~ .,
                 data = train_data,
                 importance=TRUE,
                 ntree=500)

# Test the model using the predict function
rf_pred <- predict(rf_fit, test_data)

# Confusion matrix
rf_cMatrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease, rf_pred))

# Extract parameters from the confusion matrix
accuracy_rf <- rf_cMatrix$overall["Accuracy"]
sensitivity_rf <- rf_cMatrix$byClass["Sensitivity"]
specificity_rf <- rf_cMatrix$byClass["Specificity"]
f1_rf <- 2*sensitivity_rf*specificity_rf/(sensitivity_rf + specificity_rf)

# AUC
rf_AUC <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(rf_pred))["auc"]
```

```
## Setting levels: control = No_HD, case = HD
```

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

```
rf_AUC <- as.numeric(substr(rf_AUC, -7, 5))

# RF output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Random Forest Model",
                               Accuracy = accuracy_rf,
```

```r
                               "Sensitivity/Recall" = sensitivity_rf,
                               "F1 score" = f1_rf,
                               AUC = rf_AUC)
)
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858

6.5 SVM

```r
set.seed(123)
# Fit the SVM model with the svm function
svm_fit <- svm(Diagnosis_Heart_Disease ~ .,
                data = train_data)

# Test the model using the predict function
svm_pred <- predict(svm_fit, test_data)

# Confusion matrix
svm_cMatrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease, svm_pred))

# Extract parameters from the confusion matrix
accuracy_svm <- svm_cMatrix$overall["Accuracy"]
sensitivity_svm <- svm_cMatrix$byClass["Sensitivity"]
specificity_svm <- svm_cMatrix$byClass["Specificity"]
f1_svm <- 2*sensitivity_svm*specificity_svm/(sensitivity_svm+specificity_svm)

# AUC
svm_AUC <- roc(test_data$Diagnosis_Heart_Disease,
               as.numeric(svm_pred))["auc"]
```

```
## Setting levels: control = No_HD, case = HD

## Setting direction: controls < cases
```

```r
svm_AUC <- as.numeric(substr(svm_AUC, -7, 5))

# SVM output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Support-Vector Machine",
                               Accuracy = accuracy_svm,
                               "Sensitivity/Recall" = sensitivity_svm,
                               "F1 score" = f1_svm,
                               AUC = svm_AUC)
)
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871

6.6 Neural Networks

```
set.seed(123)
# Since neuralnet only deals with quantitative variables, we convert all the qualitative variables (fac
model_train <- as.data.frame(model.matrix( ~ Age + Sex + Chest_pain_type + Resting_bp + Cholesterol + Fa
                     Exercise_induced_angina + ST_depression_by_exercise + Slope_peak_exercise_ST + Num_
                    data = train_data))

names(model_train) <- make.unique(names(model_train))
# Fit the neural networks model with the neuralnet function
nn_fit <- neuralnet::neuralnet(Diagnosis_Heart_DiseaseHD ~ Age + Sexmale + Chest_pain_typeatypical +
                                                            Chest_pain_typenon_angina + Chest_pain_type
                                                            Cholesterol + Fasting_blood_sugar_More_120mg
                                                            Resting_ecgLV_hypertrophy + Max_heart_rate +
                                                            ST_depression_by_exercise + Slope_peak_exerc
                                                            Num_major_vessels_flouro + Thalassemiafixed_
                    data = model_train ,
                    hidden = 10,
                    act.fct = "logistic",
                    threshold = 0.05,
                    linear.output = FALSE,
                    lifesign = "minimal") # thresh varying from 0.01, to 0.05
```

```
## hidden: 10    thresh: 0.05    rep: 1/1    steps:     106 error: 1.51896  time: 0.03 secs
```

```
# Plot the fit
plot(nn_fit)

# Test the model using the predict function
# Since neuralnet only deals with quantitative variables, we convert all the qualitative variables (fac
model_test <- model.matrix( ~ Age + Sex + Chest_pain_type + Resting_bp + Cholesterol + Fasting_blood_sug
                     Exercise_induced_angina + ST_depression_by_exercise + Slope_peak_exercise_ST + Num_
                    data = test_data)

nn_pred <- neuralnet::compute(nn_fit, model_test)
nn_prob <- nn_pred$net.result
nn_pred <- as.data.frame(ifelse(nn_prob>0.6, 1, 0))
nn_pred$V1 <- as.factor(nn_pred$V1)
nn_pred <- nn_pred %>% mutate(status = recode(V1, `0` = "No_HD", `1` = "HD"))

