Heart Disease Prediction

Gayathri Mamidibatthula

07/18/2021

The main goal of this project is to be able to accurately classify whether a patient has a heart disease or not based on diagnostic test data. The prediction is done using classification algorithms in R like SVM(Support Vector Machine) models and Random Forest.

The dataset used here is the Heart Disease UCI(https://archive.ics.uci.edu/ml/datasets/Heart+Disease) which contains 76 attributes, where a subset of 14 of them are used for the prediction.

In particular, the Cleveland Clinic Heart Disease dataset contains 14 variables related to patient diagnostics and one outcome variable indicating the presence or absence of heart disease.

Here is a summary of the variables used in the dataset:

1. Age: Age of subject
2. Sex: Gender of subject: 0 = female 1 = male
3. Chest-pain type: Type of chest-pain experienced by the individual: 1 = typical angina 2 = atypical angina 3 = non-angina pain 4 = asymptomatic angina
4. Resting Blood Pressure: Resting blood pressure in mm Hg
5. Serum Cholesterol: Serum cholesterol in mg/dl
6. Fasting Blood Sugar: Fasting blood sugar level relative to 120 mg/dl: 0 = fasting blood sugar <= 120 mg/dl 1 = fasting blood sugar > 120 mg/dl
7. Resting ECG: Resting electrocardiographic results 0 = normal 1 = ST-T wave abnormality 2 = left ventricle hyperthrophy
8. Max Heart Rate Achieved: Max heart rate of subject
9. Exercise Induced Angina: 0 = no 1 = yes
10. ST Depression Induced by Exercise Relative to Rest: ST Depression of subject
11. Peak Exercise ST Segment: 1 = Up-sloaping 2 = Flat 3 = Down-sloaping
12. Number of Major Vessels (0-3) Visible on Flouroscopy: Number of visible vessels under flouro
13. Thal: Form of thalassemia: 3 3 = normal 6 = fixed defect 7 = reversible defect
14. Diagnosis of Heart Disease: Indicates whether subject is suffering from heart disease or not: 0 = absence 1, 2, 3, 4 = heart disease present

Key Steps performed:

1. Loading Libraries and Importing the Dataset

2. Exploratory Data Analysis(EDA)
3. Data Cleaning
4. Building the ML Models
5. Validating the Results

LOADING LIBRARIES

```
#Loading Libraries

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(naniar)) install.packages("naniar", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: naniar
```

```
library(tidyverse)
library(readr)
library(ggplot2)
library(naniar)
library(caTools)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

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

IMPORTING DATASET

```
#Importing Dataset
setwd("C:/Users/gayathri_sri_mamidib/Desktop/Gayathri Imp/HDSC/Capstone2 - Heart Disease Prediction")
df<- read_csv("heart.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trestbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalach = col_double(),
##   exang = col_double(),
##   oldpeak = col_double(),
##   slope = col_double(),
##   ca = col_double(),
##   thal = col_double(),
##   target = col_double()
## )
```

```
head(df)
```

```
## # A tibble: 6 x 14
##     age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##   <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1    63     1     3      145   233     1       0     150     0     2.3     0
## 2    37     1     2      130   250     0       1     187     0     3.5     0
## 3    41     0     1      130   204     0       0     172     0     1.4     2
## 4    56     1     1      120   236     0       1     178     0     0.8     2
## 5    57     0     0      120   354     0       1     163     1     0.6     2
## 6    57     1     0      140   192     0       1     148     0     0.4     1
## # ... with 3 more variables: ca <dbl>, thal <dbl>, target <dbl>
```

```r
#Examining the data
str(df)
```

```
## spec_tbl_df [303 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age     : num [1:303] 63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : num [1:303] 1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : num [1:303] 3 2 1 1 0 0 1 1 2 2 ...
##  $ trestbps: num [1:303] 145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num [1:303] 233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : num [1:303] 1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : num [1:303] 0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : num [1:303] 150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : num [1:303] 0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num [1:303] 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : num [1:303] 0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : num [1:303] 0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : num [1:303] 1 2 2 2 2 1 2 3 3 2 ...
##  $ target  : num [1:303] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_double(),
##   ..   cp = col_double(),
##   ..   trestbps = col_double(),
##   ..   chol = col_double(),
##   ..   fbs = col_double(),
##   ..   restecg = col_double(),
##   ..   thalach = col_double(),
##   ..   exang = col_double(),
##   ..   oldpeak = col_double(),
##   ..   slope = col_double(),
##   ..   ca = col_double(),
##   ..   thal = col_double(),
##   ..   target = col_double()
##   .. )
```

All the variables in the data are in numerical form. Converting variables from integer to factors for further analysis:

