Heart Disease Detection: A Predictive Model Evaluation

Claire Bentzen, JV Deniega, and Ravita Kartawinata

Shiley-Marcos School of Engineering, University of San Diego

Team 4 - ADS 503 Applied Predictive Modeling (Summer 2024)

Table of Contents

Problem Statement and Justification

Heart disease is the leading cause of death in both men and women worldwide (*The Top 10 Causes of Death*, 2020). With such an existential problem for humanity, the use of rapidly developing modeling and machine learning techniques may hold the key in providing insights in prevention as well as potentially earlier detection of likely heart disease cases in patients. In an attempt to identify these critical factors that may indicate the presence of heart disease, this project aims to explore the feasibility of using such predictive modeling and machine learning techniques to potentially be deployed as a tool that may augment human-in-the-loop diagnoses.

Methodology

Using binary classification predictive modeling with various R-based libraries, the team's objective is to detect and identify heart disease early in individuals based on physical and medical characteristics. To evaluate the candidate models, though accuracy is generally seen as the colloquial "gold standard" metric, the risk that a baseline model that simply makes a single prediction for all cases toward the majority class (no heart disease) carries dangerous implications borne from a false negative diagnosis where a model predicts that a patient is healthy, but actually suffers from heart disease.

Medical Diagnosis Focus: Sensitivity-Specificity Harmonic Mean

Sensitivity and specificity are viewed of equal importance when the priority is in maximizing detection rates while minimizing Type II error in false negative diagnoses. Detecting the maximum number of real cases ensures that a model will better predict when patients who have heart disease should be considered for further treatment while also mitigating the likelihood that a model would recommend a clean bill of health to a patient with heart disease. Both outcomes are equally dangerous and so both cases must be considered when selecting a model.

Therefore, the maximum harmonic mean between these two metrics will be preferred in this case.

Resource Conscious: Receiver Operating Characteristic – Area Under Curve (ROC-AUC)

However, if there is a higher-priority to consider that a given region or medical treatment facility may experience scarcity in resources (either in medication, medical professionals, or other), then false positives would need to be a part of model evaluation. The implication of an error in this scenario would result in utilizing limited resources on a patient who did not actually need the treatment, which ultimately could have been used on an actual patient with heart disease. Therefore, ROC-AUC would be the preferred model evaluation metric in this case.

Exploratory Data Analysis

The heart disease dataset was obtained through Kaggle and is an accumulation of data from four different databases including Cleveland, Hungary, Switzerland, and Long Beach (Lapp, 2019). It consists of 14 variables, 13 predictors and 1 target variable, and 1026 records. The variables are listed as follows:

1. age (in years)
2. sex (0 = female; 1 = male)
3. cp (chest pain type 0-3)
4. trestbps (resting blood pressure in mm Hg on admission to the hospital)
5. chol (serum cholesterol in mg/dl)
6. fbs (fasting blood sugar 120 mg/dl 1 = true; 0 = false)
7. restecp (resting electrocardiographic results 0-2)
8. thalach (maximum heart rate achieved)
9. exang (exercise induced angina 1 = yes; 0 = no)
10. oldpeak (ST depression induced by exercise relative to rest)
11. slope (the slope of the peak exercise ST segment)
12. ca (number of major vessels (0-3) colored by fluoroscopy)

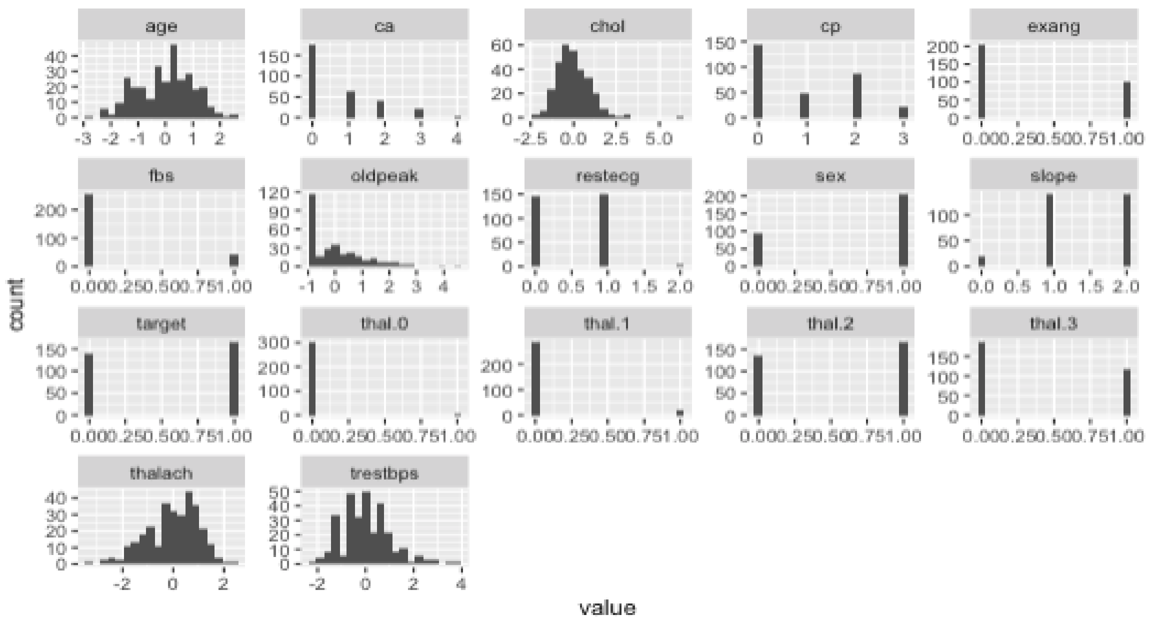
13. thal (1 = normal; 2 = fixed defect; 3 = reversible defect)
14. target (0 = no disease; 1 = disease)

**Note: There are reasons to believe that the target outcome values may have been swapped during the data creation process on Kaggle. Due to the scope of this project, this is not something that was explored, but a related thesis suggests that this error may have taken place (Simmons II, 2021).*

On initial exploration of the data, there are no missing values but there are 723 duplicate rows. Using the ggplot method from the ggplot2 library, we plot the distribution of each variable to check for any notable or unusual distributions. Depicted in Figure 1, chol and trestbps are slightly right skewed whereas age is slightly left skewed. The variable, oldpeak, is very right skewed. Additionally, it is worth mentioning that some of the categorical variables such as sex, ca, cp, fbs, and exang are unbalanced.

Figure 1

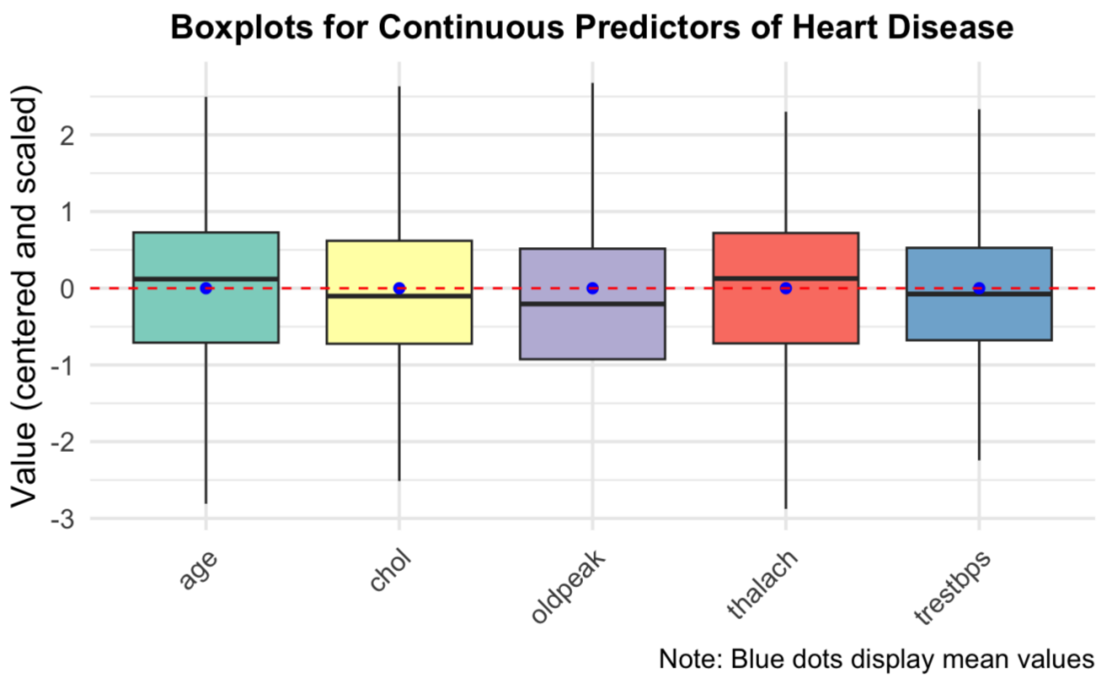
Variable Distributions (Histogram)



The boxplots for the continuous variables depicted in Figure 2 are used to analyze the distributions on a deeper level by visualizing the interquartile range.

Figure 2

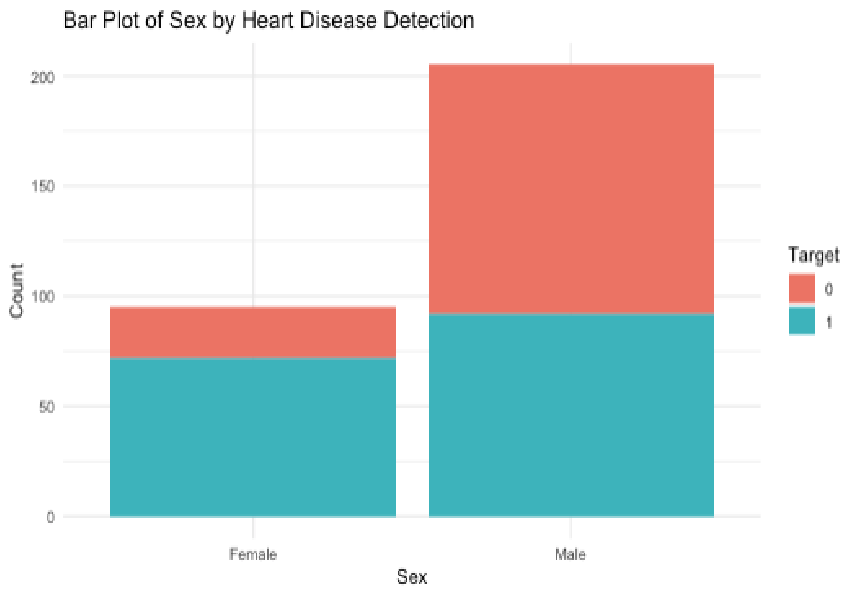
Boxplots for Continuous Predictors



Intervariable relationships to the target variable such as via sex, thal, and age were initial candidates to explore. As previously mentioned, the data is unbalanced between males and females at a roughly two-to-one ratio as depicted in Figure 3.

Figure 3

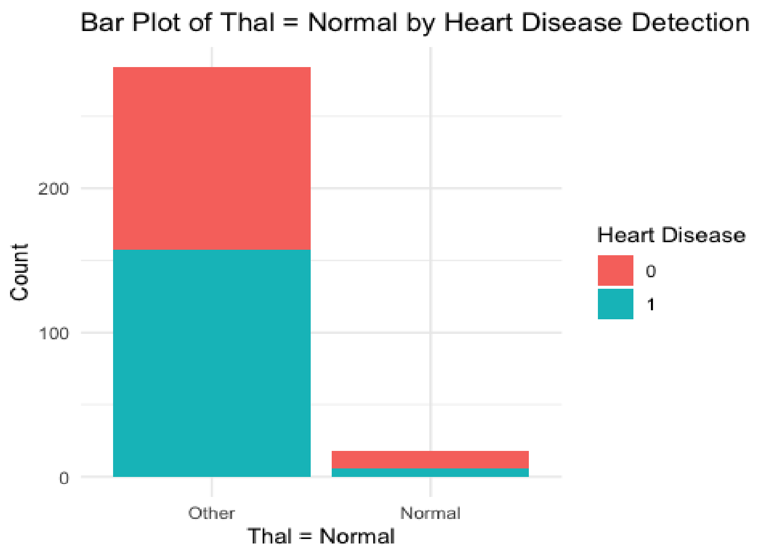
Stacked Bar Graph of Heart Disease by Sex



Furthermore, the following figures show the relationship between normal, fixed defect, and reversible defect with heart disease detection. Figure 4, 5 and 6 suggest that heart disease is detected more frequently in fixed defect cases compared to reversible defect cases.