# Confusion matrix
nn_cMatrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease, nn_pred$status))
nn_cMatrix
```

```
## Confusion Matrix and Statistics
##
##
##          No_HD HD
##   No_HD    44  4
##   HD       11 30
##
##                  Accuracy : 0.8315
##                    95% CI : (0.7373, 0.9025)
##     No Information Rate : 0.618
##     P-Value [Acc > NIR] : 1.018e-05
##
##                     Kappa : 0.6565
##
##  Mcnemar's Test P-Value : 0.1213
##
##               Sensitivity : 0.8000
##               Specificity : 0.8824
##            Pos Pred Value : 0.9167
##            Neg Pred Value : 0.7317
##                Prevalence : 0.6180
##            Detection Rate : 0.4944
##      Detection Prevalence : 0.5393
##         Balanced Accuracy : 0.8412
##
##         'Positive' Class : No_HD
##
```

```
# Extract parameters from the confusion matrix
accuracy_nn <- nn_cMatrix$overall["Accuracy"]
sensitivity_nn <- nn_cMatrix$byClass["Sensitivity"]
specificity_nn <- nn_cMatrix$byClass["Specificity"]
f1_nn <- 2*sensitivity_nn*specificity_nn/(sensitivity_nn+specificity_nn)

# AUC
nn_AUC <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(nn_pred[,1]))["auc"]
```

```
## Setting levels: control = No_HD, case = HD
## Setting direction: controls < cases
```

```
nn_AUC <- as.numeric(substr(nn_AUC, -7, 5))

# NN output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Neural Network",
                               Accuracy = accuracy_nn,
                               "Sensitivity/Recall" = sensitivity_nn,
                               "F1 score" = f1_nn,
                               AUC = nn_AUC)
)
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871
Neural Network	0.8314607	0.8000000	0.8391608	0.824

```r
# Variable Importance
varImp_nn <- olden(nn_fit, "Diagnosis_Heart_DiseaseHD", bar_plot=FALSE)
varImp_nn
```

```
##                                     importance
## Age                                   4.260377
## Sexmale                              25.864933
## Chest_pain_typeatypical              28.769352
## Chest_pain_typenon_angina            16.325718
## Chest_pain_typeasymptomatic          71.829945
## Resting_bp                           33.658582
## Cholesterol                           4.732586
## Fasting_blood_sugar_More_120mg_dl   -33.336381
## Resting_ecgST_T_abnormality         116.973733
## Resting_ecgLV_hypertrophy            -8.461010
## Max_heart_rate                      -15.765118
## Exercise_induced_anginayes            6.054052
## ST_depression_by_exercise            33.904828
## Slope_peak_exercise_STflat           56.673285
## Slope_peak_exercise_STdown_sloping   31.028895
## Num_major_vessels_flouro             62.718292
## Thalassemiafixed_defect             118.259520
## Thalassemiareversible_defect         55.879110
```

6.7 K-Nearest Neighbour (KNN)

```r
# Control parameter to pass to the fit function
tr_control <- trainControl(method="repeatedcv",
                           number=10,
                           repeats=3,
                           classProbs = T,
                           summaryFunction = twoClassSummary)

# Fit the kNN model with the train function and knn method
knn_fit <- train(Diagnosis_Heart_Disease ~ .,
                 data = train_data,
                 method = "knn",
                 metric = 'ROC',
                 tuneLength = 20,
                 tuneGrid = expand.grid(k=1:70),
                 trControl = tr_control,
                 preProc=c("center", "scale"))
```

```
# Test the model using the predict function
knn_pred <- predict(knn_fit, newdata=test_data)
#knn_pred <- ifelse(knn_pred > 0.4, 1, 0)

# Confusion matrix
knn_cMatrix <- confusionMatrix(table(test_data$Diagnosis_Heart_Disease, knn_pred))

# Extract parameters from the confusion matrix
accuracy_knn <- knn_cMatrix$overall["Accuracy"]
sensitivity_knn <- knn_cMatrix$byClass["Sensitivity"]
specificity_knn <- knn_cMatrix$byClass["Specificity"]
f1_knn <- 2*sensitivity_knn*specificity_knn/(sensitivity_knn+specificity_knn)

# AUC
knn_AUC <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(knn_pred))["auc"]
```

```
## Setting levels: control = No_HD, case = HD
```

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

```
knn_AUC <- as.numeric(substr(knn_AUC, -7, 5))

# KNN output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "k-Nearest Neighbors",
                               Accuracy = accuracy_knn,
                               "Sensitivity/Recall" = sensitivity_knn,
                               "F1 score" = f1_knn,
                               AUC = knn_AUC)
)
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871
Neural Network	0.8314607	0.8000000	0.8391608	0.824
k-Nearest Neighbors	0.8764045	0.8490566	0.8815672	0.871

6.8 LASSO

```
set.seed(123)
# Lasso train
lasso_train <- train_data

# LASSO model
```

```r
lambda <- 10^seq(-3, 3, length = 100)

