DSC 411:Final Report

Sakshi Gorkhali

2024-11-14

A. Data Gathering:

I found this data from Kaggle named as Heart Failure Prediction Data set, it is used to predict whether a person is likely to have a heart failure by studying their medical report.

```
heart <- read.csv("/Users/cdmstudent/Downloads/DSC411-FundamentalsofDataScience/heart.csv")
head(heart)
```

```
##   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
```

```
#Looking for missing values
sum(is.na(heart))
```

```
## [1] 0
```

```
summary(heart)
```

```
##       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
```

From searching to see if there are any NA values in our dataset, we can see that there are non but from the summary statistics we can see that there are zero values in resting BP and Cholesterol. As from our knowledge we know that resting BP and cholesterol both cannot be zero, so we search them.

```
sum(heart$Cholesterol==0)
```

```
## [1] 172
```

```
sum(heart$RestingBP==0)
```

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

```
dim(heart)
```

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

There are total of 172 data with cholesterol= 0 and 1 data with RestingBP= 0. As we have 918 data and also cholesterol for each patient is different and is better not to assume them, we remove them.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
heart<- heart %>%
  filter(RestingBP != 0, Cholesterol != 0)
dim(heart)
```

```
## [1] 746  12
```

B. Data Exploration:

```
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
#Checking the summary of our data
str(heart)
```

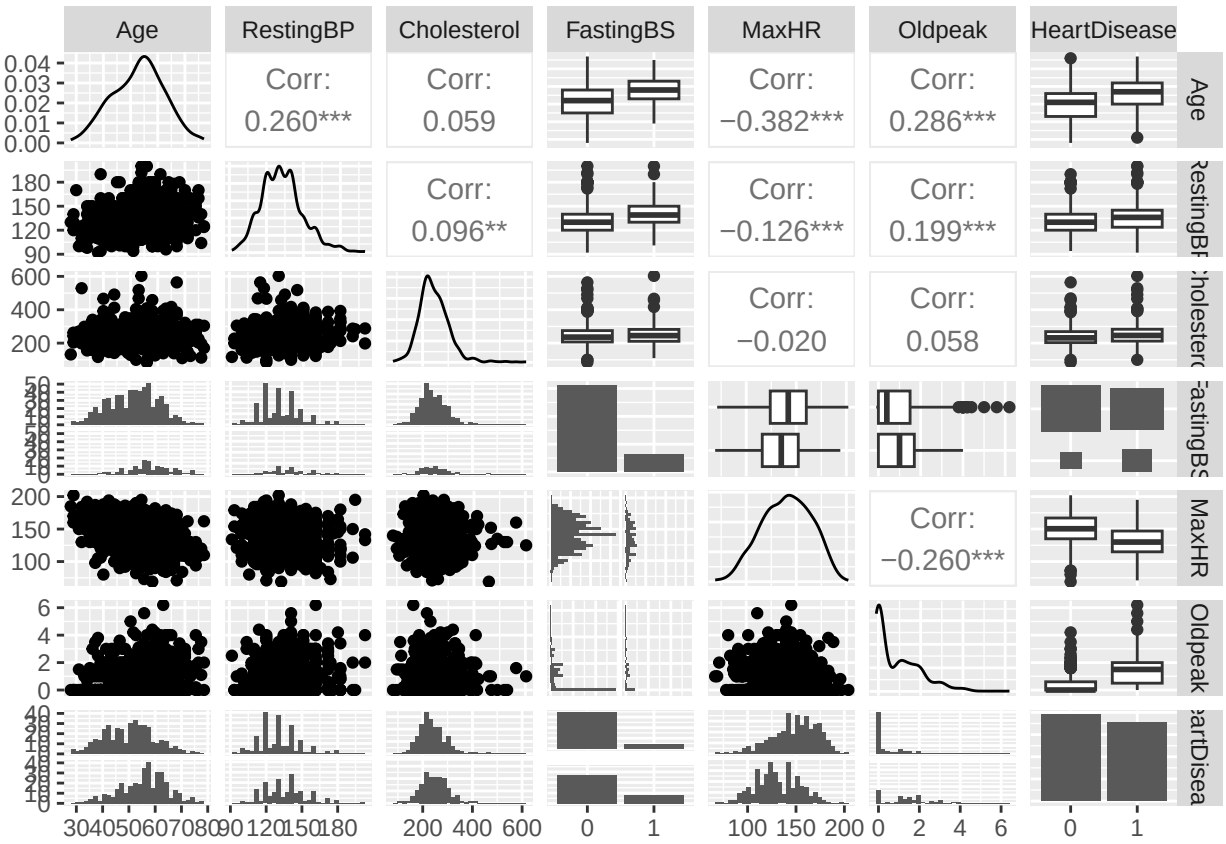
```
## 'data.frame':    746 obs. of  12 variables:
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
```

```
##  $ 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 ...
```

```
heart$HeartDisease<- as.factor(heart$HeartDisease)
heart$FastingBS<- as.factor(heart$FastingBS)
heart$Sex<- as.factor(heart$Sex)
heart$ChestPainType<- as.factor(heart$ChestPainType)
heart$ExerciseAngina<- as.factor(heart$ExerciseAngina)
heart$ST_Slope<- as.factor(heart$ST_Slope)
heart$RestingECG<- as.factor(heart$RestingECG)



#Exploring correlation of our numerical variables with eachother and heartdisease:
numerical<- c("Age","RestingBP","Cholesterol","FastingBS","MaxHR", "Oldpeak","HeartDisease")
ggpairs(heart[numerical])
```

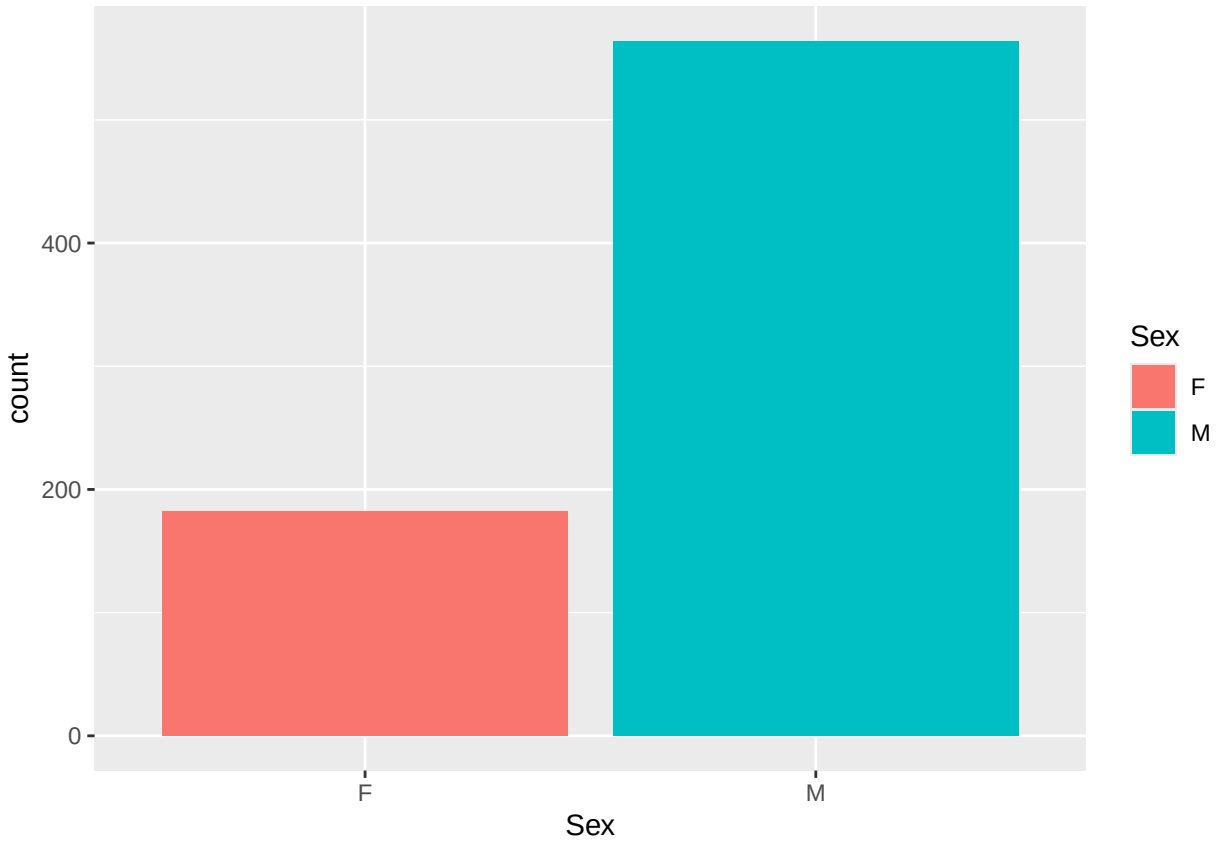
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

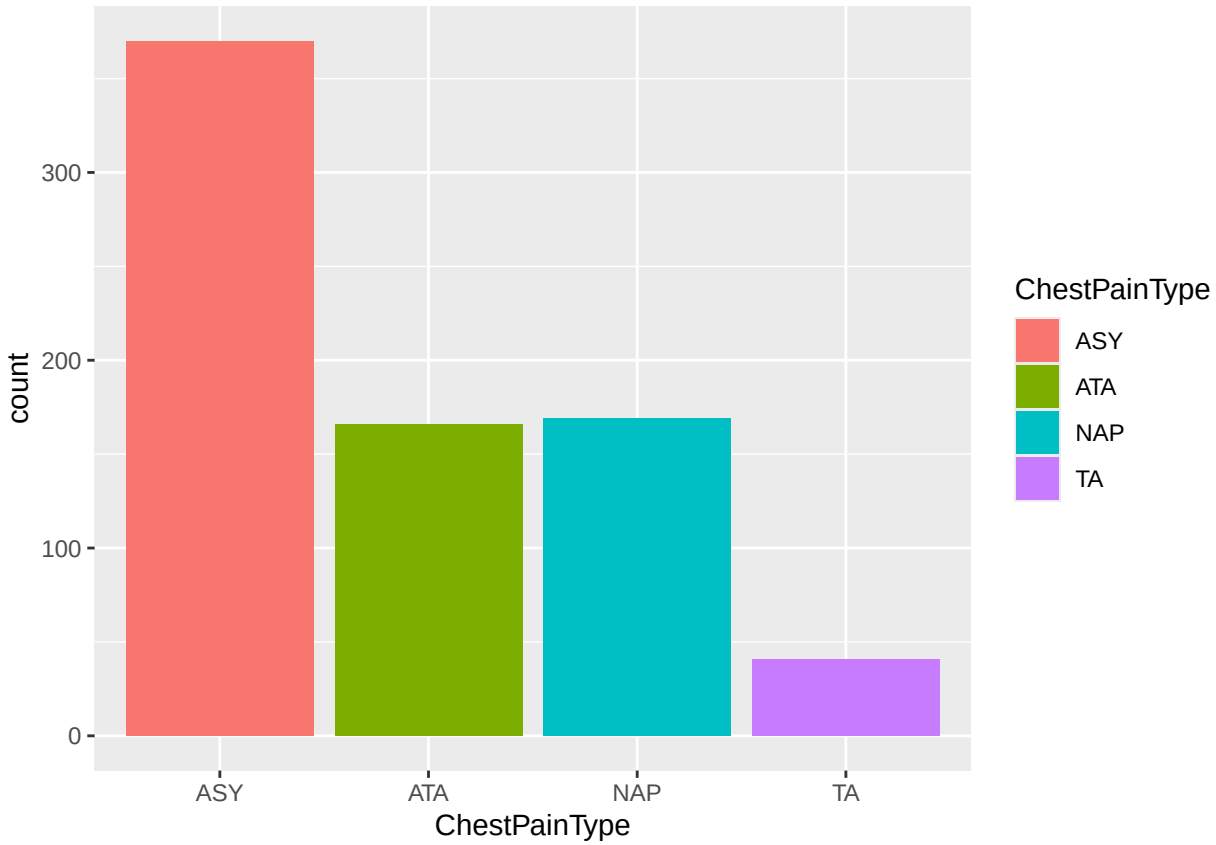


- Exploring the distribution of our data:

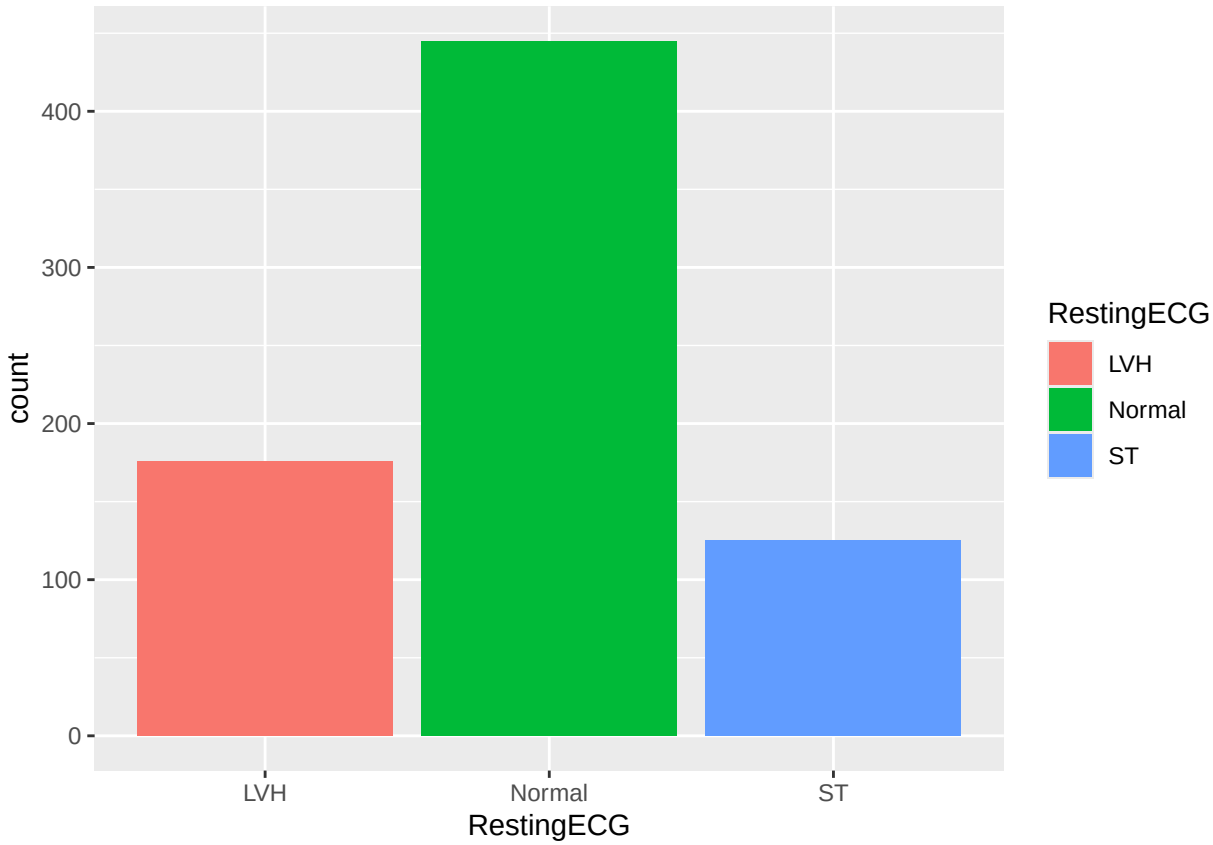
```
#For categorical data:
ggplot(heart, aes(x=Sex, fill=Sex)) +
geom_bar()
```



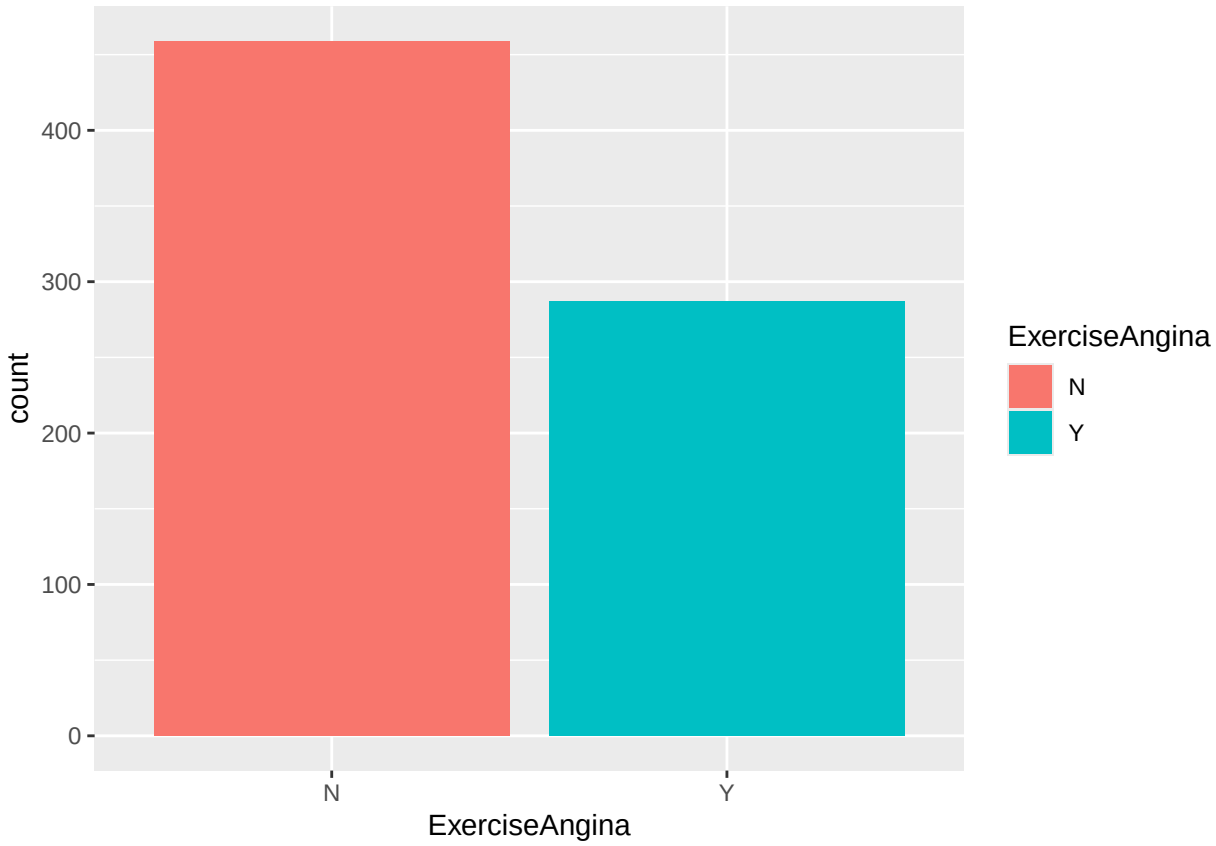

