Heart Failure Prediction

Cloud Nine (OCC2 G9)

15 June 2023

Introduction

Millions of individuals throughout the world suffer from heart failure, making it a major health problem. Heart failure is a serious medical condition that occurs when the heart muscles become weakened and unable to pump blood effectively, resulting in a diminished supply of oxygen to the tissues of the body. The early detection and precise diagnosis of heart failure can considerably improve the prognosis and outcome of afflicted persons. With the advent of machine learning, it is now possible to develop predictive models that can identify and classify cases of heart failure with high accuracy.

This project's objective is to develop machine learning algorithm capable of accurately recognizing and classifying heart failure cases. This entails feeding patient data into the algorithm and enabling it to determine whether the patient is afflicted with heart failure. By analyzing patient data and identifying patterns indicating the presence of the disease, machine learning algorithms can aid in the early detection of heart failure. With the use of this dataset, machine learning models may be trained to recognize heart failure using patient data such as demographics, medical history, and results of clinical tests.

The project will involve identifying, developing, and training a machine learning algorithm, with the end aim of creating a tool that can be used in clinical settings to assist healthcare professionals in the early detection and accurate diagnosis of heart failure. The project intends to improve the diagnosis capability for heart failure disease by attaining an extensive level of precision.

Objectives

The main objectives of the project:

1. Identify the most important parameters in the dataset for accurately classifying Heart Failure cases.
2. Train machine learning algorithm to recognize and classify Heart Failure cases with precision.
3. Acquire a high level of classification precision to enhance the diagnosis and treatment of heart failure.

Project Goals

The main goals of the project:

1. To leverage the dataset so that to gain a better understanding of the factors influencing heart disease and to build models that can predict the likelihood of heart disease occurrence accurately, enabling early detection and intervention.
2. Uncover the key parameters within the dataset that are crucial for the precise identification of Heart Failure instances.
3. Develop a machine learning model capable of accurately identifying and categorizing cases of Heart Failure.
4. Achieve a high degree of accuracy in classification to improve the diagnosis process and the effectiveness of heart failure treatment strategies.

Asking Questions

Below shows a few possible questions that may arise prior or during data analysis. These questions can be classified into two types: regression question and classification question. Regression questions involve anticipating a continuous numeric output, while classification questions involve expecting a discrete categorical value as the output.

Regression question:

1. What is the possible correlation between an individual's age and the resting blood pressure?
2. Is it possible to predict an individual's cholesterol level based on their age?
3. How does the cholesterol level relate to other variables, considering it was initially rejected by the Boruta algorithm? *(This question arises during feature selections process.)*
4. Is there an observable relationship between Oldpeak and ST_Slope, given that Oldpeak is commonly used as a feature to evaluate the severity of ST segment depression? *(This question arises from the skewed histogram observed for Oldpeak.)*

Classification question:

1. Can an individual being accurately classified as having heart disease or not based on their age, sex, chest pain type, resting blood pressure, cholesterol level, etc.?

Data Acquisition

Data Source

Dataset title: **Heart Failure Prediction Dataset**

Dataset source: Kaggle (last update: 2021)

This dataset was created in September 2021 by FEDESORIANO, by combining five independent heart disease datasets with over 11 common features.

The five datasets that have been used as per below:

- Cleveland [Year 1990]: 303 observations
- Hungarian [Year 1990]: 294 observations
- Switzerland [Year 1989]: 123 observations
- Long Beach VA [Year 1989]: 200 observations
- Stalog (Heart) Data Set [Year 1990]: 270 observations

Total: 1190 observations

Duplicated: 272 observations

Final dataset: 918 observations

Data Feature & Purpose

Importing Libraries

```
library(data.table)
library(caTools)
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
library(caret)
```

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

```
library(Boruta)
library(rpart)
library(rpart.plot)
library(knitr)
library(RColorBrewer)
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

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

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

Load & Read Data

```
mydata <- read.csv("heart.csv")
head(mydata)
```

```
##   Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
## 1  40   M           ATA       140         289         0     Normal   172
## 2  49   F           NAP       160         180         0     Normal   156
## 3  37   M           ATA       130         283         0         ST    98
## 4  48   F           ASY       138         214         0     Normal   108
## 5  54   M           NAP       150         195         0     Normal   122
## 6  39   M           NAP       120         339         0     Normal   170
##   ExerciseAngina Oldpeak ST_Slope HeartDisease
## 1              N     0.0       Up            0
## 2              N     1.0     Flat            1
## 3              N     0.0       Up            0
## 4              Y     1.5     Flat            1
## 5              N     0.0       Up            0
## 6              N     0.0       Up            0
```

Result above shows a glimpse of 1st 6 rows of raw data set obtained directly from source.

As shown above, the final dataset has 918 observations & 12 variables.

With 11 common clinical features and 1 heart disease output class found in the dataset, this dataset may serve a purpose to predict the possibility of heart disease outcome.

Data Understanding

Data understanding involves exploring the data acquired, assessing its quality and gaining insights into its characteristics and structure.

Data Shape

```
dim(mydata)
```

```
## [1] 918  12
```

Data consisting of 918 rows and 12 columns. Consistent to as introduced in data feature above, there are 918 observations (rows) and 11 common features in addition to 1 output class (columns).

General Information

```
str(mydata)
```

```
## 'data.frame':    918 obs. of  12 variables:
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sex           : chr  "M" "F" "M" "F" ...
##  $ ChestPainType : chr  "ATA" "NAP" "ATA" "ASY" ...
##  $ RestingBP     : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol   : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RestingECG    : chr  "Normal" "Normal" "ST" "Normal" ...
##  $ MaxHR         : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina: chr  "N" "N" "N" "Y" ...
##  $ Oldpeak       : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ ST_Slope      : chr  "Up" "Flat" "Up" "Flat" ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

Below further describe the variables and its factor (if applicable):

- Age: Age of the patient in years
- Sex: Sex of the patient
  - M: Male
  - F: Female
- ChestPainType: Chest pain type
  - TA: Typical Angina
  - ATA: Atypical Angina
  - NAP: Non-Anginal Pain
  - ASY: Asymptomatic
- RestingBP: Resting blood pressure in mm Hg
- Cholesterol: Serum cholesterol in mm/dl
- FastingBS: Fasting blood sugar
  - 1: If FastingBS > 120 mg/dl
  - 0: Otherwise
- RestingECG: Resting electrocardiogram results
  - Normal: Normal
  - ST: Having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
  - LVH: Showing probable or definite left ventricular hypertrophy by Estes' criteria]
- MaxHR: Maximum heart rate achieved (Numeric value between 60 and 202.
- ExerciseAngina: Exercise-induced angina
  - Y: Yes
  - N: No
- Oldpeak: Oldpeak = ST (Numeric value measured in depression)
- ST_Slope: The slope of the peak exercise ST segment
  - Up: Upsloping
  - Flat: Flat
  - Down: Downsloping
- HeartDisease: Output/target class
  - 1: Heart disease
  - 0: Normal

Statistical Information

```
summary(mydata)
```

```
##       Age            Sex            ChestPainType        RestingBP
##  Min.   :28.00   Length:918         Length:918         Min.   :  0.0
##  1st Qu.:47.00   Class :character   Class :character   1st Qu.:120.0
##  Median :54.00   Mode  :character   Mode  :character   Median :130.0
##  Mean   :53.51                                         Mean   :132.4
##  3rd Qu.:60.00                                         3rd Qu.:140.0
##  Max.   :77.00                                         Max.   :200.0
##   Cholesterol      FastingBS       RestingECG           MaxHR
##  Min.   :  0.0   Min.   :0.0000   Length:918         Min.   : 60.0
```

```
##  1st Qu.:173.2   1st Qu.:0.0000   Class :character   1st Qu.:120.0
##  Median :223.0   Median :0.0000   Mode  :character   Median :138.0
##  Mean   :198.8   Mean   :0.2331                      Mean   :136.8
##  3rd Qu.:267.0   3rd Qu.:0.0000                      3rd Qu.:156.0
##  Max.   :603.0   Max.   :1.0000                      Max.   :202.0
##  ExerciseAngina        Oldpeak          ST_Slope          HeartDisease
##  Length:918         Min.   :-2.6000   Length:918         Min.   :0.0000
##  Class :character   1st Qu.: 0.0000   Class :character   1st Qu.:0.0000
##  Mode  :character   Median : 0.6000   Mode  :character   Median :1.0000
##                     Mean   : 0.8874                      Mean   :0.5534
##                     3rd Qu.: 1.5000                      3rd Qu.:1.0000
##                     Max.   : 6.2000                      Max.   :1.0000
```

In relation to the target (Heart Disease), the data are less divergent from the arithmetic mean because the higher the standard deviation value, the more the data deviate from the mean, and the smaller the standard deviation, the less the data deviate from the mean.

Data Preprocessing

Data pre-processing is to prepare and transform raw data for analysis or modeling use, by ensuring data is in a suitable format and quality for further processing.

Data Cleaning

In data cleaning, missing (null) values, duplication of observations and outliers will be investigated.

Missing Values

```
# Finding and cleaning null values
colSums(is.na(mydata))
```

```
##           Age           Sex  ChestPainType      RestingBP    Cholesterol
##             0             0              0              0              0
##     FastingBS     RestingECG          MaxHR ExerciseAngina        Oldpeak
##             0             0              0              0              0
##      ST_Slope   HeartDisease
##             0             0
```

There is no null values in dataset.

Duplicated Rows

```
#  Delete duplicate data
mydata <- unique(mydata)
dim(mydata)
```

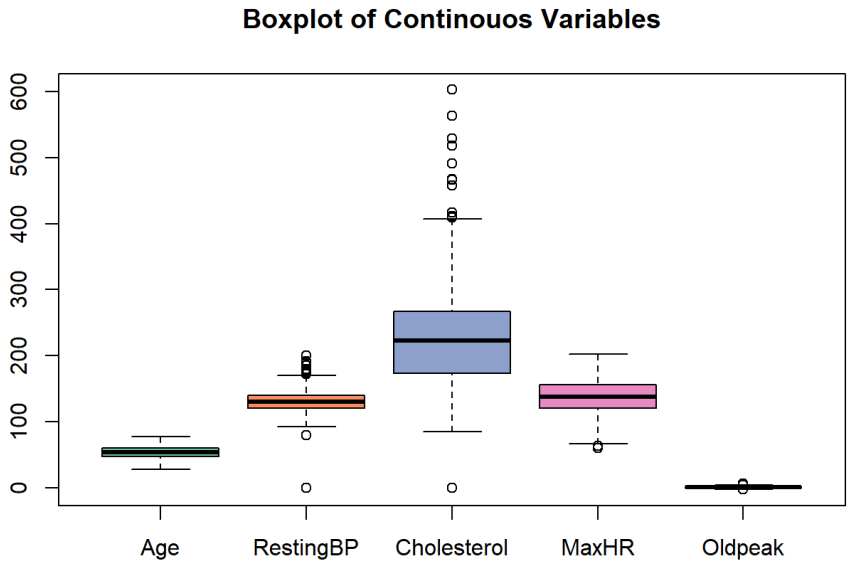
```
## [1] 918  12
```

There is no duplicate data in the dataset.

Outliers

Handling of outliers will only be performed continuous data only, i.e. Age, RestingBP, Cholesterol, MaxHR and Oldpeak. Box plot is employed to identify the potential outliers visually.

```
variables <- c("Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak")
boxplot_data <- mydata[, variables]

# Define the colors using a color palette from RColorBrewer
box_colors <- brewer.pal(length(variables), "Set2")

# Create the boxplot with customized colors
boxplot(boxplot_data, col = box_colors, main = "Boxplot of Continouos Variables")
```



From boxplots, it can be seen that the RestingBP and Cholesterol are having a number of observations that are exceed that maximum range, while for RestingBP, Cholesterol and MaxHR are observed to have outliers that stay at below minimum range.