# Matrix of Predictor Variables
pred <- data.matrix(lasso_train[, -ncol(train_data)])

# Convert the disease status (class) to a numerical variable
outcome <- ifelse(lasso_train$Diagnosis_Heart_Disease == "HD", 1, 0)

# Find the best lambda using 10 fold cross-validation. alpha = 1 is for lasso
# aka Find the optimal value of lambda that minimizes the CV error
cv.lasso <- cv.glmnet(pred, outcome, alpha = 1, family = "binomial", nfolds=10)

# Fit the model with lambda.min
lasso_fit <- glmnet(pred,
                   outcome,
                   alpha = 1,
                   family = "binomial",
                   lambda = cv.lasso$lambda.min)

# Test the model using the predict function
lasso_test <- test_data
x.test <- data.matrix(lasso_test[,-ncol(lasso_test)])
probabilities <- lasso_fit %>% predict(newx = x.test)
lasso_pred <- as.data.frame(ifelse(probabilities > 0.5, "1", "0"))
lasso_pred[,1] <- as.factor(lasso_pred[,1])
lasso_pred <- lasso_pred %>% mutate(status = recode(s0, `0` = "No_HD", `1` = "HD"))

# Confusion matrix
lasso_conf_matrix <- confusionMatrix(table(lasso_test$Diagnosis_Heart_Disease,
                                          lasso_pred$status))

# Extract parameters from the confusion matrix
accuracy_lasso <- lasso_conf_matrix$overall["Accuracy"]
sensitivity_lasso <- lasso_conf_matrix$byClass["Sensitivity"]
specificity_lasso <- lasso_conf_matrix$byClass["Specificity"]
f1_lasso <- 2*sensitivity_lasso*specificity_lasso/(sensitivity_lasso + specificity_lasso)

# AUC
lasso_test <- lasso_test %>% mutate(Diagnosis_Heart_Disease = recode(Diagnosis_Heart_Disease, "No_HD" =
lasso_AUC <- roc(lasso_test$Diagnosis_Heart_Disease, as.numeric(lasso_pred$s0))["auc"]
```

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

## Setting direction: controls < cases
```

```r
lasso_AUC <- as.numeric(substr(lasso_AUC, -7, 5))

# LASSO output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "LASSO",
                               Accuracy = accuracy_lasso,
                               "Sensitivity/Recall" = sensitivity_lasso,
                               "F1 score" = f1_lasso,
                              AUC = lasso_AUC))
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871
Neural Network	0.8314607	0.8000000	0.8391608	0.824
k-Nearest Neighbors	0.8764045	0.8490566	0.8815672	0.871
LASSO	0.8314607	0.7704918	0.8565643	0.818

6.9 Ridge

```
set.seed(123)
# ridge train
ridge_train <- train_data

# ridge model
lambda <- 10^seq(-3, 3, length = 100)

# Matrix of Predictor Variables
pred <- data.matrix(ridge_train[, -ncol(train_data)])

# Convert the disease status (class) to a numerical variable
outcome <- ifelse(ridge_train$Diagnosis_Heart_Disease == "HD", 1, 0)

# Find the best lambda using 10 fold cross-validation. alpha = 1 is for ridge
# aka Find the optimal value of lambda that minimizes the CV error
cv.ridge <- cv.glmnet(pred, outcome, alpha = 0, family = "binomial", nfolds=10)

# Fit the model with lambda.min
ridge_fit <- glmnet(pred,
                   outcome,
                   alpha = 1,
                   family = "binomial",
                   lambda = cv.ridge$lambda.min)

# Test the model using the predict function
ridge_test <- test_data
x.test <- data.matrix(ridge_test[,-ncol(ridge_test)])
probabilities <- ridge_fit %>% predict(newx = x.test)
ridge_pred <- as.data.frame(ifelse(probabilities > 0.5, "1", "0"))
ridge_pred[,1] <- as.factor(ridge_pred[,1])
ridge_pred <- ridge_pred %>% mutate(status = recode(s0, `0` = "No_HD", `1` = "HD"))

# Confusion matrix
ridge_conf_matrix <- confusionMatrix(table(ridge_test$Diagnosis_Heart_Disease,
                                          ridge_pred$status))
```

```
# Extract parameters from the confusion matrix
accuracy_ridge <- ridge_conf_matrix$overall["Accuracy"]
sensitivity_ridge <- ridge_conf_matrix$byClass["Sensitivity"]
specificity_ridge <- ridge_conf_matrix$byClass["Specificity"]
f1_ridge <- 2*sensitivity_ridge*specificity_ridge/(sensitivity_ridge + specificity_ridge)