```
ggplot(heart, aes(x=ChestPainType, fill=ChestPainType)) +
geom_bar()
```



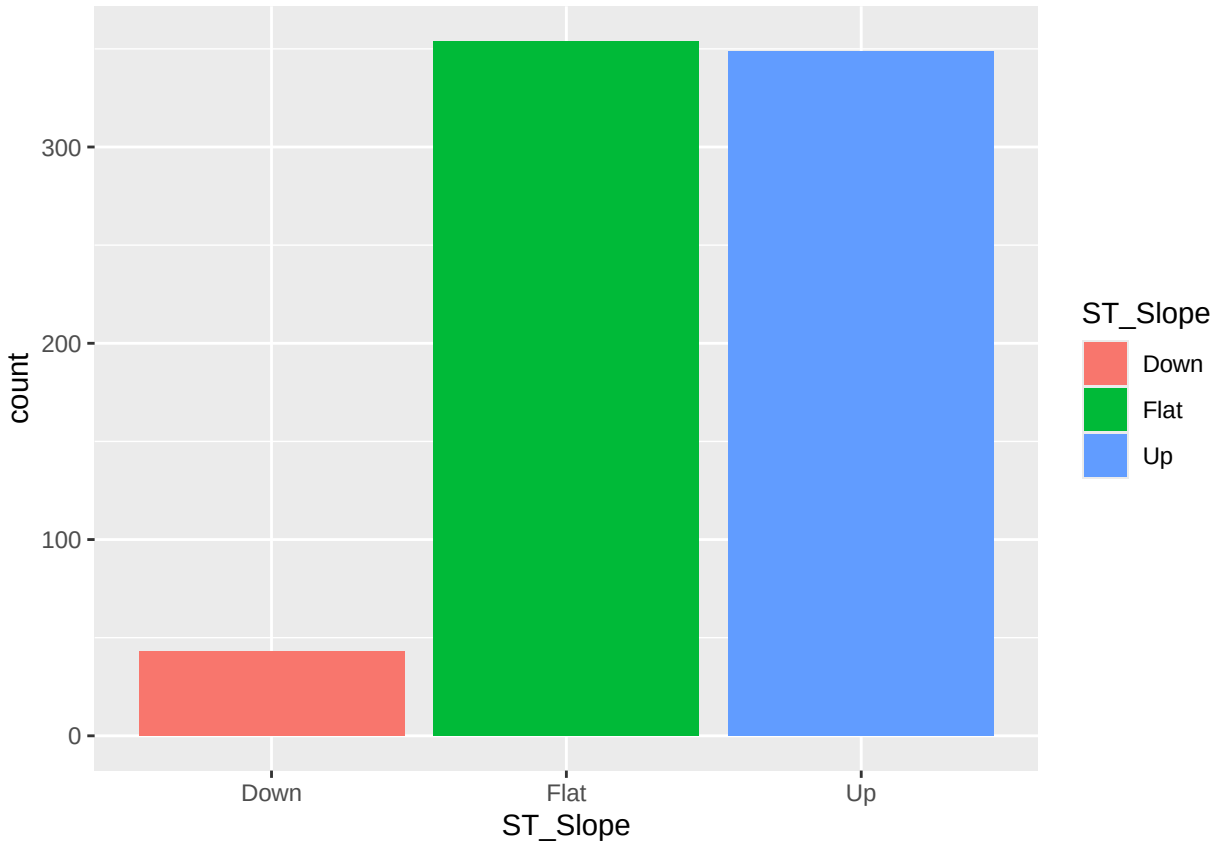

```
ggplot(heart, aes(x=RestingECG, fill=RestingECG)) +
geom_bar()
```



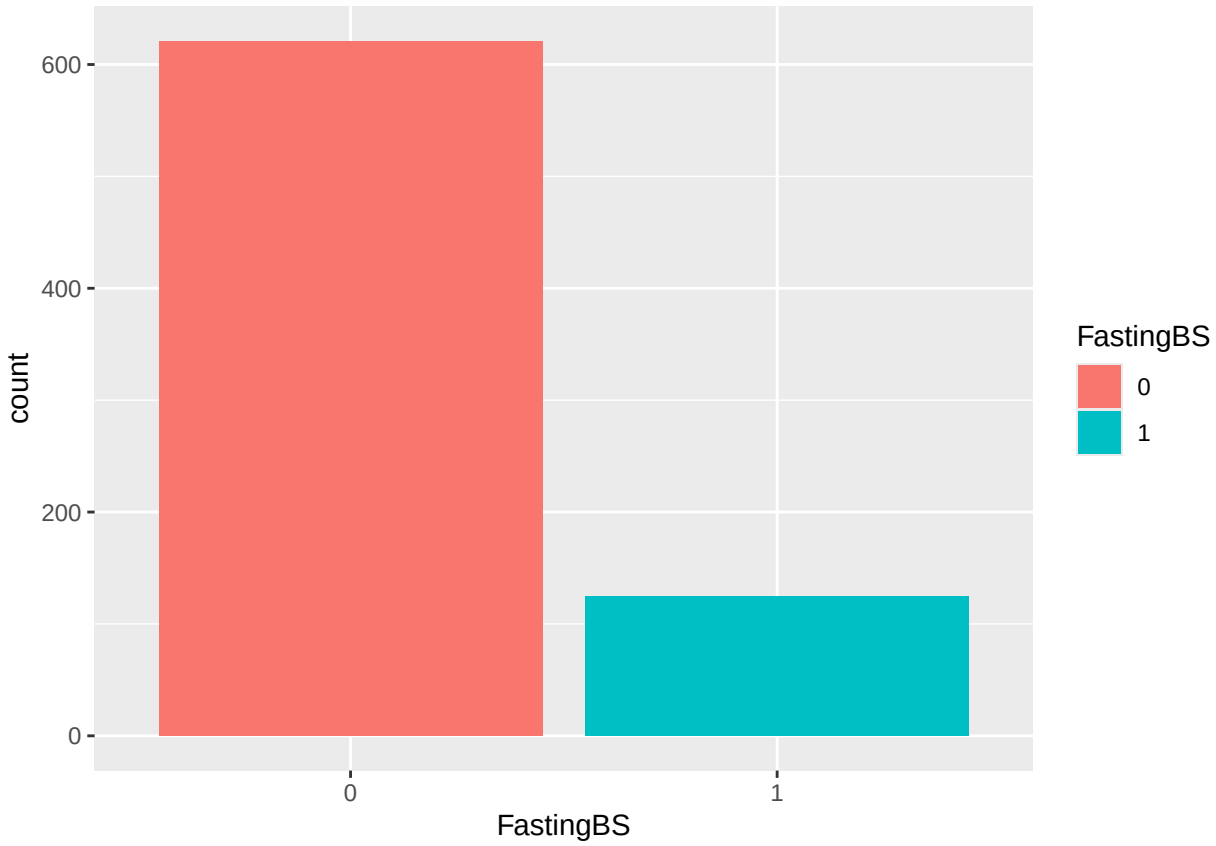

```
ggplot(heart, aes(x=ExerciseAngina, fill=ExerciseAngina)) +
geom_bar()
```




```
ggplot(heart, aes(x=ST_Slope, fill=ST_Slope)) +
geom_bar()
```



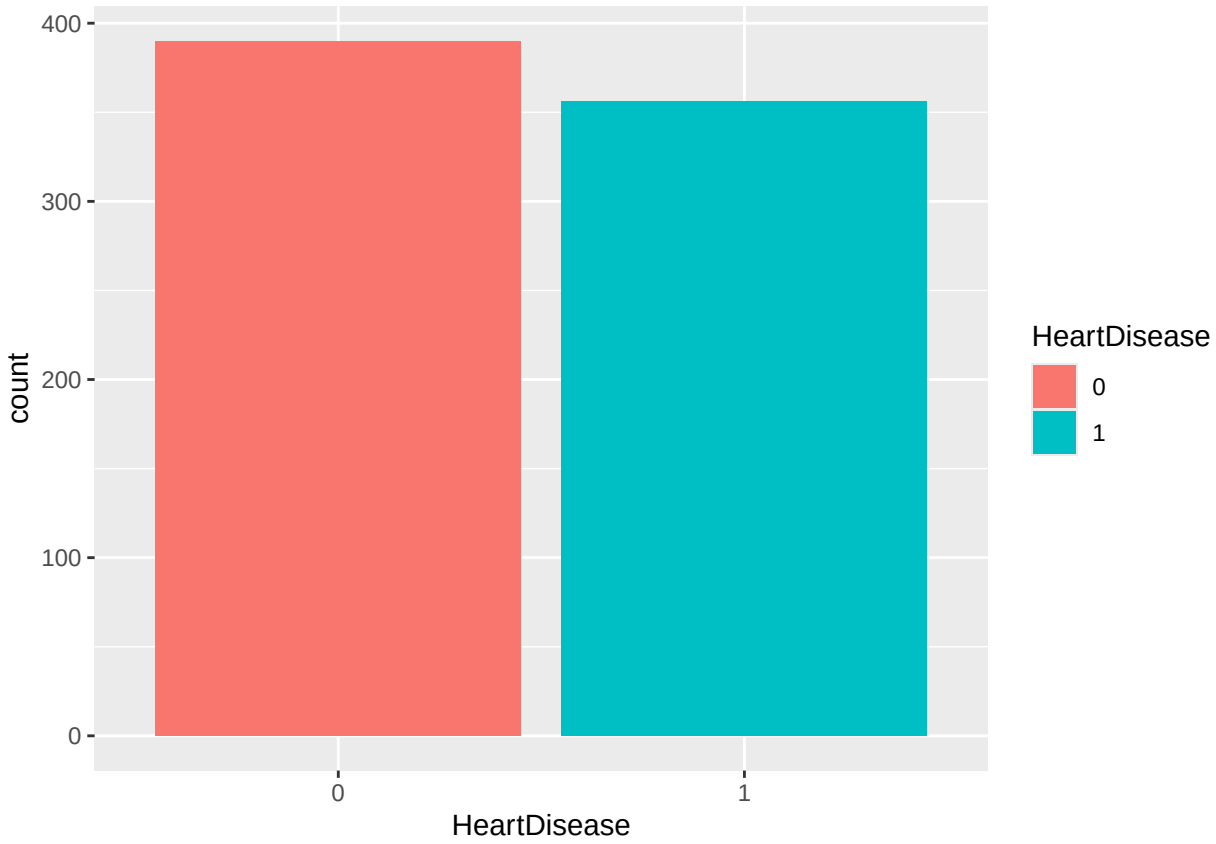