After inspected the plots, outliers are decided to be removed via interquartile range (IQR) methods, where any values that fall outside of 1.5 times the IQR will be considered as outliers.

```
# IQR methods to remove outliers
# Copy df for modification
mydata_clean <- mydata

# Iterate over continuous data columns only
for (col in c("Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak")) {

  # Calculate Q1, Q3 & IQR
  q1 <- quantile(mydata[[col]], 0.25)
  q3 <- quantile(mydata[[col]], 0.75)
  iqr <- q3 - q1

  # Define the lower bound and upper bound
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  cat(col, ": lower bound is", round(lower_bound, 3), ", upper bound is", round(upper_bound, 3), "\n")

  # Remove outliers by filtering based on lower & upper bounds
  mydata_clean <- mydata_clean[mydata_clean[[col]] >= lower_bound & mydata_clean[[col]] <= upper_bound, ]
}
```

```
## Age : lower bound is 27.5 , upper bound is 79.5
## RestingBP : lower bound is 90 , upper bound is 170
## Cholesterol : lower bound is 32.625 , upper bound is 407.625
## MaxHR : lower bound is 66 , upper bound is 210
## Oldpeak : lower bound is -2.25 , upper bound is 3.75
```

```
# Dimension of dataset after outlier removal
cat("Number of rows after remove outliers:", nrow(mydata_clean), "\n")
```

```
## Number of rows after remove outliers: 702
```

```
# Set back df to original name
mydata <- mydata_clean
```

After removing the outliers for all continuous data columns, the dimension of dataset is updated from 918 rows to 702 rows with 12 variables. **216 rows** of observations has been **categorised as outliers** and removed.

Label Encoding

Label encoding means converting categorical features into numerical values to ease future analysis. Label encoding will only be performed on character-type data only, i.e. Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope.

Sex: M → 1; F → 0

```
# Sex
unique(mydata$Sex)
```

```
## [1] "M" "F"
```

```
mydata$Sex <- ifelse(mydata$Sex == "M",1,0)
```

ChestPainType: ASY → 1; ATA → 2; NAP → 3; TA → 4

```
# ChestPainType
unique(mydata$ChestPainType)
```

```
## [1] "ATA" "NAP" "ASY" "TA"
```

```
mydata$ChestPainType <- as.numeric(factor(mydata$ChestPainType,
                                          levels = c('ASY','ATA','NAP','TA' ),
                                          labels = c(1,2,3,4)))
```

RestingECG: Normal → 1; ST → 2; LVH → 3

```
# RestingECG
unique(mydata$RestingECG)
```

```
## [1] "Normal" "ST"     "LVH"
```

```
mydata$RestingECG <- as.numeric(factor(mydata$RestingECG,
                                       levels = c('Normal','ST','LVH'),
                                       labels = c(1,2,3)))
```

ExerciseAngina: Y → 1; N → 2

```
# ExerciseAngina
unique(mydata$ExerciseAngina)
```

```
## [1] "N" "Y"
```

```
mydata$ExerciseAngina <- ifelse(mydata$ExerciseAngina == "Y",1,0)
```

ST_Slope: Up → 1; Flat → 2; Down → 3

```
# ST_Slope
unique(mydata$ST_Slope)
```

```
## [1] "Up"   "Flat" "Down"
```

```
mydata$ST_Slope <- as.numeric(factor(mydata$ST_Slope,
                                     levels = c('Up','Flat','Down'),
                                     labels = c(1,2,3)))
```

Finally to confirm data structure & datatype after encode.

```
str(mydata)
```

```
## 'data.frame':    702 obs. of  12 variables:
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sex           : num  1 0 1 0 1 1 0 1 1 0 ...
##  $ ChestPainType : num  2 3 2 1 3 3 2 2 1 2 ...
##  $ RestingBP     : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol   : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RestingECG    : num  1 1 2 1 1 1 1 1 1 1 ...
##  $ MaxHR         : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina: num  0 0 0 1 0 0 0 0 1 0 ...
##  $ Oldpeak       : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ ST_Slope      : num  1 2 1 2 1 1 1 1 2 1 ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

All character data is converted to numeric/integer data type.

Feature Selection

Feature selection is a techniques to select the best set of features (subset of relevant features) for use in construction of optimized models.

```
feature_select <- Boruta(HeartDisease ~ ., data = mydata)
feature_select$finalDecision
```

```
##           Age           Sex  ChestPainType      RestingBP    Cholesterol
##     Confirmed     Confirmed     Confirmed     Confirmed       Rejected
##     FastingBS    RestingECG         MaxHR ExerciseAngina        Oldpeak
##     Confirmed     Confirmed     Confirmed     Confirmed     Confirmed
##      ST_Slope
##     Confirmed
## Levels: Tentative Confirmed Rejected
```

According to the Boruta algorithm's output, the Cholesterol has been rejected. However, it is worth noting that high cholesterol levels have been scientifically linked to the development of fatty deposits in blood vessels, leading to reduced blood flow and an increased risk of heart disease. Considering this significant association, it is believed that Cholesterol may still hold valuable insights. Therefore, this variable will be retained for further analysis, i.e. to investigate possibilities of transforming the variable or combining it with other variables to create more informative features.

For all other variables in the dataset, the result shows that they are in *Confirmed* status, indicating that the features has been confirmed important by the Boruta algorithm and significantly contributes to predict the ouput (HeartDisease) variable.

Data Splitting

Data splitting is fundamental to ensure model effectiveness, prevent over-fitting, and evaluate performance on unseen data.

A 70-30 split is aimed to strike a balance between providing enough data for effective model training and reserving a separate portion for unbiased model evaluation.

```
set.seed(123)
split <- sample.split(mydata$HeartDisease, SplitRatio = 0.7)
train_set <- subset(mydata, split == TRUE)
test_set <- subset(mydata, split == FALSE)
```

- **train_set** consist of 70% data randomly picked from original data, where it is used to train machine learning models by learning patterns and relationships in the data to make predictions or classifications.
- **test_set** consist of 30% data randomly picked from original data, where it is used to evaluate the performance of the trained model and serves as a benchmark to measure how well the model generalizes to new, unseen data.

Confirming the data is splited successfully by inspect the train/test set dimension:

```
dim(train_set)
```

```
## [1] 491  12
```

491 observations (70%)

```
dim(test_set)
```

```
## [1] 211  12
```

211 observations (30%)

Data Transformation

Data transformation involves modifying the original data to make it more suitable for further exploration, modeling, or visualization. The main goal of data transformation is to improve the quality, structure and usefulness of the data.

Data transformation is conducted on *train_set* and *test_set* separately. In this case, scaling and normalization is performed, by scaling all the continuous numerical variable into a common scale to help in comparing variables with different units and ranges. Formula employed: $x = (x - mean)/stdev$ ��� = (��� − ������������) / ��������������.

Continuous data variable include Age, RestingBP, Cholesterol, MaxHR and Oldpeak.

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

Finally to confirm data has been successfully transformed by inspected the 1st few lines of each train and test set:

```
head(train_set)
```

```
##            Age Sex ChestPainType  RestingBP Cholesterol FastingBS RestingECG
## 1  -1.42227348   1             2  0.5634569   0.9354451         0          1
## 2  -0.44527573   0             3  1.8903505  -1.2056928         0          1
## 4  -0.55383104   0             1  0.4307675  -0.5378149         0          1
## 5   0.09750079   1             3  1.2269037  -0.9110408         0          1
## 8   0.09750079   1             2 -1.4268836  -0.6556757         0          1
## 10 -0.55383104   0             2 -0.7634368   0.8372278         0          1
##         MaxHR ExerciseAngina    Oldpeak ST_Slope HeartDisease
## 1   1.3594775              0 -0.8622414        1            0
```

```
## 2   0.6802575             0  0.1678329        2           1
## 4  -1.3574025             1  0.6828700        2           1
## 5  -0.7630850             0 -0.8622414        1           0
## 8   0.0859400             0 -0.8622414        1           0
## 10 -0.8479875             0 -0.8622414        1           0
```