Figure 4

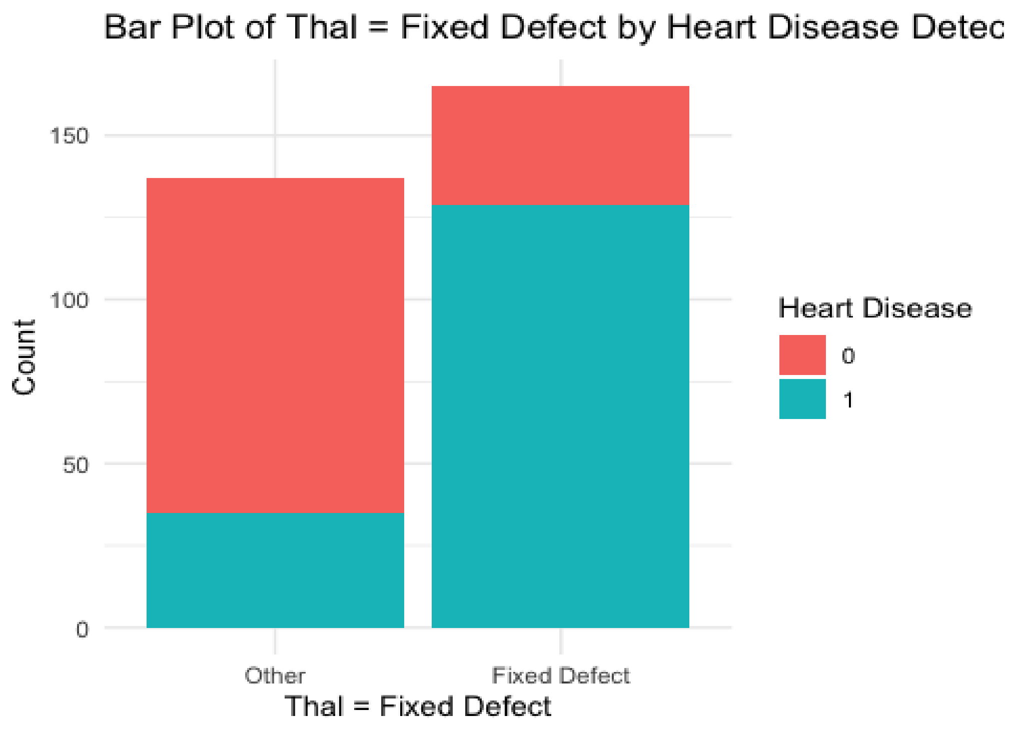
Stacked Bar Graph of thal = "normal" vs. Heart Disease



A notable observation then thal is given a normal factor value is that there do exist cases where thal is considered to be normal, but still were labeled to have heart disease. Conversely, there are cases where the factor value “other” (or abnormal), were nearly equally distributed to have positive or negative heart disease labels. From this, the presence of a defect in a patient does not initially seem to be a strong indicator of heart disease.

Figure 5

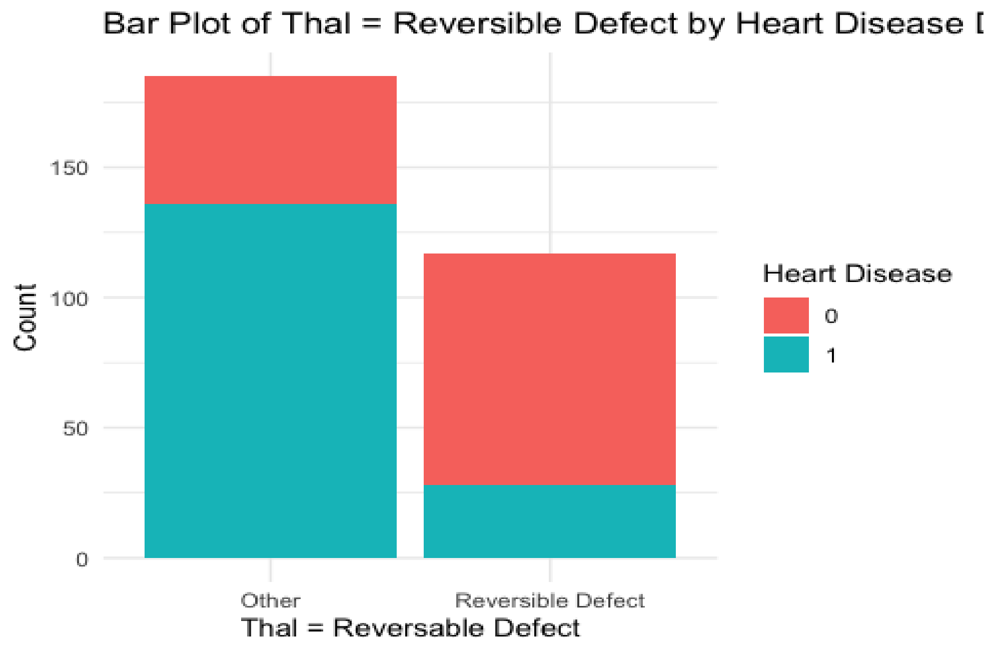
Stacked Bar Graph of thal = “fixed” vs. Heart Disease



With respect to thal factor values with fixed defects, we may see a relatively stronger relationship to the presense of heart disease. However, it must be considered that there may exist confounding factors where fixed defects imply that the patient suffers a proclivity or a lifestyle that may correlate to heart disease rather than a fixed defect specifically causing a heart disease label.

Figure 6

Stacked Bar Graph of thal = "fixed" vs. Heart Disease

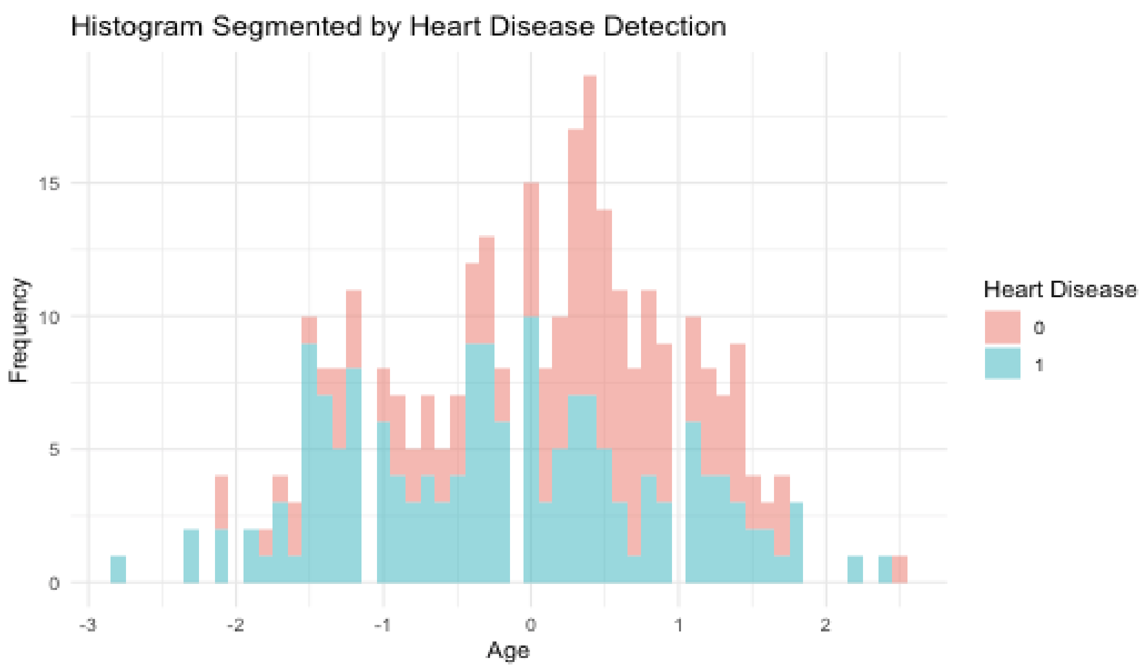


Reversible defects appear to have an opposite relationship than that of fixed defects. The greater proportion of no heart disease label for this factor value may be an indicator of a negative diagnosis.

Lastly, the following segmented histogram in Figure 7 illustrates the distribution of heart disease detection by age (centered and scaled). Heart disease appears to display relatively large kurtosis for positive diagnoses, which suggests that age may potentially be a weak predictor. The distribution of the negative cases also suggest that higher ages are more likely to be tested for heart disease, suggesting that younger patients may be missing not at random. Therefore, this should be noted as the data may be biased toward those who do perform tests, or more senior patients.

Figure 7

Histogram of Age, Segmented by Heart Disease



Data Wrangling and Pre-processing

Missing and Duplicate Data

For the 723 duplicate rows. The team removed the duplicate rows as this data contains continuous variables operating in a medical context. The odds that any number of patients would have the exact same medically-related readings like blood pressure, blood sugar, and cholesterol versus the records being duplicated upon data ingest or other technical errors on transcription were considered to be significantly low, which resulted in 302 rows remaining. This is not an issue as the number of rows in still more than the number of columns squared. Using a general practice of $\# \, columns \leq \sqrt[2]{\# \, rows}$ to determine we have a sufficient quantity of data, 302 rows is sufficient for our purposes where we use 17 or fewer predictors.

Outlier Handling

Continuous outliers were handled by detecting any values that were 1.5 times greater than the minimum Q1 or the maximum Q3 quartile for each continuous predictor and reassigning the values to the newer clipped values. This allows the data point to remain in the dataset while minimizing the extreme effect the outlier value may have in calculating the mean and standard deviation values used for scaling.

Continuous Data Preprocessing

Once outliers were handled, the continuous variables were preprocessed by being centered at zero to share a common mean and scaled by the calculated standard deviation of the dataset by respective variable. This is especially crucial for certain algorithms susceptible to outliers such as k-Nearest Neighbors (kNN)

Categorical Data Preprocessing

The variable thal contains the values 1, 2, or 3 and is categorical due to the lack of ordinal importance. To account for this data type, dummy variables are created to better facilitate classification modeling and to identify individual feature importances between these categorical variables. The created dummy variables - thal.1, thal.2, and thal.3 – still map to normal, fixed defect, and reversible defect, respectively.

Near-Zero Variance

Using the nearZeroVar() method at this stage of the dataset, the thal.0 dummy variable is identified as a near-zero variance column and is removed. It is important to note that the original documentation had the thal values shifted down by one value to where normal is supposed to be assigned a zero value and so on. However, inspecting the actual dataset, the inclusive range of the thal variable only ranged from one to three and is considered a potential one-shift error in transcription.

High Correlation

Separate analysis of correlation was split by continuous and categorical variables. Continuous variables were compared with each other using the Pearson method while categorical variables were compared using the Spearman method due to their respective data types. No variables were found to have coefficients greater than 0.75, our selected threshold as depicted in Figures 8 and 9 except for thal.2 and thal.3. This relationship is surprising given that there appears to be a strong negative relationship between reversible and fixed heart defects, but not with those considered to be normal.