# AUC
ridge_test <- ridge_test %>% mutate(Diagnosis_Heart_Disease = recode(Diagnosis_Heart_Disease, "No_HD" =
ridge_AUC <- roc(ridge_test$Diagnosis_Heart_Disease, as.numeric(ridge_pred$s0))["auc"]
```

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

## Setting direction: controls < cases
```

```
ridge_AUC <- as.numeric(substr(ridge_AUC, -7, 5))

# ridge output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Ridge",
                               Accuracy = accuracy_ridge,
                               "Sensitivity/Recall" = sensitivity_ridge,
                               "F1 score" = f1_ridge,
                               AUC = ridge_AUC))
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871
Neural Network	0.8314607	0.8000000	0.8391608	0.824
k-Nearest Neighbors	0.8764045	0.8490566	0.8815672	0.871
LASSO	0.8314607	0.7704918	0.8565643	0.818
Ridge	0.8314607	0.7619048	0.8648649	0.817

6.10 Elastic-Net

```
set.seed(123)
# Elastic net train
elastic_train <- train_data

# elastic model
lambda <- 10^seq(-3, 3, length = 100)

# Matrix of Predictor Variables
pred <- data.matrix(elastic_train[, -ncol(train_data)])
```

```r
# Convert the disease status (class) to a numerical variable
outcome <- ifelse(elastic_train$Diagnosis_Heart_Disease == "HD", 1, 0)

# Find the best lambda using 10 fold cross-validation. alpha = 1 is for elastic
# aka Find the optimal value of lambda that minimizes the CV error
cv.elastic <- cv.glmnet(pred, outcome, alpha = 0, family = "binomial", nfolds=10)

# Fit the model with lambda.min
elastic_fit <- glmnet(pred,
                   outcome,
                   alpha = 1,
                   family = "binomial",
                   lambda = cv.elastic$lambda.min)

# Test the model using the predict function
elastic_test <- test_data
x.test <- data.matrix(elastic_test[,-ncol(elastic_test)])
probabilities <- elastic_fit %>% predict(newx = x.test)
elastic_pred <- as.data.frame(ifelse(probabilities > 0.5, "1", "0"))
elastic_pred[,1] <- as.factor(elastic_pred[,1])
elastic_pred <- elastic_pred %>% mutate(status = recode(s0, `0` = "No_HD", `1` = "HD"))

# Confusion matrix
elastic_conf_matrix <- confusionMatrix(table(elastic_test$Diagnosis_Heart_Disease,
                                         elastic_pred$status))

# Extract parameters from the confusion matrix
accuracy_elastic <- elastic_conf_matrix$overall["Accuracy"]
sensitivity_elastic <- elastic_conf_matrix$byClass["Sensitivity"]
specificity_elastic <- elastic_conf_matrix$byClass["Specificity"]
f1_elastic <- 2*sensitivity_elastic*specificity_elastic/(sensitivity_elastic + specificity_elastic)

# AUC
elastic_test <- elastic_test %>% mutate(Diagnosis_Heart_Disease = recode(Diagnosis_Heart_Disease, "No_HI
elastic_AUC <- roc(elastic_test$Diagnosis_Heart_Disease, as.numeric(elastic_pred$s0))["auc"]
```

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

## Setting direction: controls < cases
```

```r
elastic_AUC <- as.numeric(substr(elastic_AUC, -7, 5))

# Elastic net output for viewing later
model_accs <- bind_rows(model_accs,
                        tibble(Method = "Elastic Net",
                               Accuracy = accuracy_elastic,
                               "Sensitivity/Recall" = sensitivity_elastic,
                               "F1 score" = f1_elastic,
                               AUC = elastic_AUC))
model_accs %>% knitr::kable()
```

Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871
Neural Network	0.8314607	0.8000000	0.8391608	0.824
k-Nearest Neighbors	0.8764045	0.8490566	0.8815672	0.871
LASSO	0.8314607	0.7704918	0.8565643	0.818
Ridge	0.8314607	0.7619048	0.8648649	0.817
Elastic Net	0.8314607	0.7619048	0.8648649	0.817