```r
df$sex <- as.factor(df$sex)
df$cp <- as.factor(df$cp)
df$fbs <- as.factor(df$fbs)
df$restecg <- as.factor(df$restecg)
df$exang <- as.factor(df$exang)
df$slope <- as.factor(df$slope)
df$ca <- as.factor(df$ca)
df$thal <- as.factor(df$thal)
df$target <- as.factor(df$target)

str(df)
```

```
## spec_tbl_df [303 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age     : num [1:303] 63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
##  $ trestbps: num [1:303] 145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num [1:303] 233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
##  $ thalach : num [1:303] 150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ oldpeak : num [1:303] 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
##  $ ca      : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ thal    : Factor w/ 4 levels "0","1","2","3": 2 3 3 3 3 2 3 4 4 3 ...
##  $ target  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_double(),
##   ..   cp = col_double(),
##   ..   trestbps = col_double(),
##   ..   chol = col_double(),
##   ..   fbs = col_double(),
##   ..   restecg = col_double(),
##   ..   thalach = col_double(),
##   ..   exang = col_double(),
##   ..   oldpeak = col_double(),
##   ..   slope = col_double(),
##   ..   ca = col_double(),
##   ..   thal = col_double(),
##   ..   target = col_double()
##   .. )
```

2. EXPLORATORY DATA ANALYSIS(EDA)

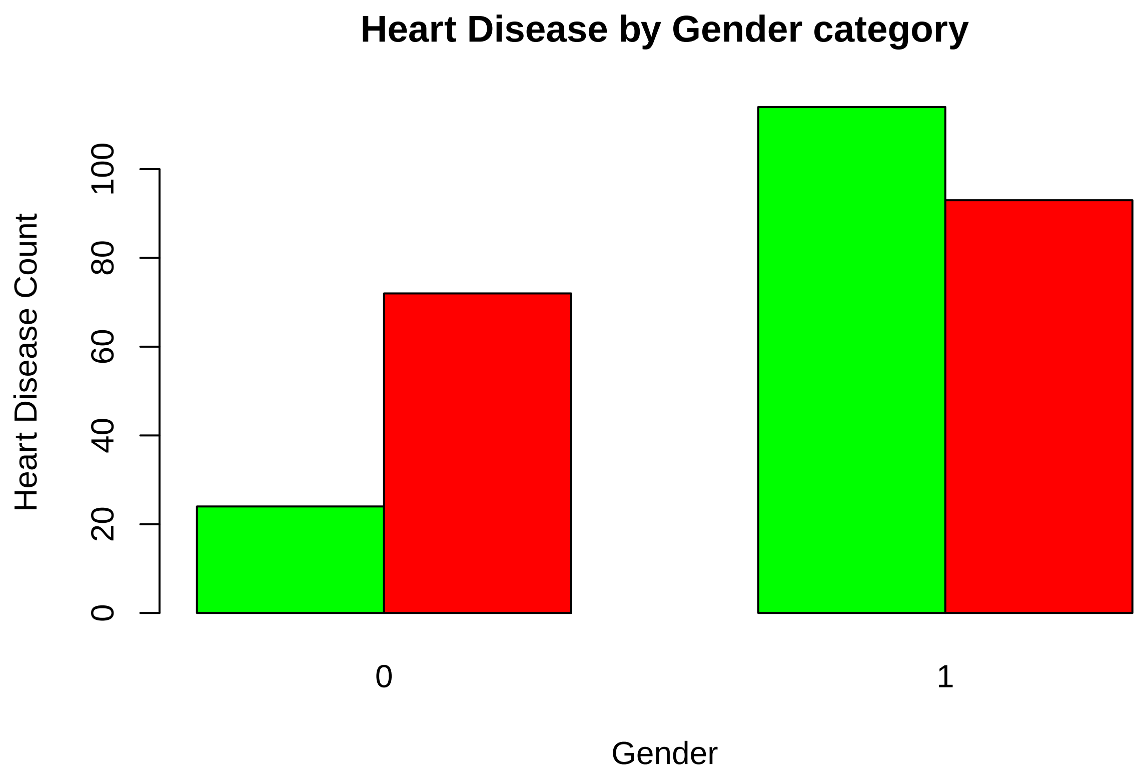
```
#Analyze no of people suffering from heart disease
table(df$target)
```