```
head(test_set)
```

```
##           Age Sex ChestPainType  RestingBP Cholesterol FastingBS RestingECG
## 3  -1.4523953   1             2 -0.1030373  0.94228695         0          2
## 6  -1.2566396   1             3 -0.7189257  2.06083856         0          1
## 7  -0.6693728   0             2 -0.1030373  0.02347671         0          1
## 9  -1.4523953   1             1  0.5128512 -0.57574736         0          1
## 11 -1.4523953   0             3 -0.1030373 -0.49585082         0          1
## 14 -0.2778616   1             1  0.5128512 -0.03644570         0          1
##           MaxHR ExerciseAngina    Oldpeak ST_Slope HeartDisease
## 3  -1.678434537              0 -0.8728218        1            0
## 6   1.075213140              0 -0.8728218        1            0
## 7   1.075213140              0 -0.8728218        1            0
## 9  -0.454591125              1  0.7304754        2            1
## 11  0.004350154              0 -0.8728218        1            0
## 14 -0.072140059              1  0.1960430        2            1
```

Data Exploration

Data exploration is essential to gain a better understanding of the dataset. It involves examining and visualizing the data to identify patterns, trends, relationships, and anomalies with the aim of extracting useful insights and inform further analysis or decision-making.

train_set data will be used to conduct EDA.

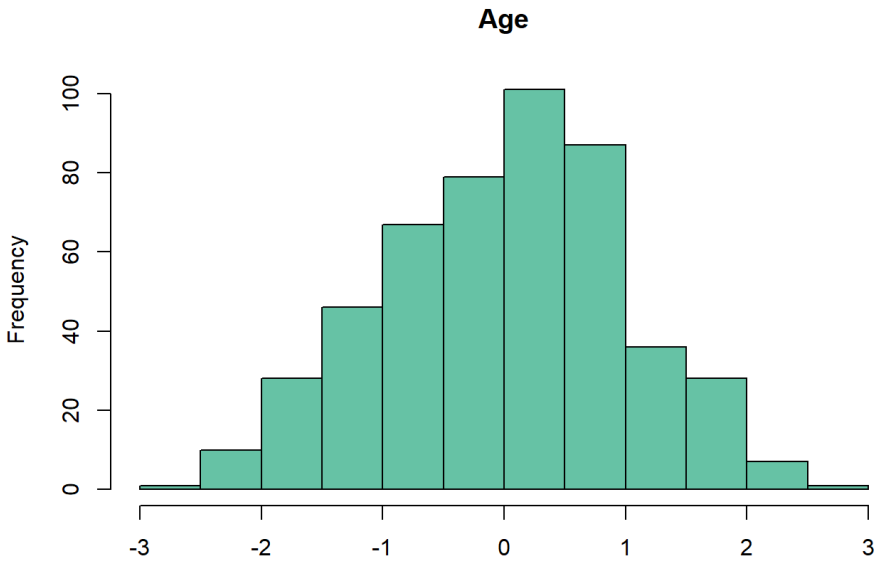
Univariate Analysis

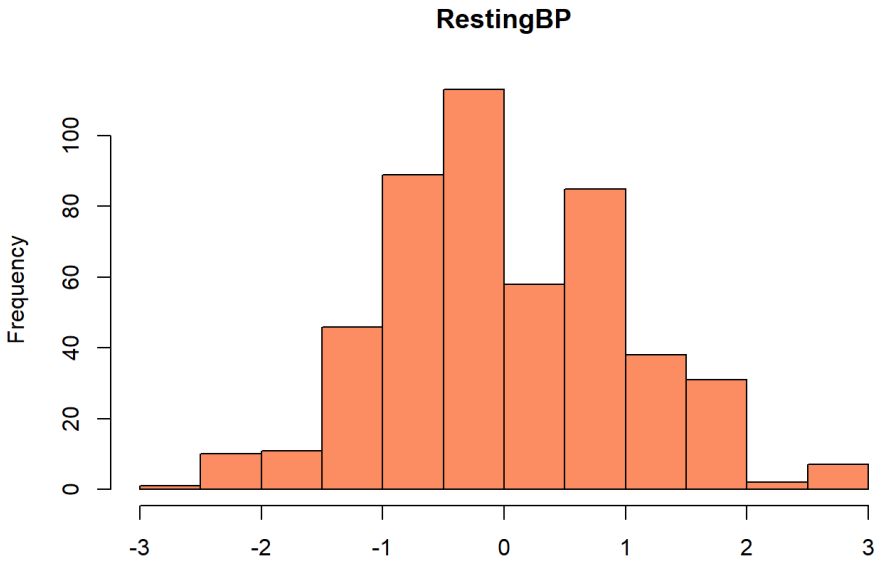
Analysis carried out by considering one variable at a time.

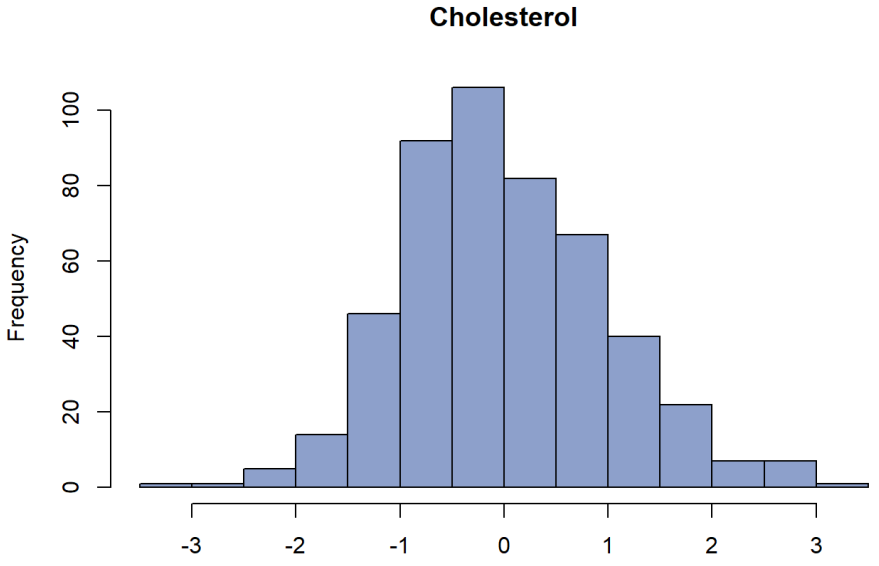
Histogram

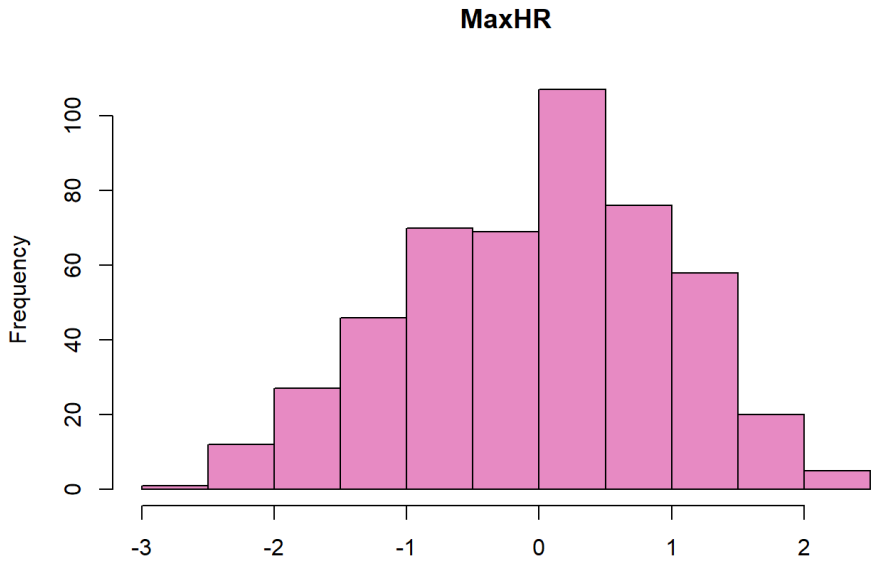
All continuous data is analysed with histogram to observe the overall distribution.

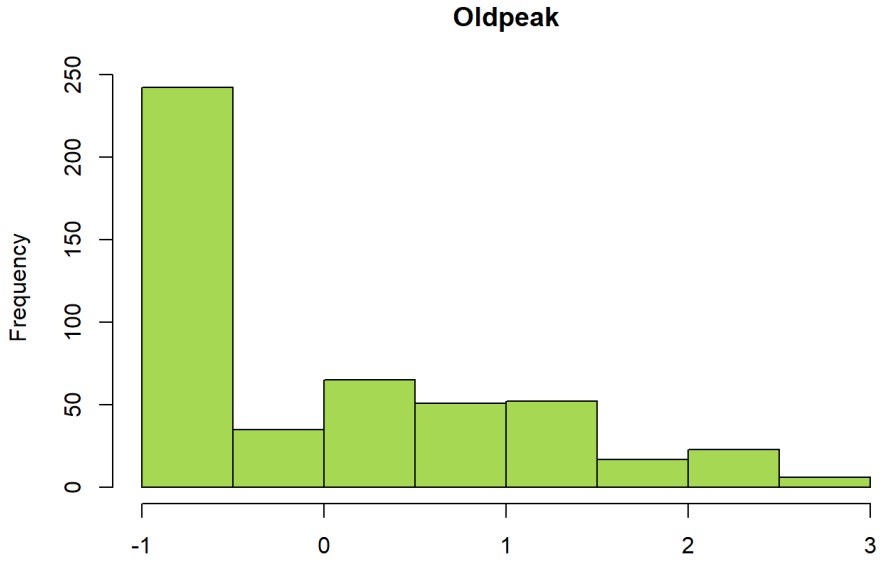
```
variables <- c("Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak")

# Define the colors using a color palette from RColorBrewer
hist_colors <- brewer.pal(length(variables), "Set2")

for (i in seq_along(variables)) {
  col <- variables[i]
  if (is.numeric(train_set[[col]])) {
    hist(train_set[[col]], main = col, xlab = "", col = hist_colors[i])
  }
}
```


Age
Frequency
0
20
40
60
80
100
-3
-2
-1
0
1
2
3


RestingBP
Frequency
0
20
40
60
80
100
-3
-2
-1
0
1
2
3


Cholesterol
Frequency
0
20
40
60
80
100
-3
-2
-1
0
1
2
3





The histogram of Age, RestingBP, Cholesterol and MaxHR are observed to be normally distributed, no outlier, skewed data or any abnormality.

For Oldpeak, data seems skewed to the left. Further inspect the raw (before transformed) data, there are around 43% of observation has 0 as the Oldpeak value, suggests that there is no ST segment depression observed in the electrocardiogram (ECG) reading *(Oldpeak = ST (Numeric value measured in depression))*.

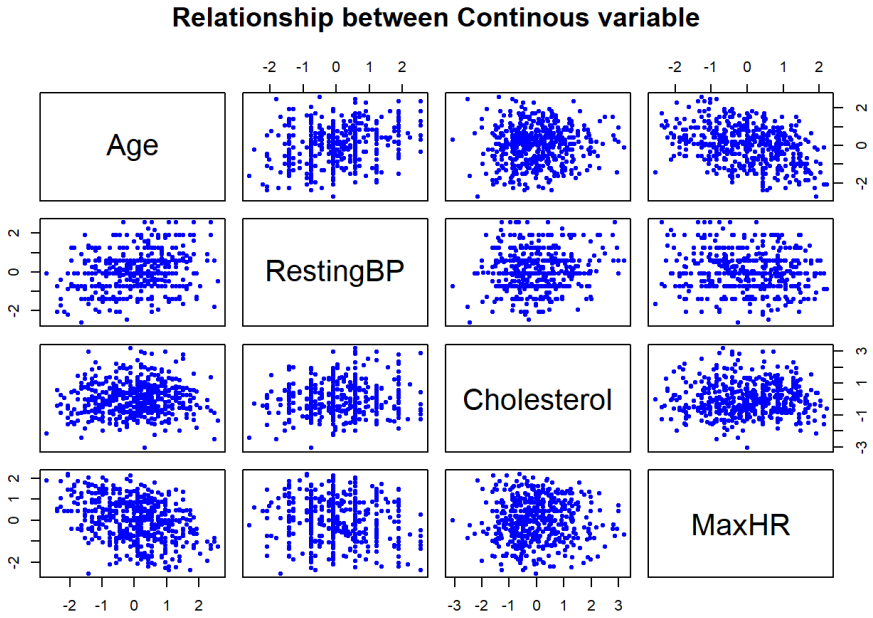
Bivariate Analysis

The relationships between pairs of variables are explored.

Correlation & Heatmap

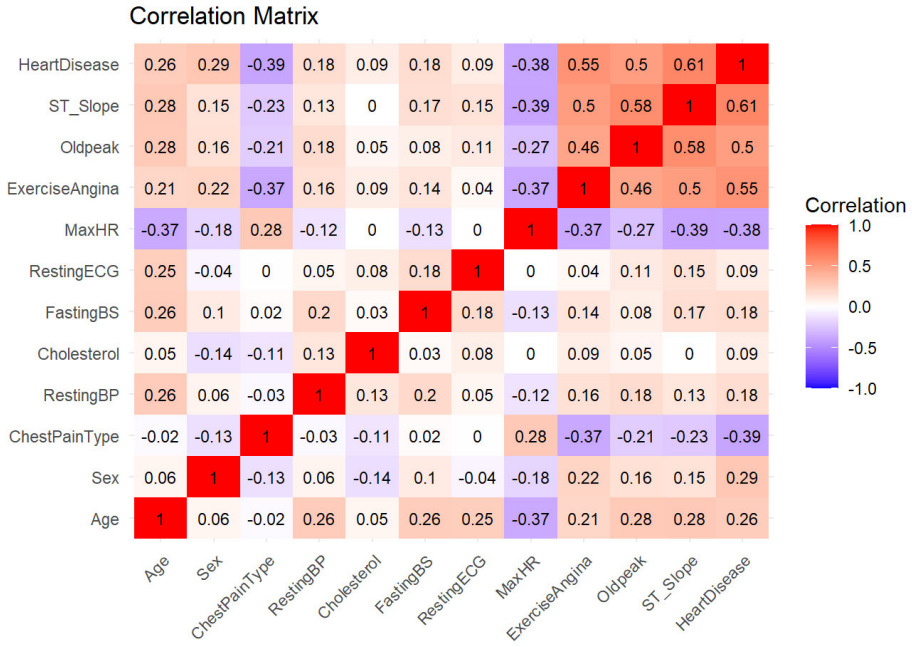
The relationship between 2 continuous variables (normally distributed) are investigated.

```
# to see relationship between 2 continuous variables
pairs(train_set[c("Age", "RestingBP", "Cholesterol", "MaxHR")], main = "Relationship between Continous variable",
      col = "blue", pch = 20)
```



The relationship between each variables are investigated through the correlation heatmap as below.