```
ggplot(heart, aes(x=FastingBS, fill=FastingBS)) +
geom_bar()
```



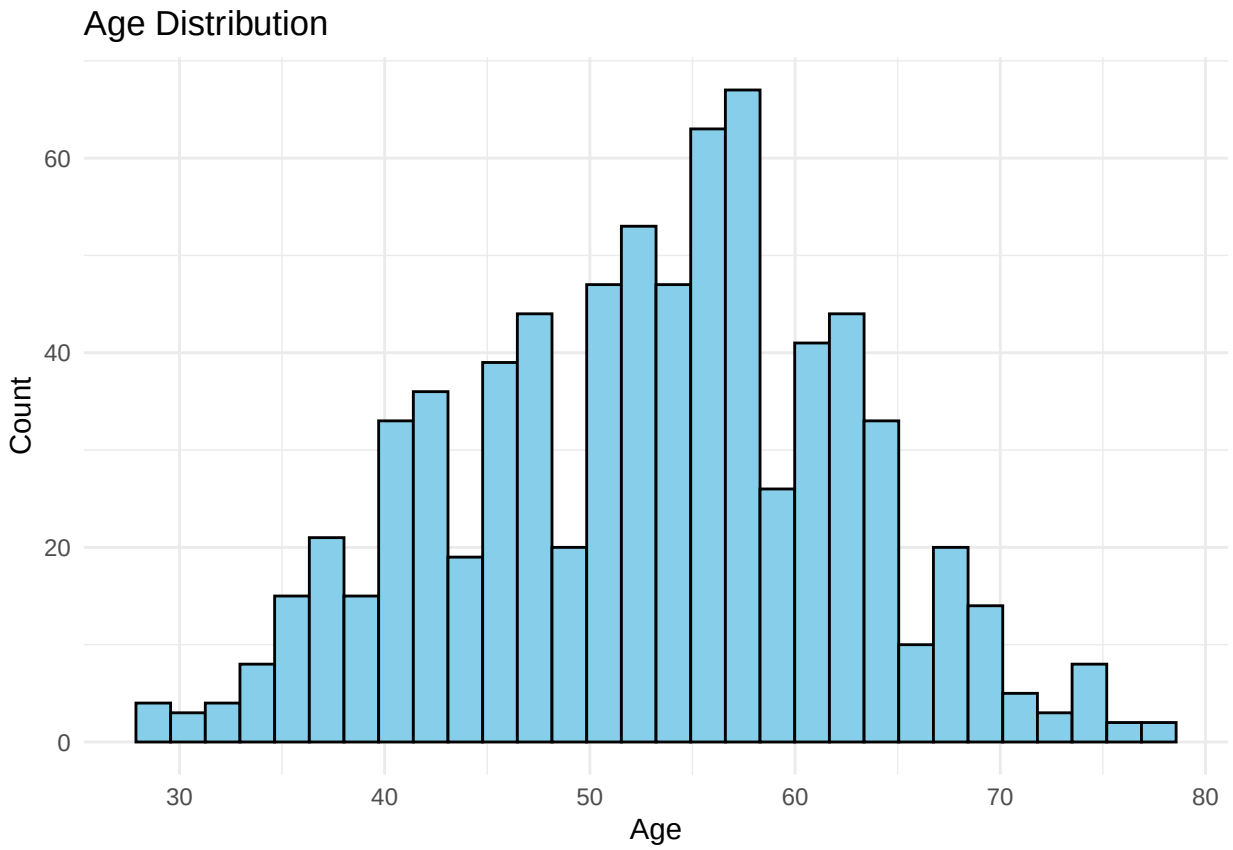

```
heart%>%
   group_by(HeartDisease)%>%
   summarise('count'=n())
```

```
## # A tibble: 2 x 2
##   HeartDisease count
##   <fct>        <int>
## 1 0              390
## 2 1              356
```

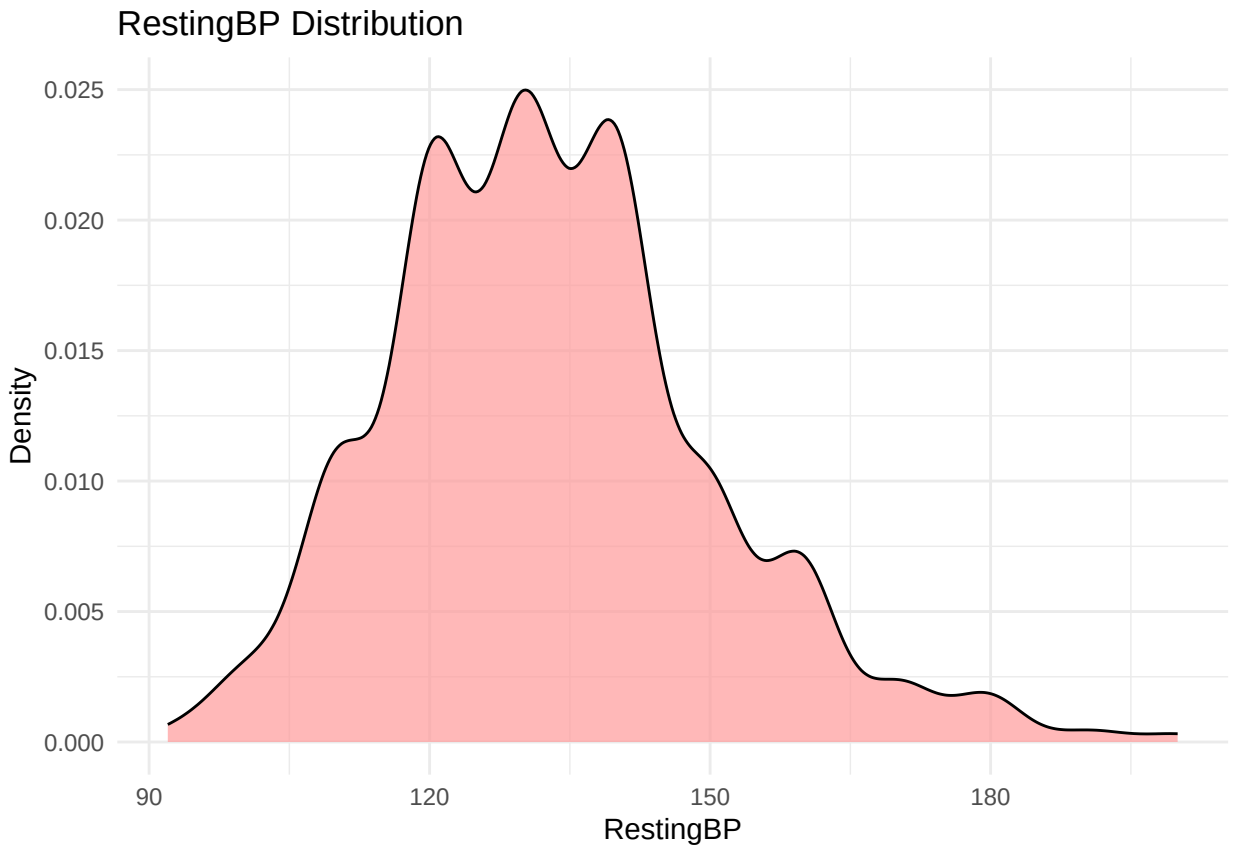
```
ggplot(heart, aes(x=HeartDisease, fill=HeartDisease)) +
geom_bar()
```



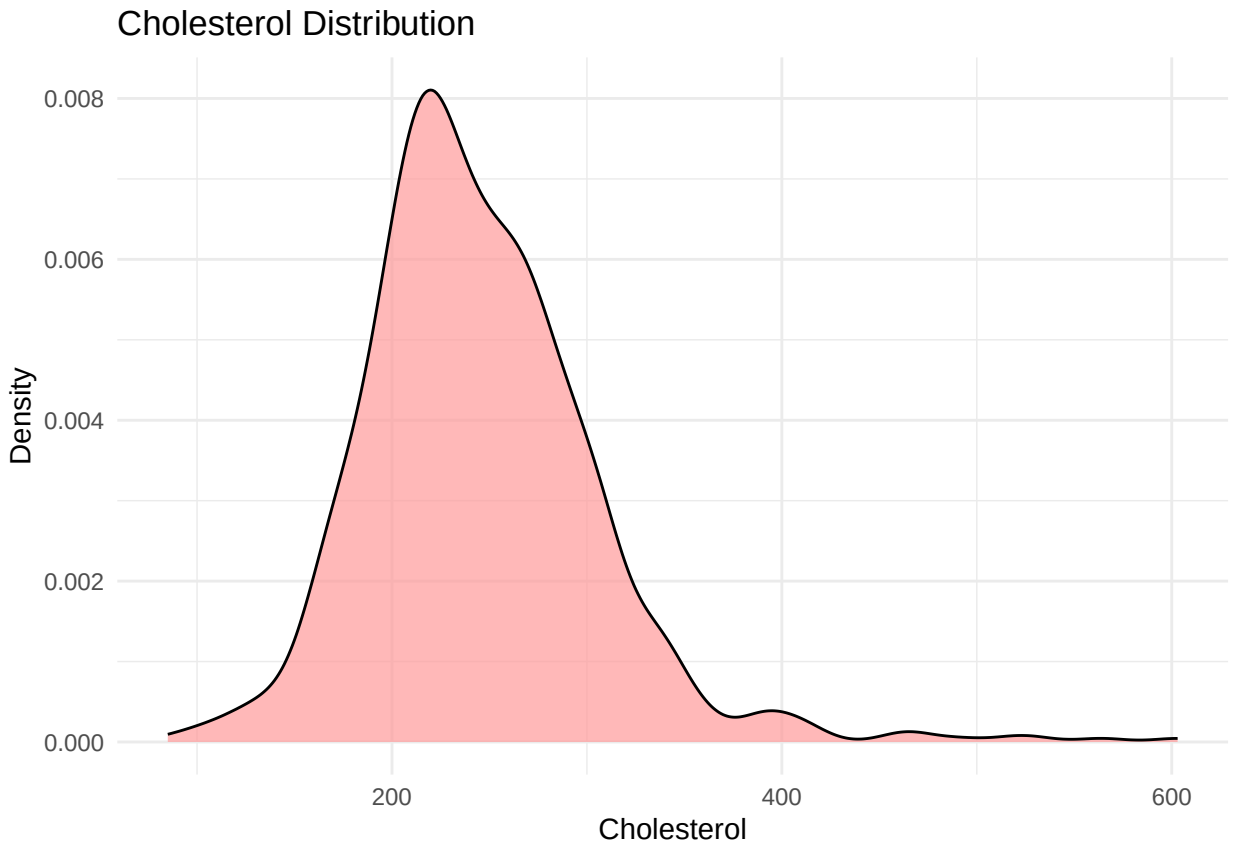

```
#For numerical variables:
ggplot(heart, aes(x = Age)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  theme_minimal() +
  labs(title = "Age Distribution", x = "Age", y = "Count")
```



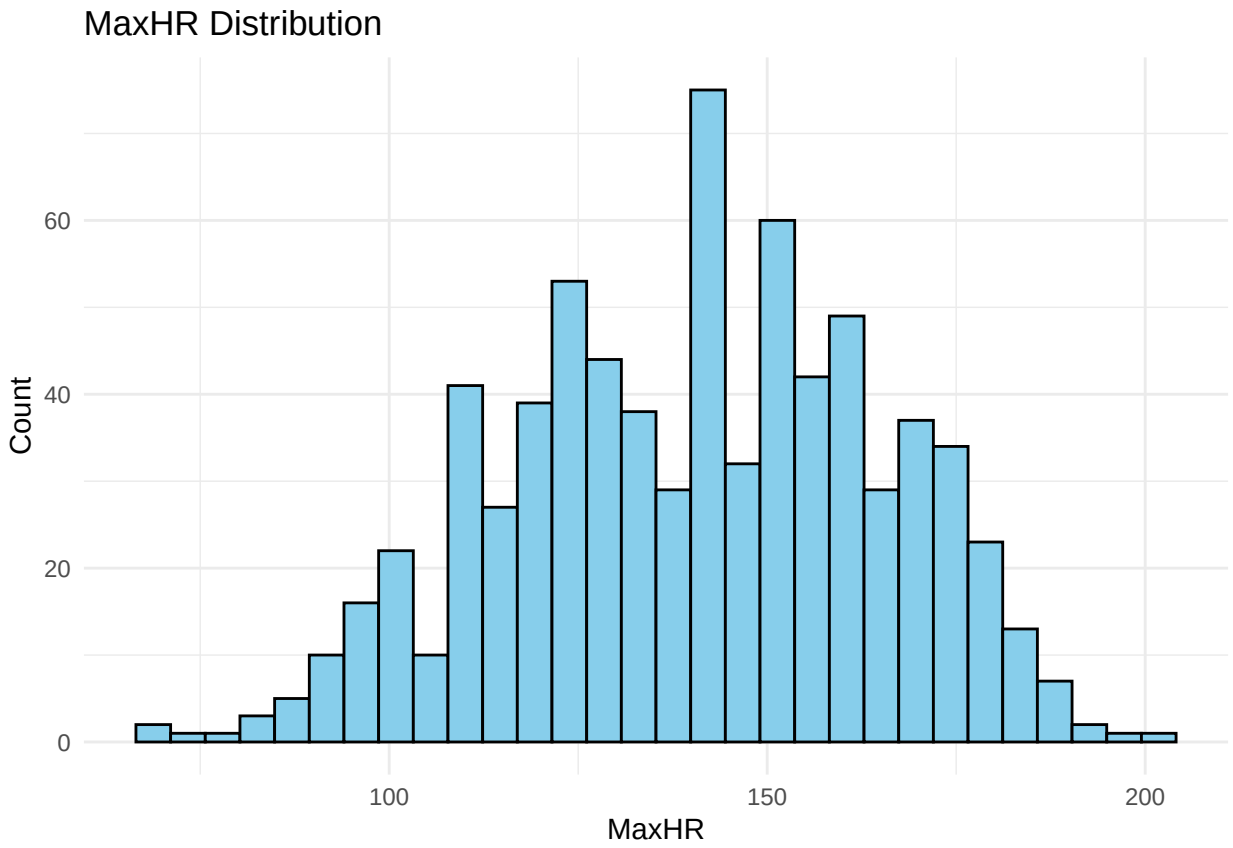

```
ggplot(heart, aes(x=RestingBP)) +
  geom_density(fill="#FF9999", alpha=0.7) +
  labs(title="RestingBP Distribution", x="RestingBP", y="Density") +
  theme_minimal()
```



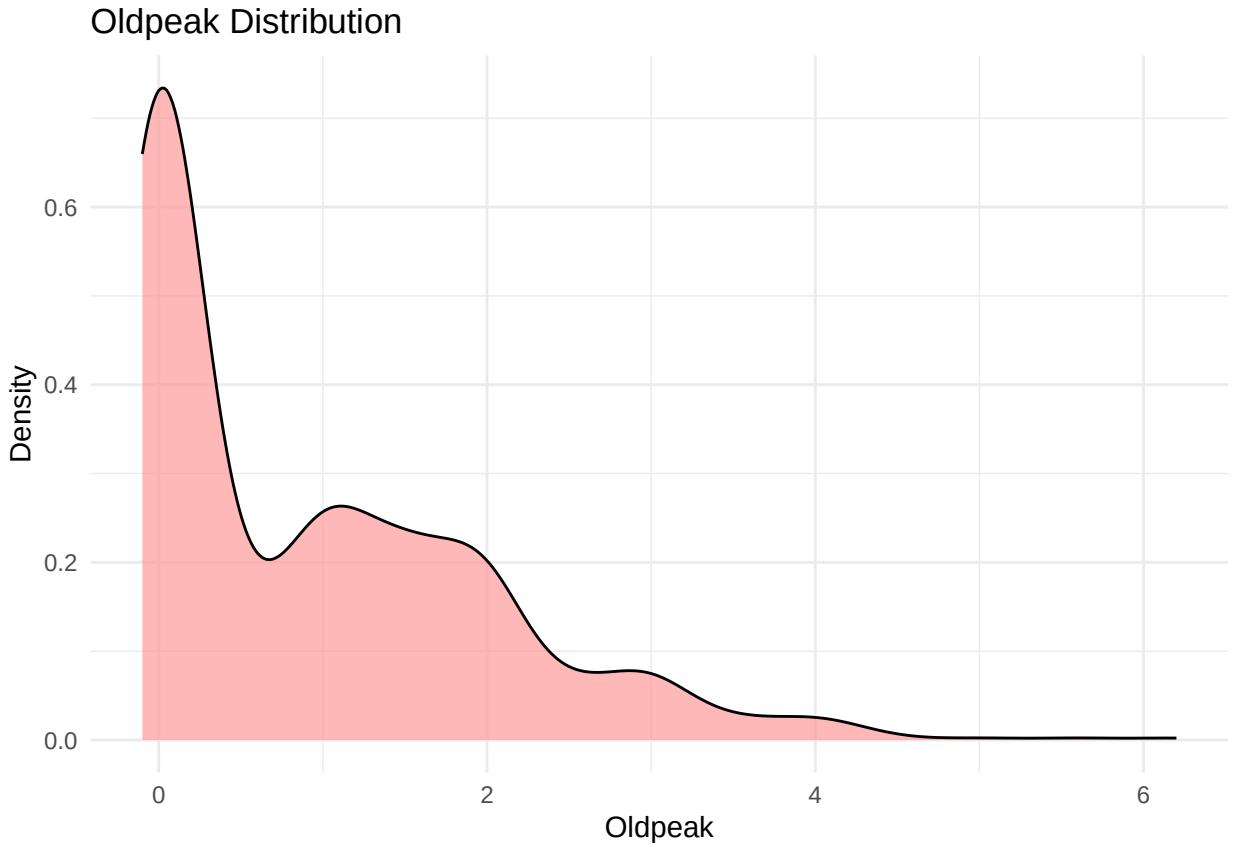