```
##
##   0   1
## 138 165
```

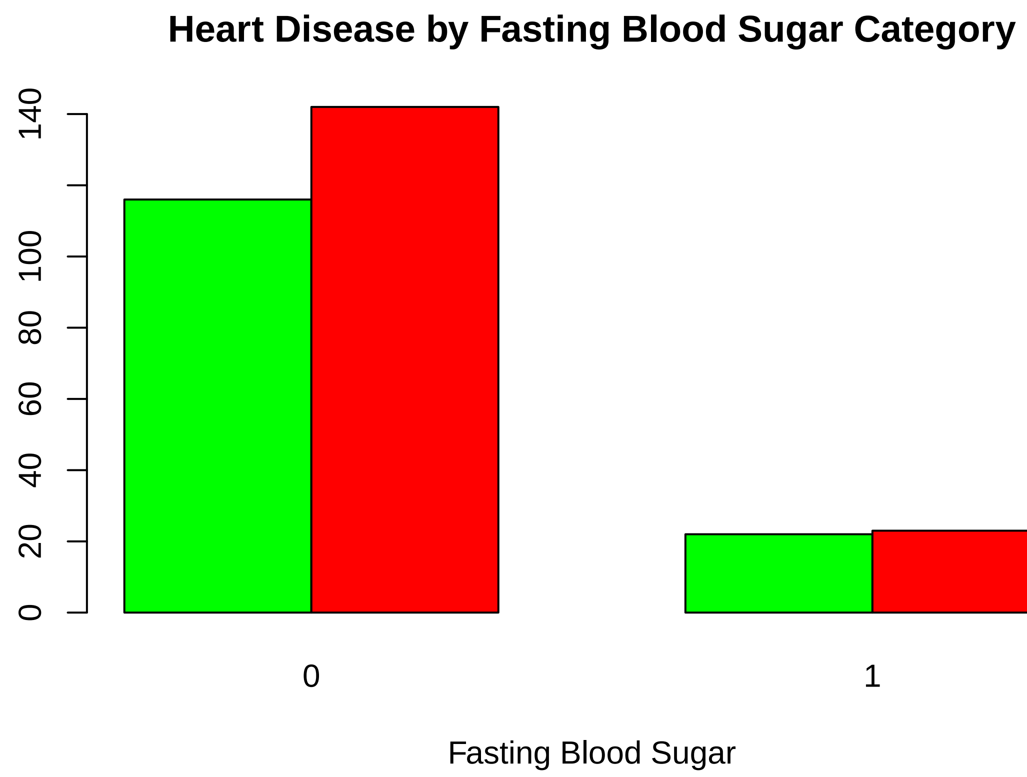
From the above analysis, it is deduced that the number of people who are suffering from heart disease is more than number of people who are not suffering from heart disease.

Plot Heart Disease Count based on Gender:

Variables used here: a) Sex: Male - 1, Female - 0 b) Target(whether patient has heart disease or not): Yes - 1, No - 0

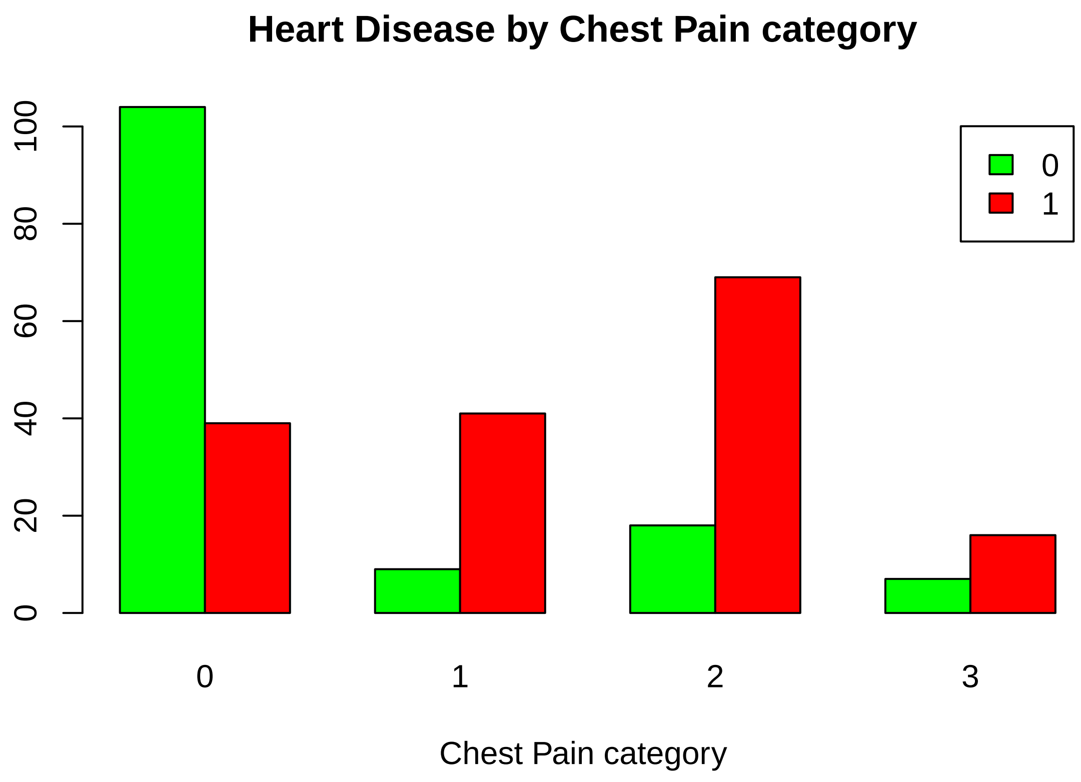