```
#create correlation heatmap
corr_matrix <- cor(train_set)

# Convert the correlation matrix to a data frame
corr_df <- melt(corr_matrix)

ggplot(data = corr_df, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(Var2, Var1, label = round(value, 2)), size = 3) +
  scale_fill_gradient2(low = "blue", high = "red",
                       limit = c(-1,1), name="Correlation") +
  theme_minimal() +
  labs(x = "", y = "", title = "Correlation Matrix") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels
```



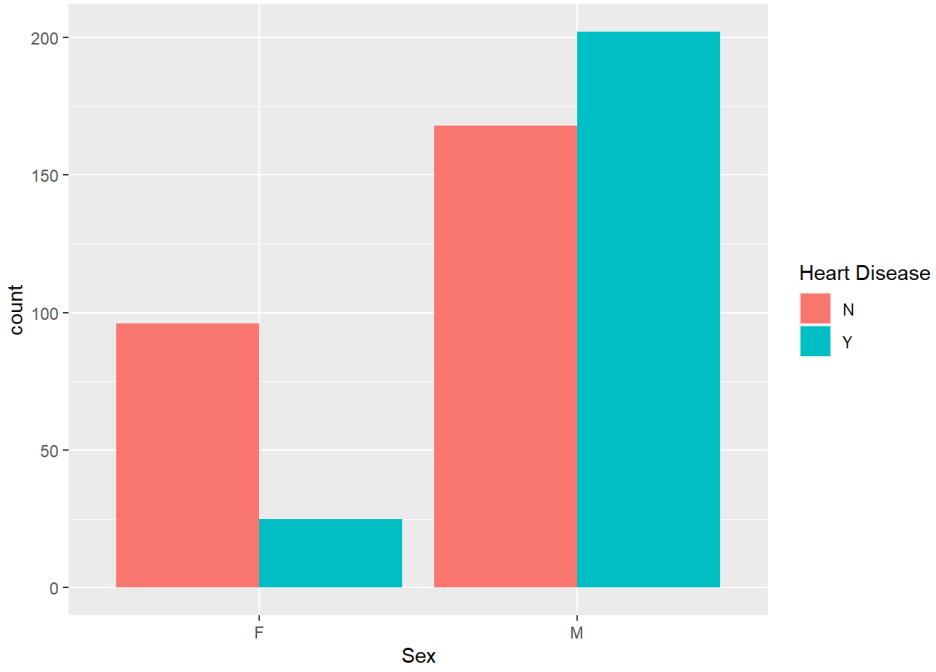
From the correlation plot and heatmap above, it can be seen that ST_Slope and Heart Disease, the target variable having the strongest positive correlation among all the other variables. And for negative correlation, the strongest relationship is held between Chest Pain Type vs. Heart Disease and ST_slope vs. MaxHR.

However, with a correlation coefficient of 0.61 and -0.39, the variables are still classified under moderate correlation in general. Thus, it is consider safe to proceed analysis with this dataset. No deletion of column is necessary (up until this stage).

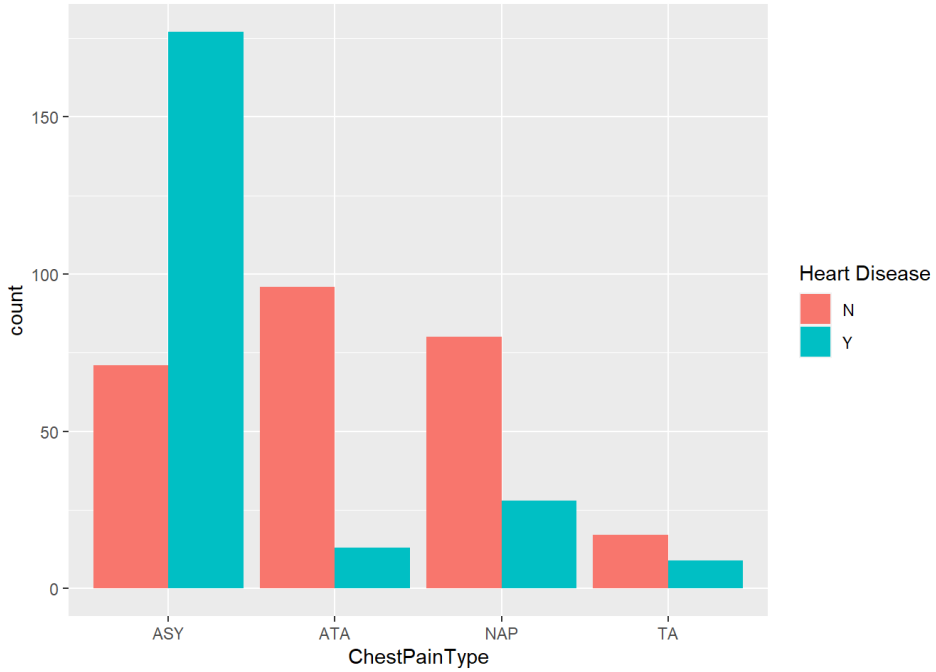
Bar Charts

Bar charts for categorical variable (Sex, ChestPainType, ExerciseAngina, RestingECG, ST_Slope, FastingBS) are generated to see the distribution of target variable, *heart disease*, across each category within the independent variable.

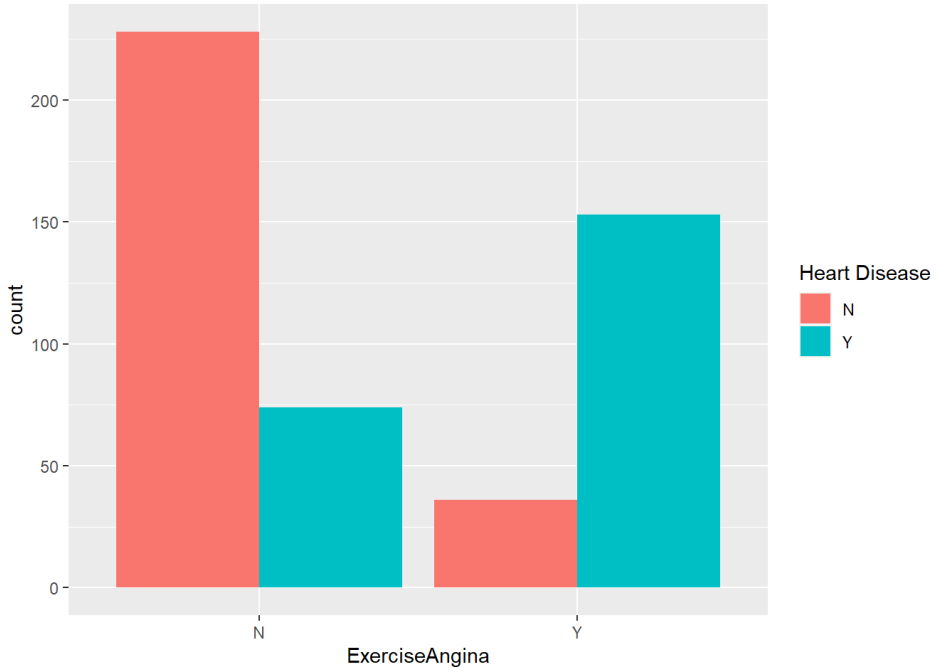
```
# Sex with heartdisease
ggplot(train_set, aes(x = factor(Sex, labels = c("F", "M")))) +
  geom_bar(aes(fill = factor(HeartDisease, labels = c("N", "Y"))), position = "dodge") +
  labs(x = "Sex", fill = "Heart Disease")
```



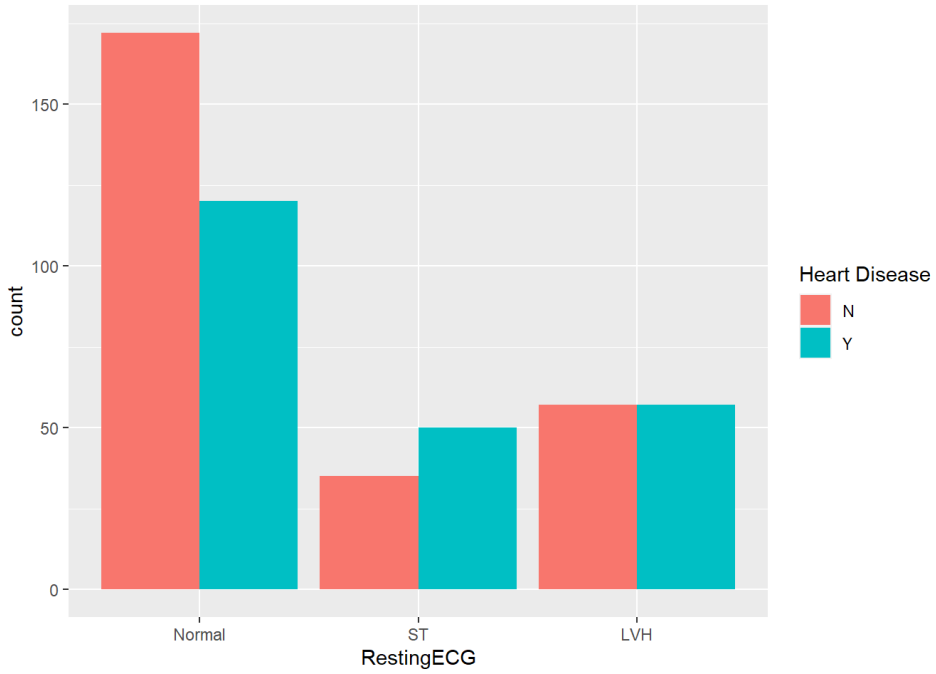

```
# ChestPainType with heartdisease
ggplot(train_set, aes(x = factor(ChestPainType, labels = c("ASY", "ATA", "NAP", "TA")))) +
  geom_bar(aes(fill = factor(HeartDisease, labels = c("N", "Y"))), position = "dodge") +
  labs(x = "ChestPainType", fill = "Heart Disease")
```



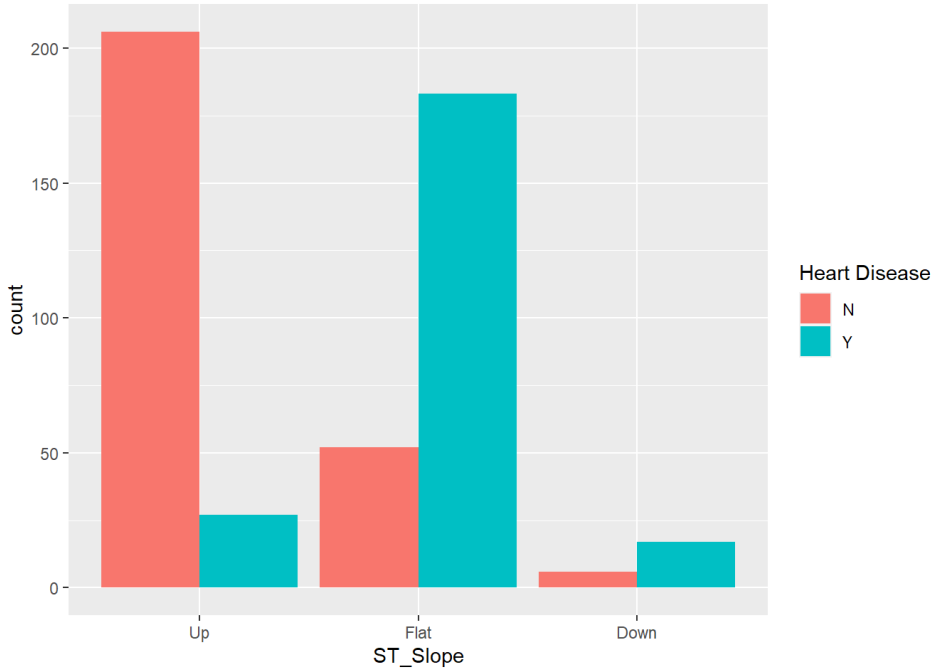