Figure 8

Pearson Correlation Coefficient Matrix

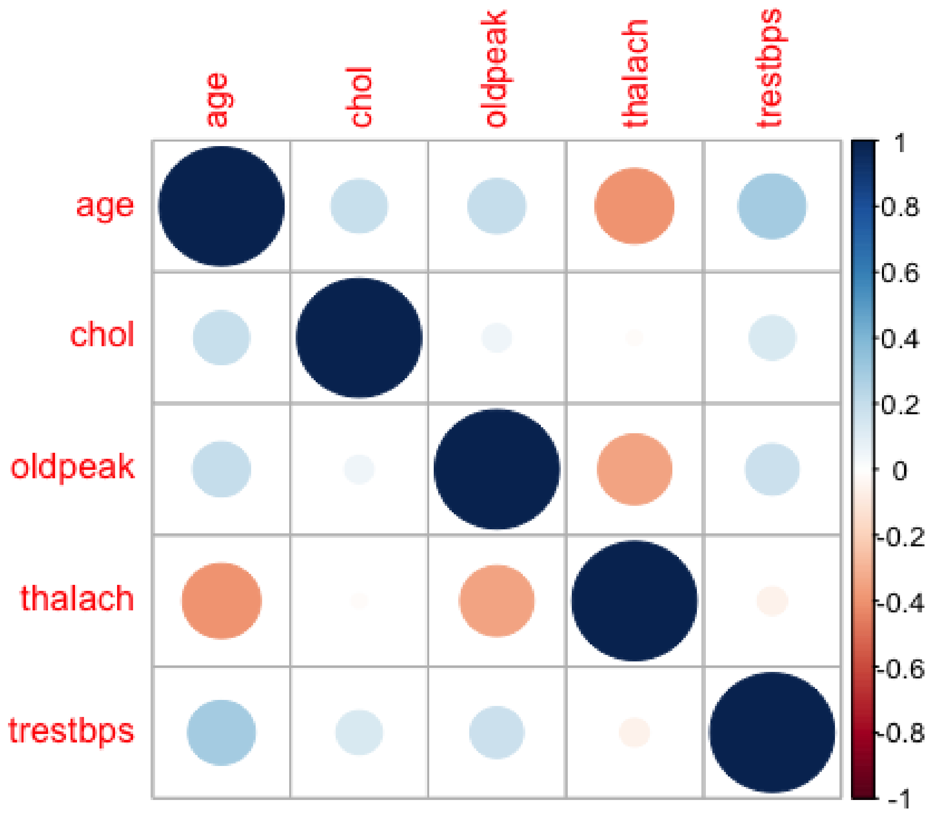
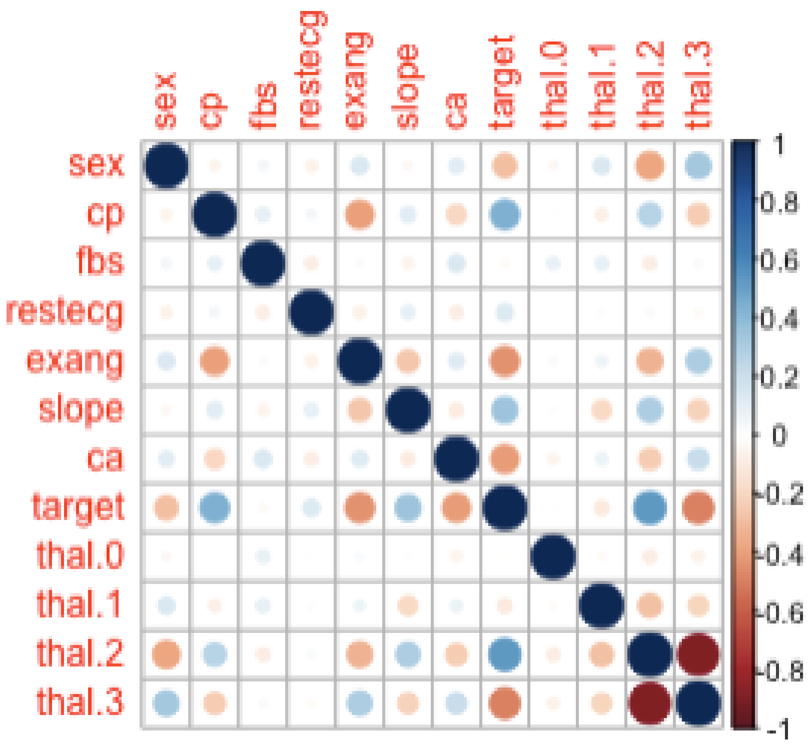


Figure 9

Spearman Correlation Coefficient Matrix



Confounding Variables

To check for the presence of confounding variables, a simple logistic regression model to the variables that were suspected of being confounding: age, sex, thal.2, and thal.3. This model suggests that age, sex, and thal.2 are possible confounding variables due to their low p-values, however it does not confirm that suspicion. The next step looks at the correlation between these three predictors and the target variable. Since none of the coefficients are particularly strong, none of these variables were identified as confounding variables or removed.

Data Splitting

After data pre-processing and clean-up, data was split by using the caret library to generate a stratified random split to ensure that the proportions of different categories in the target variable are preserved in both training and test sets. The data was divided into 80 percent for the training set and 20 percent for the test set, which resulted in 241 training samples and 59 test sample, respectively. The training dataset was also transformed into numeric type in R to allow specific implementations of classification model algorithm formats. The selected method of sampling included repeated cross-validation at 5 repeats or iterations to help reduce potential variance error on the test set.

Model Strategies

As previous mentioned, the harmonic mean of sensitivity and specificity is one of our two primary evaluation metrics, which we wrote as a custom function. In the preparation of hyperparameter tunings, we set up the grid of hyperparameter combinations by initializing alpha and lambda hyperparameters, though we do acknowledge that the limited range may potentially only find a local minimum on convergence. Expand.grid generates a data frame which will have 70 rows from 7 values of alpha at values 0, .1, .2, .4, .6, .8, and 1. Lambda was evaluated from 10 equidistant values between a range of .01 to .2.

Table 1

List of Algorithms used for Training (Kuhn & Johnson, 2013)

Algorithm	Definition	Caret method
Neural Network	Utilizes layers of interconnected nodes to learn complex patterns in data	nnetGrid
Support Vector Machine	find the optimal hyperplane that separates data points of different classes	svmRadial
K-Nearest Neighbors	Classify data points based on the majority class among its nearest neighbors.	knn
Logistic Regression	Use logic function to estimate the probability outcome	glm
Linear Discriminant Analysis (LDA)	Use Gaussian distribution with equal covariance matrices for normal distributed predictor	lda

Penalized Logistic Regression	Extension of logistic regression that adds penalty term such as Lasso Ridge	glmnet
Nearest Shrunken Centroids	Use for high-dimensional classification to shrink class centroid and reduce noise	pam

Validation and Testing

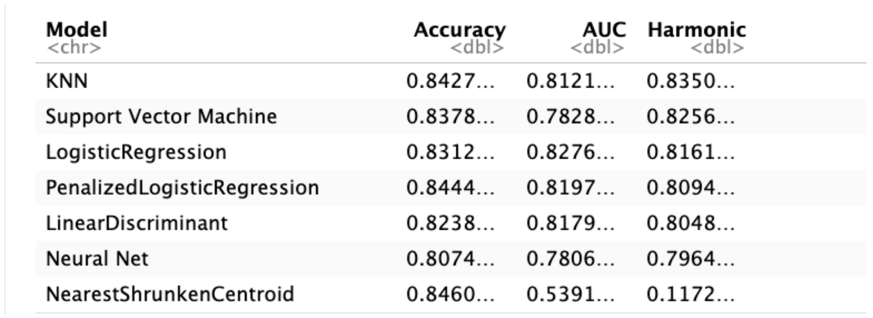
Harmonic mean of sensitivity and specificity, accuracy, and ROC-AUC values were the evaluation metrics used to identify the optimally performing model to predict heart disease. The harmonic mean is used as a metric to optimize for maximum diagnostic detection by minimizing the most dangerous outcome from a medical standpoint either by Type I or Type II error. Since sensitivity measures true positives and specificity measures true negatives, the harmonic mean optimizes both of these metrics, thus reducing the likelihood of any false positives or false negatives. This metric is ideal where heart disease in the universe of cases is minimized by the model.

AUC is another metric used for this scenario to detect the possibility of consequences as a result of a false positive. A patient mistakenly diagnosed with heart disease subsequently has to deal with emotional and financial impacts. Additionally, false positives contribute to resource allocation considerations, meaning false positives can potentially subtract treatment options from true positive cases. These factors lead us to favor optimizing for AUC if resource allocation and scarcity mitigation is paramount.

After each model was trained on the training set of 241 observations, the model performance was recorded as follows with the KNN model leading the performance for the harmonic metric.

Figure 9

Model Performance Evaluation Metrics



Model <chr>	Accuracy <dbl>	AUC <dbl>	Harmonic <dbl>
KNN	0.8427...	0.8121...	0.8350...
Support Vector Machine	0.8378...	0.7828...	0.8256...
LogisticRegression	0.8312...	0.8276...	0.8161...
PenalizedLogisticRegression	0.8444...	0.8197...	0.8094...
LinearDiscriminant	0.8238...	0.8179...	0.8048...
Neural Net	0.8074...	0.7806...	0.7964...
NearestShrunkenCentroid	0.8460...	0.5391...	0.1172...

Following evaluation of model performance, each model was tested on the test set of 59 observations. The results are shown visually in Figures 10 and 11.

Figure 10

Harmonic Mean Test Results

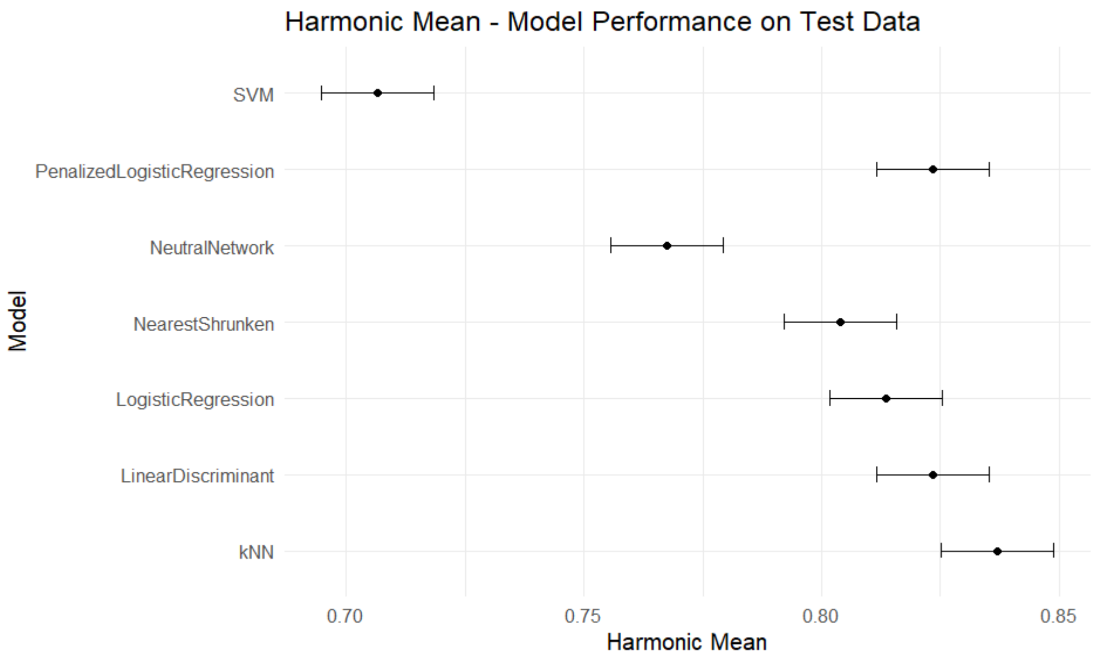
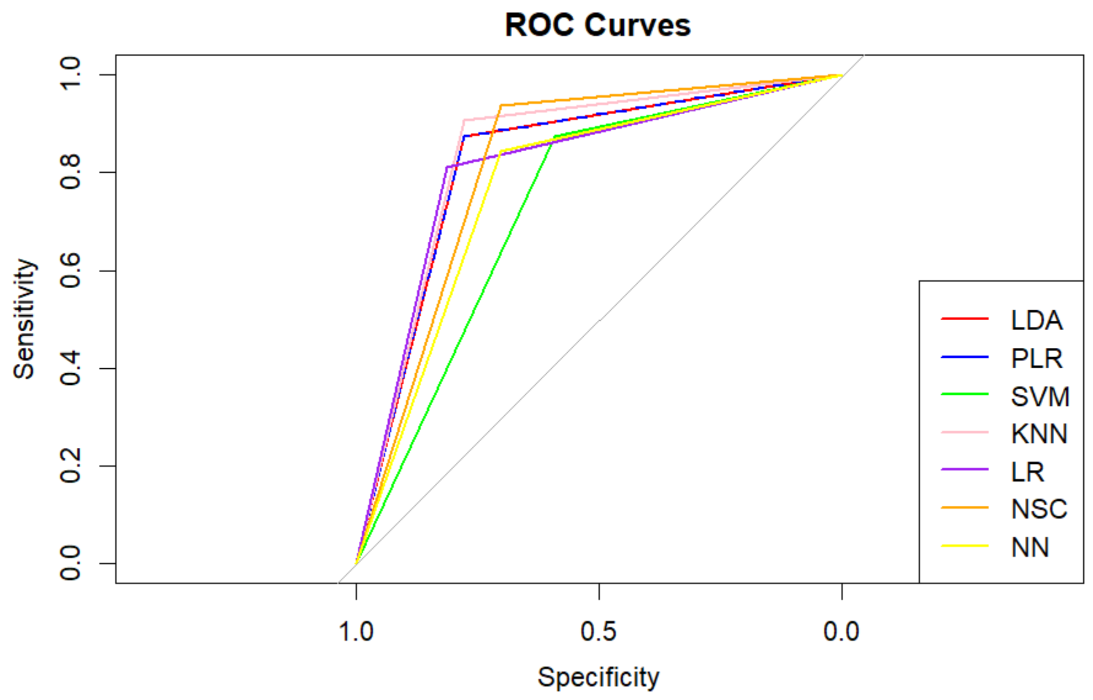


Figure 11

ROC-AUC Test Results



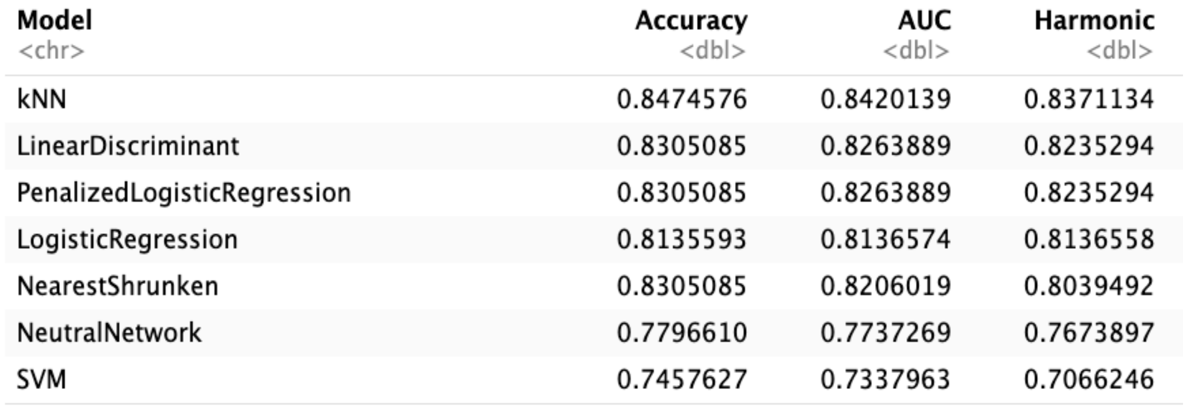
Results and Final Model Selection

Based on the harmonic mean and AUC model performance and test results, kNN performed the best in detecting heart disease. The best tuned model using the kNN algorithm used 20 neighbors to make a diagnostic prediction. The kNN model achieved .847 accuracy and .837 harmonic mean of sensitivity and specificity. According to the variable importance plot, the most signficant contributing factors are cp (chest pain), thal.2 (maximum heart rate achieve [1-3]), thalach (maximum heart rate), ca (number of major vessels [0-3] colored by fluoroscopy), and exang (exercise induced angina). These predictors are also the most important variables for the other models.