Plot Heart Disease Count by Fasting Blood Sugar



It can be see that people having fasting blood sugar (fbs) < 120 have more chance of having heart disease than people having fbs > 120.

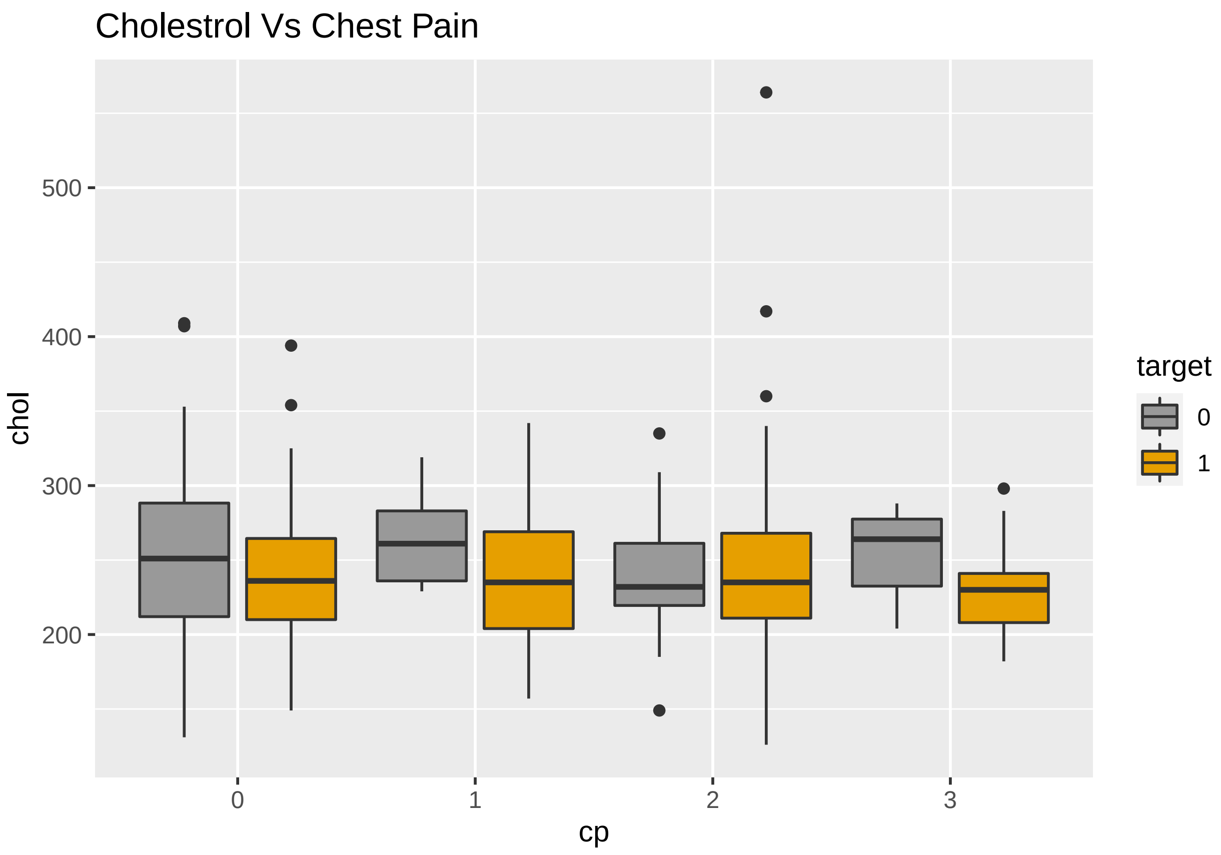
Heart Disease by Chest Pain category



There are 4-Levels of chest pain given in the barplot where 3 is highest

People who are on 3rd level of chest pain are very less as compared to people who are on 2nd level of chest pain. The inference here is that most people must have died after 2nd level of chest pain.