```
# ExerciseAngina with heartdisease
ggplot(train_set, aes(x = factor(ExerciseAngina, labels = c("N", "Y")))) +
  geom_bar(aes(fill = factor(HeartDisease, labels = c("N", "Y"))), position = "dodge") +
  labs(x = "ExerciseAngina", fill = "Heart Disease")
```



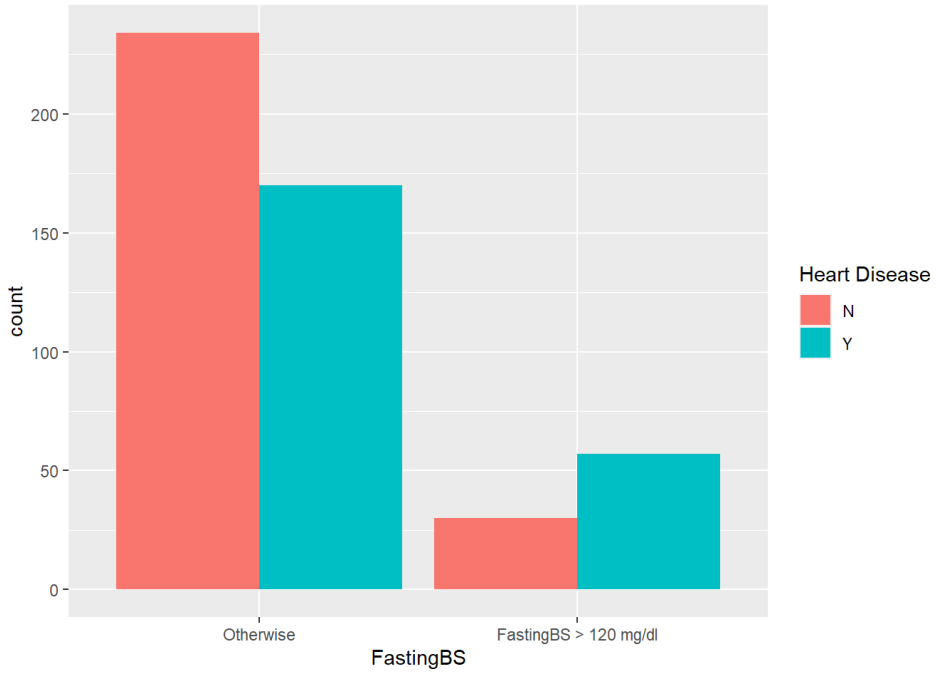

```
# RestingECG with heartdisease
ggplot(train_set, aes(x = factor(RestingECG, labels = c("Normal", "ST", "LVH")))) +
  geom_bar(aes(fill = factor(HeartDisease, labels = c("N", "Y"))), position = "dodge") +
  labs(x = "RestingECG", fill = "Heart Disease")
```




```
# ST_Slope with heartdisease
ggplot(train_set, aes(x = factor(ST_Slope, labels = c("Up", "Flat", "Down")))) +
  geom_bar(aes(fill = factor(HeartDisease, labels = c("N", "Y"))), position = "dodge") +
  labs(x = "ST_Slope", fill = "Heart Disease")
```




```
# FastingBS with heartdisease
ggplot(train_set, aes(x = factor(FastingBS, labels = c("Otherwise", "FastingBS > 120 mg/dl")))) +
  geom_bar(aes(fill = factor(HeartDisease, labels = c("N", "Y"))), position = "dodge") +
  labs(x = "FastingBS", fill = "Heart Disease")
```



- Most of the heart disease individuals is male.
- Most of the heart disease individuals are with Asymptomatic Chest Pain Type.
- Most of the non-heart disease individuals without exercise-induced angina.
- Individuals who have ST-T wave abnormality on resting electrocardiogram results have the highest percentage in the train dataset.
- Most of the heart disease individuals with flat ST slope while most of the non-heart disease individuals with up ST slope.
- For individuals with fasting blood sugar > 120mg/dl, most of the individuals with heart disease.

Multivariate Analysis

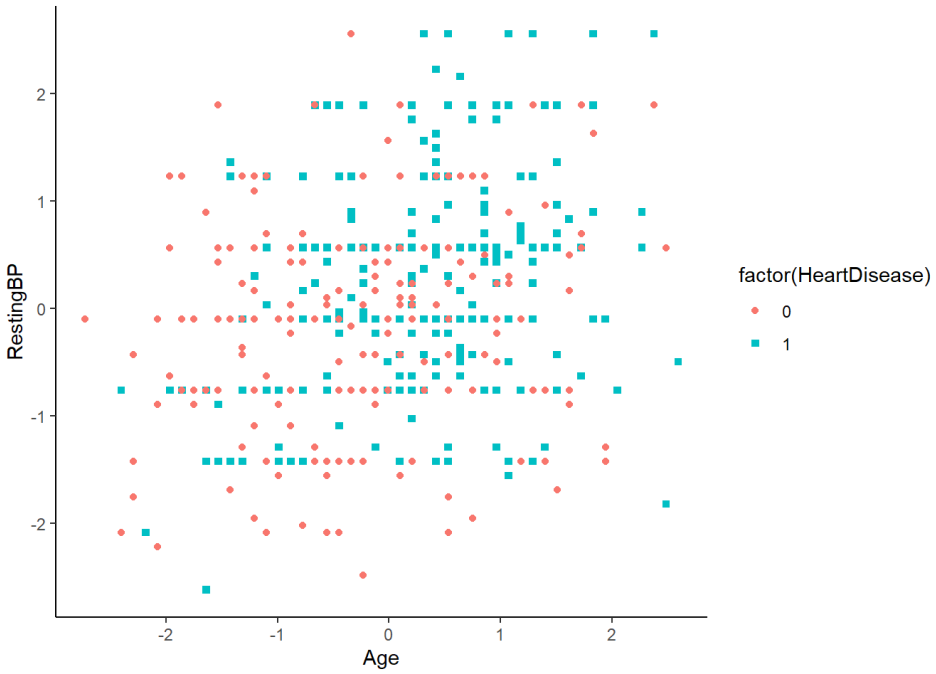
The relationships between multiple variables are studied simultaneously.

Scatterplots

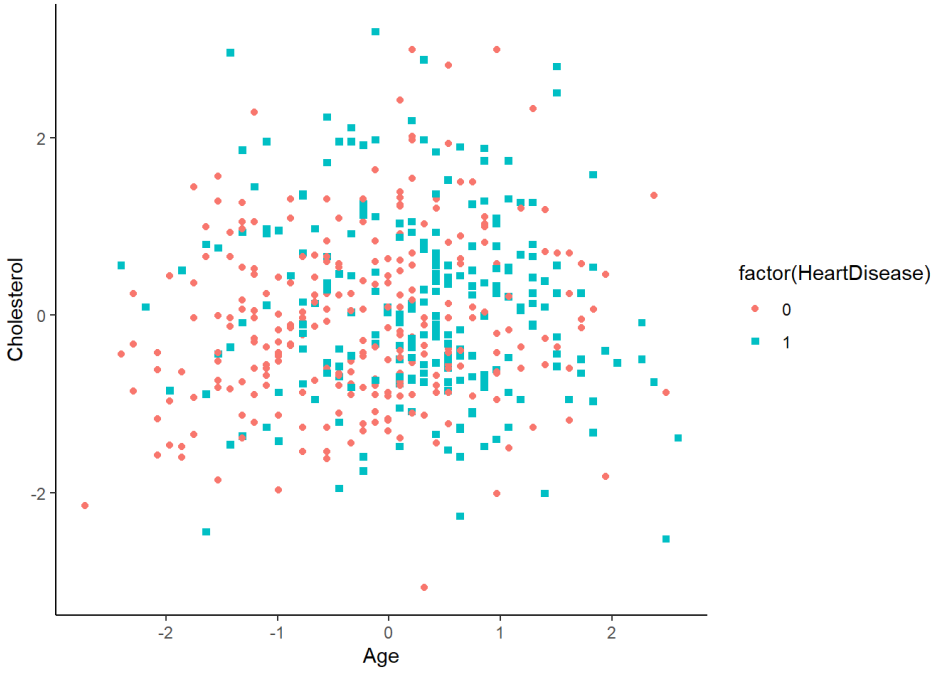
Scatterplots for continuous variable (Age, RestingBP, Cholesterol, MaxHR) are generated to see the relationships between them with the consideration of heart disease, the target variable as factor.

```
# Age & RestingBP with heartdisease
ggplot(train_set, aes(x = Age, y = RestingBP, shape = factor(HeartDisease))) +
```

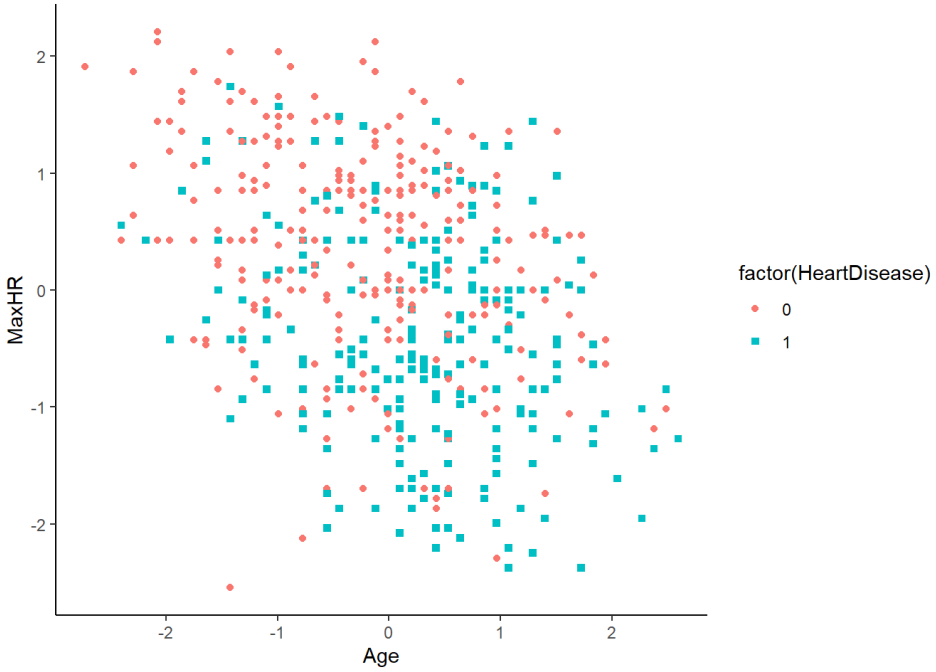
```
  geom_point(aes(color = factor(HeartDisease))) +
  labs(x = "Age", y = "RestingBP") +
  scale_shape_manual(values = c(19, 15)) +
  theme_classic()
```



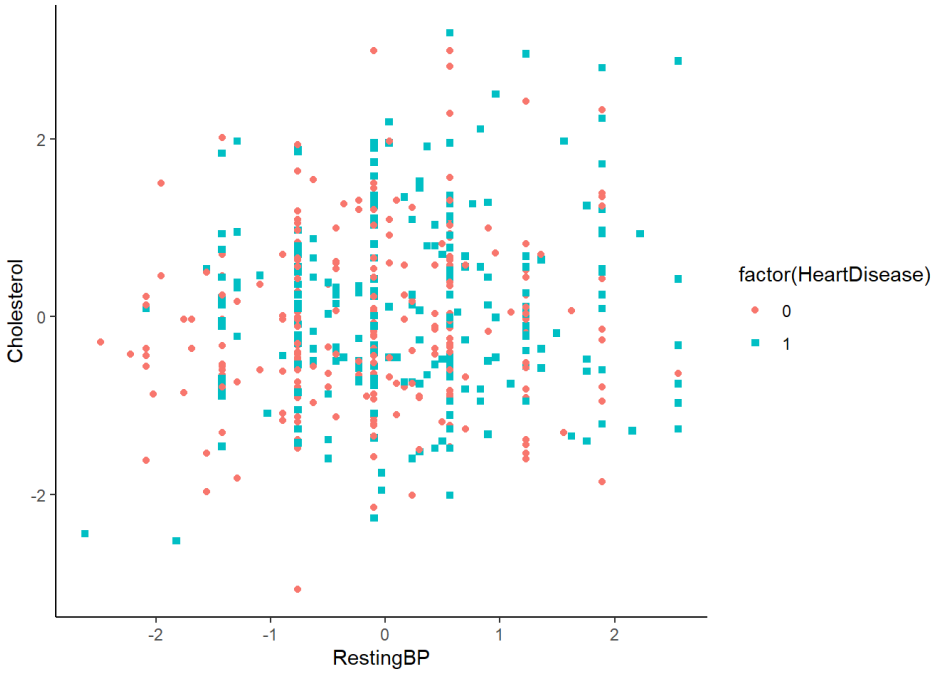