```r
ggplot(heart, aes(x=Cholesterol)) +
  geom_density(fill="#FF9999", alpha=0.7) +
  labs(title="Cholesterol Distribution", x="Cholesterol", y="Density") +
  theme_minimal()
```




```
ggplot(heart, aes(x = MaxHR)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  theme_minimal() +
  labs(title = "MaxHR Distribution", x = "MaxHR", y = "Count")
```




```
ggplot(heart, aes(x=Oldpeak)) +
  geom_density(fill="#FF9999", alpha=0.7) +
  labs(title="Oldpeak Distribution", x="Oldpeak", y="Density") +
  theme_minimal()
```

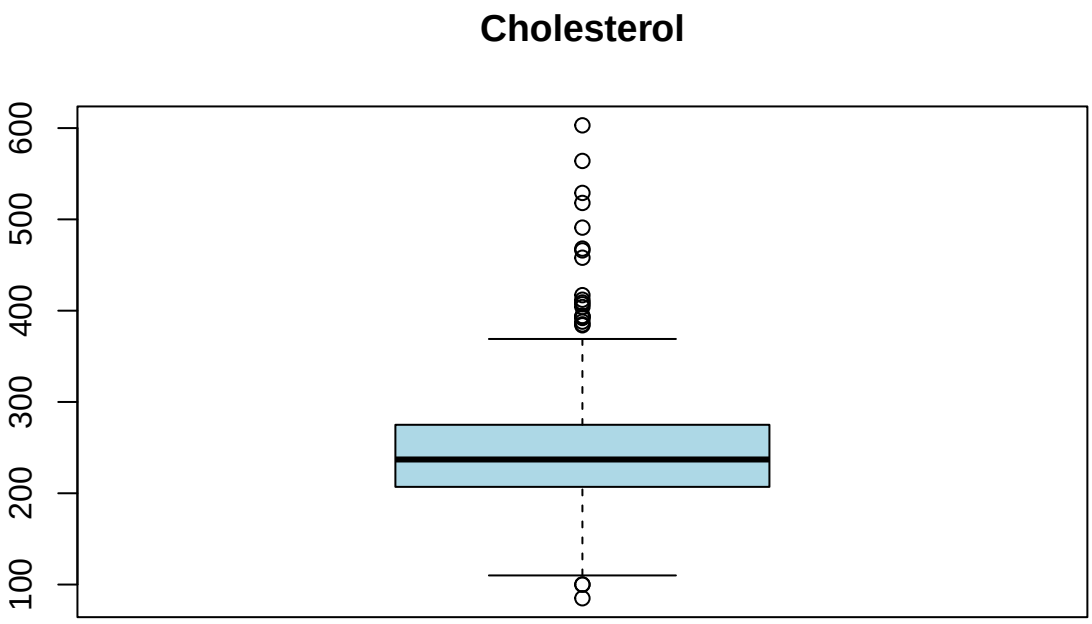


From the distribution most of our seem to be distributed well. Our target variable Heart Disease is also equally distributed.

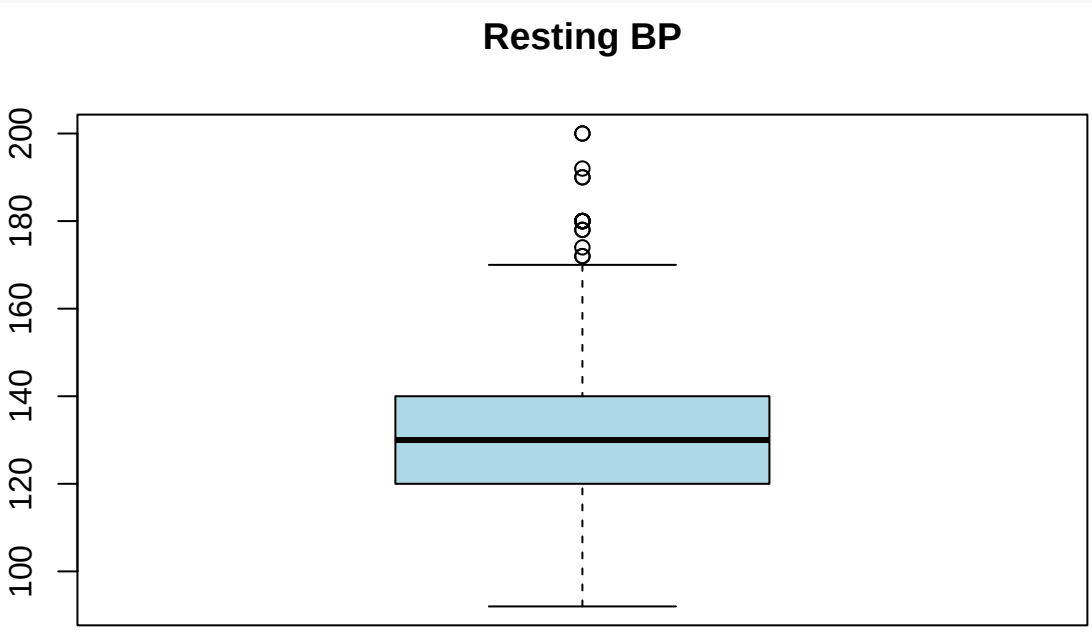
C. Data Cleaning:

We removed na and zero values from our data set in **data gathering process** and from the study of the distribution our data seems to be well distributed for both categorical as well as numerical. Now using box plot we look at the outliers that are present in our data.

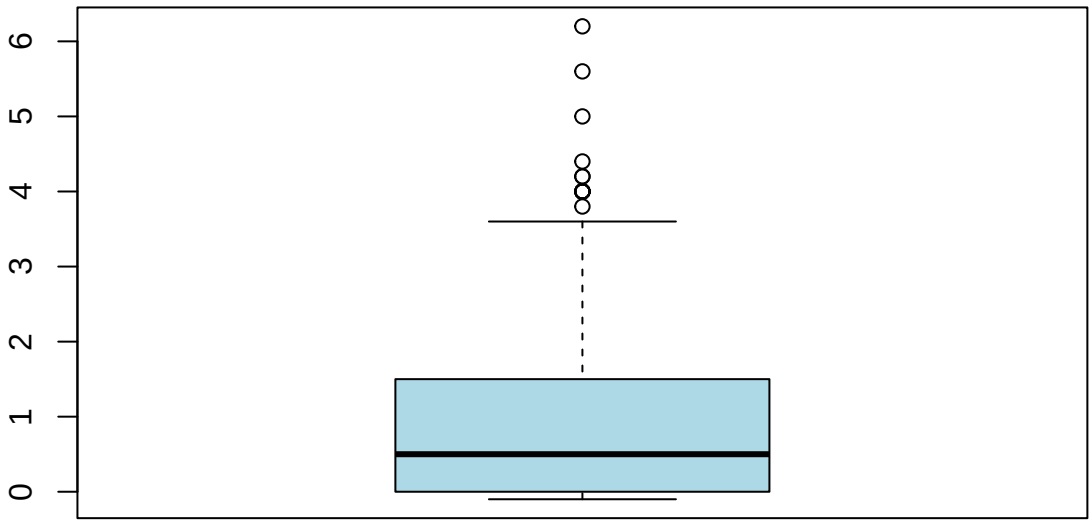
```
boxplot(heart$Cholesterol,main="Cholesterol", col="lightblue")
```



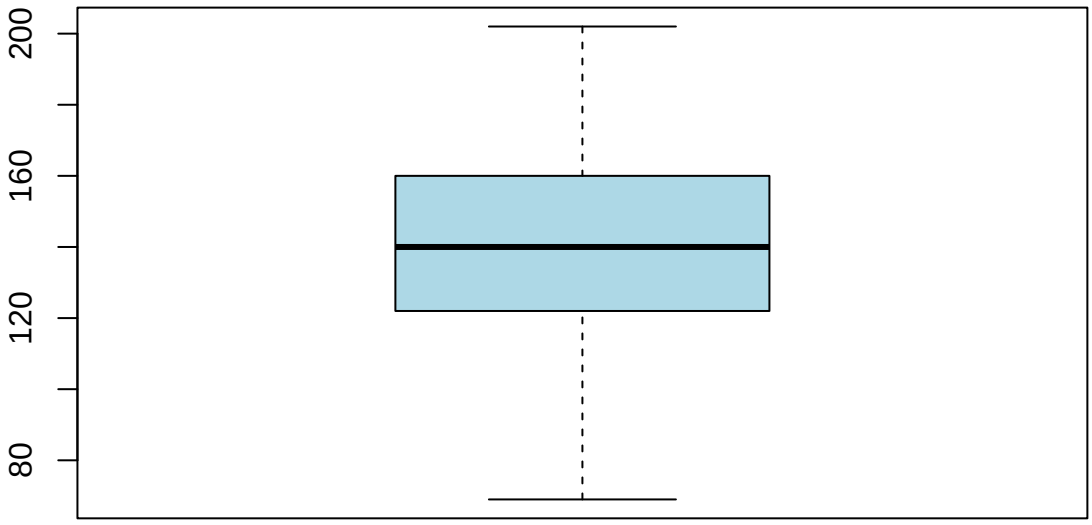

```
boxplot(heart$RestingBP,main="Resting BP", col="lightblue")
```




```
boxplot(heart$Oldpeak,main="Old Peak", col="lightblue")
```




```r
boxplot(heart$MaxHR,main="Max HR", col="lightblue")
```




```r
numerical<- c("RestingBP","Cholesterol", "Oldpeak")
clean_heart<- heart
for (col in numerical) {
  Q1 <- quantile(clean_heart[[col]], 0.25)
  Q3 <- quantile(clean_heart[[col]], 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR

# Keep only rows where the values are within the bounds
clean_heart <- clean_heart[clean_heart[[col]] >= lower_bound & clean_heart[[col]] <= upper_bound, ]
}
```

```
summary(clean_heart)
```

```
##       Age        Sex     ChestPainType   RestingBP      Cholesterol
##  Min.   :28.00   F:165   ASY:332       Min.   : 92.0   Min.   :110.0
##  1st Qu.:46.00   M:527   ATA:158       1st Qu.:120.0   1st Qu.:206.0
##  Median :54.00           NAP:163       Median :130.0   Median :234.5
##  Mean   :52.68           TA : 39       Mean   :131.4   Mean   :238.8
##  3rd Qu.:59.00                         3rd Qu.:140.0   3rd Qu.:271.0
##  Max.   :77.00                         Max.   :170.0   Max.   :369.0
##  FastingBS  RestingECG      MaxHR       ExerciseAngina    Oldpeak
##  0:580     LVH   :162   Min.   : 71.0   N:434          Min.   :-0.1000
##  1:112     Normal:417   1st Qu.:122.0   Y:258          1st Qu.: 0.0000
##            ST    :113   Median :141.0                  Median : 0.4000
##                         Mean   :140.6                  Mean   : 0.8299
##                         3rd Qu.:160.0                  3rd Qu.: 1.5000
##                         Max.   :202.0                  Max.   : 3.6000
##  ST_Slope   HeartDisease
##  Down: 32   0:372
##  Flat:326   1:320
##  Up  :334
##
##
##
```

```
dim(clean_heart)
```

```
## [1] 692  12
```

From our plot, we can see there are some presence of some outliers, I first tried to bin the data into separate bins for cholesterol and Resting BP according to medical standards instead of just **binning** according to the number of bins I want. But, according to different sources, the separation of low normal and high depends on various factor such as sex, age of the patient so it is difficult to generalize a specific point to separate the labels. So instead I opted to removing them. Also, our target variable has 2 labels so all the outliers at the two end will belong to either class so removing them might not impact the understanding of the data and also as our classifiers are sensitive to outliers, removing them will help in better prediction. Our data set has 692 data points after removal.

D. Data Pre processing:

Normalizing our numerical data with min and max for better predictions as non of our data have negative and having all data normalized to same scale helps better clustering.

```
library(caret)
```

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

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

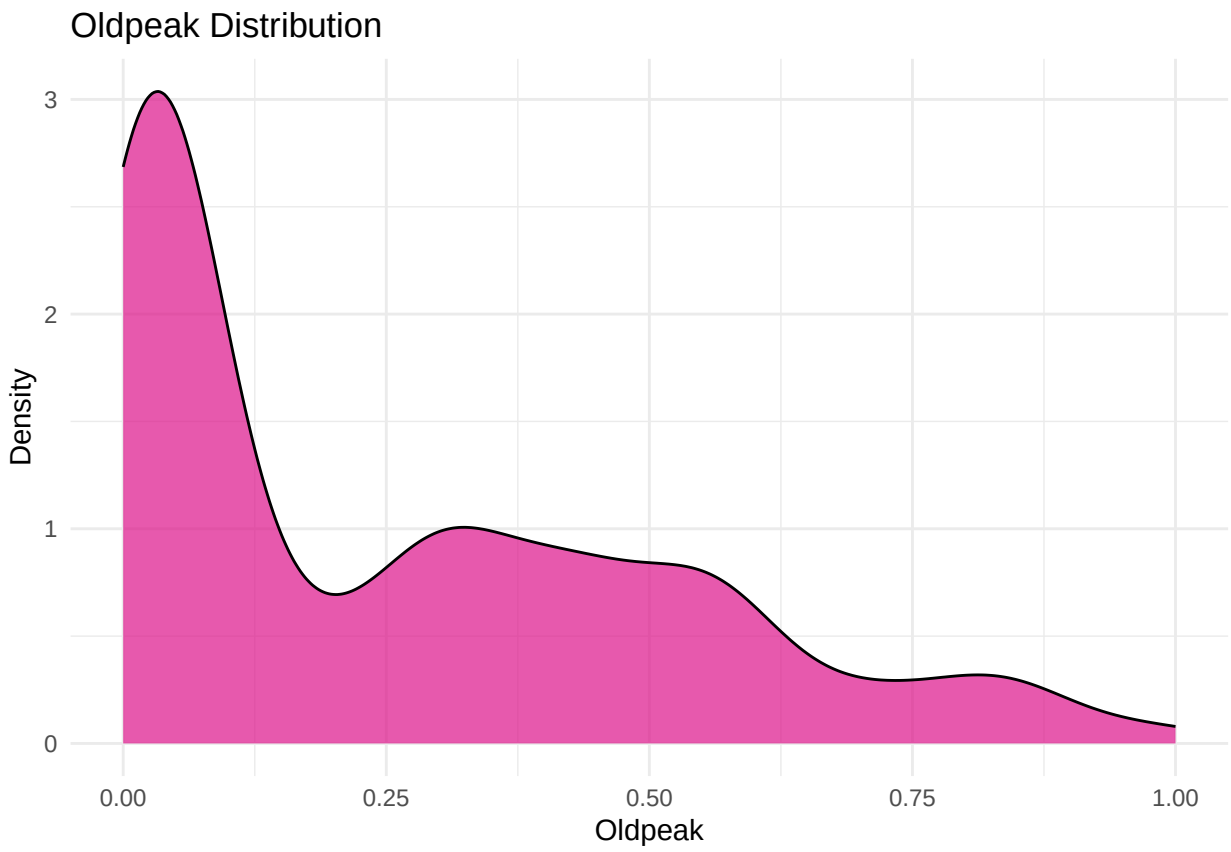
```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
numerical<- c("Cholesterol", "Oldpeak","Age","MaxHR", "RestingBP")
preprocess_model <- preProcess(clean_heart[numerical], method = c("range"))