The neural network and support vector machine models underperform on the test data in every category.

Figure 12

Model Test Performance Results



Model <chr>	Accuracy <dbl>	AUC <dbl>	Harmonic <dbl>
kNN	0.8474576	0.8420139	0.8371134
LinearDiscriminant	0.8305085	0.8263889	0.8235294
PenalizedLogisticRegression	0.8305085	0.8263889	0.8235294
LogisticRegression	0.8135593	0.8136574	0.8136558
NearestShrunken	0.8305085	0.8206019	0.8039492
NeutralNetwork	0.7796610	0.7737269	0.7673897
SVM	0.7457627	0.7337963	0.7066246

Discussion and Conclusion

As previously mentioned, there are concerns about the accuracy and quality of this heart disease dataset. We have reason to believe that the target variables values (0 and 1) are mislabeled, resulting in opposite predictions. In a future possible iteration of this experiment, we would have liked to further verify the data source or use a direct source and have greater confidence that the target values are indeed labeled correctly.

Similarly, the overall quality of the data is an aspect of improvement to consider moving forward. The Kaggle dataset consists of a combination of heart disease data from Cleveland, Hungary, Switzerland, and Long Beach and is from 1988. This introduces a bias due to the observations being from limited locations as well as how well data from 1988 can be used to make predictions into and beyond 2024 given changes in health and lifestyle patterns. A portion of the next steps for this project include broadening the scope of location and conducting an analysis on more recent data. With newer data, this also opens up the possibility for additional predictor variables and more observations to explore. While the dataset size was sufficient for the scope of this project, a larger dataset could help reduce the confidence intervals of the evaluation metrics due to the Central Limit Theorem.

Lastly, we are inclined to further explore the true nature of confounding variables. Given the opportunity to have additional for developing more robust EDA, we hypothesize there may be other underlying relationships between predictors and the target variable that we were not able to uncover at this iteration of the experiment.

References

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York: Springer

Lapp, D. (2019). *Heart Disease Dataset*. Kaggle. Retrieved June 19, 2024, from https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/data

Simmons II, B. (2021, May). *Investigating Heart Disease Datasets and Building Predictive Models*. Retrieved June 21, 2024, from https://libres.uncg.edu/ir/ecsu/f/Brandon_Simmons_Thesis-Final.pdf

The top 10 causes of death. (2020, December 9). World Health Organization (WHO). Retrieved June 19, 2024, from https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

Appendix A

Rendered Source Code Output

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

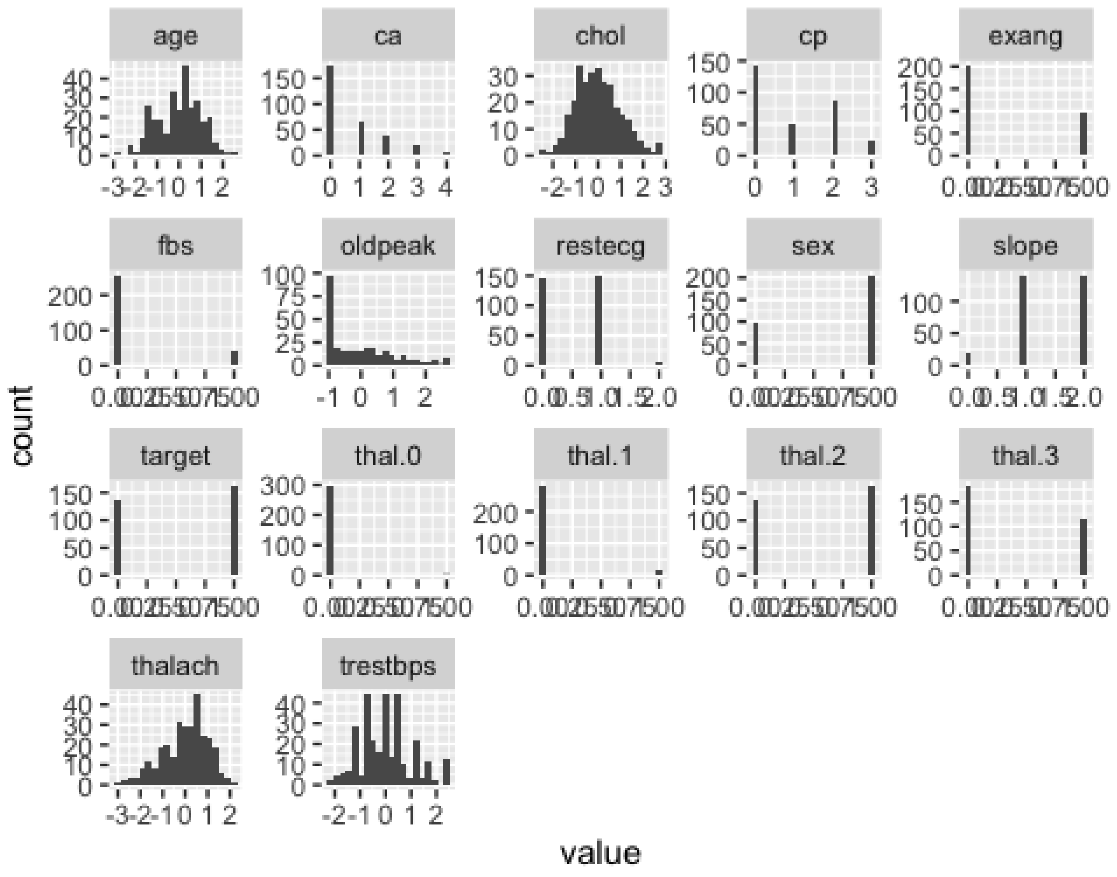
```
seed <- 123
#Ingest
data <- read.csv("heart.csv") #Change to your respective local path

#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  thalac
h
       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 tha
n #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 var
iables
cat('NewData dimension: ',nrow(data3),'remaining rows. This is still s
ufficient since ncol^2 is less than #nrows \n')

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

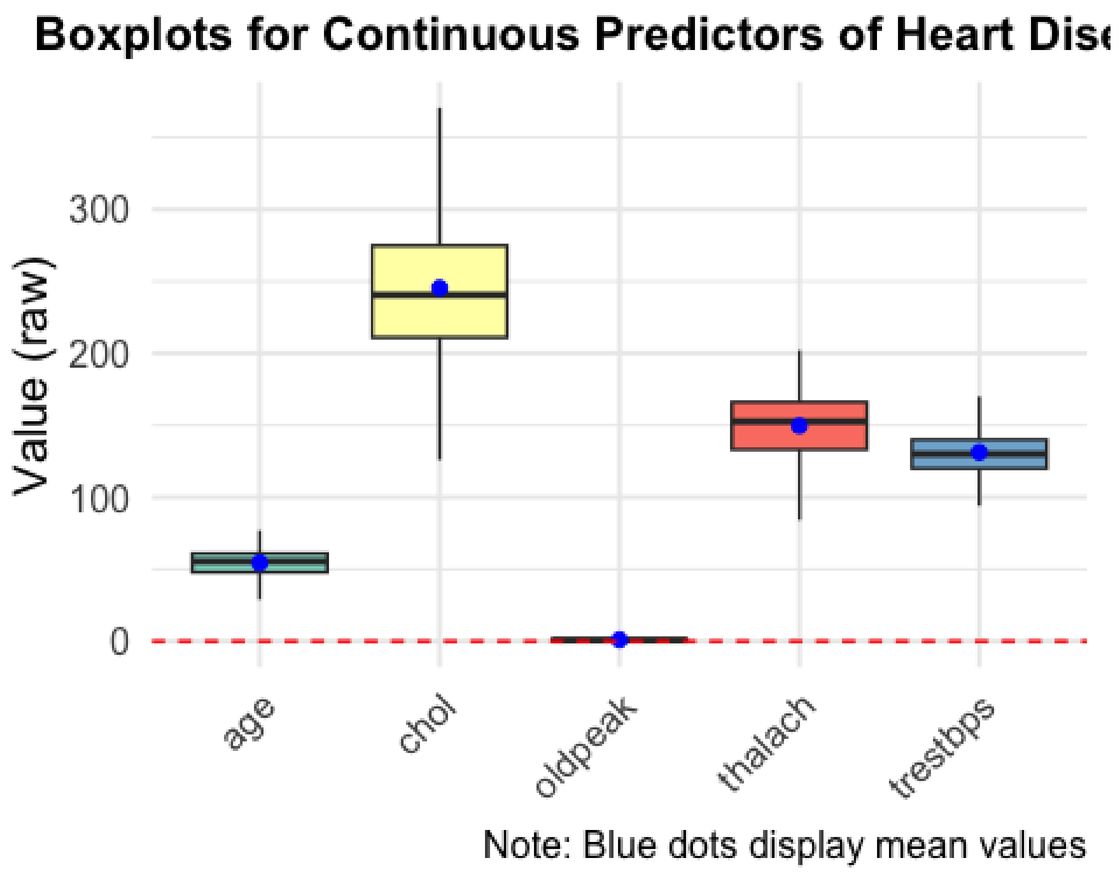
Exploratory Data Analysis