```
# Age & Cholesterol with heartdisease
ggplot(train_set, aes(x = Age, y = Cholesterol, shape = factor(HeartDisease))) +
  geom_point(aes(color = factor(HeartDisease))) +
  labs(x = "Age", y = "Cholesterol") +
  scale_shape_manual(values = c(19, 15)) +
  theme_classic()
```



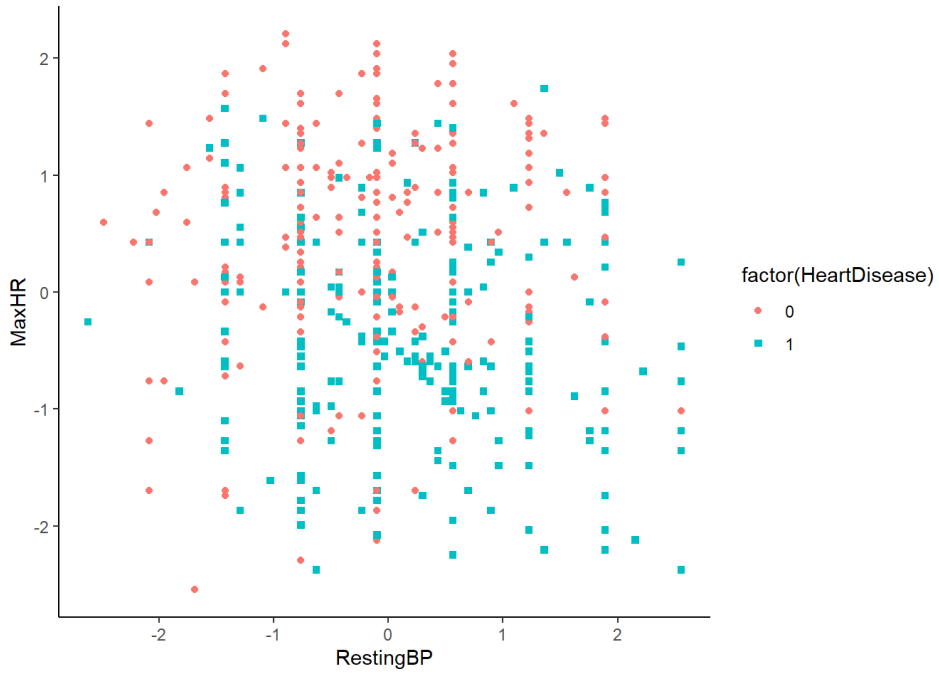

```
# Age & MaxHR with heartdisease
ggplot(train_set, aes(x = Age, y = MaxHR, shape = factor(HeartDisease))) +
  geom_point(aes(color = factor(HeartDisease))) +
  labs(x = "Age", y = "MaxHR") +
  scale_shape_manual(values = c(19, 15)) +
  theme_classic()
```



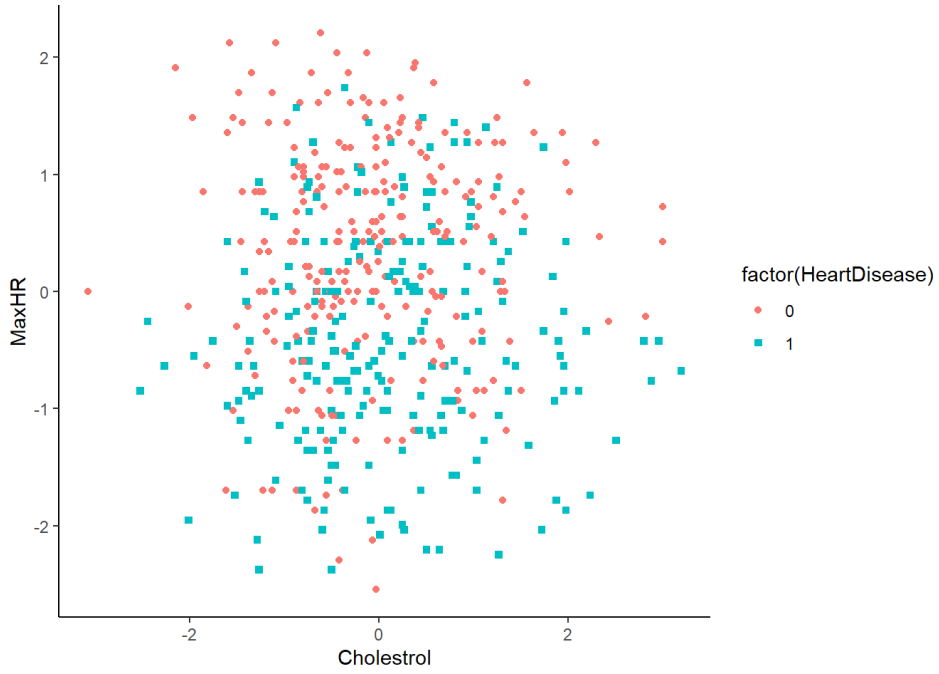

```
# RestingBP & Cholesterol with heartdisease
ggplot(train_set, aes(x = RestingBP, y = Cholesterol, shape = factor(HeartDisease))) +
  geom_point(aes(color = factor(HeartDisease))) +
  labs(x = "RestingBP", y = "Cholesterol") +
  scale_shape_manual(values = c(19, 15)) +
  theme_classic()
```




```
# RestingBP & MaxHR with heartdisease
ggplot(train_set, aes(x = RestingBP, y = MaxHR, shape = factor(HeartDisease))) +
  geom_point(aes(color = factor(HeartDisease))) +
  labs(x = "RestingBP", y = "MaxHR") +
  scale_shape_manual(values = c(19, 15)) +
  theme_classic()
```




```
# Cholesterol & MaxHR with heartdisease
ggplot(train_set, aes(x = Cholesterol, y = MaxHR, shape = factor(HeartDisease))) +
  geom_point(aes(color = factor(HeartDisease))) +
  labs(x = "Cholestrol", y = "MaxHR") +
  scale_shape_manual(values = c(19, 15)) +
  theme_classic()
```



- Age & RestingBP with heartdisease
  - The scatterplot shows that as age and resting blood pressure increases, there is an increase in the proportion of individuals with heart disease compared to those without heart disease. This suggests a positive correlation between resting blood pressure and heart disease.
- Age & Cholesterol with heartdisease
  - The scatterplot shows that as age and cholesterol level increases, there is a slight increase in the proportion of individuals with heart disease compared to those without heart disease. This suggests a positive correlation between an increase in age, cholesterol with the presence of heart disease.
- Age & MaxHR with heartdisease
  - The scatterplot shows that when the maximum heart rate decreases with an increase in age, there is an increase in the proportion of individuals with heart disease. This suggests a negative correlation between maximum heart rate and heart disease.
- RestingBP & Cholesterol with heartdisease
  - The scatterplot shows that as cholesterol levels and resting blood pressure increases, there is an increase in the proportion of individuals with heart disease. This suggests a positive correlation between cholesterol levels, resting blood pleasure and heart disease.
- RestingBP & MaxHR with heartdisease
  - The scatterplot shows the relationship between resting blood pressure and maximum heart rate, with individuals with no heart disease (red dots) and those with heart disease (blue dots) represented separately. The scatterplot suggests that there is no clear pattern or correlation between

resting blood pressure and maximum heart rate for both groups.

- Cholesterol & MaxHR with heartdisease
  - The scatterplot visually represents the correlation between maximum heart rate and cholesterol levels, with two distinct groups: individuals without heart disease (represented by red dots) and those with heart disease (represented by blue dots). The plot clearly demonstrates that as cholesterol levels increase, maximum heart rate decreases, indicating a heightened risk of heart disease.

Data Modelling

Data modeling refers to the process of developing a predictive model based on available data and problem requirements. Appropriate modeling algorithms are selected to train the model, with the aim of producing a reliable and accurate model that capable of generating valuable insights.

Modeling algorithms used in this project includes:

1. Logistic Regression
2. Decision Tree
3. Gradient Boosting Classifier
4. K-Nearest Neighbours (KNN)
5. Support Vector Machines (SVM)
6. Random Forest

Logistic Regression

A common machine learning algorithm used to model binary or multi-class classification problems. It is employed when the outcome variable only has two possible values, such as “yes” or “no”. A logistic function, which converts the input variables into a probability between 0 and 1, is used in logistic regression to model the relationship between a dependent binary variable and one or more independent variables.

Table below lists the advantages and disadvantages of Logistic Regression:

Advantages	Disadvantages
• Easy to understand and interpretable. • Performs well for large datasets, especially when the relationship between features and target variable is approximately linear. • Works well with linearly separable data.	• Assumes linear relationship between features and the log-odds of the target variable. • Sensitive to outliers and multicollinearity. • May not perform well with nonlinear or highly imbalanced datasets.

```
# Fit logistic regression model
log_model <- glm(HeartDisease ~ ., data = train_set, family = binomial())

# Make predictions on test set
log_pred <- predict(log_model, newdata = test_set, type = "response")

# Create confusion matrix
log_cm <- table(test_set$HeartDisease, log_pred > 0.5)

# Calculate accuracy
log_accuracy <- sum(diag(log_cm))/sum(log_cm)

# Compute other evaluation metrics
log_precision <- log_cm[2,2] / sum(log_cm[,2])
log_recall <- log_cm[2,2] / sum(log_cm[2,])
log_f1_score <- 2 * log_precision * log_recall / (log_precision + log_recall)

# Print results
# cat("Logistic Regression Model Results:\n")
# cat("Confusion Matrix:\n")
print(log_cm)
```

```
##
##     FALSE TRUE
##   0   102   11
##   1    20   78
```

```
cat(paste0("Accuracy: ", round(log_accuracy, 4), "\n"))
```

```
## Accuracy: 0.8531
```

```
cat(paste0("Precision: ", round(log_precision, 4), "\n"))
```

```
## Precision: 0.8764
```

```
cat(paste0("Recall: ", round(log_recall, 4), "\n"))
```

```
## Recall: 0.7959
```

```
cat(paste0("F1 Score: ", round(log_f1_score, 4), "\n"))
```

```
## F1 Score: 0.8342
```

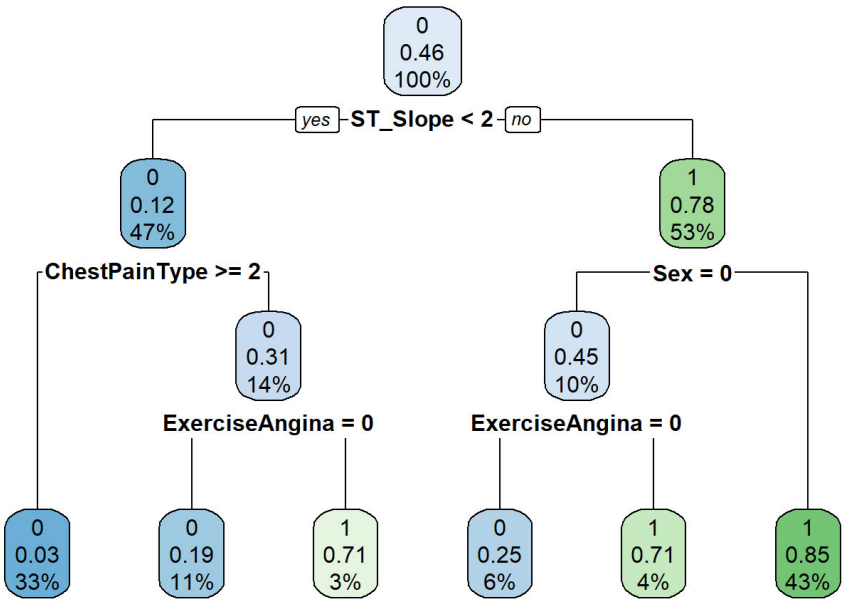
Decision Tree

In decision tree modeling, the data is divided into subsets based on different feature values, creating a tree-like structure of decisions. At each node of the tree, a feature is selected based on its ability to split the data in a way that maximizes the separation of classes or minimizes the variance of the target variable. This process is repeated recursively until a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples in each leaf node.

Table below lists the advantages and disadvantages of Decision Tree:

Advantages	Disadvantages
• Easy to understand and interpretable. • Does not require extensive data preprocessing. • Robust to outliers and can handle missing values.	• Prone to overfitting. • Sensitive to small changes in the data. • May not perform well on unseen data due to limited generalization ability.