6.11 Model Performance Tradeoffs

6.11.1 Plot AUC-ROCs for all the models

```
# create object to combine plots
par(mfrow=c(3,3))
par(pty = "s")
par(oma = c(3, 3, 3, 3), mar=c(1, 1, 1, 1), mgp=c(2, 0.8, 0), las=0)

# plot AUC-ROC for the Logistic Regression model
glm_obj <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(glm_pred), ci=TRUE, ci.alpha=0.9,
               plot=TRUE, legacy.axes = TRUE, percent=TRUE,
               #xlab="False Positive Percentage", ylab="True Positive Percentage",
               xlab="FPP", ylab="TPP",
               print.auc.x=80, print.auc.y=60,
               print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = No_HD, case = HD

## Setting direction: controls < cases
```

```
sens.ci_glm <- ci.se(glm_obj)
plot(sens.ci_glm, type = "shape", col="lightblue",
     title(main="GLM - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_glm, type = "shape", col = "lightblue", title(main
## = "GLM - ROC", : Low definition shape.
```

```
     #title(main="Generalized Linear Model - ROC", line=3))


# plot AUC-ROC for Regression Tree model
rt_obj <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(rt_pred), ci=TRUE, ci.alpha=0.9,
              plot=TRUE, legacy.axes = TRUE, percent=TRUE,
              #xlab="False Positive Percentage", ylab="True Positive Percentage",
              xlab="FPP", ylab="TPP",
              print.auc.x=75, print.auc.y=60,
              print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = No_HD, case = HD
## Setting direction: controls < cases
```

```
sens.ci_rt <- ci.se(rt_obj)
plot(sens.ci_rt, type = "shape", col="lightblue",
     title(main="Regression Tree - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_rt, type = "shape", col = "lightblue", title(main
## = "Regression Tree - ROC", : Low definition shape.
```

```
# plot AUC-ROC for the Random Forest model
rf_obj <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(rf_pred), ci=TRUE, ci.alpha=0.9,
              plot=TRUE, legacy.axes = TRUE, percent=TRUE,
              #xlab="False Positive Percentage", ylab="True Positive Percentage",
              xlab="FPP", ylab="TPP",
              print.auc.x=75, print.auc.y=60,
              print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = No_HD, case = HD
## Setting direction: controls < cases
```

```
sens.ci_rf <- ci.se(rf_obj)
plot(sens.ci_rf, type = "shape", col="lightblue",
     title(main="Random Forest - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_rf, type = "shape", col = "lightblue", title(main
## = "Random Forest - ROC", : Low definition shape.
```

```
# plot AUC-ROC for the SVM model
svm_obj <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(svm_pred), ci=TRUE, ci.alpha=0.9,
               plot=TRUE, legacy.axes = TRUE, percent=TRUE,
               #xlab="False Positive Percentage", ylab="True Positive Percentage",
               xlab="FPP", ylab="TPP",
               print.auc.x=75, print.auc.y=60,
               print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = No_HD, case = HD
## Setting direction: controls < cases
```

```
sens.ci_svm <- ci.se(svm_obj)
plot(sens.ci_svm, type = "shape", col="lightblue",
     title(main="SVM - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_svm, type = "shape", col = "lightblue", title(main
## = "SVM - ROC", : Low definition shape.
```

```
# plot AUC-ROC for the Neural Networks model
nn_obj <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(nn_pred$V1),ci=TRUE, ci.alpha=0.9,
              plot=TRUE, legacy.axes = TRUE, percent=TRUE,
              #xlab="False Positive Percentage", ylab="True Positive Percentage",
              xlab="FPP", ylab="TPP",
              print.auc.x=75, print.auc.y=60,
              print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = No_HD, case = HD
## Setting direction: controls < cases
```

```
sens.ci_nn <- ci.se(nn_obj)
plot(sens.ci_nn, type = "shape", col="lightblue",
     title(main="Neural Network - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_nn, type = "shape", col = "lightblue", title(main
## = "Neural Network - ROC", : Low definition shape.
```

```
# plot AUC-ROC for k-Nearest Neighbors model
knn_obj <- roc(test_data$Diagnosis_Heart_Disease, as.numeric(knn_pred), ci=TRUE, ci.alpha=0.9,
               plot=TRUE, legacy.axes = TRUE, percent=TRUE,
               #xlab="False Positive Percentage", ylab="True Positive Percentage",
               xlab="FPP", ylab="TPP",
               print.auc.x=75, print.auc.y=60,
               print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = No_HD, case = HD
## Setting direction: controls < cases
```

```
sens.ci_knn <- ci.se(knn_obj)
plot(sens.ci_knn, type = "shape", col="lightblue",
     title(main="KNN - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_knn, type = "shape", col = "lightblue", title(main
## = "KNN - ROC", : Low definition shape.
```

```
# plot AUC-ROC for LASSO model
lasso_obj <- roc(lasso_test$Diagnosis_Heart_Disease, as.numeric(lasso_pred$s0), ci=TRUE, ci.alpha=0.9,
               plot=TRUE, legacy.axes = TRUE, percent=TRUE,
               #xlab="False Positive Percentage", ylab="True Positive Percentage",
               xlab="FPP", ylab="TPP",
               print.auc.x=75, print.auc.y=60,
               print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
sens.ci_lasso <- ci.se(lasso_obj)
plot(sens.ci_lasso, type = "shape", col="lightblue",
     title(main="LASSO - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_lasso, type = "shape", col = "lightblue", : Low
## definition shape.
```

```
# plot AUC-ROC for Ridge Regression model
ridge_obj <- roc(ridge_test$Diagnosis_Heart_Disease, as.numeric(ridge_pred$s0), ci=TRUE, ci.alpha=0.9,
               plot=TRUE, legacy.axes = TRUE, percent=TRUE,
               #xlab="False Positive Percentage", ylab="True Positive Percentage",
               xlab="FPP", ylab="TPP",
               print.auc.x=75, print.auc.y=60,
               print.auc=TRUE, show.thres=TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
sens.ci_ridge <- ci.se(ridge_obj)
plot(sens.ci_ridge, type = "shape", col="lightblue",
     title(main="Ridge Regression - ROC", line=.25))
```