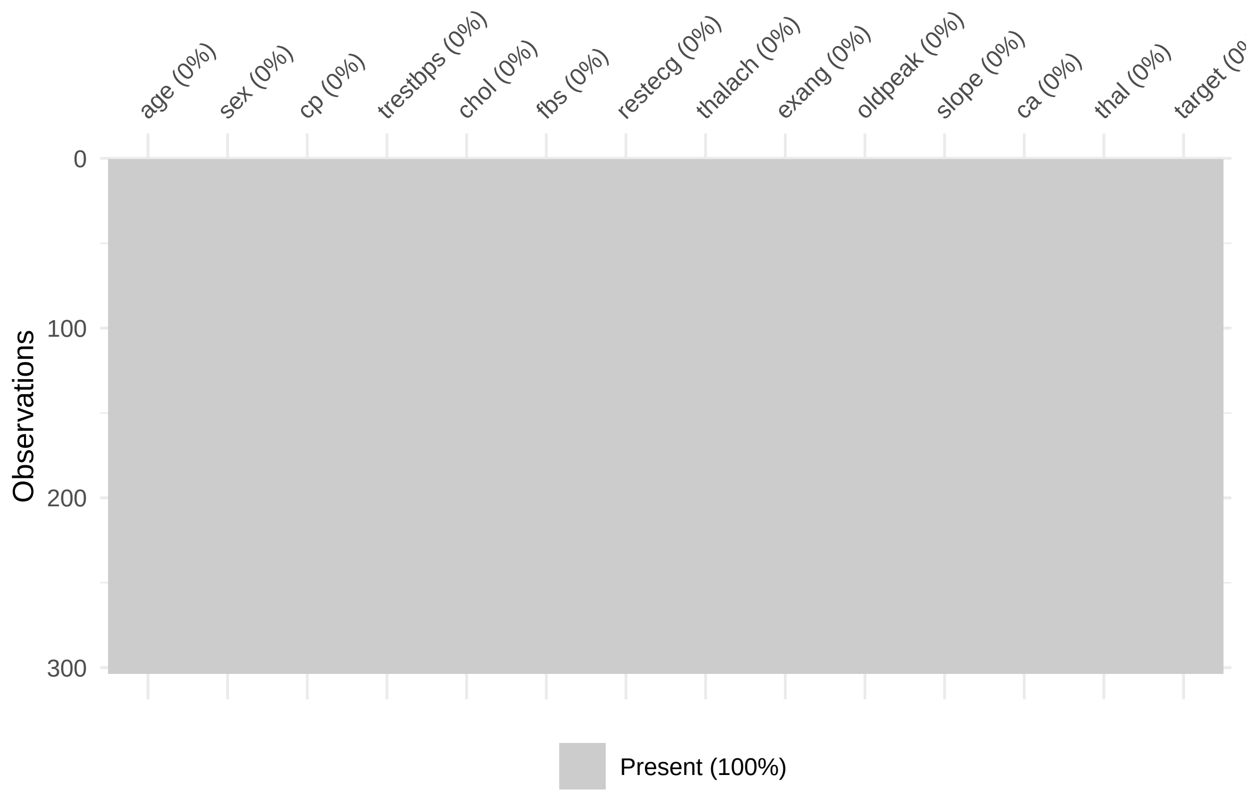
Relation between Cholestrol and Chest Pain



From above graph, we can say that if cholestrol is less than an approx value of 240 and chest pain is at level 3~4 then chances of having heart diseases are higher.

3. DATA CLEANING

Check for Missing data:



No missing data is found in the dataset. Next step is building the ML models.

4. BUILDING THE ML MODELS