```
#Check column histograms for unusual distributions
ggplot(gather(data3), aes(value)) +
  geom_histogram(bins = 20) +
  facet_wrap(~key, scales = "free")
```

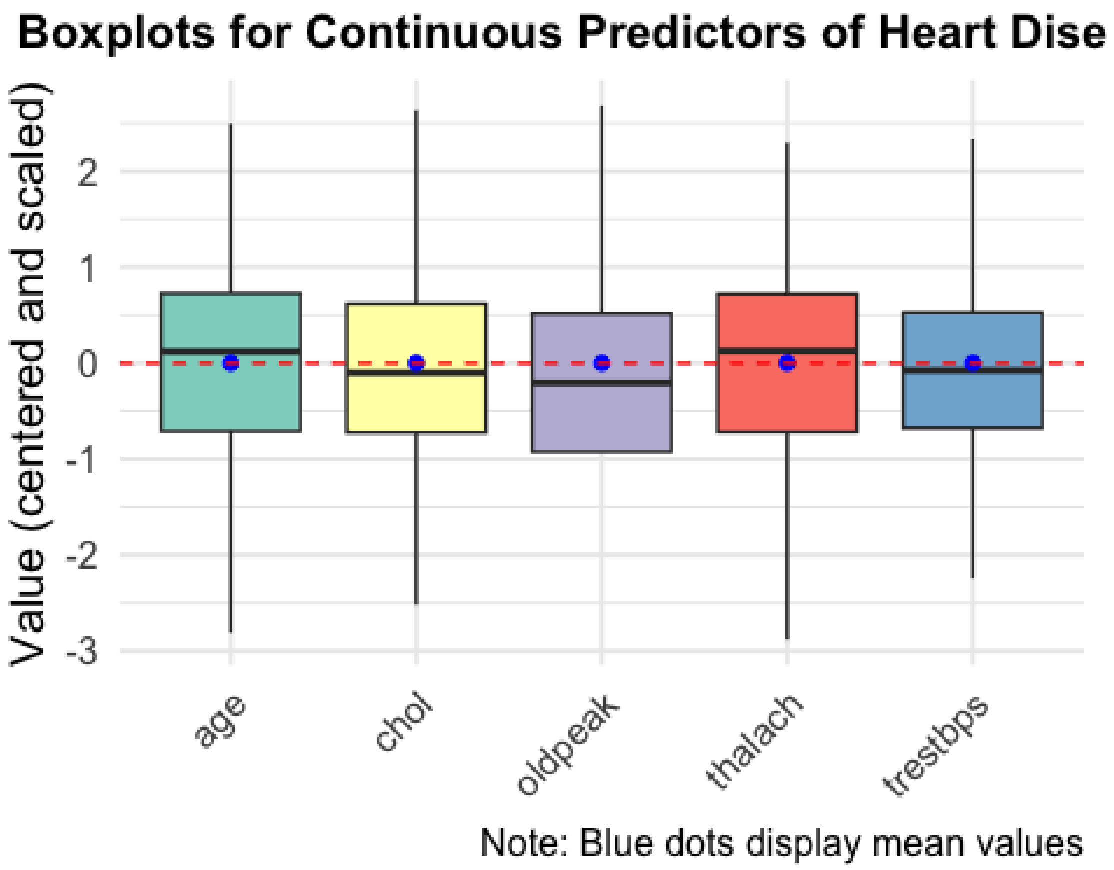


The predictors age and thalach are very slightly right skewed while the predictor oldpeak is left skewed. The predictor chol is approximately normally distributed.

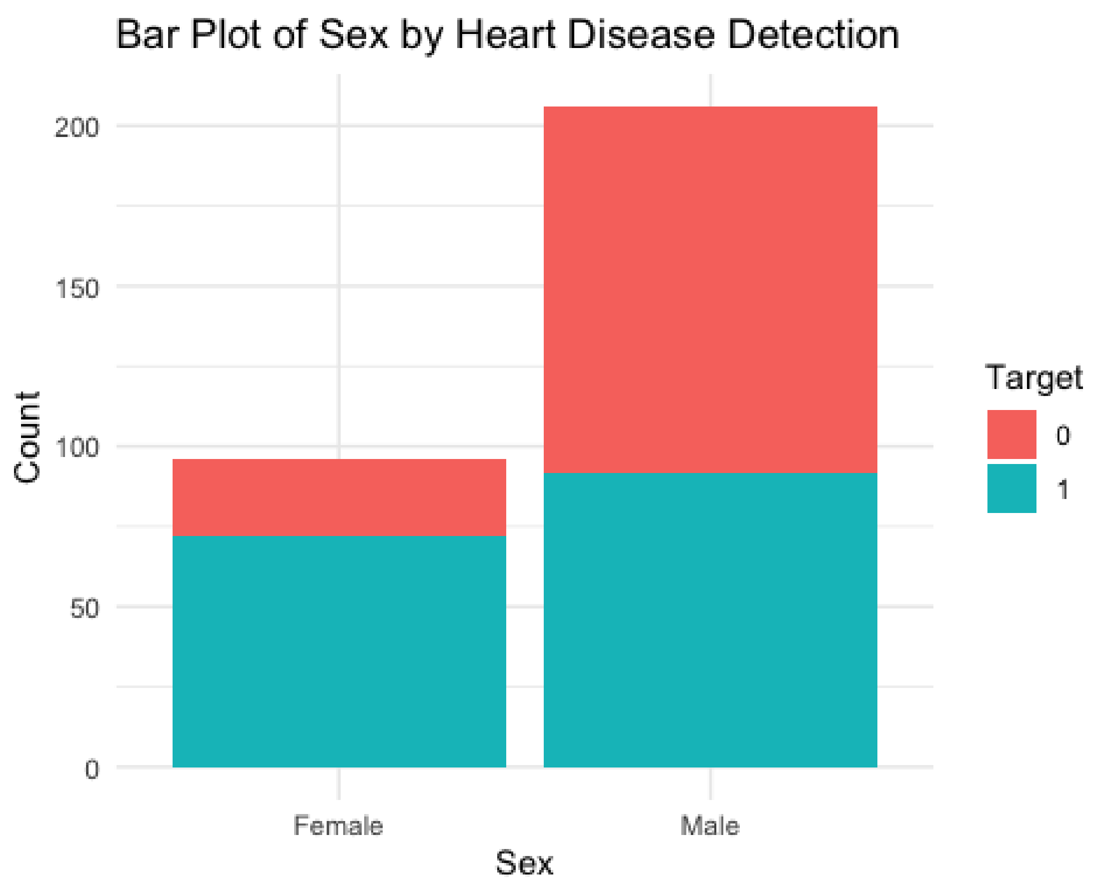
```r
cont_pred <- c('age', 'chol', 'oldpeak', 'thalach', 'trestbps')
cont_data <- data1[, cont_pred]
cont_data_long <- gather(cont_data)

# 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"))
```



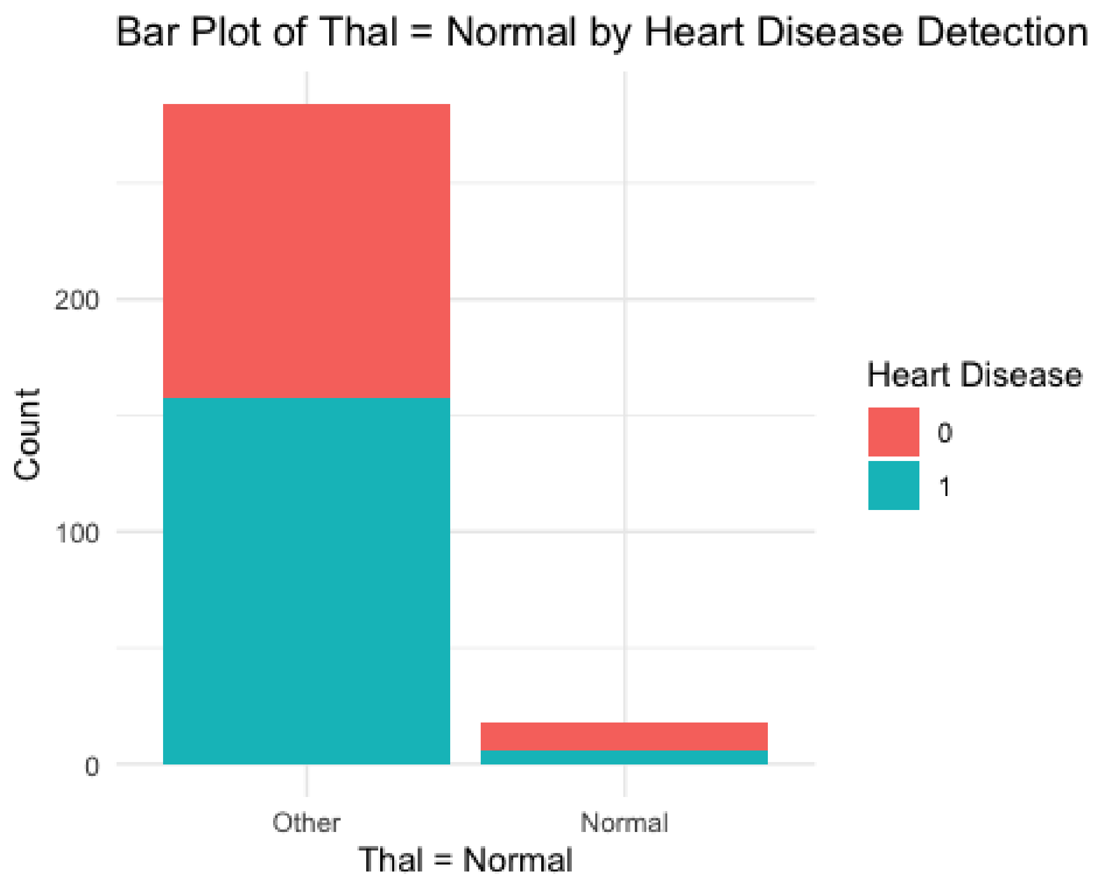
```
# continuous predictors
cont_pred <- c('age', 'chol', 'oldpeak', 'thalach', 'trestbps')
cont_data <- data3[, cont_pred]
cont_data_long <- gather(cont_data) #
ex_cont_data <- data3[,!names(data3) %in% cont_pred]
# boxplots
ggplot(cont_data_long, aes(x = key, y = value, fill = key)) +
  geom_boxplot(outlier.color = "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")
```



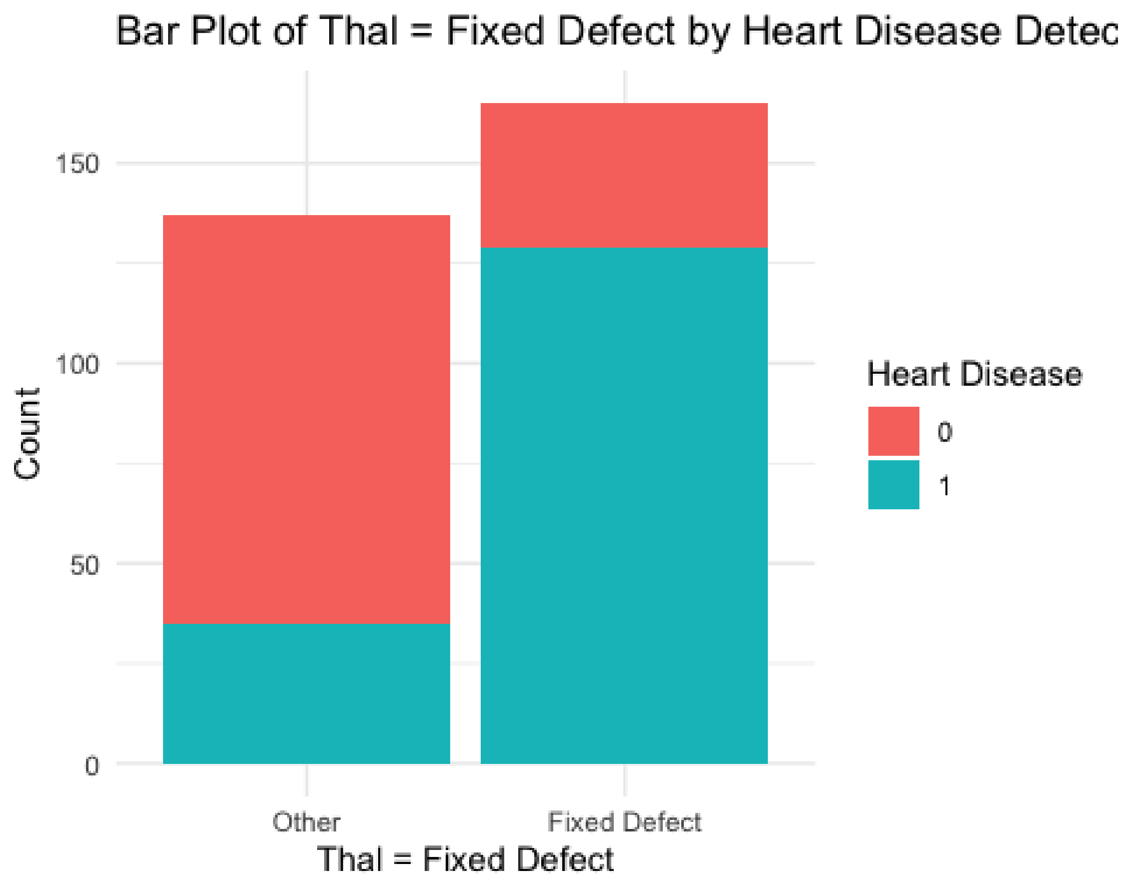
```r
data3$target <- as.factor(data3$target)
data3$sex <- as.factor(data3$sex)
# stacked bar plot for heart disease detection in males vs females