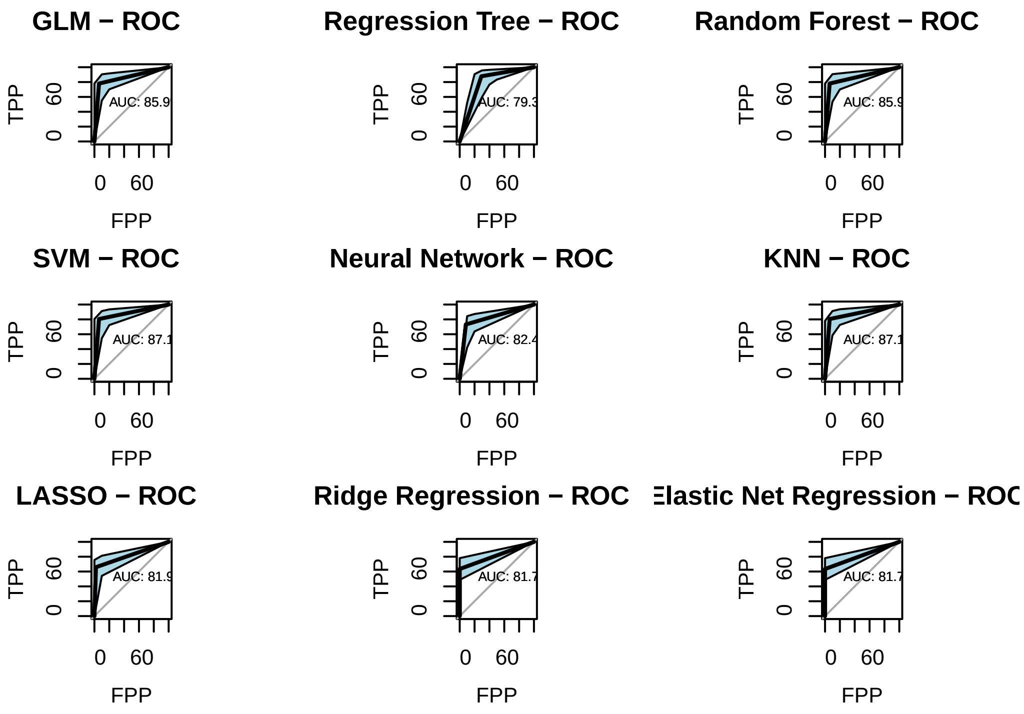
```
## Warning in plot.ci.se(sens.ci_ridge, type = "shape", col = "lightblue", : Low
## definition shape.
```

```
# plot AUC-ROC for Elastic Net Regression model
elastic_obj <- roc(elastic_test$Diagnosis_Heart_Disease, as.numeric(elastic_pred$s0), ci=TRUE, ci.alpha
               plot=TRUE, legacy.axes = TRUE, percent=TRUE,
               #xlab="False Positive Percentage", ylab="True Positive Percentage",
               xlab="FPP", ylab="TPP",
               print.auc.x=75, print.auc.y=60,
               print.auc=TRUE, show.thres=TRUE)
```

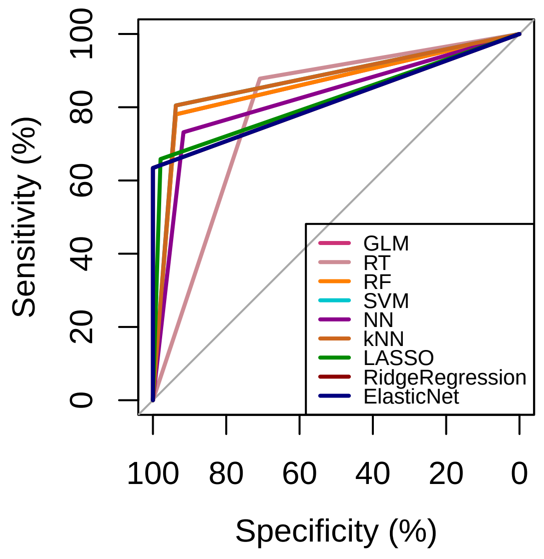
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
sens.ci_elastic <- ci.se(elastic_obj)
plot(sens.ci_elastic, type = "shape", col="lightblue",
     title(main="Elastic Net Regression - ROC", line=.25))
```