```
# Create & visualize decision tree model
model_DT <- rpart(HeartDisease ~., data=train_set, method="class")
rpart.plot(model_DT)
```




```
# Model Evaluation
# Compare model prediction outcome to result from test set
predict_DT <- predict(model_DT, test_set, type="class")

# Confusion Matrix
cm_DT <- table(test_set$HeartDisease, predict_DT)

# Calculate evaluation metrics
TN_DT <- cm_DT[1, 1]  # True Negative (correctly predict people with no heart disease (HD))
FP_DT <- cm_DT[1, 2]  # False Positive (incorrectly classify people with HD under no)
FN_DT <- cm_DT[2, 1]  # False Negative (incorrectly classify people with no HD as yes)
TP_DT <- cm_DT[2, 2]  # True Positive (correctly predict people with HD)

# Measure model performance
accuracy_DT <- (TP_DT + TN_DT) / sum(cm_DT) # Accuracy
```

```
precision_DT <- TP_DT / (TP_DT + FP_DT) # Precision
recall_DT <- TP_DT / (TP_DT + FN_DT)  # Recall(Sensitivity)
f1_DT <- 2 * (precision_DT * recall_DT) / (precision_DT + recall_DT)  # F1 Score

# Print result
print(cm_DT)
```

```
##    predict_DT
##       0  1
##   0 94 19
##   1 13 85
```

```
print(paste('Accuracy: ', round(accuracy_DT, 4)))
```

```
## [1] "Accuracy:  0.8483"
```

```
print(paste('Precision: ', round(precision_DT, 4)))
```

```
## [1] "Precision:  0.8173"
```

```
print(paste('Recall: ', round(recall_DT, 4)))
```

```
## [1] "Recall:  0.8673"
```

```
print(paste('F1 Score: ', round(f1_DT, 4)))
```

```
## [1] "F1 Score:  0.8416"
```

Gradient Boosting Classifier

The Gradient Boosting Classifier is an algorithm used for classification tasks in machine learning. It manages to generate a robust predictive model by combining several weaker models, usually decision trees. It does so by continuously enhancing the inferiority of previous models. To update the model and minimize errors, it employs gradient descent optimization. The ultimate outcome is obtained by gathering all the model predictions. The algorithm is well-known for its capability to generate precise prediction outcomes for convoluted data sets.

Table below lists the advantages and disadvantages of Gradient Boosting Classifier:

Advantages	Disadvantages
• Able to handles mixed data types and missing values. • Reduces bias and variance by combining weak learners through boosting. • Can capture complex interactions between features.	• Prone to overfitting. • Relatively difficult to interpret. • Hyperparameter tuning may be required for optimal performance.

```
#Copy train and test dataset as new dataset to avoid overwrite
train_set2 <- train_set
test_set2 <- test_set

# Convert the Species column to a factor variable
train_set2$HeartDisease <- as.factor(train_set2$HeartDisease)
test_set2$HeartDisease <- as.factor(test_set2$HeartDisease)

# Combine the factor levels from both datasets
levels(test_set2$HeartDisease) <- levels(train_set2$HeartDisease)

# fit GBM model on training set
gbm_model <- train(HeartDisease ~ ., data = train_set2, method = "gbm", verbose = FALSE)

# make predictions on testing set
gbm_predicted <- predict(gbm_model, newdata = test_set2)

# create confusion matrix and calculate performance metrics
gbm_confusion <- confusionMatrix(gbm_predicted, test_set2$HeartDisease)
gbm_accuracy <- gbm_confusion$overall['Accuracy']
gbm_precision <- gbm_confusion$byClass['Pos Pred Value']
gbm_recall <- gbm_confusion$byClass['Sensitivity']
gm_f1_score <- gbm_confusion$byClass["F1"]
```

```
# Print result
gbm_confusion$table
```

```
##           Reference
## Prediction  0  1
##          0 99 19
##          1 14 79
```

```
print(paste('Accuracy: ', round(gbm_accuracy, 4)))
```

```
## [1] "Accuracy:  0.8436"
```

```
print(paste('Precision: ', round(gbm_precision, 4)))
```

```
## [1] "Precision:  0.839"
```

```
print(paste('Recall: ', round(gbm_recall, 4)))
```

```
## [1] "Recall:  0.8761"
```

```
print(paste('F1 Score: ', round(gm_f1_score, 4)))
```

```
## [1] "F1 Score:  0.8571"
```

K-Nearest Neighbours (KNN)

K-Nearest Neighbors (KNN) is a machine learning method employed for classification and regression tasks. It determines the classification or value of a new data point by looking at how close it is to the K nearest labeled examples in the training set. Unlike algorithms that make assumptions about the data distribution, KNN does not make any such assumptions. Instead, it computes the distances between data points and identifies the K nearest neighbors to make predictions. In classification, the predicted class is determined based on the majority vote of the neighbors.

Table below lists the advantages and disadvantages of KNN:

Advantages	Disadvantages
• Simple and easy to understand. • No training phase required. • Can handle multi-class classification problems.	• Sensitive to the choice of distance metric and the K value. • Requires careful preprocessing and scaling of features. • Performs poorly when the feature space is high-dimensional.

```
#Copy train and test dataset as new dataset to avoid overwrite
train_set3 <- train_set
test_set3 <- test_set

# Combine the factor levels from both datasets
levels(test_set2$HeartDisease) <- levels(train_set2$HeartDisease)

train_set3$HeartDisease <- factor(train_set3$HeartDisease, levels = c(0, 1))
test_set3$HeartDisease <- factor(test_set3$HeartDisease, levels = c(0, 1))

# Specify the number of neighbors (k)
k <- 5

# Train the KNN model
knn_model <- knn(train = train_set3[, 1:4], test = test_set3[, 1:4], cl = train_set3$HeartDisease, k = k)

# generate confusion matrix
knn_conf_matrix <- confusionMatrix(knn_model, test_set3$HeartDisease)
knn_accuracy <- knn_conf_matrix$overall['Accuracy']
knn_recall <- knn_conf_matrix$byClass['Sensitivity']
knn_precision <- knn_conf_matrix$byClass['Pos Pred Value']
#knn_f1_score <- 2 * (precision * recall) / (precision + recall)
knn_f1_score <- knn_conf_matrix$byClass["F1"]

# Print result
print(knn_conf_matrix$table)
```

```
##           Reference
## Prediction  0  1
##          0 93 26
##          1 20 72
```

```
print(paste('Accuracy: ', round(knn_accuracy, 4)))
```

```
## [1] "Accuracy:  0.782"
```

```
print(paste('Precision: ', round(knn_precision, 4)))
```

```
## [1] "Precision:  0.7815"
```

```
print(paste('Recall: ', round(knn_recall, 4)))
```

```
## [1] "Recall:  0.823"
```

```
print(paste('F1 Score: ', round(knn_f1_score, 4)))
```

```
## [1] "F1 Score:  0.8017"
```

Support Vector Machines (SVM)

Support Vector Machines algorithm is a machine learning technique that identifies the optimal decision boundary to distinguish different categories. It strives to increase the margin among the supporting vectors and can handle complicated data sets. To be effective in high-dimensional spaces, SVM uses kernel functions. It is frequently used for classification assignments, although it can also be employed for regression tasks.

Table below lists the advantages and disadvantages of SVM:

Advantages	Disadvantages
• Effective in high-dimensional spaces and with small to medium-sized datasets. • Able to handle both linear and non-linear classification. • Robust to overfitting.	• Requires proper selection of hyperparameters. • Sensitive to noise and outliers. • Relatively difficult to interpret.

```
# Convert the Species column to a factor variable
# Using Train set 2 and test set 2 which already converted.

# Train an SVM model with a linear kernel
svm_model <- svm(HeartDisease ~ ., data = train_set2, kernel = "linear")

# Make predictions on the test set
svm_predicted <- predict(svm_model, test_set2)

# Show the confusion matrix and performance metrics
svm_cm <- confusionMatrix(svm_predicted, test_set2$HeartDisease)
svm_accuracy <- svm_cm$overall["Accuracy"]
svm_precision <- svm_cm$byClass["Precision"]
svm_recall <- svm_cm$byClass["Recall"]
svm_f1_score <- svm_cm$byClass["F1"]

# Print result
print(svm_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 97 18
##          1 16 80
##
##                Accuracy : 0.8389
##                  95% CI : (0.7822, 0.8858)
##     No Information Rate : 0.5355
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6756
##
```

```
##  Mcnemar's Test P-Value : 0.8638
##
##             Sensitivity : 0.8584
##             Specificity : 0.8163
##          Pos Pred Value : 0.8435
##          Neg Pred Value : 0.8333
##              Prevalence : 0.5355
##          Detection Rate : 0.4597
##    Detection Prevalence : 0.5450
##       Balanced Accuracy : 0.8374
##
##        'Positive' Class : 0
##
```

```
print(paste('Accuracy: ', round(svm_accuracy, 4)))
```

```
## [1] "Accuracy:  0.8389"
```

```
print(paste('Precision: ', round(svm_precision, 4)))
```

```
## [1] "Precision:  0.8435"
```

```
print(paste('Recall: ', round(svm_recall, 4)))
```

```
## [1] "Recall:  0.8584"
```

```
print(paste('F1 Score: ', round(svm_f1_score, 4)))
```

```
## [1] "F1 Score:  0.8509"
```

Random Forest (RF)

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

Table below lists the advantages and disadvantages of Random Forest:

Advantages	Disadvantages
• Able to handles both numerical and categorical data. • Robust to outliers and can handle missing values. • Provides feature importance measures for variable selection.	• Lack of interpretability compared to individual decision trees. • May not perform well when dealing with imbalanced datasets. • May need Hyperparameter tuning for optimal performance.