#Apply normalization to numerical variables
```

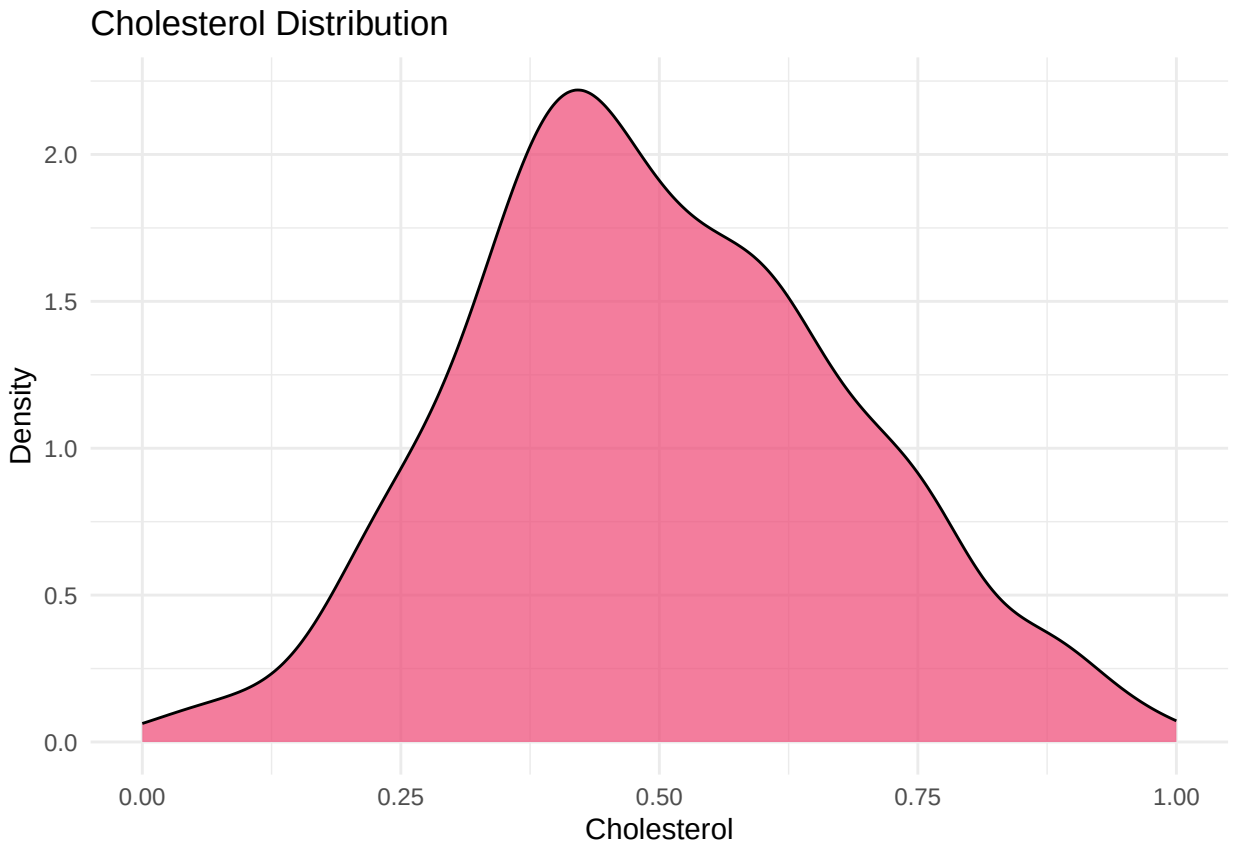
```r
clean_heart_normal <- predict(preprocess_model, clean_heart)
head(clean_heart_normal)
```

```
##         Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG
## 1 0.2448980   M           ATA 0.6153846   0.6911197         0     Normal
## 2 0.4285714   F           NAP 0.8717949   0.2702703         0     Normal
## 3 0.1836735   M           ATA 0.4871795   0.6679537         0         ST
## 4 0.4081633   F           ASY 0.5897436   0.4015444         0     Normal
## 5 0.5306122   M           NAP 0.7435897   0.3281853         0     Normal
## 6 0.2244898   M           NAP 0.3589744   0.8841699         0     Normal
##       MaxHR ExerciseAngina    Oldpeak ST_Slope HeartDisease
## 1 0.7709924              N 0.02702703       Up            0
## 2 0.6488550              N 0.29729730     Flat            1
## 3 0.2061069              N 0.02702703       Up            0
## 4 0.2824427              Y 0.43243243     Flat            1
## 5 0.3893130              N 0.02702703       Up            0
## 6 0.7557252              N 0.02702703       Up            0
```

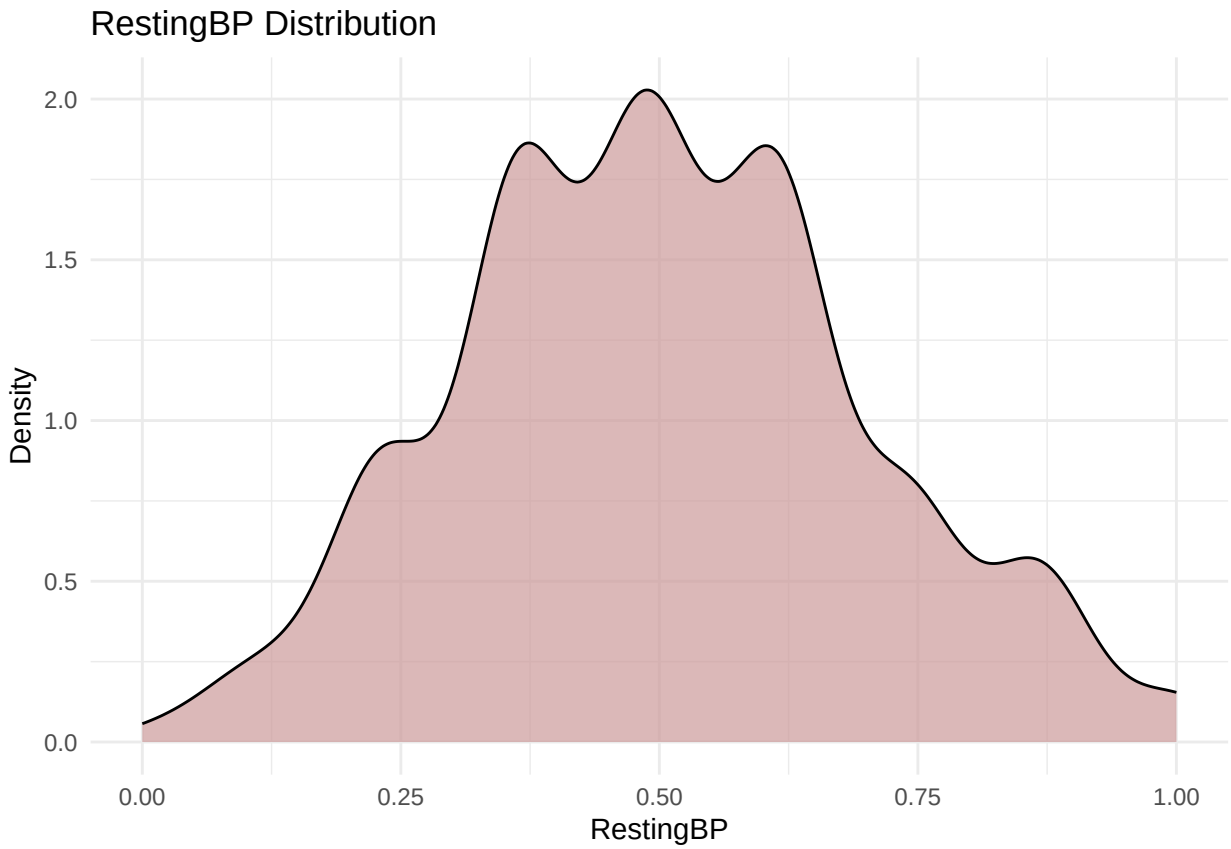
```r
ggplot(clean_heart_normal, aes(x=Oldpeak)) + geom_density(fill="#DD1189", alpha=0.7) + labs(title="Oldp
```




```r
ggplot(clean_heart_normal, aes(x=Cholesterol)) + geom_density(fill="#EE4573", alpha=0.7) + labs(title="
```




```
ggplot(clean_heart_normal, aes(x=RestingBP)) + geom_density(fill="#CC9999", alpha=0.7) + labs(title="Res
```



Making dummy variables:

```
heart11<-clean_heart_normal%>%
  select(-HeartDisease)
dummy11<- dummyVars(~., data=heart11)
clean_heart_data<- as.data.frame(predict(dummy11, newdata= heart11))
clean_heart_data$HeartDisease<-clean_heart_normal$HeartDisease
head(clean_heart_data)
```

```
##         Age Sex.F Sex.M ChestPainType.ASY ChestPainType.ATA ChestPainType.NAP
## 1 0.2448980     0     1                 0                 1                 0
## 2 0.4285714     1     0                 0                 0                 1
## 3 0.1836735     0     1                 0                 1                 0
## 4 0.4081633     1     0                 1                 0                 0
## 5 0.5306122     0     1                 0                 0                 1
## 6 0.2244898     0     1                 0                 0                 1
##   ChestPainType.TA RestingBP Cholesterol FastingBS.0 FastingBS.1 RestingECG.LVH
## 1                0 0.6153846   0.6911197           1           0              0
## 2                0 0.8717949   0.2702703           1           0              0
## 3                0 0.4871795   0.6679537           1           0              0
## 4                0 0.5897436   0.4015444           1           0              0
## 5                0 0.7435897   0.3281853           1           0              0
## 6                0 0.3589744   0.8841699           1           0              0
##   RestingECG.Normal RestingECG.ST     MaxHR ExerciseAngina.N ExerciseAngina.Y
## 1                 1             0 0.7709924                1                0
## 2                 1             0 0.6488550                1                0
## 3                 0             1 0.2061069                1                0
## 4                 1             0 0.2824427                0                1
```

```
## 5                 1             0 0.3893130                1                0
## 6                 1             0 0.7557252                1                0
##      Oldpeak ST_Slope.Down ST_Slope.Flat ST_Slope.Up HeartDisease
## 1 0.02702703             0             0           1            0
## 2 0.29729730             0             1           0            1
## 3 0.02702703             0             0           1            0
## 4 0.43243243             0             1           0            1
## 5 0.02702703             0             0           1            0
## 6 0.02702703             0             0           1            0
```

```
nzv <- nearZeroVar(clean_heart_data)
length(nzv)
```

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

```
summary(clean_heart_data)
```

```
##       Age              Sex.F            Sex.M        ChestPainType.ASY
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3673   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :0.5306   Median :0.0000   Median :1.0000   Median :0.0000
##  Mean   :0.5037   Mean   :0.2384   Mean   :0.7616   Mean   :0.4798
##  3rd Qu.:0.6327   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  ChestPainType.ATA ChestPainType.NAP ChestPainType.TA    RestingBP
##  Min.   :0.0000    Min.   :0.0000    Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000    1st Qu.:0.0000    1st Qu.:0.00000   1st Qu.:0.3590
##  Median :0.0000    Median :0.0000    Median :0.00000   Median :0.4872
##  Mean   :0.2283    Mean   :0.2355    Mean   :0.05636   Mean   :0.5054
##  3rd Qu.:0.0000    3rd Qu.:0.0000    3rd Qu.:0.00000   3rd Qu.:0.6154
##  Max.   :1.0000    Max.   :1.0000    Max.   :1.00000   Max.   :1.0000
##   Cholesterol      FastingBS.0      FastingBS.1     RestingECG.LVH
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3707   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.4807   Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   :0.4973   Mean   :0.8382   Mean   :0.1618   Mean   :0.2341
##  3rd Qu.:0.6216   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  RestingECG.Normal RestingECG.ST        MaxHR        ExerciseAngina.N
##  Min.   :0.0000    Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000    1st Qu.:0.0000   1st Qu.:0.3893   1st Qu.:0.0000
##  Median :1.0000    Median :0.0000   Median :0.5344   Median :1.0000
##  Mean   :0.6026    Mean   :0.1633   Mean   :0.5316   Mean   :0.6272
##  3rd Qu.:1.0000    3rd Qu.:0.0000   3rd Qu.:0.6794   3rd Qu.:1.0000
##  Max.   :1.0000    Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  ExerciseAngina.Y    Oldpeak        ST_Slope.Down     ST_Slope.Flat
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.02703   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.13514   Median :0.00000   Median :0.0000
##  Mean   :0.3728   Mean   :0.25133   Mean   :0.04624   Mean   :0.4711
##  3rd Qu.:1.0000   3rd Qu.:0.43243   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##   ST_Slope.Up     HeartDisease
##  Min.   :0.0000   0:372
##  1st Qu.:0.0000   1:320
```

```
##  Median :0.0000
##  Mean   :0.4827
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

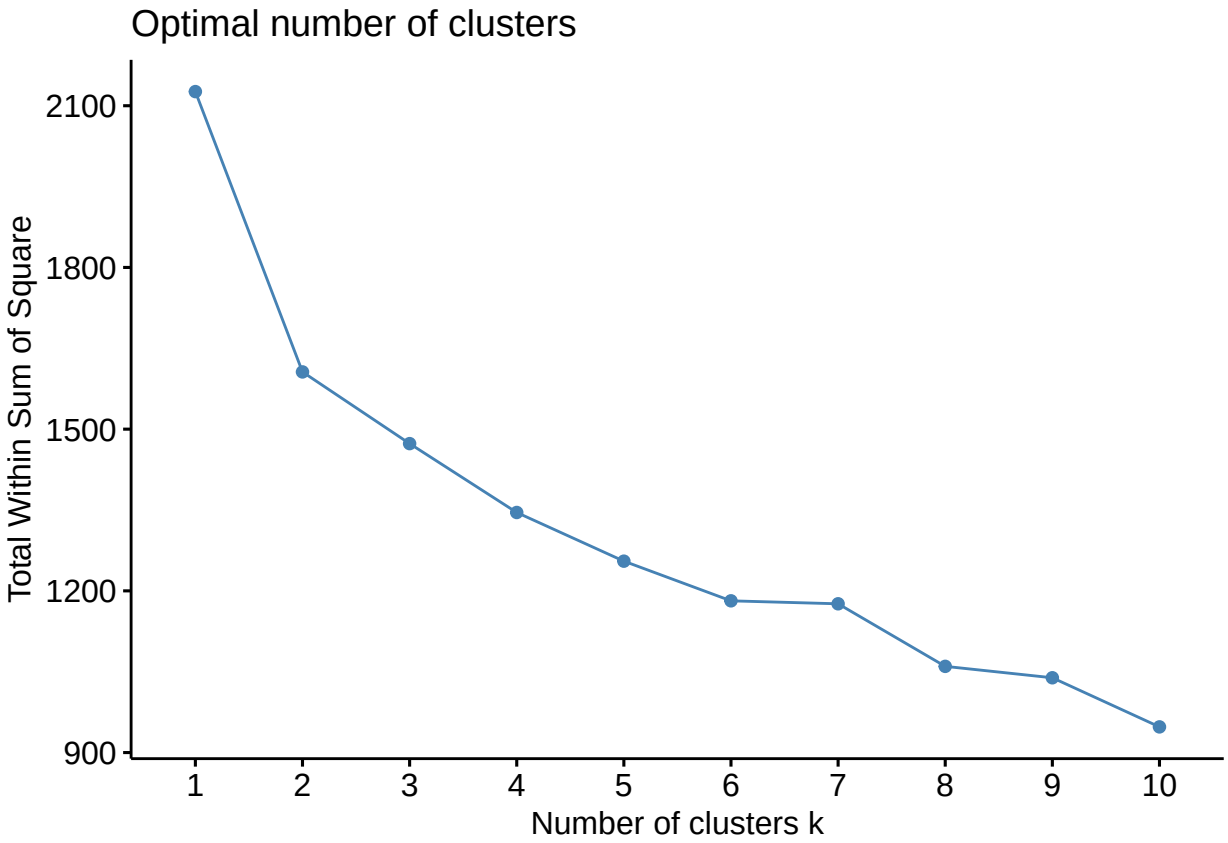
E. Clustering

As our data has both numerical as well as categorical data, and I have already converted my categorical data into dummy I used **k means** clustering.