```
#set.seed(123)
#Splitting the data into training set and test set:

split = sample.split(df$target, SplitRatio = 0.8)
training_set = subset(df, split = TRUE)
test_set = subset(df, split = FALSE)

#Scaling numeric values:

training_set[ , c(1,4,5,8, 10)] =  scale(training_set[, c(1,4,5,8, 10)])
test_set[ , c(1,4,5,8, 10)] = scale(test_set[ , c(1,4,5,8, 10)])
```

a) SVM(Support Vector Machine) Model:

Fitting SVM to the training set. First we will check the accuracy by using linear kernel:

```
#Fitting SVM to training set using linear kernel:

classifier_svm = svm(formula = target ~ .,
                     data = training_set,
                     type = 'C-classification',
                     kernel = 'linear')

# Predicting the Test set results
y_pred_svm = predict(classifier_svm, newdata = test_set[-14])

confusionMatrix(y_pred_svm , test_set$target)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 112  11
##          1  26 154
##
##                Accuracy : 0.8779
##                  95% CI : (0.8356, 0.9125)
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7516
##
##  Mcnemar's Test P-Value : 0.02136
##
##             Sensitivity : 0.8116
##             Specificity : 0.9333
##          Pos Pred Value : 0.9106
##          Neg Pred Value : 0.8556
##              Prevalence : 0.4554
##          Detection Rate : 0.3696
##    Detection Prevalence : 0.4059
##       Balanced Accuracy : 0.8725
##
##        'Positive' Class : 0
##
```

Test Results for SVM using Linear Kernel: Accuracy: 87.79% Sensitivity/Recall: 81.16% Specificity: 93.33%

We will now check the accuracy by using radial kernel:

```
#Fitting SVM to training set using radial kernel:
classifier_svm = svm(formula = target ~ .,
                     data = training_set,
                     type = 'C-classification',
                     kernel = 'radial')

# Predicting the Test set results
y_pred_svm = predict(classifier_svm, newdata = test_set[-14])

confusionMatrix(y_pred_svm , test_set$target)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 116  20
##          1  22 145
##
##                Accuracy : 0.8614
##                  95% CI : (0.8173, 0.8982)
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7202
##
##  Mcnemar's Test P-Value : 0.8774
##
##             Sensitivity : 0.8406
##             Specificity : 0.8788
##          Pos Pred Value : 0.8529
##          Neg Pred Value : 0.8683
##              Prevalence : 0.4554
##          Detection Rate : 0.3828
##    Detection Prevalence : 0.4488
##       Balanced Accuracy : 0.8597
##
##        'Positive' Class : 0
##
```

Test Results for SVM using radial kernel: Accuracy: 86.14% Sensitivity/Recall: 84.06% Specificity: 87.88% It can be deduced that linear kernel is better than radial kernel for SVM. Hence, classification is linear here.

c) Random Forest

```r
#Fitting Random Forest Model

classifier_rf = randomForest(x = training_set[-14], y = training_set$target, ntree = 400)

# Predicting the Test set results
y_pred_rf = predict(classifier_rf, newdata = test_set[-14])

confusionMatrix(y_pred_rf , test_set$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 138   0
##          1   0 165
##
##                Accuracy : 1
##                  95% CI : (0.9879, 1)
##     No Information Rate : 0.5446
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
```

```
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4554
##          Detection Rate : 0.4554
##    Detection Prevalence : 0.4554
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

Test Results using Random Forest: Accuracy: 100% Sensitivity/Recall: 100% Specificity: 100%

Since accuracy is 100%, we need to check for the possibility of overfitting.

To check overfitting, we will apply repeated k-cross validation with 10 fold.

```r
#Check possibility of overfitting using repeated k-cross validation:

# define training control
train_control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(target~., data=df, trControl=train_control, method="nb")
# summarize results
print(model)
```

```
## Naive Bayes
##
## 303 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 273, 273, 273, 272, 274, 273, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8325794  0.6603303
##    TRUE      0.7998628  0.5888622
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE and adjust
##  = 1.
```

After performing repeated k-fold cross validation, the final accuracy is around 82.2%, showing that there was some overfitting in the model.

5. Validation of Results

The final results show the following accuracy values:

1. SVM using linear kernel: 87.79%
2. SVM using radial kernel: : 86.14%
3. Random Forest: 100%
4. Random Forest(repeated k-fold cross validation): 82.2%

Conclusion:

It can be deduced that Random Forest has the best accuracy compared to all models, but has a significant effect of overfitting as seen by repeated k-fold cross validation. The accuracy of the model can be improved by reducing overfitting.