```
## Warning in plot.ci.se(sens.ci_elastic, type = "shape", col = "lightblue", : Low
## definition shape.
```



```
par(mfrow=c(1,1))
# Make a composite plot of ROC-AUCs
plot(glm_obj, col="violetred3")
plot(rt_obj, col="lightpink3", add=TRUE)
plot(rf_obj, col="darkorange1", add=TRUE)
plot(svm_obj, col="turquoise3", add=TRUE)
plot(nn_obj, col="darkmagenta", add=TRUE)
plot(knn_obj, col="chocolate3", add=TRUE)
plot(lasso_obj, col="green4", add=TRUE)
plot(ridge_obj, col="red4", add=TRUE)
plot(elastic_obj, col="navyblue", add=TRUE)
legend("bottomright", c("GLM","RT","RF","SVM","NN","kNN", "LASSO", "RidgeRegression", "ElasticNet"), lt
col=c("violetred3","lightpink3","darkorange1","turquoise3","darkmagenta","chocolate3", "green4", "red4"
```

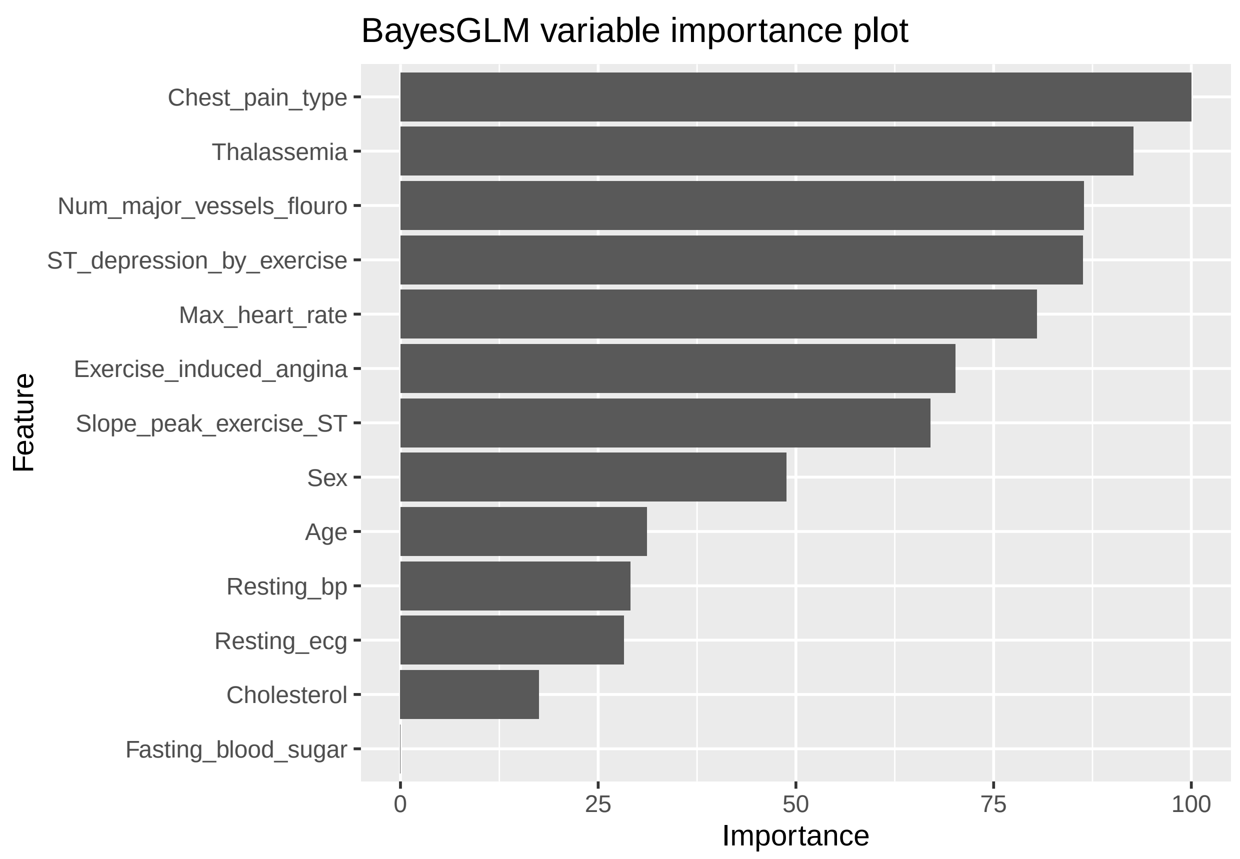


```r
#text(locator(), labels = c("GLM","RT","RF","SVM","NN","kNN", "LASSO", "RidgeRegression", "ElasticNet"),