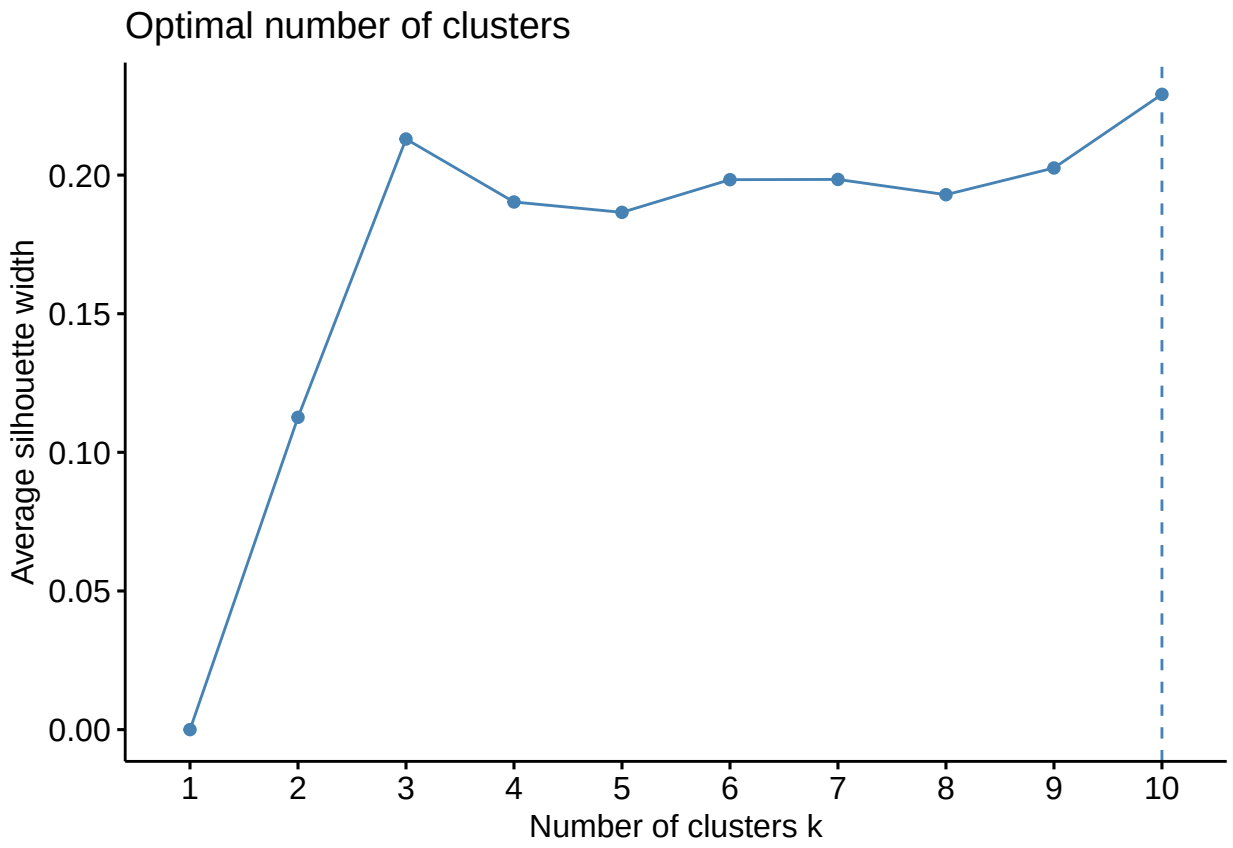
```
heart_data22<- clean_heart_data
heart_data22<-heart_data22%>%
  select(-HeartDisease)

set.seed(123)
library(stats)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(cluster)
#Finding the elbow
fviz_nbclust(heart_data22, kmeans, method="wss")
```




```
#Using Average Silhouette
fviz_nbclust(heart_data22, kmeans, method="silhouette")
```

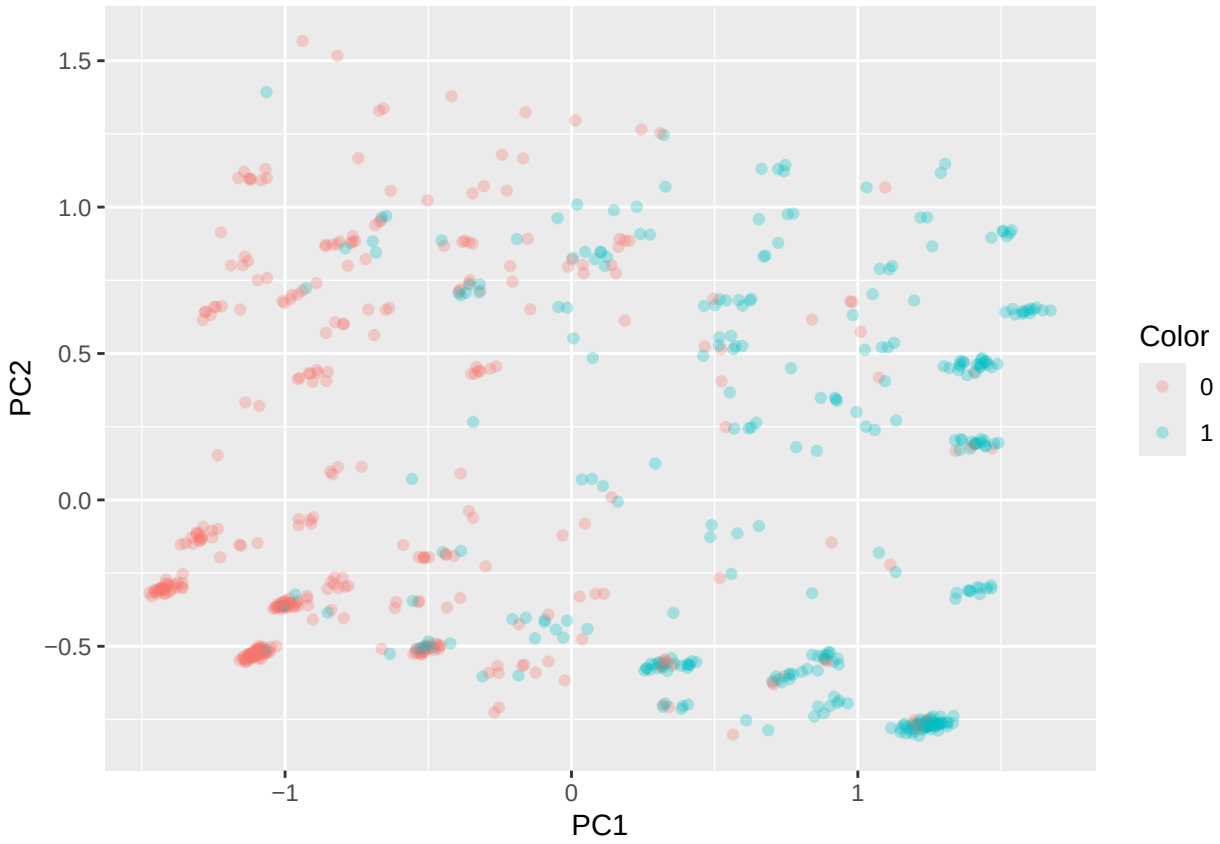


From the first plot, we can see an elbow in 2, say we can say the optimal cluster for our data is 2 in both cases, now applying k means:

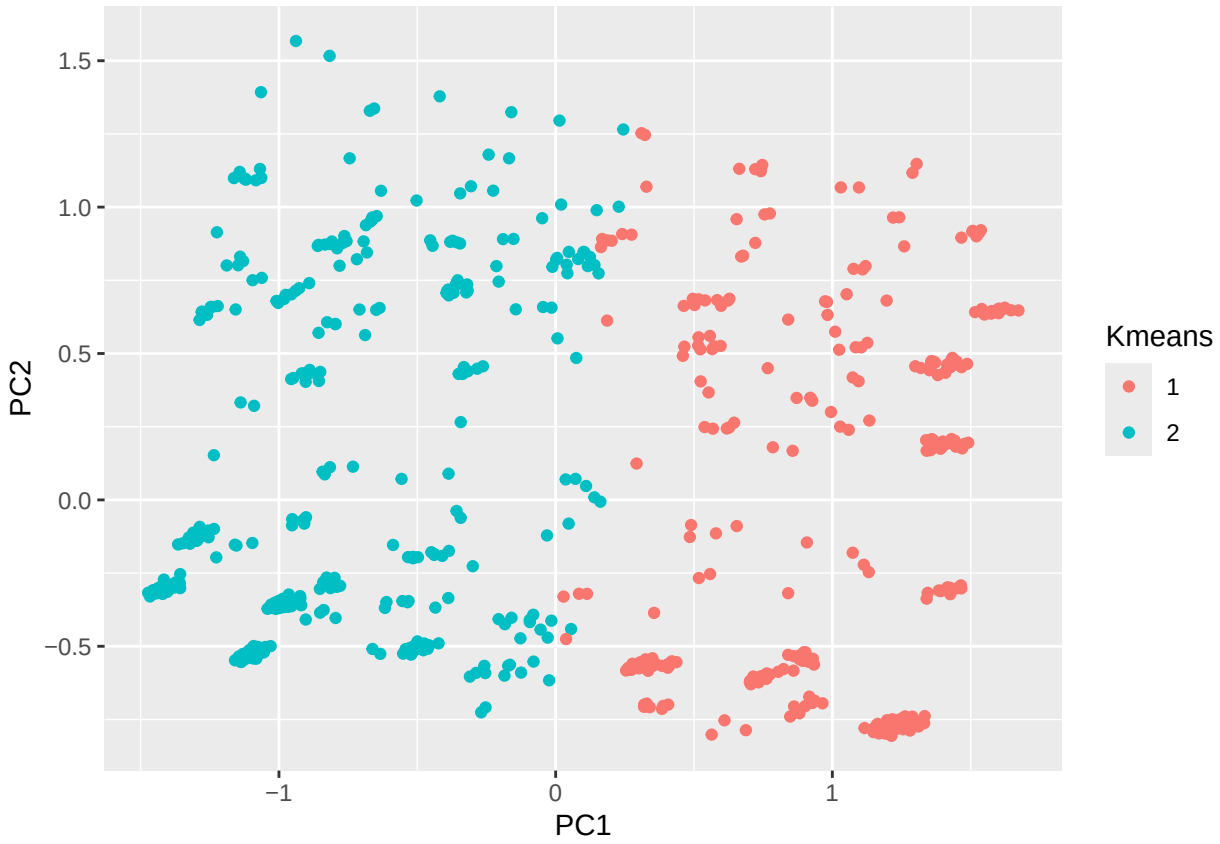
```
set.seed(123)
clean_heart_kmeans<- kmeans(heart_data22, centers= 2, nstart = 25)

#comparing with actual labels:
comparison_df<-data.frame(
  Cluster=clean_heart_kmeans$cluster,
  Label=clean_heart_data$HeartDisease
)

#creating a contingency table:
contingency_table<- table(comparison_df)
contingency_table
```

```
##         Label
## Cluster   0   1
##       1  47 259
##       2 325  61
```

```
#PCA for visualization:
pca = prcomp(heart_data22)
rotated_data = as.data.frame(pca$x)
rotated_data$Color <- clean_heart_data$HeartDisease
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha = 0.3)
```




```
# Assign clusters as a new column
rotated_data$Kmeans = as.factor(clean_heart_kmeans$cluster)
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Kmeans)) + geom_point()
```



From the contigency table, we can see that k means doing a great job separating true positives and negatives but in reverse, trying with HAC to see its clusters with our normalized data before making dummy:

```
str(clean_heart_normal)
```

```
## 'data.frame':    692 obs. of  12 variables:
##  $ Age           : num  0.245 0.429 0.184 0.408 0.531 ...
##  $ Sex           : Factor w/ 2 levels "F","M": 2 1 2 1 2 2 1 2 2 1 ...
##  $ ChestPainType : Factor w/ 4 levels "ASY","ATA","NAP",..: 2 3 2 1 3 3 2 2 1 2 ...
##  $ RestingBP     : num  0.615 0.872 0.487 0.59 0.744 ...
##  $ Cholesterol   : num  0.691 0.27 0.668 0.402 0.328 ...
##  $ FastingBS     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RestingECG    : Factor w/ 3 levels "LVH","Normal",..: 2 2 3 2 2 2 2 2 2 2 ...
##  $ MaxHR         : num  0.771 0.649 0.206 0.282 0.389 ...
##  $ ExerciseAngina: Factor w/ 2 levels "N","Y": 1 1 1 2 1 1 1 1 2 1 ...
##  $ Oldpeak       : num  0.027 0.297 0.027 0.432 0.027 ...
##  $ ST_Slope      : Factor w/ 3 levels "Down","Flat",..: 3 2 3 2 3 3 3 3 2 3 ...
##  $ HeartDisease  : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 1 2 1 ...
```

```
hac_heart<-clean_heart_normal%>%
select(-HeartDisease)
library(cluster)

#passing dataframe with metric= gower as it is categorical
dist_mat2 <-daisy(hac_heart, metric="gower")
summary(dist_mat2)
```