```
# Create random forest model
model_RF <- randomForest(HeartDisease ~ ., data = train_set, ntree = 100)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```
# Model Evaluation
# Compare model prediction outcome to the result from the test set
predict_RF <- predict(model_RF, test_set, type = "class")
compare_RF <- table(test_set$HeartDisease, predict_RF)
# compare_RF
# Output: The table will display the comparison between the actual values and the predicted values (confusion matrix)

# Measure model performance (Accuracy)
acc_test <- sum(diag(compare_RF)) / sum(compare_RF)
print(paste('Accuracy: ', round(acc_test, 4)))
```

```
## [1] "Accuracy:  0.0095"
```

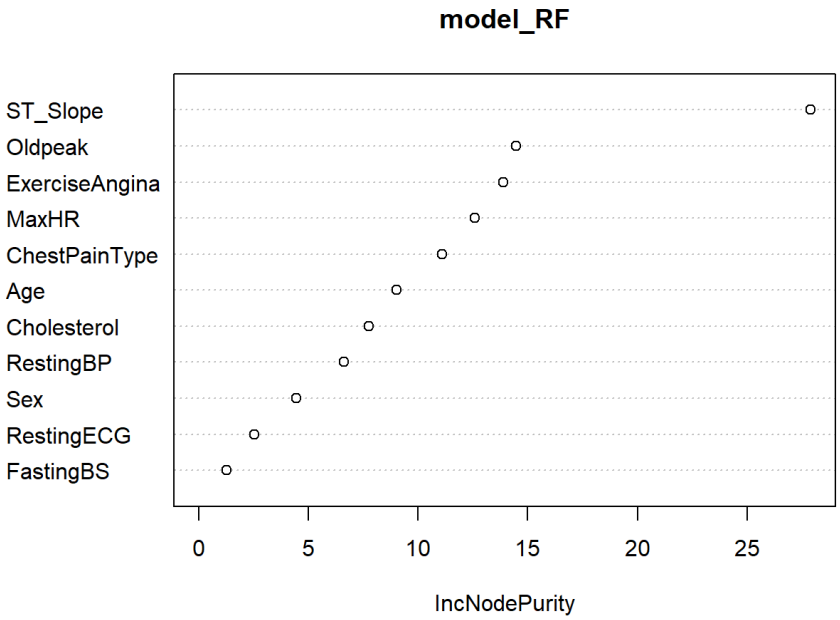
```
# Output: Accuracy of the random forest model on the test set
```

IncNodePurity of the RF

```
# Print variable importance
var_importance <- importance(model_RF)
print(var_importance)
```

```
##                IncNodePurity
## Age                 9.016104
## Sex                 4.459293
## ChestPainType      11.095867
## RestingBP           6.629511
## Cholesterol         7.743166
## FastingBS           1.270644
## RestingECG          2.535977
## MaxHR              12.580324
## ExerciseAngina     13.875845
## Oldpeak            14.462963
## ST_Slope           27.897798
```

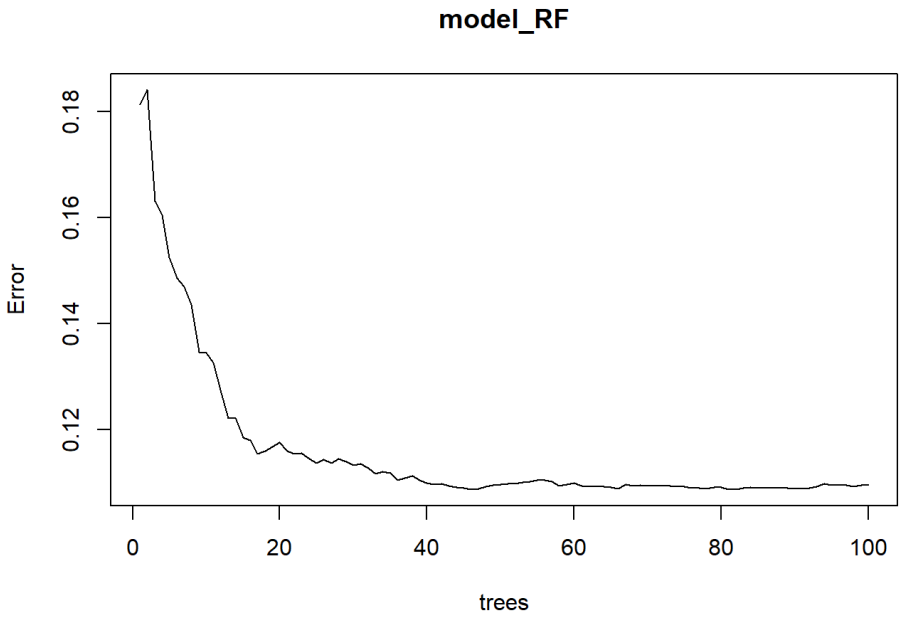
```
# Output: Importance measures for each variable in the random forest model

# Plot the random forest - IncNodePurity
varImpPlot(model_RF)
```



High IncNodePurity indicates a variable having a strong predictive power and informativeness, whereas low IncNodePurity suggests a weaker variable with limited impact on node purity and prediction accuracy. The graph demonstrates that the ST_Slope variable exhibits the highest informativeness and strongest predictive power, with a IncNodePurity of 27.9. ChestPainType, MaxHR, ExerciseAngina and Oldpeak also have relatively high IncNodePurity of greater than 10.0, indicating their significant contribution to the model's predictions. Whereas the other features showing a declining trend in terms of IncNodePurity, suggesting lesser impact on the model's predictions comparatively. Notably, Fasting BS exhibits the lowest IncNodePurity, indicating its limited usefulness for accurate predictions and minimal contribution to the overall performance of the random forest model.

Line plot of the RF

```
# Plot the random forest - line plot
plot(model_RF)
```



The line graph above demonstrates that as the number of trees in a random forest model decreases, there is an increased potential for the model's error to rise. In a random forest, each tree contributes to the overall prediction, and the final prediction is obtained by combining the predictions of all the individual trees. When there are fewer trees, the ensemble loses diversity and robustness. This reduction in diversity can hinder the model's ability to effectively capture the intricacies and patterns present in the data, ultimately leading to higher error rates.

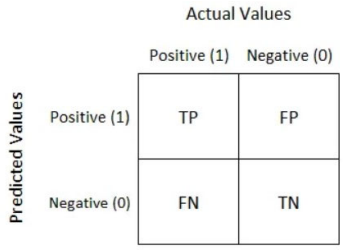
Accuracy of the RF

With an accuracy of only 1% on the test set as calculated above, the RF model significantly underperformed compared to the other assessed models, which achieved an average accuracy of over 80%. It is evident that the random forest model is not suitable for accurately classifying heart failure cases in this specific dataset, and therefore, it will not proceed to the next step within this project. Further investigation and model optimization techniques may be necessary to improve its performance.

Models Evaluation

Model evaluation is the process of assessing the performance and effectiveness of a trained model using various evaluation metrics. It involves measuring how well the model generalizes to unseen data and how accurately it predicts or describes the target variable.

In this case, confusion matrix, as generated in the data modelling phase above, is extremely useful for the evaluation metrics' calculation. Below are the general confusion matrix indicator:




```
# Define the models and their corresponding evaluation metrics
models <- c("Logistic Regression", "Decision Tree", "GradientBoostingClassifier", "KNN","SVM")
accuracy <- c(log_accuracy, accuracy_DT, gbm_accuracy, knn_accuracy, svm_accuracy)
precision <- c(log_precision, precision_DT , gbm_precision, knn_precision, svm_precision)
recall <- c(log_recall, recall_DT, gbm_recall, knn_recall, svm_recall)
f1_score <- c(log_f1_score, f1_DT, gm_f1_score, knn_f1_score, svm_f1_score)

# Create a summary table
summary_table <- data.frame(Model = models, Accuracy = accuracy, Precision = precision,
                            Recall = recall, F1_Score = f1_score)

# Print the summary table
kable(summary_table)
```

Model	Accuracy	Precision	Recall	F1_Score
Logistic Regression	0.8530806	0.8764045	0.7959184	0.8342246
Decision Tree	0.8483412	0.8173077	0.8673469	0.8415842
GradientBoostingClassifier	0.8436019	0.8389831	0.8761062	0.8571429

Model	Accuracy	Precision	Recall	F1_Score
KNN	0.7819905	0.7815126	0.8230088	0.8017241
SVM	0.8388626	0.8434783	0.8584071	0.8508772

In the process of deciding the best model, various evaluation metrics specifically for classification problem are considered to assess the performance of the models, such as:

1. **Accuracy**: To measure of how accurately a classification model predicts the true labels. It provides an overall assessment of the model's performance.

   $$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

   From the evaluation summary table above, the most accurate method appeared to be the Logistic Regression with an accuracy of 85.3%, closely followed by Decision Tree and Gradient Boosting Classifier with an accuracy of 84.8% and 84.4% respectively.

2. **Precision**: To measure of the exactness of a classification model, indicating the model's ability to avoid false positive errors.

   $$\text{Precision} = \frac{TP}{TP+FP}$$

   In this case, with the highest precision of 87.6%, Logistic Regression has a low false positive rate.

3. **Recall / Sensitivity**: To measures the model's ability to correctly identify positive instances, or in other word the ability to avoid false negative errors.

   $$\text{Recall} = \frac{TP}{TP+FN}$$

   With the highest recall rate of 87.6%, Gradient Boosting Classifier is performing well by showing the lowest false negative rate.

4. **F1 Score**: To provide a balanced assessment of the model's performance by considering both false positive and false negative errors (i.e. combining precision and recall into a single value), where a higher value indicates better overall performance.

   $$F1\text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision}+\text{Recall}}$$

   Again, Gradient Boosting Classifier achieve the best F1 score at 85.7%, which balances precision and recall well.

As a result, the **Gradient Boosting Classifier is determined to be the best model** as it performing well for accuracy, recall and F1 score. It also performing fairly good in terms of precision, attaining the 2nd highest score after Logistic Regression. Given that accuracy & F1 Score is the most appropriate and reliable metric for this classification task, the Gradient Boosting Classifier is the best model to be used in predicting heart failure in this case.

Conclusion

After selecting the best model, there are several potential future works that can be considered, such as:

1. **Feature engineering**: Investigating new features or altering existing ones to possibly enhance the models' predictive abilities. The most pertinent features for heart failure prediction could be found by using feature selection techniques.
2. **Model Optimization**: Improving the performance of the models by fine-tuning the model hyperparameters using methods like grid search or random search. This might entail experimenting with various parameter pairings and assessing how they affect the evaluation metrics.
3. **Ensemble Methods**: Examining ensemble learning strategies to possibly improve the accuracy and robustness of the models, such as combining different models or applying strategies like bagging or boosting.
4. **Handling Unbalanced Data**: Dealing with the problem of unbalanced data, particularly if the heart failure dataset has a sizable class imbalance. To increase the model's capacity to predict heart failure accurately, strategies such as oversampling the minority class (cases of heart failure) or undersampling the majority class (cases of non-heart failure) could be investigated.
5. **External Validation**: Testing the trained models on different datasets that haven't been seen before to judge how well they generalize. This step is essential to verify the models' performance and dependability beyond the available dataset.
6. **Interpretability and Explainability**: Examining methods for interpreting and explaining the model predictions. To do this and foster confidence in the model, it may be necessary to employ techniques like feature importance analysis or the creation of model-independent explanations.

To sum up, the Gradient Boosting Classifier appears to be the best model for predicting heart failure in this case as it achieves a high score in terms of accuracy, with a value of 84.4%. Additionally, it demonstrates a good balance between precision and recall, achieving 85.7% of F1 score, which combines precision and recall, demonstrating a great performance in terms of prediction accuracy and generalization ability.

Overall, the heart failure prediction dataset offers a useful starting point for developing machine learning models in this domain. However, there are several areas that can be further explored to enhance the predictive performance and practical applicability of the models, including feature engineering to extract more relevant information from the data, model optimization to fine-tune the hyperparameters for improved performance, addressing imbalanced data to prevent biased predictions, external validation to assess the generalizability of the models on unseen data, and enhancing interpretability to gain more insights into the factors contributing to heart failure.

Further research and development efforts in these areas can lead to more accurate and reliable heart failure prediction models, ultimately contributing to enhanced diagnosis and treatment of heart failure patients in real-world scenarios.

About Us

This is a university group project for the course WQD7004 Programming for Data Science under University of Malaya, Malaysia.

We are Cloud Nine (Group 9) from Occ 2, Semester 2 2022/2023.

- Tan Hooi Yi (s2160036)
- Vivian Sam (s2139601)
- Jinat Ahmed (s2177543)
- Elaine Li (s2164604)
- Tang Ying Ming (s2180377)

Thanks for your attention ;)