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()
```



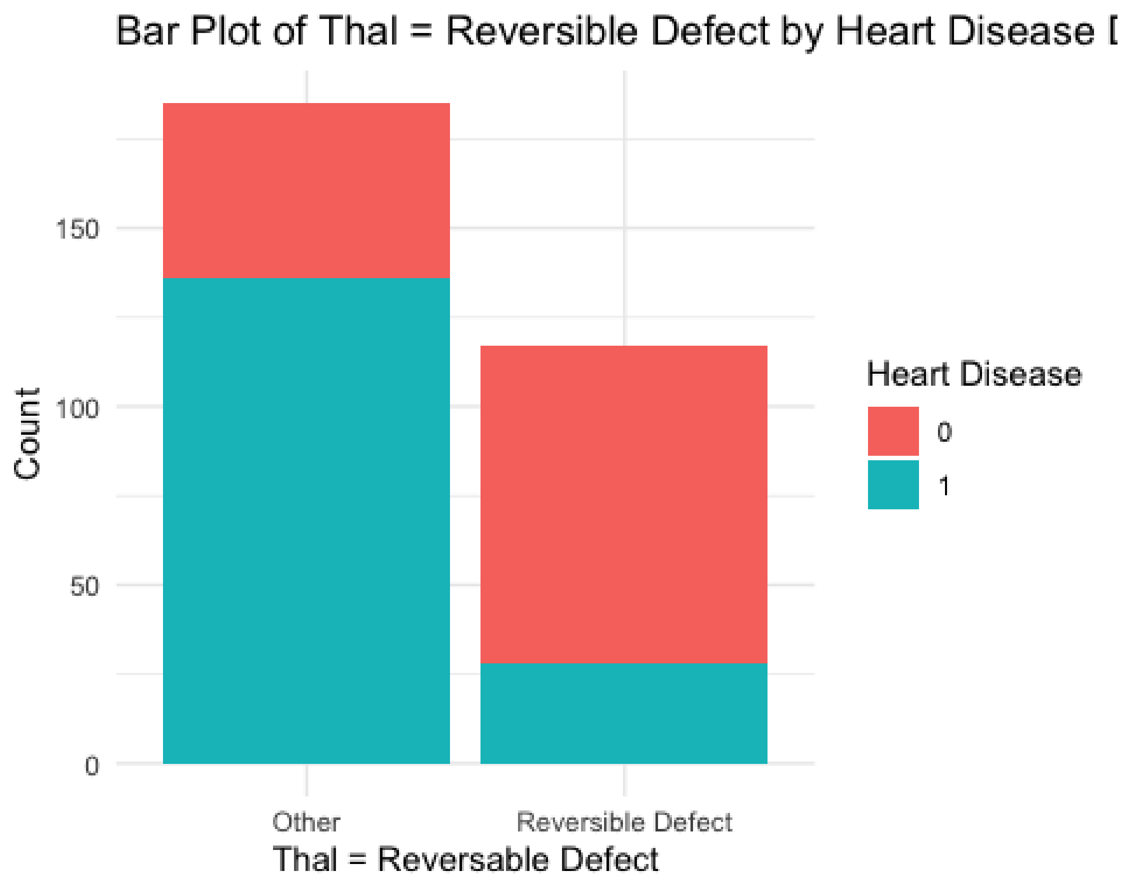
```r
data3$target <- as.factor(data3$target)
data3$thal.1 <- as.factor(data3$thal.1)
data3$thal.2 <- as.factor(data3$thal.2)
data3$thal.3 <- as.factor(data3$thal.3)

# 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()
```



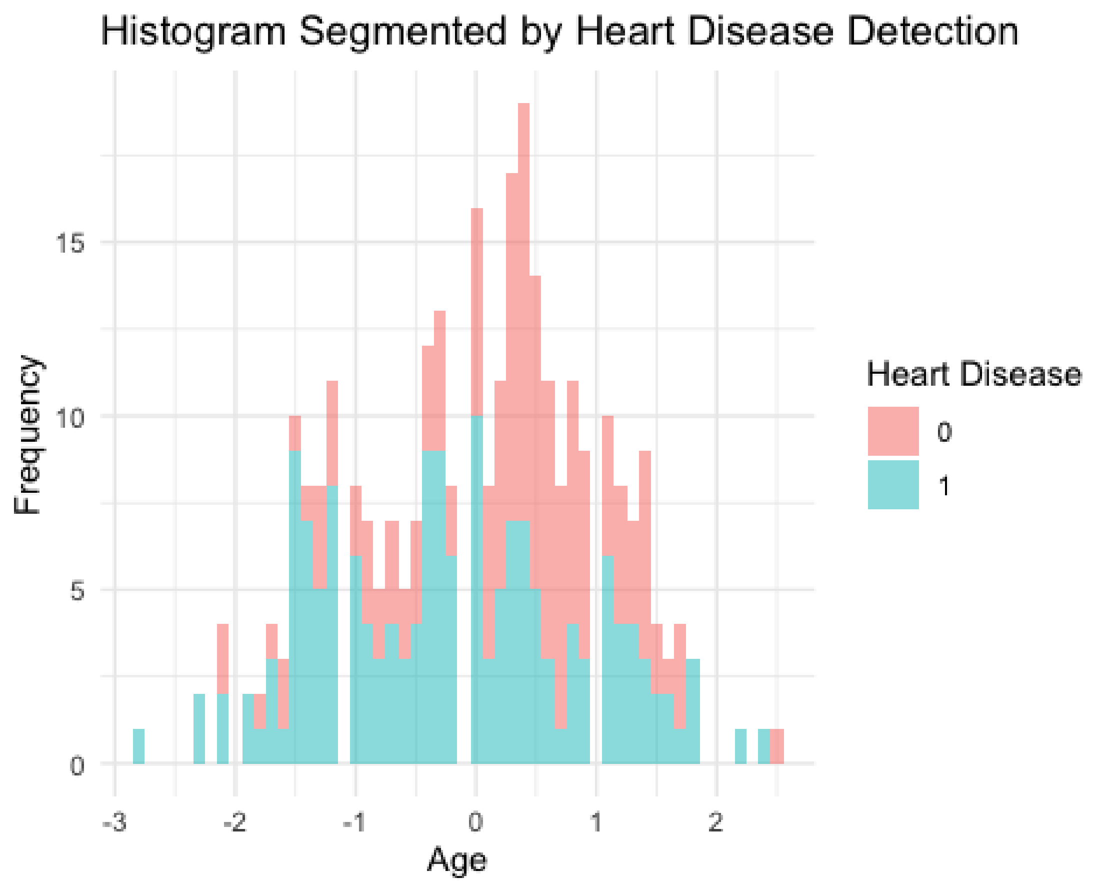
```
# stacked bar plot for heart disease detection in thal.2
ggplot(data3, aes(x = thal.2, fill = target)) +
  geom_bar() +
  labs(title = "Bar Plot of Thal = Fixed Defect by Heart Disease Detec
tion",
       x = "Thal = Fixed Defect",
       y = "Count",
       fill = "Heart Disease") +
  scale_x_discrete(labels = 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()
```

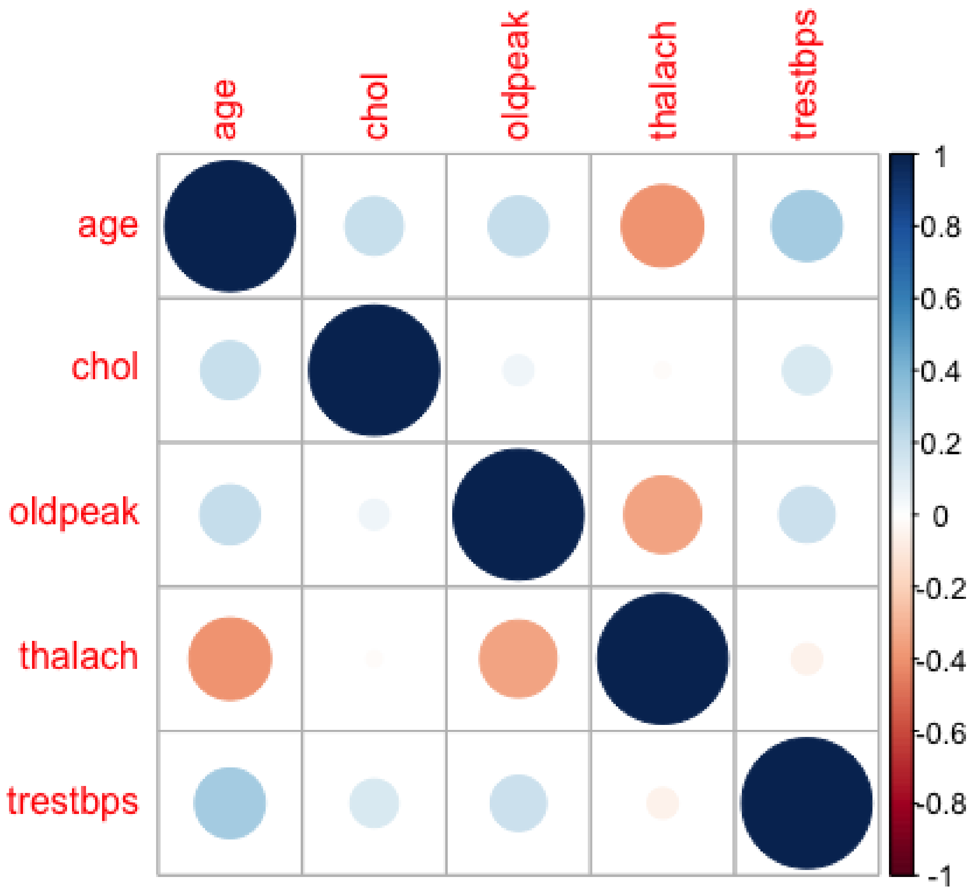


```
# histogram of age segmented by heart disease detection
ggplot(data3, aes(x = age, fill = target)) +
  geom_histogram(binwidth = .1, position = "stack", alpha = 0.5) +
  labs(title = "Histogram Segmented by Heart Disease Detection",
       x = "Age",
       y = "Frequency",
       fill = "Heart Disease")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

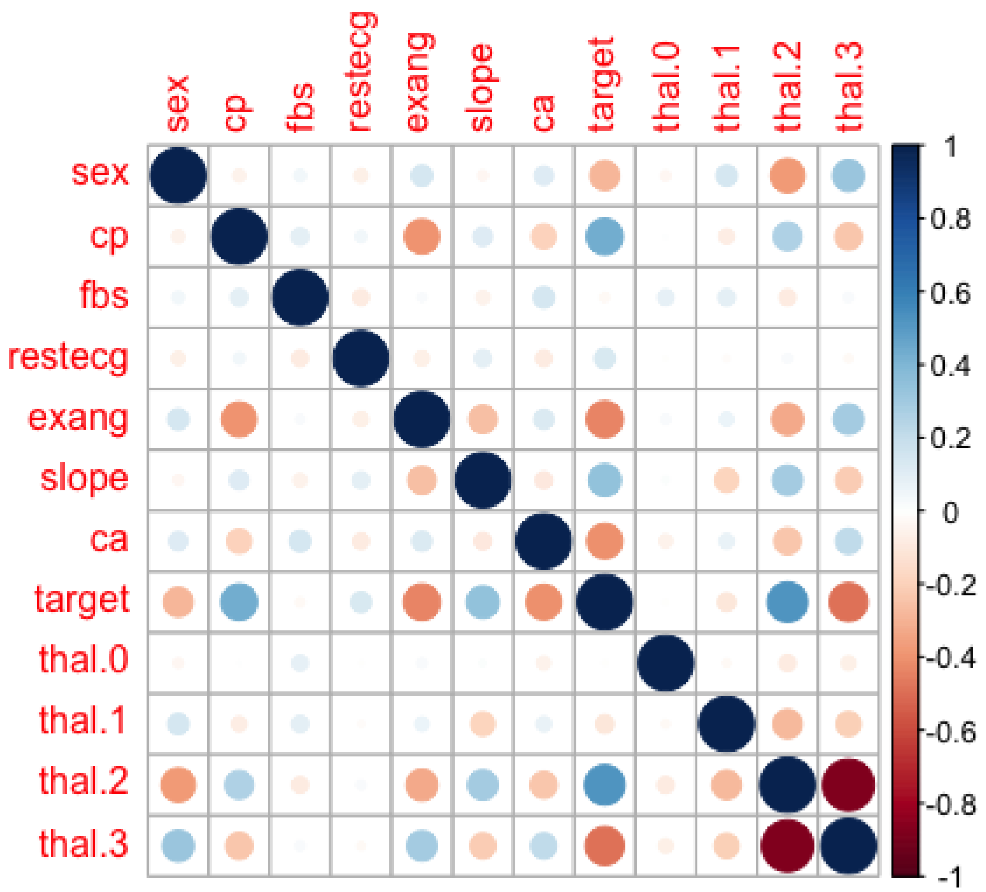


```
#Check for near-zero variance columns
nzv <- nearZeroVar(data3)
cat('Removed near zero predictor: ', colnames(data3)[nzv],'\n')

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 predict
ors: 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))
```