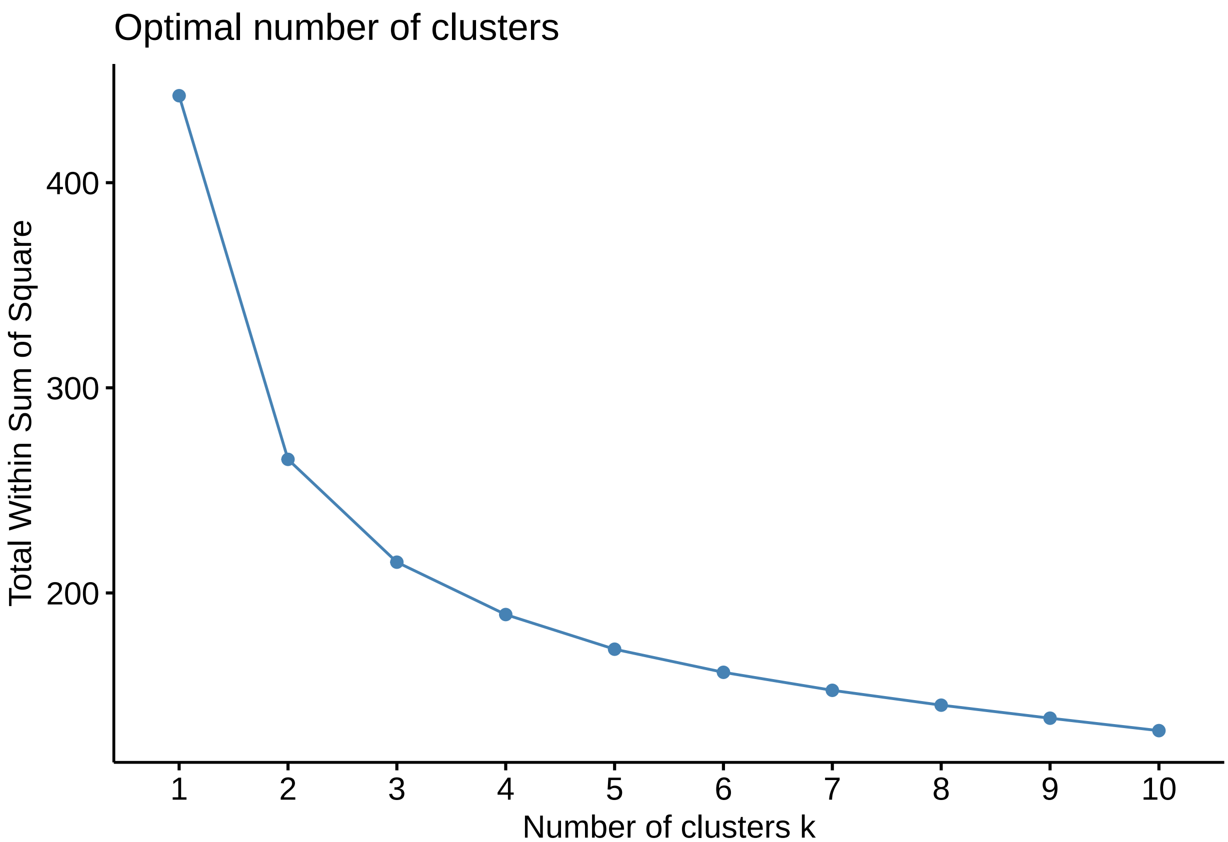
```
## 239086 dissimilarities, summarized :
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

```
## 0.004172 0.267920 0.364450 0.364320 0.459790 0.818740
## Metric :  mixed ;  Types = I, N, N, I, I, N, N, I, N, I, N
## Number of objects : 692
```

```r
hc_complete <- hclust(dist_mat2, method = "complete")
hc_average <- hclust(dist_mat2, method = "average")
hc_ward <- hclust(dist_mat2, method = "ward.D2")



fviz_nbclust(hac_heart, FUN = hcut, method = "wss")
```

```
## Warning in stats::dist(x): NAs introduced by coercion
```

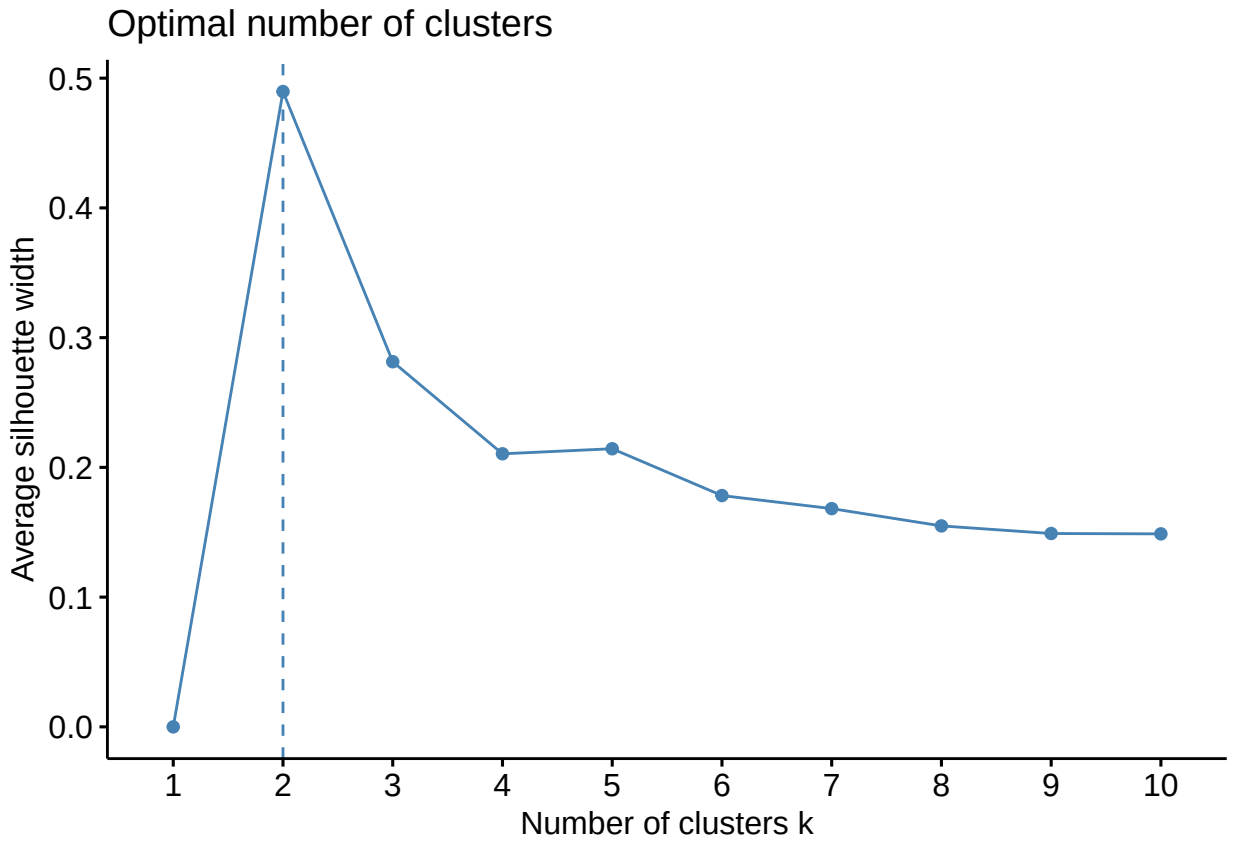
```
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
## Warning in stats::dist(x, method = method, ...): NAs introduced by coercion
```




```r
fviz_nbclust(hac_heart, FUN = hcut, method = "silhouette")
```

```
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
```

```
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
## Warning in stats::dist(x): NAs introduced by coercion
```




```
h1 <- cutree(hc_complete, k=2)
h2<- cutree(hc_average, k=2)
h3<- cutree(hc_ward, k=2)

result <- data.frame(Disease = clean_heart_normal$HeartDisease, HAC1=h1, HAC2= h2, HAC3 = h3, Kmeans = c

result %>% group_by(HAC1) %>% select(HAC1, Disease) %>% table()
```

```
##     Disease
## HAC1   0   1
##    1 285  42
##    2  87 278
```

```
result %>% group_by(HAC2) %>% select(HAC2, Disease) %>% table()
```

```
##     Disease
## HAC2   0   1
##    1 331 110
##    2  41 210
```

```
result %>% group_by(HAC3) %>% select(HAC3, Disease) %>% table()
```

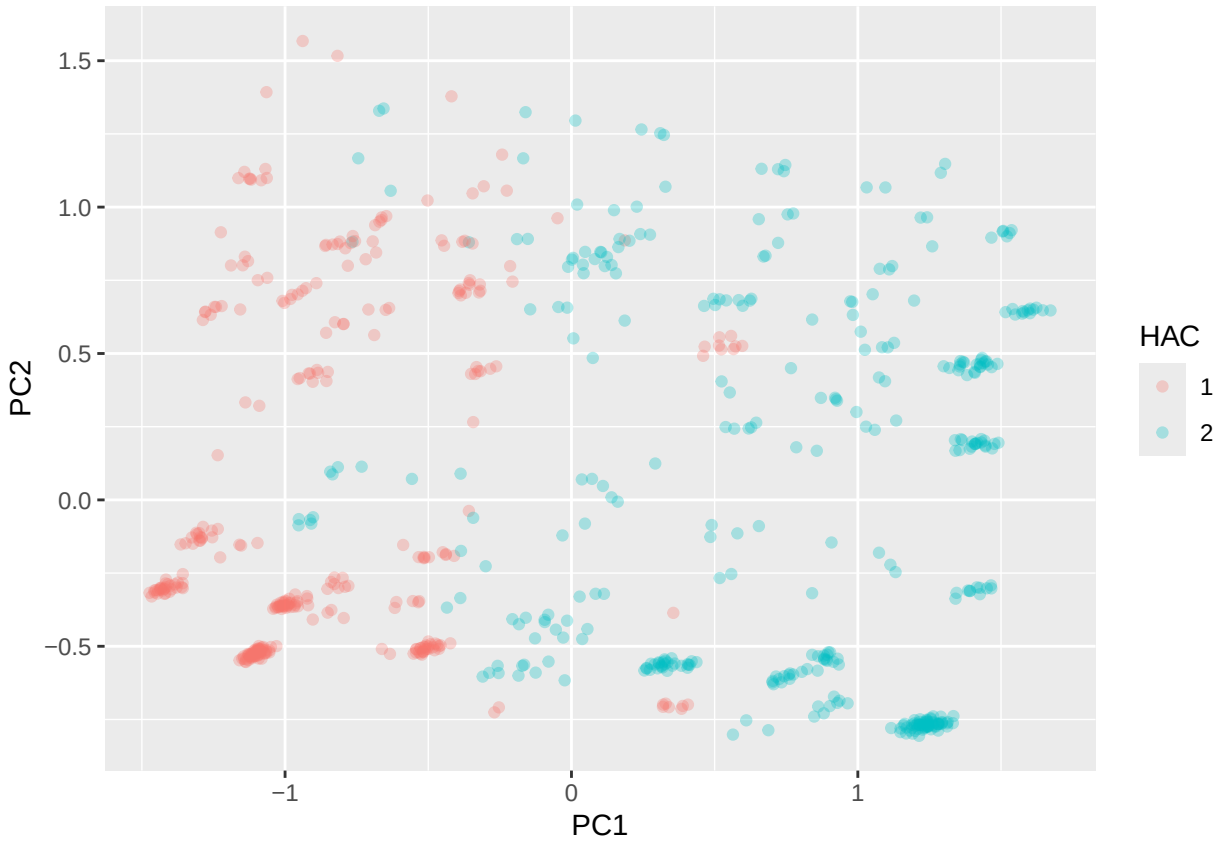
```
##     Disease
## HAC3   0   1
##    1 306  85
##    2  66 235
```

```
result %>% group_by(Kmeans) %>% select(Kmeans, Disease) %>% table()
```

```
##       Disease
## Kmeans   0   1
##      1  47 259
##      2 325  61
```

```
#pca visualization:
pca = prcomp(heart_data22)
rotated_data1 = as.data.frame(pca$x)
rotated_data1$Color <- clean_heart_normal$HeartDisease
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha = 0.3)
```


```
rotated_data$HAC = as.factor(h1)
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = HAC)) + geom_point(alpha = 0.3)
```



From the table, HAC1 gives the most accurate prediction.

F. Classification:

I chose SVM and Decision Tree:

Splitting the data into test and train:

```
library(e1071)
set.seed(123)
train_index <- createDataPartition(clean_heart_data$HeartDisease, p = 0.8, list = FALSE)
train_data <- clean_heart_data[train_index, ]
test_data <- clean_heart_data[-train_index, ]
```

```
library(rpart)
set.seed(123)

#train control for cv
train_control <- trainControl(method = "cv", number = 10)

#Tree 1
hypers = rpart.control(minsplit = 4000, maxdepth = 30, minbucket=800)
tree1<- train(HeartDisease~., data= train_data, control= hypers, trControl= train_control, method="rpar

#Training Set 1
pred_tree <- predict(tree1, train_data)
#Confusion matrix Train 1:
```

```r
cfm_train <- confusionMatrix(train_data$HeartDisease, pred_tree)

#Test Set 1
pred_tree <- predict(tree1, test_data)
#Confusion matrix Test 1:
cfm_test <- confusionMatrix(test_data$HeartDisease, pred_tree)

#Getting training accuracy:
a_train<- cfm_train$overall[1]
#Getting testing accuracy:
a_test<- cfm_test$overall[1]
#Getting number of nodes
nodes<- nrow(tree1$finalModel$frame)

#From the table
comp_tbl <-data.frame("Nodes"=nodes, "TrainAccuracy"= a_train, "TestAccuracy"= a_test, "Minsplit"=4000,


#Tree with Rpart1SE and no selection of hyperparameter:
tree<- train(HeartDisease~., data= train_data, trControl= train_control, method="rpart1SE")

#Training Set
pred_tree <- predict(tree, train_data)
#Confusion matrix Train 1:
cfm_train <- confusionMatrix(train_data$HeartDisease, pred_tree)
#Test Set
pred_tree <- predict(tree, test_data)
#Confusion matrix Test 1:
cfm_test <- confusionMatrix(test_data$HeartDisease, pred_tree)


#Getting training accuracy:
a_train<- cfm_train$overall[1]
#Getting testing accuracy:
a_test<- cfm_test$overall[1]
#Getting number of nodes
nodes<- nrow(tree$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, "Rpart1SE", "-", "-"))

#Tree 2
hypers = rpart.control(minsplit = 300, maxdepth = 4, minbucket=70)
tree2<- train(HeartDisease~., data= train_data, control= hypers, trControl= train_control, method="rpar

#Training Set 2
pred_tree <- predict(tree2, train_data)
#Confusion matrix Train 2:
cfm_train <- confusionMatrix(train_data$HeartDisease, pred_tree)

#Test Set 2
pred_tree <- predict(tree2, test_data)
#Confusion matrix Test 2:
cfm_test <- confusionMatrix(test_data$HeartDisease, pred_tree)
```

```
#Getting training accuracy:
a_train<- cfm_train$overall[1]
#Getting testing accuracy:
a_test<- cfm_test$overall[1]
#Getting number of nodes
nodes<- nrow(tree2$finalModel$frame)

#Adding rows to the table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 300, 4, 70))


#Tree 7
hypers = rpart.control(minsplit = 10000, maxdepth = 3, minbucket=2000)
tree7<- train(HeartDisease~., data= train_data, control= hypers, trControl= train_control, method="rpar

#Training Set 7
pred_tree <- predict(tree7, train_data)
#Confusion matrix Train 7:
cfm_train <- confusionMatrix(train_data$HeartDisease, pred_tree)

#Test Set 7
pred_tree <- predict(tree7, test_data)
#Confusion matrix Test 7:
cfm_test <- confusionMatrix(test_data$HeartDisease, pred_tree)

#Getting training accuracy:
a_train<- cfm_train$overall[1]
#Getting testing accuracy:
a_test<- cfm_test$overall[1]
#Getting number of nodes
nodes<- nrow(tree7$finalModel$frame)