# Summary of model performance metrics
model_accs %>% knitr::kable()
```

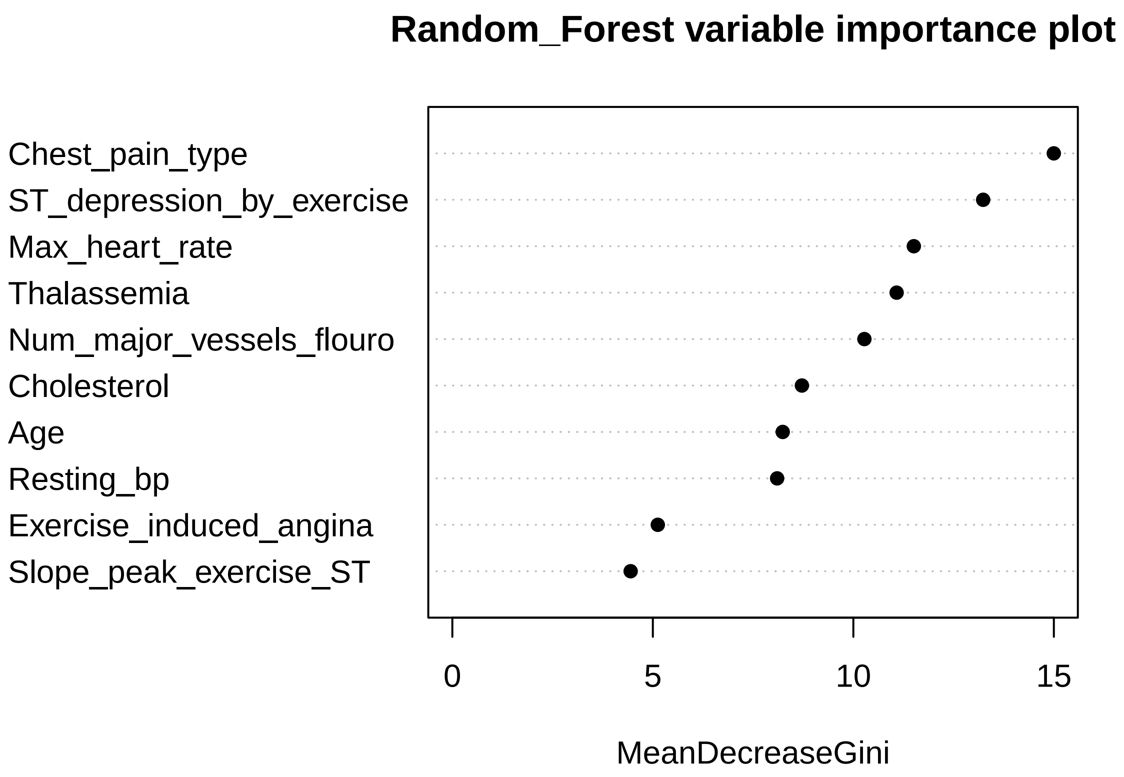
Method	Accuracy	Sensitivity/Recall	F1 score	AUC
Generalized Linear Model	0.8651685	0.8333333	0.8719346	0.858
Bayesian GLM	0.8876404	0.8518519	0.8950472	0.881
Regression Tree Model	0.7865169	0.8717949	0.7886598	0.793
Random Forest Model	0.8651685	0.8333333	0.8719346	0.858
Support-Vector Machine	0.8764045	0.8490566	0.8815672	0.871
Neural Network	0.8314607	0.8000000	0.8391608	0.824
k-Nearest Neighbors	0.8764045	0.8490566	0.8815672	0.871
LASSO	0.8314607	0.7704918	0.8565643	0.818
Ridge	0.8314607	0.7619048	0.8648649	0.817
Elastic Net	0.8314607	0.7619048	0.8648649	0.817

6.12 Variables of Importance in the Best Model - BayesGLM & RF

```r
# Important variables - Bayesian GLM
ggplot2::ggplot(varImp(bayesglm_fit)) +
                ggtitle("BayesGLM variable importance plot")
```



```r
# Important variables - RF
varImpPlot(rf_fit, type=2, main="Random_Forest variable importance plot",
           pch = 16, col.lab="black",
           n.var=min(10, nrow(rf_fit$importance)))
```



```
#knnImp <- varImp(knn_fit)
#dotPlot(knnImp, main = "kNN variable importance plot" )
#dotPlot(varImp(bayesglm_fit), main = "BayesGLM variable importance plot" )
```

We see that Bayes GLm had the highest AUC metric and also in other metrics like Sensitivity, F1 score. With an 88%correct classification rate, Bayes GLM could be a powerful tool for medical studies. The variables which were considered to be "important" are: Chest__pain__type, Thalassemia, Number of Major Vessels (0-3) Visible on Flouroscopy, ST Depression Induced by Exercise Relative to Rest, Max Heart Rate Achieved etc.