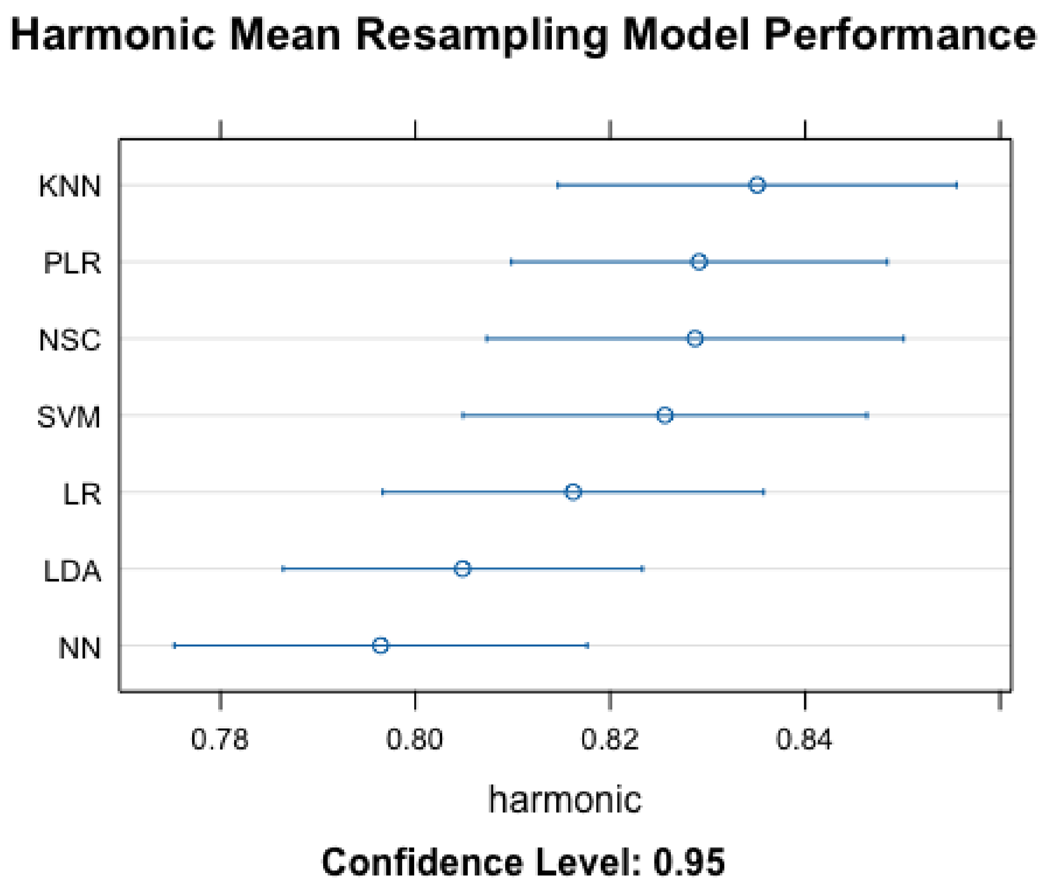
```r
# Support Vector Machine
set.seed(476)
suppressWarnings({
svm_model <-  train(x = train_x, y = train_y,
                  method = "svmRadial",
                  preProc = c("center", "scale"),
                  metric = "sens_spec_harm",
                  tuneLength = 14,
                  trControl = 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

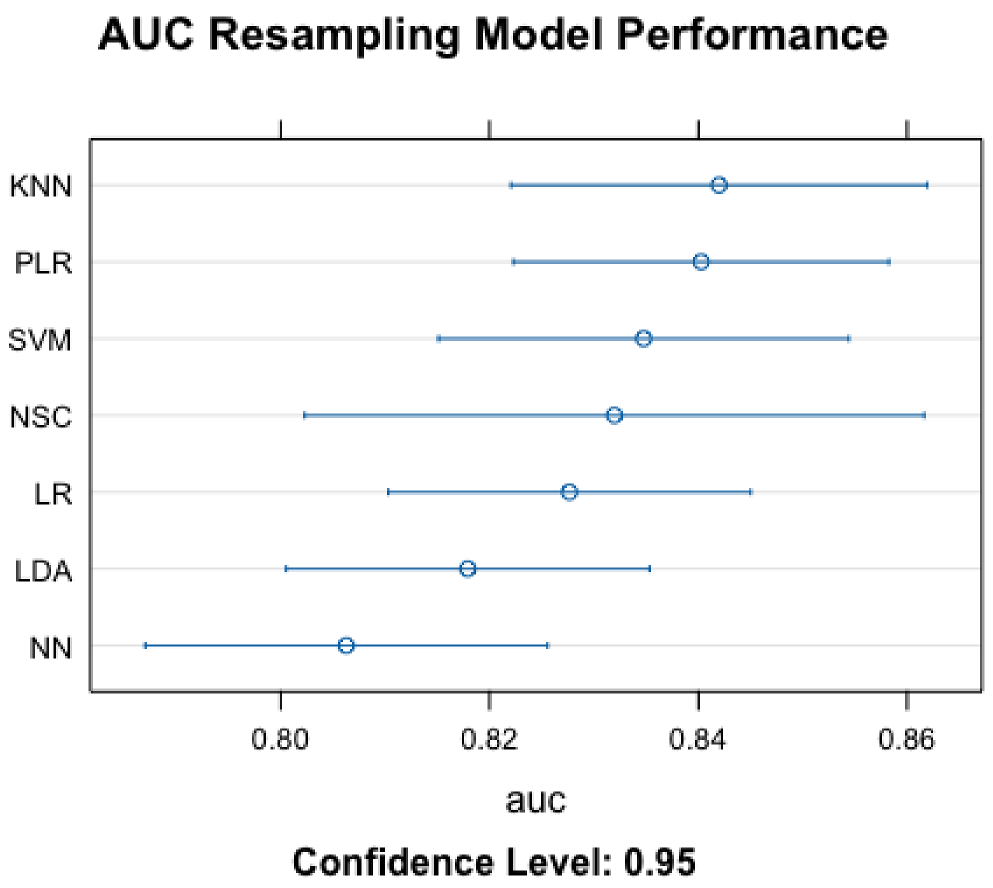
```r
#Harmonic
LR_sens_spec <- LR_model$results$harmonic
LDA_sens_spec <- LDA_model$results$harmonic
PLR_sens_spec <- mean(PLR_model$results$harmonic)
NSC_sens_spec <- mean(NSC_model$results$harmonic)

# 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")
```

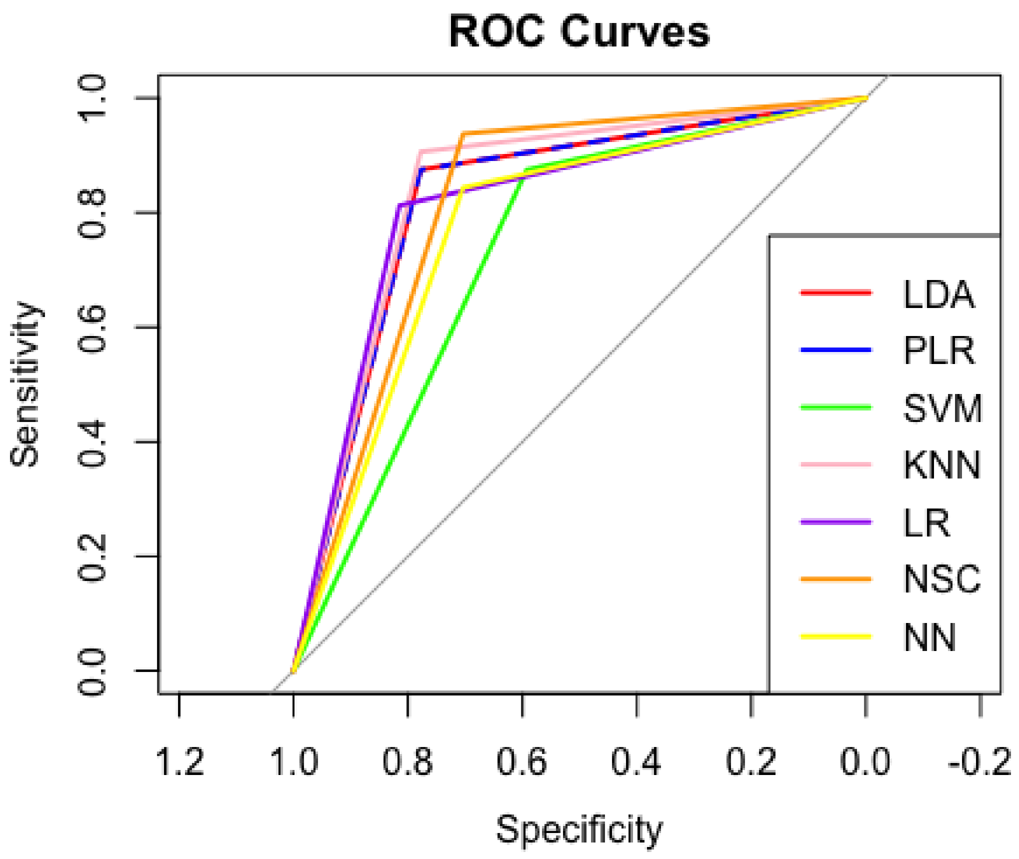


```
# convert all predictors to numeric
test_x <- data.frame(lapply(test_x, function(x)
  if (is.factor(x)) as.numeric(as.character(x)) else x))

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)
```

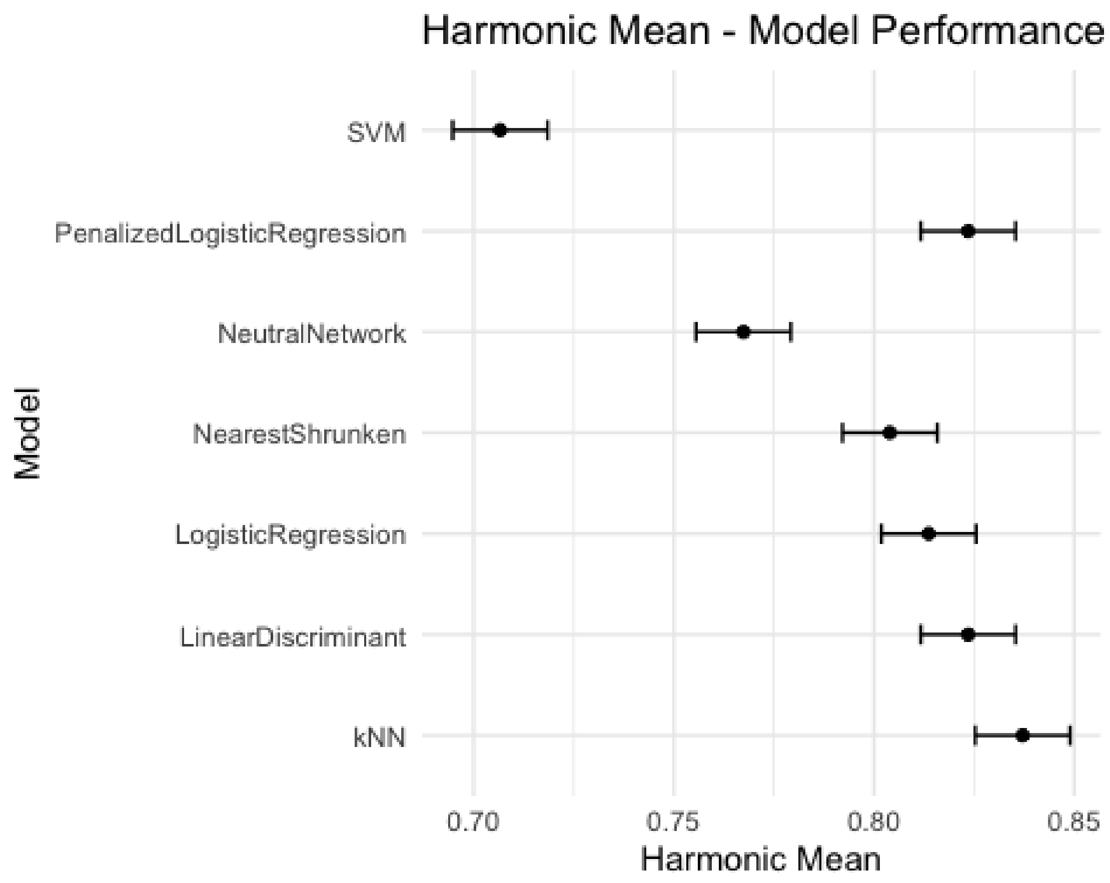
```
#ROC
LR_roc_pred <- suppressWarnings(roc(test_results$obs,
                                    as.numeric(test_results$LR)))

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.