#Adding rows to the table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 1, 4, 2))


#Tree 9
hypers = rpart.control(minsplit = 50, maxdepth = 3, minbucket=50)
tree9<- train(HeartDisease~., data= train_data, control= hypers, trControl= train_control, method="rpar

#Training Set 9
pred_tree <- predict(tree9, train_data)
#Confusion matrix Train 9:
cfm_train <- confusionMatrix(train_data$HeartDisease, pred_tree)

#Test Set 9
pred_tree <- predict(tree9, test_data)
#Confusion matrix Test 9:
cfm_test <- confusionMatrix(test_data$HeartDisease, pred_tree)

#Getting training accuracy:
a_train<- cfm_train$overall[1]
#Getting testing accuracy:
```

```
a_test<- cfm_test$overall[1]
#Getting number of nodes
nodes<- nrow(tree9$finalModel$frame)

#Adding rows to the table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 50, 3, 50))


comp_tbl
```

Using Decision tree:

```
##          Nodes TrainAccuracy TestAccuracy Minsplit MaxDepth Minbucket
## Accuracy     1     0.5379061    0.5362319     4000       30       800
## 1           15     0.8880866    0.8260870 Rpart1SE        -         -
## 11           3     0.8303249    0.7826087      300        4        70
## 12           1     0.5379061    0.5362319        1        4         2
## 13           3     0.8303249    0.7826087       50        3        50
```

I experimented with 12 different variation of Minimum split, Max Depth and minimum bucket but they gave the same nodes and accuracy so I included only few. The second one with highest node is from Rpart1SE with no altered parameters, it gives the best onw within 1 SE. Selecting the tree made by Rpart1SE as it has the best accuracy:

```
library(rattle)
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
Final_tree<- tree
#Training Set
pred_final <- predict(Final_tree, train_data)
#Confusion matrix Train:
confusionMatrix(train_data$HeartDisease, pred_final)
```

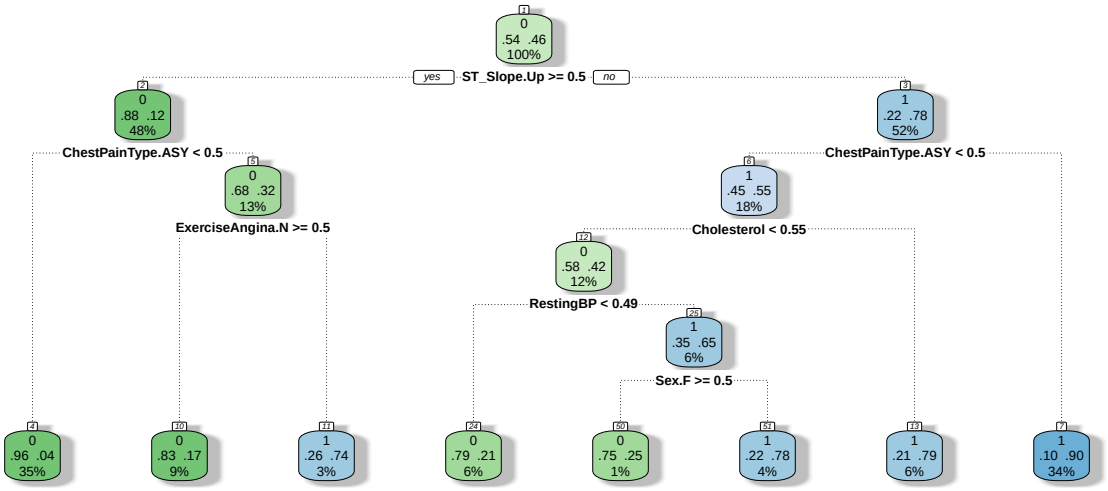
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 262  36
##          1  26 230
##
##                Accuracy : 0.8881
##                  95% CI : (0.8588, 0.9131)
##     No Information Rate : 0.5199
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7755
##
##  Mcnemar's Test P-Value : 0.253
##
##             Sensitivity : 0.9097
##             Specificity : 0.8647
##          Pos Pred Value : 0.8792
```

```
##          Neg Pred Value : 0.8984
##              Prevalence : 0.5199
##          Detection Rate : 0.4729
##    Detection Prevalence : 0.5379
##       Balanced Accuracy : 0.8872
##
##        'Positive' Class : 0
##
```

```
#Test Set
pred_tree_ <- predict(Final_tree, test_data)
#Confusion matrix
confusionMatrix(test_data$HeartDisease, pred_tree_)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 60 14
##          1 10 54
##
##                Accuracy : 0.8261
##                  95% CI : (0.7524, 0.8853)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 5.799e-15
##
##                   Kappa : 0.6518
##
##  Mcnemar's Test P-Value : 0.5403
##
##             Sensitivity : 0.8571
##             Specificity : 0.7941
##          Pos Pred Value : 0.8108
##          Neg Pred Value : 0.8437
##              Prevalence : 0.5072
##          Detection Rate : 0.4348
##    Detection Prevalence : 0.5362
##       Balanced Accuracy : 0.8256
##
##        'Positive' Class : 0
##
```

```
#Visualize our tree:
fancyRpartPlot(Final_tree$finalModel, caption = "Decision Tree")
```



Decision Tree

```
set.seed(123)
train_control= trainControl(method = "cv", number= 10)
grid<- expand.grid(C=10^seq(-5,2,0.5))
preproc= c("center","scale")
svm_grid<- train(HeartDisease~., data= train_data, method="svmLinear",              trControl = train_c
svm_grid
```

Using SVM:

```
## Support Vector Machines with Linear Kernel
##
## 554 samples
##  21 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 498, 498, 499, 498, 498, 499, ...
## Resampling results across tuning parameters:
##
##   C             Accuracy   Kappa
##   1.000000e-05  0.5379245  0.0000000
##   3.162278e-05  0.5379245  0.0000000
##   1.000000e-04  0.5379245  0.0000000
##   3.162278e-04  0.7904052  0.5677755
##   1.000000e-03  0.8716101  0.7407098
##   3.162278e-03  0.8661231  0.7305096
##   1.000000e-02  0.8643050  0.7269421
##   3.162278e-02  0.8624206  0.7230593
##   1.000000e-01  0.8587831  0.7160599
##   3.162278e-01  0.8605351  0.7196478
##   1.000000e+00  0.8605351  0.7196478
```

```
##   3.162278e+00  0.8605351  0.7196478
##   1.000000e+01  0.8605351  0.7196478
##   3.162278e+01  0.8605351  0.7196478
##   1.000000e+02  0.8605351  0.7196478
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.001.
```

```
pred_split<- predict(svm_grid, test_data)
sum(pred_split == test_data$HeartDisease) / nrow(test_data)
```

```
## [1] 0.8043478
```

```
train_control= trainControl(method = "cv", number= 10)
grid<- expand.grid(sigma = seq(0.001, 0.1), C=10^seq(-5,2,0.5))
preproc= c("center","scale")
svm_grid<- train(HeartDisease~., data= train_data, method="svmRadial",              trControl = train_c
svm_grid
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 554 samples
##  21 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 500, 499, 499, 499, 499, 498, ...
## Resampling results across tuning parameters:
##
##   C             Accuracy   Kappa
##   1.000000e-05  0.5379245  0.0000000
##   3.162278e-05  0.5379245  0.0000000
##   1.000000e-04  0.5379245  0.0000000
##   3.162278e-04  0.5379245  0.0000000
##   1.000000e-03  0.5379245  0.0000000
##   3.162278e-03  0.5379245  0.0000000
##   1.000000e-02  0.5379245  0.0000000
##   3.162278e-02  0.5379245  0.0000000
##   1.000000e-01  0.5379245  0.0000000
##   3.162278e-01  0.8714803  0.7402923
##   1.000000e+00  0.8769036  0.7521450
##   3.162278e+00  0.8551828  0.7085866
##   1.000000e+01  0.8570022  0.7120392
##   3.162278e+01  0.8625553  0.7235288
##   1.000000e+02  0.8552477  0.7088735
##
## Tuning parameter 'sigma' was held constant at a value of 0.001
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 1.
```

```
pred_split<- predict(svm_grid, test_data)
sum(pred_split == test_data$HeartDisease) / nrow(test_data)
```

```
## [1] 0.8043478
```

Comparing **Accuracy** our two classifiers:

	Train	Test
Decision Tree	0.8881	0.8261
SVM	0.8716	0.8043

G. Evaluation:

```
#Confusion matrix Train:
confusionMatrix(train_data$HeartDisease, pred_final)
```

Confusion Matrix:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 262  36
##          1  26 230
##
##                Accuracy : 0.8881
##                  95% CI : (0.8588, 0.9131)
##     No Information Rate : 0.5199
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7755
##
##  Mcnemar's Test P-Value : 0.253
##
##             Sensitivity : 0.9097
##             Specificity : 0.8647
##          Pos Pred Value : 0.8792
##          Neg Pred Value : 0.8984
##              Prevalence : 0.5199
##          Detection Rate : 0.4729
##    Detection Prevalence : 0.5379
##       Balanced Accuracy : 0.8872
##
##        'Positive' Class : 0
##
```

```
#Test Set
pred_tree_ <- predict(Final_tree, test_data)
#Confusion matrix
cm<- confusionMatrix(test_data$HeartDisease,pred_tree_)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 60 14
##          1 10 54
##
##                Accuracy : 0.8261
##                  95% CI : (0.7524, 0.8853)
```

```
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : 5.799e-15
##
##                   Kappa : 0.6518
##
##  Mcnemar's Test P-Value : 0.5403
##
##             Sensitivity : 0.8571
##             Specificity : 0.7941
##          Pos Pred Value : 0.8108
##          Neg Pred Value : 0.8437
##              Prevalence : 0.5072
##          Detection Rate : 0.4348
##    Detection Prevalence : 0.5362
##       Balanced Accuracy : 0.8256
##
##        'Positive' Class : 0
##
```

From our confusion matrix, we can see:

60 were correctly predicted with no heart disease

14 were incorrectly predicted no heart disease when actually they had heart disease

10 were incorrectly classified with having heart disease when had no heart disease

54 correctly predicted with having heart disease.

As there is no class imbalance in our label , the accuracy on our test data is good but as this is a data relating to health, better accuracy is preferred. It is cruical that our classifier should be able to classify people with heart disease even more and make less classification error in classifying wrong for patients with heart disease as not having heart disease.

Precision and Recall Manually calculating Precision and Recall:

$$Recall = TP/(TP + FN)$$

$= 60/(60 + 10)$

$= 0.857$

$$Precision = TP/(TP + FP)$$

$= 60/(60 + 10)$

$= 0.81$

Also checking with the matrix,

```
metrics <- as.data.frame(cm$byClass)
recall<- metrics["Recall",]
cat("Recall : \n", recall)
```

```
## Recall :
##  0.8571429
```

```
precision<- metrics["Precision",]
cat("Precision : \n", precision)
```

```
## Precision :
##  0.8108108
```

Precision of 0.8108 says that it is correctly able to predict about 81% of the positive label as positive.

Recall of 0.8571 means model is correctly able to identify 85% of actual positive case.

```
library(pROC)
```

ROC Curves:

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
pred_prob2<- predict(Final_tree, test_data, type="prob")
head(pred_prob2)
```

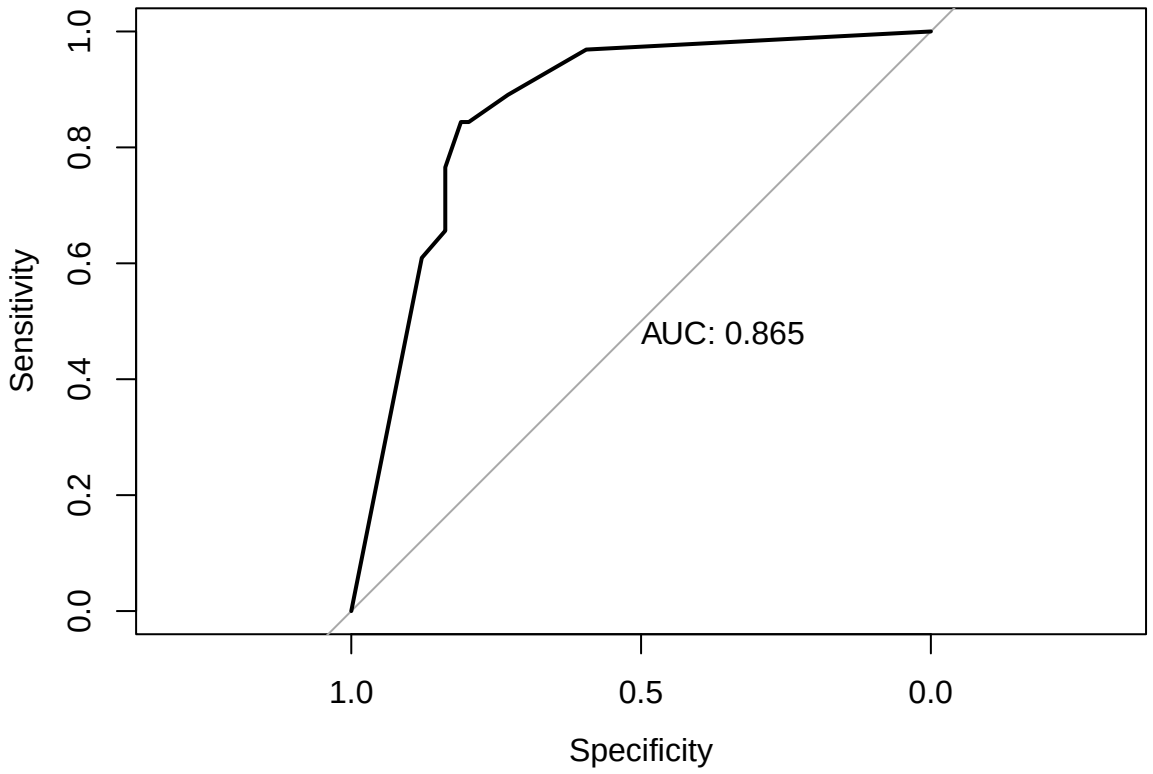
```
##             0          1
## 5  0.95897436 0.04102564
## 8  0.95897436 0.04102564
## 9  0.09947644 0.90052356
## 11 0.95897436 0.04102564
## 23 0.95897436 0.04102564
## 26 0.95897436 0.04102564
```

```
#ROC curve:
roc_obj<- roc((test_data$HeartDisease), pred_prob2[,1])
```

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

```
## Setting direction: controls > cases
```

```
plot(roc_obj, print.auc=TRUE)
```



The Area under curve is 0.865, which is a good indicator, The curve is also well above the diagonal line which indication it is able to separate the class well.

H. Report:

This data will help predict patient that are likely to have heart diseases but as this classification is critical and wrong classification of patient with heart disease as not having heart disease can impact analysis so much, the accuracy of 82% might not be enough and more classifiers can be tested to increase its prediction. I had an exciting learrning experience throughout this project deciding which steps to take along the way. I found how removing outliers could impact the study of data so much.

I. Reflection:

I enrolled in this course because I was fascinated by the power of simple data to unreveal pattern and impact lives. I was excited about being able to learn to have that power to create things that could be crucial to help grow in various field. My interest in data science grew along with the course as I learnt the entire framework of data mining and machine learning. The process of how we need to first study and understand our data to make relevant decisions and analyse and prepare them before making any big decisions on it to studying the hidden patterns in our data by clustering and eventually creating models that can classify and predict.