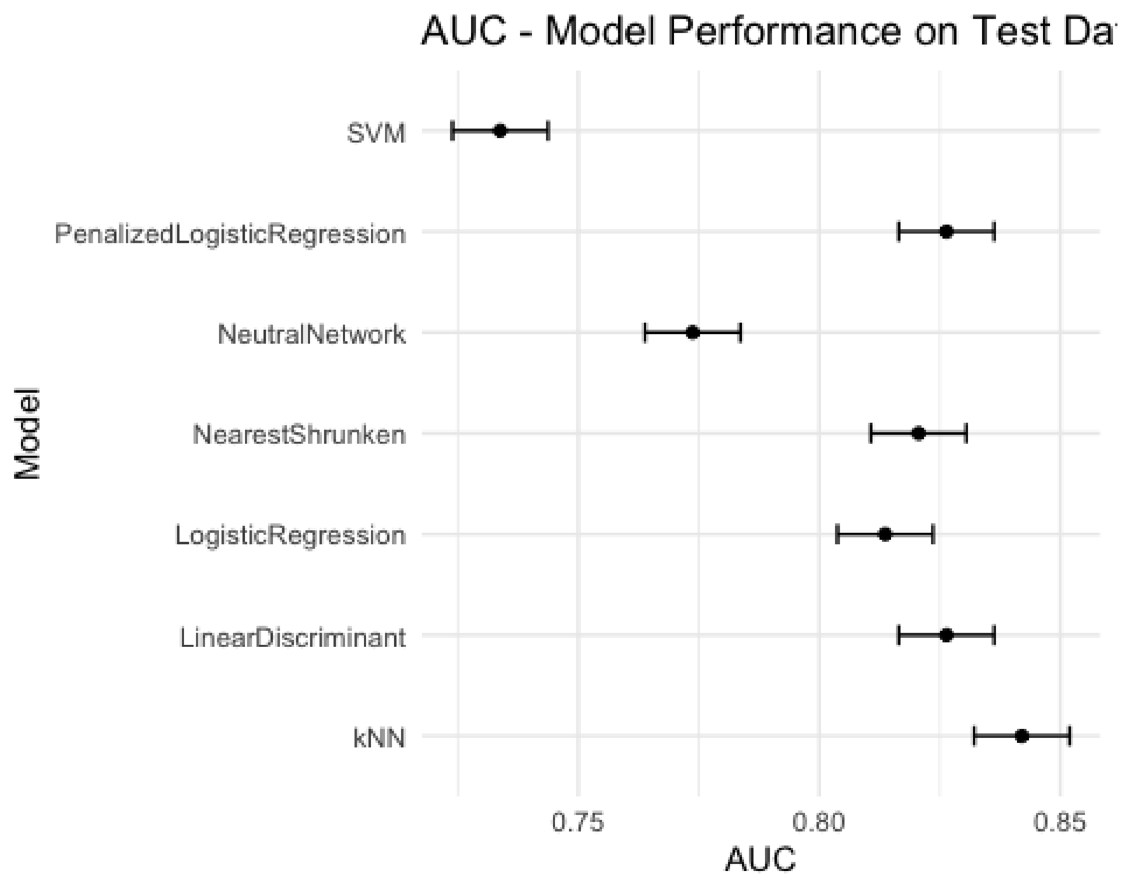
```
# compare ROC curves
plot(LDA_roc_pred, col = "red", main = "ROC Curves", lty = 1)
lines(PLR_roc_pred, col = "blue", lty = 2)
lines(svm_roc_pred, col = "green")
lines(knn_roc_pred, col = "pink")
lines(LR_roc_pred, col = "purple")
lines(NSC_roc_pred, col = "orange")
lines(nnet_roc_pred, col = "yellow")
legend("bottomright",
       legend = c("LDA", "PLR", "SVM", "KNN", "LR", "NSC", "NN"),
       col = c("red", "blue", "green", "pink", "purple", "orange", "ye
llow"),
       lwd = 2)
```



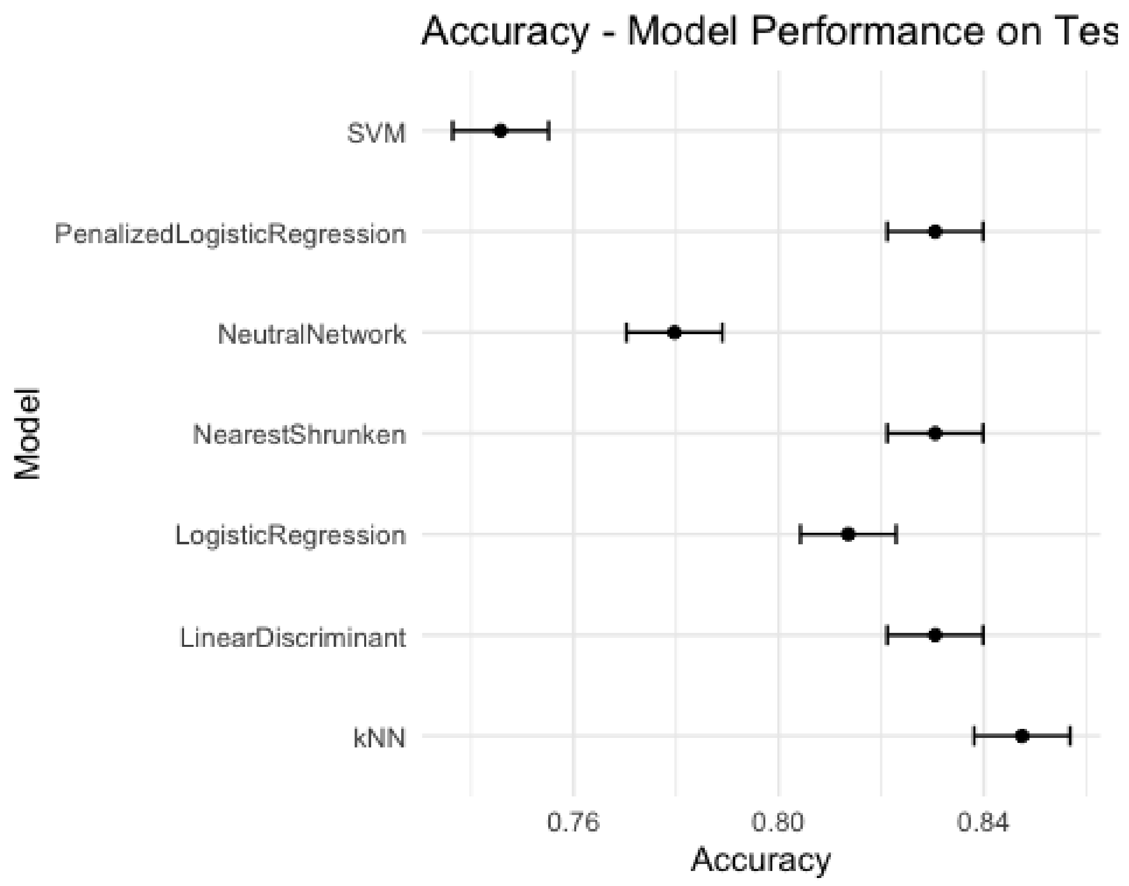
```
# calculate 95% confidence intervals for harmonic test results
test_harmonic <- Test_performance[, c('Model', 'Harmonic')] %>%
  mutate(
    LowerCI = Harmonic - qt(0.975, df = nrow(test_x) - 1) *
      (sd(Harmonic) / sqrt(nrow(test_x))),
    UpperCI = Harmonic + qt(0.975, df = nrow(test_x) - 1) *
      (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()
```



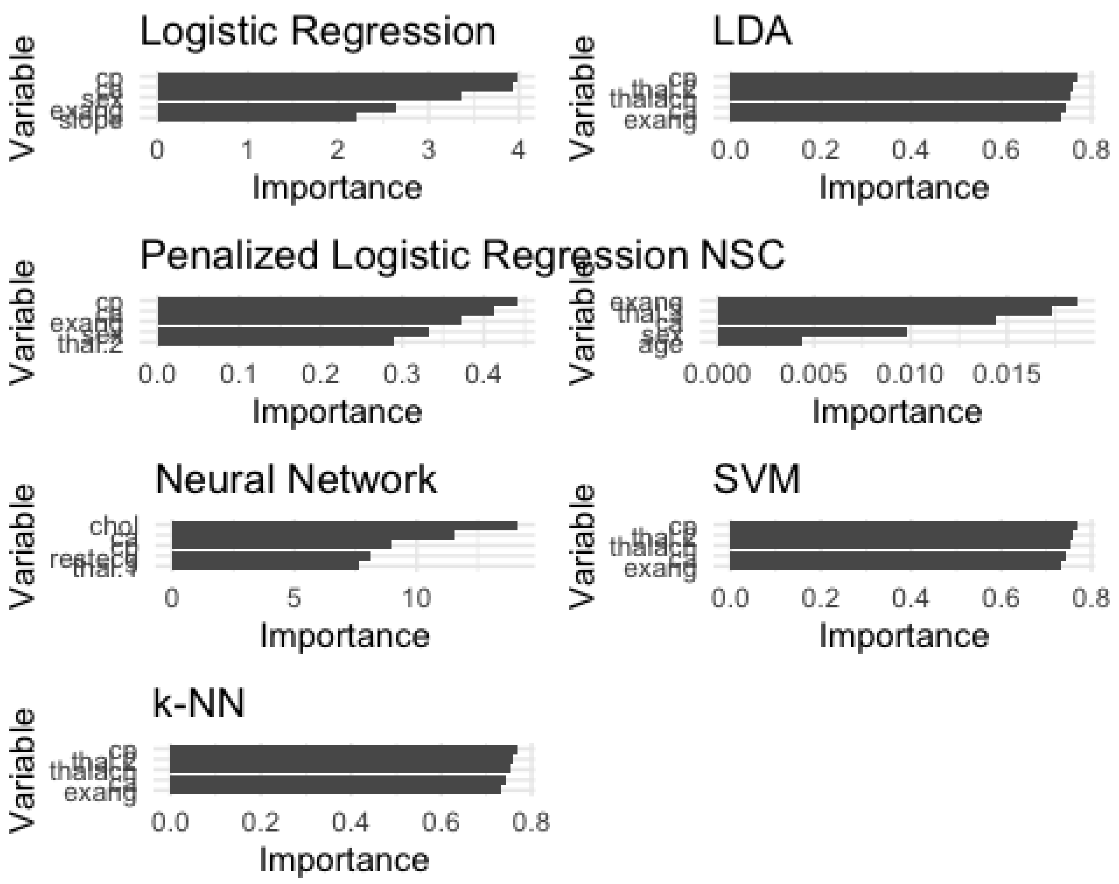
```
# calculate 95% confidence intervals for auc test results
test_auc <- Test_performance[, c('Model', 'AUC')] %>%
  mutate(
    LowerCI = AUC - qt(0.975, df = nrow(test_x) - 1) *
      (sd(AUC) / sqrt(nrow(test_x))),
    UpperCI = AUC + qt(0.975, df = nrow(test_x) - 1) *
      (sd(AUC) / sqrt(nrow(test_x)))
  )

# 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))
```

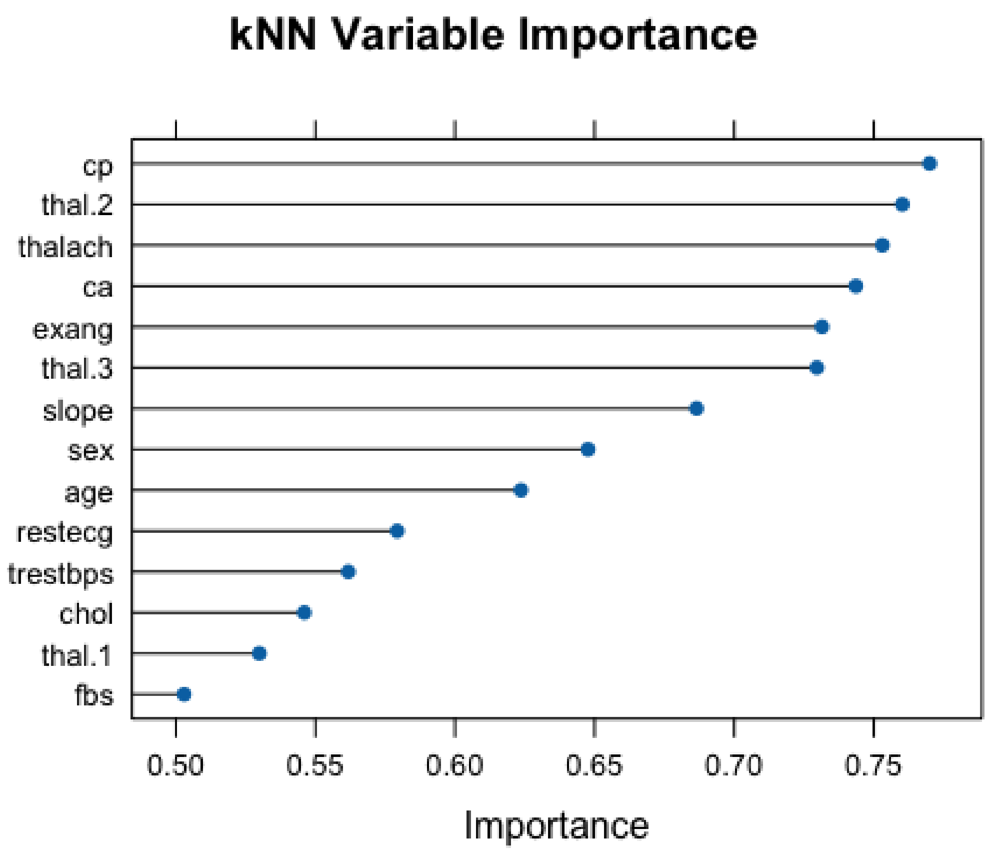


The most importance variables among all the models appear to be cp, ca, exang, thal.2, and sex.

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
```